Oracle9*i*

PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス

リリース 2（9.2）

2002 年 7 月

部品番号：J06284-01

ORACLE®

Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス , リリース 2（9.2）

部品番号：J06284-01

原本名：Oracle9*i* Supplied PL/SQL Packages and Types Reference, Release 2 (9.2)

原本部品番号：A96613-01（Vol.1）、A96614-01（Vol.2）、A96615-01（Vol.3）

原本著者：D.K. Bradshaw

原本協力者：Ted Burroughs, Shelley Higgins, Paul Lane, Roza Leyderman, Kevin Macdowell, Jack Melnick, Chuck Murray, Kathy Rich, Vivian Schupmann, Randy Urbano, D. Alpern, G. Arora, L. Barton, N. Bhatt, S. Chandrasekar, T. Chang, G. Claborn, R. Decker, A. Downing, J. Draaijer, S. Ehrsam, A. Ganesh, R. Govindarajan, B. Goyal, C. Iyer, H. Jakobsson, A. Kalra, B. Lee, J. Liu, P. Locke, A. Logan, V. Maganty, N. Mallavarupu, J. Mallory, R. Mani, S. Mavris, A. Mozes, J. Muller, K. Muthukkaruppan, R. Pang, D. Raphaely, S. Ray, A. Rhee, J. Sharma, R. Sujithan, A. Swaminathan, K. Tarkhanov, A. Tsukerman A. To, S. Urman, S. Vivian, D. Voss, W. Wang, D. Wong, Valarie Moore

目次

19 DBMS_IOT

20 DBMS_JOB

21 DBMS_LDAP

22 DBMS_LIBCACHE

23 DBMS_LOB

24 DBMS_LOCK

25 DBMS_LOGMNR

43 DBMS_OUTPUT

44 DBMS_PCLXUTIL

45 DBMS_PIPE

46 DBMS_PROFILER

47 DBMS_PROPAGATION_ADM

48 DBMS_RANDOM

49 DBMS_RECTIFIER_DIFF

67 DBMS_SPACE

68 DBMS_SPACE_ADMIN

69 DBMS_SQL

70 DBMS_STATS

71 DBMS_STORAGE_MAP

81 DBMS_XDB

82 DBMS_XDBT

83 DBMS_XDB_VERSION

84 DBMS_XMLDOM

85 DBMS_XMLGEN

86 DBMS_XMLPARSER

87 DBMS_XMLQUERY

95 UTL_FILE

96 UTL_HTTP

97 UTL_INADDR

98 UTL_RAW

99 UTL_REF

100 UTL_SMTP

101 UTL_TCP

102 UTL_URL

103 ANYDATA タイプ

104 ANYDATASET タイプ

105 ANYTYPE タイプ

106 アドバンスト・キューイング・タイプ

107 JMS タイプ

108 論理変更レコードのタイプ

109 ルール・タイプ

索引

はじめに

このマニュアルでは、Oracle データベース・サーバーに付属の Oracle PL/SQL パッケージについて説明します。このマニュアルの情報は、特に指定されていないかぎりは、すべてのプラットフォームで実行される Oracle データベース・サーバーの各バージョンに対して適用されます。

この章では、次の項目について説明します。

対象読者

このマニュアルは、プログラマ、システム・アナリスト、プロジェクト・マネージャおよびデータベース・アプリケーションの開発に関心のある方々に有効です。このマニュアルでは、読者にアプリケーション・プログラミングについての作業知識があること、リレーショナル・データベース・システムの情報にアクセスするための SQL の使用経験が十分にあることを前提としています。一部の項では、オブジェクト指向プログラミングの基本的な知識も必要です。

このマニュアルの構成

このマニュアルの構成については、1-7 ページの表 1-1「オラクル社が提供する PL/SQL パッケージの概要」を参照してください。

関連文書

詳細は、次の Oracle マニュアルを参照してください。

- 『Oracle9*i* アプリケーション開発者ガイド - 基礎編』
- 『PL/SQL ユーザーズ・ガイドおよびリファレンス』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

マニュアル・セットに含まれるマニュアルの多くでは、Oracle のインストール時にデフォルトでインストールされるシード・データベースのサンプル・スキーマを使用しています。これらのスキーマがどのように作成されているか、およびその使用方法の詳細は、『Oracle9*i* サンプル・スキーマ』を参照してください。

リリース・ノート、インストレーション・マニュアル、ホワイト・ペーパーまたはその他の関連文書は、OTN-J（Oracle Technology Network Japan）に接続すれば、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

```
http://otn.oracle.co.jp/membership/
```

OTN-J のユーザー名とパスワードを取得済みであれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

```
http://otn.oracle.co.jp/document/
```

表記規則

このマニュアル・セットの本文とコード例に使用されている表記規則について説明します。

本文の表記規則

本文中には、特別な用語が一目でわかるように様々な表記規則が使用されています。次の表は、本文の表記規則と使用例を示しています。

規則	意味	例
太字	太字は、本文中に定義されている用語または用語集に含まれている用語、あるいはその両方を示します。	この句を指定する場合は、**索引構成表**を作成します。
固定幅フォントの大文字	固定幅フォントの大文字は、システムにより指定される要素を示します。この要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージとメソッドの他、システム指定の列名、データベース・オブジェクトと構造体、ユーザー名、およびロールがあります。	この句は、NUMBER 列に対してのみ指定できます。 BACKUP コマンドを使用すると、データベースのバックアップを作成できます。 USER_TABLES データ・ディクショナリ・ビューの TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイルとサンプルのユーザー指定要素を示します。この要素には、コンピュータ名とデータベース名、ネット・サービス名、接続識別子の他、ユーザー指定のデータベース・オブジェクトと構造体、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニット、およびパラメータ値があります。	sqlplus と入力して SQL*Plus をオープンします。 パスワードは orapwd ファイルに指定されています。 データファイルと制御ファイルのバックアップを /disk1/oracle/dbs ディレクトリに作成します。 department_id、department_name および location_id の各列は、hr.departments 表にあります。 これらのメソッドは JRepUtil クラスに実装されます。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus またはその他のコマンドラインを示します。次のように、固定幅フォントで、通常の本文とは区別して記載されています。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表は、コード例の記載上の表記規則と使用例を示しています。

規則	意味	例
[ ]	大カッコで囲まれている項目は、1 つ以上のオプション項目を示します。大カッコ自体は入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコで囲まれている項目は、そのうちの 1 つのみが必要であることを示します。中カッコ自体は入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の複数の選択肢を区切るために使用します。オプションのうち 1 つを入力します。縦線自体は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略符号は、次のどちらかを示します。	
	■ 例に直接関係のないコード部分が省略されていること。	CREATE TABLE ... AS *subquery*;
	■ コードの一部が繰返し可能であること。	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略符号は、例に直接関係のない数行のコードが省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
その他の表記	大カッコ、中カッコ、縦線および省略記号以外の記号は、表示されているとおりに入力してください。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック	イタリックの文字は、特定の値を指定する必要のあるプレースホルダまたは変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

規則	意味	例

大文字	大文字は、システムにより指定される要素を示します。これらの用語は、ユーザー定義の用語と区別するために大文字で記載されています。大カッコで囲まれている場合を除き、記載されているとおりの順序とスペルで入力してください。ただし、この種の用語は大 / 小文字区別がないため、小文字でも入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザー指定のプログラム要素を示します。たとえば、表名、列名またはファイル名を示します。
	注意：一部のプログラム要素には、大文字と小文字の両方が使用されます。この場合は、記載されているとおりに入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

PL/SQL パッケージ・プロシージャおよびタイプの新機能

次の項では、Oracle PL/SQL パッケージ・プロシージャおよびタイプの新機能について説明します。

- PL/SQL パッケージ・プロシージャおよびタイプにおける Oracle9i リリース 2（9.2）の新機能
- PL/SQL パッケージ・プロシージャおよびタイプにおける Oracle9i リリース 1（9.0.1）の新機能
- PL/SQL パッケージ・プロシージャにおける Oracle8i リリース 8.1.6 の新機能
- PL/SQL パッケージ・プロシージャにおける Oracle8i リリース 8.1.5 の新機能

PL/SQL パッケージ・プロシージャおよびタイプにおける Oracle9*i* リリース 2（9.2）の新機能

このリリースに含まれる新しい章は、次のとおりです。

- アドバンスト・キューイング・タイプ
- DBMS_APPLY_ADM
- DBMS_CAPTURE_ADM
- DBMS_LOGSTDBY
- DBMS_MGWADM
- DBMS_MGWMSG
- DBMS_PROPAGATION_ADM
- DBMS_RULE
- DBMS_RULE_ADM
- DBMS_STORAGE_MAP
- DBMS_STREAMS
- DBMS_STREAMS_ADM
- DBMS_XDB
- DBMS_XDBT
- DBMS_XDB_VERSION
- DBMS_XMLDOM
- DBMS_XMLPARSER
- DBMS_XPLAN
- DBMS_XSLPROCESSOR
- JMS タイプ
- 論理変更レコードのタイプ
- ルール・タイプ

このリリースで変更された章は、次のとおりです。

- DBMS_DDL
- DBMS_FLASHBACK
- DBMS_LOB
- DBMS_LOGMNR
- DBMS_LOGMNR_CDC_PUBLISH
- DBMS_LOGMNR_CDC_SUBSCRIBE
- DBMS_LOGMNR_D
- DBMS_METADATA
- DBMS_REDEFINITION
- DBMS_RLS
- DBMS_SPACE_ADMIN
- DBMS_STATS
- DBMS_TRANSFORM
- DBMS_WM
- DBMS_XMLGEN
- DBMS_XMLQUERY
- DBMS_XMLSAVE
- DBMS_XMLSchema
- UTL_FILE
- UTL_HTTP

PL/SQL パッケージ・プロシージャおよびタイプにおける Oracle9*i* リリース 1（9.0.1）の新機能

このリリースに含まれる新しいパッケージは、次のとおりです。

- DBMS_AQELM
- DBMS_ENCODE
- DBMS_FGA
- DBMS_FLASHBACK
- DBMS_LDAP
- DBMS_LibCache
- DBMS_LOGMNR_CDC_PUBLISH
- DBMS_LOGMNR_CDC_SUBSCRIBE
- DBMS_METADATA
- DBMS_ODCI
- DBMS_OUTLN_EDIT
- DBMS_REDEFINITION
- DBMS_TRANSFORM
- DBMS_URL
- DBMS_WM
- DBMS_XMLGEN
- DBMS_XMLQuery
- DMBS_XMLSave
- UTL_ENCODE

このリリースには、次のタイプに関する新しい情報が含まれています。

- DBMS_TYPES
- ANYDATA_TYPE
- ANYDATASET_TYPE
- ANYTYPE_TYPE

このリリースには、次のパッケージの拡張機能が含まれています。

- UTL_FILE
- UTL_HTTP
- UTL_RAW

PL/SQL パッケージ・プロシージャにおける Oracle8*i* リリース 8.1.6 の新機能

このリリースに含まれる新しいパッケージは、次のとおりです。

- DBMS_BACKUP_RESTORE
- DBMS_OBFUSCATION_TOOLKIT
- UTL_INADDR
- UTL_SMTP
- UTL_TCP

このリリースには、次のパッケージの拡張機能が含まれています。

- DBMS_DEBUG
- DBMS_DISTRIBUTED_TRUST_ADMIN
- DBMS_LOGMINER
- DBMS_LOGMINER_D
- DBMS_PCLXUTIL
- DMBS_PROFILER
- DBMS_REPAIR
- DBMS_RESOURCE_MANAGER
- DBMS_ROWID
- DBMS_SQL
- DBMS_UTILITY
- UTL_HTTP

PL/SQL パッケージ・プロシージャにおける Oracle8*i* リリース 8.1.5 の新機能

このマニュアルは、当初リリース 8.1.5 を対象に作成されました。

1

概要

オラクル社では、Oracle サーバーでデータベース機能を拡張する数多くの PL/SQL パッケージを提供し、SQL 機能にアクセス可能な PL/SQL の機能を用意しています。このような提供パッケージは、アプリケーション構築または独自のストアド・プロシージャ作成の検討に使用できます。

注意： このマニュアルは、Oracle データベース・サーバーに付属するパッケージについて記載されています。Oracle Developer または Oracle Application Server など、その他の製品に付属するパッケージは対象外となります。

この章では、次の項目について説明します。

- パッケージの概要
- オラクル社が提供する PL/SQL パッケージの概要
- サブリメンタル・パッケージ内のサブプログラムの概要

関連項目： ユーザー独自のパッケージ作成方法の詳細は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』を参照してください。

パッケージの概要

パッケージとは、関連するプログラム・オブジェクトをカプセル化した集合体で、データベースにまとめて格納されています。プログラム・オブジェクトには、プロシージャ、ファンクション、変数、定数、カーソルおよび例外があります。

パッケージには、スタンドアロンのプロシージャやファンクションに比べて多くの利点があります。主な利点は次のとおりです。

- アプリケーション開発をより効率的に計画できます。
- 権限をより効率的に付与できます。
- 依存スキーマ・オブジェクトを再コンパイルせずに、パッケージ・オブジェクトを変更できます。
- Oracle で、複数のパッケージ・オブジェクトをメモリーに一度に読み込めます。
- プロシージャまたはファンクションをオーバーロードできます。オーバーロードとは、同一パッケージ内に同じ名前でプロシージャを複数作成し、各プロシージャでは異なる数またはデータ・タイプの引数を使用することです。
- パッケージ内のすべてのプロシージャとファンクションで使用できるグローバルな変数とカーソルを含めることができます。

パッケージのコンポーネント

PL/SQL パッケージは、パッケージ仕様部およびパッケージ本体の 2 つで構成されます。ただし、本体が不要な場合もあります。仕様部はアプリケーションへのインタフェースで、使用可能なタイプ、変数、定数、例外、カーソルおよびサブプログラムを宣言します。本体はカーソルとサブプログラムを完全に定義し、仕様部を完全に実装します。

パッケージは、サブプログラムとは異なり、コール、パラメータ化またはネスト化することはできません。ただし、パッケージおよびサブプログラムのフォーマットは類似しています。

```
CREATE PACKAGE name AS  -- specification (visible part)
   -- public type and item declarations
   -- subprogram specifications
END [name];

CREATE PACKAGE BODY name AS  -- body (hidden part)
   -- private type and item declarations
   -- subprogram bodies
[BEGIN
   -- initialization statements]
END [name];
```

仕様部には、アプリケーションで参照できるパブリック宣言が含まれています。本体には、実装の詳細およびプライベート宣言が含まれていますが、アプリケーションからは参照できません。仕様部を変更せずに、パッケージ本体をデバッグ、拡張または置換できます。パッケージ本体の実装の詳細はアプリケーションから隠されているため、コール元プログラムを再コンパイルせずに、パッケージ本体を変更できます。

オラクル社が提供するパッケージ

オラクル社が提供するパッケージの大部分は、データベースが作成され、CATPROC.SQL スクリプトが実行されるときに自動的にインストールされます。たとえば、DBMS_ALERT パッケージを作成するには、ユーザー SYS で接続したときに DBMSALRT.SQL および PRVTALRT.PLB スクリプトを実行する必要があります。ただし、これらのスクリプトは、CATPROC.SQL スクリプトによって自動的に実行されます。

一部のパッケージは、自動的にはインストールされません。これらのパッケージをインストールするには、該当する各章の指示に従ってください。

SQL から PL/SQL ファンクションをコールするには、コールするユーザーがそのファンクションの所有者であるか、またはユーザーにそのファンクションの EXECUTE 権限が付与されている必要があります。PL/SQL ファンクションで定義されたビューを検索するには、そのビューの SELECT 権限が必要です。ビューを検索するには個々の EXECUTE 権限は必要ありません。パッケージに関する特別要件については、それぞれの章の記述を参照してください。

新規パッケージの作成

パッケージを作成し、Oracle データベースに格納するには、CREATE PACKAGE および CREATE PACKAGE BODY 文を使用します。これらの文は、SQL*Plus または Oracle Enterprise Manager から対話形式で実行できます。

新規パッケージを作成するには、次の手順を実行します。

1. CREATE PACKAGE 文でパッケージ仕様部を作成します。

   パッケージ仕様部にはプログラム・オブジェクトを宣言できます。ここで宣言したオブジェクトは、パブリック・オブジェクトと呼ばれます。パブリック・オブジェクトは、パッケージ内の別のオブジェクトからも、パッケージの外部からも参照できます。

   注意： CREATE PACKAGE 文に OR REPLACE 句を追加するとさらに便利になる場合があります。

2. CREATE PACKAGE BODY 文でパッケージ本体を作成します。

   パッケージ本体にはプログラム・オブジェクトを宣言および定義できます。

   - パッケージ仕様部で宣言したパブリック・オブジェクトをパッケージ本体で定義する必要があります。
   - プライベート・オブジェクトと呼ばれる追加のパッケージ・オブジェクトを宣言および定義できます。プライベート・オブジェクトは、パッケージ仕様部ではなくパッケージ本体で宣言されるため、パッケージ内の他のオブジェクトからのみ参照できます。パッケージの外部からは参照できません。

     関連項目：

     - 『PL/SQL ユーザーズ・ガイドおよびリファレンス』
     - 新規パッケージ作成の詳細は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』を参照してください。
     - パッケージの格納および実行の詳細は、『Oracle9*i* データベース概要』を参照してください。

仕様部と本体の分離

パッケージの仕様部では、パブリックなタイプ、変数、定数およびサブプログラムを宣言します。これらは、そのパッケージの有効範囲外から参照できます。パッケージの本体では、仕様部で宣言されたオブジェクトおよびパッケージ外部のアプリケーションからは参照できないプライベート・オブジェクトを定義します。

パッケージの仕様部と本体は、データベースに別々に格納されます。パブリック・プログラム・オブジェクトをコールまたは参照するその他のスキーマ・オブジェクトはパッケージ仕様部にのみ依存し、パッケージ本体には依存しません。仕様部と本体が分かれているため、パッケージ本体にあるプログラム・オブジェクトの定義を変更しても、そのプログラム・オブジェクトをコールまたは参照するその他のスキーマ・オブジェクトを無効にすることはありません。パッケージ仕様部にあるプログラム・オブジェクトの宣言を変更した場合のみ、それに依存しているスキーマ・オブジェクトが無効になります。

例 次の例は、EMPLOYEE_MANAGEMENT という名前のパッケージのパッケージ仕様部です。このパッケージには、ストアド・ファンクションが 1 つ、ストアド・プロシージャが 2 つ含まれています。

```
CREATE PACKAGE employee_management AS
   FUNCTION hire_emp (name VARCHAR2, job VARCHAR2,
      mgr NUMBER, hiredate DATE, sal NUMBER, comm NUMBER,
      deptno NUMBER) RETURN NUMBER;
   PROCEDURE fire_emp (emp_id NUMBER);
   PROCEDURE sal_raise (emp_id NUMBER, sal_incr NUMBER);
END employee_management;
```

このパッケージの本体では、ファンクションおよびプロシージャが定義されます。

```
CREATE PACKAGE BODY employee_management AS
   FUNCTION hire_emp (name VARCHAR2, job VARCHAR2,
      mgr NUMBER, hiredate DATE, sal NUMBER, comm NUMBER,
      deptno NUMBER) RETURN NUMBER IS
```

ファンクションでは、従業員番号を除き、フィールドに対するすべての引数を従業員表に挿入します。フィールドに対する値は順番に設定されます。このファンクションへのコールによって生成された順序番号が戻されます。

```
       new_empno    NUMBER(10);

   BEGIN
      SELECT emp_sequence.NEXTVAL INTO new_empno FROM dual;
      INSERT INTO emp VALUES (new_empno, name, job, mgr,
         hiredate, sal, comm, deptno);
      RETURN (new_empno);
   END hire_emp;

   PROCEDURE fire_emp(emp_id IN NUMBER) AS
```

プロシージャは、引数 emp_id に対応する従業員番号を持つ従業員を削除します。該当する従業員が見つからない場合は、例外が呼び出されます。

```
   BEGIN
      DELETE FROM emp WHERE empno = emp_id;
      IF SQL%NOTFOUND THEN
      raise_application_error(-20011, 'Invalid Employee
         Number: ' || TO_CHAR(emp_id));
   END IF;
END fire_emp;

PROCEDURE sal_raise (emp_id IN NUMBER, sal_incr IN NUMBER) AS
```

プロシージャは 2 つの引数を受け入れます。emp_id は従業員番号に対応する番号です。sal_incr は、その従業員の給料増額分です。

```
   BEGIN

   -- If employee exists, then update salary with increase.

      UPDATE emp
         SET sal = sal + sal_incr
         WHERE empno = emp_id;
      IF SQL%NOTFOUND THEN
         raise_application_error(-20011, 'Invalid Employee
            Number: ' || TO_CHAR(emp_id));
```

```
      END IF;
   END sal_raise;
END employee_management;
```

注意： この例を実際に試す場合は、最初に順序番号 emp_sequence を作成してください。次の SQL*Plus 文で作成できます。

```
SQL> CREATE SEQUENCE emp_sequence
   > START WITH 8000 INCREMENT BY 10;
```

パッケージ内容の参照

パッケージ仕様部で宣言されたタイプ、アイテムおよびサブプログラムを参照するには、ドット表記法を使用します。たとえば、次のようにします。

```
package_name.type_name
package_name.item_name
package_name.subprogram_name
```

日時および期間データ型の略称

日時および期間データ型の多くには、長い名前が付けられています。これは、レプリケーション・マネージメント API のプロシージャおよびファンクションで使用できません。したがって、これらのデータ・タイプの完全な名前ではなく、略称を使用する必要があります。次の表に、データ・タイプおよびその略称を示します。DATE および TIMESTAMP データ・タイプには略称を使用する必要はありません。

データ・タイプ	略称
TIMESTAMP WITH TIME ZONE	TSTZ
TIMESTAMP LOCAL TIME ZONE	TSLTZ
INTERVAL YEAR TO MONTH	IYM
INTERVAL DAY TO SECOND	IDS

たとえば、DBMS_DEFER_QUERY.GET_*datatype*_ARG ファンクションを使用して、遅延コールの TIMESTAMP LOCAL TIME ZONE 引数の値を決定する場合、*datatype* に TSLTZ を代入します。つまり、DBMS_DEFER_QUERY.GET_TSLTZ_ARG ファンクションを実行します。

オラクル社が提供する PL/SQL パッケージの概要

表 1-1 に、提供する PL/SQL サーバー・パッケージを示します。これらのパッケージは、パッケージ所有者ではなく起動ユーザーとして実行されます。特に注記がないかぎり、パッケージは同じ名前のパブリック・シノニムを介してコールできます。

注意：

- これらのパッケージが提供するプロシージャとファンクションおよびその外部インタフェースはオラクル社が所有するもので、変更される可能性があります。
- オラクル社が提供するパッケージを変更すると、内部エラーおよびデータベースのセキュリティ違反が発生する可能性があります。提供するパッケージを変更しないでください。

表 1-1 オラクル社が提供する PL/SQL パッケージの概要

パッケージ名	説明	参照先
CWM2_OLAP_AW_ACCESS	分析作業領域オブジェクトのリレーショナル・ビューを作成するスクリプトを生成します。	『Oracle9*i* OLAP User's Guide』
DBMS_ALERT	データベース・イベントの非同期通知をサポートします。	第 2 章
DBMS_APPLICATION_INFO	監査またはパフォーマンス追跡の目的で、アプリケーション名をデータベースに登録できます。	第 3 章
DBMS_APPLY_ADM	適用プロセスを開始、停止および構成する管理プロシージャを提供します。	第 4 章
DBMS_AQ	（事前定義されたオブジェクト・タイプの）メッセージをキューに追加したり、デキューできます。	第 5 章
DBMS_AQADM	キューまたはキュー表にある管理ファンクションを、事前定義されたオブジェクト・タイプのメッセージに対して実行できます。	第 6 章
DBMS_AQELM	電子メールおよび HTTP によりアドバンスト・キューイングの非同期通知の構成管理プロシージャを提供します。	第 7 章
DBMS_AW	分析作業領域オブジェクトに対して OLAP DML 文を発行します。また、このパッケージ内のプロシージャとファンクション、および OLAP_TABLE ファンクションの実行によって作成されたセッション・ログを取り出し、印刷できます。	『Oracle9*i* OLAP User's Guide』

表 1-1 オラクル社が提供する PL/SQL パッケージの概要（続き）

パッケージ名	説明	参照先
DBMS_CAPTURE_ADM	取得プロセスを開始、停止および構成する管理プロシージャを記述します。Streams で使用されます。	第 8 章
DBMS_DDL	ストアド・プロシージャから一部の SQL DDL 文へのアクセスを提供し、DDL では使用できない特殊な管理操作を提供します。	第 9 章
DBMS_DEBUG	サーバー側のデバッガを実装し、サーバー側の PL/SQL プログラム・ユニットをデバッグする方法を提供します。	第 10 章
DBMS_DEFER	レプリケート・トランザクション遅延リモート・プロシージャ・コール（RPC）機能へのユーザー・インタフェースを提供します。分散オプションが必要です。	第 11 章
DBMS_DEFER_QUERY	ビューでは参照できない遅延リモート・プロシージャ・コール（RPC）のキュー・データの問合せを許可します。分散オプションが必要です。	第 12 章
DMBS_DEFER_SYS	レプリケート・トランザクション遅延リモート・プロシージャ・コール（RPC）機能へのシステム管理者用インタフェースを提供します。分散オプションが必要です。	第 13 章
DBMS_DESCRIBE	ストアド・プロシージャの引数をフルネーム変換およびセキュリティ・チェックとともに記述します。	第 14 章
DBMS_DISTRIBUTED_TRUST_ADMIN	Trusted Database リストをメンテナンスします。このリストは、特定のサーバーからの権限データベース・リンクが受入れ可能かどうかを判断するために使用されます。	第 15 章
DBMS_FGA	ファイングレイン・セキュリティ機能を提供します。	第 16 章
DMBS_FLASHBACK	指定した実時間または指定したシステム変更番号（SCN）にデータベースのバージョンをフラッシュバックします。	第 17 章
DBMS_HS_PASSTHROUGH	異機種間サービスを使用して、パススルー SQL 文をオラクル以外のシステムに送信できます。	第 18 章
DBMS_IOT	ANALYZE コマンドを使用して、索引構成表の連鎖行への参照を設定できる表を作成します。	第 19 章
DBMS_JOB	定期的に行う管理プロシージャをスケジュールできます。ジョブ・キュー用のインタフェースでもあります。	第 20 章

表 1-1 オラクル社が提供する PL/SQL パッケージの概要（続き）

パッケージ名	説明	参照先
DBMS_LDAP	LDAP サーバーからデータにアクセスするファンクションおよびプロシージャを提供します。	第 21 章
DBMS_LIBCACHE	リモート・インスタンスから SQL および PL/SQL を抽出し、この SQL を実行せずにローカルでコンパイルして、Oracle インスタンスにライブラリ・キャッシュを用意します。	第 22 章
DBMS_LOB	Oracle ラージ・オブジェクト（LOB）データ・タイプー BLOB、CLOB（読取り / 書込み）および BFILE（読取り専用）における操作に対する汎用ルーチンを提供します。	第 23 章
DBMS_LOCK	Oracle Lock Management サービスを使用して、ロックの要求、変換および解放を実行できます。	第 24 章
DBMS_LOGMNR	ログ・リーダーを初期化および実行するためのファンクションを提供します。	第 25 章
DBMS_LOGMNR_CDC_PUBLISH	リレーショナル表に追加、変更または削除された新規データを識別し、アプリケーションによる使用が可能な形式で変更後のデータを公開します。	第 26 章
DBMS_LOGMNR_CDC_SUBSCRIBE	DBMS_LOGMNR_CDC_PUBLISH パッケージで獲得および公開された変更データの表示および問合せを行います。	第 27 章
DBMS_LOGMNR_D	現行のデータベースのディクショナリ表を問い合せ、その内容を含んだテキスト・ベースのファイルを作成します。	第 28 章
DBMS_LOGSTDBY	ロジカル・スタンバイ・データベース環境を構成および管理するためのプロシージャを記述します。	第 29 章
DBMS_METADATA	コール元で完全なデータベース・オブジェクト定義（メタデータ）をディレクトリから容易に取り出します。	第 30 章
DBMS_MGWADM	メッセージ・ゲートウェイ管理インタフェースを記述します。アドバンスト・キューイングで使用されます。	第 31 章
DBMS_MGWMSG	オブジェクト・タイプ（メッセージ本体を変換するために標準的なメッセージ・タイプで使用）、およびメッセージ・ゲートウェイ・メッセージ・タイプを処理するためのヘルパー・メソッド、定数およびサブプログラムを記述します。アドバンスト・キューイングで使用されます。	第 32 章

表 1-1 オラクル社が提供する PL/SQL パッケージの概要（続き）

パッケージ名	説明	参照先
DBMS_MVIEW	同じリフレッシュ・グループおよびパージ・ログの一部ではない複数のスナップショットをまとめてリフレッシュします。DBMS_SNAPSHOT のシノニムです。	第 33 章
DBMS_OBFUSCATION_TOOLKIT	データ暗号化規格に対するプロシージャを提供します。	第 34 章
DBMS_ODCI	ファンクションの経過時間に基づき、ユーザー・ファンクションの CPU コストを戻します。	第 35 章
DBMS_OFFLINE_OG	マスター・グループのオフライン・インスタンス化用のパブリック API を提供します。	第 36 章
DBMS_OFFLINE_SNAPSHOT	スナップショットのオフライン・インスタンス化用のパブリック API を提供します。	第 37 章
DBMS_OLAP	サマリー、ディメンションおよびクエリー・リライト用のプロシージャを提供します。	第 38 章
DBMS_ORACLE_TRACE_AGENT	Oracle7 Server 内の Oracle TRACE Instrumentation にクライアントがコールできるインタフェースを提供します。	第 39 章
DBMS_ORACLE_TRACE_USER	Oracle7 Server の Oracle TRACE Instrumentation へのパブリック・アクセスをコール・ユーザーに提供します。	第 40 章
DBMS_OUTLN	ストアド・アウトラインの管理に関連したプロシージャおよびファンクションに対するインタフェースを提供します。OUTLN_PKG のシノニムです。	第 41 章
DBMS_OUTLN_EDIT	実行者の権限パッケージを編集できるようにします。	第 42 章
DBMS_OUTPUT	情報をバッファに蓄積して後で取り出せるようにします。	第 43 章
DBMS_PCLXUTIL	パーティション単位でローカル索引を作成するためのパーティション内並列性を提供します。	第 44 章
DBMS_PIPE	セッション間のメッセージ送信を可能にする DBMS パイプ・サービスを提供します。	第 45 章
DBMS_PROFILER	既存の PL/SQL アプリケーションをプロファイルし、パフォーマンス上のボトルネックを識別するためのプローブ・プロファイラ API を提供します。	第 46 章
DBMS_PROPAGATION_ADM	ソース・キューから宛先キューへの伝播を構成するための管理プロシージャを提供します。	第 47 章

表 1-1 オラクル社が提供する PL/SQL パッケージの概要（続き）

パッケージ名	説明	参照先
DBMS_RANDOM	組込み式の乱数ジェネレータを提供します。	第 48 章
DBMS_RECTIFIER_DIFF	2 つのレプリケート・サイト間のデータの不整合を検出および解決するために使用する API を提供します。	第 49 章
DBMS_REDEFINITION	表のオンライン再編成を実行します。	第 50 章
DBMS_REFRESH	トランザクション的に一貫性のある時点にまとめてリフレッシュできるスナップショットのグループを作成できます。分散オプションが必要です。	第 51 章
DBMS_REPAIR	データの破損修復プロシージャを提供します。	第 52 章
DBMS_REPCAT	レプリケーション・カタログと環境を管理および更新するためのルーチンを提供します。レプリケーション・オプションが必要です。	第 53 章
DBMS_REPCAT_ADMIN	対称型レプリケーション機能に必要な権限を持つユーザーを作成します。レプリケーション・オプションが必要です。	第 54 章
DBMS_REPCAT_INSTATIATE	配置テンプレートをインスタンス化します。レプリケーション・オプションが必要です。	第 55 章
DBMS_REPCAT_RGT	リフレッシュ・グループ・テンプレートのメンテナンスと定義を制御します。レプリケーション・オプションが必要です。	第 56 章
DBMS_REPUTIL	表複製用のシャドウ表、トリガーおよびパッケージを生成するルーチンを提供します。	第 57 章
DBMS_RESOURCE_MANAGER	プラン、コンシューマ・グループおよびプラン・ディレクティブをメンテナンスします。また、変更内容をプラン・スキーマにグループ化する方法も提供します。	第 58 章
DBMS_RESOURCE_MANAGER_PRIVS	リソース・コンシューマ・グループに関連付けられた権限を管理します。	第 59 章
DBMS_RESUMABLE	領域を使い果たしたり、長時間経過した後に領域制限に到達する大規模な操作を一時停止し、問題を修正した後で文を再開させます。	第 60 章
DBMS_RLS	行レベルのセキュリティ管理インタフェースを提供します。	第 61 章
DBMS_ROWID	ROWID を作成し、その内容を解釈するプロシージャを提供します。	第 62 章

表 1-1 オラクル社が提供する PL/SQL パッケージの概要（続き）

パッケージ名	説明	参照先
DBMS_RULE	EVALUATE プロシージャを記述します。Streams で使用されます。	第 63 章
DBMS_RULE_ADM	ルール、ルール・セットおよびルール評価コンテキストを作成および管理するための管理インタフェースを記述します。Streams で使用されます。	第 64 章
DBMS_SESSION	SQL ALTER SESSION 文とその他のセッション情報に、ストアド・プロシージャからアクセスできるようにします。	第 65 章
DBMS_SHARED_POOL	標準の LRU メカニズムによって期限切れで削除されないようにオブジェクトを共有メモリーに保存します。	第 66 章
DBMS_SPACE	標準 SQL を介しては使用できないセグメント領域情報を提供します。	第 67 章
DBMS_SPACE_ADMIN	標準 SQL を介しては使用できない表領域とセグメント領域の管理を提供します。	第 68 章
DBMS_SQL	動的 SQL を使用してデータベースへのアクセスを可能にします。	第 69 章
DBMS_STATS	データベース・オブジェクト用に収集したオプティマイザの統計情報を、ユーザーが表示および変更するためのメカニズムを提供します。	第 70 章
DBMS_STORAGE_MAP	FMON と通信して、マッピング操作を起動します。	第 71 章
DBMS_STRM	SYS.AnyData オブジェクトを LCR オブジェクトに変換するインタフェース、およびセッションによって生成された REDO エントリにバイナリ・タグで注釈を付けるインタフェースを記述します。	第 72 章
DBMS_STRM_A	表、スキーマおよびデータベース・レベルでの取得、伝播および適用に使用する、変換不要の単純なルールを追加および削除するための管理プロシージャを記述します。	第 73 章
DBMS_TRACE	PL/SQL トレースを起動および停止するルーチンを提供します。	第 74 章
DBMS_TRANSACTION	ストアド・プロシージャから SQL トランザクション文へのアクセスを提供し、トランザクション・アクティビティを監視します。	第 75 章
DBMS_TRANSFORM	Oracle Advanced Queuing のメッセージ形式変換機能へのインタフェースを提供します。	第 76 章

表 1-1 オラクル社が提供する PL/SQL パッケージの概要（続き）

パッケージ名	説明	参照先
DBMS_TTS	トランスポート可能なセットが自己完結タイプかどうかをチェックします。	第 77 章
DBMS_TYPES	組込みおよびユーザー定義のタイプを表現する定数で構成されます。	第 78 章
DBMS_UTILITY	様々なユーティリティ・ルーチンを提供します。	第 79 章
DBMS_WM	Oracle Database Workspace Manager へのプログラミング・インタフェースを使用して、長いトランザクションを処理する方法を記述します。	第 80 章
DBMS_XDB	PL/SQL 用のリソース管理およびアクセス制御 API を記述します。	第 81 章
DBMS_XDBT	管理者が XML DB 階層上に ConText 索引を作成し、それを自動メンテナンス用に構成する方法を記述します。	第 82 章
DBMS_XDB_VERSION	バージョニング API を記述します。	第 83 章
DBMS_XMLDOM	XMLType オブジェクトへのアクセスについて記述します。	第 84 章
DBMS_XMLGEN	SQL 問合せの結果を標準的な XML 形式に変換します。	第 85 章
DBMS_XMLPARSER	XML 文書のコンテンツおよび構造体へのアクセスについて記述します。	第 86 章
DMBS_XMLQUERY	データベースから XMLType への変換機能を提供します。	第 87 章
DBMS_XMLSAVE	XML からデータベース・タイプへの変換機能を提供します。	第 88 章
DBMS_XMLSCHEMA	XML Schema を登録および削除するプロシージャを記述します。	第 89 章
DBMS_XPLAN	EXPLAIN PLAN コマンドの出力のフォーマット方法を記述します。	第 90 章
DBMS_XSLPROCESSOR	XML 文書のコンテンツおよび構造体へのアクセスについて記述します。	第 91 章
DEBUG_EXTPROC	実行プロセスに連結するデバッガを使用して、プラットフォーム上で外部プロシージャをデバッグできるようにします。	第 92 章

表 1-1 オラクル社が提供する PL/SQL パッケージの概要（続き）

パッケージ名	説明	参照先
SDO_CS （注意 1 を参照）	座標システム変換のファンクションを提供します。	『Oracle Spatial ユーザーズ・ガイドおよびリファレンス』
SDO_GEOM （注意 1 を参照）	空間オブジェクトで形状操作を実装するファンクションを提供します。	『Oracle Spatial ユーザーズ・ガイドおよびリファレンス』
SDO_LRS （注意 1 を参照）	線形参照システムのサポート・ファンクションを提供します。	『Oracle Spatial ユーザーズ・ガイドおよびリファレンス』
SDO_MIGRATE （注意 1 を参照）	以前のリリースから空間データを移行するファンクションを提供します。	『Oracle Spatial ユーザーズ・ガイドおよびリファレンス』
SDO_TUNE （注意 1 を参照）	Oracle Spatial で使用される空間索引スキームの動作を決定するパラメータを選択するためのファンクションを提供します。	『Oracle Spatial ユーザーズ・ガイドおよびリファレンス』
SDO_UTIL （注意 1 を参照）	Oracle Spatial 用のユーティリティ・ファンクションおよびプロシージャを提供します。	『Oracle Spatial ユーザーズ・ガイドおよびリファレンス』
UTL_COLL	PL/SQL プログラムで、コレクション・ロケータを使用した問合せおよび更新を可能にします。	第 93 章
UTL_ENCODE	ホスト間でのデータ転送が可能になるように、RAW データを標準形式にエンコードするファンクションを提供します。	第 94 章
UTL_FILE	PL/SQL プログラムで、オペレーティング・システムのテキスト・ファイルの読込みおよび書込みを可能にし、標準オペレーティング・システムのストリーム・ファイル I/O の制限付きバージョンを提供します。	第 95 章
UTL_HTTP	PL/SQL と SQL から HTTP コールアウトを使用可能にして、インターネット上のデータへのアクセスまたは Oracle Web Server カートリッジのコールを可能にします。	第 96 章
UTL_INADDR	インターネット・アドレッシングをサポートするためのプロシージャを提供します。	第 97 章
UTL_PG	COBOL 数値データと Oracle の数値タイプとの間の変換を行うファンクションを提供します。	『Oracle Procedural Gateway for APPC User's Guide』

表 1-1 オラクル社が提供する PL/SQL パッケージの概要（続き）

パッケージ名	説明	参照先
UTL_RAW	複数の RAW の間で concat や substr を行う RAW データ・タイプに関する SQL ファンクションを提供します。	第 98 章
UTL_REF	オブジェクトへの参照を提供することによって、PL/SQL プログラムがオブジェクトにアクセスできるようにします。	第 99 章
UTL_SMTP	電子メールを送信するための PL/SQL 機能を提供します。	第 100 章
UTL_TCP	サーバーと外部との TCP/IP ベースの通信をサポートする PL/SQL 機能を提供します。	第 101 章
UTL_URL	URL 文字のエスケープおよびエスケープ解除のメカニズムを提供します。	第 102 章
ANYDATA TYPE	タイプおよび説明のインスタンスを含む、自己記述型のデータ・インスタンス・タイプ。	第 103 章
ANYDATASET TYPE	特定のタイプの説明およびそのタイプの一連のデータ・インスタンスを含みます。	第 104 章
ANYTYPE TYPE	名前の有無にかかわらず、オブジェクト・タイプおよびコレクション・タイプなどの永続的な SQL タイプのタイプ説明を含みます。また、一時的なタイプ説明を新しく構築する場合にも使用できます。	第 105 章
JMS TYPES	PL/SQL アプリケーションで JMS タイプの JMS キューを使用できるように JMS タイプを記述します。	第 107 章
ADVANCED QUEUING TYPES	アドバンスト・キューイングで使用されるタイプを記述します。	第 106 章
LOGICAL CHANGE RECORD TYPES	データベースの変更に関する情報が含まれているメッセージ・ペイロードである LCR タイプを記述します。Streams で使用されます。	第 108 章
RULES TYPES	ルール、ルール・セットおよび評価コンテキストで使用されるタイプを記述します。	第 109 章

注意 1

Spatial パッケージは、パブリック・シノニム付きでユーザー MDSYS にインストールされます。

サプリメンタル・パッケージ内のサブプログラムの概要

この項に記載されているパッケージは、他の Oracle マニュアルでも説明されています。各パッケージのマニュアル参照は、表 1-1 を確認してください。これらのパッケージに付属するサブプログラムは、表 1-2 から表 1-8 を参照してください。

SDO_CS パッケージ

表 1-2 SDO_CS パッケージのサブプログラム

サブプログラム	説明
SDO_CS.TRANSFORM	SRID または名前で指定する座標系システムを使用して、ジオメトリの表現を変換します。
SDO_CS.TRANSFORM_LAYER	ジオメトリのレイヤー全体（表で指定した列の全ジオメトリ）を変換します。
VIEWPORT_TRANSFORM	最適化された矩形を空間演算子および関数で使用するための有効な測地ポリゴンに変換します。

SDO_GEOM パッケージ

表 1-3 SDO_GEOM パッケージのサブプログラム

サブプログラム	説明
RELATE	2 つのオブジェクトの相互作用を決定します。
SDO_ARC_DENSIFY	円弧を直線で構成される近似値に変換し、円を近似する一連の直線で構成されるポリゴンに変換します。
SDO_AREA	二次元ポリゴンの領域を計算します。
SDO_BUFFER	ジオメトリ周辺にバッファ・ポリゴンを生成します。
SDO_CENTROID	ポリゴンの重心を戻します。
SDO_CONVEXHULL	凸タイプのジオメトリ・オブジェクトを表現するポリゴン・タイプのオブジェクトを戻します。
SDO_DIFFERENCE	2 つのジオメトリ・オブジェクトの位相的な差異（MINUS 演算）であるジオメトリ・オブジェクトを戻します。
SDO_DISTANCE	2 つのジオメトリ・オブジェクト間の距離を計算します。
SDO_INTERSECTION	2 つのジオメトリ・オブジェクトにおける位相的な共通部分（AND 演算）であるジオメトリ・オブジェクトを戻します。
SDO_LENGTH	ジオメトリの長さまたは外周を計算します。

表 1-3 SDO_GEOM パッケージのサブプログラム（続き）

サブプログラム	説明
SDO_MAX_MBR_ORDINATE	ジオメトリ・オブジェクトの境界を示す最小矩形における指定した縦座標の最大値を戻します。
SDO_MBR	ジオメトリの境界を示す最小矩形を戻します。
SDO_MIN_MBR_ORDINATE	ジオメトリ・オブジェクトの境界を示す最小矩形における指定した縦座標の最小値を戻します。
SDO_POINTONSURFACE	ポリゴンの表面上にあることが保証できる点を戻します。
SDO_UNION	2 つのジオメトリ・オブジェクトにおける位相的 UNION（OR 演算）であるジオメトリ・オブジェクトを戻します。
SDO_XOR	2 つのジオメトリ・オブジェクトにおける位相的な対称型差異（XOR 演算）であるジオメトリ・オブジェクトを戻します。
VALIDATE_GEOMETRY	ジオメトリが有効であるかどうか判別します。
VALIDATE_GEOMETRY_WITH_CONTEXT	有効なジオメトリ・タイプかどうかを調べる一貫性チェックを実行し、ジオメトリが無効な場合はコンテキスト情報を戻します。このファンクションは、表のジオメトリの表現を要素定義と照合してチェックします。
VALIDATE_LAYER	列に格納されたジオメトリがすべて有効であるかどうか判別します。
VALIDATE_LAYER_WITH_CONTEXT	ジオメトリ列を検証して、格納済みのジオメトリがジオメトリ・オブジェクトに対する定義ルールに従っているかどうかを判別し、無効なジオメトリに関するコンテキスト情報を戻します。
WITHIN_DISTANCE	2 つのジオメトリが相互に指定したユークリッド距離内にあるかどうか判別します。

SDO_LRS パッケージ

表 1-4 SDO_LRS パッケージのサブプログラム

サブプログラム	説明
CLIP_GEOM_SEGMENT	ジオメトリ・セグメント（DYNAMIC_SEGMENT のシノニム）をクリップします。
CONCATENATE_GEOM_SEGMENTS	2 つのジオメトリ・セグメントを 1 つのセグメントに連結します。
CONNECTED_GEOM_SEGMENTS	ジオメトリ・セグメントが接続しているかどうかチェックします。

表 1-4 SDO_LRS パッケージのサブプログラム（続き）

サブプログラム	説明
CONVERT_TO_LRS_DIM_ARRAY	メジャー・ディメンションを作成することにより、標準のディメンション配列を線形参照システムのディメンション配列に変換します。
CONVERT_TO_LRS_GEOM	メジャー情報を追加することにより、標準の SDO_GEOMETRY 行の文字列を線形参照システムのジオメトリ・セグメントに変換します。
CONVERT_TO_LRS_LAYER	メジャー情報を持たない標準行文字列のジオメトリにおいて SDO_GEOMETRY タイプの列に格納されたジオメトリ・オブジェクトをすべて、メジャー情報を持つ線形参照システムのジオメトリ・セグメントに変換し、メタデータを更新します。
CONVERT_TO_STD_DIM_ARRAY	メジャー・ディメンションを削除することにより、線形参照システムのディメンション配列を標準のディメンション配列に変換します。
CONVERT_TO_STD_GEOM	メジャー情報を削除することにより、線形参照システムのジオメトリ・セグメントを標準の SDO_GEOMETRY 行の文字列に変換します。
CONVERT_TO_STD_LAYER	メジャー情報を持たない標準行文字列のジオメトリにおいて SDO_GEOMETRY タイプの列に格納されたジオメトリ・オブジェクトをすべて、メジャー情報を持つ標準の行文字列ジオメトリに変換し、メタデータを更新します。
DEFINE_GEOM_SEGMENT	ジオメトリ・セグメントを定義します。
DYNAMIC_SEGMENT	ジオメトリ・セグメント（CLIP_GEOM_SEGMENT のシノニム）をクリップします。
FIND_LRS_DIM_POS	指定された SDO_GEOMETRY 列の SDO_DIM_ARRAY 構造におけるメジャー・ディメンションの位置を戻します。
FIND_MEASURE	指定した投影点に対して、セグメント上の最も近い点のメジャーを戻します。
GEOM_SEGMENT_END_MEASURE	ジオメトリ・セグメントの終了メジャーを戻します。
GEOM_SEGMENT_END_PT	ジオメトリ・セグメントの終了点を戻します。
GEOM_SEGMENT_LENGTH	ジオメトリ・セグメントの長さを戻します。
GEOM_SEGMENT_START_MEASURE	ジオメトリ・セグメントの開始メジャーを戻します。
GEOM_SEGMENT_START_PT	ジオメトリ・セグメントの開始点を戻します。
GET_MEASURE	LRS 点のメジャーを戻します。

表 1-4 SDO_LRS パッケージのサブプログラム（続き）

サブプログラム	説明
IS_GEOM_SEGMENT_DEFINED	LRS セグメントが適切に定義されているかどうかチェックします。
IS_MEASURE_DECREASING	LRS セグメントのメジャー値が減少しているか（数値が下降しているか）どうかチェックします。
IS_MEASURE_INCREASING	LRS セグメントのメジャー値が増加しているか（数値が上昇しているか）どうかチェックします。
LOCATE_PT	ジオメトリ・セグメントの開始から指定した距離にある点を戻します。
MEASURE_RANGE	ジオメトリ・セグメントのメジャー範囲を戻します。メジャー範囲とは、開始メジャーおよび終了メジャーの差異です。
MEASURE_TO_PERCENTAGE	指定したメジャーのジオメトリ・セグメントのメジャー範囲におけるパーセント（0 ～ 100）を戻します。
OFFSET_GEOM_SEGMENT	ジオメトリ・セグメントから指定したオフセットにあるジオメトリ・セグメントを戻します。
PERCENTAGE_TO_MEASURE	ジオメトリ・セグメントのメジャー範囲における指定したパーセント（0 ～ 100）のメジャー値を戻します。
PROJECT_PT	ジオメトリ・セグメントの点における投影点を戻します。
REDEFINE_GEOM_SEGMENT	開始メジャーおよび終了点メジャーに基づき、開始点および終了点間に割り当てられたメジャーを上書きすることで、ジオメトリ・セグメントのすべての形状点のメジャーを作成します。
RESET_MEASURE	開始メジャーおよび終了点メジャーを含め、割り当てられたメジャーをすべて無効にすることで、ジオメトリ・セグメントの全メジャーを NULL に設定します。
REVERSE_GEOMETRY	元のジオメトリ・セグメントのメジャー値および方向をリバースして、新しいジオメトリ・セグメントを戻します。
REVERSE_MEASURE	元のジオメトリ・セグメントをリバースして、新しいジオメトリ・セグメントを戻します。
SCALE_GEOM_SEGMENT	ジオメトリ・セグメントをスケールします。
SET_PT_MEASURE	指定した点のメジャー値を設定します。
SPLIT_GEOM_SEGMENT	ジオメトリ・セグメントを 2 つに分割します。
TRANSLATE_MEASURE	元のジオメトリ・セグメントを変換（開始メジャーおよび終了点メジャーを指定した値でシフト）することで、新しいジオメトリ・セグメントを戻します。
VALID_GEOM_SEGMENT	ジオメトリ・セグメントが有効であるかどうか判別します。

表 1-4 SDO_LRS パッケージのサブプログラム（続き）

サブプログラム	説明
VALID_LRS_PT	LRS 点が有効であるかどうか判別します。
VALID_MEASURE	ジオメトリ・セグメントのメジャー範囲内にメジャーが入っているかどうかチェックします。
VALIDATE_LRS_GEOMETRY	LRS ジオメトリが有効であるかどうか判別します。

SDO_MIGRATE パッケージ

表 1-5 SDO_MIGRATE パッケージのサブプログラム

プロシージャ	説明
FROM_815_TO_81X	Spatial リリース 8.1.5 のデータを現行のリリースに移行します。
OGIS_METADATA_FROM	OGIS（OpenGIS）のメタデータ表を移行する場合に使用される一時表を生成します。
OGIS_METADATA_TO	OGIS のメタデータ表を移行する場合に使用される一時表を読み取ります。
TO_734	Spatial Data Option の旧リリースのデータをリリース 7.3.4 に移行します。
TO_81X	Spatial Data Option 7.3.4 または Spatial Cartridge 8.0.4 の表を Oracle Spatial に移行します。
TO_CURRENT	旧リリースの Spatial からデータを現行のリリースに移行します。

SDO_TUNE パッケージ

表 1-6 SDO_TUNE パッケージのサブプログラム

サブプログラム	説明
ANALYZE_RTREE	R ツリー索引を分析します。索引の使用について統計を生成します。索引を再作成することで問合せのパフォーマンスの向上が大幅に見込まれる場合は、再作成をお薦めします。
AVERAGE_MBR	レイヤーのジオメトリに対し、境界を示す最小矩形の平均値を計算します。
ESTIMATE_INDEX_PERFORMANCE	空間索引の選択性を見積ります。
ESTIMATE_TILING_LEVEL	固定サイズ索引の四角形作成における適切なタイル表示レベルを判別します。

表 1-6 SDO_TUNE パッケージのサブプログラム（続き）

サブプログラム	説明
ESTIMATE_TILING_TIME	レイヤーのタイル表示時間を秒で見積ります。
ESTIMATE_TOTAL_NUMTILES	レイヤーの空間四角形の合計数を見積ります。
EXTENT_OF	レイヤーにおいてデータの境界を示す最小矩形を判別します。
HISTOGRAM_ANALYSIS	空間レイヤーの統計ヒストグラムを計算します。
MIX_INFO	各ジオメトリ・タイプのパーセントなど、空間レイヤーのジオメトリ・タイプ情報を計算します。
QUALITY_DEGRADATION	R ツリー索引の品質低下または R ツリー索引の全索引表における平均品質の低下を戻します。
RTREE_QUALITY	R ツリー索引の品質スコアまたは R ツリー索引の全索引表における平均品質スコアを戻します。

SDO_UTIL パッケージ

表 1-7 SDO_UTIL パッケージのサブプログラム

サブプログラム	説明
EXTRACT	入力ジオメトリの指定要素（およびオプションでリング）を表現するジオメトリを戻します。
GETVERTICES	入力ジオメトリの頂点の座標が含まれる表を戻します。

UTL_PG パッケージ

表 1-8 UTL_PG パッケージのサブプログラム

サブプログラム	説明
MAKE_NUMBER_TO_RAW_FORMAT	宣言した精度およびスケールの Oracle の数値タイプをリモート・ホストの内部フォーマットの RAW バイト文字列に変換するために使用する number_to_raw 形式変換仕様を作成します。
MAKE_RAW_TO_NUMBER_FORMAT	RAW バイト列を、リモート・ホストの内部フォーマットから対応する精度およびスケールの Oracle の数値タイプに変換するために使用する raw_to_number 形式変換仕様を作成します。
NUMBER_TO_RAW	宣言した精度およびスケールの Oracle の数値タイプをリモート・ホストの内部フォーマットの RAW バイト列に変換します。

表 1-8 UTL_PG パッケージのサブプログラム（続き）

サブプログラム	説明
NUMBER_TO_RAW_ FORMAT	number_to_raw 変換フォーマット n2rfmt に基づいて、宣言した精度およびスケールの Oracle の数値タイプ numval をリモート・ホストの内部フォーマットの RAW バイト列に変換します。
RAW_TO_NUMBER	リモート・ホストの内部フォーマットの RAW バイト列を Oracle の数値タイプに変換します。
RAW_TO_NUMBER_ FORMAT	raw_to_number 変換フォーマット r2nfmt に基づいて、リモート・ホストの内部フォーマットの RAW バイト列 rawval を Oracle の数値タイプに変換します。
WMSG	wmsgitem で指定された警告メッセージを wmsgblk から抽出します。
WMSGCNT	wmsgblk をテストして警告の数を判断します。

2

DBMS_ALERT

DBMS_ALERT は、データベース・イベント（アラート）の非同期通知をサポートします。このパッケージおよびデータベース・トリガーを適切に使用することで、アプリケーションは、データベース内で関連する値が変更されるたびに通知を受け取ることができます。

たとえば、グラフィック・ツールで、データベース表のデータをグラフ表示する場合を想定します。グラフィック・ツールは、データを読み込んでグラフ表示した後、読み込んだデータに関連するデータベース・アラート（WAITONE）を待機させることができます。他のユーザーがデータを変更すると、ツールは自動的に起動されます。必要となるのは、データベース表にトリガーを設定することのみです。この設定によってトリガーが起動されると必ず信号（SIGNAL）が送信されます。

アラートは、トランザクション単位で処理されます。つまり、アラートを通知するトランザクションがコミットされるまで、待機中セッションはアラートを受け取りません。したがって、特定のアラートに対する同時実行の送信と待機がいくつか発生する場合があります。

待機中のアプリケーションはデータベース内でブロックされるため、別の処理は実行できません。

注意： データベース・アラートはコミットを発行するため、Oracle Forms では使用できません。Oracle Forms がアクティブの状態でストアド・プロシージャをコールする場合の制限については、Oracle Forms のドキュメントを参照してください。

この章では、次の項目について説明します。

- DBMS_ALERT のセキュリティ、定数およびエラー
- アラートの使用方法
- DBMS_ALERT サブプログラムの要約

DBMS_ALERT のセキュリティ、定数およびエラー

セキュリティ

このパッケージのセキュリティは、選択したユーザーまたはロールにこのパッケージの EXECUTE 権限を付与することで制御できます。このパッケージの先頭部分に、使用するアラート名を制限するためのカバー・パッケージを記述することもできます。この場合は、パッケージではなく、このカバー・パッケージの EXECUTE 権限を付与できます。

定数

```
maxwait constant integer :=  86400000; -- 1000 days
```

アラートの最大待機時間です（実質的には無期限）。

エラー

DBMS_ALERT では、エラー状態のときにアプリケーション・エラー -20000 が発生します。表 2-1 は、エラー・メッセージおよびそのエラーが発生する可能性のあるプロシージャの一覧です。

表 2-1 DBMS_ALERT エラー・メッセージ

エラー・メッセージ	プロシージャ
ORU-10001 lock request error, status: N	SIGNAL
ORU-10015 error: N waiting for pipe status	WAITANY
ORU-10016 error: N sending on pipe 'X'	SIGNAL
ORU-10017 error: N receiving on pipe 'X'	SIGNAL
ORU-10019 error: N on lock request	WAIT
ORU-10020 error: N on lock request	WAITANY
ORU-10021 lock request error; status: N	REGISTER
ORU-10022 lock request error, status: N	SIGNAL
ORU-10023 lock request error; status N	WAITONE
ORU-10024 there are no alerts registered	WAITANY
ORU-10025 lock request error; status N	REGISTER
ORU-10037 attempting to wait on uncommitted signal from same session	WAITONE

アラートの使用方法

アプリケーションは、複数のイベントに対して登録でき、WAITANY プロシージャを使用して、すべてのイベントの発生を待機できます。

WAITONE または WAITANY プロシージャに対して、オプションの timeout パラメータを指定することもできます。timeout を 0（ゼロ）に設定すると、保留中のアラートが存在しない場合はすぐに戻されます。

通知セッションは、待機中セッションが受け取るメッセージをオプションで渡せます。

アラートは、対応するアプリケーションの待機コールよりも頻繁に通知されます。この場合、古いアラートは廃棄されます。アプリケーションは、トランザクションのコミットごとに、最新のアラートを取得します。

アプリケーションで、トランザクション単位のアラートが必要ない場合は、DBMS_PIPE パッケージを使用してください。

関連項目： 第 45 章「DBMS_PIPE」

SIGNAL のコール後にトランザクションがロールバックされた場合、アラートは発生しません。

アラートを受け取り、データを読み込んでも、データが変更されていない場合があります。これは、先行したアラートの後にデータが変更されたが、その前に先行したアラートに関するデータが読み込まれているためです。

アラートのチェック

通常、Oracle はイベント・ドリブンであるため、ポーリング・ループは発生しません。ポーリング・ループは、次の 2 つの場合に発生する可能性があります。

- 共有モード。データベースを共有モードで実行している場合は、別のインスタンスからのアラートをチェックするためにポーリング・ループが必要です。ポーリング・ループのデフォルト値は 1 秒で、SET_DEFAULTS プロシージャで設定できます。
- WAITANY プロシージャ。WAITANY プロシージャを使用していて、通知セッションが通知の 1 秒以内にコミットしない場合は、このコミットされていないアラートが別のアラートをカムフラージュしないようにするために、ポーリング・ループが必要です。ポーリング・ループは 1 秒間隔で始まり、30 秒間隔まで段階的に変化します。

DBMS_ALERT サブプログラムの要約

表 2-2 DBMS_ALERT パッケージのサブプログラム

サブプログラム	説明
「REGISTER プロシージャ」 2-4 ページ	アラートからメッセージを受け取ります。
「REMOVE プロシージャ」 2-5 ページ	アラートからの通知を無効にします。
「REMOVEALL プロシージャ」 2-5 ページ	このセッションに対するアラートを登録リストからすべて削除します。
「SET_DEFAULTS プロシージャ」 2-6 ページ	ポーリング間隔を設定します。
「SIGNAL プロシージャ」 2-6 ページ	アラートを通知します（登録セッションにメッセージを送信します）。
「WAITANY プロシージャ」 2-7 ページ	セッションに登録したアラートからのメッセージの受取りを、timeout 秒待機します。
「WAITONE プロシージャ」 2-8 ページ	名前を設定したアラートからのメッセージの受取りを、timeout 秒待機します。

REGISTER プロシージャ

セッションは、このプロシージャによってアラートを登録します。アラートの名前が IN パラメータになります。セッションは、必要な数のアラートを登録できます。アラートとの関連が不要になった場合は、REMOVE をコールしてそのアラートの登録を削除する必要があります。

構文

```
DBMS_ALERT.REGISTER (
   name  IN  VARCHAR2);
```

パラメータ

表 2-3 REGISTER プロシージャのパラメータ

パラメータ	説明
name	このセッションに関連しているアラート名

注意： 'ORA$' で始まるアラート名は、オラクル社の製品用に予約されています。アラート名は、30 バイト以内で設定してください。大文字と小文字は区別されません。

REMOVE プロシージャ

アラートとの関連が不要になったセッションは、このプロシージャによって登録リストからそのアラートを削除できます。アラートを削除すると、アラートの通知側が行う処理量が削減されます。

アラートの削除は、アラートの通知側が行う処理量を削減するための重要な処理です。セッションがアラートを削除せずに異常終了した場合、そのアラートは（即時ではありませんが）結果的に消去されます。

構文

```
DBMS_ALERT.REMOVE (
   name  IN  VARCHAR2);
```

パラメータ

表 2-4 REMOVE プロシージャのパラメータ

パラメータ	説明
name	登録リストから削除するアラート名（大 / 小文字区別なし）

REMOVEALL プロシージャ

このプロシージャによって、このセッションに関連するアラートを登録リストからすべて削除します。セッションですべてのアラートが不要になったときは、必ずこのプロシージャを使用してください。

このプロシージャは、セッションの中でこのパッケージを初めて参照すると、自動的にコールされます。したがって、前のセッションが異常終了した場合でも、そのセッションに関連するアラートが現在のセッションに影響を与えることはありません。

このプロシージャは常にコミットを実行します。

構文

```
DBMS_ALERT.REMOVEALL;
```

SET_DEFAULTS プロシージャ

ポーリング・ループが必要な場合は、この SET_DEFAULTS プロシージャを使用してポーリング間隔を設定します。

構文

```
DBMS_ALERT.SET_DEFAULTS (
   sensitivity  IN  NUMBER);
```

パラメータ

表 2-5 SET_DEFAULTS プロシージャのパラメータ

パラメータ	説明
sensitivity	ポール間でスリープするポーリング間隔（秒）。デフォルトは 5 秒です。

SIGNAL プロシージャ

このプロシージャはアラートを通知します。SIGNAL コールは、コールしたトランザクションがコミットされたときのみ有効になります。トランザクションがロールバックされると、SIGNAL は無効になります。

このアラートを登録したすべてのセッションに通知されます。関連しているセッションが現在待機中の場合は、そのセッションが起動されます。関連しているセッションが現在待機中でない場合は、次にそのセッションが待機コールを行ったときに通知されます。

複数のセッションが、同時に同じアラートの通知を受けられます。各セッションは、アラートを通知するとき、コミットするまでその他の同時セッションをすべてブロックします。この結果、トランザクションはシリアル化されます。

構文

```
DBMS_ALERT.SIGNAL (
   name     IN  VARCHAR2,
   message  IN  VARCHAR2);
```

パラメータ

表 2-6 SIGNAL プロシージャのパラメータ

パラメータ	説明
name	通知するアラート名。
message	このアラートに関連付けるメッセージ（1800 バイト以下）。 このメッセージが待機中のセッションに渡されます。待機中のセッションは、メッセージ内の情報を使用して、アラート発生後のデータベースの読込みを中止することもできます。

WAITANY プロシージャ

現行のセッションが登録されているすべてのアラートを対象として、そのいずれかの発生を待機する場合に WAITANY をコールします。プロシージャが実行される前に、暗黙的な COMMIT が発行されます。同一のセッションで、最初にアラートを通知して、その後アラートを待機する場合もあります。この場合は、通知の後および待機の前に必ずコミットしてください。この処理を行わないと、DBMS_LOCK.REQUEST（DBMS_ALERT によってコールされます）からステータス 4 が戻されます。

構文

```
DBMS_ALERT.WAITANY (
   name      OUT  VARCHAR2,
   message   OUT  VARCHAR2,
   status    OUT  INTEGER,
   timeout   IN   NUMBER DEFAULT MAXWAIT);
```

パラメータ

表 2-7 WAITANY プロシージャのパラメータ

パラメータ	説明
name	発生したアラート名を戻します。
message	アラートに関連付けられたメッセージを戻します。 これは、SIGNAL コールが提供するメッセージです。WAITANY の前に、このアラートで複数の通知が発生した場合は、最新の SIGNAL コールに対応するメッセージが戻されます。それ以前の SIGNAL コールのメッセージは廃棄されます。
status	戻される値。 0 －アラート発生。 1 －タイムアウト発生。
timeout	アラートの最大待機時間。 timeout 秒以内にアラートが発生しない場合は、ステータス 1 が戻されます。

エラー

```
-20000, ORU-10024: there are no alerts registered.
```

原因： 待機前にアラートを登録する必要があります。

WAITONE プロシージャ

このプロシージャは、特定のアラートの発生を待機します。プロシージャが実行される前に、暗黙的な COMMIT が発行されます。同一のセッションで、最初にアラートを通知して、その後のトランザクションでアラートを待機する場合もあります。この場合、通知の後および待機の前に必ずコミットしてください。この処理を行わないと、DBMS_LOCK.REQUEST（DBMS_ALERT によってコールされます）からステータス 4 が戻されます。

構文

```
DBMS_ALERT.WAITONE (
   name      IN   VARCHAR2,
   message   OUT  VARCHAR2,
   status    OUT  INTEGER,
   timeout   IN   NUMBER DEFAULT MAXWAIT);
```

パラメータ

表 2-8 WAITONE プロシージャのパラメータ

パラメータ	説明
name	待機するアラート名。
message	アラートに関連付けられたメッセージを戻します。 これは、SIGNAL コールが提供するメッセージです。WAITONE の前に、このアラートで複数の通知が発生した場合は、最新の SIGNAL コールに対応するメッセージが戻されます。それ以前の SIGNAL コールのメッセージは廃棄されます。
status	戻される値。 0 ーアラート発生。 1 ータイムアウト発生。
timeout	アラートの最大待機時間。 名前を設定したアラートが timeout 秒以内に発生しない場合は、ステータス 1 が戻されます。

例

全従業員について、部門ごとの平均給与のグラフを作成するとします。アプリケーションは、EMP の変更を常に認識しておく必要があります。アプリケーションのコードは次のようになります。

```
DBMS_ALERT.REGISTER('emp_table_alert');
    <<readagain>>:
   /* ... read the emp table and graph it */
      DBMS_ALERT.WAITONE('emp_table_alert', :message, :status);
      if status = 0 then goto <<readagain>>; else
      /* ... error condition */
```

EMP 表のトリガーは次のようになります。

```
CREATE TRIGGER emptrig AFTER INSERT OR UPDATE OR DELETE ON emp
    BEGIN
      DBMS_ALERT.SIGNAL('emp_table_alert', 'message_text');
   END;
```

アラートが不要になると、アプリケーションは次の要求を作成します。

```
DBMS_ALERT.REMOVE('emp_table_alert');
```

この要求によって、アラートの通知側の処理量が削減されます。登録したアラートが存在している間にセッションが終了（または異常終了）した場合、そのアラートは結果的に、このパッケージの次のユーザーによって消去されます。

前述の例では、アプリケーションが中間値をすべて参照するとはかぎりませんが、常に最新のデータを参照することが保証されます。

3

DBMS_APPLICATION_INFO

アプリケーション開発者は、Oracle Trace および SQL トレース機能を持つ DBMS_APPLICATION_INFO パッケージを使用して、実行しているモジュール名またはトランザクションをデータベースに記録できます。この記録は、後で行う様々なモジュールのパフォーマンスを追跡するときに使用されます。

アプリケーションを登録することによって、システム管理者およびパフォーマンス・チューニング担当者は、パフォーマンスをモジュール別に追跡できます。システム管理者は、モジュール別のリソース使用率をこの情報から追跡することもできます。アプリケーションをデータベースに登録すると、その名前およびアクションが V$SESSION および V$SQLAREA ビューに記録されます。

ユーザーがモジュールを入力するたびに、アプリケーションがモジュール名およびアクション名を自動的に設定するようにしてください。モジュール名は、Oracle Forms アプリケーションのフォーム名または Oracle プリコンパイラ・アプリケーションのコード・セグメント名である場合があります。アクション名は、通常、モジュール内の現行トランザクションの名前または説明にしてください。

モジュールに基づいて独自の統計情報を収集する場合は、統計を最初に収集する別のスキーマにこのパッケージのバージョンを記述してこのパッケージにラッパーを実装してから、SYS バージョンのパッケージをコールします。このようにして、DBMS_APPLICATION_INFO のパブリック・シノニムを、DBA バージョンのパッケージまで変更できます。

この章では、次の項目について説明します。

注意： DBMS_APPLICATION_INFO のパブリック・シノニムが作成前に削除されることはありません。これは、ユーザーがパブリック・シノニムをユーザー独自のパッケージまでリダイレクトできるようにするためです。

権限

これ以上の権限は不要です。DBMSUTIL.SQL スクリプトは、CATPROC.SQL によりすでに実行されています。

DBMS_APPLICATION_INFO サブプログラムの要約

表 3-1 DBMS_APPLICATION_INFO パッケージのサブプログラム

サブプログラム	説明
「SET_MODULE プロシージャ」3-3 ページ	現在実行中のモジュール名を新規モジュールに設定します。
「SET_ACTION プロシージャ」3-4 ページ	現行モジュール内の現行アクション名を設定します。
「READ_MODULE プロシージャ」3-5 ページ	現行セッションのモジュールおよびアクションのフィールド値を読み込みます。
「SET_CLIENT_INFO プロシージャ」3-6 ページ	セッションのクライアント情報フィールドを設定します。
「READ_CLIENT_INFO プロシージャ」3-6 ページ	現行セッションの client_info フィールドの値を読み込みます。
「SET_SESSION_LONGOPS プロシージャ」3-7 ページ	V$SESSION_LONGOP 表に行を設定します。

SET_MODULE プロシージャ

このプロシージャは、現行のアプリケーションまたはモジュールの名前を設定します。モジュール名には、プロシージャ名（ストアド・プロシージャを使用している場合）またはアプリケーション名を設定してください。アクション名には、実行されるアクションを説明する名前を設定してください。

構文

```
DBMS_APPLICATION_INFO.SET_MODULE (
   module_name IN VARCHAR2,
   action_name IN VARCHAR2);
```

パラメータ

表 3-2 SET_MODULE プロシージャのパラメータ

パラメータ	説明
module_name	現在実行中のモジュール名。現行のモジュールが終了したときは、新規モジュールがある場合はその名前で、ない場合は NULL でこのプロシージャをコールします。48 バイトを超える名前は切り捨てられます。
action_name	現行モジュール内の現行アクション名。アクションを指定しない場合は、この値を NULL に設定してください。32 バイトを超える名前は切り捨てられます。

例

```
CREATE or replace PROCEDURE add_employee(
  name VARCHAR2,
  salary NUMBER,
  manager NUMBER,
  title VARCHAR2,
  commission NUMBER,
  department NUMBER) AS
BEGIN
  DBMS_APPLICATION_INFO.SET_MODULE(
    module_name => 'add_employee',
    action_name => 'insert into emp');
  INSERT INTO emp
    (ename, empno, sal, mgr, job, hiredate, comm, deptno)
    VALUES (name, emp_seq.nextval, salary, manager, title, SYSDATE,
            commission, department);
  DBMS_APPLICATION_INFO.SET_MODULE(null,null);
END;
```

SET_ACTION プロシージャ

このプロシージャは、現行モジュール内の現行アクション名を設定します。アクション名には、実行されている現行のアクションを説明する名前を設定してください。アクション名は、すべてのトランザクションの開始前に設定することをお薦めします。

構文

```
DBMS_APPLICATION_INFO.SET_ACTION (
   action_name IN VARCHAR2);
```

パラメータ

表 3-3 SET_ACTION プロシージャのパラメータ

パラメータ	説明
action_name	現行モジュール内の現行アクション名。現行アクションが終了したときは、次のアクションがある場合はその名前で、ない場合は NULL でこのプロシージャをコールします。32 バイトを超える名前は切り捨てられます。

使用上の注意

後続のトランザクションのログが正しく記録されるように、トランザクション完了後はトランザクション名を NULL に設定してください。トランザクション名を NULL に設定しないと、後続のトランザクションのログがその前のトランザクション名で記録される可能性があります。

例

次のコードは、登録プロシージャを使用するトランザクションの例です。

```
CREATE OR REPLACE PROCEDURE bal_tran (amt IN NUMBER(7,2)) AS
BEGIN

-- balance transfer transaction

   DBMS_APPLICATION_INFO.SET_ACTION(
      action_name => 'transfer from chk to sav');
   UPDATE chk SET bal = bal + :amt
      WHERE acct# = :acct;
   UPDATE sav SET bal = bal - :amt
      WHERE acct# = :acct;
   COMMIT;
   DBMS_APPLICATION_INFO.SET_ACTION(null);

END;
```

READ_MODULE プロシージャ

このプロシージャは、現行セッションのモジュールおよびアクション・フィールドの値を読み込みます。

構文

```
DBMS_APPLICATION_INFO.READ_MODULE (
   module_name OUT VARCHAR2,
   action_name OUT VARCHAR2);
```

パラメータ

表 3-4 READ_MODULE プロシージャのパラメータ

パラメータ	説明
module_name	SET_MODULE のコールによってモジュール名に設定された最後の値
action_name	SET_ACTION または SET_MODULE のコールによってアクション名に設定された最後の値

使用上の注意

登録アプリケーションのモジュール名およびアクション名は、V$SQLAREA を問い合せるか、または READ_MODULE プロシージャをコールして取り出せます。クライアント情報は、V$SESSION ビューを問い合せるか、または READ_CLIENT_INFO プロシージャをコールして取り出せます。

例

次の問合せのサンプルは、V$SQLAREA の MODULE および ACTION 列の使用方法の例です。

```
SELECT sql_text, disk_reads, module, action
FROM v$sqlarea
WHERE module = 'add_employee';

SQL_TEXT DISK_READS MODULE ACTION
------------------ ---------- ------------------ ----------------
INSERT INTO emp 1 add_employee insert into emp
(ename, empno, sal, mgr, job, hiredate, comm, deptno)
VALUES
(name, next.emp_seq, manager, title, SYSDATE, commission, department)

1 row selected.
```

SET_CLIENT_INFO プロシージャ

このプロシージャは、クライアント・アプリケーションに関する追加情報を提供します。

構文

```
DBMS_APPLICATION_INFO.SET_CLIENT_INFO (
   client_info IN VARCHAR2);
```

パラメータ

表 3-5 SET_CLIENT_INFO プロシージャのパラメータ

パラメータ	説明
client_info	クライアント・アプリケーションに関するあらゆる追加情報を提供します。この情報は、V$SESSIONS ビューに格納されています。64 バイトを超える情報は切り捨てられます。

注意： CLIENT_INFO は、ユーザーによる読込みおよび書込みが可能です。保護アプリケーション属性の格納には、アプリケーション・コンテキスト機能を使用できます。

READ_CLIENT_INFO プロシージャ

このプロシージャでは、現行セッションの client_info フィールドの値を読み込みます。

構文

```
DBMS_APPLICATION_INFO.READ_CLIENT_INFO (
   client_info OUT VARCHAR2);
```

パラメータ

表 3-6 READ_CLIENT_INFO プロシージャのパラメータ

パラメータ	説明
client_info	SET_CLIENT_INFO プロシージャに提供された最新のクライアント情報の値

SET_SESSION_LONGOPS プロシージャ

このプロシージャは、V$SESSION_LONGOPS ビューに行を設定します。このビューは、長時間にわたって実行する操作の進行状況を示すために使用されます。パラレル実行や Server Managed Recovery などの一部の Oracle 機能は、このビューの行を使用してデータベース・バックアップなどの状態を示します。

アプリケーション固有の長時間実行タスクの進行状況に関する情報を通知するために、アプリケーションで set_session_longops プロシージャを使用できます。この結果、V$SESSION_LONGOPS ビューで進行状況を監視できます。

構文

```
DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS (
   rindex      IN OUT BINARY_INTEGER,
   slno        IN OUT BINARY_INTEGER,
   op_name     IN     VARCHAR2       DEFAULT NULL,
   target      IN     BINARY_INTEGER DEFAULT 0,
   context     IN     BINARY_INTEGER DEFAULT 0,
   sofar       IN     NUMBER         DEFAULT 0,
   totalwork   IN     NUMBER         DEFAULT 0,
   target_desc IN     VARCHAR2       DEFAULT 'unknown target',
   units       IN     VARCHAR2       DEFAULT NULL)

set_session_longops_nohint constant BINARY_INTEGER := -1;
```

プラグマ

```
pragma TIMESTAMP('1998-03-12:12:00:00');
```

パラメータ

表 3-7 SET_SESSION_LONGOPS プロシージャのパラメータ

パラメータ	説明
rindex	更新する v$session_longops 行を示すトークン。新規行を使用するには、このトークンを set_session_longops_nohint に設定します。行を再利用する場合は、先行するコールの戻り値を使用します。
slno	set_session_longops へのコール全体の情報を保存します。内部使用のためのパラメータであるため、コール元で修正しないでください。
op_name	長時間実行タスクの名前を指定します。v$session_longops の OPNAME 列に表示されます。最大長は 64 バイトです。
target	長時間実行操作中に処理されるオブジェクトを指定します。たとえば、ソートされる表の ID などを指定します。v$session_longops の TARGET 列に表示されます。
context	クライアントが格納する数。v$session_longops の CONTEXT 列に表示されます。
sofar	クライアントが格納する数。v$session_longops の SOFAR 列に表示されます。これは通常、その時点までに処理した作業量です。
totalwork	クライアントが格納する数。v$session_longops の TOTALWORK 列に表示されます。これは通常、この長時間実行操作で行う必要がある推定合計作業量です。
target_desc	この長時間操作で操作されるオブジェクトの説明を指定します。この結果、target パラメータにキャプションが提供されます。この値は、v$session_longops の TARGET_DESC フィールドに表示されます。最大長は 32 バイトです。
units	sofar および totalwork を表す単位を指定します。v$session_longops の UNITS フィールドに表示されます。最大長は 32 バイトです。

例

この例では、ループ内で 10 個のオブジェクトに対してタスクを実行します。各オブジェクトの処理が完了するたびに、プロシージャの進行状況に関する V$SESSION_LONGOPS が更新されます。

```
DECLARE
        rindex    BINARY_INTEGER;
        slno      BINARY_INTEGER;
        totalwork number;
        sofar     number;
        obj       BINARY_INTEGER;

      BEGIN
        rindex := dbms_application_info.set_session_longops_nohint;
        sofar := 0;
        totalwork := 10;

        WHILE sofar < 10 LOOP
          -- update obj based on sofar
          -- perform task on object target

          sofar := sofar + 1;
          dbms_application_info.set_session_longops(rindex, slno,
            "Operation X", obj, 0, sofar, totalwork, "table", "tables");
        END LOOP;
      END;
```

4

DBMS_APPLY_ADM

DBMS_APPLY_ADM パッケージは、適用プロセスを開始、停止および構成する管理プロシージャを提供します。

この章では、次の項目について説明します。

関連項目： 適用プロセスの詳細は、『Oracle9*i* Streams』を参照してください。

DBMS_APPLY_ADM サブプログラムの要約

表 4-1 DBMS_APPLY_ADM サブプログラム

サブプログラム	説明
「ALTER_APPLY プロシージャ」4-4 ページ	適用プロセスを変更します。
「CREATE_APPLY プロシージャ」4-9 ページ	適用プロセスを作成します。
「DELETE_ALL_ERRORS プロシージャ」4-13 ページ	指定した適用プロセスに対するすべてのエラー・トランザクションをエラー・キューから削除します。
「DELETE_ERROR プロシージャ」4-14 ページ	指定したエラー・トランザクションをエラー・キューから削除します。
「DROP_APPLY プロシージャ」4-15 ページ	適用プロセスを削除します。
「EXECUTE_ALL_ERRORS プロシージャ」4-16 ページ	指定した適用プロセスに対するエラー・キュー・トランザクションを再実行します。
「EXECUTE_ERROR プロシージャ」4-17 ページ	指定したエラー・キュー・トランザクションを再実行します。
「GET_ERROR_MESSAGE ファンクション」4-18 ページ	指定したメッセージ番号とトランザクション識別子に対するメッセージ・ペイロードをエラー・キューから戻します。
「SET_DML_HANDLER プロシージャ」4-19 ページ	指定した適用プロセスを使用する指定オブジェクトに対する操作オプションを変更します。
「SET_GLOBAL_INSTANTIATION_SCN プロシージャ」4-24 ページ	指定したソース・データベースに対する指定インスタンス化 SCN を記録します。
「SET_KEY_COLUMNS プロシージャ」4-27 ページ	ローカル適用の目的で代替主キーとして使用される列セットを記録し、指定オブジェクトに対する既存の代替主キー列が存在する場合は、それを削除します。
「SET_PARAMETER プロシージャ」4-29 ページ	適用パラメータを指定した値に設定します。
「SET_SCHEMA_INSTANTIATION_SCN プロシージャ」4-33 ページ	指定したソース・データベース内の指定スキーマに対する指定インスタンス化 SCN を記録します。
「SET_TABLE_INSTANTIATION_SCN プロシージャ」4-36 ページ	指定したソース・データベース内の指定表に対する指定インスタンス化 SCN を記録します。

表 4-1 DBMS_APPLY_ADM サブプログラム（続き）

サブプログラム	説明
「SET_UPDATE_CONFLICT_HANDLER プロシージャ」4-38 ページ	指定オブジェクトに対する更新競合ハンドラを追加、更新または削除します。
「START_APPLY プロシージャ」4-41 ページ	適用プロセスにイベントの適用開始を指示します。
「STOP_APPLY プロシージャ」4-42 ページ	適用プロセスを停止してイベントの適用を中断し、適用が終了していないトランザクションをロールバックします。

注意： 特に指定されていないかぎり、プロシージャとファンクションはすべてコミットします。

ALTER_APPLY プロシージャ

適用プロセスを変更します。

構文

```
DBMS_APPLY_ADM.ALTER_APPLY(
   apply_name               IN  VARCHAR2,
   rule_set_name            IN  VARCHAR2  DEFAULT NULL,
   remove_rule_set          IN  BOOLEAN   DEFAULT false,
   message_handler          IN  VARCHAR2  DEFAULT NULL
   remove_message_handler   IN  BOOLEAN   DEFAULT false,
   ddl_handler              IN  VARCHAR2  DEFAULT NULL,
   remove_ddl_handler       IN  BOOLEAN   DEFAULT false,
   apply_user               IN  VARCHAR2  DEFAULT NULL,
   apply_tag                IN  RAW       DEFAULT NULL,
   remove_apply_tag         IN  BOOLEAN   DEFAULT false);
```

パラメータ

表 4-2 ALTER_APPLY プロシージャのパラメータ

パラメータ	説明
apply_name	変更する適用プロセスの名前。既存の適用プロセス名を指定する必要があります。
rule_set_name	この適用プロセスに対する適用ルールが含まれているルール・セットの名前。適用プロセスに対してルール・セットを使用する場合は、既存のルール・セットを [*schema_name*.]*rule_set_name* の形式で指定する必要があります。たとえば、hr スキーマ内の job_apply_rules という名前のルール・セットを指定するには、hr.job_apply_rules と入力します。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。 指定したルール・セットが存在しない場合はエラーが戻されます。ルール・セットを作成し、そのルール・セットにルールを追加するには、DBMS_RULE_ADM パッケージを使用します。 NULL を指定すると、適用プロセスは、そのキュー内のすべての LCR およびユーザー・メッセージを適用します。

表 4-2 ALTER_APPLY プロシージャのパラメータ（続き）

パラメータ	説明
remove_rule_set	TRUE に設定すると、指定した適用プロセスに対するルール・セットが削除されます。 FALSE に設定すると、指定した適用プロセスに対するルール・セットは保持されます。 rule_set_name パラメータが NULL 以外の場合、このパラメータは FALSE に設定してください。
message_handler	適用プロセスに対する、キュー内の非 LCR メッセージを処理するユーザー定義プロシージャ。既存のプロシージャを、次のいずれかの形式で指定する必要があります。 ■ [*schema_name*.]*procedure_name* ■ [*schema_name*.]*package_name.procedure_name* プロシージャがパッケージ内にある場合は、*package_name* を指定する必要があります。たとえば、hr スキーマ内の apply_pkg パッケージにある process_msgs という名前のプロシージャを指定するには、hr.apply_pkg.process_msgs と入力します。指定したプロシージャが存在しない場合はエラーが戻されます。 スキーマが指定されない場合は、ALTER_APPLY プロシージャの起動ユーザー名がデフォルトで使用されます。このユーザーには、指定したメッセージ・ハンドラの EXECUTE 権限が必要です。
remove_message_handler	TRUE に設定すると、指定した適用プロセスに対するメッセージ・ハンドラが削除されます。 FALSE に設定すると、指定した適用プロセスに対するメッセージ・ハンドラは保持されます。 message_handler パラメータが NULL 以外の場合、このパラメータは FALSE に設定してください。

表 4-2 ALTER_APPLY プロシージャのパラメータ（続き）

パラメータ	説明
ddl_handler	適用プロセスに対する、キュー内の DDL LCR を処理するユーザー定義プロシージャ。既存のプロシージャを [*schema_name*.]*procedure_name* の形式で指定する必要があります。たとえば、hr スキーマ内の process_ddls という名前のプロシージャを指定するには、hr.process_ddls と入力します。指定したプロシージャが存在しない場合はエラーが戻されます。 スキーマが指定されない場合は、ALTER_APPLY プロシージャの起動ユーザー名がデフォルトで使用されます。このユーザーには、指定した DDL ハンドラの EXECUTE 権限が必要です。 適用された DDL LCR はすべて自動的にコミットします。したがって、DDL ハンドラが、DDL LCR の EXECUTE メンバー・プロシージャをコールすると、コミットが自動的に実行されます。
remove_ddl_handler	TRUE に設定すると、指定した適用プロセスに対する DDL ハンドラが削除されます。 FALSE に設定すると、指定した適用プロセスに対する DDL ハンドラは保持されます。 ddl_handler パラメータが NULL 以外の場合、このパラメータは FALSE に設定してください。

表 4-2 ALTER_APPLY プロシージャのパラメータ（続き）

パラメータ	説明
apply_user	DML 変更と DDL 変更をすべて適用し、ユーザー定義の適用ハンドラを実行するユーザー。NULL の場合、適用ユーザーは変更されません。 指定したユーザーには、適用オブジェクトで DML 変更と DDL 変更を実行する権限、および適用ハンドラを実行する権限が必要です。また、適用プロセスで使用されるキューに対するデキュー権限、および適用プロセスで使用されるルール・セットと変換ファンクションを実行する権限も必要です。これらの権限は、適用ユーザーに直接付与する必要があります。ロールを通して付与することはできません。 デフォルトでは、このパラメータは適用プロセスを作成したユーザーに設定されます。適用プロセスを作成するには、このパッケージ内の CREATE_APPLY プロシージャ、または次に示す DBMS_STREAMS_ADM パッケージ内のいずれかのプロシージャを、streams_type パラメータを apply に設定して実行します。 ■ ADD_GLOBAL_RULES ■ ADD_SCHEMA_RULES ■ ADD_TABLE_RULES ■ ADD_SUBSET_RULES **注意：** 指定したユーザーが DROP USER . . . CASCADE を使用して削除されると、適用プロセスに対する apply_user は NULL に自動的に設定されます。適用プロセスを実行するには、適用ユーザーを指定する必要があります。
apply_tag	指定した適用プロセスによって生成された REDO エントリに追加されるバイナリ・タグ。タグは、LCR の追跡に使用できるバイナリ値です。 タグは、適用プロセスを実行しているデータベースの取得プロセスが、適用プロセスによって加えられた変更を取得した場合にのみ関係します。この場合、取得した変更に、このパラメータで指定されたタグが含まれます。 NULL（デフォルト）の場合、適用プロセスに対する適用タグは変更されません。 次は、16 進値 17 のタグの例です。 HEXTORAW('17') **関連項目：** タグの詳細は、『Oracle9*i* Streams』を参照してください。

表 4-2 ALTER_APPLY プロシージャのパラメータ（続き）

パラメータ	説明
remove_apply_tag	TRUE に設定すると、指定した適用プロセスに対する適用タグに NULL が設定され、適用プロセスが生成した REDO エントリに NULL タグが付加されます。 FALSE に設定すると、指定した適用プロセスに対する適用タグは保持されます。 apply_tag パラメータが NULL 以外の場合、このパラメータは FALSE に設定してください。

使用上の注意

適用プロセスは、次に示す ALTER_APPLY プロシージャのパラメータ値を 1 つ以上変更すると、自動的に停止および再開します。

- message_handler
- ddl_handler
- apply_user
- apply_tag

CREATE_APPLY プロシージャ

適用プロセスを作成します。

構文

```
DBMS_APPLY_ADM.CREATE_APPLY(
   queue_name           IN  VARCHAR2,
   apply_name           IN  VARCHAR2,
   rule_set_name        IN  VARCHAR2  DEFAULT NULL,
   message_handler      IN  VARCHAR2  DEFAULT NULL,
   ddl_handler          IN  VARCHAR2  DEFAULT NULL,
   apply_user           IN  VARCHAR2  DEFAULT NULL,
   apply_database_link  IN  VARCHAR2  DEFAULT NULL,
   apply_tag            IN  RAW       DEFAULT '00',
   apply_captured       IN  BOOLEAN   DEFAULT false);
```

パラメータ

表 4-3 CREATE_APPLY プロシージャのパラメータ

パラメータ	説明
queue_name	適用プロセスが LCR およびユーザー・メッセージをデキューするキュー名。既存のキューを [*schema_name*.] *queue_name* の形式で指定する必要があります。たとえば、hr スキーマ内の streams_queue という名前のキューを指定するには、hr.streams_queue と入力します。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。 **注意：** queue_name の設定は、適用プロセスの作成後には変更できません。
apply_name	作成する適用プロセスの名前。NULL 指定は許可されていません。 **注意：** apply_name の設定は、適用プロセスの作成後には変更できません。

表 4-3 CREATE_APPLY プロシージャのパラメータ（続き）

パラメータ	説明
rule_set_name	この適用プロセスに対する適用ルールが含まれているルール・セットの名前。適用プロセスに対してルール・セットを使用する場合は、既存のルール・セットを [*schema_name*.]*rule_set_name* の形式で指定する必要があります。たとえば、hr スキーマ内の job_apply_rules という名前のルール・セットを指定するには、hr.job_apply_rules と入力します。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。 指定したルール・セットが存在しない場合はエラーが戻されます。ルール・セットを作成し、そのルール・セットにルールを追加するには、DBMS_RULE_ADM パッケージを使用します。 NULL を指定すると、適用プロセスは、そのキュー内のすべての LCR およびユーザー・メッセージを適用します。
message_handler	適用プロセスに対する、キュー内の非 LCR メッセージを処理するユーザー定義プロシージャ。既存のプロシージャを、次のいずれかの形式で指定する必要があります。 ■ [*schema_name*.]*procedure_name* ■ [*schema_name*.]*package_name.procedure_name* プロシージャがパッケージ内にある場合は、*package_name* を指定する必要があります。たとえば、hr スキーマ内の apply_pkg パッケージにある process_msgs という名前のプロシージャを指定するには、hr.apply_pkg.process_msgs と入力します。指定したプロシージャが存在しない場合はエラーが戻されます。 スキーマが指定されない場合は、CREATE_APPLY プロシージャの起動ユーザー名がデフォルトで使用されます。このユーザーには、指定したメッセージ・ハンドラの EXECUTE 権限が必要です。 メッセージ・ハンドラ・プロシージャの詳細は、4-13 ページの「使用上の注意」を参照してください。

表 4-3 CREATE_APPLY プロシージャのパラメータ（続き）

パラメータ	説明
ddl_handler	適用プロセスに対する、キュー内の DDL LCR を処理するユーザー定義プロシージャ。既存のプロシージャを、次のいずれかの形式で指定する必要があります。 ■ [*schema_name*.]*procedure_name* ■ [*schema_name*.]*package_name*.*procedure_name* プロシージャがパッケージ内にある場合は、*package_name* を指定する必要があります。たとえば、hr 内の apply_pkg パッケージにある process_ddls という名前のプロシージャを指定するには、hr.apply_pkg.process_ddls と入力します。指定したプロシージャが存在しない場合はエラーが戻されます。 スキーマが指定されない場合は、CREATE_APPLY プロシージャの起動ユーザー名がデフォルトで使用されます。このユーザーには、指定した DDL ハンドラの EXECUTE 権限が必要です。 適用された DDL LCR はすべて自動的にコミットします。したがって、DDL ハンドラが、DDL LCR の EXECUTE メンバー・プロシージャをコールすると、コミットが自動的に実行されます。 DDL ハンドラ・プロシージャの詳細は、4-13 ページの「使用上の注意」を参照してください。
apply_user	DML 変更と DDL 変更をすべて適用し、ユーザー定義の適用ハンドラを実行するユーザー。NULL の場合は、CREATE_APPLY プロシージャの実行ユーザーが使用されます。 ユーザーには、適用オブジェクトに対して DML 変更と DDL 変更を実行する権限、および適用ハンドラを実行する権限が必要です。また、適用プロセスで使用されるキューに対するデキュー権限、および適用プロセスで使用されるルール・セットと変換ファンクションを実行する権限も必要です。これらの権限は、適用ユーザーに直接付与する必要があります。ロールを通して付与することはできません。 **注意：** 指定したユーザーが DROP USER . . . CASCADE を使用して削除されると、適用プロセスに対する apply_user 設定は NULL に自動的に設定されます。適用プロセスを実行するには、適用ユーザーを指定する必要があります。 **関連項目：** 変更の適用に必要な権限の詳細は、『Oracle9*i* Streams』を参照してください。

表 4-3 CREATE_APPLY プロシージャのパラメータ（続き）

パラメータ	説明
apply_database_link	適用プロセスがメッセージを適用するデータベース。このパラメータは、Oracle からオラクル以外のシステム（例 : Sybase）に変更を適用するときに使用されます。ローカル・データベースでメッセージを適用するように指定するには、このパラメータを NULL に設定します。 **注意 :** apply_database_link の設定は、適用プロセスの作成後には変更できません。
apply_tag	指定した適用プロセスによって生成された REDO エントリに追加されるバイナリ・タグ。タグは、LCR の追跡に使用できるバイナリ値です。 タグは、適用プロセスを実行しているデータベースの取得プロセスが、適用プロセスによって加えられた変更を取得した場合にのみ関係します。この場合、取得した変更に、このパラメータで指定されたタグが含まれます。 デフォルトでは、適用プロセスに対するタグは、'00'（ダブル・ゼロ）に相当する 16 進値です。 次は、16 進値 17 のタグの例です。 `HEXTORAW('17')` NULL に設定すると、適用プロセスは NULL タグを付加して REDO エントリを生成します。 **関連項目 :** タグの詳細は、『Oracle9*i* Streams』を参照してください。
apply_captured	TRUE または FALSE。 TRUE に設定すると、適用プロセスは、キュー内にある、Streams 取得プロセスによって取得されたイベントのみを適用します。 FALSE に設定すると、キュー内にある、ユーザーがエンキューしたイベントのみを適用します。これらのイベントは、Streams 取得プロセスによって取得されなかったユーザー・メッセージです。これらのメッセージには、ユーザーが作成した LCR が含まれる場合と含まれない場合があります。 キュー内にある、取得されたイベントとユーザーがエンキューしたイベントの両方を適用するには、少なくとも 2 つの適用プロセスを作成する必要があります。 **注意 :** apply_captured の設定は、適用プロセスの作成後には変更できません。 **関連項目 :** 取得されたイベントとユーザーがエンキューしたイベントの詳細は、『Oracle9*i* Streams』を参照してください。

使用上の注意

message_handler パラメータおよび ddl_handler パラメータに指定されたプロシージャには、次の署名が必要です。

```
PROCEDURE handler_procedure (
   parameter_name   IN  SYS.AnyData);
```

handler_procedure はプロシージャ名、*parameter_name* はプロシージャに渡されるパラメータ名を表します。メッセージ・ハンドラの場合、プロシージャに渡されるパラメータは、ユーザー・メッセージを SYS.AnyData にカプセル化したものです。DDL ハンドラの場合は、DDL LCR を SYS.AnyData にカプセル化したものです。

> **関連項目：** DDL LCR の詳細は、第 108 章「論理変更レコードのタイプ」を参照してください。

DELETE_ALL_ERRORS プロシージャ

指定した適用プロセスに対するすべてのエラー・トランザクションをエラー・キューから削除します。

構文

```
DBMS_APPLY_ADM.DELETE_ALL_ERRORS(
   apply_name     IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 4-4 DELETE_ALL_ERRORS プロシージャのパラメータ

パラメータ	説明
apply_name	トランザクション処理時にエラーが発生した適用プロセスの名前。 NULL の場合は、すべての適用プロセスに対するすべてのエラー・トランザクションが削除されます。

DELETE_ERROR プロシージャ

指定したエラー・トランザクションをエラー・キューから削除します。

構文

```
DBMS_APPLY_ADM.DELETE_ERROR(
   local_transaction_id    IN VARCHAR2);
```

パラメータ

表 4-5 DELETE_ERROR プロシージャのパラメータ

パラメータ	説明
local_transaction_id	削除するエラー・トランザクションの識別番号。指定したトランザクションがエラー・キューに存在しない場合は、エラーが発生します。

DROP_APPLY プロシージャ

適用プロセスを削除します。

構文

```
DBMS_APPLY_ADM.DROP_APPLY(
   apply_name          IN VARCHAR2);
```

パラメータ

表 4-6 DROP_APPLY プロシージャのパラメータ

パラメータ	説明
apply_name	削除する適用プロセスの名前。既存の適用プロセス名を指定する必要があります。

EXECUTE_ALL_ERRORS プロシージャ

指定した適用プロセスに対するエラー・キュー・トランザクションを再実行します。

トランザクションは、コミット SCN 順に再実行されます。エラーが発生すると、エラーの再実行を停止します。

構文

```
DBMS_APPLY_ADM.EXECUTE_ALL_ERRORS(
   apply_name           IN VARCHAR2  DEFAULT NULL
   execute_as_user      IN BOOLEAN   DEFAULT false);
```

パラメータ

表 4-7 EXECUTE_ALL_ERRORS プロシージャのパラメータ

パラメータ	説明
apply_name	トランザクション処理時にエラーが発生した適用プロセスの名前。 NULL の場合は、すべての適用プロセスに対するすべてのエラー・トランザクションが再実行されます。
execute_as_user	TRUE に設定すると、カレント・ユーザーのセキュリティ・コンテキスト内でトランザクションが再実行されます。 FALSE に設定すると、トランザクションの元の受信者のセキュリティ・コンテキスト内で各トランザクションが再実行されます。元の受信者とは、エラー発生時にトランザクションを処理していたユーザーです。DBA_APPLY_ERROR データ・ディクショナリ・ビューに、エラー・キュー内の各トランザクションに対する元の受信者がリストされます。 トランザクションを実行するユーザーには、適用オブジェクトに対して DML 変更と DDL 変更を実行する権限、および適用ハンドラを実行する権限が必要です。また、適用プロセスで使用されるキューに対するデキュー権限も必要です。

EXECUTE_ERROR プロシージャ

指定したエラー・キュー・トランザクションを再実行します。

構文

```
DBMS_APPLY_ADM.EXECUTE_ERROR(
   local_transaction_id     IN VARCHAR2,
   execute_as_user          IN BOOLEAN   DEFAULT FALSE);
```

パラメータ

表 4-8 EXECUTE_ERROR プロシージャのパラメータ

パラメータ	説明
local_transaction_id	実行するエラー・トランザクションの識別番号。指定したトランザクションがエラー・キューに存在しない場合は、エラーが発生します。
execute_as_user	TRUE に設定すると、カレント・ユーザーのセキュリティ・コンテキスト内でトランザクションが再実行されます。 FALSE に設定すると、トランザクションの元の受信者のセキュリティ・コンテキスト内でトランザクションが再実行されます。元の受信者とは、エラー発生時にトランザクションを処理していたユーザーです。DBA_APPLY_ERROR データ・ディクショナリ・ビューに、エラー・キュー内の各トランザクションに対する元の受信者がリストされます。 トランザクションを実行するユーザーには、適用オブジェクトに対して DML 変更と DDL 変更を実行する権限、および適用ハンドラを実行する権限が必要です。また、適用プロセスで使用されるキューに対するデキュー権限も必要です。

GET_ERROR_MESSAGE ファンクション

指定したメッセージ番号とトランザクション識別子に対するメッセージ・ペイロードをエラー・キューから戻します。

構文

```
DBMS_APPLY_ADM.GET_ERROR_MESSAGE(
   message_number          IN NUMBER,
   local_transaction_id    IN VARCHAR2)
RETURN Sys.Anydata;
```

パラメータ

表 4-9 GET_ERROR_MESSAGE ファンクションのパラメータ

パラメータ	説明
message_number	メッセージの識別番号。DBA_APPLY_ERROR データ・ディクショナリ・ビューを問い合せて、各適用エラーのメッセージ番号を表示します。
local_transaction_id	メッセージを戻すエラー・トランザクションの識別子。

SET_DML_HANDLER プロシージャ

指定オブジェクトでの指定操作に対する DML ハンドラとしてユーザー・プロシージャを設定します。ユーザー・プロシージャは、指定オブジェクトでの指定操作に対する適用動作を変更します。このプロシージャは、接続先データベースで実行します。SET_DML_HANDLER プロシージャは、カスタマイズされた適用を使用して、DML 変更（行 LCR）が含まれる論理変更レコードを適用する方法をユーザーに提供します。

error_handler パラメータを TRUE に設定すると、ユーザー・プロシージャがエラー・ハンドラとして指定されます。エラー・ハンドラは、行 LCR で適用プロセス・エラーが発生したときに起動されます。このようなエラーは、競合ハンドラが指定されていないか、または更新競合ハンドラで競合を解消できない場合に、データ競合が原因で発生します。error_handler を FALSE に設定すると、ユーザー・プロシージャは、エラー・ハンドラではなく DML ハンドラとして指定され、指定オブジェクトで指定操作を実行するかわりに、DML ハンドラが常に実行されます。

このプロシージャは、オブジェクトでの特定操作に対して、DML ハンドラまたはエラー・ハンドラのいずれかを設定します。同じオブジェクトと操作に対して DML ハンドラとエラー・ハンドラの両方は設定できません。

ソース・データベースでは、DML ハンドラまたはエラー・ハンドラで必要な列に対して、絶対的な補助ログ・グループを指定する必要があります。

注意： 現在、Oracle 以外のデータベースに変更を適用している適用プロセスに対するエラー・ハンドラの設定はサポートされていません。

構文

```
DBMS_APPLY_ADM.SET_DML_HANDLER(
   object_name          IN  VARCHAR2,
   object_type          IN  VARCHAR2,
   operation_name       IN  VARCHAR2,
   error_handler        IN  BOOLEAN  DEFAULT false,
   user_procedure       IN  VARCHAR2,
   apply_database_link  IN  VARCHAR2  DEFAULT NULL);
```

パラメータ

表 4-10 SET_DML_HANDLER プロシージャのパラメータ

パラメータ	説明
object_name	ソース・オブジェクトの名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr.employees のようになります。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。
object_type	ソース・オブジェクトのタイプ。現在、指定できるソース・オブジェクトのタイプは TABLE のみです。
operation_name	操作名。次の操作名を指定できます。 ■ INSERT ■ UPDATE ■ DELETE ■ LOB_UPDATE たとえば、hr.employees 表に対してこのプロシージャを 2 回実行するとします。1 回目のコールで、operation_name を UPDATE、user_procedure を employees_update に設定します。2 回目のコールでは、operation_name を INSERT、user_procedure を employees_insert に設定します。両方とも、error_handler を FALSE に設定します。 この場合、hr.employees 表の UPDATE 操作に対して employees_update プロシージャが実行され、hr.employees 表の INSERT 操作に対しては employees_insert プロシージャが実行されます。
error_handler	TRUE に設定すると、指定オブジェクトでの指定操作に関連する行 LCR で適用プロセス・エラーが発生したときに、指定したユーザー・プロシージャが実行されます。ユーザー・プロシージャは、潜在的なエラー条件を解決しようとするか、あるいは管理者にエラーを通知するか、エラーをログに記録しようとします。 FALSE に設定すると、指定オブジェクトでの指定操作に関連するすべての行 LCR に対して、設定されるハンドラが実行されます。 **注意：** 現在、Oracle 以外のデータベースに変更を適用するときに、エラー・ハンドラはサポートされません。

表 4-10 SET_DML_HANDLER プロシージャのパラメータ（続き）

パラメータ	説明
user_procedure	指定オブジェクトでの指定操作に対する適用時に起動されるユーザー定義プロシージャ。プロシージャが DML ハンドラの場合は、Oracle で実行されるデフォルト適用のかわりに起動されます。プロシージャがエラー・ハンドラの場合は、適用プロセスでエラーが発生したときに起動されます。
apply_database_link	Oracle 以外のデータベースへのデータベース・リンクの名前。このパラメータは、接続先データベースが Oracle 以外のデータベースの場合にのみ設定する必要があります。

使用上の注意

SET_DML_HANDLER を使用すると、指定オブジェクトで指定操作を実行する行 LCR に対して、一般的な DML ハンドラまたはエラー・ハンドラのいずれかを設定できます。次の項では、一般的な DML ハンドラ・プロシージャの署名およびエラー・ハンドラ・プロシージャの署名について説明します。

どちらの場合も、次のいずれかの形式で user_procedure パラメータに完全なプロシージャ名を指定する必要があります。

- [*schema_name*.]*package_name.procedure_name*
- [*schema_name*.]*procedure_name*

プロシージャがパッケージ内にある場合は、*package_name* を指定する必要があります。スキーマが指定されない場合は、SET_DML_HANDLER プロシージャの起動ユーザー名がデフォルトで使用されます。このユーザーには、指定したプロシージャの EXECUTE 権限が必要です。

たとえば、procedure_name のプロパティが次のとおりであるとします。

- *schema_name* は hr です。
- *package_name* は apply_pkg です。
- *procedure_name* は employees_default です。

この場合、次のように指定します。

```
hr.apply_pkg.employees_default
```

ユーザー・プロシージャには、次の制限事項が適用されます。

- COMMIT 文または ROLLBACK 文を実行しないでください。これらの文を実行すると、LCR が含まれるトランザクションの一貫性が失われる危険性があります。
- 行 LCR に対して EXECUTE メンバー・プロシージャを使用して行を操作している場合は、行操作で複数行を操作しないでください。複数行を操作する DML 文は、手動で構成および実行する必要があります。
- コマンド・タイプが UPDATE または DELETE の場合、LCR に対して EXECUTE メンバー・プロシージャを使用して再実行される行操作では、以前の値リストにキー全体が含まれている必要があります。SET_KEY_COLUMNS プロシージャで代替キーが指定されていないかぎり、キーは主キーです。
- コマンド・タイプが INSERT の場合、LCR に対して EXECUTE メンバー・プロシージャを使用して再実行される行操作では、新規の値リストにキー全体が含まれている必要があります。キー全体が組み込まれていない場合は、行が重複する可能性があります。SET_KEY_COLUMNS プロシージャで代替キーが指定されていないかぎり、キーは主キーです。

一般的な DML ハンドラ・プロシージャの署名

user_procedure パラメータに指定されるプロシージャには、次の署名が必要です。

```
PROCEDURE user_procedure (
   parameter_name   IN  SYS.AnyData);
```

user_procedure はプロシージャ名、*parameter_name* はプロシージャに渡されるパラメータ名を表します。プロシージャに渡されるパラメータは、行 LCR を SYS.AnyData にカプセル化したものです。

> **関連項目：** LCR の詳細は、第 108 章「論理変更レコードのタイプ」を参照してください。

エラー・ハンドラ・プロシージャの署名

エラー処理用に作成するプロシージャには、次の署名が必要です。

```
PROCEDURE user_procedure (
     message             IN SYS.AnyData,
     error_stack_depth   IN NUMBER,
     error_numbers       IN DBMS_UTILITY.NUMBER_ARRAY,
     error_messages      IN emsg_array);
```

注意 :

- 各パラメータは必須で、指定したデータ・タイプであることが必要です。ただし、パラメータ名は変更できます。
- emsg_array パラメータはユーザー定義配列で、76 文字以上の VARCHAR2 タイプの表であることが必要です。

エラー・ハンドラを実行すると、次のいずれかの結果になります。

- エラー・ハンドラがエラーを正常に解決し、適用プロセスに制御を戻します。
- エラー・ハンドラがエラーを解決できず、エラーが発生します。エラーが発生すると、トランザクションはロールバックされ、エラー・キューに入れられます。

DML 操作を再試行する場合は、エラー・ハンドラ・プロシージャで、LCR に対して EXECUTE メンバー・プロシージャを実行してください。

SET_GLOBAL_INSTANTIATION_SCN プロシージャ

指定したソース・データベースに対する指定インスタンス化 SCN を記録します。このプロシージャは、データベースに対する既存のインスタンス化 SCN を上書きします。

このプロシージャを使用すると、データベースに対する DDL LCR の中で、適用プロセスで無視するものと適用するものを正確に制御できます。ソース・データベースからのデータベース・オブジェクトに対する DDL LCR のコミット SCN が、一部の接続先データベースでそのデータベースに対するインスタンス化 SCN 以下の場合は、接続先データベースでの適用プロセスで、その DDL LCR は無視されます。それ以外の場合は、DDL LCR が適用されます。

このプロシージャによって指定されるインスタンス化 SCN は、DDL LCR に、object_owner、base_table_owner および base_table_name が指定されていない場合にのみ、DDL LCR に対して使用されます。たとえば、このプロシージャによって設定されるインスタンス化 SCN は、command_type が CREATE USER の DDL LCR に使用されます。

注意： データベースに対して SET_GLOBAL_INSTANTIATION_SCN を実行する場合は、データベース内の既存のスキーマすべてに対して SET_SCHEMA_INSTANTIATION_SCN を実行し、データベース内の既存の表すべてに対して SET_TABLE_INSTANTIATION_SCN を実行する必要があります。これらの実行後、データベースに新しいスキーマや表を追加する場合は、その新しいスキーマや表に対してこれらのプロシージャを実行する必要はありません。

注意：

- このプロシージャは、DDL LCR に対してのみインスタンス化 SCN を設定します。DML 変更の結果を記録する行 LCR に対してインスタンス化 SCN を設定するには、SET_TABLE_INSTANTIATION_SCN を使用します。
- SET_SCHEMA_INSTANTIATION_SCN プロシージャによって設定されるインスタンス化 SCN は、object_owner が指定されている DDL LCR に対して使用されます。
- SET_TABLE_INSTANTIATION_SCN プロシージャによって設定されるインスタンス化 SCN は、base_table_owner と base_table_name の両方が指定された DDL LCR に対して使用されます。ただし、command_type が CREATE TABLE の DDL LCR には使用されません。
- このプロシージャによって指定されるインスタンス化 SCN は、取得プロセスによって取得された LCR に対してのみ使用されます。ユーザーが作成した LCR には使用されません。

関連項目：

- 4-33 ページ「SET_SCHEMA_INSTANTIATION_SCN プロシージャ」
- 4-36 ページ「SET_TABLE_INSTANTIATION_SCN プロシージャ」
- DDL LCR の詳細は、108-3 ページの「LCR$_DDL_RECORD タイプ」を参照してください。
- 『Oracle9*i* Streams』

構文

```
DBMS_APPLY_ADM.SET_GLOBAL_INSTANTIATION_SCN(
  source_database_name    IN  VARCHAR2,
  instantiation_scn       IN  NUMBER,
  apply_database_link     IN  VARCHAR2  DEFAULT NULL);
```

パラメータ

表 4-11 SET_GLOBAL_INSTANTIATION_SCN プロシージャのパラメータ

パラメータ	説明
source_database_name	ソース・データベースのグローバル名。たとえば、DBS1.NET のようになります。 ドメイン名を指定しないと、ローカル・ドメインがデータベース名に自動的に追加されます。たとえば、DBS1 を指定し、ローカル・ドメインが .NET の場合は、DBS1.NET が自動的に指定されます。
instantiation_scn	インスタンス化 SCN 番号。NULL を指定すると、ソース・データベースに対するインスタンス化 SCN のメタデータがデータ・ディクショナリから削除されます。
apply_database_link	Oracle 以外のデータベースへのデータベース・リンクの名前。このパラメータは、ローカル適用プロセスの接続先データベースが Oracle 以外のデータベースの場合にのみ設定する必要があります。

SET_KEY_COLUMNS プロシージャ

適用の目的で代替主キーとして使用される列セットを記録し、指定オブジェクトに対する既存の代替主キー列が存在する場合は、それを削除します。本来の主キーとは異なり、これらの列には NULL が含まれる場合があります。

空でない場合、この列セットは、指定オブジェクトの主キーより優先されます。オブジェクトにすでに主キー列があり、これらの主キー列をキーとして使用する場合は、代替キー列を指定しないでください。

このプロシージャは、接続先データベースで実行します。ソース・データベースでは、代替キー列に対して絶対的な補助ログ・グループを指定する必要があります。

注意：

- 代替キー列として指定する各列は NOT NULL 列にすることをお薦めします。また、すべての列を代替キーに含める単一の索引も作成してください。これらのガイドラインに従うと、Oracle は関連する行をさらに効率的に検索できるため、LOB に対する更新、削除およびピース単位の更新のパフォーマンスが向上します。
- 表の主キー列または代替キー列は、アプリケーションで更新できないようにしてください。更新できないようにすることにより、Oracle は行を識別でき、データの整合性を保つことができます。

注意： このプロシージャはオーバーロードされています。column_list パラメータと column_table パラメータは、両方同時には指定できません。

構文

```
DBMS_APPLY_ADM.SET_KEY_COLUMNS(
   object_name            IN   VARCHAR2,
   { column_list          IN   VARCHAR2, |
     column_table         IN   DBMS_UTILITY.NAME_ARRAY, }
   apply_database_link    IN   VARCHAR2  DEFAULT NULL);
```

パラメータ

表 4-12 SET_KEY_COLUMNS プロシージャのパラメータ

パラメータ	説明
object_name	表の名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr.employees のようになります。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。適用プロセスが異機種環境の Oracle 以外のデータベースに変更を適用している場合、オブジェクト名は検証されません。
column_list	代替主キーとして使用する、表内の列のカンマで区切られたリスト。列名の間に空白は入れません。 column_list パラメータが空または NULL の場合は、現行のキー列のセットが削除されます。
column_table	代替主キーとして使用する、表内の列の DBMS_UTILITY.NAME_ARRAY タイプの PL/SQL 索引付き表。column_table の索引は 1 から始まり、1 ずつ増え、NULL で終了する必要があります。 column_table パラメータが空または NULL の場合は、現行のキー列のセットが削除されます。
apply_database_link	Oracle 以外のデータベースへのデータベース・リンクの名前。このパラメータは、接続先データベースが Oracle 以外のデータベースの場合にのみ設定する必要があります。

SET_PARAMETER プロシージャ

適用パラメータを指定した値に設定します。

パラメータ値を変更したとき、パラメータの新しい値が有効になるまでに時間がかかる場合があります。

構文

```
DBMS_APPLY_ADM.SET_PARAMETER (
   apply_name      IN VARCHAR2,
   parameter       IN VARCHAR2,
   value           IN VARCHAR2);
```

パラメータ

表 4-13 SET_PARAMETER プロシージャのパラメータ

パラメータ	説明
apply_name	適用プロセス名。
parameter	設定するパラメータの名前。これらのパラメータのリストについては、4-30 ページの「適用プロセスのパラメータ」を参照してください。
value	パラメータに設定する値。

適用プロセスのパラメータ

次の表に、適用プロセスのパラメータを示します。

表 4-14 適用プロセスのパラメータ

パラメータ名	設定可能な値	デフォルト	説明
commit_serialization	full または none	full	適用済みトランザクションがコミットされる順序。 full の場合、適用プロセスは、ソース・データベースでコミットされた順序で、適用済みトランザクションをコミットします。 none の場合は、任意の順序でコミットされる場合があります。none を指定すると、パフォーマンスは最大になります。 指定に関係なく、適用済みトランザクションは、データ依存性と制約依存性の影響によってパラレルで実行される場合があります。 ロジカル・スタンバイ環境では、通常 full を指定します。
disable_on_error	Y または N	Y	Y の場合は、未解決の最初のエラーによって、そのエラーが致命的なエラーでない場合でも、適用プロセスは無効化されます。 N の場合は、未解決のエラーがある場合でも適用プロシージャは続行されます。
disable_on_limit	Y または N	N	Y の場合は、time_limit パラメータまたは transaction_limit パラメータによって指定された値に達したために適用プロセスが終了すると、適用プロセスは無効化されます。 N の場合、適用プロセスは、制限に達したために停止した後すぐに再開されます。
maximum_scn	有効な SCN 値または infinite	infinite	適用プロセスは、指定された値以上のコミット SCN を持つトランザクションの適用前に無効化されます。 infinite の場合は、SCN の値に関係なく適用プロセスが実行されます。

表 4-14 適用プロセスのパラメータ（続き）

パラメータ名	設定可能な値	デフォルト	説明
parallelism	正の整数	1	同時に適用可能なトランザクション数。 **注意：** ■ このパラメータの値を変更すると、適用プロセスは自動的に停止し、再開します。現在適用中のトランザクションのサイズによって、時間がかかる場合があります。 ■ parallelism パラメータを、使用可能なパラレル実行サーバー数より大きい値に設定すると、適用プロセスが無効化される場合があります。parallelism 適用プロセス・パラメータを設定するときは、PROCESSES および PARALLEL_MAX_SERVERS 初期化パラメータが適切に設定されていることを確認してください。
startup_seconds	0、正の整数または infinite	0	同じ適用プロセスの別のインスタンス化が終了するのを待機する最大秒数。同じ適用プロセスの別のインスタンス化がこの時間内に終了しない場合、適用プロセスは開始しません。 infinite の場合、適用プロセスは、同じ適用プロセスの別のインスタンス化が終了した後で開始します。
time_limit	正の整数または infinite	infinite	適用プロセスは、開始してから指定秒数が経過した後、可能なかぎり早く停止します。 infinite の場合、適用プロセスは明示的に停止されるまで実行し続けます。
trace_level	0 または正の整数	0	オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。
transaction_limit	正の整数または infinite	infinite	適用プロセスは、指定した数のトランザクションを適用した後停止します。 infinite の場合は、適用したトランザクションの数に関係なく実行し続けます。
write_alert_log	Y または N	Y	Y の場合、適用プロセスは、終了時にアラート・ログにメッセージを書き込みます。 N の場合は、終了時にアラート・ログにメッセージを書き込みません。 メッセージには、適用プロセスの停止理由が示されます。

注意：

- 正の整数として解釈されるすべてのパラメータについて、設定可能な最大値は 4,294,967,295 です。適用可能な場合、比較的大きい値には infinite を指定してください。
- SCN 設定が必要なパラメータについては、任意の有効な SCN 値を指定できます。

SET_SCHEMA_INSTANTIATION_SCN プロシージャ

指定したソース・データベース内の指定スキーマに対する指定インスタンス化 SCN を記録します。このプロシージャは、特定のスキーマに対する既存のインスタンス化 SCN を上書きします。

このプロシージャを使用すると、スキーマに対する DDL LCR の中で、適用プロセスで無視するものと適用するものを正確に制御できます。ソース・データベースからのスキーマ内のデータベース・オブジェクトに対する DDL LCR のコミット SCN が、一部の接続先データベースでそのデータベース・オブジェクトに対するインスタンス化 SCN 以下の場合は、接続先データベースでの適用プロセスで、その DDL LCR は無視されます。それ以外の場合は、DDL LCR が適用されます。

このプロシージャによって指定されるインスタンス化 SCN は、次のタイプの DDL LCR で使用されます。

- command_type が CREATE TABLE の DDL LCR
- NULL 以外の object_owner が指定されていて、base_table_owner と base_table_name がいずれも指定されていない DDL LCR

たとえば、このプロシージャによって設定されるインスタンス化 SCN は、command_type が CREATE TABLE および ALTER USER の DDL LCR に対して使用されます。

command_type が CREATE USER の DDL LCR に対しては使用されません。

注意： スキーマに対して SET_SCHEMA_INSTANTIATION_SCN を実行する場合は、スキーマ内の既存の表すべてに対して SET_TABLE_INSTANTIATION_SCN を実行する必要があります。この実行後、スキーマに新しい表を追加する場合は、これらの表に対して SET_TABLE_INSTANTIATION_SCN を実行する必要はありません。

注意：

- このプロシージャは、DDL LCR に対してのみインスタンス化 SCN を設定します。DML 変更の結果を記録する行 LCR に対してインスタンス化 SCN を設定するには、SET_TABLE_INSTANTIATION_SCN を使用します。
- SET_TABLE_INSTANTIATION_SCNによって設定されるインスタンス化 SCN は、base_table_owner と base_table_name の両方が指定された DDL LCR に対して使用されます。ただし、command_type が CREATE TABLE の DDL LCR には使用されません。
- このプロシージャによって指定されるインスタンス化 SCN は、取得プロセスによって取得された LCR に対してのみ使用されます。ユーザーが作成した LCR には使用されません。

関連項目：

- 4-24 ページ「SET_GLOBAL_INSTANTIATION_SCN プロシージャ」
- 4-36 ページ「SET_TABLE_INSTANTIATION_SCN プロシージャ」
- DDL LCR の詳細は、108-3 ページの「LCR$_DDL_RECORD タイプ」を参照してください。
- 『Oracle9*i* Streams』

構文

```
DBMS_APPLY_ADM.SET_SCHEMA_INSTANTIATION_SCN(
  source_schema_name     IN  VARCHAR2,
  source_database_name   IN  VARCHAR2,
  instantiation_scn      IN  NUMBER,
  apply_database_link    IN  VARCHAR2  DEFAULT NULL);
```

パラメータ

表 4-15 SET_SCHEMA_INSTANTIATION_SCN プロシージャのパラメータ

パラメータ	説明
source_schema_name	ソース・スキーマの名前。たとえば、hr のようになります。
source_database_name	ソース・データベースのグローバル名。たとえば、DBS1.NET のようになります。 ドメイン名を指定しないと、ローカル・ドメインがデータベース名に自動的に追加されます。たとえば、DBS1 を指定し、ローカル・ドメインが .NET の場合は、DBS1.NET が自動的に指定されます。
instantiation_scn	インスタンス化 SCN 番号。NULL を指定すると、ソース・スキーマに対するインスタンス化 SCN のメタデータがデータ・ディクショナリから削除されます。
apply_database_link	Oracle 以外のデータベースへのデータベース・リンクの名前。このパラメータは、ローカル適用プロセスの接続先データベースが Oracle 以外のデータベースの場合にのみ設定する必要があります。

SET_TABLE_INSTANTIATION_SCN プロシージャ

指定したソース・データベース内の指定表に対する指定インスタンス化 SCN を記録します。このプロシージャは、特定の表に対する既存のインスタンス化 SCN を上書きします。

このプロシージャを使用すると、表に対する LCR の中で、適用プロセスで無視するものと適用するものを正確に制御できます。ソース・データベースからの表に対する LCR のコミット SCN が、一部の接続先データベースでその表に対するインスタンス化 SCN 以下の場合は、接続先データベースでの適用プロセスで、その LCR は無視されます。それ以外の場合は、LCR が適用されます。

このプロシージャによって指定されるインスタンス化 SCN は、次のタイプの LCR で使用されます。

- 表に対する行 LCR。
- NULL 以外の base_table_owner と base_table_name が指定されている DDL LCR。ただし、command_type が CREATE TABLE の DDL LCR は除きます。

たとえば、このプロシージャによって設定されるインスタンス化 SCN は、command_type が ALTER TABLE または CREATE TRIGGER の DDL LCR に対して使用されます。

注意： このプロシージャによって指定されるインスタンス化 SCN は、取得プロセスによって取得された LCR に対してのみ使用されます。ユーザーが作成した LCR には使用されません。

関連項目：

- 4-24 ページ「SET_GLOBAL_INSTANTIATION_SCN プロシージャ」
- 4-33 ページ「SET_SCHEMA_INSTANTIATION_SCN プロシージャ」
- LCR の詳細は、108-13 ページの「LCR$_ROW_RECORD タイプ」を参照してください。
- DDL LCR の詳細は、108-3 ページの「LCR$_DDL_RECORD タイプ」を参照してください。
- 『Oracle9*i* Streams』

構文

```
DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN(
  source_object_name     IN  VARCHAR2,
  source_database_name   IN  VARCHAR2,
  instantiation_scn      IN  NUMBER,
  apply_database_link    IN  VARCHAR2  DEFAULT NULL);
```

パラメータ

表 4-16 SET_TABLE_INSTANTIATION_SCN プロシージャのパラメータ

パラメータ	説明
source_object_name	ソース・オブジェクトの名前。[*schema_name*.]*object_name* の形式で指定されます。たとえば、hr.employees のようになります。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。
source_database_name	ソース・データベースのグローバル名。たとえば、DBS1.NET のようになります。 ドメイン名を指定しないと、ローカル・ドメイン名がデータベース名に自動的に追加されます。たとえば、DBS1 を指定し、グローバル・ドメインが .NET の場合は、DBS1.NET が自動的に指定されます。
instantiation_scn	インスタンス化 SCN 番号。NULL を指定すると、ソース表に対するインスタンス化 SCN のメタデータがデータ・ディクショナリから削除されます。
apply_database_link	Oracle 以外のデータベースへのデータベース・リンクの名前。このパラメータは、ローカル適用プロセスの接続先データベースが Oracle 以外のデータベースの場合にのみ設定する必要があります。

SET_UPDATE_CONFLICT_HANDLER プロシージャ

指定オブジェクトに対する更新競合ハンドラを追加、変更または削除します。

既存の更新競合ハンドラを変更する場合は、既存の更新競合ハンドラの表と解消列を指定します。事前作成メソッドまたは列リストを変更できます。

既存の更新競合ハンドラを削除する場合は、事前作成メソッドに NULL を指定し、既存の更新競合ハンドラの表、列リストおよび解消列を指定します。

更新の競合が発生した場合、次の一連の処理が実行されます。

1. 適切な更新競合ハンドラをコールして、競合を解消します。
2. 更新競合ハンドラが指定されていない場合、または更新競合ハンドラで競合を解消できない場合は、適用プロセス、表および操作に対する適切なエラー・ハンドラをコールしてエラーを処理します。
3. エラー・ハンドラが指定されていない、またはエラー・ハンドラでエラーを解決できない場合は、エラーが発生し、エラーの原因となった行 LCR が含まれているトランザクションがエラー・キューに移されます。

注意： 現在、Oracle 以外のデータベースに適用している適用プロセスに対する更新競合ハンドラの設定はサポートされていません。

関連項目： エラー・ハンドラの設定方法の詳細は、4-23 ページの「エラー・ハンドラ・プロシージャの署名」を参照してください。

構文

```
DBMS_APPLY_ADM.SET_UPDATE_CONFLICT_HANDLER(
   object_name           IN  VARCHAR2,
   method_name           IN  VARCHAR2,
   resolution_column     IN  VARCHAR2,
   column_list           IN  DBMS_UTILITY.NAME_ARRAY,
   apply_database_link   IN  VARCHAR2  DEFAULT NULL);
```

パラメータ

表 4-17 SET_UPDATE_CONFLICT_HANDLER プロシージャのパラメータ

パラメータ	説明
object_name	更新競合ハンドラが追加、変更または削除されるスキーマと表の名前。[*schema_name*.]*object_name* の形式で指定します。 たとえば、ユーザー hr が所有する表 employees に更新競合ハンドラが追加される場合は、hr.employees と指定します。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。
method_name	作成する更新競合ハンドラのタイプ。 組込みのハンドラの 1 つを指定できます。このハンドラは、行に対してソース・データベースの列リストを適用するかどうか、または接続先データベースの行の値を保持するかどうかを決定します。 ■ MAXIMUM: ソース・データベースの列リストの値が解消列に関して大きい場合は、そのソース・データベースの列リストを適用します。それ以外の場合は、接続先データベースの値を保持します。 ■ MINIMUM: ソース・データベースの列リストの値が解消列に関して小さい場合は、そのソース・データベースの列リストを適用します。それ以外の場合は、接続先データベースの値を保持します。 ■ OVERWRITE: ソース・データベースの列リストを適用し、接続先データベースの列の値を上書きします。 ■ DISCARD: 接続先データベースの列リストを保持し、ソース・データベースの列リストを廃棄します。 NULL の場合は、同じ object_name、resolution_column および column_list を持つ既存の更新競合ハンドラを削除します。NULL 以外の場合は、同じ object_name および resolution_column を持つ既存の更新競合ハンドラを置換します。
resolution_column	更新競合ハンドラを一意に識別するために使用される列の名前。MAXIMUM および MINIMUM 事前作成メソッドの場合、解消列は競合の解消にも使用されます。解消列は、column_list パラメータにリストされた列の 1 つであることが必要です。 このパラメータに NULL は指定できません。OVERWRITE および DISCARD 事前作成メソッドの場合は、列リスト内のどの列でも使用できます。

表 4-17 SET_UPDATE_CONFLICT_HANDLER プロシージャのパラメータ

パラメータ	説明
column_list	競合ハンドラがコールされる列のリスト。 適用プロセスが行 LCR を適用しようとしたときに、リスト内の 1 つ以上の列に対して競合が発生すると、競合を解消するために競合ハンドラがコールされます。競合ハンドラは、リストにない列でのみ競合が発生した場合はコールされません。 **注意：** 競合解消では LOB 列はサポートされていません。したがって、column_list パラメータに LOB 列を指定しないでください。
apply_database_link	Oracle 以外のデータベースへのデータベース・リンクの名前。このパラメータは、接続先データベースが Oracle 以外のデータベースの場合にのみ設定する必要があります。 **注意：** 現在、Oracle 以外のデータベースに変更を適用するときに、競合ハンドラはサポートされません。

使用上の注意

次は、hr スキーマ内の employees 表に対して更新競合ハンドラを設定する場合の例です。

```
DECLARE
  cols  DBMS_UTILITY.NAME_ARRAY;
BEGIN
  cols(1) := 'salary';
  cols(2) := 'commission_pct';
  DBMS_APPLY_ADM.SET_UPDATE_CONFLICT_HANDLER(
    object_name           =>  'hr.employees',
    method_name           =>  'MAXIMUM',
    resolution_column     =>  'salary',
    column_list           =>  cols);
END;
/
```

この例では、hr.employees 表内の salary 列または commission_pct 列で競合が発生した場合にコールされる競合ハンドラが設定されます。このような競合が発生した場合は、競合を解消するために salary 列が評価されます。job_id 列など、列リストにない列でのみ競合が発生した場合、この競合ハンドラはコールされません。

START_APPLY プロシージャ

適用プロセスにイベントの適用開始を指示します。

開始ステータスは永続的に記録されます。したがって、ステータスが START の場合、適用プロセスはデータベース・インスタンスの起動時に開始されます。各適用プロセスは Oracle バックグラウンド・プロセスであり、接頭辞 AP が付加されます。

DBMS_AQADM.START_QUEUE および DBMS_AQADM.STOP_QUEUE のエンキューとデキューの状態は、適用プロセスの開始ステータスには影響を与えません。

適用プロセスは、次のプロシージャを使用して作成できます。

- DBMS_APPLY_ADM.CREATE_APPLY
- DBMS_STREAMS_ADM.ADD_GLOBAL_RULES
- DBMS_STREAMS_ADM.ADD_SCHEMA_RULES
- DBMS_STREAMS_ADM.ADD_TABLE_RULES
- DBMS_STREAMS_ADM.ADD_SUBSET_RULES

関連項目： 第 73 章「DBMS_STREAMS_ADM」

構文

```
DBMS_APPLY_ADM.START_APPLY(
   apply_name  IN VARCHAR2);
```

パラメータ

表 4-18 START_APPLY プロシージャのパラメータ

パラメータ	説明
apply_name	適用プロセス名。NULL 設定は許可されていません。

STOP_APPLY プロシージャ

適用プロセスを停止してイベントの適用を中断し、適用が終了していないトランザクションをロールバックします。

停止ステータスは永続的に記録されます。したがって、ステータスが STOP の場合、適用プロセスはデータベース・インスタンスの起動時に開始されません。

DBMS_AQADM.START_QUEUE および DBMS_AQADM.STOP_QUEUE のエンキューとデキューの状態は、適用プロセスの STOP ステータスには影響を与えません。

構文

```
DBMS_APPLY_ADM.STOP_APPLY(
   apply_name   IN  VARCHAR2,
   force        IN  BOOLEAN DEFAULT false);
```

パラメータ

表 4-19 STOP_APPLY プロシージャのパラメータ

パラメータ	説明
apply_name	適用プロセス名。NULL 設定は許可されていません。
force	TRUE に設定すると、適用プロセスは可能なかぎり早く停止されます。 FALSE の場合は、適用された一連のトランザクションに不整合のないことが確認された後で、適用プロセスが停止されます。 適用プロセスの動作は、force パラメータに指定された設定、および commit_serialization 適用プロセス・パラメータに指定された設定によって決まります。詳細は、「使用上の注意」を参照してください。

使用上の注意

次の表では、STOP_APPLY プロシージャの force パラメータ、および commit_serialization 適用プロセス・パラメータの各設定に対する適用プロセスの動作を説明します。すべての場合で、停止時に、適用プロセスは未完了のトランザクションをロールバックします。

force	commit_serialization	適用プロセスの動作
TRUE	full	適用プロセスは即時に停止され、未完了のトランザクションは適用されません。
TRUE	none	適用プロセスの停止時に、ローカルで適用済みの一部のトランザクションが、ローカルで適用されていない一部のトランザクションより、時間的に遅れてソース・データベースでコミットされる場合があります。
FALSE	full	適用プロセスは、コミット順で次のコミットされていないトランザクションが処理中の場合、そのトランザクションを適用した後で停止します。
FALSE	none	停止する前に、適用プロセスは、コミット・タイムが最新の適用済みトランザクションより早いコミット・タイムを持つトランザクションをすべて適用します。

たとえば、commit_serialization 適用プロセス・パラメータが none に設定されていて、コミット・タイムが最も早いトランザクション 1、トランザクション 1 の後にコミットされたトランザクション 2、およびコミット・タイムが最も遅いトランザクション 3 の 3 つのトランザクションがあるとします。また、STOP_APPLY プロシージャの実行時に、適用プロセスはトランザクション 1 とトランザクション 3 を適用済みで、トランザクション 2 の適用処理中であるとします。この例では、force パラメータが TRUE に設定されていると、トランザクション 2 は適用されず、適用プロセスは停止します（トランザクション 2 はロールバックされます）。ただし、force パラメータが FALSE に設定されていると、トランザクション 2 は、適用プロセスが停止する前に適用されます。

commit_serialization 適用プロセス・パラメータが full に設定されている別の例では、次のようになります。たとえば、commit_serialization 適用プロセス・パラメータが full に設定されていて、コミット・タイムが最も早いトランザクション A、トランザクション A の後にコミットされたトランザクション B、およびコミット・タイムが最も遅いトランザクション C の 3 つのトランザクションがあるとします。この場合、STOP_APPLY プロシージャの実行時に、適用プロセスはトランザクション A を適用済みで、トランザクション B と C の適用処理中であるとします。この例では、force パラメータが TRUE に設定されていると、トランザクション B と C は適用されず、適用プロセスは停止します（トランザクション B と C はロールバックされます）。ただし、force パラメータが FALSE に設定されていると、トランザクション B は適用プロセスが停止する前に適用され、トランザクション C はロールバックされます。

関連項目： commit_serialization 適用プロセス・パラメータの詳細は、4-29 ページの「SET_PARAMETER プロシージャ」を参照してください。

5

DBMS_AQ

DBMS_AQ パッケージは、Oracle のアドバンスト・キューイングへのインタフェースを提供します。

関連項目：

- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』
- DBMS_AQ で使用する TYPE に関する情報は、第 106 章「アドバンスト・キューイング・タイプ」を参照してください。

この章では、次の項目について説明します。

Java クラス

DBMS_AQ および DBMS_AQADM には、Java インタフェースを使用できます。Java インタフェースは、$ORACLE_HOME/rdbms/jlib/aqapi.jar にあります。これらのインタフェースを使用するには、ユーザーに DBMS_AQ パッケージの EXECUTE 権限が必要です。

列挙定数

BROWSE、LOCKED または REMOVE などの列挙定数を使用するときは、それを定義しているパッケージの有効範囲内で、PL/SQL 定数を指定する必要があります。操作インタフェースに関連付けられているすべてのタイプに、DBMS_AQ を付加する必要があります。たとえば、DBMS_AQ.BROWSE のようにします。

表 5-1 列挙定数

パラメータ	オプション
visibility	IMMEDIATE、ON_COMMIT
dequeue mode	BROWSE、LOCKED、REMOVE、REMOVE_NODATA
navigation	FIRST_MESSAGE、NEXT_MESSAGE、NEXT_TRANSACTION
state	WAITING、READY、PROCESSED、EXPIRED
sequence_deviation	BEFORE、TOP
wait	FOREVER、NO_WAIT
delay	NO_DELAY
expiration	NEVER
namespace	NAMESPACE_AQ、NAMESPACE_ANONYMOUS

DBMS_AQ のデータ構造

表 5-2　DBMS_AQ のデータ構造

データ構造
「オブジェクト名」5-3 ページ
「タイプ名」5-3 ページ
「AQ PL/SQL コールバック」5-4 ページ

オブジェクト名

object_name データ構造は、データベース・オブジェクトの名前を付けます。この構造は、キュー、キュー表、エージェント名およびオブジェクト・タイプに適用されます。

構文

```
object_name := VARCHAR2;
object_name := [<schema_name>.]<name>;
```

使用上の注意

オブジェクト名は、オプションのスキーマ名および名前で指定します。スキーマ名が指定されない場合は、現在のスキーマ名が使用されます。名前は、予約語に関して、『Oracle9*i* SQL リファレンス』のオブジェクト名のガイドラインに従う必要があります。スキーマ、エージェントおよびオブジェクト・タイプの名前は、30 バイト以内で設定します。キューおよびキュー表の名前は、24 バイト以内で設定します。

タイプ名

type_name データ構造は、キュー・タイプを定義します。

構文

```
type_name := VARCHAR2;
type_name := <object_type> | "RAW";
```

属性

表 5-3 タイプ名の属性

属性	説明
<object_types>	オブジェクト・タイプ内の属性の最大数は 900 です。
"RAW"	RAW タイプのペイロードを格納するために、AQ は、ペイロード・リポジトリとして LOB 列を含むキュー表を作成します。メッセージ・ペイロードの理論上の最大サイズは、LOB 列に格納可能な最大データ量です。ただし、ペイロードの最大サイズは、AQ にアクセスするときに使用するプログラム環境によって決まります。PL/SQL、Java およびプリコンパイラの場合は 32K、OCI の場合は 4G です。PL/SQL のエンキューおよびデキューのインタフェースは、RAW バッファをペイロード・パラメータとして受け入れるため、32KB に制限されます。OCI では、ユーザーの RAW データの最大サイズは、OCI オブジェクト・キャッシュによる割当てが可能な最大連続メモリー量（OCIRaw が単純にバイト配列の場合）に制限されます。通常は、最低 32KB ですが、ほとんどの場合それ以上になります。 LOB 列は RAW ペイロードの格納に使用されるため、AQ 管理者は、LOB 表領域を選択して、キュー表作成時に storage_clause パラメータに LOB 記憶域文字列を設定すると、LOB 記憶域を構成できます。

AQ PL/SQL コールバック

plsqlcallback データ構造は、メッセージ通知時に起動されるようにデータベースで定義することで、ユーザー定義の PL/SQL プロシージャを指定します。

構文

RAW ペイロード・エンキューの通知メッセージが要求されている場合、PL/SQL コールバックには次の署名が付加されている必要があります。

```
procedure plsqlcallback(
  context  IN  RAW,
  reginfo  IN  SYS.AQ$_REG_INFO,
  descr    IN  SYS.AQ$_DESCRIPTOR,
  payload  IN  RAW,
  payloadl IN  NUMBER);
```

属性

表 5-4 AQ PL/SQL コールバックの属性

属性	説明
context	dbms_aq.register により渡されたコールバック関数のコンテキストを指定します。106-5 ページの「AQ$_REG_INFO タイプ」を参照してください。
reginfo	106-5 ページの「AQ$_REG_INFO タイプ」を参照してください。
descr	106-3 ページの「AQ$_DESCRIPTOR タイプ」を参照してください。
payload	RAW ペイロード・エンキューの通知メッセージが要求されている場合、非永続キューにエンキューされた RAW ペイロードが含まれます。 RAW ペイロードが含まれる永続キューの場合、パラメータは NULL です。
payloadl	payload の長さを指定します。payload が NULL の場合、payload1 = 0 です。

ユーザー定義型のペイロード・エンキューの通知メッセージが要求されている場合、PL/SQL コールバックには次の署名が付加されている必要があります。

```
procedure plsqlcallback(
  context  IN RAW,
  reginfo  IN SYS.AQ$_REG_INFO,
  descr    IN SYS.AQ$_DESCRIPTOR,
  payload  IN VARCHAR2,
  payloadl IN NUMBER);
```

DBMS_AQ サブプログラムの要約

表 5-5　DBMS_AQ パッケージのサブプログラム

サブプログラム	説明
「ENQUEUE プロシージャ」 5-7 ページ	指定したキューにメッセージを追加します。
「DEQUEUE プロシージャ」 5-9 ページ	指定したキューからメッセージをデキューします。
「LISTEN プロシージャ」 5-12 ページ	エージェントのリストにかわって 1 つ以上のキューをリスニングします。
「REGISTER プロシージャ」 5-13 ページ	メッセージ通知を登録します。
「UNREGISTER プロシージャ」 5-14 ページ	通知をオフにするサブスクリプションを登録解除します。
「POST プロシージャ」 5-15 ページ	匿名サブスクリプションに転送します。サブスクリプションに登録された全クライアントが通知を受信できます。
「BIND_AGENT プロシージャ」 5-15 ページ	LDAP ディレクトリに AQ エージェントのエントリを作成します。
「UNBIND_AGENT プロシージャ」 5-16 ページ	LDAP ディレクトリから AQ エージェントのエントリを削除します。

注意： DBMS_AQ パッケージには、純正レベルが定義されていません。したがって、RNDS、WNDS、RNPS または WNPS 制約が定義されている他のプロシージャからこのパッケージ内のプロシージャをコールできません。

ENQUEUE プロシージャ

このプロシージャは、指定したキューにメッセージを追加します。

構文

```
DBMS_AQ.ENQUEUE (
   queue_name           IN      VARCHAR2,
   enqueue_options      IN      enqueue_options_t,
   message_properties   IN      message_properties_t,
   payload              IN      "<type_name>",
   msgid                OUT     RAW);
```

パラメータ

表 5-6 ENQUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	このメッセージをエンキューするキュー名を指定します。例外キューは指定できません。
enqueue_options	106-11 ページの「ENQUEUE_OPTIONS_T タイプ」を参照してください。
message_properties	106-12 ページの「MESSAGE_PROPERTIES_T タイプ」および 5-8 ページの「保護キューの使用方法」を参照してください。
payload	Oracle AQ では解釈されません。 ペイロードは、関連するキュー表の仕様に基づいて指定する必要があります。NULL は受け入れ可能なパラメータの 1 つです。 <type_name> の定義の詳細は、5-3 ページの「タイプ名」を参照してください。
msgid	システムが生成するメッセージ ID。 これは、デキュー時にメッセージを識別するために使用するグローバルな一意の ID です。

使用上の注意

enqueue_options の sequence_deviation パラメータを使用すると、2 つのメッセージ間の処理順序を変更できます。参照されるメッセージの ID は、enqueue_options のパラメータ relative_msgid で指定できます。関係は、sequence_deviation パラメータによって識別されます。

メッセージに sequence_deviation を指定すると、このメッセージに指定できる遅延および優先の順位値が一部制限されます。遅延の値は、このメッセージより後にエンキューされるメッセージの遅延以下に設定する必要があります。優先順位は、このメッセージより後にエンキューされるメッセージの優先順位以上に設定する必要があります。

メッセージが受信者のいないマルチ・コンシューマ・キューにエンキューされ、かつそのキューにサブスクライバが存在しない（またはこのメッセージと一致するルールベースのサブスクライバが存在しない）場合は、Oracle エラー ORA-24033 が発生します。これは、配信可能な受信者またはサブスクライバが存在しないために、そのメッセージが廃棄されることを示す警告です。

保護キューの使用方法

保護キューの場合は、messages_properties パラメータに sender_id を指定する必要があります。sender_id の詳細は、106-12 ページの「MESSAGE_PROPERTIES_T タイプ」を参照してください。

保護キューを使用するときは、次の事項を満たしている必要があります。

- DBMS_AQADM.CREATE_AQ_AGENT を使用して、有効な AQ エージェントを作成している必要があります。6-30 ページの「CREATE_AQ_AGENT プロシージャ」を参照してください。
- 保護キューに対するエンキュー権限のあるデータベース・ユーザーに sender_id をマップする必要があります。これは、DBMS_AQADM.ENABLE_DB_ACCESS を使用して行います。6-33 ページの「ENABLE_DB_ACCESS プロシージャ」を参照してください。

関連項目： 保護キューの詳細は、『Oracle9*i* Streams』を参照してください。

DEQUEUE プロシージャ

このプロシージャは、指定したキューからメッセージをデキューします。

構文

```
DBMS_AQ.DEQUEUE (
   queue_name           IN      VARCHAR2,
   dequeue_options      IN      dequeue_options_t,
   message_properties   OUT     message_properties_t,
   payload              OUT     "<type_name>",
   msgid                OUT     RAW);
```

パラメータ

表 5-7 DEQUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	キュー名を指定します。
dequeue_options	106-8 ページの「DEQUEUE_OPTIONS_T タイプ」および 5-11 ページの「保護キューの使用方法」を参照してください。
message_properties	106-12 ページの「MESSAGE_PROPERTIES_T タイプ」を参照してください。
payload	Oracle AQ では解釈されません。ペイロードは、関連するキュー表の仕様に基づいて指定する必要があります。 <type_name> の定義の詳細は、5-3 ページの「タイプ名」を参照してください。
msgid	システムが生成するメッセージ ID。

使用上の注意

デキューされるメッセージの検索基準は、dequeue_options の consumer_name、msgid、correlation および deq_condition パラメータにより判断されます。

- msgid はデキューされるメッセージを一意に識別します。
- 相関識別子は、AQ によって解釈されないアプリケーション定義の識別子です。
- デキュー条件は、メッセージ・プロパティ、メッセージ・データ・プロパティおよび PL/SQL ファンクションに基づいた式です。deq_condition は SQL 問合せの WHERE 句と同様の構文を使用して、ブール式で指定されます。このブール式には、メッセージ・プロパティの状態、ユーザー・データのプロパティ（オブジェクト・ペイロードのみ）、および PL/SQL または SQL ファンクション（SQL 問合せの WHERE 句で指定）の状態を組み込むことができます。メッセージ・プロパティには、priority、corrid およびキュー表におけるその他の列が含まれます。

  メッセージ・ペイロード（オブジェクト・ペイロード）のデキュー条件を指定するには、句にオブジェクト・タイプの属性を使用します。各属性の前に修飾子として tab.user_data を付加して、ペイロードを格納するキュー表の特定の列を示します。

  例：tab.user_data.orderstatus='EXPRESS''

msgid が指定されていないかぎり、READY 状態のメッセージのみがデキューされます。

デキュー順序は、dequeue_options の msgid と相関 ID で上書きされないかぎり、キュー表の作成時に指定した値によって判断されます。

キュー操作には、データベース読込み一貫性メカニズムが適用されます。たとえば、BROWSE コールは、ブラウズ・トランザクションの開始後にエンキューされたメッセージを参照しない場合があります。

デキュー時のデフォルトの NAVIGATION パラメータは、NEXT_MESSAGE です。この場合、後続のデキューは、最初のデキューで取得したスナップショットに基づいて、キューからメッセージを取り出します。特に、最初のデキュー・コマンド後にエンキューされたメッセージは、キュー内の残りのメッセージがすべて処理されるまで処理されません。これは、すべてのメッセージがすでにキューにエンキューされている場合、またはキューに優先順位が設定されていない場合は、通常問題ありません。ただし、デキュー・コマンドのたびにキュー内の先頭メッセージを処理する必要があるときには、アプリケーションで FIRST_MESSAGE ナビゲーション・オプションを使用する必要があります。この処理は、すでにエンキューされたメッセージの処理中に優先順位の高いメッセージがキューに登録された場合に必要になります。

注意： 同時にエンキューされている複数のメッセージがある場合は、FIRST_MESSAGE ナビゲーション・オプションを使用すると効率が向上します。FIRST_MESSAGE オプションが指定されていない場合、AQ は最初のデキュー・コマンド時のスナップショットを生成し続けるため、パフォーマンスが低下します。FIRST_MESSAGE オプションが指定されている場合、AQ はすべてのデキュー・コマンドに対して新しいスナップショットを使用します。

同一トランザクションでメッセージのグループ化に対応しているキューにエンキューされたメッセージは、グループを形成します。そのトランザクションでエンキューされたメッセージが 1 つのみの場合は、実質的に 1 つのメッセージでグループを形成します。1 つのトランザクションでグループ化できるメッセージの数に上限はありません。

メッセージのグループ化に対応していないキューでは、LOCKED または REMOVE モードのデキューは、1 つのメッセージのみロックします。これに対して、グループの一部のメッセージをデキューしようとするデキュー操作は、グループ全体をロックします。これは、グループ内のすべてのメッセージを基本単位で処理する必要がある場合に有効です。

グループ内のすべてのメッセージがデキューされている場合、そのデキューはグループ内のすべてのメッセージが処理済みであることを示すエラーを戻します。この場合、アプリケーションは NEXT_TRANSACTION を使用して、次に使用可能なグループからメッセージのデキューを開始します。使用可能なグループがない場合、デキューは指定した WAIT 期間後にタイムアウトします。

保護キューの使用方法

保護キューの場合は、dequeue_options パラメータに consumer_name を指定する必要があります。consumer_name の詳細は、106-8 ページの「DEQUEUE_OPTIONS_T タイプ」を参照してください。

保護キューを使用するときは、次の事項を満たしている必要があります。

- DBMS_AQADM.CREATE_AQ_AGENT を使用して、有効な AQ エージェントを作成している必要があります。6-30 ページの「CREATE_AQ_AGENT プロシージャ」を参照してください。
- 保護キューに対するデキュー権限のあるデータベース・ユーザーに AQ エージェントをマップする必要があります。これは、DBMS_AQADM.ENABLE_DB_ACCESS を使用して行います。6-33 ページの「ENABLE_DB_ACCESS プロシージャ」を参照してください。

関連項目： 保護キューの詳細は、『Oracle9*i* Streams』を参照してください。

LISTEN プロシージャ

このプロシージャは、エージェントのリストのかわりに 1 つ以上のキューをリスニングします。エージェントのアドレス・フィールドは、エージェントが監視するキューを示します。ローカル・キューのみアドレスとしてサポートされています。将来使用するためにプロトコルが予約されています。

エージェント・アドレスがマルチ・コンシューマ・キューである場合、エージェント名は必須項目です。単一コンシューマ・キューの場合は、エージェント名を指定する必要はありません。

これは、リスト内のエージェントに対してコンシューマ用のメッセージが準備されているときに戻されるブロック・コールです。待機時間が期限切れしてもメッセージが見つからない場合は、エラーが発生します。

構文

```
DBMS_AQ.LISTEN (
   agent_list IN    aq$_agent_list_t,
   wait       IN    BINARY_INTEGER DEFAULT DBMS_AQ.FOREVER,
   agent      OUT   sys.aq$_agent);

TYPE aq$_agent_list_t IS TABLE of aq$_agent INDEXED BY BINARY_INTEGER;
```

パラメータ

表 5-8 LISTEN プロシージャのパラメータ

パラメータ	説明
agent_list	リスニングするエージェントのリスト。
wait	リスニング・コールのタイムアウト（秒数）。デフォルトでは、コールは永続的にブロックします。
agent	コンシューマ用のメッセージがあるエージェント。

使用上の注意

このプロシージャは、引数としてエージェントのリストを使用します。リストされた各エージェントのアドレス・フィールドに、監視するキューを指定します。マルチ・コンシューマ・キューを監視するときは、エージェントの名前も指定する必要があります。単一コンシューマ・キューの場合は、エージェント名を指定する必要はありません。ローカル・キューのみアドレスとしてサポートされています。将来使用するためにプロトコルが予約されています。

これは、リスト内のエージェントに対してコンシューマ用のメッセージが準備されているときに戻されるブロック・コールです。複数のエージェントに対するメッセージがある場合は、リストの最初のエージェントのみ戻されます。待機時間が期限切れしてもメッセージが見つからない場合は、エラーが発生します。

リスニング・コールからの正常な戻りは、指定したキューの中に、リストされたエージェントの 1 つに対するメッセージがあることを示しているにすぎません。対象となっているエージェントは、関連メッセージを継続してデキューする必要があります。

非永続キューでは、リスニングはコールできない点に注意してください。

REGISTER プロシージャ

このプロシージャにより、電子メール・アドレス、ユーザー定義の PL/SQL プロシージャまたはメッセージ通知用の HTTP URL を登録します。

構文

```
DBMS_AQ.REGISTER (
   reg_list IN SYS.AQ$_REG_INFO_LIST,
   count    IN NUMBER);
```

パラメータ

表 5-9 REGISTER プロシージャのパラメータ

パラメータ	説明
reg_list	メッセージ通知用に登録するサブスクリプションのリストを指定します。AQ$_REG_INFO タイプのリストです。
count	reg_list のエントリ数を指定します。

使用上の注意

このプロシージャは、通知の登録に使用します。メッセージ通知の送信先電子メール・アドレスの指定、通知時に起動するプロシージャの登録または通知転送先の HTTP URL の登録ができます。複数のサブスクリプションを一度に登録することもできます。

電子メール通知を登録する場合、データベースが電子メール通知の送信に使用する SMTP サーバーのホスト名およびポート名を設定する必要があります。必要であれば、発信元電子メール・アドレスを設定します。これは、データベースの `sent from` フィールドで設定します。電子メール通知の詳細は、第 7 章「DBMS_AQELM」を参照してください。この機能を使用するには、Java 対応のデータベースが必要です。

HTTP 通知を登録する場合、データベースが HTTP 通知の転送に使用するプロキシ・サーバーおよび非プロキシ・ドメイン・リストのホスト名およびポート名を設定する必要があります。HTTP 通知の詳細は、第 7 章「DBMS_AQELM」を参照してください。

UNREGISTER プロシージャ

このプロシージャでは、通知をオフにするサブスクリプションを登録解除します。

構文

```
DBMS_AQ.UNREGISTER (
   reg_list IN SYS.AQ$_REG_INFO_LIST,
   count    IN NUMBER);
```

パラメータ

表 5-10 UNREGISTER プロシージャのパラメータ

パラメータ	説明
reg_list	メッセージ通知用に登録するサブスクリプションのリストを指定します。AQ$_REG_INFO タイプのリストです。
count	reg_list のエントリ数を指定します。

使用上の注意

このプロシージャにより、通知をオフにするサブスクリプションを登録解除します。複数のサブスクリプションを一度に登録解除することもできます。

POST プロシージャ

このプロシージャにより、匿名サブスクリプションのリストに転送します。匿名サブスクリプションの場合、サブスクリプションに登録された全クライアントが通知を受信できます。

構文

```
DBMS_AQ.POST (
  post_list  IN  SYS.AQ$_POST_INFO_LIST,
  count      IN NUMBER);
```

パラメータ

表 5-11 POST プロシージャのパラメータ

パラメータ	説明
post_list	転送する匿名サブスクリプションのリストを指定します。AQ$_POST_INFO タイプのリストです。
count	post_list のエントリ数を指定します。

使用上の注意

このプロシージャにより、匿名サブスクリプションに転送します。匿名サブスクリプションの場合、サブスクリプションに登録された全クライアントが通知を受信できます。複数のサブスクリプションを一度に転送することもできます。

BIND_AGENT プロシージャ

このプロシージャにより、LDAP サーバーに AQ エージェントのエントリを作成します。

構文

```
DBMS_AQ.BIND_AGENT(
   agent        IN SYS.AQ$_AGENT,
   certificate  IN VARCHAR2 default NULL);
```

パラメータ

表 5-12 BIND_AGENT プロシージャのパラメータ

パラメータ	説明
agent	LDAP サーバーで登録されるエージェント。
certificate	このエージェントでデジタル証明書（usercertificate 属性）が使用される LDAP における、organizationalperson エントリの場所（LDAP の識別名）。 例 : cn=OE, cn=ACME, cn=com は、指定したエージェントで証明書が使用される OrganizationalPerson OE の DN です。

使用上の注意

LDAP サーバーでは、デジタル証明書が OrganizationalPerson エンティティの属性（usercertificate）として格納されます。この OrganizationalPerson の識別名をエージェントのバインド時に指定する必要があります。

UNBIND_AGENT プロシージャ

このプロシージャにより、LDAP サーバーから AQ エージェントのエントリを削除します。

構文

```
DBMS_AQ.UNBIND_AGENT(
   agent     IN SYS.AQ$_AGENT);
```

パラメータ

表 5-13 BIND_AGENT プロシージャのパラメータ

パラメータ	説明
agent	LDAP サーバーから削除されるエージェント。

6

DBMS_AQADM

DBMS_AQADM パッケージは、アドバンスト・キューイングの構成および管理情報を管理するプロシージャを提供します。

関連項目：

- 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』
- DBMS_AQADM で使用する TYPE に関する情報は、第 106 章「アドバンスト・キューイング・タイプ」を参照してください。

この章では、次の項目について説明します。

列挙定数

INFINITE、TRANSACTIONAL または NORMAL_QUEUE などの列挙定数を使用するときは、それを定義するパッケージの範囲内で、その記号を指定する必要があります。管理インタフェースに関連付けられているすべてのタイプに、DBMS_AQADM を付加する必要があります。たとえば、DBMS_AQADM.NORMAL_QUEUE のようにします。

表 6-1 管理インタフェース内の列挙タイプ

パラメータ	オプション
retention	0、1、2...INFINITE
message_grouping	TRANSACTIONAL、NONE
queue_type	NORMAL_QUEUE、EXCEPTION_QUEUE、NON_PERSISTENT_QUEUE

関連項目： DBMS_AQ および DBMS_AQADM で使用される Java クラスとデータ構造の詳細は、第 5 章「DBMS_AQ」を参照してください。

DBMS_AQADM サブプログラムの要約

表 6-2 DBMS_AQADM パッケージのサブプログラム

サブプログラム	説明
「CREATE_QUEUE_TABLE プロシージャ」6-4 ページ	事前定義タイプのメッセージのキュー表を作成します。
「ALTER_QUEUE_TABLE プロシージャ」6-8 ページ	既存のキュー表を変更します。
「DROP_QUEUE_TABLE プロシージャ」6-9 ページ	既存のキュー表を削除します。
「CREATE_QUEUE プロシージャ」6-10 ページ	指定したキュー表にキューを作成します。
「CREATE_NP_QUEUE プロシージャ」6-12 ページ	非永続の RAW キューを作成します。
「ALTER_QUEUE プロシージャ」6-13 ページ	キューの既存のプロパティを変更します。
「DROP_QUEUE プロシージャ」6-14 ページ	既存のキューを削除します。

表 6-2 DBMS_AQADM パッケージのサブプログラム（続き）

サブプログラム	説明
「START_QUEUE プロシージャ」6-15 ページ	指定したキューに対するエンキューまたはデキューを使用可能にします。
「STOP_QUEUE プロシージャ」6-16 ページ	指定したキューに対するエンキューまたはデキューを使用禁止にします。
「GRANT_SYSTEM_PRIVILEGE プロシージャ」6-17 ページ	ユーザーおよびロールに AQ システム権限を付与します。
「REVOKE_SYSTEM_PRIVILEGE プロシージャ」6-18 ページ	ユーザーおよびロールから AQ システム権限を取り消します。
「GRANT_QUEUE_PRIVILEGE プロシージャ」6-18 ページ	ユーザーおよびロールにキューの権限を付与します。
「REVOKE_QUEUE_PRIVILEGE プロシージャ」6-19 ページ	ユーザーおよびロールからキューの権限を取り消します。
「ADD_SUBSCRIBER プロシージャ」6-20 ページ	キューにデフォルトのサブスクライバを追加します。
「ALTER_SUBSCRIBER プロシージャ」6-22 ページ	指定したキューに対するサブスクライバの既存プロパティを変更します。
「REMOVE_SUBSCRIBER プロシージャ」6-22 ページ	キューからデフォルトのサブスクライバを削除します。
「SCHEDULE_PROPAGATION プロシージャ」6-23 ページ	キューから特定の DB リンクで指定された宛先へのメッセージ伝播をスケジュールします。
「UNSCHEDULE_PROPAGATION プロシージャ」6-25 ページ	キューから、特定の DB リンクで指定された宛先へのメッセージ伝播の旧スケジュールを取り消します。
「VERIFY_QUEUE_TYPES プロシージャ」6-26 ページ	ソース・キューおよび宛先キューのタイプが同じであるかどうかを検証します。
「ALTER_PROPAGATION_SCHEDULE プロシージャ」6-27 ページ	伝播スケジュールのパラメータを変更します。
「ENABLE_PROPAGATION_SCHEDULE プロシージャ」6-28 ページ	以前使用禁止にした伝播スケジュールを使用可能にします。
「DISABLE_PROPAGATION_SCHEDULE プロシージャ」6-29 ページ	伝播スケジュールを使用禁止にします。

表 6-2 DBMS_AQADM パッケージのサブプログラム（続き）

サブプログラム	説明
「MIGRATE_QUEUE_TABLE プロシージャ」6-30 ページ	8.0 互換キュー表から 8.1 互換キュー表へのアップグレード、または 8.1 互換キュー表から 8.0 互換キュー表へのダウングレードを実行します。
「CREATE_AQ_AGENT プロシージャ」6-30 ページ	AQ インターネット・アクセスのエージェントを登録します。
「ALTER_AQ_AGENT プロシージャ」6-32 ページ	AQ インターネット・アクセスに登録されたエージェントを変更します。
「DROP_AQ_AGENT プロシージャ」6-33 ページ	AQ インターネット・アクセスに登録されたエージェントを削除します。
「ENABLE_DB_ACCESS プロシージャ」6-33 ページ	AQ インターネット・エージェントに特定のデータベース・ユーザーとしての権限を付与します。
「DISABLE_DB_ACCESS プロシージャ」6-34 ページ	AQ インターネット・エージェントのデータベース・ユーザーとしての権限を取り消します。
「ADD_ALIAS_TO_LDAP プロシージャ」6-35 ページ	LDAP のキュー、エージェントまたは JMS ConnectionFactory の別名を作成します。
「DEL_ALIAS_FROM_LDAP プロシージャ」6-36 ページ	LDAP のキュー、エージェントまたは JMS ConnectionFactory の別名を削除します。

CREATE_QUEUE_TABLE プロシージャ

このプロシージャは、事前定義タイプのメッセージ用のキュー表を作成します。デキュー順序のソート・キーがある場合は、表作成時に定義する必要があります。このときに次のオブジェクトが作成されます。

- キュー表に関連付けられているデフォルトの例外キュー。`aq$_<queue_table_name>_e` という名前になります。
- AQ アプリケーションがキュー・データの問合せに使用する読取り専用ビュー。`aq$<queue_table_name>` という名前になります。
- 索引、またはキュー・モニター操作用の複数コンシューマ・キューの場合の索引構成表（IOT）。`aq$_<queue_table_name>_t` という名前になります。
- 索引、またはデキュー操作用の複数コンシューマ・キューの場合の索引構成表。`aq$_<queue_table_name>_i` という名前になります。

Oracle8*i* 互換のキュー表の場合は、次の索引構成表が作成されます。

- aq$_<queue_table_name>_s という名前の表。この表には、サブスクライバに関する情報が格納されます。
- aq$_<queue_table_name>_r という名前の表。この表には、サブスクライバのルールに関する情報が格納されます。
- aq$_<queue_table_name>_h という名前の索引構成表。この表には、デキュー履歴データが格納されます。

構文

```
DBMS_AQADM.CREATE_QUEUE_TABLE (
   queue_table          IN      VARCHAR2,
   queue_payload_type   IN      VARCHAR2,
   storage_clause       IN      VARCHAR2        DEFAULT NULL,
   sort_list            IN      VARCHAR2        DEFAULT NULL,
   multiple_consumers   IN      BOOLEAN         DEFAULT FALSE,
   message_grouping     IN      BINARY_INTEGER DEFAULT NONE,
   comment              IN      VARCHAR2        DEFAULT NULL,
   auto_commit          IN      BOOLEAN         DEFAULT TRUE,
   primary_instance     IN      BINARY_INTEGER DEFAULT 0,
   secondary_instance   IN      BINARY_INTEGER DEFAULT 0,
   compatible           IN      VARCHAR2        DEFAULT NULL);
```

パラメータ

表 6-3 CREATE_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
queue_table	作成するキュー表の名前。
queue_payload_type	格納されるユーザー・データのタイプ。このパラメータの有効値については、第 5 章「DBMS_AQ」の「タイプ名」を参照してください。

表 6-3 CREATE_QUEUE_TABLE プロシージャのパラメータ（続き）

パラメータ	説明
storage_clause	記憶領域パラメータ。 記憶領域パラメータは、キュー表の作成時に、CREATE TABLE 文に組み込まれます。記憶領域パラメータは、次のパラメータの任意の組合せで作成できます。PCTFREE、PCTUSED、INITRANS、MAXTRANS、TABLEPSACE、LOB およびテーブルの記憶域句。 ここで表領域が指定されない場合は、キュー表とそのすべての関連オブジェクトが、デフォルトのユーザー表領域に作成されます。ここで表領域が指定されると、キュー表およびそのすべての関連オブジェクトは、テーブルの記憶域句で指定された表領域に作成されます。これらのパラメータの使用方法については、『Oracle9*i* SQL リファレンス』を参照してください。
sort_list	昇順ソート・キーに使用する列。 Sort_list の書式は次のとおりです。 '<sort_column_1>,<sort_column_2>' 許可される列名は、priority および enq_time です。両方の列を指定する場合は、<sort_column_1> に優先する順序を定義します。 特定の順序付けメカニズムでキュー表が作成されると、キュー表内のすべてのキューが同じデフォルトを使用します。キュー表の順序は、キュー表の作成後には変更できません。 ソート・リストが指定されていない場合、このキュー表内のキューはすべてエンキュー時に昇順ソートされます。この順序は FIFO 順と同じです。 デフォルトの順序が定義されている場合でも、msgid、correlation.msgid または correlation を指定して、デキューするメッセージを選択できます。sequence_deviation が指定されている場合は、デフォルトより優先されます。
multiple_consumers	FALSE: 表内で作成されたキューには、各メッセージに対して 1 つのコンシューマのみが設定できます。これがデフォルトです。 TRUE: 表内で作成されたキューには、各メッセージに対して複数のコンシューマを設定できます。
message_grouping	表内で作成されたキューのメッセージ・グループ化に関する動作。 NONE: 各メッセージは個々に処理されます。 TRANSACTIONAL: あるトランザクションの一部としてエンキューされた複数のメッセージは、同じグループの一部とみなされ、関連するメッセージのグループとしてデキューできます。
comment	キュー表に関するユーザー指定の説明。このユーザー・コメントは、キュー・カタログに追加されます。

表 6-3 CREATE_QUEUE_TABLE プロシージャのパラメータ（続き）

パラメータ	説明
auto_commit	TRUE: 現行トランザクションがある場合は、CREATE_QUEUE_TABLE 操作が実行される前にコミットされます。CREATE_QUEUE_TABLE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行のトランザクションの一部で、コール元がコミットを入力したときのみ持続されます。 **注意 :** このパラメータは使用しないことをお薦めします。
primary_instance	キュー表のプライマリ所有者。キュー表内のキューに対するキュー・モニターのスケジューリングおよび伝播は、このインスタンス内で行われます。 プライマリ・インスタンスのデフォルト値は 0（ゼロ）で、キュー・モニターのスケジューリングおよび伝播は、使用可能なすべてのインスタンス内で行われます。
secondary_instance	プライマリ・インスタンスが使用不可の場合、キュー表はセカンダリ・インスタンスにフェイルオーバーします。デフォルト値は 0（ゼロ）で、キュー表は使用可能なすべてのインスタンスにフェイルオーバーします。
compatible	キューの互換性があるデータベースの最下位バージョン。現在使用可能な値は、'8.0' または '8.1' です。 ■ データベースが 8.1 以上との互換モードである場合、デフォルト値は '8.1' です。 ■ データベースが 8.0 互換モードである場合、デフォルト値は '8.0' です。

使用上の注意

CLOB、BLOB および BFILE は、AQ オブジェクト・タイプのペイロードに対する有効な属性です。ただし、Oracle8*i* リリース 8.1.6 以上で AQ を使用して伝播できるのは、CLOB および BLOB のみです。詳細は、『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。

互換パラメータのデフォルト値は、init.ora のデータベース互換モードにより異なります。

- データベースが 8.1 以上との互換モードである場合、デフォルト値は 8.1 です。
- データベースが 8.0 互換モードである場合、デフォルト値は 8.0 です。

primary_instance と secondary_instance は、8.1 互換モードでのみ指定および変更できます。

プライマリ・インスタンスが存在しないと、セカンダリ・インスタンスは指定できません。

ALTER_QUEUE_TABLE プロシージャ

このプロシージャは、キュー表の既存のプロパティを変更します。

構文

```
DBMS_AQADM.ALTER_QUEUE_TABLE (
   queue_table          IN   VARCHAR2,
   comment              IN   VARCHAR2       DEFAULT NULL,
   primary_instance     IN   BINARY_INTEGER DEFAULT NULL,
   secondary_instance   IN   BINARY_INTEGER DEFAULT NULL);
```

パラメータ

表 6-4 ALTER_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
queue_table	作成するキュー表の名前。
comment	キュー表に関するユーザー指定の説明を変更します。このユーザー・コメントは、キュー・カタログに追加されます。デフォルト値は NULL で、値が変更されないことを意味します。
primary_instance	キュー表のプライマリ所有者。キュー表内のキューに対するキュー・モニターのスケジューリングおよび伝播は、このインスタンス内で行われます。 デフォルト値は NULL で、現行の値が変更されないことを意味します。
secondary_instance	プライマリ・インスタンスが使用不可の場合、キュー表はセカンダリ・インスタンスにフェイルオーバーします。 デフォルト値は NULL で、現行の値が変更されないことを意味します。

DROP_QUEUE_TABLE プロシージャ

このプロシージャは、既存のキュー表を削除します。キュー表を削除するには、キュー表内のすべてのキューを停止し、削除しておく必要があります。この処理を自動的に実行する force オプションを使用しない場合は、明示的に実行する必要があります。

構文

```
DBMS_AQADM.DROP_QUEUE_TABLE (
   queue_table        IN    VARCHAR2,
   force              IN    BOOLEAN DEFAULT FALSE,
   auto_commit        IN    BOOLEAN DEFAULT TRUE);
```

パラメータ

表 6-5 DROP_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
queue_table	削除するキュー表の名前。
force	FALSE: 表内にキューが存在する場合、操作は成功しません。これがデフォルトです。 TRUE: 表内のすべてのキューが自動的に停止および削除されます。
auto_commit	TRUE: 現行のトランザクションがある場合は、DROP_QUEUE_TABLE 操作が実行される前にコミットされます。DROP_QUEUE_TABLE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行のトランザクションの一部で、コール元がコミットを入力したときのみ持続されます。 **注意 :** このパラメータは使用しないことをお薦めします。

CREATE_QUEUE プロシージャ

このプロシージャは、指定したキュー表にキューを作成します。

構文

```
DBMS_AQADM.CREATE_QUEUE (
   queue_name          IN       VARCHAR2,
   queue_table         IN       VARCHAR2,
   queue_type          IN       BINARY_INTEGER DEFAULT NORMAL_QUEUE,
   max_retries         IN       NUMBER         DEFAULT NULL,
   retry_delay         IN       NUMBER         DEFAULT 0,
   retention_time      IN       NUMBER         DEFAULT 0,
   dependency_tracking IN       BOOLEAN        DEFAULT FALSE,
   comment             IN       VARCHAR2       DEFAULT NULL,
   auto_commit         IN       BOOLEAN        DEFAULT TRUE);
```

パラメータ

表 6-6 CREATE_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	作成するキューの名前。名前はスキーマ内で重複しないようにし、予約語については、『Oracle9*i* SQL リファレンス』のオブジェクト名のガイドラインに従う必要があります。
queue_table	キューを格納するキュー表の名前。
queue_type	作成されるキューが例外キューか標準キューかを指定します。 NORMAL_QUEUE: キューは標準キューです。これがデフォルトです。 EXCEPTION_QUEUE: キューは例外キューです。例外キューでは、デキュー操作のみ許可されます。
max_retries	REMOVE モードのデキューがメッセージ上で試行される回数を制限します。max_retries の最大値は 2**31 -1 です。 デキュー実行後、アプリケーションがロールバックを発行するたびにカウントが増加します。指定した max_retries に達すると、メッセージは例外キューに移されます。 max_retries はすべての単一コンシューマ・キューおよび 8.1 互換の複数コンシューマ・キューでサポートされていますが、8.0 互換の複数コンシューマ・キューではサポートされていないことに注意してください。

表 6-6 CREATE_QUEUE プロシージャのパラメータ（続き）

パラメータ	説明
retry_delay	アプリケーションのロールバック後、このメッセージの再処理をスケジュールするまでの遅延時間（秒数）。 デフォルトは 0（ゼロ）で、メッセージを最も迅速に取り出せます。このパラメータは、max_retries が 0（ゼロ）に設定されている場合は無効です。retry_delay は、単一コンシューマ・キューおよび 8.1 互換の複数コンシューマ・キューでサポートされていますが、8.0 互換の複数コンシューマ・キューではサポートされていないことに注意してください。
retention_time	キューからデキューされた後に、メッセージがキュー表に保持される秒数。 INFINITE: メッセージは無期限で保持されます。 数値 : メッセージを保持する秒数。デフォルトは 0（ゼロ）で、保持されません。
dependency_tracking	将来の使用のために確保。 FALSE: これがデフォルトです。 TRUE: このリリースでは指定できません。
comment	キューに関するユーザー指定の説明。このユーザー・コメントは、キュー・カタログに追加されます。
auto_commit	TRUE: 現行のトランザクションがある場合は、CREATE_QUEUE 操作が実行される前にコミットされます。CREATE_QUEUE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行のトランザクションの一部で、コール元がコミットを入力したときのみ持続されます。 **注意 :** このパラメータは使用しないことをお薦めします。

使用上の注意

すべてのキュー名はスキーマ内で重複しないようにしてください。CREATE_QUEUE でキューが作成された後、START_QUEUE をコールするとキューが使用可能になります。デフォルトでは、キューはエンキューおよびデキューともに使用禁止で作成されます。

CREATE_NP_QUEUE プロシージャ

非永続の RAW キューを作成します。

構文

```
DBMS_AQADM.CREATE_NP_QUEUE (
   queue_name              IN        VARCHAR2,
   multiple_consumers      IN        BOOLEAN  DEFAULT FALSE,
   comment                 IN        VARCHAR2 DEFAULT NULL);
```

パラメータ

表 6-7 CREATE_NP_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	作成する非永続キューの名前。名前はスキーマ内で重複しないようにし、『Oracle9*i* SQL リファレンス』のオブジェクト名のガイドラインに従う必要があります。
multiple_consumers	FALSE: 表内で作成されたキューには、各メッセージに対して 1 つのコンシューマのみが設定できます。これがデフォルトです。 TRUE: 表内で作成されたキューには、各メッセージに対して複数のコンシューマを設定できます。 非永続キューは、ユーザーが作成したキュー表からこの特性を継承しないため、このパラメータはキュー・レベルで識別されることに注意してください。
comment	キューに関するユーザー指定の説明。このユーザー・コメントは、キュー・カタログに追加されます。

使用上の注意

キューは、単一コンシューマ・キューまたはマルチ・コンシューマ・キューのいずれかです。すべてのキュー名はスキーマ内で重複しないようにしてください。キューは、キュー名が指定している同じスキーマ内の 8.1 互換のシステム作成キュー表（AQ$_MEM_SC または AQ$_MEM_MC）に作成されます。

キュー名がスキーマ名を指定していない場合、キューはログイン・ユーザーのスキーマ内に作成されます。CREATE_NP_QUEUE でキューが作成された後、START_QUEUE をコールするとそのキューが使用可能になります。デフォルトでは、キューはエンキューおよびデキューともに使用禁止で作成されます。

非永続キューからはデキューできません。非永続キューからメッセージを取り出す唯一の方法は、OCI 通知メカニズムを使用することです。非永続キューでは、listen コールは起動できません。

ALTER_QUEUE プロシージャ

このプロシージャは、キューの既存プロパティを変更します。パラメータ max_retries、retention_time および retry_delay は、非永続キューではサポートされていません。

構文

```
DBMS_AQADM.ALTER_QUEUE (
   queue_name        IN    VARCHAR2,
   max_retries       IN    NUMBER   DEFAULT NULL,
   retry_delay       IN    NUMBER   DEFAULT NULL,
   retention_time    IN    NUMBER   DEFAULT NULL,
   auto_commit       IN    BOOLEAN  DEFAULT TRUE,
   comment           IN    VARCHAR2 DEFAULT NULL);
```

パラメータ

表 6-8 ALTER_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	変更するキューの名前。
max_retries	REMOVE モードのデキューがメッセージ上で試行される回数を制限します。max_retries の最大値は 2**31 -1 です。 デキュー実行後、アプリケーションがロールバックを発行するたびにカウントが増加します。試行回数内で期限切れとなった場合、その後の再試行は行われません。デフォルトは NULL で、値が変更されないことを意味します。 max_retries はすべての単一コンシューマ・キューおよび 8.1 互換の複数コンシューマ・キューでサポートされていますが、8.0 互換の複数コンシューマ・キューではサポートされていないことに注意してください。
retry_delay	アプリケーションのロールバック後、このメッセージの再処理をスケジュールするまでの遅延時間（秒数）。デフォルトは NULL で、値が変更されないことを意味します。 retry_delay は単一コンシューマ・キューおよび 8.1 互換の複数コンシューマ・キューでサポートされていますが、8.0 互換の複数コンシューマ・キューではサポートされていないことに注意してください。
retention_time	デキュー後、メッセージがキュー表内に保持される時間（秒数）。デフォルトは NULL で、値が変更されないことを意味します。

表 6-8 ALTER_QUEUE プロシージャのパラメータ（続き）

パラメータ	説明
auto_commit	TRUE: 現行のトランザクションがある場合は、ALTER_QUEUE 操作が実行される前にコミットされます。ALTER_QUEUE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行のトランザクションの一部で、コール元がコミットを入力したときのみ持続されます。 **注意 :** このパラメータは使用しないことをお薦めします。
comment	キューに関するユーザー指定の説明。このユーザー・コメントは、キュー・カタログに追加されます。デフォルト値は NULL で、値が変更されないことを意味します。

DROP_QUEUE プロシージャ

このプロシージャは、既存のキューを削除します。DROP_QUEUE は、STOP_QUEUE をコールして、キューに対するエンキューおよびデキューを使用禁止にしないと許可されません。すべてのキュー・データが、削除操作の一部として削除されます。

構文

```
DBMS_AQADM.DROP_QUEUE (
   queue_name         IN    VARCHAR2,
   auto_commit        IN    BOOLEAN DEFAULT TRUE);
```

パラメータ

表 6-9 DROP_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	削除するキューの名前。
auto_commit	TRUE: 現行のトランザクションがある場合は、DROP_QUEUE 操作が実行される前にコミットされます。DROP_QUEUE 操作は、コールから戻ると持続されます。これがデフォルトです。 FALSE: 操作は現行のトランザクションの一部で、コール元がコミットを入力したときのみ持続されます。 **注意 :** このパラメータは使用しないことをお薦めします。

START_QUEUE プロシージャ

このプロシージャは、指定したキューに対するエンキューまたはデキューを使用可能にします。

キューの作成後に、管理者は START_QUEUE を使用してキューを使用可能にする必要があります。デフォルトでは、ENQUEUE および DEQUEUE の両方が使用可能になります。例外キューでは、デキュー操作のみ許可されます。この操作はコールが完了すると有効になり、トランザクションの特性はありません。

構文

```
DBMS_AQADM.START_QUEUE (
   queue_name      IN     VARCHAR2,
   enqueue         IN     BOOLEAN DEFAULT TRUE,
   dequeue         IN     BOOLEAN DEFAULT TRUE);
```

パラメータ

表 6-10 START_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	使用可能にするキューの名前。
enqueue	このキューで ENQUEUE を使用可能にするかどうかを指定します。 TRUE: ENQUEUE を使用可能にします。これがデフォルトです。 FALSE: 現在の設定を変更しません。
dequeue	このキューで DEQUEUE を使用可能にするかどうかを指定します。 TRUE: DEQUEUE を使用可能にします。これがデフォルトです。 FALSE: 現在の設定を変更しません。

STOP_QUEUE プロシージャ

このプロシージャは、指定したキューに対するエンキューまたはデキューを使用禁止にします。

デフォルトでは、このプロシージャをコールすると ENQUEUE および DEQUEUE の両方が使用禁止になります。キューに対する未処理のトランザクションがある場合、キューは停止できません。この操作はコールが完了すると有効になり、トランザクションの特性はありません。

構文

```
DBMS_AQADM.STOP_QUEUE (
   queue_name      IN   VARCHAR2,
   enqueue         IN   BOOLEAN DEFAULT TRUE,
   dequeue         IN   BOOLEAN DEFAULT TRUE,
   wait            IN   BOOLEAN DEFAULT TRUE);
```

パラメータ

表 6-11 STOP_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_name	使用禁止にするキューの名前。
enqueue	このキューで ENQUEUE を使用禁止にするかどうかを指定します。 TRUE: ENQUEUE を使用禁止にします。これがデフォルトです。 FALSE: 現在の設定を変更しません。
dequeue	このキューで DEQUEUE を使用禁止にするかどうかを指定します。 TRUE: DEQUEUE を使用禁止にします。これがデフォルトです。 FALSE: 現在の設定を変更しません。
wait	未処理のトランザクションの完了を待つかどうかを指定します。 TRUE: 未処理のトランザクションがある場合は待機します。この状態では、新規トランザクションは、このキューへのエンキューまたはこのキューからのデキューを許可されません。 FALSE: 正常終了かエラーかをすぐに戻します。

GRANT_SYSTEM_PRIVILEGE プロシージャ

このプロシージャは、ユーザーおよびロールに AQ システム権限を付与します。権限には、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY があります。初期設定では、SYS および SYSTEM のみこのプロシージャを正常に使用できます。

構文

```
DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE (
   privilege         IN    VARCHAR2,
   grantee           IN    VARCHAR2,
   admin_option      IN    BOOLEAN := FALSE);
```

パラメータ

表 6-12 GRANT_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	付与する AQ システム権限。ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY から選択できます。 各システム権限に許可される操作は次のとおりです。 ENQUEUE_ANY: この権限を付与されたユーザーは、データベース内のすべてのキューにメッセージをエンキューできます。 DEQUEUE_ANY: この権限を付与されたユーザーは、データベース内のすべてのキューからメッセージをデキューできます。 MANAGE_ANY: この権限を付与されたユーザーは、データベース内のすべてのスキーマで DBMS_AQADM コールを実行できます。
grantee	権限受領者。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。
admin_option	システム権限を ADMIN オプション付きで付与するかどうかを指定します。 ADMIN オプション付きで権限が付与されると、権限受領者はこのプロシージャを使用して、他のユーザーまたはロールにシステム権限を付与できます。デフォルトは FALSE です。

REVOKE_SYSTEM_PRIVILEGE プロシージャ

このプロシージャは、ユーザーおよびロールから AQ システム権限を取り消します。権限には、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY があります。システム権限の ADMIN オプションは、選択的に取り消すことはできません。

構文

```
DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE (
   privilege         IN   VARCHAR2,
   grantee           IN   VARCHAR2);
```

パラメータ

表 6-13 REVOKE_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	取り消す AQ システム権限。ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY から選択できます。 システム権限の ADMIN オプションは、選択的に取り消すことはできません。
grantee	権限受領者。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。

GRANT_QUEUE_PRIVILEGE プロシージャ

このプロシージャは、ユーザーおよびロールにキューの権限を付与します。権限は、ENQUEUE または DEQUEUE です。初期設定では、キュー表の所有者のみこのプロシージャを使用してそのキューの権限を付与できます。

構文

```
DBMS_AQADM.GRANT_QUEUE_PRIVILEGE (
   privilege        IN    VARCHAR2,
   queue_name       IN    VARCHAR2,
   grantee          IN    VARCHAR2,
   grant_option     IN    BOOLEAN := FALSE);
```

パラメータ

表 6-14 GRANT_QUEUE_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	付与する AQ キュー権限。ENQUEUE、DEQUEUE および ALL から選択できます。ALL は、ENQUEUE および DEQUEUE の両方の権限を付与します。
queue_name	キュー名。
grantee	権限受領者。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。
grant_option	アクセス権限を GRANT オプション付きで付与するかどうかを指定します。 GRANT オプション付きで権限が付与されると、権限受領者はキュー表の所有権に関係なく、このプロシージャを使用して他のユーザーまたはロールにアクセス権限を付与できます。デフォルトは FALSE です。

REVOKE_QUEUE_PRIVILEGE プロシージャ

このプロシージャは、ユーザーおよびロールからキューの権限を取り消します。権限は、ENQUEUE または DEQUEUE です。権限を取り消すには、取消し実行者がその権限の付与者である必要があります。GRANT オプションを使用して付与された権限は、付与者の権限が取り消されると、同様に取り消されます。

構文

```
DBMS_AQADM.REVOKE_QUEUE_PRIVILEGE (
   privilege          IN      VARCHAR2,
   queue_name         IN      VARCHAR2,
   grantee            IN      VARCHAR2);
```

パラメータ

表 6-15 REVOKE_QUEUE_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	取り消す AQ キュー権限。ENQUEUE、DEQUEUE および ALL から選択できます。ALL は、ENQUEUE および DEQUEUE の両方の権限を付与します。
queue_name	キュー名。
grantee	権限受領者。権限受領者には、ユーザー、ロールまたは PUBLIC ロールを指定できます。権限が GRANT オプションを使用して付与された場合は、その権限も取り消されます。

ADD_SUBSCRIBER プロシージャ

このプロシージャは、キューにデフォルトのサブスクライバを追加します。

構文

```
DBMS_AQADM.ADD_SUBSCRIBER (
   queue_name     IN    VARCHAR2,
   subscriber     IN    sys.aq$_agent,
   rule           IN    VARCHAR2 DEFAULT NULL,
   transformation IN    VARCHAR2 DEFAULT NULL);
```

パラメータ

表 6-16 ADD_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
queue_name	キュー名。
subscriber	サブスクリプションを定義しようとしているエージェント。

表 6-16 ADD_SUBSCRIBER プロシージャのパラメータ（続き）

パラメータ	説明
rule	メッセージ・プロパティ、メッセージ・データ・プロパティおよびPL/SQL ファンクションに基づいた条件式。 ルールは SQL 問合せの WHERE 句と同様の構文を使用して、ブール式で指定されます。このブール式には、メッセージ・プロパティの状態、ユーザー・データのプロパティ（オブジェクト・ペイロードのみ）、および PL/SQL または SQL ファンクション（SQL 問合せの WHERE 句で指定）の状態を組み込むことができます。現在サポートされているメッセージ・プロパティは、priority および corrid です。 メッセージ・ペイロード（オブジェクト・ペイロード）のルールを指定するには、句にオブジェクト・タイプの属性を使用します。各属性の前に修飾子として tab.user_data を付加して、ペイロードを格納するキュー表の特定の列を示します。ルールのパラメータは 4000 バイト以内です。
transformation	このサブスクライバがメッセージをデキューしたときに適用される変換を指定します。変換のソース・タイプは、キューのタイプと一致させる必要があります。 サブスクライバがリモートで操作している場合、リモート・キューに伝播される前に変換が適用されます。

使用上の注意

プログラムは、特定の受信者リストまたはデフォルトのサブスクライバ・リストに対してメッセージをエンキューできます。この操作は、複数コンシューマが許可されているキューでのみ成功します。この操作はすぐに有効となり、含まれるトランザクションがコミットされます。このコール後に実行されたエンキュー要求は、新しい動作を反映します。

ルール内の任意の文字列を次に示します。

```
rule   => 'PRIORITY <= 3 AND CORRID =  ''FROM JAPAN'''
```

すべて一重引用符が使用されていることに注意してください。

ALTER_SUBSCRIBER プロシージャ

このプロシージャは、指定したキューに対するサブスクライバの既存プロパティを変更します。ルールのみ変更できます。

構文

```
DBMS_AQADM.ALTER_SUBSCRIBER (
   queue_name     IN    VARCHAR2,
   subscriber     IN    sys.aq$_agent,
   rule           IN    VARCHAR2
   transformation IN    VARCHAR2);
```

パラメータ

表 6-17 ALTER_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
queue_name	キュー名。
subscriber	サブスクリプションを変更するエージェント。106-3 ページの「AQ$_AGENT タイプ」を参照してください。
rule	メッセージ・プロパティ、メッセージ・データ・プロパティおよび PL/SQL ファンクションに基づいた条件式。 **注意 :** ルールのパラメータは 4000 バイト以内です。ルールを削除するには、ルール・パラメータを NULL に設定します。
transformation	このサブスクライバがメッセージをデキューしたときに適用される変換を指定します。変換のソース・タイプは、キューのタイプと一致させる必要があります。 サブスクライバがリモートで操作している場合、リモート・キューに伝播される前に変換が適用されます。

使用上の注意

このプロシージャは、サブスクライバのルールおよび変換の両方を変更します。いずれかの既存値を保持するには、元の値を指定する必要があります。サブスクライバのルールおよび変換の現在の値を取得するには、<schema>.AQ$<queue_table>_R および <schema>.AQ$<queue_table>_S のビューを使用します。

REMOVE_SUBSCRIBER プロシージャ

このプロシージャは、キューからデフォルトのサブスクライバを取り消します。この操作はすぐに有効となり、含まれるトランザクションがコミットされます。既存のメッセージ内のサブスクライバへの参照は、この操作の一部としてすべて削除されます。

構文

```
DBMS_AQADM.REMOVE_SUBSCRIBER (
   queue_name          IN          VARCHAR2,
   subscriber          IN          sys.aq$_agent);
```

パラメータ

表 6-18 REMOVE_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
queue_name	キュー名。
subscriber	削除するエージェント。106-3 ページの「AQ$_AGENT タイプ」を参照してください。

SCHEDULE_PROPAGATION プロシージャ

このプロシージャは、キューから特定の DB リンクによって識別される宛先へのメッセージ伝播をスケジュールします。

宛先に NULL を指定すると、同じデータベース内の他のキューにもメッセージを伝播できます。同じ宛先にメッセージの受信者が複数存在する場合は、同じキューにあるか、異なるキューにあるかに関係なく、そのすべてに同時に伝播されます。

構文

```
DBMS_AQADM.SCHEDULE_PROPAGATION (
   queue_name       IN    VARCHAR2,
   destination      IN    VARCHAR2 DEFAULT NULL,
   start_time       IN    DATE     DEFAULT SYSDATE,
   duration         IN    NUMBER   DEFAULT NULL,
   next_time        IN    VARCHAR2 DEFAULT NULL,
   latency          IN    NUMBER   DEFAULT 60);
```

パラメータ

表 6-19 SCHEDULE_PROPAGATION プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されない場合は、デフォルトで管理ユーザーのスキーマ名に設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。
start_time	ソース・キューから宛先へのメッセージに対する伝播ウィンドウの初期起動時間。
duration	伝播ウィンドウの継続期間（秒数）。 NULL 値の場合、伝播ウィンドウは無期限または伝播スケジュールが取り消されるまで継続します。
next_time	現在の伝播ウィンドウの終了から次の伝播ウィンドウの開始を計算する日付ファンクション。 この値が NULL の場合は、現在のウィンドウが終了すると伝播は停止されます。たとえば、毎日同時刻にウィンドウを起動するには、next_time に 'SYSDATE+ 1 - duration/86400' と指定します。
latency	伝播ウィンドウ内にエンキュー後、メッセージが伝播されるまでの最大待機時間（秒数）。 たとえば、待機時間が 60 秒で伝播ウィンドウにある場合は、伝播するメッセージがないと、その宛先に対するそのキューのメッセージは、最低 60 秒間伝播されません。 指定した宛先に対してメッセージを伝播するためにキューを再度チェックするまでの時間は、最短で 60 秒です。待機時間が 600 秒の場合、キューは 10 分間チェックされず、待機時間が 0（ゼロ）の場合は、宛先に対するメッセージがエンキューされるまでジョブ・キュー・プロセスが待機することになります。メッセージは、エンキューされるとすぐに伝播されます。

UNSCHEDULE_PROPAGATION プロシージャ

このプロシージャは、キューから特定の DB リンクで指定された宛先へのメッセージの伝播の旧スケジュールを取り消します。

構文

```
DBMS_AQADM.UNSCHEDULE_PROPAGATION (
   queue_name     IN   VARCHAR2,
   destination    IN   VARCHAR2 DEFAULT NULL);
```

パラメータ

表 6-20 UNSCHEDULE_PROPAGATION プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されない場合は、デフォルトで管理ユーザーのスキーマ名に設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。

VERIFY_QUEUE_TYPES プロシージャ

このプロシージャは、ソース・キューおよび宛先キューのタイプが同じであるかどうかを検証します。検証結果は、sys.aq$_message_types 表に格納され、このコマンドの以前の出力がすべて上書きされます。

構文

```
DBMS_AQADM.VERIFY_QUEUE_TYPES (
   src_queue_name    IN    VARCHAR2,
   dest_queue_name   IN    VARCHAR2,
   destination       IN    VARCHAR2 DEFAULT NULL,
   rc                OUT   BINARY_INTEGER);
```

パラメータ

表 6-21 VERIFY_QUEUE_TYPES プロシージャのパラメータ

パラメータ	説明
src_queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
dest_queue_name	メッセージが伝播される宛先キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。
rc	プロシージャの結果を示すリターン・コード。 エラーがなく、ソースと宛先のキュー・タイプが一致している場合、結果は 1 になります。一致していない場合、結果は 0（ゼロ）になります。Oracle エラーが発生した場合は、rc に戻されます。

ALTER_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、伝播スケジュールのパラメータを変更します。

構文

```
DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE (
   queue_name      IN     VARCHAR2,
   destination     IN     VARCHAR2 DEFAULT NULL,
   duration        IN     NUMBER   DEFAULT NULL,
   next_time       IN     VARCHAR2 DEFAULT NULL,
   latency         IN     NUMBER   DEFAULT 60);
```

パラメータ

表 6-22 ALTER_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。
duration	伝播ウィンドウの継続期間（秒数）。 NULL 値の場合、伝播ウィンドウは無期限または伝播スケジュールが取り消されるまで継続します。
next_time	現在の伝播ウィンドウの終了から次の伝播ウィンドウの開始を計算する日付ファンクション。 この値が NULL の場合は、現在のウィンドウが終了すると伝播は停止されます。たとえば、毎日同時刻にウィンドウを起動するには next_time に 'SYSDATE + 1 - duration/86400' と指定します。

表 6-22 ALTER_PROPAGATION_SCHEDULE プロシージャのパラメータ（続き）

パラメータ	説明
latency	伝播ウィンドウ内にエンキュー後、メッセージが伝播されるまでの最大待機時間（秒数）。 デフォルト値は 60 です。 **注意：**このコールに対して待機時間が指定されないと、待機時間は既存の値をデフォルト値で上書きします。 たとえば、待機時間が 60 秒で伝播ウィンドウにある場合は、伝播するメッセージがないと、その宛先に対するそのキューのメッセージは、最低 60 秒間伝播されません。指定した宛先に対してメッセージを伝播するためにキューを再度チェックするまでの時間は、最短で 60 秒です。待機時間が 600 の場合、キューは 10 分間チェックされず、待機時間が 0（ゼロ）の場合は、宛先に対するメッセージがエンキューされるまでジョブ・キュー・プロセスが待機することになります。メッセージは、エンキューされるとすぐに伝播されます。

ENABLE_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、以前に使用禁止にした伝播スケジュールを使用可能にします。

構文

```
DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE (
   queue_name      IN     VARCHAR2,
   destination     IN     VARCHAR2 DEFAULT NULL);
```

パラメータ

表 6-23 ENABLE_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。

DISABLE_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、伝播スケジュールを無効にします。

構文

```
DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE (
   queue_name      IN     VARCHAR2,
   destination     IN     VARCHAR2 DEFAULT NULL);
```

パラメータ

表 6-24 DISABLE_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
queue_name	伝播対象のメッセージがあるソース・キューの名前。スキーマ名を含みます。 スキーマ名が指定されていない場合は、デフォルトでユーザーのスキーマ名が設定されます。
destination	宛先の DB リンク。 この宛先の受信者に対するソース・キュー内のメッセージが伝播されます。宛先が NULL の場合は、ローカル・データベースが宛先となり、メッセージはローカル・データベース内の他のキューに伝播されます。このフィールドの長さは 128 バイトに制限されており、名前が完全に修飾されていない場合は、デフォルトのドメイン名が使用されます。

MIGRATE_QUEUE_TABLE プロシージャ

このプロシージャは、8.0 互換キュー表から 8.1 互換キュー表へのアップグレード、または 8.1 互換キュー表から 8.0 互換キュー表へのダウングレードを実行します。

構文

```
DBMS_AQADM.MIGRATE_QUEUE_TABLE (
   queue_table    IN   VARCHAR2,
   compatible     IN   VARCHAR2);
```

パラメータ

表 6-25 MIGRATE_QUEUE_TABLE プロシージャのパラメータ

パラメータ	説明
queue_table	移行するキュー表の名前を指定します。
compatible	8.0 互換キュー表を 8.1 互換キュー表にアップグレードするには '8.1' を設定します。8.1 互換キュー表を 8.0 互換キュー表にダウングレードするには '8.0' に設定します。

CREATE_AQ_AGENT プロシージャ

このプロシージャは、HTTP/SMTP プロトコルを使用して、AQ インターネット・アクセスのエージェントを登録します。また、保護キューにアクセスする AQ エージェントを作成するために使用されます。

> **関連項目：** 保護キューの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
DBMS_AQADM.CREATE_AQ_AGENT (
  agent_name                IN VARCHAR2,
  certificate_location      IN VARCHAR2 DEFAULT NULL,
  enable_http               IN BOOLEAN DEFAULT FALSE,
  enable_smtp               IN BOOLEAN DEFAULT FALSE,
  enable_anyp               IN BOOLEAN DEFAULT FALSE )
```

パラメータ

表 6-26 CREATE_AQ_AGENT プロシージャのパラメータ

パラメータ	説明
agent_name	AQ インターネット・エージェントのユーザー名を指定します。
certification_location	LDAP におけるエージェントの証明書の位置（デフォルトは NULL）。 エージェントが SMTP を介して AQ へのアクセスを許可されている場合は、証明書を LDAP に登録する必要があります。 HTTP を介してアクセスする場合、証明書の位置は必要ありません。
enable_http	TRUE: エージェントは HTTP を介して AQ にアクセスできます。 FALSE: エージェントは HTTP を介して AQ にアクセスできません。
enable_smtp	TRUE: エージェントは SMTP（電子メール）を介して AQ にアクセスできます。 FALSE: エージェントは SMTP を介して AQ にアクセスできません。
enable_anyp	TRUE: エージェントはプロトコル（HTTP または SMTP）を介して AQ にアクセスできます。

使用上の注意

SYS.AQ$INTERNET_USERS ビューには、AQ インターネット・エージェントがすべてリストされます。

ALTER_AQ_AGENT プロシージャ

このプロシージャは、AQ インターネット・アクセスに登録されたエージェントを変更します。また、保護キューにアクセスする AQ エージェントを変更するために使用されます。

> **関連項目：** 保護キューの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
DBMS_AQADM.ALTER_AQ_AGENT (
  agent_name                IN VARCHAR2,
  certificate_location      IN VARCHAR2 DEFAULT NULL,
  enable_http               IN BOOLEAN DEFAULT FALSE,
  enable_smtp               IN BOOLEAN DEFAULT FALSE,
  enable_anyp               IN BOOLEAN DEFAULT FALSE );
```

パラメータ

表 6-27 ALTER_AQ_AGENT プロシージャのパラメータ

パラメータ	説明
agent_name	AQ インターネット・エージェントのユーザー名を指定します。
certification_ location	LDAP におけるエージェントの証明書の位置（デフォルトは NULL）。 エージェントが SMTP を介して AQ へのアクセスを許可されている場合、証明書を LDAP に登録する必要があります。 HTTP を介してアクセスする場合、証明書の位置は必要ありません。
enable_http	TRUE: エージェントは HTTP を介して AQ にアクセスできます。 FALSE: エージェントは HTTP を介して AQ にアクセスできません。
enable_smtp	TRUE: エージェントは SMTP（電子メール）を介して AQ にアクセスできます。 FALSE: エージェントは SMTP を介して AQ にアクセスできません。
enable_anyp	TRUE: エージェントはプロトコル（HTTP または SMTP）を介して AQ にアクセスできます。

DROP_AQ_AGENT プロシージャ

このプロシージャは、AQ インターネット・アクセスに以前に登録されたエージェントを削除します。

構文

```
DBMS_AQADM.DROP_AQ_AGENT (
  agent_name                IN VARCHAR2)
```

パラメータ

表 6-28 DROP_AQ_AGENT プロシージャのパラメータ

パラメータ	説明
agent_name	AQ インターネット・エージェントのユーザー名を指定します。

ENABLE_DB_ACCESS プロシージャ

このプロシージャは、AQ インターネット・エージェントに特定のデータベース・ユーザーとしての権限を付与します。CREATE_AQ_AGENT プロシージャを使用して、事前に AQ インターネット・エージェントが作成されている必要があります。

保護キューの場合、メッセージの送信者エージェントと受信者エージェントが、エンキュー操作またはデキュー操作を実行するデータベース・ユーザーにマップされる必要があります。

> **関連項目：** 保護キューの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
DBMS_AQADM.ENABLE_DB_ACCESS (
  agent_name                IN VARCHAR2,
  db_username               IN VARCHAR2)
```

パラメータ

表 6-29 ENABLE_DB_ACCESS プロシージャのパラメータ

パラメータ	説明
agent_name	AQ インターネット・エージェントのユーザー名を指定します。
db_username	AQ インターネット・エージェントへの権限を付与されるデータベース・ユーザーを指定します。

使用上の注意

SYS.AQ$INTERNET_USERS ビューには、AQ インターネット・エージェントおよび権限が付与されるデータベース・ユーザーの名前がすべてリストされます。

DISABLE_DB_ACCESS プロシージャ

このプロシージャは、AQ インターネット・エージェントから特定のデータベース・ユーザーとしての権限を取り消します。ENABLE_DB_ACCESS プロシージャを使用して、事前に AQ インターネット・エージェントへそれらの権限が付与されている必要があります。

構文

```
DBMS_AQADM.DISABLE_DB_ACCESS (
  agent_name                IN VARCHAR2,
  db_username               IN VARCHAR2)
```

パラメータ

表 6-30 DISABLE_DB_ACCESS プロシージャのパラメータ

パラメータ	説明
agent_name	AQ インターネット・エージェントのユーザー名を指定します。
db_username	AQ インターネット・エージェントから権限を取り消されるデータベース・ユーザーを指定します。

ADD_ALIAS_TO_LDAP プロシージャ

このプロシージャは、キュー、エージェントまたは JMS ConnectionFactory の別名を LDAP に作成します。別名は、LDAP 階層においてデータベース・サーバーの識別名の下に直接配置されます。

構文

```
DBMS_AQADM.ADD_ALIAS_TO_LDAP(
   alias          IN VARCHAR2,
   obj_location   IN VARCHAR2);
```

パラメータ

表 6-31 ADD_ALIAS_TO_LDAP プロシージャのパラメータ

パラメータ	説明
alias	別名。 例 : 'west_shipping'
obj_location	alias が参照するオブジェクト（キュー、エージェントまたは接続ファクトリ）の識別名。

使用上の注意

この方法は、キュー、エージェントおよび JMS ConnectionFactory オブジェクトの別名の作成に使用できます。別名を作成するには、オブジェクトが存在している必要があります。作成した別名は、JMS の JNDI 参照および AQ インターネット・アクセスに使用できます。

DEL_ALIAS_FROM_LDAP プロシージャ

このプロシージャは、キュー、エージェントまたは JMS ConnectionFactory の別名を LDAP から取り消します。

構文

```
DBMS_AQ.DEL_ALIAS_FROM_LDAP(
   alias IN VARCHAR2);
```

パラメータ

表 6-32 DEL_ALIAS_FROM_LDAP プロシージャのパラメータ

パラメータ	説明
alias	削除される別名。

7

DBMS_AQELM

DBMS_AQELM パッケージは、電子メールおよび HTTP によるアドバンスト・キューイングの非同期通知の構成を管理するプロシージャを提供します。

> **関連項目：** DBMS_AQELM の詳細は、『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。

この章では、次の項目について説明します。

DBMS_AQELM サブプログラムの要約

表 7-1 DBMS_AQELM サブプログラム

サブプログラム	説明
「SET_MAILHOST プロシージャ」7-3 ページ	SMTP サーバーのホスト名を設定します。
「GET_MAILHOST プロシージャ」7-3 ページ	SMTP サーバーのホスト名を取得します。
「SET_MAILPORT プロシージャ」7-4 ページ	SMTP サーバーのポート番号を設定します。
「GET_MAILPORT プロシージャ」7-4 ページ	SMTP サーバーのポート番号を取得します。
「SET_SENDFROM プロシージャ」7-5 ページ	発信元電子メール・アドレスを設定します。
「GET_SENDFROM プロシージャ」7-5 ページ	発信元電子メール・アドレスを取得します。
「SET_PROXY プロシージャ」7-6 ページ	no_proxy_domains で指定されたドメインに属するホストへの要求を除外し、HTTP プロトコルの要求に使用するプロキシ・サーバー名を設定します。
「GET_PROXY プロシージャ」7-7 ページ	DBMS_AQELM.SET_PROXY により HTTP 通知用に設定されたプロキシ・サーバー名および no_proxy_domains セットを取得します。

SET_MAILHOST プロシージャ

このプロシージャは、SMTP サーバーのホスト名を設定します。電子メール通知を構成する一部分として、DBMS_AQELM パッケージに対して AQ_ADMINISTRATOR_ROLE または EXECUTE の許可を付与されたユーザーは、電子メール通知を登録する前にホスト名を設定する必要があります。データベースではこの SMTP サーバーのホスト名を使用して、電子メール通知を送信します。

構文

```
DBMS_AQELM.SET_MAILHOST (
   mailhost  IN  VARCHAR2);
```

パラメータ

表 7-2 SET_MAILHOST プロシージャのパラメータ

パラメータ	説明
mailhost	SMTP サーバーのホスト名。

GET_MAILHOST プロシージャ

このプロシージャは、DBMS_AQELM.SET_MAILHOST を使用して SMTP サーバー用に設定されたホスト名を取得します。

構文

```
DBMS_AQELM.GET_MAILHOST (
   mailhost  OUT VARCHAR2);
```

パラメータ

表 7-3 GET_MAILHOST プロシージャのパラメータ

パラメータ	説明
mailhost	SMTP サーバーのホスト名。

SET_MAILPORT プロシージャ

このプロシージャは、SMTP サーバーのポート番号を設定します。電子メール通知を構成する一部分として、DBMS_AQELM パッケージに対して AQ_ADMINISTRATOR_ROLE または EXECUTE の許可を付与されたユーザーは、電子メール通知を登録する前にポート番号を設定する必要があります。データベースではこの SMTP サーバーのポート番号を使用して、電子メール通知を送信します。設定されていない場合、SMTP の mailport のデフォルトは 25 となります。

構文

```
DBMS_AQELM.SET_MAILPORT (
   mailport  IN NUMBER);
```

パラメータ

表 7-4 に、SET_MAILPORT プロシージャのパラメータを示します。

表 7-4 SET_MAILPORT プロシージャのパラメータ

パラメータ	説明
mailport	SMTP サーバーのポート番号。

GET_MAILPORT プロシージャ

このプロシージャは、DBMS_AQELM.SET_MAILPORT を使用して設定された SMTP サーバーのポート番号またはデフォルト値（25）を取得します。

構文

```
DBMS_AQELM.GET_MAILPORT (
   mailport  OUT NUMBER);
```

パラメータ

表 7-5 GET_MAILPORT プロシージャのパラメータ

パラメータ	説明
mailport	SMTP サーバーのポート番号。

SET_SENDFROM プロシージャ

このプロシージャは、発信元電子メール・アドレスを設定します。電子メール通知を構成する一部分として、DBMS_AQELM パッケージに対して AQ_ADMINISTRATOR_ROLE または EXECUTE の許可を付与されたユーザーは、電子メール通知を登録する前に発信元アドレスを設定する必要があります。この電子メール・アドレスは、データベースから登録済み電子メール・アドレスに送信されるすべての電子メール通知において、sent-from フィールドに使用されます。

構文

```
DBMS_AQELM.SET_SENDFROM (
   sendfrom  IN  VARCHAR2);
```

パラメータ

表 7-6 SET_SENDFROM プロシージャのパラメータ

パラメータ	説明
sendfrom	発信元電子メール・アドレス。

GET_SENDFROM プロシージャ

このプロシージャは、DBMS_AQELM.SET_SENDFROM プロシージャを使用して設定された発信元電子メール・アドレスを取得します。

構文

```
DBMS_AQELM.GET_SENDFROM (
   sendfrom  OUT  VARCHAR2);
```

パラメータ

表 7-7 GET_SENDFROM プロシージャのパラメータ

パラメータ	プロシージャ
sendfrom	発信元電子メール・アドレス。

SET_PROXY プロシージャ

このプロシージャは、no_proxy_domains で指定されたドメインに属するホストへの要求を除外し、HTTP プロトコルの要求に使用するプロキシ・サーバー名を設定します。プロキシ・サーバー名には、オプションで、プロキシ・サーバーがリスニングする TCP/IP ポート番号を含めることができます。プロキシ・サーバーのポートを指定していない場合のデフォルト・ポートは 80 になります。no_proxy_domains は、プロキシ・サーバーを経由せずに、HTTP サーバーの宛先に HTTP 要求が直接送信されるドメインまたはホストのリストです。必要に応じて、ドメインまたはホストごとにポート番号を指定できます。ポート番号を指定している場合、非プロキシの制限が適用されるのは、特定のドメインまたはホストのポートでの要求のみです。no_proxy_domains の値が NULL でプロキシ・サーバーが設定されている場合、要求はすべてプロキシ・サーバーを経由して送信されます。プロキシ・サーバーが設定されていない場合は、http_send により要求が目的の Web サーバーに直接送信されます。

HTTP 通知の構成の一環として、DBMS_AQELM パッケージに対して AQ_ADMINISTRATOR_ROLE または EXECUTE の許可を付与されたユーザーは、HTTP 通知を登録する前に、必要に応じてプロキシ・サーバー名および no_proxy_domains のリストを設定する選択ができます。データベースでは、この情報を使用して HTTP 通知を転送します。

構文

```
DBMS_AQELM.SET_PROXY (
   proxy            IN VARCHAR2,
   no_proxy_domains IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 7-8 SET_PROXY プロシージャのパラメータ

パラメータ	プロシージャ
proxy	プロキシ・サーバーのホストおよびポート番号。構文は、"[http://]host[:port][/]" です。たとえば、"www-proxy.my-company.com:80" のようになります。
no_proxy_domains	非プロキシ・ドメインまたはホストのリスト。構文は、ホストまたはドメインのリストです。カンマ、セミコロンまたはスペースで区切ったオプションのポート番号を入力できます。たとえば、"corp.my-company.com, eng.my-company.com:80" のようになります。

GET_PROXY プロシージャ

このプロシージャは、DBMS_AQELM.SET_PROXY を使用して HTTP 通知向けに設定されたプロキシ・サーバー名および no_proxy_domains を取得します。

構文

```
DBMS_AQELM.GET_PROXY (
   proxy            OUT VARCHAR2,
   no_proxy_domains OUT VARCHAR2);
```

パラメータ

表 7-9 GET_PROXY プロシージャのパラメータ

パラメータ	プロシージャ
proxy	プロキシ・サーバーのホストおよびポート番号。
no_proxy_domains	非プロキシ・ドメインまたはホストのリスト。

8

DBMS_CAPTURE_ADM

DBMS_CAPTURE_ADM パッケージは、取得プロセスを開始、停止および構成するための管理プロシージャを提供します。取得した変更のソースは REDO ログで、および取得した変更のリポジトリはキュー（DBMS_AQADM パッケージまたは DBMS_STEAMS_ADM.SET_UP_QUEUE プロシージャを使用して作成）です。

この章では、次の項目について説明します。

関連項目： 取得プロセスの詳細は、『Oracle9*i* Streams』を参照してください。

DBMS_CAPTURE_ADM サブプログラムの要約

表 8-1 DBMS_CAPTURE_ADM サブプログラム

サブプログラム	説明
「ABORT_GLOBAL_INSTANTIATION プロシージャ」8-3 ページ	PREPARE_GLOBAL_INSTANTIATION プロシージャの実行効果を元に戻します。
「ABORT_SCHEMA_INSTANTIATION プロシージャ」8-3 ページ	PREPARE_SCHEMA_INSTANTIATION プロシージャの実行効果を元に戻します。
「ABORT_TABLE_INSTANTIATION プロシージャ」8-4 ページ	PREPARE_TABLE_INSTANTIATION プロシージャの実行効果を元に戻します。
「ALTER_CAPTURE プロシージャ」8-4 ページ	取得プロセスを変更します。
「CREATE_CAPTURE プロシージャ」8-6 ページ	取得プロセスを作成します。
「DROP_CAPTURE プロシージャ」8-8 ページ	取得プロセスを削除します。
「PREPARE_GLOBAL_INSTANTIATION プロシージャ」8-8 ページ	別のデータベースで、データベース内のすべての表をインスタンス化するために必要な同期化を行います。
「PREPARE_SCHEMA_INSTANTIATION プロシージャ」8-9 ページ	別のデータベースで、スキーマ内のすべての表をインスタンス化するために必要な同期化を行います。
「PREPARE_TABLE_INSTANTIATION プロシージャ」8-9 ページ	別のデータベースで、表をインスタンス化するために必要な同期化を行います。
「SET_PARAMETER プロシージャ」8-10 ページ	取得プロセス・パラメータを指定した値に設定します。
「START_CAPTURE プロシージャ」8-13 ページ	取得プロセスを開始します。取得プロセスは、REDO ログを調査し、取り出した REDO 情報を関連キューにエンキューします。
「STOP_CAPTURE プロシージャ」8-14 ページ	取得プロセスを停止して REDO ログの調査を中断します。

注意： 特に指定されていないかぎり、プロシージャはすべてコミットします。

ABORT_GLOBAL_INSTANTIATION プロシージャ

PREPARE_GLOBAL_INSTANTIATION プロシージャの実行結果を元に戻します。

具体的には、このプロシージャを実行すると、データベースのインスタンス化に関連するデータ・ディクショナリ情報が削除されます。

構文

```
DBMS_CAPTURE_ADM.ABORT_GLOBAL_INSTANTIATION();
```

ABORT_SCHEMA_INSTANTIATION プロシージャ

PREPARE_SCHEMA_INSTANTIATION プロシージャの実行結果を元に戻します。

具体的には、このプロシージャを実行すると、スキーマのインスタンス化に関連するデータ・ディクショナリ情報が削除されます。

構文

```
DBMS_CAPTURE_ADM.ABORT_SCHEMA_INSTANTIATION(
   schema_name     IN   VARCHAR2);
```

パラメータ

表 8-2 ABORT_SCHEMA_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
schema_name	準備中のインスタンス化の結果を元に戻すスキーマの名前。

ABORT_TABLE_INSTANTIATION プロシージャ

PREPARE_TABLE_INSTANTIATION プロシージャの実行結果を元に戻します。

具体的には、このプロシージャを実行すると、表のインスタンス化に関連するデータ・ディクショナリ情報が削除されます。

構文

```
DBMS_CAPTURE_ADM.ABORT_TABLE_INSTANTIATION(
   table_name    IN   VARCHAR2);
```

パラメータ

表 8-3 ABORT_TABLE_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
table_name	準備中のインスタンス化の結果を元に戻す表の名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr.employees のようになります。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。

ALTER_CAPTURE プロシージャ

取得プロセスを変更します。

構文

```
DBMS_CAPTURE_ADM.ALTER_CAPTURE(
   capture_name      IN VARCHAR2,
   rule_set_name     IN VARCHAR2  DEFAULT NULL,
   remove_rule_set   IN BOOLEAN   DEFAULT false,
   start_scn         IN NUMBER    DEFAULT NULL);
```

パラメータ

表 8-4 ALTER_CAPTURE プロシージャのパラメータ

パラメータ	説明
capture_name	変更する取得プロセスの名前。既存の取得プロセス名を指定する必要があります。
rule_set_name	この取得プロセスに対する取得ルールが含まれているルール・セットの名前。取得プロセスに対してルール・セットを使用する場合は、既存のルール・セットを [*schema_name.*]*rule_set_name* の形式で指定する必要があります。たとえば、hr スキーマ内の job_capture_rules という名前のルール・セットを指定するには、hr.job_capture_rules と入力します。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。 指定したルール・セットが存在しない場合はエラーが戻されます。ルール・セットを作成し、そのルール・セットにルールを追加するには、DBMS_RULE_ADM パッケージを使用します。 **関連項目：**取得プロセスによって取得可能な変更の詳細は、『Oracle9*i* Streams』を参照してください。
remove_rule_set	TRUE に設定すると、指定した取得プロセスに対するルール・セットが削除されます。取得プロセスに対するルール・セットを削除すると、取得プロセスは、データベース内のすべてのオブジェクトに対してサポートされているすべての変更を取得します。ただし、SYS スキーマと SYSTEM スキーマ内のデータベース・オブジェクトは除きます。 FALSE に設定すると、指定した取得プロセスに対するルール・セットは保持されます。 rule_set_name パラメータが NULL 以外の場合、このパラメータは FALSE に設定してください。
start_scn	取得プロセスが変更を取得しているデータベースに対する有効な過去の SCN。取得プロセスは、指定された SCN から変更の取得を開始します。 指定される SCN 値は、データベースに対する最初の取得プロセスが作成された後のある時点の値であることが必要です。データベースに対する最初の取得プロセスは、変更対象の取得プロセスの場合と、そうでない場合があります。無効な SCN が指定された場合はエラーが戻されます。 **注意：** 取得プロセスに対する開始 SCN を変更すると、取得プロセスは自動的に停止し、再開します。

CREATE_CAPTURE プロシージャ

取得プロセスを作成します。

CREATE_CAPTURE プロシージャを実行するユーザーは、変更を取得するユーザーです。このユーザーには、変更を取得するためのいくつかの権限が必要です。これらの権限は、次のとおりです。

- 取得プロセスで使用されるルール・セットの実行権限
- ルール・セットで使用されるすべての変換ファンクションの実行権限
- 取得プロセスで使用されるキューのエンキュー権限

注意： データベースで最初の取得プロセスを作成するときは、作成時にデータ・ディクショナリが複製されるため、時間がかかる場合があります。

関連項目： ルールとルール・セットの詳細は、『Oracle9*i* Streams』および第 64 章「DBMS_RULE_ADM」を参照してください。

構文

```
DBMS_CAPTURE_ADM.CREATE_CAPTURE(
   queue_name      IN VARCHAR2,
   capture_name    IN VARCHAR2,
   rule_set_name   IN VARCHAR2  DEFAULT NULL,
   start_scn       IN NUMBER    DEFAULT NULL);
```

パラメータ

表 8-5 CREATE_CAPTURE プロシージャのパラメータ

パラメータ	説明
queue_name	取得プロセスが変更をエンキューするキューの名前。既存のキューを [*schema_name*.]*queue_name* の形式で指定する必要があります。たとえば、hr スキーマ内の streams_queue という名前のキューを指定するには、hr.streams_queue と入力します。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。 **注意：** queue_name の設定は、取得プロセスの作成後には変更できません。

表 8-5 CREATE_CAPTURE プロシージャのパラメータ（続き）

パラメータ	説明
capture_name	作成する取得プロセスの名前。NULL 指定は許可されていません。 **注意：** capture_name の設定は、取得プロセスの作成後には変更できません。
rule_set_name	この取得プロセスに対する取得ルールが含まれているルール・セットの名前。取得プロセスに対してルール・セットを使用する場合は、既存のルール・セットを [*schema_name*.]*rule_set_name* の形式で指定する必要があります。たとえば、hr スキーマ内の job_capture_rules という名前のルール・セットを指定するには、hr.job_capture_rules と入力します。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。 指定したルール・セットが存在しない場合はエラーが戻されます。ルール・セットを作成し、そのルール・セットにルールを追加するには、DBMS_RULE_ADM パッケージを使用します。 NULL を指定すると、取得プロセスは、データベース内のすべてのオブジェクトに対してサポートされているすべての変更を取得します。ただし、SYS スキーマと SYSTEM スキーマ内のデータベース・オブジェクトは除きます。 **関連項目：**取得プロセスによって取得可能な変更の詳細は、『Oracle9*i* Streams』を参照してください。
start_scn	取得プロセスが変更を取得しているデータベースに対する有効な過去の SCN。取得プロセスは、指定された SCN から変更の取得を開始します。 指定される SCN 値は、データベースに対する最初の取得プロセスが作成された後のある時点の値であることが必要です。作成される取得プロセスが、現行のデータベースに対して作成される最初の取得プロセスの場合、start_scn には NULL を指定する必要があります。無効な SCN が指定された場合はエラーが戻されます。

DROP_CAPTURE プロシージャ

取得プロセスを削除します。

構文

```
DBMS_CAPTURE_ADM.DROP_CAPTURE(
   capture_name     IN VARCHAR2);
```

パラメータ

表 8-6 DROP_CAPTURE プロシージャのパラメータ

パラメータ	説明
capture_name	削除する取得プロセスの名前。既存の取得プロセス名を指定します。

PREPARE_GLOBAL_INSTANTIATION プロシージャ

別のデータベースで、データベース内のすべての表をインスタンス化するために必要な同期化を行います。このプロシージャは、ソース・データベースで実行します。

このプロシージャは、インスタンス化のためにデータベース内の各オブジェクトの最小 SCN を記録します。オブジェクトに対する最小 SCN の次の SCN をオブジェクトのインスタンス化に使用できます。このプロシージャを実行すると、データベース内のすべての現行オブジェクトと将来のオブジェクトのインスタンス化が準備されます。

構文

```
DBMS_CAPTURE_ADM.PREPARE_GLOBAL_INSTANTIATION;
```

PREPARE_SCHEMA_INSTANTIATION プロシージャ

別のデータベースで、スキーマ内のすべての表をインスタンス化するために必要な同期化を行います。このプロシージャは、ソース・データベースで実行します。

このプロシージャは、インスタンス化のためにスキーマ内の各オブジェクトの最小 SCN を記録します。オブジェクトに対する最小 SCN の次の SCN をオブジェクトのインスタンス化に使用できます。このプロシージャを実行すると、スキーマ内のすべての現行オブジェクトと将来のオブジェクトのインスタンス化が準備されます。

構文

```
DBMS_CAPTURE_ADM.PREPARE_SCHEMA_INSTANTIATION(
   schema_name  IN VARCHAR2);
```

パラメータ

表 8-7 PREPARE_SCHEMA_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマの名前。たとえば、hr のようになります。

PREPARE_TABLE_INSTANTIATION プロシージャ

別のデータベースで、表をインスタンス化するために必要な同期化を行います。このプロシージャは、ソース・データベースで実行します。

このプロシージャは、インスタンス化のために表の最小 SCN を記録します。オブジェクトに対する最小 SCN の次の SCN をオブジェクトのインスタンス化に使用できます。

構文

```
DBMS_CAPTURE_ADM.PREPARE_TABLE_INSTANTIATION(
   table_name  IN VARCHAR2);
```

パラメータ

表 8-8 PREPARE_TABLE_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
table_name	表の名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr.employees のようになります。スキーマが指定されない場合は、カレント・ユーザー名がデフォルトで使用されます。

SET_PARAMETER プロシージャ

取得プロセス・パラメータを指定した値に設定します。

パラメータ値を変更したとき、パラメータの新しい値が有効になるまでに時間がかかる場合があります。

構文

```
DBMS_CAPTURE_ADM.SET_PARAMETER(
   capture_name    IN VARCHAR2,
   parameter       IN VARCHAR2,
   value           IN VARCHAR2);
```

パラメータ

表 8-9 SET_PARAMETER プロシージャのパラメータ

パラメータ	説明
capture_name	取得プロセスの名前。取得プロセスは、LogMiner を使用して REDO ログから変更を取得します。
parameter	設定するパラメータの名前。これらのパラメータのリストについては、8-11 ページの「取得プロセスのパラメータ」を参照してください。
value	パラメータに設定する値。

取得プロセスのパラメータ

次の表に、取得プロセスのパラメータを示します。

表 8-10 取得プロセスのパラメータ

パラメータ名	設定可能な値	デフォルト	説明
disable_on_limit	Y または N	N	Y の場合は、time_limit パラメータまたは message_limit パラメータによって指定された値に達したために取得プロセスが終了すると、取得プロセスは無効化されます。 N の場合、取得プロセスは、制限に達したために停止した後すぐに再開されます。
maximum_scn	有効な SCN 値または infinite	infinite	取得プロセスは、指定された値以上の SCN を持つ変更レコードの取得前に無効化されます。 infinite の場合は、SCN の値に関係なく取得プロセスが実行されます。
message_limit	正の整数または infinite	infinite	取得プロセスは、指定した数のメッセージを取得した後停止します。 infinite の場合は、取得したメッセージの数に関係なく実行し続けます。
parallelism	正の整数	1	REDO ログを同時に調査できるパラレル実行サーバーの数。 **注意：** ■ このパラメータの値を変更すると、取得プロセスは自動的に停止し、再開します。 ■ parallelism パラメータを、使用可能なパラレル実行サーバー数より大きい値に設定すると、取得プロセスが無効化される場合があります。parallelism 取得プロセス・パラメータを設定するときは、PROCESSES および PARALLEL_MAX_SERVERS 初期化パラメータが適切に設定されていることを確認してください。
startup_seconds	0、正の整数または infinite	0	同じ取得プロセスの別のインスタンス化が終了するのを待機する最大秒数。同じ取得プロセスの別のインスタンス化がこの時間内に終了しない場合、取得プロセスは開始しません。 infinite の場合、取得プロセスは、同じ取得プロセスの別のインスタンス化が終了した後で開始します。

表 8-10 取得プロセスのパラメータ（続き）

パラメータ名	設定可能な値	デフォルト	説明
time_limit	正の整数または infinite	infinite	取得プロセスは、開始してから指定秒数が経過した後、可能なかぎり早く停止します。 infinite の場合、取得プロセスは明示的に停止されるまで実行し続けます。
trace_level	0 または正の整数	0	オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。
write_alert_log	Y または N	Y	Y の場合、取得プロセスは、終了時にアラート・ログにメッセージを書き込みます。 N の場合は、終了時にアラート・ログにメッセージを書き込みません。 メッセージには、取得プロセスの停止理由が示されます。

注意：

- 正の整数として解釈されるすべてのパラメータについて、設定可能な最大値は 4,294,967,295 です。適用可能な場合、比較的大きい値には infinite を指定してください。
- SCN 設定が必要なパラメータについては、任意の有効な SCN 値を指定できます。

START_CAPTURE プロシージャ

取得プロセスを開始します。取得プロセスは、REDO ログを調査し、取り出した REDO 情報を関連キューにエンキューします。

開始ステータスは永続的に記録されます。したがって、ステータスが ENABLED の場合、取得プロセスはデータベース・インスタンスの起動時に開始されます。

取得プロセスはバックグラウンドの Oracle プロセスであり、接頭辞 CP が付加されます。

DBMS_AQADM.START_QUEUE および DBMS_AQADM.STOP_QUEUE のエンキューとデキューの状態は、取得プロセスの開始ステータスには影響を与えません。

取得プロセスは、次のプロシージャを使用して作成できます。

- DBMS_CAPTURE_ADM.CREATE_CAPTURE
- DBMS_STREAMS_ADM.ADD_GLOBAL_RULES
- DBMS_STREAMS_ADM.ADD_SCHEMA_RULES
- DBMS_STREAMS_ADM.ADD_TABLE_RULES

関連項目： 第 73 章「DBMS_STREAMS_ADM」

構文

```
DBMS_CAPTURE_ADM.START_CAPTURE(
   capture_name  IN VARCHAR2);
```

パラメータ

表 8-11 START_CAPTURE プロシージャのパラメータ

パラメータ	説明
capture_name	取得プロセスの名前。取得プロセスは、LogMiner を使用して REDO 情報内の変更を取得します。NULL 設定は許可されていません。

STOP_CAPTURE プロシージャ

取得プロセスを停止して REDO ログの調査を中断します。

停止ステータスは永続的に記録されます。したがって、ステータスが DISABLED の場合、取得プロセスはデータベース・インスタンスの起動時に開始されません。

DBMS_AQADM.START_QUEUE および DBMS_AQADM.STOP_QUEUE のエンキューとデキューの状態は、取得プロセスの停止ステータスには影響を与えません。

構文

```
DBMS_CAPTURE_ADM.STOP_CAPTURE(
   capture_name  IN VARCHAR2,
   force         IN BOOLEAN  DEFAULT false);
```

パラメータ

表 8-12 STOP_CAPTURE プロシージャのパラメータ

パラメータ	説明
capture_name	取得プロセスの名前。NULL 設定は許可されていません。
force	TRUE に設定すると、取得プロセスは即時に停止します。 FALSE に設定すると、取得プロセスは、その現行トランザクションを取得した後で停止します。

9

DBMS_DDL

このパッケージは、ストアド・プロシージャから一部の SQL データ定義言語（DDL）文へのアクセスを提供します。DDL では使用できない特殊な管理操作も提供します。

ALTER_COMPILE および ANALYZE_OBJECT プロシージャは、現行のトランザクションをコミットし、操作を実行してから再度コミットします。

このパッケージは、パッケージ所有者 SYS ではなく、コール・ユーザーの権限で実行されます。

この章では、次の項目について説明します。

DBMS_DDL サブプログラムの要約

表 9-1　DBMS_DDL パッケージのサブプログラム

サブプログラム	説明
「ALTER_COMPILE プロシージャ」9-2 ページ	PL/SQL オブジェクトをコンパイルします。
「ANALYZE_OBJECT プロシージャ」9-3 ページ	データベース・オブジェクトの統計情報を提供します。
「IS_TRIGGER_FIRE_ONCE ファンクション」9-5 ページ	指定した DML または DDL トリガーが 1 回起動されるように設定されている場合は、TRUE を戻します。それ以外の場合は FALSE を戻します。
「SET_TRIGGER_FIRING_PROPERTY プロシージャ」9-6 ページ	指定した DML または DDL トリガーの起動プロパティを設定します。
「ALTER_TABLE_REFERENCEABLE プロシージャ」9-7 ページ	オブジェクト表を再編成し、参照を再設定します。
「ALTER_TABLE_NOT_REFERENCEABLE プロシージャ」9-8 ページ	オブジェクト表を再編成し、参照を再設定します。

ALTER_COMPILE プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ALTER PROCEDURE|FUNCTION|PACKAGE [<schema>.] <name> COMPILE [BODY]
```

構文

```
DBMS_DDL.ALTER_COMPILE (
   type   VARCHAR2,
   schema VARCHAR2,
   name   VARCHAR2);
```

パラメータ

表 9-2 ALTER_COMPILE プロシージャのパラメータ

パラメータ	説明
type	PROCEDURE、FUNCTION、PACKAGE、PACKAGE BODY または TRIGGER のいずれかを設定します。
schema	スキーマ名。 NULL の場合は、現行のスキーマ（大 / 小文字区別）が使用されます。
name	オブジェクトの名前（大 / 小文字区別）。

例外

表 9-3 ALTER_COMPILE プロシージャの例外

例外	説明
ORA-20000:	権限が不十分であるか、またはオブジェクトが存在しません。
ORA-20001:	リモート・オブジェクトであるためコンパイルできません。
ORA-20002:	オブジェクト・タイプの値が正しくありません。 PACKAGE、PACKAGE BODY、PROCEDURE、FUNCTION または TRIGGER のいずれかを設定してください。

ANALYZE_OBJECT プロシージャ

このプロシージャは、指定された表、索引またはクラスタに関する統計情報を提供します。このプロシージャは、次の SQL 文と同じです。

```
ANALYZE TABLE|CLUSTER|INDEX [<schema>.]<name> [<method>] STATISTICS [SAMPLE <n>
[ROWS|PERCENT]]
```

構文

```
DBMS_DDL.ANALYZE_OBJECT (
   type             VARCHAR2,
   schema           VARCHAR2,
   name             VARCHAR2,
   method           VARCHAR2,
   estimate_rows    NUMBER   DEFAULT NULL,
   estimate_percent NUMBER   DEFAULT NULL,
   method_opt       VARCHAR2 DEFAULT NULL,
   partname         VARCHAR2 DEFAULT NULL);
```

パラメータ

表 9-4 ANALYZE_OBJECT プロシージャのパラメータ

パラメータ	説明
type	TABLE、CLUSTER または INDEX のいずれかを設定します。何も設定されていない場合には、ORA-20001 エラーが発生します。
schema	分析するオブジェクトのスキーマ。NULL の場合は現行スキーマ（大 / 小文字区別）が使用されます。
name	分析するオブジェクトの名前（大 / 小文字区別）。
method	ESTIMATE、COMPUTE または DELETE のいずれかを設定します。 ESTIMATE を設定した場合は、estimate_rows または estimate_percent のいずれかを 0（ゼロ）以外に設定する必要があります。
estimate_rows	推定する行数。
estimate_percent	推定する行のパーセント。 estimate_rows が指定されている場合、このパラメータは無視されます。
method_opt	次の書式のメソッド・オプション。 [FOR TABLE] [FOR ALL [INDEXED] COLUMNS] [SIZE n] [FOR ALL INDEXES]
partname	分析する特定のパーティション。

例外

表 9-5 ANALYZE_OBJECT プロシージャの例外

例外	説明
ORA-20000:	権限が不十分であるか、またはオブジェクトが存在しません。
ORA-20001:	オブジェクト・タイプの値が正しくありません。 TABLE、INDEX または CLUSTER のいずれかを設定してください。
ORA-20002:	METHOD には、COMPUTE、ESTIMATE または DELETE のいずれかを設定する必要があります。

IS_TRIGGER_FIRE_ONCE ファンクション

このファンクションは、指定した DML または DDL トリガーが 1 回起動されるように設定されている場合は、TRUE を戻します。それ以外の場合は FALSE を戻します。

トリガーはユーザー・セッションで 1 回起動されますが、次の場合には起動されません。

- Streams 適用プロセスによる変更
- DBMS_APPLY_ADM パッケージの EXECUTE_ERROR プロシージャまたは EXECUTE_ALL_ERRORS プロシージャを使用して、Streams 適用エラーを 1 つ以上実行したことによる変更

注意： DML および DDL トリガーは、1 回のみ起動されます。他のタイプのトリガーはすべて、何回でも起動されます。

関連項目： 9-6 ページ「SET_TRIGGER_FIRING_PROPERTY プロシージャ」

構文

```
DBMS_DDL.IS_TRIGGER_FIRE_ONCE
   trig_owner          IN  VARCHAR2,
   trig_name           IN  VARCHAR2)
RETURN BOOLEAN;
```

パラメータ

表 9-6 IS_TRIGGER_FIRE_ONCE ファンクションのパラメータ

パラメータ	説明
trig_owner	トリガーのスキーマ。
trig_name	トリガーの名前。

SET_TRIGGER_FIRING_PROPERTY プロシージャ

このプロシージャは、指定した DML または DDL トリガーの起動プロパティを設定します。次の変更に対する DML または DDL トリガーの起動プロパティを制御するために使用します。

- Streams 適用プロセスによって適用された変更
- DBMS_APPLY_ADM パッケージ内の EXECUTE_ERROR プロシージャまたは EXECUTE_ALL_ERRORS プロシージャを使用して、Streams 適用エラーを 1 つ以上実行したことによる変更

トリガーの起動プロパティに関して次のいずれかの設定を指定できます。

- トリガーに対する fire_once パラメータが TRUE に設定されている場合、これらのタイプの変更に対してトリガーは起動されません。
- トリガーに対する fire_once パラメータが FALSE に設定されている場合、これらのタイプの変更に対してトリガーは起動されます。

このプロシージャで設定された起動プロパティに関係なく、適用プロセスまたは適用エラー実行以外の方法で変更が行われた場合、トリガーは引き続き起動されます。たとえば、ユーザー・セッションまたはアプリケーションによって変更が行われた場合、起動プロパティに関係なくトリガーは引き続き起動されます。

注意：

- エラー・キューからエラー・トランザクションをデキューし、DBMS_APPLY_ADM パッケージを使用せずに実行した場合は、この実行結果に関連して変更が行われると、トリガー起動プロパティに関係なく、トリガーが起動されます。
- DML および DDL トリガーは、1 回のみ起動されます。他のタイプのトリガーはすべて、何回でも起動されます。

関連項目： 適用プロセスおよびトリガー起動プロパティの制御方法の詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
DBMS_DDL.SET_TRIGGER_FIRING_PROPERTY
   trig_owner         IN  VARCHAR2,
   trig_name          IN  VARCHAR2,
   fire_once          IN  BOOLEAN);
```

パラメータ

表 9-7 SET_TRIGGER_FIRING_PROPERTY プロシージャのパラメータ

パラメータ	説明
trig_owner	設定するトリガーのスキーマ。
trig_name	設定するトリガーの名前。
fire_once	TRUE に設定すると、トリガーは 1 回起動されるように設定されます。デフォルトでは、DML および DDL トリガーに対する fire_once は TRUE に設定されます。 FALSE に設定すると、トリガーは何回でも起動されるように設定されます。

ALTER_TABLE_REFERENCEABLE プロシージャ

このプロシージャは、オブジェクト表を再編成し、参照を再設定します。たとえば、オブジェクト表 FOO と、FOO に格納されたオブジェクトを指す参照が他の表にあるとします。表編成の一部を変更する場合（たとえば、その表を IOT またはパーティション表にする、あるいはデータをより効率的に再編成する場合）は、FOO から FOO2 にすべてのデータをコピーします。次に、alter_table_referenceable プロシージャと alter_table_not_referenceable プロシージャを使用して、既存のすべての参照を、FOO ではなく FOO2 を参照するように再設定します。

構文

```
DBMS_DDL.ALTER_TABLE_REFERENCEABLE
   TABLE_NAME       IN           VARCHAR2,
   TABLE_SCHEMA     IN  DEFAULT  VARCHAR2,
   AFFECTED_SCHEMA  IN  DEFAULT  VARCHAR2;
```

ALTER_TABLE_NOT_REFERENCEABLE プロシージャ

9-7 ページの「ALTER_TABLE_REFERENCEABLE プロシージャ」を参照してください。

構文

```
DBMS_DDL.ALTER_TABLE_NOT_REFERENCEABLE
   TABLE_NAME       IN           VARCHAR2,
   TABLE_SCHEMA     IN  DEFAULT  VARCHAR2,
   AFFECTED_SCHEMA  IN  DEFAULT  VARCHAR2;
```

10

DBMS_DEBUG

DBMS_DEBUG は、Oracle サーバーにおける PL/SQL デバッガ・レイヤー、プローブへの PL/SQL の API です。

この API は、主にサーバー側のデバッガを実装することを目的としており、サーバー側の PL/SQL プログラム・ユニットをデバッグする方法を提供します。

> **注意：** プログラム・ユニットという用語は、各種の PL/SQL プログラム（プロシージャ、ファンクション、パッケージ、パッケージ本体、トリガー、無名ブロック、オブジェクト・タイプまたはオブジェクト・タイプ本体）のことを指します。

この章では、次の項目について説明します。

DBMS_DEBUG の使用方法

サーバー側のコードをデバッグするには、2 つのデータベース・セッションが必要です。1 つはコードをデバッグ・モードで実行するセッション（ターゲット・セッション）、他の 1 つはそのターゲット・セッションを監視するセッション（デバッグ・セッション）です。

ターゲット・セッションは、DBMS_DEBUG で初期化コールを行うことでデバッグ可能になります。この結果、そのセッションにマークが付けられるため、PL/SQL インタプリタがデバッグ・モードで実行され、デバッグ・イベントが生成されます。デバッグ・イベントが生成されると、それらはセッションから転送されます。多くの場合、デバッグ・イベントには戻り通知が必要です。インタプリタは応答があるまで一時停止します。

この間に、デバッグ・セッション自体は DBMS_DEBUG を使用して初期化する必要があります。この結果、監視するターゲット・セッションが識別されます。次に、デバッグ・セッションは DBMS_DEBUG のエントリ・ポイントをコールして、ターゲット・セッションから転送されたイベントを読み込み、ターゲット・セッションと通信します。

DBMS_DEBUG は、PL/SQL コンパイラへのインタフェースは提供しませんが、コンパイラがオプションで生成するデバッグ情報には依存します。デバッグ情報がないと、パラメータまたは変数の値の検証や変更を実行できません。デバッグ情報の生成を確認する方法は 2 通りあります。セッション・スイッチを使用する方法と、個別に再コンパイルする方法です。

セッション・スイッチを設定するには、次の文を入力します。

```
ALTER SESSION SET PLSQL_DEBUG = true;
```

この文によって、コンパイラはセッションの残りの部分に関するデバッグ情報を生成します。既存の PL/SQL は再コンパイルしません。

既存の PL/SQL コードのデバッグ情報を生成するには、次の文のいずれかを使用します（2 番目の文はパッケージまたはタイプの本体を再コンパイルします）。

```
ALTER [PROCEDURE | FUNCTION | PACKAGE | TRIGGER | TYPE] <name> COMPILE DEBUG;
ALTER [PACKAGE | TYPE] <name> COMPILE DEBUG BODY;
```

図 10-1 および図 10-2 は、デバッグ対象のセッションおよびデバッグ・セッションにおける操作の例です。

図 10-1 ターゲット・セッション

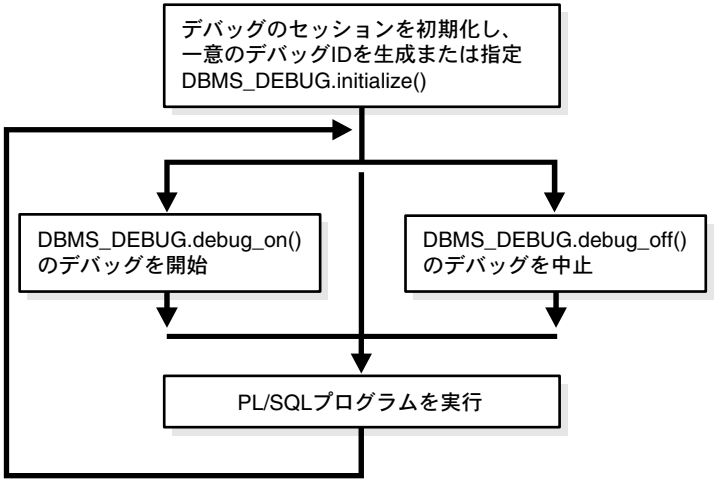


図 10-2 デバッグ・セッション

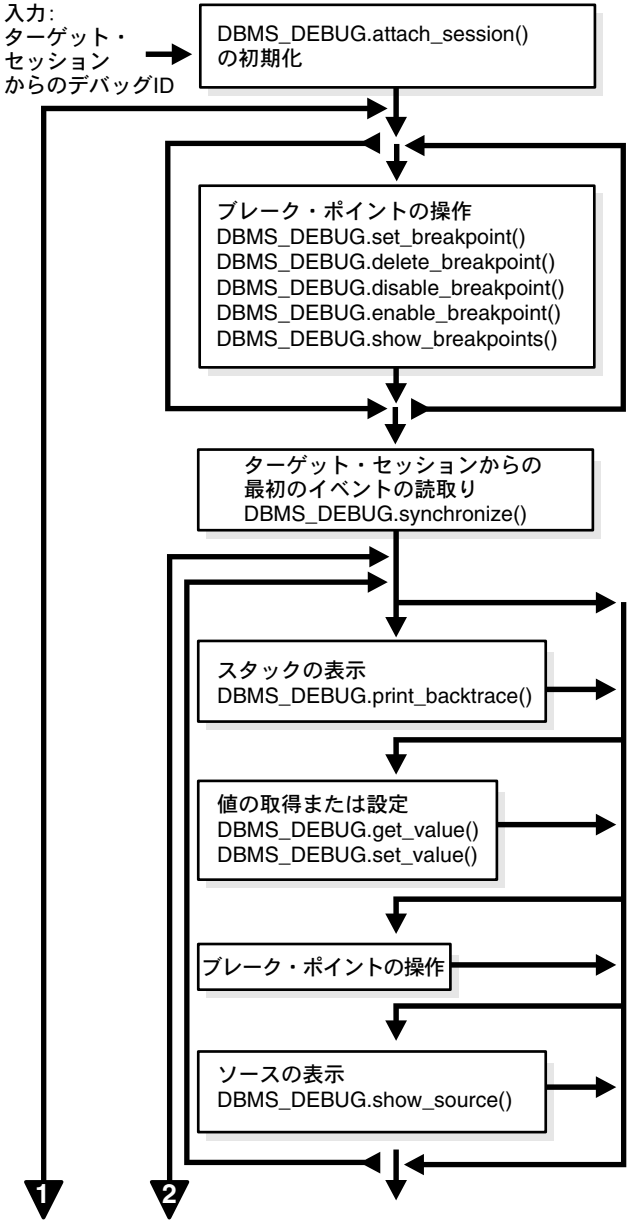
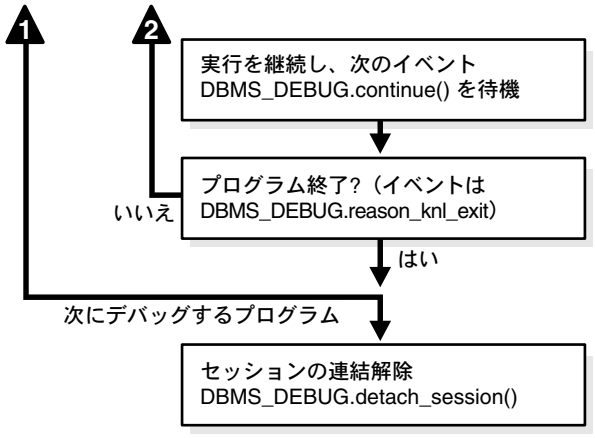


図 10-2 デバッグ・セッション（続き）



インタプリタの管理

インタプリタは、次の場合に実行を一時停止します。

1. インタプリタの起動時。実行前に、遅延ブレーク・ポイントをインストールできるようにするため。
2. 使用可能なブレーク・ポイントを含んだ行に達したとき。
3. 関連のあるイベントが発生した行に達したとき。関連イベントのセットは、`breakflags` パラメータの `DBMS_DEBUG.CONTINUE` に渡されるフラグで指定します。

使用上の注意

セッションの終了

セッション終了のイベントはありません。したがって、ターゲット・セッションが終了していないことを、デバッグ・セッションでチェックして確認する必要があります。ターゲット・セッションが終了した後に DBMS_DEBUG.SYNCHRONIZE をコールすると、タイムアウトするまでデバッグ・セッションがハングアップします。

遅延操作

図では、ターゲット・セッションの前にブレーク・ポイントを設定できることが示されています。これは確かに可能です。この場合、プローブはブレーク・ポイント要求をキャッシュして、最初の同期でターゲット・セッションに送信します。ただし、ブレーク・ポイント要求がこのように遅延した場合は次のようになります。

- SET_BREAKPOINT はブレーク・ポイント番号を設定しません（必要に応じて後で SHOW_BREAKPOINTS から取得できます）。
- SET_BREAKPOINT はブレーク・ポイント要求を検証しません。要求されたソース行が存在しない場合は、同期時にエラーが単的に発生し、ブレーク・ポイントは設定されません。

診断出力

プローブをデバッグするために、DBMS_DEBUG のコールの一部に対して *diagnostics* パラメータが用意されています。これらのパラメータは、RDBMS トレース・ファイルに診断出力を格納するかどうかを指定します。RDBMS トレース・ファイルに出力できない場合、このパラメータは無効になります。

タイプおよび定数

PROGRAM_INFO タイプ

このタイプはプログラムの位置を指定します。プログラム・ユニットの中の行番号を使用します。これは、スタックのバックトレース用およびブレーク・ポイントの設定と検査用に使用されます。読取り専用フィールドは、ブレーク・ポイント操作に関してプローブでは現在無視されています。スタックのバックトレース用のみにプローブが設定します。

タイプ	説明
EntrypointName	ネストされたプロシージャまたはファンクション以外は NULL です。
LibunitType	同じネームスペースを共有するオブジェクト（プロシージャやパッケージ仕様部など）を一意化します。 詳細は、10-9 ページの「Libunit タイプ」を参照してください。

```
TYPE program_info IS RECORD
(
    -- The following fields are used when setting a breakpoint
    Namespace        BINARY_INTEGER,  -- See 'NAMESPACES' section below.
    Name             VARCHAR2(30),    -- name of the program unit
    Owner            VARCHAR2(30),    -- owner of the program unit
    Dblink           VARCHAR2(30),    -- database link, if remote
    Line#            BINARY_INTEGER,
    -- Read-only fields (set by Probe when doing a stack backtrace)
    LibunitType      BINARY_INTEGER,
    EntrypointName   VARCHAR2(30)
);
```

RUNTIME_INFO タイプ

このタイプは、実行プログラムに関するコンテキスト情報を提供します。

```
TYPE runtime_info IS RECORD
(
    Line#            BINARY_INTEGER,  -- (duplicate of program.line#)
    Terminated       BINARY_INTEGER,  -- has the program terminated?
    Breakpoint       BINARY_INTEGER,  -- breakpoint number
    StackDepth       BINARY_INTEGER,  -- number of frames on the stack
    InterpreterDepth BINARY_INTEGER,  -- <reserved field>
    Reason           BINARY_INTEGER,  -- reason for suspension
    Program          program_info     -- source location
);
```

BREAKPOINT_INFO タイプ

このタイプは、ブレーク・ポイントに関して、現在の状態や配置されたプログラム・ユニットなどの情報を提供します。

```
TYPE breakpoint_info IS RECORD
(
   -- These fields are duplicates of 'program_info':
   Name        VARCHAR2(30),
   Owner       VARCHAR2(30),
   DbLink      VARCHAR2(30),
   Line#       BINARY_INTEGER,
   LibunitType BINARY_INTEGER,
   Status      BINARY_INTEGER   -- see breakpoint_status_* below
);
```

INDEX_TABLE タイプ

このタイプは、索引表で使用可能な索引を戻すために、GET_INDEXES で使用されます。

```
TYPE index_table IS table of BINARY_INTEGER INDEX BY BINARY_INTEGER;
```

BACKTRACE_TABLE タイプ

このタイプは、PRINT_BACKTRACE で使用されます。

```
TYPE backtrace_table IS TABLE OF program_info INDEX BY BINARY_INTEGER;
```

BREAKPOINT_TABLE タイプ

このタイプは、SHOW_BREAKPOINTS で使用されます。

```
TYPE breakpoint_table IS TABLE OF breakpoint_info INDEX BY BINARY_INTEGER;
```

VC2_TABLE タイプ

このタイプは、SHOW_SOURCE で使用されます。

```
TYPE vc2_table IS TABLE OF VARCHAR2(90) INDEX BY BINARY_INTEGER;
```

定数

ブレーク・ポイントの状態には次の値があります。

- breakpoint_status_unused －ブレーク・ポイントは使用されていません。

ブレーク・ポイントが使用されている場合、状態は次の値のマスクになります。

- breakpoint_status_active －行ブレーク・ポイント。
- breakpoint_status_disabled －ブレーク・ポイントは現在使用できません。
- breakpoint_status_remote － shadow ブレーク・ポイント（リモート・ブレーク・ポイントのローカル表示）。

ネームスペース

サーバー上のプログラム・ユニットは、異なるネームスペースに常駐しています。ブレーク・ポイントの設定時には、希望するネームスペースを指定してください。

1. Namespace_cursor にはカーソル（無名ブロック）が含まれています。
2. Namespace_pgkspec_or_toplevel には次のものが含まれています。
   - パッケージ仕様部。
   - 他のパッケージ、プロシージャまたはファンクション内にネストされていないプロシージャおよびファンクション。
   - オブジェクト・タイプ。
3. Namespace_pkg_body にはパッケージ本体およびタイプ本体が含まれています。
4. Namespace_trigger にはトリガーが含まれています。

Libunit タイプ

この値は、特定のネームスペースのオブジェクトを一意化するために使用されます。これらの定数は、プローブがスタックのバックトレースを提供しているときに、PROGRAM_INFO で使用されます。

- LibunitType_cursor
- LibunitType_procedure
- LibunitType_function
- LibunitType_package
- LibunitType_package_body
- LibunitType_trigger
- LibunitType_Unknown

ブレーク・フラグ

この値は、クライアントに関連のあるイベントをプローブに通知するために、CONTINUE に対する breakflags パラメータで使用されます。これらのフラグは結合できます。

値	説明
break_next_line	次のソース行でブレークします（コールをスキップ）。
break_any_call	次のソース行でブレークします（コールを開始）。
break_any_return	現行のエントリポイントから戻された後ブレーク（現行のルーチンからコールされたエントリポイントはすべてスキップ）します。

値	説明
break_return	次回のエントリポイントが戻し処理の準備ができた時点でブレークします。(現行のエントリポイントからコールされたエントリポイントが含まれます。インタプリタが Proc2 をコールする Proc1 を実行している場合、break_return は Proc2 の終了時に停止します。)
break_exception	例外が発生したときにブレークします。
break_handler	例外ハンドラが実行されたときにブレークします。
abort_execution	実行を停止し、DBMS_DEBUG.CONTINUE がコールされるとすぐに、'exit' イベントを強制的に実行します。

情報フラグ

このフラグは、info_requested パラメータとして、SYNCHRONIZE、CONTINUE および GET_RUNTIME_INFO に渡されます。

フラグ	説明
info_getStackDepth	スタックの現在の深さを取得します。
info_getBreakpoint	ブレーク・ポイント数を取得します。
info_getLineinfo	プログラム・ユニット情報を取得します。

中断理由

CONTINUE の実行後、プログラムは最後まで実行されるか、または途中の行でブレークします。

理由	説明
reason_none	—
reason_interpreter_starting	インタプリタは起動中です。
reason_breakpoint	ブレーク・ポイントに到達しました。
reason_enter	プロシージャ・エントリ。
reason_return	プロシージャが戻ります。
reason_finish	プロシージャが終了しました。
reason_line	改行に到達しました。
reason_interrupt	割込みが発生しました。

理由	説明
reason_exception	例外が発生しました。
reason_exit	インタプリタは終了処理中です（旧形式）。
reason_knl_exit	カーネルは終了処理中です。
reason_handler	例外ハンドラを起動します。
reason_timeout	タイムアウトが発生しました。
reason_instantiate	インスタンス化ブロック。
reason_abort	インタプリタは異常終了中です。

エラー・コード、例外および変数

エラー・コード

この値は、デバッグ・セッション（SYNCHRONIZE、CONTINUE、SET_BREAKPOINT など）でコールされる様々なファンクションによって戻されます。PL/SQL 例外がクライアント / サーバーおよびサーバー / サーバーの境界を越えて発生した場合は、すべて例外となり、エラー・コードは戻されません。

値	説明
success	正常終了。

GET_VALUE および SET_VALUE が戻すステータスは次のとおりです。

ステータス	説明
error_bogus_frame	該当するエントリポイントがスタックにありません。
error_no_debug_info	プログラムがデバッグ記号なしにコンパイルされました。
error_no_such_object	該当する変数またはパラメータがありません。
error_unknown_type	デバッグ情報を読み取れません。
error_indexed_table	オブジェクトが表で、索引が提供されていない場合に GET_VALUE で戻されます。
error_illegal_index	該当する要素がコレクション内に存在しません。
error_nullcollection	表がアトミック NULL です。
error_nullvalue	値が NULL です。

SET_VALUE が戻すステータスは次のとおりです。

ステータス	説明
error_illegal_value	制約違反。
error_illegal_null	制約違反。
error_value_malformed	指定された値を解読できません。
error_other	その他のエラー。
error_name_incomplete	名前をスカラーに変換できません。

ブレーク・ポイント・ファンクションが戻すステータスは次のとおりです。

ステータス	説明
error_no_such_breakpt	該当するブレーク・ポイントがありません。
error_idle_breakpt	未使用のブレーク・ポイントは使用可能または使用禁止にできません。
error_bad_handle	指定されたプログラムにブレーク・ポイントを設定できません（存在していないか、またはセキュリティ違反です）。

一般的なエラー・コード（多数の DBMS_DEBUG サブプログラムが戻す）は次のとおりです。

ステータス	説明
error_unimplemented	機能が実装されていません。
error_deferred	プログラムが実行されていません。操作は延期されました。
error_exception	サーバー上の DBMS_DEBUG またはプローブ・パッケージで例外が発生しました。
error_communication	タイムアウト以外のエラーが発生しました。
error_timeout	タイムアウトが発生しました。

例外

例外	説明
illegal_init	INITIALIZE の前に DEBUG_ON がコールされました。

次の例外は、プロシージャ SELF_CHECK によって発生します。

例外	説明
pipe_creation_failure	パイプを作成できませんでした。
pipe_send_failure	パイプにデータを書き込めませんでした。
pipe_receive_failure	パイプからデータを読み込めませんでした。
pipe_datatype_mismatch	パイプ内のデータ・タイプが正しくありませんでした。
pipe_data_error	データがパイプ内で混同されていました。

変数

例外	説明
default_timeout	タイムアウトの値（両方のセッションが使用します）。最小許容値は 1 秒です。この値が 0（ゼロ）に設定された場合は、大きい値（3600）が使用されます。

共通セクションおよびデバッグ・セッション・セクション

共通セクション

次に示すサブプログラムは、ターゲットまたはデバッグ・セッションのいずれでもコールできます。

- PROBE_VERSION プロシージャ
- SELF_CHECK プロシージャ
- SET_TIMEOUT ファンクション

デバッグ・セッション・セクション

次のサブプログラムは、デバッグ・セッションでのみ実行してください。

- ATTACH_SESSION プロシージャ
- SYNCHRONIZE ファンクション
- SHOW_SOURCE プロシージャ
- PRINT_BACKTRACE プロシージャ
- CONTINUE ファンクション
- SET_BREAKPOINT ファンクション
- DELETE_BREAKPOINT ファンクション
- DISABLE_BREAKPOINT ファンクション
- ENABLE_BREAKPOINT ファンクション
- SHOW_BREAKPOINTS プロシージャ
- GET_VALUE ファンクション
- SET_VALUE ファンクション
- DETACH_SESSION プロシージャ
- GET_RUNTIME_INFO ファンクション
- GET_INDEXES ファンクション
- EXECUTE プロシージャ

OER ブレーク・ポイント

PL/SQL プログラム内で宣言される例外は、ユーザー定義の例外として認識されています。この他に、Oracle カーネルから戻される Oracle エラー（OER）があります。この 2 つのメカニズムを結合するために、PL/SQL はユーザー定義の例外を OER に変換する exception_init プラグマを提供しています。この処理には PL/SQL ハンドラが使用され、PL/SQL エンジンは、OER を Oracle カーネルに戻すことができます。現行のリリースでは、OER に関する使用可能な情報はその番号のみです。2 つのユーザー定義例外が同じ OER に exception_init されると、区別できません。

DBMS_DEBUG サブプログラムの要約

表 10-1 DBMS_DEBUG パッケージのサブプログラム

サブプログラム	説明
「PROBE_VERSION プロシージャ」10-17 ページ	サーバー上の DBMS_DEBUG のバージョン番号を戻します。
「SELF_CHECK プロシージャ」10-17 ページ	内部一貫性チェックを実行します。
「SET_TIMEOUT ファンクション」10-18 ページ	タイムアウト値を設定します。
「INITIALIZE ファンクション」10-19 ページ	ターゲット・セッションのデバッグ ID を設定します。
「DEBUG_ON プロシージャ」10-20 ページ	デバッグ・モードをオンにします。
「DEBUG_OFF プロシージャ」10-20 ページ	デバッグ・モードをオフにします。
「ATTACH_SESSION プロシージャ」10-21 ページ	デバッグ・セッションにターゲット・デバッグ ID に関する情報を通知します。
「SYNCHRONIZE ファンクション」10-21 ページ	プログラムの実行開始を待機します。
「SHOW_SOURCE プロシージャ」10-22 ページ	プログラム・ソースをフェッチします。
「PRINT_BACKTRACE プロシージャ」10-25 ページ	スタックのバックトレースを印刷します。
「CONTINUE ファンクション」10-26 ページ	ターゲット・プログラムの実行を継続します。

表 10-1 DBMS_DEBUG パッケージのサブプログラム（続き）

サブプログラム	説明
「SET_BREAKPOINT ファンクション」10-27 ページ	プログラム・ユニットにブレーク・ポイントを設定します。
「DELETE_BREAKPOINT ファンクション」10-28 ページ	ブレーク・ポイントを削除します。
「DISABLE_BREAKPOINT ファンクション」10-29 ページ	ブレーク・ポイントを使用禁止にします。
「ENABLE_BREAKPOINT ファンクション」10-30 ページ	既存のブレーク・ポイントをアクティブにします。
「SHOW_BREAKPOINTS プロシージャ」10-31 ページ	現行のブレーク・ポイントのリストを戻します。
「GET_VALUE ファンクション」10-32 ページ	現在実行中のプログラムから値を取得します。
「SET_VALUE ファンクション」10-34 ページ	現在実行中のプログラムに値を設定します。
「DETACH_SESSION プロシージャ」10-37 ページ	ターゲット・プログラムのデバッグを停止します。
「GET_RUNTIME_INFO ファンクション」10-37 ページ	現行のプログラムに関する情報を戻します。
「GET_INDEXES ファンクション」10-38 ページ	索引表に対する一連の索引を戻します。
「EXECUTE プロシージャ」10-39 ページ	ターゲット・セッションで SQL または PL/SQL を実行します。

PROBE_VERSION プロシージャ

このプロシージャは、サーバー上の DBMS_DEBUG のバージョン番号を戻します。

構文

```
DBMS_DEBUG.PROBE_VERSION (
   major out BINARY_INTEGER,
   minor out BINARY_INTEGER);
```

パラメータ

表 10-2 PROBE_VERSION プロシージャのパラメータ

パラメータ	説明
major	バージョン番号。
minor	リリース番号。機能が追加されるたびに増加します。

SELF_CHECK プロシージャ

このプロシージャは、内部一貫性チェックを実行します。SELF_CHECK は、プローブ・プロセスが通信可能かどうかを確認するために、通信テストも実行します。

SELF_CHECK が正常に終了しなかった場合は、このサーバーにインストールされている DBMS_DEBUG のバージョンが適切ではない可能性があります。解決方法は、正しいバージョンをインストールすることです（pbload.sql を実行すると、DBMS_DEBUG およびその他の関連パッケージがロードされます）。

構文

```
DBMS_DEBUG.SELF_CHECK (
   timeout IN binary_integer := 60);
```

パラメータ

表 10-3 SELF_CHECK プロシージャのパラメータ

パラメータ	説明
timeout	通信テストに使用するタイムアウト時間。デフォルトは 60 秒です。

例外

表 10-4 SELF_CHECK プロシージャの例外

例外	説明
OER-6516	プローブのバージョンに一貫性がありません。
pipe_creation_failure	パイプを作成できませんでした。
pipe_send_failure	パイプにデータを書き込めませんでした。
pipe_receive_failure	パイプからデータを読み込めませんでした。
pipe_datatype_mismatch	パイプ内のデータ・タイプが正しくありませんでした。
pipe_data_error	データがパイプ内で混同されていました。

これらはすべて致命的な例外です。プローブの正常な実行を妨げる重大な問題であることを示しています。

SET_TIMEOUT ファンクション

このファンクションは、タイムアウト値を設定し、新しいタイムアウト値を戻します。

構文

```
DBMS_DEBUG.SET_TIMEOUT (
   timeout BINARY_INTEGER)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-5 SET_TIMEOUT ファンクションのパラメータ

パラメータ	説明
timeout	ターゲットとデバッグ・セッション間の通信に使用されるタイムアウト。

TARGET SESSION セクション

次のサブプログラムは、ターゲット・セッション（デバッグ対象のセッション）で実行されます。

- INITIALIZE ファンクション
- DEBUG_ON プロシージャ
- DEBUG_OFF プロシージャ

INITIALIZE ファンクション

このファンクションは、デバッグ用にターゲット・セッションを初期化します。

構文

```
DBMS_DEBUG.INITIALIZE (
   debug_session_id  IN VARCHAR2       := NULL,
   diagnostics       IN BINARY_INTEGER := 0)
  RETURN VARCHAR2;
```

パラメータ

表 10-6 INITIALIZE ファンクションのパラメータ

パラメータ	説明
debug_session_id	セッション ID の名前。NULL の場合は、一意の ID が生成されます。
diagnostics	診断出力をトレースファイルにダンプするかどうかを示します。 0 = （デフォルト）診断出力なし。 1 = 診断出力あり。

戻り値

新たに登録されたデバッグ・セッション ID（デバッグ ID）

DEBUG_ON プロシージャ

このプロシージャは、すべての PL/SQL がデバッグ・モードで実行されるように、ターゲット・セッションにマークを設定します。この処理は、デバッグの開始前に実行する必要があります。

構文

```
DBMS_DEBUG.DEBUG_ON (
   no_client_side_plsql_engine BOOLEAN := TRUE,
   immediate                   BOOLEAN := FALSE);
```

パラメータ

表 10-7 DEBUG_ON プロシージャのパラメータ

パラメータ	説明
no_client_side_plsql_engine	デバッグ・セッションがクライアント側の PL/SQL エンジンから起動されていないかぎり、デフォルト値のままにしてください。
immediate	TRUE の場合、インタプリタは標準モードで処理を継続せずに、コール中にすぐにデバッグ・モードに切り替わります。

注意： immediate が TRUE の場合、デバッグ・セッションは待機する必要があります。

DEBUG_OFF プロシージャ

このプロシージャは、そのセッションでデバッグを起動する必要がなくなったことをターゲット・セッションに通知します。セッションの終了前にこのファンクションをコールする必要はありません。

構文

```
DBMS_DEBUG.DEBUG_OFF;
```

使用上の注意

サーバーは、このエントリポイントを特別には処理しません。したがって、このエントリポイントをデバッグしようとします。

ATTACH_SESSION プロシージャ

このプロシージャは、ターゲット・プログラムに関する情報をデバッグ・セッションに通知します。

構文

```
DBMS_DEBUG.ATTACH_SESSION (
   debug_session_id  IN VARCHAR2,
   diagnostics       IN BINARY_INTEGER := 0);
```

パラメータ

表 10-8 ATTACH_SESSION プロシージャのパラメータ

パラメータ	説明
debug_session_id	ターゲット・セッションの INITIALIZE コールで取得したデバッグ ID。
diagnostics	0（ゼロ）以外の場合に診断出力を生成します。

SYNCHRONIZE ファンクション

このファンクションは、ターゲット・プログラムがイベントを通知するまで待機します。info_requested が NULL でない場合は、GET_RUNTIME_INFO がコールされます。

構文

```
DBMS_DEBUG.SYNCHRONIZE (
   run_info       OUT  runtime_info,
   info_requested IN   BINARY_INTEGER := NULL)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-9 SYNCHRONIZE ファンクションのパラメータ

パラメータ	説明
run_info	プログラムに関する情報を書き込むためのデータ構造。デフォルトでは、実行中のプログラムおよび一時停止している行に関する情報が含まれます。
info_requested	デフォルト（info_getStackDepth + info_getLineInfo）以外の情報を要求するためのオプションのビット・フィールド。0（ゼロ）は、情報を要求しないことを意味します。 10-10 ページの「情報フラグ」を参照してください。

戻り値

表 10-10 SYNCHRONIZE ファンクションの戻り値

戻り値	説明
success	
error_timeout	プログラムが実行を開始する前にタイムアウトしました。
error_communication	その他の通信エラー。

SHOW_SOURCE プロシージャ

（実行されているプログラムの）ソース・コードを取得する最適な方法は、SQL を使用することです。たとえば、次のようにします。

```
DECLARE
   info DBMS_DEBUG.runtime_info;
BEGIN
   -- call DBMS_DEBUG.SYNCHRONIZE, CONTINUE,
   -- or GET_RUNTIME_INFO to fill in 'info'
   SELECT text INTO <buffer> FROM all_source
   WHERE owner = info.Program.Owner
     AND name  = info.Program.Name
     AND line  = info.Line#;
END;
```

ただし、このコードは非永続プログラム（無名ブロックやトリガー起動ブロックなど）では機能しません。非永続プログラムの場合は、SHOW_SOURCE をコールしてください。2 通りの方法があり、1 つはソース行の索引表を戻し、他の 1 つはパック（およびフォーマット）されたバッファを戻します。

SHOW_SOURCE プロシージャは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.SHOW_SOURCE (
   first_line  IN   BINARY_INTEGER,
   last_line   IN   BINARY_INTEGER,
   source      OUT  vc2_table);
```

パラメータ

表 10-11 SHOW_SOURCE プロシージャのパラメータ

パラメータ	説明
first_line	フェッチする最初の行の行番号（PL/SQL プログラムは、常に行 1 から始まり、途中の行が抜けることはありません）。
last_line	フェッチする最後の行の行番号。プログラムの行数を超えると行はフェッチされません。
source	結果の表。行番号で索引が設定されている場合があります。

戻り値

ソース行の索引表。ソース行は、first_line から格納されます。エラーが発生した場合は、空の表が戻されます。

使用上の注意

次の 2 番目の SHOW_SOURCE の書式は、フォーマット済みバッファに行番号の付いたソースを戻します。索引表を使用するより処理は高速ですが、すべてのソースがフェッチされるとはかぎりません。

ソースがバッファ長（buflen）にすべて格納できなかった場合は、GET_MORE_SOURCE プロシージャを使用して追加のピースを取り出せます（pieces は、取り出す必要のある追加ピースの数を戻します）。

構文

```
DBMS_DEBUG.SHOW_SOURCE (
   first_line   IN     BINARY_INTEGER,
   last_line    IN     BINARY_INTEGER,
   window       IN     BINARY_INTEGER,
   print_arrow  IN     BINARY_INTEGER,
   buffer       IN OUT VARCHAR2,
   buflen       IN     BINARY_INTEGER,
   pieces       OUT    BINARY_INTEGER);
```

パラメータ

表 10-12 SHOW_SOURCE プロシージャのパラメータ

パラメータ	説明
first_line	出力を開始する行番号。
last_line	出力を終了する行番号。
window	行の ' ウィンドウ ' (現行のソース行の概数)。
print_arrow	0（ゼロ）以外の場合は、カレント行の前に矢印が出力されます。
buffer	ソース・リストを格納するバッファ。
buflen	バッファの長さ。
pieces	指定したバッファにすべてのソースを格納できない可能性がある場合は、0（ゼロ）以外の値が設定されます。

PRINT_BACKTRACE プロシージャ

このプロシージャは、現在の実行スタックのバックトレース・リストを出力します。これは、プログラムが実行中の場合のみコールしてください。

PRINT_BACKTRACE プロシージャは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.PRINT_BACKTRACE (
  listing IN OUT VARCHAR2);
```

パラメータ

表 10-13 PRINT_BACKTRACE プロシージャのパラメータ

パラメータ	説明
listing	埋込み改行付きのフォーマット済み文字バッファ。

構文

```
DBMS_DEBUG.PRINT_BACKTRACE (
   backtrace OUT backtrace_table);
```

パラメータ

表 10-14 PRINT_BACKTRACE プロシージャのパラメータ

パラメータ	説明
backtrace	バックトレース・エントリ 1 ベースの索引表。現在実行中のプロシージャは、表の最終エントリです（つまり、フレーム番号は、GET_VALUE が使用しているものと同一です）。エントリ 1 は、スタック上で最も古いプロシージャです。

CONTINUE ファンクション

このファンクションは、所定のブレーク・フラグ（関連のあるイベントのマスク）をターゲット・プロセスのプローブに渡します。プローブに、ターゲット・プロセスの実行を継続するように通知し、ターゲット・プロセスが実行を終了するか、またはイベントを通知するまで待機します。

info_requested が NULL でない場合は、GET_RUNTIME_INFO をコールします。

構文

```
DBMS_DEBUG.CONTINUE (
   run_info       IN OUT runtime_info,
   breakflags     IN     BINARY_INTEGER,
   info_requested IN     BINARY_INTEGER := NULL)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-15 CONTINUE ファンクションのパラメータ

パラメータ	説明
run_info	プログラムの状態に関する情報。
breakflags	関連のあるイベントのマスク。10-9 ページの「ブレーク・フラグ」を参照してください。
info_requested	プログラムが停止したときに、run_info に戻される必要のある情報。10-10 ページの「情報フラグ」を参照してください。

戻り値

表 10-16 CONTINUE ファンクションの戻り値

戻り値	説明
success	
error_timeout	プログラムが実行を開始する前にタイムアウトしました。
error_communication	その他の通信エラー。

SET_BREAKPOINT ファンクション

このファンクションは、現行セッションを持続するためのブレーク・ポイントをプログラム・ユニットに設定します。ターゲット・プログラムがブレーク・ポイントに到達すると、実行は一時停止します。

構文

```
DBMS_DEBUG.SET_BREAKPOINT (
   program     IN  program_info,
   line#       IN  BINARY_INTEGER,
   breakpoint# OUT BINARY_INTEGER,
   fuzzy       IN  BINARY_INTEGER := 0,
   iterations  IN  BINARY_INTEGER := 0)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-17 SET_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
program	ブレーク・ポイントが設定されるプログラム・ユニットに関する情報（バージョン 2.1 以降では、ネームスペース、名前、所有者および DB リンクを NULL に設定でき、この場合のブレーク・ポイントは、現在実行中のプログラム・ユニットに設定されます）。
line#	ブレーク・ポイントが設定される行。
breakpoint#	正常に完了すると、ブレーク・ポイントを参照するための一意のブレーク・ポイント番号が含まれます。
fuzzy	指定した行に実行可能コードがない場合にのみ適用されます。 0（ゼロ）の場合は、error_illegal_line が戻されます。 1 の場合は、ブレーク・ポイントを設定する行が指定行から順方向に検索されます。 -1 の場合は、ブレーク・ポイントを設定する行が指定行から逆方向に検索されます。
iterations	このブレーク・ポイントを通知するまでの待機回数。

注意： fuzzy および iterations パラメータは、まだ実装されていません。

戻り値

表 10-18 SET_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_illegal_line	この行にブレーク・ポイントは設定できません。
error_bad_handle	該当するプログラム・ユニットが存在しません。

DELETE_BREAKPOINT ファンクション

このファンクションはブレーク・ポイントを削除します。

構文

```
DBMS_DEBUG.DELETE_BREAKPOINT (
   breakpoint IN BINARY_INTEGER)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-19 DELETE_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
breakpoint	以前の SET_BREAKPOINT コールから戻されたブレーク・ポイント番号。

戻り値

表 10-20 DELETE_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_no_such_breakpt	該当するブレーク・ポイントが存在しません。
error_idle_breakpt	未使用のブレーク・ポイントは削除できません。
error_stale_breakpt	ブレーク・ポイントが設定された後にプログラム・ユニットが再定義されました。

DISABLE_BREAKPOINT ファンクション

このファンクションは、既存のブレーク・ポイントを使用禁止にしますが、削除しないでそのまま残します。

構文

```
DBMS_DEBUG.DISABLE_BREAKPOINT (
   breakpoint IN BINARY_INTEGER)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-21 DISABLE_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
breakpoint	以前の SET_BREAKPOINT コールから戻されたブレーク・ポイント番号。

戻り値

表 10-22 DISABLE_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_no_such_breakpt	該当するブレーク・ポイントが存在しません。
error_idle_breakpt	未使用のブレーク・ポイントは使用禁止にできません。

ENABLE_BREAKPOINT ファンクション

このファンクションは、使用禁止の逆の処理を実行します。以前に使用禁止にしたブレーク・ポイントを使用可能にします。

構文

```
DBMS_DEBUG.ENABLE_BREAKPOINT (
   breakpoint IN BINARY_INTEGER)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-23 ENABLE_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
breakpoint	以前の SET_BREAKPOINT コールから戻されたブレーク・ポイント番号。

戻り値

表 10-24 ENABLE_BREAKPOINT ファンクションの戻り値

戻り値	説明
success	
error_no_such_breakpt	該当するブレーク・ポイントが存在しません。
error_idle_breakpt	未使用のブレーク・ポイントは使用可能にできません。

SHOW_BREAKPOINTS プロシージャ

このプロシージャは、現在のブレーク・ポイントのリストを戻します。
SHOW_BREAKPOINTS プロシージャは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.SHOW_BREAKPOINTS (
   listing    IN OUT VARCHAR2);
```

パラメータ

表 10-25 SHOW_BREAKPOINTS プロシージャのパラメータ

パラメータ	説明
listing	ブレーク・ポイントのフォーマット済みバッファ（改行を含む）。

構文

```
DBMS_DEBUG.SHOW_BREAKPOINTS (
   listing  OUT breakpoint_table);
```

パラメータ

表 10-26 SHOW_BREAKPOINTS プロシージャのパラメータ

パラメータ	説明
listing	ブレーク・ポイント・エントリの索引表。ブレーク・ポイント番号は、表に対する索引で示されます。ブレーク・ポイント番号は 1 から始まり、削除されると再使用されます。

GET_VALUE ファンクション

このファンクションは、現在実行中のプログラムから値を取得します。GET_VALUE ファンクションは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.GET_VALUE (
   variable_name  IN  VARCHAR2,
   frame#         IN  BINARY_INTEGER,
   scalar_value   OUT VARCHAR2,
   format         IN  VARCHAR2 := NULL)
RETURN BINARY_INTEGER;
```

パラメータ

表 10-27 GET_VALUE ファンクションのパラメータ

パラメータ	説明
variable_name	変数またはパラメータの名前。
frame#	値が存在するフレーム。0（ゼロ）の場合は現行のプロシージャです。
scalar_value	値。
format	使用するオプションの日付書式（指定する必要がある場合）。

戻り値

表 10-28 GET_VALUE ファンクションの戻り値

戻り値	説明
success	
error_bogus_frame	フレームが存在しません。
error_no_debug_info	エントリポイントにデバッグ情報がありません。
error_no_such_object	variable_name が frame# に存在しません。
error_unknown_type	デバッグ情報内のタイプ情報が判読不能です。
error_nullvalue	値が NULL です。
error_indexed_table	オブジェクトは表ですが、索引が提供されていません。

次の GET_VALUE 構文は、パッケージ変数フェッチ用です。フレーム番号のかわりに、変数を含んだパッケージを説明するハンドルを使用します。

構文

```
DBMS_DEBUG.GET_VALUE (
   variable_name  IN  VARCHAR2,
   handle         IN  program_info,
   scalar_value   OUT VARCHAR2,
   format         IN  VARCHAR2 := NULL)
RETURN BINARY_INTEGER;
```

パラメータ

表 10-29 GET_VALUE ファンクションのパラメータ

パラメータ	説明
variable_name	変数またはパラメータの名前。
handle	変数を含んだパッケージの説明。
scalar_value	値。
format	使用するオプションの日付書式（指定する必要がある場合）。

戻り値

表 10-30 GET_VALUE ファンクションの戻り値

戻り値	説明
error_no_such_object	次のいずれかです。 － パッケージが存在しません。 － パッケージがインスタンス化されていません。 － ユーザーにパッケージをデバッグする権限がありません。 －オブジェクトがパッケージ内に存在しません。
error_indexed_table	オブジェクトは表ですが、索引が提供されていません。

例

この例は、スキーマ SCOTT 内の変数 VAR を含んだ任意のパッケージ PACK の値を取得する方法を示しています。

```
DECLARE
   handle     dbms_debug.program_info;
   resultbuf  VARCHAR2(500);
   retval     BINARY_INTEGER;
BEGIN
   handle.Owner     := 'SCOTT';
   handle.Name      := 'PACK';
   handle.namespace := dbms_debug.namespace_pkgspec_or_toplevel;
   retval           := dbms_debug.get_value('VAR', handle, resultbuf, NULL);
END;
```

SET_VALUE ファンクション

このファンクションは、現在実行中のプログラムに値を設定します。SET_VALUE ファンクションは 2 種類、オーバーロードされています。

構文

```
DBMS_DEBUG.SET_VALUE (
   frame#               IN binary_integer,
   assignment_statement IN varchar2)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-31 SET_VALUE ファンクションのパラメータ

パラメータ	説明
frame#	値を設定するフレーム。0（ゼロ）は現在実行中のフレームを意味します。
assignment_statement	値を設定するために実行する代入文（有効な PL/SQL である必要があります）。たとえば、'x := 3;' のように指定します。 このリリースでは、スカラー値のみサポートされています。代入文の右辺はスカラーである必要があります。

戻り値

表 10-32 SET_VALUE ファンクションの戻り値

戻り値	説明
success	—
error_illegal_value	指定した値を設定できません。
error_illegal_null	オブジェクト・タイプは 'NOT NULL' で指定されているため、NULL は設定できません。
error_value_malformed	値がスカラーではありません。
error_name_incomplete	代入文をスカラーに変換できません。たとえば、'x := 3;'（x がレコードの場合）のように指定します。

次の書式の SET_VALUE は、パッケージ変数の値を設定します。

構文

```
DBMS_DEBUG.SET_VALUE (
   handle               IN program_info,
   assignment_statement IN VARCHAR2)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-33 SET_VALUE ファンクションのパラメータ

パラメータ	説明
handle	変数を含んだパッケージの説明。
assignment_statement	値を設定するために実行する代入文（有効な PL/SQL である必要があります）。たとえば、'x := 3;' のように指定します。 このリリースでは、スカラー値のみサポートされています。代入文の右辺はスカラーである必要があります。

表 10-34 SET_VALUE ファンクションの戻り値

戻り値	説明
error_no_such_object	次のいずれかです。 – パッケージが存在しません。 – パッケージがインスタンス化されていません。 – ユーザーにパッケージをデバッグする権限がありません。 – オブジェクトがパッケージ内に存在しません。

PL/SQL コンパイラが一時ファイルを使用してパッケージ変数にアクセスし、プローブがこのような一時ファイルの更新を保証しない場合があります。ほとんど発生しませんが、SET_VALUE を使用したパッケージ変数の変更が及ばない行があります。

例

SCOTT.PACK.var の値を 6 に設定する方法は次のとおりです。

```
DECLARE
   handle  dbms_debug.program_info;
   retval  BINARY_INTEGER;
BEGIN
   handle.Owner     := 'SCOTT';
   handle.Name      := 'PACK';
   handle.namespace := dbms_debug.namespace_pkgspec_or_toplevel;
   retval           := dbms_debug.set_value(handle, 'var := 6;');
END;
```

DETACH_SESSION プロシージャ

このプロシージャは、ターゲット・プログラムのデバッグを停止します。このプロシージャはいつでもコール可能ですが、デバッグ・セッションの連結が解除されたことはターゲット・セッションに通知されず、ターゲット・セッションの実行は中断されません。したがって、ターゲット・セッションが独自にハングアップしないように注意してください。

構文

```
DBMS_DEBUG.DETACH_SESSION;
```

GET_RUNTIME_INFO ファンクション

このファンクションは、現行のプログラムに関する情報を戻します。これは、SYNCHRONIZE または CONTINUE の info_requested パラメータが 0（ゼロ）に設定された場合のみ必要です。

> **注意：** このファンクションは、現在クライアント側の PL/SQL でのみ使用されます。

構文

```
DBMS_DEBUG.GET_RUNTIME_INFO (
   info_requested  IN  BINARY_INTEGER,
   run_info        OUT runtime_info)
  RETURN BINARY_INTEGER;
```

パラメータ

表 10-35 GET_RUNTIME_INFO ファンクションのパラメータ

パラメータ	説明
info_requested	プログラムが停止したときに、run_info に戻される必要のある情報。10-10 ページの「情報フラグ」を参照してください。
run_info	プログラムの状態に関する情報。

GET_INDEXES ファンクション

変数またはパラメータの名前を指定すると、索引表の場合はその一連の索引を戻します。索引表以外の場合はエラーが戻されます。

構文

```
DBMS_DEBUG.GET_INDEXES (
   varname    IN  VARCHAR2,
   frame#     IN  BINARY_INTEGER,
   handle     IN  program_info,
   entries    OUT index_table)
RETURN BINARY_INTEGER;
```

パラメータ

表 10-36 GET_INDEXES ファンクションのパラメータ

パラメータ	説明
varname	索引情報を取得する変数の名前。
frame#	変数またはパラメータが常駐しているフレームの番号。パッケージ変数の場合は NULL です。
handle	パッケージの説明（オブジェクトがパッケージ変数の場合）。
entries	1 ベースの索引表。NULL 以外の場合は、entries(1) にその行の 1 番目の索引が含まれ、entries(2) に 2 番目の索引、以下同様に索引が含まれます。

戻り値

表 10-37 GET_INDEXES ファンクションの戻り値

戻り値	説明
error_no_such_object	次のいずれかです。 — パッケージが存在しません。 — パッケージがインスタンス化されていません。 — ユーザーにパッケージをデバッグする権限がありません。 — オブジェクトがパッケージ内に存在しません。

EXECUTE プロシージャ

このプロシージャは、ターゲット・セッションで SQL または PL/SQL コードを実行します。ターゲット・セッションは、ブレーク・ポイント（またはその他のイベント）で待機中であるとみなされます。デバッグ・セッションで DBMS_DEBUG.EXECUTE がコールされ、ターゲット・セッションにコードの実行を要求します。

構文

```
DBMS_DEBUG.EXECUTE (
   what         IN VARCHAR2,
   frame#       IN BINARY_INTEGER,
   bind_results IN BINARY_INTEGER,
   results      IN OUT NOCOPY dbms_debug_vc2coll,
   errm         IN OUT NOCOPY VARCHAR2);
```

パラメータ

表 10-38 EXECUTE プロシージャのパラメータ

パラメータ	説明
what	実行する SQL または PL/SQL のソース。
frame#	コードを実行するコンテキスト。現在は -1（グローバル・コンテキスト）のみサポートされています。
bind_results	ターゲット・セッションから値を戻すために、ソースを results に結合するかどうかを指定します。
	0 = 結合しない。
	1 = 結合する。
results	結果を格納するコレクション（bind_results が 0（ゼロ）以外の場合）。
errm	エラーが発生した場合はエラー・メッセージ、それ以外の場合は NULL です。

例 1

この例は SQL 文の実行例です。結果は戻されません。

```
DECLARE
   coll sys.dbms_debug_vc2coll; -- results (unused)
   errm VARCHAR2(100);
BEGIN
   dbms_debug.execute('insert into emp(ename,empno,deptno) ' ||
                      'values(''LJE'', 1, 1)',
                      -1, 0, coll, errm);
END;
```

例 2

この例は PL/SQL ブロックの実行例で、結果は戻されません。ブロックは自律型トランザクションで、表に挿入された値はデバッグ・セッションで参照できます。

```
DECLARE
   coll sys.dbms_debug_vc2coll;
   errm VARCHAR2(100);
BEGIN
   dbms_debug.execute(
       'DECLARE PRAGMA autonomous_transaction; ' ||
       'BEGIN ' ||
       '   insert into emp(ename, empno, deptno) ' ||
       '   values(''LJE'', 1, 1); ' ||
       ' COMMIT; ' ||
       'END;',
       -1, 0, coll, errm);
END;
```

例 3

この例は PL/SQL ブロックの実行例で、結果がいくつか戻されます。

```
DECLARE
   coll sys.dbms_debug_vc2coll;
   errm VARCHAR2(100);
BEGIN
   dbms_debug.execute(
      'DECLARE ' ||
      '   pp SYS.dbms_debug_vc2coll := SYS.dbms_debug_vc2coll(); ' ||
      '   x  PLS_INTEGER; ' ||
      '   i  PLS_INTEGER := 1; ' ||
      'BEGIN ' ||
      '   SELECT COUNT(*) INTO x FROM emp; ' ||
      '   pp.EXTEND(x * 6); ' ||
      '   FOR c IN (SELECT * FROM emp) LOOP ' ||
      '      pp(i) := ''Ename: '' || c.ename; i := i+1; ' ||
      '      pp(i) := ''Empno: '' || c.empno; i := i+1; ' ||
      '      pp(i) := ''Job:   '' || c.job;   i := i+1; ' ||
      '      pp(i) := ''Mgr:   '' || c.mgr;   i := i+1; ' ||
      '      pp(i) := ''Sal:   '' || c.sal;   i := i+1; ' ||
      '      pp(i) := null;                   i := i+1; ' ||
      '   END LOOP; ' ||
      '   :1 := pp;' ||
      'END;',
       -1, 1, coll, errm);
   each := coll.FIRST;
   WHILE (each IS NOT NULL) LOOP
      dosomething(coll(each));
      each := coll.NEXT(each);
   END LOOP;
END;
```

PRINT_INSTANTIATIONS プロシージャ

このプロシージャは、現行のセッションでインスタンス化されたパッケージのリストを戻します。

構文

```
DBMS_DEBUG.PRINT_INSTANTIATIONS (
   pkgs  IN OUT NOCOPY backtrace_table,
   flags  IN BINARY_INTEGER);
```

パラメータ

表 10-39 PRINT_INSTANTIATIONS プロシージャのパラメータ

パラメータ	説明
pkgs（OUT）	インスタンス化されたパッケージ。
flags	オプションのビットマスク： ■ 1 －仕様部の表示。 ■ 2 －本体の表示。 ■ 4 －ローカル・インスタンス化の表示。 ■ 8 －リモート・インスタンス化の表示（まだ実装されていません）。 ■ 16 －高速ジョブの実行。デバッグ情報が存在しているか、またはライブラリ・ユニットがシュリンクラップされているかどうかはテストされません。

例外

no_target_program: ターゲット・セッションは現在実行されていません。

使用上の注意

pkgs の戻り値には、各インスタンス化されたパッケージの program_info が含まれます。有効なフィールドは、Namespace、Name、Owner および LibunitType です。

また、Line# には次のビットマスクが含まれます。

- 1 －ライブラリ・ユニットにデバッグ情報が含まれています。
- 2 －ライブラリ・ユニットはシュリンクラップされています。

TARGET_PROGRAM_RUNNING プロシージャ

このプロシージャは、ターゲット・セッションが現在ストアド・プロシージャを実行中の場合は TRUE、実行していない場合は FALSE を戻します。

構文

```
FUNCTION target_program_running RETURN BOOLEAN;
```

PING プロシージャ

このプロシージャは、ターゲット・セッションがタイムアウトしないように ping します。ターゲット・セッションで実行が中断したとき、ブレーク・ポイントなどでこのプロシージャを使用します。

timeout_behavior が retry_on_timeout に設定されている場合、このプロシージャは不要です。

構文

```
DBMS_DEBUG.PING;
```

例外

ターゲット・プログラムがない場合、またはターゲット・セッションがデバッグ・セッションからの入力を待機していない場合は、no_target_program 例外が表示されます。

タイムアウト・オプション

ターゲット・セッションのタイムアウト・オプションは、set_timeout_behavior をコールすると、ターゲット・セッションに登録されます。

- retry_on_timeout: 再試行します。タイムアウトの効果はありません。timeout に無限に大きい値を設定した場合と同じです。
- continue_on_timeout: 同じイベント・フラグを使用して、実行を継続します。
- nodebug_on_timeout: debug-mode をオフにして（つまり、debug_off をコール）、実行を継続します。debug_on をコールして初期化しなおさないかぎり、これ以降、このターゲット・セッションでイベントは生成されません。
- abort_on_timeout: abort_execution フラグを使用して実行を継続します。プログラムが即時に異常終了します。セッションはデバッグ・モードのままです。

```
retry_on_timeout CONSTANT BINARY_INTEGER:= 0;
continue_on_timeout CONSTANT BINARY_INTEGER:= 1;
nodebug_on_timeout CONSTANT BINARY_INTEGER:= 2;
abort_on_timeout      CONSTANT BINARY_INTEGER:= 3;
```

SET_TIMEOUT_BEHAVIOR プロシージャ

このプロシージャは、タイムアウト発生時のターゲット・セッションの処理方法をプローブに通知します。このコールは、ターゲット・セッションで行われます。

構文

```
DBMS_DEBUG.SET_TIMEOUT_BEHAVIOR (
   behavior IN PLS_INTEGER);
```

パラメータ

表 10-40 SET_TIMEOUT_BEHAVIOR プロシージャのパラメータ

パラメータ	説明
behavior: 次のいずれかです。	
■ retry_on_timeout	再試行します。タイムアウトの効果はありません。timeout に無限に大きい値を設定した場合と同じです。
■ continue_on_timeout	同じイベント・フラグを使用して、実行を継続します。
■ nodebug_on_timeout	debug-mode をオフにして（つまり、debug_off をコール）、実行を継続します。debug_on をコールして初期化しなおさないかぎり、これ以降、このターゲット・セッションでイベントは生成されません。
■ abort_on_timeout	abort_execution フラグを使用して実行を継続します。プログラムが即時に異常終了します。セッションはデバッグ・モードのままです。

例外

unimplemented: 要求された動作が認識されていません。

使用上の注意

デフォルトの動作（このプロシージャがコールされない場合）は、continue_on_timeout です。これは、デバッガ・クライアントが（次のイベントで）制御を再確立でき、ターゲット・セッションが無期限にハングアップすることはないためです。

GET_TIMEOUT_BEHAVIOR ファンクション

このプロシージャは、現行のタイムアウト動作を戻します。このコールは、ターゲット・セッションで行われます。

構文

```
DBMS_DEBUG.GET_TIMEOUT_BEHAVIOR (
RETURN BINARY_INTEGER;
```

情報フラグ

```
info_getOerInfo CONSTANT PLS_INTEGER:= 32;
```

理由

```
reason_oer_breakpoint   CONSTANT BINARY_INTEGER:= 26;
```

RUNTIME_INFO

Runtime_info は、実行プログラムに関するコンテキスト情報を提供します。

プローブ v2.4 では OER が追加されています。info_getOerInfo が設定されている場合は、設定を取得します。OER は正数です。1403 を 100 に、6510 を 1 に変換し、その他の値を無効にすると、SQLCODE に変換できます。

```
TYPE runtime_info IS RECORD
   (
       Line#            BINARY_INTEGER,   (duplicate of program.line#)
       Terminated       BINARY_INTEGER,   has the program terminated?
       Breakpoint       BINARY_INTEGER,   breakpoint number
       StackDepth       BINARY_INTEGER,   number of frames on the stack
       InterpreterDepth BINARY_INTEGER,   <reserved field>
       Reason           BINARY_INTEGER,   reason for suspension
       Program          program_info,     source location
Following fields were added in Probe v2.4 oer              PLS_INTEGER       OER
(exception), if any
   );
```

OER_TABLE

show_breakpoints によって使用されます。

TYPE oer_table IS TABLE OF BINARY_INTEGER INDEX BY BINARY_INTEGER;

SET_OER_BREAKPOINT

OER にブレーク・ポイントを設定します。コード・ブレーク・ポイント同様、ブレーク・ポイントはセッションに対して（または削除されるまで）持続されます。

パラメータ

表 10-41

パラメータ	説明
oer	OER（4 バイトの正数）

戻り値

success

使用上の注意

OER ブレーク・ポイントでサポートされている機能は、コード・ブレーク・ポイントと比較すると少なくなります。特に、次の点に注意してください。

- ブレーク・ポイント番号は戻されません。かわりに OER の番号が使用されます。したがって、指定した OER に複数のブレーク・ポイントは設定できません（操作できません）。
- OER ブレーク・ポイントは使用禁止にできません（ただし、クライアントは、これを削除して自由にシミュレートできます）。
- OER ブレーク・ポイントは、delete_oer_breakpoint を使用して削除されます。

SET_OER_BREAKPOINT ファンクション

このファンクションは、OER ブレーク・ポイントを設定します。

構文

```
DBMS_DEBUG.SET_OER_BREAKPOINT (
   oer  IN PLS_INTEGER)
RETURN PLS_INTEGER;
```

パラメータ

表 10-42 SET_OER_BREAKPOINT ファンクションのパラメータ

パラメータ	説明
oer	OER（4 バイトの正数）

戻り値

success

error_no_such_breakpt: 該当する OER ブレーク・ポイントが存在しません。

DELETE_OER_BREAKPOINT ファンクション

このファンクションは、OER ブレーク・ポイントを削除します。

構文

```
DBMS_DEBUG.DELETE_OER_BREAKPOINT (
   oer  IN PLS_INTEGER)
RETURN PLS_INTEGER;
```

SHOW_BREAKPOINTS プロシージャ

構文

```
DBMS_DEBUG.SHOW_BREAKPOINTS (
   code_breakpoints  OUT breakpoint_table,
   oer_breakpoints   OUT oer_table);
```

パラメータ

表 10-43 SHOW_BREAKPOINTS プロシージャのパラメータ

パラメータ	説明
code_breakpoints	ブレーク・ポイント番号で索引付けされた、ブレーク・ポイント・エントリの索引表。
oer_breakpoints	OER で索引付けされた、OER ブレーク・ポイントの索引表。

- code_breakpoints: ブレーク・ポイント番号で索引付けされた、ブレーク・ポイント・エントリの索引表。
- oer_breakpoints: OER で索引付けされた、OER ブレーク・ポイントの索引表。
- PROCEDURE show_breakpoints (code_breakpoints OUT breakpoint_table, oer_breakpoints OUT oer_table);

11

DBMS_DEFER

DBMS_DEFER は、レプリケート・トランザクションの遅延リモート・プロシージャ・コール（RPC）機能へのユーザー・インタフェースです。レプリケート・アプリケーションは、このインタフェース内のコールを使用して、後でトランザクションをリモート・ノードで実行するためにプロシージャ・コールをキューに登録します。

このプロシージャは通常、AFTER 行トリガーまたはアプリケーションで指定した更新プロシージャからコールされます。

この章では、次の項目について説明します。

DBMS_DEFER サブプログラムの要約

表 11-1 DBMS_DEFER パッケージのサブプログラム

サブプログラム	説明
「CALL プロシージャ」 11-2 ページ	リモート・プロシージャに対する遅延コールを作成します。
「COMMIT_WORK プロシージャ」 11-4 ページ	遅延リモート・プロシージャ・コール（RPC）が適切な構成かどうかをチェックし、その後トランザクション・コミットを実行します。
「datatype_ARG プロシージャ」 11-5 ページ	遅延リモート・プロシージャ・コール（RPC）に渡されるデータを提供します。
「TRANSACTION プロシージャ」 11-7 ページ	新規遅延トランザクションの開始を指示します。

CALL プロシージャ

このプロシージャは、リモート・プロシージャに対する遅延コールを作成します。

構文

```
DBMS_DEFER.CALL (
   schema_name       IN   VARCHAR2,
   package_name      IN   VARCHAR2,
   proc_name         IN   VARCHAR2,
   arg_count         IN   NATURAL,
   { nodes           IN   node_list_t
   | group_name      IN   VARCHAR2 :=''});
```

注意： このプロシージャはオーバーロードされています。nodes パラメータおよび group_name パラメータは、両方同時には指定できません。

パラメータ

表 11-2 CALL プロシージャのパラメータ

パラメータ	説明
schema_name	ストアド・プロシージャが置かれているスキーマ名。
package_name	ストアド・プロシージャを含んでいるパッケージの名前。ストアド・プロシージャはパッケージの一部であることが必要です。スタンドアロン・プロシージャに対する遅延コールは、サポートされていません。
proc_name	コールを延期するリモート・プロシージャの名前。
arg_count	プロシージャのパラメータ数。これらの各パラメータに対しては、DBMS_DEFER.*datatype*_ARG のコールが 1 つずつ必要です。 **注意 :** 一部のパラメータにデフォルトが設定されている場合でも、プロシージャに対してすべてのパラメータを指定する必要があります。
nodes	遅延コール伝播先の完全修飾データベース名の PL/SQL 索引付き表。この表は、位置 1 から始まり、NULL エントリが検出されるか、または NO_DATA_FOUND 例外が発生するまで索引付けされています。表内のデータは、大文字と小文字が区別されません。このパラメータはオプションです。
group_name	内部的に使用されるパラメータです。

例外

表 11-3 CALL プロシージャの例外

例外	説明
ORA-23304 (malformedcall)	引数の数が正しく構成されませんでした。
ORA-23319	パラメータ値が不適正です。
ORA-23352	(nodes または前の DBMS_DEFER.TRANSACTION コールで指定された) 宛先リストの値が重複しています。

COMMIT_WORK プロシージャ

このプロシージャは、遅延リモート・プロシージャ・コール（RPC）が適切な構成かどうかをチェックし、その後トランザクション・コミットを実行します。

構文

```
DBMS_DEFER.COMMIT_WORK (
   commit_work_comment IN VARCHAR2);
```

パラメータ

表 11-4 COMMIT_WORK プロシージャのパラメータ

パラメータ	説明
commit_work_comment	SQL の COMMIT COMMENT 文と同じです。

例外

表 11-5 COMMIT_WORK プロシージャの例外

例外	説明
ORA-23304 (malformedcall)	トランザクションが正しく発行されなかったか、または終了しませんでした。

*datatype*_ARG プロシージャ

このプロシージャは、遅延リモート・プロシージャ・コール（RPC）に渡されるデータを提供します。プロシージャに渡す必要があるデータのタイプによって、プロシージャの各引数ごとに次のプロシージャの 1 つをコールする必要があります。

プロシージャの各パラメータは、DBMS_DEFER.CALL の実行後に、*datatype*_ARG プロシージャを使用して指定してください。つまり、遅延リモート・プロシージャ・コール（RPC）にデフォルトのパラメータは使用できません。たとえば、次のプロシージャがあると想定します。

```
CREATE OR REPLACE PACKAGE my_pack AS
   PROCEDURE my_proc(a VARCHAR2, b VARCHAR2 DEFAULT 'SALES');
END;
/
```

DBMS_DEFER.CALL プロシージャを実行するときは、MY_PROC プロシージャの各パラメータ用に別のプロシージャ・コールを挿入する必要があります。

```
CREATE OR REPLACE PROCEDURE load_def_tx IS
   node DBMS_DEFER.NODE_LIST_T;
BEGIN
   node(1) := 'MYCOMPUTER.WORLD';
   node(2) := NULL;
   DBMS_DEFER.TRANSACTION(node);
   DBMS_DEFER.CALL('PR', 'MY_PACK', 'MY_PROC', 2);
   DBMS_DEFER.VARCHAR2_ARG('TEST');
   DBMS_DEFER.VARCHAR2_ARG('SALES'); -- required, cannot omit to use default
END;
/
```

注意：

- AnyData_ARG プロシージャがサポートするユーザー定義型は、オブジェクト型、コレクションおよび REF です。AnyData データ型の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
- このプロシージャは、一部の日時および期間データ型の略称を使用します。たとえば、TIMESTAMP WITH TIME ZONE データ・タイプには TSTZ が使用されます。略称の詳細は、1-6 ページの「日時および期間データ型の略称」を参照してください。

構文

```
DBMS_DEFER.AnyData_ARG      (arg  IN SYS.AnyData);
DBMS_DEFER.NUMBER_ARG       (arg  IN NUMBER);
DBMS_DEFER.DATE_ARG         (arg  IN DATE);
DBMS_DEFER.VARCHAR2_ARG     (arg  IN VARCHAR2);
DBMS_DEFER.CHAR_ARG         (arg  IN CHAR);
DBMS_DEFER.ROWID_ARG        (arg  IN ROWID);
DBMS_DEFER.RAW_ARG          (arg  IN RAW);
DBMS_DEFER.BLOB_ARG         (arg  IN BLOB);
DBMS_DEFER.CLOB_ARG         (arg  IN CLOB);
DBMS_DEFER.NCLOB_ARG        (arg  IN NCLOB);
DBMS_DEFER.NCHAR_ARG        (arg  IN NCHAR);
DBMS_DEFER.NVARCHAR2_ARG    (arg  IN NVARCHAR2);
DBMS_DEFER.ANY_CLOB_ARG     (arg  IN CLOB);
DBMS_DEFER.ANY_VARCHAR2_ARG (arg  IN VARCHAR2);
DBMS_DEFER.ANY_CHAR_ARG     (arg  IN CHAR);
DBMS_DEFER.IDS_ARG          (arg  IN DSINTERVAL_UNCONSTRAINED);
DBMS_DEFER.IYM_ARG          (arg  IN YMINTERVAL_UNCONSTRAINED);
DBMS_DEFER.TIMESTAMP_ARG    (arg  IN TIMESTAMP_UNCONSTRAINED);
DBMS_DEFER.TSLTZ_ARG        (arg  IN TIMESTAMP_LTZ_UNCONSTRAINED);
DBMS_DEFER.TSTZ_ARG         (arg  IN TIMESTAMP_TZ_UNCONSTRAINED);
```

パラメータ

表 11-6 *datatype*_ARG プロシージャのパラメータ

パラメータ	説明
arg	事前に遅延コールを実行したリモート・プロシージャに渡すパラメータの値。

例外

表 11-7 *datatype*_ARG プロシージャの例外

例外	説明
ORA-23323	引数の値が長すぎます。

TRANSACTION プロシージャ

このプロシージャは、新規遅延トランザクションの開始を指示します。このコールを省略すると、DBMS_DEFER.CALL への最初のコールが新規トランザクションの開始とみなされます。

構文

```
DBMS_DEFER.TRANSACTION (
   nodes  IN   node_list_t);
```

> **注意：** このプロシージャはオーバーロードされています。入力パラメータが指定されていない場合も指定されている場合と同様に動作しますが、指定されていない場合は、nodes パラメータのノードを使用するかわりに、DEFDEFAULTDEST ビューの nodes が使用されます。

パラメータ

表 11-8 TRANSACTION プロシージャのパラメータ

パラメータ	説明
nodes	トランザクションの遅延コール伝播先の完全修飾データベース名の PL/SQL 索引付き表。この表は、位置 1 から始まり、NULL エントリが検出されるか、または NO_DATA_FOUND 例外が発生するまで索引付けされています。表内のデータは、大文字と小文字が区別されません。

例外

表 11-9 TRANSACTION プロシージャの例外

例外	説明
ORA-23304 (malformedcall)	前のトランザクションが正しく発行されなかったか、または終了しませんでした。
ORA-23319	パラメータ値が不適正です。
ORA-23352	ノード・リストの値が重複している場合に DBMS_DEFER.CALL で生成されます。

12

DBMS_DEFER_QUERY

DBMS_DEFER_QUERY によって、ビューでは参照できない遅延トランザクションのキュー・データを問い合せることができます。

この章では、次の項目について説明します。

DBMS_DEFER_QUERY サブプログラムの要約

表 12-1 DBMS_DEFER_QUERY パッケージのサブプログラム

サブプログラム	説明
「GET_ARG_FORM ファンクション」12-2 ページ	遅延コールの引数の形式を判別します。
「GET_ARG_TYPE ファンクション」12-4 ページ	遅延コールの引数のタイプを判別します。
「GET_CALL_ARGS プロシージャ」12-6 ページ	指定されたコールに対する様々な引数をテキストで戻します。
「GET_datatype_ARG ファンクション」12-7 ページ	遅延コールの引数の値を判別します。
「GET_OBJECT_NULL_VECTOR_ARG ファンクション」12-9 ページ	列オブジェクトのタイプ情報を戻します。

GET_ARG_FORM ファンクション

このファンクションは、遅延コール・パラメータのキャラクタ・セット・フォームを戻します。

関連項目： レプリケーション管理ツールにおける遅延トランザクションおよびエラー・トランザクションの表示方法は、レプリケーション管理ツールのオンライン・ヘルプを参照してください。

構文

```
DBMS_DEFER_QUERY.GET_ARG_FORM (
   callno                IN   NUMBER,
   arg_no                IN   NUMBER,
   deferred_tran_id      IN   VARCHAR2)
  RETURN NUMBER;
```

パラメータ

表 12-2 GET_ARG_FORM ファンクションのパラメータ

パラメータ	説明
callno	DEFCALL ビューからのコール識別子。
arg_no	コール引数リスト上の目的のパラメータの位置。パラメータの位置は、コール内のパラメータを 1 から順に数えます。
deferred_tran_id	遅延トランザクション ID。

例外

表 12-3 GET_ARG_FORM ファンクションの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。

戻り値

表 12-4 GET_ARG_FORM ファンクションの戻り値

定数の戻り値	戻り値	設定可能なデータ・タイプ
DBMS_DEFER_QUERY.ARG_FORM_NONE	0	DATE
		NUMBER
		ROWID
		RAW
		BLOB
		ユーザー定義型
DBMS_DEFER_QUERY.ARG_FORM_IMPLICIT	1	CHAR
		VARCHAR2
		CLOB
DBMS_DEFER_QUERY.ARG_FORM_NCHAR	2	NCHAR
		NVARCHAR2
		NCLOB

GET_ARG_TYPE ファンクション

このファンクションは、遅延コールの引数のタイプを判別します。遅延リモート・プロシージャ・コール（RPC）のパラメータのタイプが戻されます。

> **関連項目：** レプリケーション管理ツールにおける遅延トランザクションおよびエラー・トランザクションの表示方法は、レプリケーション管理ツールのオンライン・ヘルプを参照してください。

構文

```
DBMS_DEFER_QUERY.GET_ARG_TYPE (
   callno            IN   NUMBER,
   arg_no            IN   NUMBER,
   deferred_tran_id  IN   VARCHAR2)
  RETURN NUMBER;
```

パラメータ

表 12-5 GET_ARG_TYPE ファンクションのパラメータ

パラメータ	説明
callno	遅延リモート・プロシージャ・コール（RPC）の DEFCALL ビューでの ID 番号。
arg_no	判別するタイプのコールに対する引数の数値的な位置。プロシージャに対する最初の引数の位置は 1 です。
deferred_tran_id	遅延トランザクションの ID。

例外

表 12-6 GET_ARG_TYPE ファンクションの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。

戻り値

表 12-7 GET_ARG_TYPE ファンクションの戻り値

定数の戻り値	戻り値	対応するデータ・タイプ
DBMS_DEFER_QUERY.ARG_TYPE_VARCHAR2	1	VARCHAR2
DBMS_DEFER_QUERY.ARG_TYPE_NUM	2	NUMBER
DBMS_DEFER_QUERY.ARG_TYPE_ROWID	11	ROWID
DBMS_DEFER_QUERY.ARG_TYPE_DATE	12	DATE
DBMS_DEFER_QUERY.ARG_TYPE_RAW	23	RAW
DBMS_DEFER_QUERY.ARG_TYPE_CHAR	96	CHAR
DBMS_DEFER_QUERY.ARG_TYPE_AnyData	109	AnyData
DBMS_DEFER_QUERY.ARG_TYPE_CLOB	112	CLOB
DBMS_DEFER_QUERY.ARG_TYPE_BLOB	113	BLOB
DBMS_DEFER_QUERY.ARG_TYPE_BFIL	114	BFILE
DBMS_DEFER_QUERY.ARG_TYPE_OBJECT_NULL_VECTOR	121	OBJECT_NULL_VECTOR
DBMS_DEFER_QUERY.ARG_TYPE_TIMESTAMP	180	TIMESTAMP
DBMS_DEFER_QUERY.ARG_TYPE_TSTZ	181	TSTZ
DBMS_DEFER_QUERY.ARG_TYPE_IYM	182	IYM
DBMS_DEFER_QUERY.ARG_TYPE_IDS	183	IDS
DBMS_DEFER_QUERY.ARG_TYPE_TSLTZ	231	TSLTZ

注意：

- AnyData データ型がサポートするユーザー定義型は、オブジェクト型、コレクションおよび REF です。AnyData データ型の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
- このファンクションは、一部の日時および期間データ型の略称を使用します。たとえば、TIMESTAMP WITH TIME ZONE データ型には TSTZ が使用されます。略称の詳細は、1-6 ページの「日時および期間データ型の略称」を参照してください。

GET_CALL_ARGS プロシージャ

このプロシージャは、指定されたコールに対する様々な引数をテキストで戻します。テキストは最初の 2000 バイトまでに制限されています。

関連項目：

- 12-7 ページ「GET_datatype_ARG ファンクション」
- AnyData データ型の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

構文

```
DBMS_DEFER_QUERY.GET_CALL_ARGS (
   callno    IN  NUMBER,
   startarg  IN  NUMBER := 1,
   argcnt    IN  NUMBER,
   argsize   IN  NUMBER,
   tran_id   IN  VARCHAR2,
   date_fmt  IN  VARCHAR2,
   types     OUT TYPE_ARY,
   forms     OUT TYPE_ARY,
   vals      OUT VAL_ARY);
```

パラメータ

表 12-8 GET_CALL_ARGS プロシージャのパラメータ

パラメータ	説明
callno	遅延リモート・プロシージャ・コール（RPC）の DEFCALL ビューでの ID 番号。
startarg	記述する最初の引数の数値的な位置。
argcnt	コールにある引数の数。
argsize	戻される引数の最大サイズ。
tran_id	遅延トランザクションの ID。
date_fmt	日付を戻す書式。
types	引数のタイプを含んでいる配列。
forms	キャラクタ・セット・フォームの引数を含んでいる配列。
vals	テキスト形式で引数の値を含んでいる配列。

例外

表 12-9 GET_CALL_ARGS プロシージャの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。

GET_*datatype*_ARG ファンクション

このファンクションは、遅延コールの引数の値を判別します。

AnyData タイプがサポートするユーザー定義型は、オブジェクト・タイプ、コレクションおよび REF です。このファンクションでサポートされているすべてのタイプが DBMS_DEFER パッケージの AnyData_ARG プロシージャを使用してエンキューできるわけではありません。

引数タイプに戻されるテキストには、タイプの所有者、タイプの名前、タイプのバージョン、長さ、精度、スケール、キャラクタ・セットの識別子、キャラクタ・セット・フォームおよびコレクションの要素の数またはオブジェクト・タイプの属性の数が含まれます。これらの値は、コロン（:）で区切ります。

関連項目：

- 11-5 ページ「datatype_ARG プロシージャ」
- レプリケーション管理ツールにおける遅延トランザクションおよびエラー・トランザクションの表示方法は、レプリケーション管理ツールのオンライン・ヘルプを参照してください。
- AnyData データ型の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
- このファンクションは、一部の日時および期間データ型の略称を使用します。たとえば、TIMESTAMP WITH TIME ZONE データ・タイプには TSTZ が使用されます。略称の詳細は、1-6 ページの「日時および期間データ型の略称」を参照してください。

構文

取り出す引数値のタイプに応じて、該当するファンクションの構文は次のように異なります。各ファンクションは、それぞれ指定された引数の値を戻します。

```
DBMS_DEFER_QUERY.GET_datatype_ARG (
   callno             IN   NUMBER,
   arg_no             IN   NUMBER,
   deferred_tran_id   IN   VARCHAR2 DEFAULT NULL)
  RETURN datatype;
```

datatype には次のいずれかを指定します。

```
{ AnyData
| NUMBER
| VARCHAR2
| CHAR
| DATE
| RAW
| ROWID
| BLOB
| CLOB
| NCLOB
| NCHAR
| NVARCHAR2
| IDS
| IYM
| TIMESTAMP
| TSLTZ
| TSTZ }
```

パラメータ

表 12-10 GET_*datatype*_ARG ファンクションのパラメータ

パラメータ	説明
callno	遅延リモート・プロシージャ・コール（RPC）の DEFCALL ビューでの ID 番号。
arg_no	判別する値のコールに対する引数の数値的な位置。プロシージャに対する最初の引数の位置は 1 です。
deferred_tran_id	遅延トランザクションの ID。GET_ARG_TYPE ファンクションに渡された最後のトランザクション ID に設定されます。デフォルトは NULL です。

例外

表 12-11 GET_*datatype*_ARG ファンクションの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。
ORA-26564	入力パラメータが、指定されたタイプまたは AnyData タイプにサポートされるタイプに該当しません。

GET_OBJECT_NULL_VECTOR_ARG ファンクション

このファンクションは、タイプの所有者、名前およびハッシュコードなど、列オブジェクトのタイプ情報を戻します。

構文

```
DBMS_DEFER_QUERY.GET_OBJECT-NULL_VECTOR_ARG (
   callno                IN   NUMBER,
   arg_no                IN   NUMBER,
   deferred_tran_id      IN   VARCHAR2)
  RETURN SYSTEM.REPCAT$_OBJECT_NULL_VECTOR;
```

パラメータ

表 12-12 GET_OBJECT_NULL_VECTOR_ARG ファンクションのパラメータ

パラメータ	説明
callno	DEFCALL ビューからのコール識別子。
arg_no	コール引数リスト上の目的のパラメータの位置。パラメータの位置は、コール内のパラメータを 1 から順に数えます。
deferred_tran_id	遅延トランザクション ID。

例外

表 12-13 GET_OBJECT_NULL_VECTOR_ARG ファンクションの例外

例外	説明
NO_DATA_FOUND	入力パラメータが遅延コールのパラメータに対応していません。
ORA-26564	パラメータは object_null_vector タイプではありません。

戻り値

表 12-14 GET_OBJECT_NULL_VECTOR_ARG ファンクションの戻り値

戻り値	タイプ定義

SYSTEM.REPCAT$_OBJECT_NULL_VECTOR タイプ	CREATE TYPE
	SYSTEM.REPCAT$_OBJECT_NULL_VECTOR
	AS OBJECT (
	  type_owner VARCHAR2(30),
	  type_name VARCHAR2(30),
	  type_hashcode RAW(17),
	  null_vector RAW(2000));

13

DBMS_DEFER_SYS

DBMS_DEFER_SYS プロシージャは、デフォルトのレプリケーション・ノード・リストを管理します。このパッケージは、レプリケート・トランザクションの遅延リモート・プロシージャ・コール機能へのシステム管理者用インタフェースです。管理者およびレプリケーション・デーモンは、この機能を使用して、リモート・ノードのキューに入れられたトランザクションを実行できます。また、管理者はリモート・コールの宛先のノードを制御できます。

この章では、次の項目について説明します。

DBMS_DEFER_SYS サブプログラムの要約

表 13-1 DBMS_DEFER_SYS パッケージのサブプログラム

サブプログラム	説明
「ADD_DEFAULT_DEST プロシージャ」13-3 ページ	DEFDEFAULTDEST ビューに接続先データベースを追加します。
「CLEAR_PROP_STATISTICS プロシージャ」13-4 ページ	DEFSCHEDULE データ・ディクショナリ・ビューの伝播統計を消去します。
「DELETE_DEFAULT_DEST プロシージャ」13-5 ページ	DEFDEFAULTDEST ビューから接続先データベースを削除します。
「DELETE_DEF_DESTINATION プロシージャ」13-5 ページ	DEFSCHEDULE ビューから接続先データベースを削除します。
「DELETE_ERROR プロシージャ」13-6 ページ	DEFERROR ビューからトランザクションを削除します。
「DELETE_TRAN プロシージャ」13-6 ページ	DEFTRANDEST ビューからトランザクションを削除します。
「DISABLED ファンクション」13-7 ページ	現行のサイトから指定サイトへの遅延トランザクション・キューの伝播が使用可能かどうかを判断します。
「EXCLUDE_PUSH ファンクション」13-8 ページ	遅延トランザクションの PUSH を防止する排他ロックを取得します。
「EXECUTE_ERROR プロシージャ」13-9 ページ	トランザクションの元の受信者のセキュリティ・コンテキスト内で正常に完了しなかった遅延トランザクションを再実行します。
「EXECUTE_ERROR_AS_USER プロシージャ」13-10 ページ	このプロシージャを実行しているユーザーのセキュリティ・コンテキスト内で正常に完了しなかった遅延トランザクションを再実行します。
「PURGE ファンクション」13-11 ページ	現行のマスター・サイトまたはマテリアライズド・ビュー・サイトの遅延トランザクション・キューから、送信済みトランザクションをパージします。
「PUSH ファンクション」13-13 ページ	現行のマスター・サイトまたはマテリアライズド・ビュー・サイトの遅延リモート・プロシージャ・コール（RPC）のキューを、リモート・サイトに強制的に送信します。
「REGISTER_PROPAGATOR プロシージャ」13-16 ページ	指定したユーザーをローカル・データベースのプロパゲータとして登録します。

表 13-1 DBMS_DEFER_SYS パッケージのサブプログラム（続き）

サブプログラム	説明
「SCHEDULE_PURGE プロシージャ」13-17 ページ	現行のマスター・サイトまたはマテリアライズド・ビュー・サイトの遅延トランザクション・キューから、送信済みトランザクションをパージするジョブをスケジュールします。
「SCHEDULE_PUSH プロシージャ」13-19 ページ	遅延トランザクション・キューをリモート・サイトに送信するジョブをスケジュールします。
「SET_DISABLED プロシージャ」13-21 ページ	現行のサイトから指定宛先サイトへの遅延トランザクション・キューの伝播を使用禁止または使用可能にします。
「UNREGISTER_PROPAGATOR プロシージャ」13-23 ページ	ローカル・データベースからプロパゲータとしてのユーザーの登録を解除します。
「UNSCHEDULE_PURGE プロシージャ」13-24 ページ	マスター・サイトまたはマテリアライズド・ビュー・サイトの遅延トランザクション・キューからの送信済みトランザクションの自動パージを停止します。
「UNSCHEDULE_PUSH プロシージャ」13-24 ページ	マスター・サイトまたはマテリアライズド・ビュー・サイトからリモート・サイトへの遅延トランザクション・キューの自動送信を停止します。

ADD_DEFAULT_DEST プロシージャ

このプロシージャは、DEFDEFAULTDEST データ・ディクショナリ・ビューに接続先データベースを追加します。

構文

```
DBMS_DEFER_SYS.ADD_DEFAULT_DEST (
   dblink   IN   VARCHAR2);
```

パラメータ

表 13-2 ADD_DEFAULT_DEST プロシージャのパラメータ

パラメータ	説明
dblink	DEFDEFAULTDEST ビューに追加するノードの完全修飾データベース名。

例外

表 13-3 ADD_DEFAULT_DEST プロシージャの例外

例外	説明
ORA-23352	指定した dblink は、デフォルト・リストにすでに存在します。

CLEAR_PROP_STATISTICS プロシージャ

このプロシージャは、DEFSCHEDULE データ・ディクショナリ・ビューの伝播統計を消去します。プロシージャが正常に実行された場合、このビューのすべての統計は 0（ゼロ）に戻され、統計の収集が最初から開始されます。

特に、このプロシージャは、DEFSCHEDULE データ・ディクショナリ・ビューにおける次の列の統計を消去します。

- TOTAL_TXN_COUNT
- AVG_THROUGHPUT
- AVG_LATENCY
- TOTAL_BYTES_SENT
- TOTAL_BYTES_RECEIVED
- TOTAL_ROUND_TRIPS
- TOTAL_ADMIN_COUNT
- TOTAL_ERROR_COUNT
- TOTAL_SLEEP_TIME

構文

```
DBMS_DEFER_SYS.CLEAR_PROP_STATISTICS (
   dblink    IN   VARCHAR2);
```

パラメータ

表 13-4 CLEAR_PROP_STATISTICS プロシージャのパラメータ

パラメータ	説明
dblink	統計を消去するノードの完全修飾データベース名。消去されるのは、現行のノードから dblink を指定するノードへ遅延トランザクションを伝播する統計です。

DELETE_DEFAULT_DEST プロシージャ

このプロシージャは、DEFDEFAULTDEST ビューから接続先データベースを削除します。

構文

```
DBMS_DEFER_SYS.DELETE_DEFAULT_DEST (
   dblink    IN    VARCHAR2);
```

パラメータ

表 13-5 DELETE_DEFAULT_DEST プロシージャのパラメータ

パラメータ	説明
dblink	DEFDEFAULTDEST ビューから削除するノードの完全修飾データベース名。ビューにこの DB リンクが見つからなかった場合は、何の処理も行われません。

DELETE_DEF_DESTINATION プロシージャ

このプロシージャは、DEFSCHEDULE ビューから接続先データベースを削除します。

構文

```
DBMS_DEFER_SYS.DELETE_DEF_DESTINATION (
   destination    IN    VARCHAR2,
   force          IN    BOOLEAN := false);
```

パラメータ

表 13-6 DELETE_DEF_DESTINATION プロシージャのパラメータ

パラメータ	説明
destination	DEFSCHEDULE ビューから削除する宛先の完全修飾データベース名。ビューにこの宛先が見つからなかった場合は、何の処理も行われません。
force	TRUE に設定すると、安全チェックはすべて無視され、宛先が削除されます。

DELETE_ERROR プロシージャ

このプロシージャは、DEFERROR ビューからトランザクションを削除します。

構文

```
DBMS_DEFER_SYS.DELETE_ERROR(
   deferred_tran_id     IN   VARCHAR2,
   destination          IN   VARCHAR2);
```

パラメータ

表 13-7 DELETE_ERROR プロシージャのパラメータ

パラメータ	説明
deferred_tran_id	DEFERROR ビューから削除する遅延トランザクションの DEFERROR ビューでの ID 番号。このパラメータが NULL の場合は、他のパラメータの要件と一致するすべてのトランザクションが削除されます。
destination	トランザクションが最初にキューに入れられたデータベースの DEFERROR ビューでの完全修飾データベース名。このパラメータが NULL の場合は、他のパラメータの要件と一致するすべてのトランザクションが DEFERROR ビューから削除されます。

DELETE_TRAN プロシージャ

このプロシージャは、DEFTRANDEST ビューからトランザクションを削除します。トランザクションに対する DEFTRANDEST または DEFERROR のエントリが他にない場合、そのトランザクションは、DEFTRAN および DEFCALL のビューからも削除されます。

構文

```
DBMS_DEFER_SYS.DELETE_TRAN (
   deferred_tran_id     IN   VARCHAR2,
   destination          IN   VARCHAR2);
```

パラメータ

表 13-8 DELETE_TRAN プロシージャのパラメータ

パラメータ	説明
deferred_tran_id	削除する遅延トランザクションの DEFTRAN ビューでの ID 番号。このパラメータが NULL の場合は、他のパラメータの要件と一致するすべてのトランザクションが削除されます。
destination	トランザクションが最初にキューに入れられたデータベースの DEFTRANDEST ビューでの完全修飾データベース名。このパラメータが NULL の場合は、他のパラメータの要件と一致するすべてのトランザクションが削除されます。

DISABLED ファンクション

このファンクションは、現行のサイトから指定サイトへの遅延トランザクション・キューの伝播が使用可能かどうかを判断します。指定した宛先に対する遅延リモート・プロシージャ・コール（RPC）のキューが使用禁止の場合、DISABLED ファンクションは TRUE を戻します。

構文

```
DBMS_DEFER_SYS.DISABLED (
   destination  IN   VARCHAR2)
  RETURN BOOLEAN;
```

パラメータ

表 13-9 DISABLED ファンクションのパラメータ

パラメータ	説明
destination	伝播状態をチェックするノードの完全修飾データベース名。

戻り値

表 13-10 DISABLED ファンクションの戻り値

戻り値	説明
TRUE	現行のサイトから指定したサイトへの伝播は使用禁止です。
FALSE	現行のサイトから指定したサイトへの伝播は使用可能です。

例外

表 13-11 DISABLED ファンクションの例外

例外	説明
NO_DATA_FOUND	指定した destination が DEFSCHEDULE ビューにありません。

EXCLUDE_PUSH ファンクション

このファンクションは、遅延トランザクションの PUSH（シリアルまたはパラレル）を防止する排他ロックを取得します。ロックの取得時にコミットが実行されます。ロックは RELEASE_ON_COMMIT => TRUE で取得されるため、遅延トランザクション・キューの送信は、次のコミットの後に再開できます。

構文

```
DBMS_DEFER_SYS.EXCLUDE_PUSH (
   timeout    IN    INTEGER)
  RETURN INTEGER;
```

パラメータ

表 13-12 EXCLUDE_PUSH ファンクションのパラメータ

パラメータ	説明
timeout	タイムアウト（秒数）。この時間内にロックを取得できない場合（エラーまたは PUSH が現在進行中であるため）は、値 1 が戻されます。タイムアウト値が DBMS_LOCK.MAXWAIT の場合は、無期限に待機します。

戻り値

表 13-13 EXCLUDE_PUSH ファンクションの戻り値

戻り値	説明
0	正常終了。ロックが取得されました。
1	タイムアウト。ロックは取得されていません。
2	デッドロック。ロックは取得されていません。
4	ロックはすでに取得されています。

EXECUTE_ERROR プロシージャ

このプロシージャは、トランザクションの元の受信者のセキュリティ・コンテキスト内で正常に完了しなかった遅延トランザクションを再実行します。

構文

```
DBMS_DEFER_SYS.EXECUTE_ERROR (
   deferred_tran_id IN   VARCHAR2,
   destination      IN   VARCHAR2);
```

パラメータ

表 13-14 EXECUTE_ERROR プロシージャのパラメータ

パラメータ	説明
deferred_tran_id	再実行する遅延トランザクションの DEFERROR ビューでの ID 番号。NULL の場合は、destination のキューにあるすべてのトランザクションが再実行されます。
destination	トランザクションが最初にキューに入れられたデータベースの DEFERROR ビューでの完全修飾データベース名。NULL 以外の値を設定してください。指定したデータベース名が完全に修飾されていないか、無効な場合、エラーは発生しません。

例外

表 13-15 EXECUTE_ERROR プロシージャの例外

例外	説明
ORA-24275 エラー	NULL および NULL 以外のパラメータを不正に組み合せて使用しました。
badparam	パラメータが未指定または無効です（たとえば、destination が NULL の場合）。
missinguser	無効なユーザーです。

EXECUTE_ERROR_AS_USER プロシージャ

このプロシージャは、正常に完了しなかった遅延トランザクションを再実行します。各トランザクションは接続ユーザーのセキュリティ・コンテキスト内で実行されます。

構文

```
DBMS_DEFER_SYS.EXECUTE_ERROR_AS_USER (
   deferred_tran_id IN   VARCHAR2,
   destination      IN   VARCHAR2);
```

パラメータ

表 13-16 EXECUTE_ERROR_AS_USER プロシージャのパラメータ

パラメータ	説明
deferred_tran_id	再実行する遅延トランザクションの DEFERROR ビューでの ID 番号。NULL の場合は、destination のキューにあるすべてのトランザクションが再実行されます。
destination	トランザクションが最初にキューに入れられたデータベースの DEFERROR ビューでの完全修飾データベース名。NULL 以外の値を設定してください。

例外

表 13-17 EXECUTE_ERROR_AS_USER プロシージャの例外

例外	説明
ORA-24275 エラー	NULL および NULL 以外のパラメータを不正に組み合せて使用しました。
badparam	パラメータが未指定または無効です（たとえば、destination が NULL の場合）。
missinguser	無効なユーザーです。

PURGE ファンクション

このファンクションは、現行のマスター・サイトまたはマテリアライズド・ビュー・サイトの遅延トランザクション・キューから、送信済みトランザクションをパージします。

構文

```
DBMS_DEFER_SYS.PURGE (
   purge_method          IN  BINARY_INTEGER := purge_method_quick,
   rollback_segment      IN  VARCHAR2       := NULL,
   startup_seconds       IN  BINARY_INTEGER := 0,
   execution_seconds     IN  BINARY_INTEGER := seconds_infinity,
   delay_seconds         IN  BINARY_INTEGER := 0,
   transaction_count     IN  BINARY_INTEGER := transactions_infinity,
   write_trace           IN  BOOLEAN        := NULL);
  RETURN BINARY_INTEGER;
```

パラメータ

表 13-18 PURGE ファンクションのパラメータ

パラメータ	説明
purge_method	遅延トランザクション・キューのパージ方法を制御します。purge_method_quick はコストが低く、一方 purge_method_precise は精度が高くなります。 purge_method_quick を使用するには、このパラメータに対して次の値を指定します。 dbms_defer_sys.purge_method_quick purge_method_precise を使用するには、このパラメータに対して次の値を指定します。 dbms_defer_sys.purge_method_precise purge_method_quick を使用すると、正常に送信された遅延トランザクションおよび遅延プロシージャ・コールが、パージ前に予測した時間を超えて、DEFTRAN データ・ディクショナリ・ビューおよび DEFCALL データ・ディクショナリ・ビューに残る場合があります。詳細は、13-13 ページの「使用上の注意」を参照してください。
rollback_segment	パージに使用するロールバック・セグメント名。デフォルトは NULL です。
startup_seconds	同じ遅延トランザクション・キューの直前のパージを待つ最大秒数。
execution_seconds	> 0 の場合は、指定した秒数後にパージを即時停止します。
delay_seconds	遅延トランザクション・キューにパージするトランザクションが delay_seconds 秒間ない場合は、パージを停止します。

表 13-18 PURGE ファンクションのパラメータ（続き）

パラメータ	説明
transaction_count	> 0 の場合は、transaction_count が示す件数のトランザクションをパージした後、シャットダウンします。
write_trace	TRUE に設定すると、PURGE ファンクションによって戻された結果値がサーバーのトレース・ファイルに記録されます。FALSE に設定した場合、結果値は記録されません。

戻り値

表 13-19 PURGE ファンクションの戻り値

戻り値	説明
result_ok	OK。delay_seconds 秒経過後に終了しました。
result_startup_seconds	起動中にロック・タイムアウトで終了しました。
result_execution_seconds	execution_seconds 秒経過したため終了しました。
result_transaction_count	transaction_count 件を超えたため終了しました。
result_errors	エラー発生後に終了しました。
result_split_del_order_limit	排他モードでエンキューの取得に失敗した後、終了しました。このリターン・コードを受け取った場合は、パージを再試行してください。問題が解決しない場合は、オラクル社カスタマ・サポート・センターに連絡してください。
result_purge_disabled	停止せずに新規マスター・サイトを追加した場合、キューの同期をとるために内部的にパージが無効となります。

例外

表 13-20 PURGE ファンクションの例外

例外	説明
argoutofrange	パラメータ値が有効範囲外です。
executiondisabled	パージの実行が使用禁止です。
defererror	内部エラーです。

使用上の注意

DBMS_DEFER_SYS.PURGE ファンクションの purge_method パラメータに purge_method_quick を使用すると、遅延トランザクションおよび遅延プロシージャ・コールは、正常に送信された後、DEFCALL データ・ディクショナリ・ビューおよび DEFTRAN データ・ディクショナリ・ビューに残る場合があります。この動作は、複数のデータベース・リンクを持ち、送信が 1 つのデータベース・リンクに対してのみ実行されるレプリケーション環境で発生します。

遅延トランザクションおよび遅延プロシージャ・コールをパージするには、次のいずれかの処理を実行してください。

- purge_method パラメータに、purge_method_quick ではなく、purge_method_precise パラメータを使用します。purge_method_precise を使用すると、コストがかかりますが、遅延トランザクションおよび遅延プロシージャ・コールは、正常に送信された後、確実にパージされます。
- purge_method パラメータに purge_method_quick を使用すると、遅延トランザクションがすべてのデータベース・リンクに送信されます。遅延トランザクションおよび遅延プロシージャ・コールは、最後のデータベース・リンクへの送信が正常に完了するとパージされます。

PUSH ファンクション

このファンクションは、現行のマスター・サイトまたはマテリアライズド・ビュー・サイトの遅延リモート・プロシージャ・コール（RPC）のキューを、シリアルまたはパラレル伝播を使用してリモート・サイトへ強制的に送信（伝播）します。

構文

```
DBMS_DEFER_SYS.PUSH (
   destination          IN  VARCHAR2,
   parallelism          IN  BINARY_INTEGER := 0,
   heap_size            IN  BINARY_INTEGER := 0,
   stop_on_error        IN  BOOLEAN        := false,
   write_trace          IN  BOOLEAN        := false,
   startup_seconds      IN  BINARY_INTEGER := 0,
   execution_seconds    IN  BINARY_INTEGER := seconds_infinity,
   delay_seconds        IN  BINARY_INTEGER := 0,
   transaction_count    IN  BINARY_INTEGER := transactions_infinity,
   delivery_order_limit IN  NUMBER         := delivery_order_infinity)
  RETURN BINARY_INTEGER;
```

パラメータ

表 13-21 PUSH ファンクションのパラメータ

パラメータ	説明
destination	変更内容の転送先であるマスター・サイトまたはマテリアライズド・ビュー・サイトの完全修飾データベース名。
parallelism	0（ゼロ）はシリアル伝播を指定します。 *n* > 1 は *n* のパラレル処理を使用するパラレル伝播を指定します。 1 は単一のパラレル処理を使用するパラレル伝播を指定します。
heap_size	パラレル伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。 **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。
stop_on_error	デフォルトは FALSE で、競合などのエラーが発生しても処理は継続されます。TRUE を設定すると、宛先サイトでトランザクションにエラーが発生したことが最初に通知されたときに伝播を停止します。
write_trace	TRUE に設定すると、ファンクションによって戻された結果値がサーバーのトレース・ファイルに記録されます。FALSE に設定した場合、結果値は記録されません。
startup_seconds	同じ宛先への直前の送信を待つ最大秒数。
execution_seconds	> 0 の場合は、指定した秒数後に送信を停止します。transaction_count および execution_seconds が 0（デフォルト）の場合は、キュー内のトランザクションがなくなるまで実行されます。 execution_seconds パラメータは、操作を開始できる時間を制御するのみです。トランザクションがリモート・サイトで必要とする時間は含まれていません。したがって、execution_seconds パラメータは、リモート・サイトへのトランザクションの伝播を停止するための正確な制御に使用することを目的としていません。正確な制御が必要な場合は、transaction_count パラメータまたは delivery_order パラメータを使用してください。
delay_seconds	キューが空の場合でも、指定した秒数が経過するまで戻しません。PUSH がタイトなループからコールされる場合は、実行オーバーヘッドの削減に役立ちます。

表 13-21 PUSH ファンクションのパラメータ（続き）

パラメータ	説明
transaction_count	> 0 の場合は、停止までに送信されるトランザクションの最大数。transaction_count および execution_seconds が 0（デフォルト）の場合は、送信する必要があるトランザクションがキュー内になくなるまで実行されます。
delivery_order_limit	delivery_order >= delivery_order_limit の場合は、トランザクションを送信する前に実行を正確に停止します。

戻り値

表 13-22 PUSH ファンクションの戻り値

戻り値	説明
result_ok	OK。delay_seconds 秒経過後に終了しました。
result_startup_seconds	起動中にロック・タイムアウトで終了しました。
result_execution_seconds	execution_seconds 秒経過したため終了しました。
result_transaction_count	transaction_count 件を超えたため終了しました。
result_delivery_order_limit	delivery_order_limit 数を超えたため終了しました。
result_errors	エラー発生後に終了しました。
result_push_disabled	送信は内部的に無効にされました。戻り値は通常、マスター・グループを停止せずに新規マスター・サイトをマスター・グループへ追加する場合、伝播の同期をとるために、宛先への伝播が Oracle により内部的に無効にされたことを意味します。伝播は後で自動的に有効化されます。
result_split_del_order_limit	排他モードでエンキューの取得に失敗した後、終了しました。このリターン・コードを受け取った場合は、送信を再試行してください。問題が解決しない場合は、オラクル社カスタマ・サポート・センターに連絡してください。

例外

表 13-23 PUSH ファンクションの例外

例外	説明
incompleteparallelpush	シリアル伝播を実行するときは、パラレル伝播が完全にシャットダウンされている必要があります。
executiondisabled	宛先での遅延リモート・プロシージャ・コール（RPC）の実行が使用禁止です。
crt_err_err	DEFERROR のエントリ作成時にエラーが発生しました。
deferred_rpc_quiesce	レプリケーション・グループのレプリケーション・アクティビティが中断されています。
commfailure	遅延リモート・プロシージャ・コール（RPC）の間における通信障害。
missingpropagator	プロパゲータが存在しません。

REGISTER_PROPAGATOR プロシージャ

このプロシージャは、指定したユーザーをローカル・データベースのプロパゲータとして登録します。また、特定のユーザーに次の権限を付与します（権限を付与されたユーザーはラッパーを作成できます）。

- CREATE SESSION
- CREATE PROCEDURE
- CREATE DATABASE LINK
- EXECUTE ANY PROCEDURE

構文

```
DBMS_DEFER_SYS.REGISTER_PROPAGATOR (
   username  IN  VARCHAR2);
```

パラメータ

表 13-24 REGISTER_PROPAGATOR プロシージャのパラメータ

パラメータ	説明
username	ユーザー名。

例外

表 13-25 REGISTER_PROPAGATOR プロシージャの例外

例外	説明
missinguser	指定したユーザーが存在しません。
alreadypropagator	指定したユーザーはすでにプロパゲータです。
duplicatepropagator	別のプロパゲータがすでに存在します。

SCHEDULE_PURGE プロシージャ

このプロシージャは、現行のマスター・サイトまたはマテリアライズド・ビュー・サイトの遅延トランザクション・キューから、送信済みトランザクションをパージするジョブをスケジュールします。スケジュールするパージ・ジョブは 1 つにしてください。

> **関連項目：** このプロシージャを使用した遅延トランザクション・キューの継続的スケジュールまたは定期的パージの詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_DEFER_SYS.SCHEDULE_PURGE (
   interval             IN  VARCHAR2,
   next_date            IN  DATE,
   reset                IN  BOOLEAN        := NULL,
   purge_method         IN  BINARY_INTEGER := NULL,
   rollback_segment     IN  VARCHAR2       := NULL,
   startup_seconds      IN  BINARY_INTEGER := NULL,
   execution_seconds    IN  BINARY_INTEGER := NULL,
   delay_seconds        IN  BINARY_INTEGER := NULL,
   transaction_count    IN  BINARY_INTEGER := NULL,
   write_trace          IN  BOOLEAN        := NULL);
```

パラメータ

表 13-26 SCHEDULE_PURGE プロシージャのパラメータ

パラメータ	説明
interval	パージの次回実行時間を計算するファンクションを提供します。この値は DEFSCHEDULE ビューの interval フィールドに格納され、このビューの next_date フィールドが計算されます。このパラメータのデフォルト値 NULL を使用すると、このフィールドの値は変更されません。フィールドに前の値が設定されていない場合は、NULL で作成されます。このフィールドに値を指定しない場合は、next_date の値を指定する必要があります。
next_date	サイトのキューから送信済みトランザクションをパージする時間を指定できます。この値は DEFSCHEDULE ビューの next_date フィールドに格納されます。このパラメータのデフォルト値 NULL を使用すると、このフィールドの値は変更されません。このフィールドに前の値が設定されていない場合は、NULL で作成されます。このフィールドに値を指定しない場合は、interval の値を指定する必要があります。
reset	TRUE に設定すると、LAST_TXN_COUNT、LAST_ERROR および LAST_MSG の値が NULL にリセットされます。
purge_method	遅延トランザクション・キューのパージ方法を制御します。purge_method_quick はコストが低く、一方 purge_method_precise は精度が高くなります。 purge_method_quick を使用するには、このパラメータに対して次の値を指定します。 dbms_defer_sys.purge_method_quick purge_method_precise を使用するには、このパラメータに対して次の値を指定します。 dbms_defer_sys.purge_method_precise purge_method_quick を使用すると、正常に送信された遅延トランザクションおよび遅延プロシージャ・コールが、パージ前に予測した時間を超えて、DEFTRAN データ・ディクショナリ・ビューおよび DEFCALL データ・ディクショナリ・ビューに残る場合があります。詳細は、13-13 ページの「使用上の注意」を参照してください。ここで説明した使用方法は DBMS_DEFER_SYS.PURGE ファンクション用ですが、DBMS_DEFER_SYS.SCHEDULE_PURGE プロシージャにも適用できます。
rollback_segment	パージに使用するロールバック・セグメント名。デフォルトは NULL です。
startup_seconds	同じ遅延トランザクション・キューの直前のパージを待つ最大秒数。

表 13-26 SCHEDULE_PURGE プロシージャのパラメータ（続き）

パラメータ	説明
execution_seconds	> 0 の場合は、指定した秒数後にパージを即時停止します。
delay_seconds	遅延トランザクション・キューにパージするトランザクションが delay_seconds 秒間ない場合は、パージを停止します。
transaction_count	> 0 の場合は、transaction_count が示す件数のトランザクションをパージした後、シャットダウンします。
write_trace	TRUE に設定すると、PURGE ファンクションによって戻された結果値がサーバーのトレース・ファイルに記録されます。

SCHEDULE_PUSH プロシージャ

このプロシージャは、遅延トランザクション・キューをリモート・サイトに送信するジョブをスケジュールします。このプロシージャは COMMIT を実行します。

> **関連項目：** このプロシージャを使用した遅延トランザクション・キューの継続的スケジュールまたは定期的送信の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_DEFER_SYS.SCHEDULE_PUSH (
   destination          IN  VARCHAR2,
   interval             IN  VARCHAR2,
   next_date            IN  DATE,
   reset                IN  BOOLEAN        := false,
   parallelism          IN  BINARY_INTEGER := NULL,
   heap_size            IN  BINARY_INTEGER := NULL,
   stop_on_error        IN  BOOLEAN        := NULL,
   write_trace          IN  BOOLEAN        := NULL,
   startup_seconds      IN  BINARY_INTEGER := NULL,
   execution_seconds    IN  BINARY_INTEGER := NULL,
   delay_seconds        IN  BINARY_INTEGER := NULL,
   transaction_count    IN  BINARY_INTEGER := NULL);
```

パラメータ

表 13-27 SCHEDULE_PUSH プロシージャのパラメータ

パラメータ	説明
destination	変更内容の転送先であるマスター・サイトまたはマテリアライズド・ビュー・サイトの完全修飾データベース名。
interval	送信の次回実行時間を計算するファンクションを提供します。この値は DEFSCHEDULE ビューの interval フィールドに格納され、このビューの next_date フィールドが計算されます。このパラメータのデフォルト値 NULL を使用すると、このフィールドの値は変更されません。フィールドに前の値が設定されていない場合は、NULL で作成されます。このフィールドに値を指定しない場合は、next_date の値を指定する必要があります。
next_date	遅延トランザクションをリモート・サイトへ送信する時間を指定できます。この値は DEFSCHEDULE ビューの next_date フィールドに格納されます。このパラメータのデフォルト値 NULL を使用すると、このフィールドの値は変更されません。フィールドに前の値が設定されていない場合は、NULL で作成されます。このフィールドに値を指定しない場合は、interval の値を指定する必要があります。
reset	TRUE に設定すると、LAST_TXN_COUNT、LST_ERROR および LAST_MSG の値が NULL にリセットされます。
parallelism	0（ゼロ）はシリアル伝播を指定します。 *n* > 1 は *n* のパラレル処理を使用するパラレル伝播を指定します。 1 は単一のパラレル処理を使用するパラレル伝播を指定します。
heap_size	パラレル伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。 **注意：**オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。
stop_on_error	デフォルトは FALSE で、競合などのエラーが発生しても処理は継続されます。TRUE を設定すると、宛先サイトでトランザクションにエラーが発生したことが最初に通知されたときに伝播を停止します。
write_trace	TRUE に設定すると、ファンクションによって戻された結果値がサーバーのトレース・ファイルに記録されます。
startup_seconds	同じ宛先への直前の送信を待つ最大秒数。
execution_seconds	> 0 の場合は、指定した秒数後に実行を停止します。transaction_count および execution_seconds が 0（デフォルト）の場合は、キュー内のトランザクションがなくなるまで実行されます。

表 13-27 SCHEDULE_PUSH プロシージャのパラメータ（続き）

パラメータ	説明
delay_seconds	キューが空の場合でも、指定した秒数が経過するまで戻しません。PUSH がタイトなループからコールされる場合は、実行オーバーヘッドの削減に役立ちます。
transaction_count	> 0 の場合は、停止までに送信されるトランザクションの最大数。transaction_count および execution_seconds が 0（デフォルト）の場合は、送信する必要があるトランザクションがキュー内になくなるまで実行されます。

SET_DISABLED プロシージャ

このプロシージャは、現行のサイトから指定宛先サイトへの遅延トランザクション・キューの伝播を使用禁止または使用可能にします。disabled パラメータが TRUE の場合は、指定した宛先への伝播が使用禁止になり、その後に PUSH を起動しても、遅延リモート・プロシージャ・コール（RPC）のキューは送信されません。結果的に、SET_DISABLED は、指定した宛先にキューをすでに送信しているセッションには効果がありますが、DBMS_DEFER でキューを追加しているセッションには効果がありません。

disabled パラメータが FALSE の場合は、指定した宛先への伝播は使用可能になります。この処理でキューは送信されませんが、その後に PUSH を起動すると指定した宛先にキューを送信できます。disabled パラメータが TRUE または FALSE に関係なく、他のセッションで効果を発揮する設定にするには COMMIT が必要です。

構文

```
DBMS_DEFER_SYS.SET_DISABLED (
   destination   IN   VARCHAR2,
   disabled      IN   BOOLEAN := true,
   catchup       IN   RAW := '00',
   override      IN   BOOLEAN := false);
```

パラメータ

表 13-28 SET_DISABLED プロシージャのパラメータ

パラメータ	説明
destination	伝播状態を変更するノードの完全修飾データベース名。
disabled	デフォルトでは、このパラメータは現行のサイトから指定した宛先への遅延トランザクション・キューの伝播を使用禁止にします。伝播を使用可能にするには、FALSE に設定してください。
catchup	マスター・グループを停止せずに新規マスター・サイトをマスター・グループへ追加する拡張識別子。新規マスター・サイトが宛先です。既存の拡張識別子については、DEFSCHEDULE データ・ディクショナリ・ビューを問い合せてください。
override	デフォルトの FALSE では、disabled パラメータが FALSE に設定され、Oracle により伝播が内部的に無効化された場合、cantsetdisabled の例外が発生するように指定されます。 TRUE に設定されている場合、Oracle は、同期のために内部的に無効な状態が設定されていたかどうかを無視し、disabled パラメータにより指定された状態に設定するよう常に試行します。 **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

例外

表 13-29 SET_DISABLED プロシージャの例外

例外	説明
NO_DATA_FOUND	DEFSCHEDULE ビューに、指定した destination のエントリが見つかりません。
cantsetdisabled	このサイトの無効な状態は、マスター・グループを停止せずに新規マスター・サイトをマスター・グループへ追加するとき、同期をとるために Oracle によって内部的に設定されます。マスター・グループを停止せずに新規マスター・サイトが追加された後で、プロシージャを起動してください。

UNREGISTER_PROPAGATOR プロシージャ

このプロシージャは、ローカル・データベースからプロパゲータとしてのユーザーの登録を解除します。このプロシージャは次の処理を実行します。

- 指定したプロパゲータを DEFPROPAGATOR から削除します。
- REGISTER_PROPAGATOR で付与した権限を、指定したユーザーから取り消します（個々に付与された同様の権限も含まれます）。
- 指定したプロパゲータのスキーマで生成されたラッパーを削除し、レプリケーション・カタログに削除のマークを設定します。

構文

```
DBMS_DEFER_SYS.UNREGISTER_PROPAGATOR (
   username  IN  VARCHAR2
   timeout   IN  INTEGER DEFAULT DBMS_LOCK.MAXWAIT);
```

パラメータ

表 13-30 UNREGISTER_PROPAGATOR プロシージャのパラメータ

パラメータ	説明
username	プロパゲータのユーザー名。
timeout	タイムアウト（秒数）。プロパゲータが使用中の場合は、タイムアウトまで待機します。デフォルトは DBMS_LOCK.MAXWAIT です。

例外

表 13-31 UNREGISTER_PROPAGATOR プロシージャの例外

パラメータ	説明
missingpropagator	指定したユーザーはプロパゲータではありません。
propagator_inuse	プロパゲータが使用中であるため、登録を解除できません。後で再試行してください。

UNSCHEDULE_PURGE プロシージャ

このプロシージャは、マスター・サイトまたはマテリアライズド・ビュー・サイトの遅延トランザクション・キューからの送信済みトランザクションの自動パージを停止します。

構文

```
DBMS_DEFER_SYS.UNSCHEDULE_PURGE();
```

UNSCHEDULE_PUSH プロシージャ

このプロシージャは、マスター・サイトまたはマテリアライズド・ビュー・サイトからリモート・サイトへの遅延トランザクション・キューの自動送信を停止します。

構文

```
DBMS_DEFER_SYS.UNSCHEDULE_PUSH (
   dblink    IN    VARCHAR2);
```

パラメータ

表 13-32 UNSCHEDULE_PUSH プロシージャのパラメータ

パラメータ	説明
dblink	遅延リモート・プロシージャ・コール（RPC）の定期的な実行のスケジュールを解除するデータベースへの完全修飾パス名。

表 13-33 UNSCHEDULE_PUSH プロシージャの例外

例外	説明
NO_DATA_FOUND	DEFSCHEDULE ビューに、指定した dblink のエントリが見つかりません。

14

DBMS_DESCRIBE

DBMS_DESCRIBE パッケージによって、PL/SQL オブジェクトに関する情報を取得できます。オブジェクト名を指定すると、DBMS_DESCRIBE は結果を含む索引表のセットを戻します。名前変換が完全に実行され、目的のオブジェクトに対するセキュリティ・チェックも行われます。

このパッケージは、Oracle Call Interface の OCIDescribeAny コールと同じ機能を提供します。

関連項目：『Oracle Call Interface プログラマーズ・ガイド』

この章では、次の項目について説明します。

- DBMS_DESCRIBE のセキュリティ、タイプおよびエラー
- DBMS_DESCRIBE サブプログラムの要約

DBMS_DESCRIBE のセキュリティ、タイプおよびエラー

セキュリティ

このパッケージは PUBLIC で使用でき、記述されているスキーマ・オブジェクトに基づいた独自のセキュリティ・チェックが実行されます。

タイプ

DBMS_DESCRIBE パッケージは、2 つのタイプの PL/SQL 表を宣言します。この表は、DESCRIBE_PROCEDURE が戻すデータを、その OUT パラメータに格納するために使用されます。2 つのタイプは次のとおりです。

```
TYPE VARCHAR2_TABLE IS TABLE OF VARCHAR2(30)
    INDEX BY BINARY_INTEGER;

TYPE NUMBER_TABLE IS TABLE OF NUMBER
    INDEX BY BINARY_INTEGER;
```

エラー

DBMS_DESCRIBE では、ORA-20000 ～ ORA-20004 までの範囲のアプリケーション・エラーが発生する可能性があります。

表 14-1 DBMS_DESCRIBE のエラー

エラー	説明
ORA-20000	ORU-10035: cannot describe a package ('X') only a procedure within a package.
ORA-20001	ORU-10032: procedure 'X' within package 'Y' does not exist.
ORA-20002	ORU-10033: object 'X' is remote, cannot describe; expanded name 'Y'.
ORA-20003	ORU-10036: object 'X' is invalid and cannot be described.
ORA-20004	Syntax error attempting to parse 'X'

DBMS_DESCRIBE サブプログラムの要約

DBMS_DESCRIBE に含まれるプロシージャは DESCRIBE_PROCEDURE のみです。

DESCRIBE_PROCEDURE プロシージャ

プロシージャ DESCRIBE_PROCEDURE は、ストアド・プロシージャ名、プロシージャの記述およびその各パラメータを受け入れます。

構文

```
DBMS_DESCRIBE.DESCRIBE_PROCEDURE(
   object_name    IN  VARCHAR2,
   reserved1      IN  VARCHAR2,
   reserved2      IN  VARCHAR2,
   overload       OUT NUMBER_TABLE,
   position       OUT NUMBER_TABLE,
   level          OUT NUMBER_TABLE,
   argument_name  OUT VARCHAR2_TABLE,
   datatype       OUT NUMBER_TABLE,
   default_value  OUT NUMBER_TABLE,
   in_out         OUT NUMBER_TABLE,
   length         OUT NUMBER_TABLE,
   precision      OUT NUMBER_TABLE,
   scale          OUT NUMBER_TABLE,
   radix          OUT NUMBER_TABLE,
   spare          OUT NUMBER_TABLE);
```

パラメータ

表 14-2 DBMS_DESCRIBE.DESCRIBE_PROCEDURE のパラメータ

パラメータ	説明
object_name	記述するプロシージャの名前。 このパラメータの構文は、SQL の識別子に使用されている規則に従っています。名前はシノニムでもかまいません。このパラメータは必須であり、NULL は指定できません。名前の長さは合計で 197 バイトまでです。OBJECT_NAME が正しく指定されないと、次の例外のいずれかが発生する場合があります。 ORA-20000: パッケージが指定されました。指定できるのは、ストアド・プロシージャ、ストアド・ファンクション、パッケージ・プロシージャまたはパッケージ・ファンクションのみです。 ORA-20001: 指定したプロシージャまたはファンクションは、所定のパッケージ内に存在しません。 ORA-20002: 指定したオブジェクトはリモート・オブジェクトです。このプロシージャは、現在リモート・オブジェクトを記述できません。 ORA-20003: 指定したオブジェクトは無効であるため記述できません。 ORA-20004: オブジェクトの指定に構文エラーがあります。

表 14-2 DBMS_DESCRIBE.DESCRIBE_PROCEDURE のパラメータ（続き）

パラメータ	説明
reserved1 reserved2	将来使用される予定です。NULL または空文字列を設定してください。
overload	プロシージャの署名に割り当てられた一意の番号。 プロシージャがオーバーロードされている場合、このフィールドには、プロシージャのバージョンごとに異なる値が格納されます。
position	パラメータ・リスト内の引数の位置。 位置 0（ゼロ）には、ファンクションの戻り値のタイプが戻されます。
level	引数がレコードなどのコンポジット・タイプの場合は、そのデータ・タイプのレベルが戻されます。ODESSP コールの例の詳細は、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。
argument_name	記述するプロシージャに関連付けられた引数の名前。
datatype	記述する引数の Oracle データ・タイプ。 データ・タイプおよびその数値タイプのコードは、次のとおりです。 0 引数を持たないプロシージャのプレースホルダ 1 VARCHAR、VARCHAR、STRING 2 NUMBER、INTEGER、SMALLINT、REAL、FLOAT、DECIMAL 3 BINARY_INTEGER、PLS_INTEGER、POSITIVE、NATURAL 8 LONG 11 ROWID 12 DATE 23 RAW 24 LONG RAW 96 CHAR (ANSI FIXED CHAR)、CHARACTER 106 MLSLABEL 250 PL/SQL RECORD 251 PL/SQL TABLE 252 PL/SQL BOOLEAN
default_value	記述される引数にデフォルト値がある場合は 1、デフォルト値がない場合は 0（ゼロ）です。

表 14-2 DBMS_DESCRIBE.DESCRIBE_PROCEDURE のパラメータ（続き）

パラメータ	説明
in_out	パラメータのモードを記述します。値は次のとおりです。 0 IN 1 OUT 2 IN OUT
length	記述する引数のデータ長（バイト）。データ・タイプがプレースホルダの場合、データ長 0（ゼロ）が返されます。
precision	記述する引数のデータ・タイプが 2（NUMBER）の場合、このパラメータはその数値の精度を表します。
scale	記述する引数のデータ・タイプが 2（NUMBER）の場合、このパラメータはその数値の位取りを表します。
radix	記述する引数のデータ・タイプが 2（NUMBER）の場合、このパラメータはその数値の基数を表します。
spare	将来使用するために予約されています。

戻り値

DESCRIBE_PROCEDURE からの戻り値はすべて、その OUT パラメータに戻されます。このデータ・タイプは、パラメータの変数値に適応するための、PL/SQL の表です。

DBMS_DESCRIBE の使用方法 : 例

DESCRIBE_PROCEDURE プロシージャは、外部サービス・インタフェースとしても使用できます。

たとえば、OBJECT_NAME に SCOTT.ACCOUNT_UPDATE を指定するクライアントを想定します。ACCOUNT_UPDATE は次の仕様を持つオーバーロードされたファンクションです。

```
table account (account_no number, person_id number,
               balance number(7,2))
table person  (person_id number(4), person_nm varchar2(10))

function ACCOUNT_UPDATE (account_no   number,
                         person       person%rowtype,
                         amounts      dbms_describe.number_table,
                         trans_date   date)
                         return       accounts.balance%type;

function ACCOUNT_UPDATE (account_no   number,
                         person       person%rowtype,
                         amounts      dbms_describe.number_table,
                         trans_no     number)
                         return       accounts.balance%type;
```

このプロシージャでは、次のように出力されます。

```
overload position  argument level  datatype length prec scale rad
-------- --------- -------- ------ -------- ------ ---- ----- ---
       1         0                0        2     22    7     2  10
       1         1   ACCOUNT      0        2      0    0     0   0
       1         2   PERSON       0      250      0    0     0   0
       1         1   PERSON_ID    1        2     22    4     0  10
       1         2   PERSON_NM    1        1     10    0     0   0
       1         3   AMOUNTS      0      251      0    0     0   0
       1         1                1        2     22    0     0   0
       1         4   TRANS_DATE   0       12      0    0     0   0
       2         0                0        2     22    7     2  10
       2         1   ACCOUNT_NO   0        2     22    0     0   0
       2         2   PERSON       0        2     22    4     0  10
       2         3   AMOUNTS      0      251     22    4     0  10
       2         1                1        2      0    0     0   0
       2         4   TRANS_NO     0        2      0    0     0   0
```

次の PL/SQL プロシージャには、そのパラメータとしてすべての PL/SQL データ・タイプがあります。

```
CREATE OR REPLACE PROCEDURE p1 (
        pvc2    IN     VARCHAR2,
        pvc     OUT    VARCHAR,
        pstr    IN OUT STRING,
        plong   IN     LONG,
        prowid  IN     ROWID,
        pchara  IN     CHARACTER,
        pchar   IN     CHAR,
        praw    IN     RAW,
        plraw   IN     LONG RAW,
        pbinint IN     BINARY_INTEGER,
        pplsint IN     PLS_INTEGER,
        pbool   IN     BOOLEAN,
        pnat    IN     NATURAL,
        ppos    IN     POSITIVE,
        pposn   IN     POSITIVEN,
        pnatn   IN     NATURALN,
        pnum    IN     NUMBER,
        pintgr  IN     INTEGER,
        pint    IN     INT,
        psmall  IN     SMALLINT,
        pdec    IN     DECIMAL,
        preal   IN     REAL,
        pfloat  IN     FLOAT,
        pnumer  IN     NUMERIC,
        pdp     IN     DOUBLE PRECISION,
        pdate   IN     DATE,
        pmls    IN     MLSLABEL) AS

BEGIN
    NULL;
END;
```

このプロシージャを次のパッケージを使用して記述するとします。

```
CREATE OR REPLACE PACKAGE describe_it AS

    PROCEDURE desc_proc (name VARCHAR2);

END describe_it;

CREATE OR REPLACE PACKAGE BODY describe_it AS

  PROCEDURE prt_value(val VARCHAR2, isize INTEGER) IS
    n INTEGER;
  BEGIN
    n := isize - LENGTHB(val);
    IF n < 0 THEN
      n := 0;
    END IF;
    DBMS_OUTPUT.PUT(val);
    FOR i in 1..n LOOP
      DBMS_OUTPUT.PUT(' ');
    END LOOP;
  END prt_value;

  PROCEDURE desc_proc (name VARCHAR2) IS

      overload     DBMS_DESCRIBE.NUMBER_TABLE;
      position     DBMS_DESCRIBE.NUMBER_TABLE;
      c_level      DBMS_DESCRIBE.NUMBER_TABLE;
      arg_name     DBMS_DESCRIBE.VARCHAR2_TABLE;
      dty          DBMS_DESCRIBE.NUMBER_TABLE;
      def_val      DBMS_DESCRIBE.NUMBER_TABLE;
      p_mode       DBMS_DESCRIBE.NUMBER_TABLE;
      length       DBMS_DESCRIBE.NUMBER_TABLE;
      precision    DBMS_DESCRIBE.NUMBER_TABLE;
      scale        DBMS_DESCRIBE.NUMBER_TABLE;
      radix        DBMS_DESCRIBE.NUMBER_TABLE;
      spare        DBMS_DESCRIBE.NUMBER_TABLE;
      idx          INTEGER := 0;

  BEGIN
      DBMS_DESCRIBE.DESCRIBE_PROCEDURE(
              name,
              null,
              null,
              overload,
              position,
              c_level,
              arg_name,
```

```
                dty,
                def_val,
                p_mode,
                length,
                precision,
                scale,
                radix,
                spare);

        DBMS_OUTPUT.PUT_LINE('Position    Name        DTY  Mode');
        LOOP
            idx := idx + 1;
            prt_value(TO_CHAR(position(idx)), 12);
            prt_value(arg_name(idx), 12);
            prt_value(TO_CHAR(dty(idx)), 5);
            prt_value(TO_CHAR(p_mode(idx)), 5);
            DBMS_OUTPUT.NEW_LINE;
        END LOOP;
    EXCEPTION
       WHEN NO_DATA_FOUND THEN
          DBMS_OUTPUT.NEW_LINE;
          DBMS_OUTPUT.NEW_LINE;

    END desc_proc;
END describe_it;
```

この結果、PL/SQL データ・タイプのすべての数値コードがリストされます。

```
Position  Name     Datatype_Code  Mode
1         PVC2       1              0
2         PVC        1              1
3         PSTR       1              2
4         PLONG      8              0
5         PROWID     11             0
6         PCHARA     96             0
7         PCHAR      96             0
8         PRAW       23             0
9         PLRAW      24             0
10        PBININT    3              0
11        PPLSINT    3              0
12        PBOOL      252            0
13        PNAT       3              0
14        PPOS       3              0
15        PPOSN      3              0
16        PNATN      3              0
17        PNUM       2              0
18        PINTGR     2              0
19        PINT       2              0
20        PSMALL     2              0
21        PDEC       2              0
22        PREAL      2              0
23        PFLOAT     2              0
24        PNUMER     2              0
25        PDP        2              0
26        PDATE      12             0
27        PMLS       106            0
```

使用上の注意

現在、第三世代言語で、タイプが record または boolean の引数は直接バインドできません。ブール値の場合は、次の方法があります。

- ファンクション F がブールを戻すと想定します。G は、1 つの IN ブール引数を持つプロシージャで、H は 1 つの OUT ブール引数を持つプロシージャです。これらのファンクションを DTYINT（システム固有の整数）にバインドして実行できます。この場合、0=>FALSE および 1=>TRUE です。

```
begin :dtyint_bind_var := to_number(f); end;

begin g(to_boolean(:dtyint_bind_var)); end;

declare b boolean; begin h(b); if b then :dtyint_bind_var := 1;
else :dtyint_bind_var := 0; end if; end;
```

- タイプが record の引数を使用するプロシージャにアクセスするには、前述の例（ファンクション H を参照）のラッパーと同様のラッパーを記述する必要があります。

15

DBMS_DISTRIBUTED_TRUST_ADMIN

DBMS_DISTRIBUTED_TRUST_ADMIN プロシージャは、Trusted Servers リストをメンテナンスします。サーバーが信頼されているかどうかを定義するには、これらのプロシージャを使用します。データベースが信頼されていない場合は、そのデータベースからの現行のユーザーのデータベース・リンクは拒否されます。

Oracle は、エンタープライズ LDAP ディレクトリ・サービスに格納されているドメイン・メンバーシップ・リストとともにローカルの Trusted Servers リストを使用して、別のデータベースが信頼されているかどうかを判断します。LDAP ディレクトリ・サービスのエントリは、Oracle Enterprise Manager の Enterprise Security Manager Tool で管理されます。

Oracle は、次の条件に一致する場合、別のデータベースを信頼できるとみなします。

1. ディレクトリ・サービスのエンタープライズ・ドメインがローカル・データベースと同じである。
2. エンタープライズ・ドメインがディレクトリ・サービスにおいて信頼するとマークされている。
3. ローカルの Trusted Servers リストに信頼しないと表示されていない。現行のユーザーのデータベース・リンクが別のデータベースから受け入れられるのは、関係する両方のデータベースが互いに信頼されている場合のみです。

ディレクトリ・サービスにリストされているかどうかに関係なく、データベース・サーバーを Trusted Servers リストにローカルでリストできます。ただし、同じドメイン内にローカル・データベースとして存在しないデータベースをリストする場合、またはそのドメインが信頼されていない場合、そのエントリは無効になります。

この機能は、Oracle Advanced Security オプションの Enterprise User Security 機能の一部です。

この章では、次の項目について説明します。

要件

DBMS_DISTRIBUTED_TRUST_ADMIN を実行するには、EXECUTE_CATALOG_ROLE ロールが DBA に付与されている必要があります。TRUSTED_SERVERS ビューを検索するには、SELECT_CATALOG_ROLE ロールが DBA に付与されている必要があります。

すべてのサーバーについて、信頼されているかどうかを認識することが重要です。データベースがすべてのデータベースをすでに信頼している場合、またはそのデータベースがすでに信頼されている場合は、ALLOW_SERVER プロシージャで特定のサーバーを信頼しても効果はありません。同様に、そのデータベースがすべてのデータベースをすでに信頼していない場合、またはそのデータベースがすでに信頼されていない場合は、DENY_SERVER プロシージャで特定のサーバーを拒否しても効果はありません。

プロシージャ DENY_ALL および ALLOW_ALL は、それぞれ ALLOW_SERVER プロシージャまたは DENY_SERVER プロシージャで明示的に許可または拒否されたすべてのエントリ（サーバー名）を削除します。

DBMS_DISTRIBUTED_TRUST_ADMIN サブプログラムの要約

表 15-1 DBMS_DISTRIBUTED_TRUST_ADMIN パッケージのサブプログラム

サブプログラム	説明
「ALLOW_ALL プロシージャ」15-3 ページ	リストを空にし、すべてのサーバーを信頼することを示す行を挿入します。
「ALLOW_SERVER プロシージャ」15-3 ページ	特定のサーバーへのアクセスを可能にします。リストに deny all が指定されている場合でも有効です。
「DENY_ALL プロシージャ」15-4 ページ	リストを空にし、すべてのサーバーを信頼しないことを示す行を挿入します。
「DENY_SERVER プロシージャ」15-4 ページ	特定のサーバーへのアクセスを拒否します。リストに allow all が指定されている場合でも有効です。

ALLOW_ALL プロシージャ

このプロシージャは、Trusted Servers リストを空にし、エンタープライズ・ディレクトリ・サービスで信頼されているドメインのメンバーであるすべてのサーバー、および同じドメイン内のすべてのサーバーがアクセスを許可されることを指定します。

TRUSTED_SERVERS ビューには、「TRUSTED ALL」と表示されます。これは、エンタープライズ・ディレクトリ・サービスで現在信頼されているすべてのサーバーを、データベースが信頼していることを意味します。

構文

```
DBMS_DISTRIBUTED_TRUST_ADMIN.ALLOW_ALL;
```

使用上の注意

ALLOW_ALL は、エンタープライズ・ディレクトリ・サービスで信頼されているとしてリストされているサーバー、および同じエンタープライズ・ドメイン内のサーバーにのみ適用されます。

ALLOW_SERVER プロシージャ

このプロシージャは、指定したサーバーが信頼されていることを保証します（deny all が指定されている場合でも有効です）。

構文

```
DBMS_DISTRIBUTED_TRUST_ADMIN.ALLOW_SERVER (
   server IN VARCHAR2);
```

パラメータ

表 15-2 ALLOW_SERVER プロシージャのパラメータ

パラメータ	説明
server	信頼するサーバーの一意の完全修飾名。

使用上の注意

Trusted Servers リストにエントリ deny all が含まれている場合、このプロシージャは、特定のデータベース（例：DBx）を信頼することを示す指定を追加します。

Trusted Servers リストにエントリ allow all が含まれており、そのリストに deny DBx エントリがない場合は、このプロシージャを実行しても何も変更されません。

Trusted Servers リストにエントリ allow all が含まれており、そのリストに deny DBx エントリがある場合は、そのエントリが削除されます。

DENY_ALL プロシージャ

このプロシージャは、Trusted Servers リストを空にし、すべてのサーバーがアクセスを拒否されることを指定します。

TRUSTED_SERVERS ビューには、「UNTRUSTED ALL」が表示されます。これは、現在どのサーバーも信頼されていないことを示します。

構文

```
DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_ALL;
```

DENY_SERVER プロシージャ

このプロシージャは、指定したサーバーを信頼しないことを保証します（allow all が指定されている場合でも有効です）。

構文

```
DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_SERVER (
   server IN VARCHAR2);
```

パラメータ

表 15-3 DENY_SERVER プロシージャのパラメータ

パラメータ	説明
server	信頼しないサーバーの一意の完全修飾名。

使用上の注意

Trusted Servers リストにエントリ allow all が含まれている場合、このプロシージャは、指定したデータベース（例：DBx）を信頼しないことを示すエントリを追加します。

Trusted Servers リストにエントリ deny all が含まれており、そのリストに allow DBx エントリがない場合は、このプロシージャを実行しても何も変更されません。

Trusted Servers リストにエントリ deny all が含まれており、allow DBx エントリがある場合は、そのエントリが削除されます。

例

パッケージ DBMS_DISTRIBUTED_TRUST_ADMIN を使用して信頼リストを変更したことがない場合、デフォルトでは、同じ企業ドメイン内のすべてのデータベースを信頼します（そのドメインがディレクトリ・サービスで信頼されているとしてリストされている場合に限ります）。

```
SELECT * FROM TRUSTED_SERVERS;
TRUST      NAME
---------- --------------------
Trusted   All

1 row selected.
```

現在すべてのサーバーが信頼されているため、DENY_SERVER プロシージャを実行すると、特定のサーバーを信頼しないことを指定できます。

```
EXECUTE DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_SERVER
                           ('SALES.US.AMERICAS.ACME_AUTO.COM');

Statement processed.

SELECT * FROM TRUSTED_SERVERS;

TRUST      NAME
---------- -----------------------------------------------
Untrusted SALES.US.AMERICAS.ACME_AUTO.COM

1 row selected
```

DENY_ALL プロシージャを実行すると、すべてのデータベース・サーバーを信頼しないことを選択できます。

```
EXECUTE DBMS_DISTRIBUTED_TRUST_ADMIN.DENY_ALL;

Statement processed.

SELECT * FROM TRUSTED_SERVERS;

TRUST      NAME
---------- -----------------------------------------------
Untrusted All

1 row selected.
```

ALLOW_SERVER プロシージャを使用すると、特定のデータベース・サーバーを信頼することを指定できます。

```
EXECUTE
DBMS_DISTRIBUTED_TRUST_ADMIN.ALLOW_SERVER
                            ('SALES.US.AMERICAS.ACME_AUTO.COM');

Statement processed.

SELECT * FROM TRUSTED_SERVERS;

TRUST      NAME
---------- ------------------------------------------------
Trusted   SALES.US.AMERICAS.ACME_AUTO.COM

1 row selected.
```

16

DBMS_FGA

DBMS_FGA パッケージは、ファイングレイン・セキュリティ機能を提供します。監査方針を管理するには、DBMS_FGA に対する EXECUTE 権限が必要です。監査機能はユーザー環境およびアプリケーション・コンテキスト値をすべて獲得できるため、ポリシーを管理できるのは権限を付与されたユーザーに限定されます。

> **関連項目：** DBMS_FGA の説明および使用情報は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』を参照してください。

この機能は、コストベースの最適化にのみ使用できます。ルールベースのオプティマイザでは、行のフィルタの前に監査の監視が発生する可能性があるため、不要な監査レコードが生成される場合があります。ルールベースのオプティマイザおよびコストベースのオプティマイザのどちらの場合でも、DBA_FGA_AUDIT_TRAIL を参照して、SQL テキストおよび対応して発行されるバインド変数を分析できます。

この章では、次の項目について説明します。

DBMS_FGA サブプログラムの要約

表 16-1 DBMS_FGA サブプログラムの要約

サブプログラム	説明
「ADD_POLICY プロシージャ」16-2 ページ	監査条件として提供された述語を使用して、監査方針を作成します。
「DROP_POLICY プロシージャ」16-3 ページ	監査方針を削除します。
「ENABLE_POLICY プロシージャ」16-4 ページ	監査方針を有効化します。
「DISABLE_POLICY プロシージャ」16-5 ページ	監査方針を無効化します。

ADD_POLICY プロシージャ

このプロシージャは、監査条件として提供された述語を使用して、監査方針を作成します。

構文

```
DBMS_FGA.ADD_POLICY(
   object_schema   VARCHAR2,
   object_name     VARCHAR2,
   policy_name     VARCHAR2,
   audit_condition VARCHAR2,
   audit_column    VARCHAR2,
   handler_schema  VARCHAR2,
   handler_module  VARCHAR2,
   enable          BOOLEAN  );
```

パラメータ

表 16-2 ADD_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	監査するオブジェクトのスキーマ。
object_name	監査するオブジェクトの名前。
policy_name	ポリシーの一意の名前。
audit_condition	監視条件を示す行の条件。
audit_column	アクセスのチェックを行う列。デフォルト値は全列です。

表 16-2 ADD_POLICY プロシージャのパラメータ（続き）

パラメータ	説明
handler_schema	イベント・ハンドラを含むスキーマ。デフォルトは現行のスキーマです。
handler_module	イベント・ハンドラのファンクション名。必要に応じてパッケージ名も含みます。監査条件に一致する最初の行が問合せで処理された後にのみ起動されます。プロシージャが例外で失敗した場合、ユーザーの SQL 文も失敗します。デフォルトは NULL です。
enable	TRUE（デフォルト）の場合、ポリシーを有効化します。

使用上の注意

- 監視条件が TRUE になる場合、イベント・レコードは常に fga_log$ に挿入されます。
- 監査機能には、次のインタフェースが必要です。
  PROCEDURE <fname> (object_schema VARCHAR2, object_name VARCHAR2, policy_name VARCHAR2) AS ...
  ここで fname はプロシージャ名、schema は監査対象の表のスキーマ、table は監査対象の表および policy は施行するポリシーを表しています。
- 監査機能は、自律型トランザクションとして実行されます。
- 各監査方針は、問合せに対し個別に適用されます。つまり、戻される行が表で定義された監査条件のいずれかに適合するかぎり、監査レコードが生成され、各ポリシーにつき最大 1 レコードが生成されます。

DROP_POLICY プロシージャ

このプロシージャは、監査方針を削除します。

構文

```
DBMS_FGA.DROP_POLICY(
   object_schema  VARCHAR2,
   object_name    VARCHAR2,
   policy_name    VARCHAR2 );
```

パラメータ

表 16-3 DROP_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	監査するオブジェクトのスキーマ。
object_name	監査するオブジェクトの名前。
policy_name	ポリシーの一意の名前。

使用上の注意

DBMS_FGA プロシージャは、現行の DML トランザクションがある場合は、操作前にコミットします。ただし、プロシージャが DDL イベント・トリガーの内部にある場合、プロシージャは最初にコミットを実行しません。DDL トランザクションでは、DBMS_FGA は、DDL トランザクションの一部となります。

ENABLE_POLICY プロシージャ

このプロシージャは、監査方針を有効化します。

構文

```
DBMS_FGA.ENABLE_POLICY(
   object_schema  VARCHAR2 := NULL,
   object_name    VARCHAR2,
   policy_name    VARCHAR2,
   enable         BOOLEAN := TRUE);
```

パラメータ

表 16-4 ENABLE_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	監査するオブジェクトのスキーマ。
object_name	監査するオブジェクトの名前。
policy_name	ポリシーの一意の名前。
enable	TRUE（デフォルト）の場合、ポリシーを有効化します。

DISABLE_POLICY プロシージャ

このプロシージャは、監査方針を無効化します。

構文

```
DBMS_FGA.DISABLE_POLICY(
   object_schema  VARCHAR2,
   object_name    VARCHAR2,
   policy_name    VARCHAR2 );
```

パラメータ

表 16-5 DISABLE_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	監査するオブジェクトのスキーマ。
object_name	監査するオブジェクトの名前。
policy_name	ポリシーの一意の名前。

17

DBMS_FLASHBACK

DBMS_FLASHBACK を使用して、指定した実時間または指定したシステム変更番号（SCN）にデータベースのバージョンをフラッシュバックできます。DBMS_FLASHBACK が有効化された場合、ユーザー・セッションはデータベースのフラッシュバック・バージョンを使用します。また、アプリケーションをデータベースのフラッシュバック・バージョンで実行することができます。DBMS_FLASHBACK は、接続切断または別の接続の開始によりセッションが終了すると、自動的に無効化されます。

フラッシュバック・モードで開かれた PL/SQL カーソルは、フラッシュバック時間または SCN の時点での行を戻します。データベースにおける異なる同時セッション（接続）では、異なる実時間または SCN に対してフラッシュバックを実行できます。セッションがフラッシュバック・モードで実行されている間は、DML および DDL 処理および分散処理を行うことはできません。フラッシュバックを無効にして DML を実行するまで、オープンしている PL/SQL カーソルを使用することができます。

自動 UNDO 管理（AUM）モードにおいては、保存制御を使用して、必要なデータベースのバージョンをどの程度の時間の間戻すかを制御できます。24 時間フラッシュバックを実行する必要がある場合、DBA は undo_retention パラメータを 24 時間に設定する必要があります。このように、システムはデータの旧バージョンを再生成するのに十分なロールバック情報を保持しています。

実時間を使用してフラッシュバックを有効化した場合、データベースは指定した時間の 5 分以内に生成された SCN を選択します。フラッシュバックのファイングレイン制御を向上させるために、SCN を有効化できます。SCN は、データベースのバージョンを正確に識別します。フラッシュバックが有効化されたセッションでは、SYSDATE は影響を受けません。引き続き、現在の時間が提供されます。

ログイン・トリガーにおいて DBMS_FLASHBACK を使用し、アプリケーション・コードを変更せずにフラッシュバックを有効化できます。

DBMS_FLASHBACK は、次のような場合に使用できます。

- 自己修復。誤って表から行を削除してしまった場合、その行をリカバリできます。
- 電子メールおよびボイスメールなどのパッケージ・アプリケーション。フラッシュバックを使用して、現行のメッセージ・ボックスに削除したメッセージを再度挿入し、その電子メールをリストアできます。
- 意思決定支援システム（DSS）およびオンライン分析処理（OLAP）アプリケーション。データ分析またはデータ・モデリングを実行し、時期的な需要などを追跡できます。

このパッケージを使用するには、データベース管理者が DBMS_FLASHBACK に対する EXECUTE 権限を付与する必要があります。

> **関連項目：** DBMS_FLASHBACK の詳細は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』および『Oracle9*i* SQL リファレンス』を参照してください。

この章では、次の項目について説明します。

DBMS_FLASHBACK エラー・メッセージ

表 17-1 DBMS_FLASHBACK エラー・メッセージ

エラー	説明
8182	フラッシュバック・モードでは、DML または DDL 操作を実行できません。
8184	ユーザーは、別のフラッシュバック・セッションにおいてフラッシュバックを有効化できません。
8183	ユーザーは、コミットされていないトランザクションにおいてフラッシュバックを有効化できません。
8185	SYS はフラッシュバック・モードを有効化できません。 ユーザーは、フラッシュバック・モードにおいて、読取り専用またはシリアル化可能トランザクションを開始できません。
8180	指定された時間が古すぎます。
8181	無効なシステム変更番号が指定されました。

DBMS_FLASHBACK の使用方法 : 例

高年齢従業員を削除したことにより、その従業員に提出された人員レポートがすべて削除された場合に、フラッシュバックを使用する方法を次に示します。フラッシュバック機能を使用すると、消失した従業員をリカバリし、再度挿入できます。

```
drop table employee;
drop table keep_scn;

REM keep_scn is a temporary table to store scns that we are interested in

create table keep_scn (scn number);
set echo on
create table employee (
   employee_no   number(5) primary key,
   employee_name varchar2(20),
   employee_mgr  number(5)
      constraint mgr_fkey references employee on delete cascade,
   salary        number,
   hiredate      date
);

REM Populate the company with employees
insert into employee values (1, 'John Doe', null, 1000000, '5-jul-81');
insert into employee values (10, 'Joe Johnson', 1, 500000, '12-aug-84');
insert into employee values (20, 'Susie Tiger', 10, 250000, '13-dec-90');
```

```
insert into employee values (100, 'Scott Tiger', 20, 200000, '3-feb-86');
insert into employee values (200, 'Charles Smith', 100, 150000, '22-mar-88');
insert into employee values (210, 'Jane Johnson', 100, 100000, '11-apr-87');
insert into employee values (220, 'Nancy Doe', 100, 100000, '18-sep-93');
insert into employee values (300, 'Gary Smith', 210, 75000, '4-nov-96');
insert into employee values (310, 'Bob Smith', 210, 65000, '3-may-95');
commit;

REM Show the entire org
select lpad(' ', 2*(level-1)) || employee_name Name
from employee
connect by prior employee_no = employee_mgr
start with employee_no = 1
order by level;

REM Sleep for 5 minutes to avoid querying close to the table creation
REM (the mapping of scn->time has 5 minutes granularity)
execute dbms_lock.sleep(300);

REM Store this snapshot for later access through Flashback
declare
I number;
begin
I := dbms_flashback.get_system_change_number;
insert into keep_scn values (I);
commit;
end;
/

REM Scott decides to retire but the transaction is done incorrectly
delete from employee where employee_name = 'Scott Tiger';
commit;

REM notice that all of scott's employees are gone
select lpad(' ', 2*(level-1)) || employee_name Name
from employee
connect by prior employee_no = employee_mgr
start with employee_no = 1
order by level;
```

```
REM Flashback to see Scott's organization
declare
   restore_scn number;
begin
   select  scn into restore_scn from keep_scn;
   dbms_flashback.enable_at_system_change_number (restore_scn);
end;
/

REM Show Scott's org.
select lpad(' ', 2*(level-1)) || employee_name Name
from employee
connect by prior employee_no = employee_mgr
start with employee_no =
   (select employee_no from employee where employee_name = 'Scott Tiger')
order by level;

REM Restore scott's organization.

declare
   scotts_emp number;
   scotts_mgr number;
   cursor c1 is
      select employee_no, employee_name, employee_mgr, salary, hiredate
      from employee
      connect by prior employee_no = employee_mgr
      start with employee_no =
         (select employee_no from employee where employee_name = 'Scott Tiger');
   c1_rec c1 % ROWTYPE;
begin
   select employee_no, employee_mgr into scotts_emp, scotts_mgr from employee
   where employee_name = 'Scott Tiger';
   /* Open c1 in flashback mode */
   open c1;
   /* Disable Flashback */
   dbms_flashback.disable;
 loop
   fetch c1 into c1_rec;
   exit when c1%NOTFOUND;
   /*
     Note that all the DML operations inside the loop are performed
     with Flashback disabled
   */
   if (c1_rec.employee_mgr = scotts_emp) then
      insert into employee values (c1_rec.employee_no,
         c1_rec.employee_name,
         scotts_mgr,
```

```
         c1_rec.salary,
         c1_rec.hiredate);
   else
   if (c1_rec.employee_no != scotts_emp) then
   insert into employee values (c1_rec.employee_no,
         c1_rec.employee_name,
         c1_rec.employee_mgr,
         c1_rec.salary,
         c1_rec.hiredate);
      end if;
    end if;
 end loop;
end;
/

REM Show the restored organization.
select lpad(' ', 2*(level-1)) || employee_name Name
from employee
connect by prior employee_no = employee_mgr
start with employee_no = 1
order by level;
```

DBMS_FLASHBACK サブプログラムの要約

表 17-2 DBMS_FLASHBACK サブプログラム

サブプログラム	説明
「ENABLE_AT_TIME プロシージャ」17-7 ページ	セッション全体においてフラッシュバックを使用できるようにします。スナップショット・タイムは、query_time で指定された時間に最も近い SCN に設定されます。
「ENABLE_AT_SYSTEM_CHANGE_NUMBER プロシージャ」17-8 ページ	SCN を Oracle の数値タイプとして使用し、セッションのスナップショットを指定した数値に設定します。 フラッシュバック・モードでは、すべての問合せにおいて、指定した実時間または SCN の時点と一致したデータが戻されます。
「GET_SYSTEM_CHANGE_NUMBER ファンクション」17-9 ページ	現在の SCN を Oracle の数値タイプとして戻します。SCN を使用して、特定のスナップショットを格納できます。
「DISABLE プロシージャ」17-9 ページ	セッション全体においてフラッシュバック・モードを無効化します。

ENABLE_AT_TIME プロシージャ

このプロシージャは、セッション全体においてフラッシュバックを使用できるようにします。スナップショット・タイムは、query_time で指定された時間に最も近い SCN に設定されます。

構文

```
DBMS_FLASHBACK.ENABLE_AT_TIME (
   query_time    IN TIMESTAMP);
```

パラメータ

表 17-3 ENABLE_AT_TIME プロシージャのパラメータ

パラメータ	説明
query_time	TIMESTAMP タイプの入力パラメータです。タイム・スタンプは次の方法で指定できます。 TIMESTAMP コンストラクタを使用。 例：execute dbms_flashback.enable_at_time(TIMESTAMP '2001-01-09 12:31:00')。 グローバリゼーション・サポート（NLS）フォーマットを使用し、文字列を指定します。フォーマットは、グローバリゼーション・サポートの設定により異なります。 TO_TIMESTAMP ファンクションを使用。 例：execute dbms_flashback.enable_at_time(TO_TIMESTAMP('12-02-2001 14:35:00', 'DD-MM-YYYY HH24:MI:SS'))。 使用する書式を指定します。この例では、2001 年 2 月 12 日午後 2 時 35 分の TO_TIMESTAMP ファンクションを示しています。 問合せ時間から時間が省略された場合、デフォルトはその日の午前 12 時となります。 問合せ時間にタイム・ゾーンが含まれる場合、タイム・ゾーンの情報は切り捨てられます。

ENABLE_AT_SYSTEM_CHANGE_NUMBER プロシージャ

このプロシージャは、SCN を入力パラメータとして使用し、セッションのスナップショットを指定した数値に設定します。

フラッシュバック・モードでは、すべての問合せにおいて、指定した実時間または SCN の時点と一致したデータが戻されます。

構文

```
DBMS_FLASHBACK.ENABLE_AT_SYSTEM_CHANGE_NUMBER (
   query_scn IN NUMBER);
```

パラメータ

表 17-4 ENABLE_AT_SYSTEM_CHANGE_NUMBER プロシージャのパラメータ

パラメータ	説明
query_scn	システム変更番号（SCN）。トランザクションのコミットごとに増分するデータベースのバージョン・ナンバーです。

GET_SYSTEM_CHANGE_NUMBER ファンクション

このファンクションは、現在の SCN を Oracle の数値タイプのデータ・タイプとして戻します。後で使用するために現行の変更番号を取得できます。特定のスナップショットの格納に便利です。

構文

```
DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER
RETURN NUMBER;
```

DISABLE プロシージャ

このプロシージャは、セッション全体においてフラッシュバック・モードを無効化します。

構文

```
DBMS_FLASHBACK.DISABLE;
```

例

次の例は、従業員 Joe の 2000 年 8 月 30 日時点での給与を示しています。

```
EXECUTE dbms_flashback.enable_at_time('30-AUG-2000');
SELECT salary from emp where name = 'Joe'
EXECUTE dbms_flashback.disable;
```

18

DBMS_HS_PASSTHROUGH

パススルー SQL 機能を使用すると、アプリケーション開発者は、Oracle サーバーによる解釈を経由せずに非 Oracle システムに文を直接送信できます。この機能は、非 Oracle システムで使用できる文と同等の文が Oracle にない場合に役立ちます。

PL/SQL パッケージ DBMS_HS_PASSTHROUGH を使用すると、これらの文を非 Oracle システムで直接実行できます。このパッケージで実行される文は、標準の透過的な SQL 文と同じトランザクションで実行されます。

この章では、次の項目について説明します。

セキュリティ

DBMS_HS_PASSTHROUGH パッケージは、概念的には非 Oracle システムに常駐します。パッケージ内のプロシージャおよびファンクションは、非 Oracle システムへの適切なデータベース・リンクを使用してコールする必要があります。

DBMS_HS_PASSTHROUGH サブプログラムの要約

表 18-1 DBMS_HS_PASSTHROUGH パッケージのサブプログラム

サブプログラム	説明
「BIND_VARIABLE プロシージャ」18-3 ページ	位置を基準にして IN 変数を PL/SQL プログラム変数にバインドします。
「BIND_VARIABLE_RAW プロシージャ」18-4 ページ	RAW タイプ IN 変数をバインドします。
「BIND_OUT_VARIABLE プロシージャ」18-5 ページ	OUT 変数を PL/SQL プログラム変数にバインドします。
「BIND_OUT_VARIABLE_RAW プロシージャ」18-7 ページ	RAW データ・タイプの OUT 変数を PL/SQL プログラム変数にバインドします。
「BIND_INOUT_VARIABLE プロシージャ」18-8 ページ	IN OUT バインド変数をバインドします。
「BIND_INOUT_VARIABLE_RAW プロシージャ」18-10 ページ	RAW データ・タイプの IN OUT バインド変数をバインドします。
「CLOSE_CURSOR プロシージャ」18-11 ページ	SQL 文が非 Oracle システムで実行された後、カーソルをクローズし、関連メモリーを解放します。
「EXECUTE_IMMEDIATE ファンクション」18-12 ページ	バインド変数を使用せずに、（SELECT 以外の）SQL 文を即時実行します。
「EXECUTE_NON_QUERY ファンクション」18-13 ページ	（SELECT 以外の）SQL 文を実行します。
「FETCH_ROW ファンクション」18-14 ページ	問合せから行をフェッチします。
「GET_VALUE プロシージャ」18-15 ページ	SELECT 文から列値を取り出す、または OUT バインド・パラメータを取り出します。
「GET_VALUE_RAW プロシージャ」18-17 ページ	RAW データ・タイプが対象である以外は GET_VALUE と同じです。

表 18-1 DBMS_HS_PASSTHROUGH パッケージのサブプログラム（続き）

サブプログラム	説明
「OPEN_CURSOR ファンクション」18-18 ページ	非 Oracle システムでパススルー SQL 文を実行するためのカーソルをオープンします。
「PARSE プロシージャ」18-19 ページ	非 Oracle システムで SQL 文を解析します。

BIND_VARIABLE プロシージャ

このプロシージャは、位置を基準にして IN 変数を PL/SQL プログラム変数にバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_VARIABLE (
   c       IN BINARY_INTEGER NOT NULL,
   pos     IN BINARY_INTEGER NOT NULL,
   val     IN <dty>,
   name    IN VARCHAR2);
```

<dty> は DATE、NUMBER または VARCHAR2 のいずれかです。

> **関連項目：** RAW 変数のバインドには、18-4 ページの「BIND_VARIABLE_RAW プロシージャ」を使用します。

パラメータ

表 18-2 BIND_VARIABLE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	バインド変数名に渡す必要がある値。
name	（オプション）バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 18-3 BIND_VARIABLE プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined: WNDS, RNDS
```

BIND_VARIABLE_RAW プロシージャ

このプロシージャは、RAW タイプ IN 変数をバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_VARIABLE_RAW (
   c      IN BINARY_INTEGER NOT NULL,
   pos    IN BINARY_INTEGER NOT NULL,
   val    IN RAW,
   name   IN VARCHAR2);
```

パラメータ

表 18-4 BIND_VARIABLE_RAW プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	バインド変数に渡す必要がある値。

表 18-4 BIND_VARIABLE_RAW プロシージャのパラメータ（続き）

パラメータ	説明
name	（オプション）バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 18-5 BIND_VARIABLE_RAW プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS, RNDS
```

BIND_OUT_VARIABLE プロシージャ

このプロシージャは、OUT 変数を PL/SQL プログラム変数にバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_OUT_VARIABLE (
   c        IN  BINARY_INTEGER NOT NULL,
   pos      IN  BINARY_INTEGER NOT NULL,
   val      OUT <dty>,
   name     IN  VARCHAR2);
```

<dty> は DATE、NUMBER または VARCHAR2 のいずれかです。

> **関連項目：** RAW データ・タイプの OUT 変数のバインド方法は、18-7 ページの「BIND_OUT_VARIABLE_RAW プロシージャ」を参照してください。

パラメータ

表 18-6 BIND_OUT_VARIABLE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	OUT バインド変数がその値を格納する変数。パッケージは、変数のサイズのみ記憶しています。SQL 文が実行された後、GET_VALUE を使用して OUT パラメータの値を取り出せます。取り出される値のサイズが、BIND_OUT_VARIABLE を使用して渡したパラメータのサイズを超えないようにしてください。
name	（オプション）バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 18-7 BIND_OUT_VARIABLE プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS, RNDS
```

BIND_OUT_VARIABLE_RAW プロシージャ

このプロシージャは、RAW データ・タイプの OUT 変数を PL/SQL プログラム変数にバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_OUT_VARIABLE_RAW (
   c       IN  BINARY_INTEGER NOT NULL,
   pos     IN  BINARY_INTEGER NOT NULL,
   val     OUT RAW,
   name    IN  VARCHAR2);
```

パラメータ

表 18-8 BIND_OUT_VARIABLE_RAW プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	OUT バインド変数がその値を格納する変数。パッケージは、変数のサイズのみ記憶しています。SQL 文が実行された後、GET_VALUE を使用して OUT パラメータの値を取り出せます。取り出される値のサイズは、BIND_OUT_VARIABLE_RAW を使用して渡したパラメータのサイズを超えないようにしてください。
name	（オプション）バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 18-9 BIND_OUT_VARIABLE_RAW プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS, RNDS
```

BIND_INOUT_VARIABLE プロシージャ

このプロシージャは、IN OUT バインド変数をバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_INOUT_VARIABLE (
   c       IN     BINARY_INTEGER NOT NULL,
   pos     IN     BINARY_INTEGER NOT NULL,
   val     IN OUT <dty>,
   name    IN     VARCHAR2);
```

<dty> は DATE、NUMBER または VARCHAR2 のいずれかです。

> **関連項目：** RAW データ・タイプの IN OUT 変数のバインド方法は、18-10 ページの「BIND_INOUT_VARIABLE_RAW プロシージャ」を参照してください。

パラメータ

表 18-10 BIND_INOUT_VARIABLE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	この値は次の 2 つの目的で使用されます。 − SQL 文が実行される前に IN の値を設定します。 − OUT 値のサイズを判別します。
name	（オプション）バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 18-11 BIND_INOUT_VARIABLE プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS, RNDS
```

BIND_INOUT_VARIABLE_RAW プロシージャ

このプロシージャは、RAW データ・タイプの IN OUT 変数をバインドします。

構文

```
DBMS_HS_PASSTHROUGH.BIND_INOUT_VARIABLE (
   c        IN     BINARY_INTEGER NOT NULL,
   pos      IN     BINARY_INTEGER NOT NULL,
   val      IN OUT RAW,
   name     IN     VARCHAR2);
```

パラメータ

表 18-12 BIND_INOUT_VARIABLE_RAW プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数の位置。1 から始まります。
val	この値は次の 2 つの目的で使用されます。 − SQL 文が実行される前に IN の値を設定します。 − OUT 値のサイズを判別します。
name	（オプション）バインド変数名。 たとえば、SELECT * FROM emp WHERE ename=:ename では、バインド変数 :ename の位置は 1、名前は :ename です。このパラメータは、非 Oracle システムで位置によるバインドではなく名前によるバインドがサポートされている場合に使用できます。この場合も位置を渡す必要があります。

例外

表 18-13 BIND_INOUT_VARIABLE_RAW プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS, RNDS
```

CLOSE_CURSOR プロシージャ

このファンクションは、SQL 文が非 Oracle システムで実行された後に、カーソルをクローズし、関連メモリーを解放します。カーソルがオープンされていない場合、操作は何も行われません。

構文

```
DBMS_HS_PASSTHROUGH.CLOSE_CURSOR (
   c IN BINARY_INTEGER NOT NULL);
```

パラメータ

表 18-14 CLOSE_CURSOR プロシージャのパラメータ

パラメータ	説明
c	解放するカーソル。

例外

表 18-15 CLOSE_CURSOR プロシージャの例外

例外	説明
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS, RNDS
```

EXECUTE_IMMEDIATE ファンクション

このファンクションは、SQL 文を即時実行します。有効な SQL コマンド（SELECT を除く）をすぐに実行できます。文にバインド変数を含めることはできません。文は引数で VARCHAR2 として渡されます。SQL 文は、内部的には、OPEN_CURSOR、PARSE、EXECUTE_NON_QUERY、CLOSE_CURSOR のパススルー SQL プロトコル・シーケンスを使用して実行されます。

構文

```
DBMS_HS_PASSTHROUGH.EXECUTE_IMMEDIATE (
   S IN VARCHAR2 NOT NULL)
RETURN BINARY_INTEGER;
```

パラメータ

表 18-16 EXECUTE_IMMEDIATE ファンクションのパラメータ

パラメータ	説明
s	即時実行対象の文を含む VARCHAR2 変数。

戻り値

SQL 文の実行によって影響を受ける行数。

例外

表 18-17 EXECUTE_IMMEDIATE ファンクションの例外

例外	説明
ORA-28551	SQL 文が正しくありません。
ORA-28544	オープンしているカーソルが最大数に達しました。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

EXECUTE_NON_QUERY ファンクション

このファンクションは SQL 文を実行します。SQL 文に SELECT 文は使用できません。SQL 文を実行する前に、カーソルをオープンし、SQL 文を解析する必要があります。

構文

```
DBMS_HS_PASSTHROUGH.EXECUTE_NON_QUERY (
   c IN BINARY_INTEGER NOT NULL)
  RETURN BINARY_INTEGER;
```

パラメータ

表 18-18 EXECUTE_NON_QUERY ファンクションのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。

戻り値

非 Oracle システムで SQL 文によって影響を受ける行数。

例外

表 18-19 EXECUTE_NON_QUERY プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	BIND_VARIABLE プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

FETCH_ROW ファンクション

このファンクションは、結果セットから行をフェッチします。結果セットは、SQL SELECT 文で定義されます。それ以上フェッチする行がなくなると、例外 NO_DATA_FOUND が発生します。行をフェッチする前に、カーソルをオープンし、SQL 文を解析する必要があります。

構文

```
DBMS_HS_PASSTHROUGH.FETCH_ROW (
   c       IN BINARY_INTEGER NOT NULL,
   first   IN BOOLEAN)
  RETURN BINARY_INTEGER;
```

パラメータ

表 18-20 FETCH_ROW ファンクションのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
first	（オプション）SELECT 文を再実行します。設定可能な値は次のとおりです。 － TRUE: SELECT 文を再実行します。 － FALSE: 次の行をフェッチするか、または初めて実行された場合は、SELECT 文を実行して行をフェッチします（デフォルト）。

戻り値

フェッチされた行数が戻されます。最終行がすでにフェッチされた場合は、0（ゼロ）が戻されます。

例外

表 18-21 FETCH_ROW プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS
```

GET_VALUE プロシージャ

このプロシージャには 2 つの目的があります。

- 行がフェッチされた後に、SELECT 文の選択リスト項目を取り出します。
- SQL 文が実行された後に、OUT バインド値を取り出します。

構文

```
DBMS_HS_PASSTHROUGH.GET_VALUE (
   c      IN  BINARY_INTEGER NOT NULL,
   pos    IN  BINARY_INTEGER NOT NULL,
   val    OUT <dty>);
```

<dty> は DATE、NUMBER または VARCHAR2 のいずれかです。

> **関連項目：** RAW データ・タイプの値の取出し方法は、18-17 ページの「GET_VALUE_RAW プロシージャ」を参照してください。

パラメータ

表 18-22 GET_VALUE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数または選択リスト項目の位置。1 から始まります。
val	OUTバインド変数または選択リスト項目がその値を格納する変数。

例外

表 18-23 GET_VALUE プロシージャの例外

例外	説明
ORA-1403	最終行がフェッチされた後に、GET_VALUE を実行すると NO_DATA_FOUND 例外が戻されます（つまり、FETCH_ROW によって 0（ゼロ）が戻されます）。
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります）。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS
```

GET_VALUE_RAW プロシージャ

このプロシージャは、RAW データ・タイプが対象である以外は GET_VALUE と同じです。

構文

```
DBMS_HS_PASSTHROUGH.GET_VALUE_RAW (
   c      IN  BINARY_INTEGER NOT NULL,
   pos    IN  BINARY_INTEGER NOT NULL,
   val    OUT RAW);
```

パラメータ

表 18-24 GET_VALUE_RAW プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。カーソルは、ルーチン OPEN_CURSOR を使用してオープンし、PARSE を使用して解析する必要があります。
pos	SQL 文におけるバインド変数または選択リスト項目の位置。1 から始まります。
val	OUTバインド変数または選択リスト項目がその値を格納する変数。

例外

表 18-25 GET_VALUE_RAW プロシージャの例外

例外	説明
ORA-1403	最終行がフェッチされた後に、GET_VALUE を実行すると NO_DATA_FOUND 例外が戻されます（つまり、FETCH_ROW によって 0（ゼロ）が戻されます)。
ORA-28550	渡されたカーソルが無効です。
ORA-28552	プロシージャが正しい順序で実行されません（最初にカーソルをオープンし、SQL 文を解析する必要があります)。
ORA-28553	バインド変数の位置が有効範囲外です。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS
```

OPEN_CURSOR ファンクション

このファンクションは、非 Oracle システムでパススルー SQL 文を実行するためのカーソルをオープンします。このファンクションは、すべてのタイプの SQL 文に対してコールする必要があります。

後続のコールで使用する必要があるカーソルが戻されます。このコールによってメモリーが割り当てられます。関連付けられたメモリーの割当てを解除するには、プロシージャ CLOSE_CURSOR をコールします。

構文

```
DBMS_HS_PASSTHROUGH.OPEN_CURSOR
  RETURN BINARY_INTEGER;
```

戻り値

後続のプロシージャおよびファンクション・コールで使用されるカーソル。

例外

表 18-26 OPEN_CURSOR ファンクションの例外

例外	説明
ORA-28554	カーソルの最大オープン数を超えました。異機種間サービスの OPEN_CURSORS 初期化パラメータの値を増やしてください。

プラグマ

```
Purity level defined : WNDS, RNDS
```

PARSE プロシージャ

このプロシージャは、非 Oracle システムで SQL 文を解析します。

構文

```
DBMS_HS_PASSTHROUGH.GET_VALUE_RAW (
   c       IN  BINARY_INTEGER NOT NULL,
   stmt    IN  VARCHAR2       NOT NULL);
```

パラメータ

表 18-27 PARSE プロシージャのパラメータ

パラメータ	説明
c	パススルー SQL 文に関連付けられたカーソル。ファンクション OPEN_CURSOR を使用してオープンする必要があります。
stmt	解析する文。

例外

表 18-28 GET_VALUE プロシージャの例外

例外	説明
ORA-28550	渡されたカーソルが無効です。
ORA-28551	SQL 文が正しくありません。
ORA-28555	NOT NULL のパラメータに NULL 値が渡されました。

プラグマ

```
Purity level defined : WNDS, RNDS
```

19

DBMS_IOT

DBMS_IOT パッケージは、索引構成表に対する連鎖行への参照を ANALYZE コマンドを使用して格納できる表を作成します。また、enable_constraint 操作時に、索引構成表の中で制約に違反する行を格納できる例外表も作成できます。

DBMS_IOT は、データベース・インストール時にロードされません。DBMS_IOT をインストールするには、admin ディレクトリにある dbmsiotc.sql と prvtiotc.plb を実行します。

この章では、次の項目について説明します。

DBMS_IOT サブプログラムの要約

表 19-1 DBMS_IOT パッケージのサブプログラム

サブプログラム	説明
「BUILD_CHAIN_ROWS_TABLE プロシージャ」 19-2 ページ	索引構成表に対する連鎖行への参照を ANALYZE コマンドを使用して格納できる表を作成します。
「BUILD_EXCEPTIONS_TABLE プロシージャ」 19-3 ページ	enable_constraint 操作時に、索引構成表の中で制約に違反する行を格納できる例外表を作成します。

BUILD_CHAIN_ROWS_TABLE プロシージャ

BUILD_CHAIN_ROWS_TABLE プロシージャは、索引構成表に対する連鎖行への参照を ANALYZE コマンドを使用して格納できる表を作成します。

構文

```
DBMS_IOT.BUILD_CHAIN_ROWS_TABLE (
   owner               IN VARCHAR2,
   iot_name            IN VARCHAR2,
   chainrow_table_name IN VARCHAR2 default 'IOT_CHAINED_ROWS');
```

パラメータ

表 19-2 BUILD_CHAIN_ROWS_TABLE プロシージャのパラメータ

パラメータ	説明
owner	索引構成表の所有者。
iot_name	索引構成表名。
chainrow_table_name	連鎖行表の目的名。

例

```
CREATE TABLE l(a char(16),b char(16), c char(16), d char(240),
PRIMARY KEY(a,b,c)) ORGANIZATION INDEX pctthreshold 10 overflow;
EXECUTE DBMS_IOT.BUILD_CHAIN_ROWS_TABLE('SYS','L','LC');
```

次の列を含んだ連鎖行表が作成されます。

```
Column Name                    Null?    Type
------------------------------ -------- ----
OWNER_NAME                              VARCHAR2(30)
TABLE_NAME                              VARCHAR2(30)
CLUSTER_NAME                            VARCHAR2(30)
PARTITION_NAME                          VARCHAR2(30)
SUBPARTITION_NAME                       VARCHAR2(30)
HEAD_ROWID                              ROWID
TIMESTAMP                               DATE
A                                       CHAR(16)
B                                       CHAR(16)
C                                       CHAR(16)
```

BUILD_EXCEPTIONS_TABLE プロシージャ

BUILD_EXCEPTIONS_TABLE プロシージャは、enable_constraint 操作時に、索引構成表の中で制約に違反する行を格納できる例外表を作成します。

各索引構成表について、その主キーに対応する別々の連鎖行表および例外表を作成する必要があります。

> **注意：** このフォームの連鎖行表と例外表は、Oracle8、リリース 8.0 互換で稼働しているサーバーにのみ必要です。

構文

```
DBMS_IOT.BUILD_EXCEPTIONS_TABLE (
   owner                IN VARCHAR2,
   iot_name             IN VARCHAR2,
   exceptions_table_name IN VARCHAR2 default 'IOT_EXCEPTIONS');
```

パラメータ

表 19-3 BUILD_EXCEPTIONS_TABLE プロシージャのパラメータ

パラメータ	説明
owner	索引構成表の所有者。
iot_name	索引構成表名。
exceptions_table_name	例外表の目的名。

例

```
EXECUTE DBMS_IOT.BUILD_EXCEPTIONS_TABLE('SYS','L','LE');
```

前述の索引構成表に対する例外表の列は、次のようになります。

```
Column Name                    Null?    Type
------------------------------ -------- ----
ROW_ID                                  VARCHAR2(30)
OWNER                                   VARCHAR2(30)
TABLE_NAME                              VARCHAR2(30)
CONSTRAINT                              VARCHAR2(30)
A                                       CHAR(16)
B                                       CHAR(16)
C                                       CHAR(16)
```

20

DBMS_JOB

DBMS_JOB サブプログラムは、ジョブ・キュー内のジョブをスケジュールおよび管理します。

> **関連項目：** DBMS_JOB パッケージおよびジョブ・キューの詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

この章では、次の項目について説明します。

要件

ジョブに関連付けられているデータベース権限はありません。DBMS_JOB では、ユーザーのジョブ以外のジョブにはアクセスできません。

Oracle Real Application Clusters での DBMS_JOB パッケージの使用方法

この例では、DBMS_JOB 内の定数は、ジョブとインスタンスの間でマッピングなしを示します。つまり、いずれのインスタンスでもジョブを実行できます。

DBMS_JOB.SUBMIT

ジョブをジョブ・キューに送るには、次の構文を使用します。

```
DBMS_JOB.SUBMIT( JOB OUT BINARY_INTEGER,
WHAT     IN VARCHAR2, NEXT_DATE IN DATE DEFAULTSYSDATE,
INTERVAL IN VARCHAR2 DEFAULT 'NULL',
NO_PARSE IN BOOLEAN DEFAULT FALSE,
INSTANCE IN BINARY_INTEGER DEFAULT ANY_INSTANCE,
FORCE    IN BOOLEAN DEFAULT FALSE);
```

パラメータ INSTANCE および FORCE を使用して、ジョブおよびインスタンスの親和性を制御します。INSTANCE のデフォルト値は 0（ゼロ）で、いずれのインスタンスでもジョブを実行できることを示します。特定のインスタンスでジョブを実行するには、INSTANCE の値を指定します。INSTANCE の値が負数または NULL の場合は、エラー ORA-23319 が表示されます。

FORCE パラメータは FALSE にデフォルト設定されます。TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。FORCE が FALSE の場合は、指定したインスタンスが実行されている必要があります。実行されていない場合は、エラー番号 ORA-23428 が表示されます。

DBMS_JOB.INSTANCE

ジョブを実行するために特定のインスタンスを割り当てるには、次の構文を使用します。

```
DBMS_JOB.INSTANCE(  JOB IN BINARY_INTEGER,
INSTANCE                IN BINARY_INTEGER,
FORCE                   IN BOOLEAN DEFAULT FALSE);
```

この例の FORCE パラメータは FALSE にデフォルト設定されます。INSTANCE の値が 0（ゼロ）の場合は、ジョブ親和性が変更され、FORCE の値に関係なく、使用可能なすべてのインスタンスがジョブを実行できます。INSTANCE の値が正数で FORCE パラメータが FALSE の場合、ジョブ親和性は、指定したインスタンスが実行されている場合のみ変更されます。実行されていない場合は、エラー ORA-23428 が表示されます。

FORCE パラメータが TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられ、ジョブ親和性が変更されます。INSTANCE の値が負数または NULL の場合は、エラー ORA-23319 が表示されます。

DBMS_JOB.CHANGE

ジョブに関連付けられたユーザー定義可能なパラメータを変更するには、次の構文を使用します。

```
DBMS_JOB.CHANGE(  JOB IN BINARY_INTEGER,
WHAT                  IN VARCHAR2 DEFAULT NULL,
NEXT_DATE             IN DATE DEFAULT NULL,
INTERVAL              IN VARCHAR2 DEFAULT NULL,
INSTANCE              IN BINARY_INTEGER DEFAULT NULL,
FORCE                 IN BOOLEAN DEFAULT FALSE );
```

この例では、INSTANCE および FORCE の 2 つのパラメータが使用されています。INSTANCE のデフォルト値は NULL で、ジョブ親和性を変更しないことを示しています。

FORCE のデフォルト値は FALSE です。指定したインスタンスが実行されていない場合はエラー ORA-23428 が、INSTANCE の数値が負数の場合はエラー ORA-23319 が表示されます。

DBMS_JOB.RUN

DBMS_JOB.RUN の FORCE パラメータは FALSE にデフォルト設定されます。FORCE が TRUE の場合は、フォアグラウンド・プロセスでのジョブ実行にインスタンス親和性は無関係です。FORCE が FALSE の場合は、指定したインスタンスでのみフォアグラウンドでジョブを実行できます。FORCE が FALSE で、接続先のインスタンスが不適切なインスタンスの場合は、エラー ORA-23428 が表示されます。

```
DBMS_JOB.RUN(
   JOB    IN BINARY_INTEGER,
   FORCE  IN BOOLEAN DEFAULT FALSE);
```

> **関連項目：** 詳細は、『Oracle9*i* Real Application Clusters 概要』を参照してください。

DBMS_JOB サブプログラムの要約

表 20-1 DBMS_JOB パッケージのサブプログラム

サブプログラム	説明
「SUBMIT プロシージャ」 20-5 ページ	新規ジョブをジョブ・キューに送ります。
「REMOVE プロシージャ」 20-6 ページ	指定したジョブをジョブ・キューから削除します。
「CHANGE プロシージャ」 20-7 ページ	ジョブに関連付けられているユーザー定義パラメータを変更します。
「WHAT プロシージャ」 20-8 ページ	指定したジョブに関するジョブの記述を変更します。
「NEXT_DATE プロシージャ」 20-9 ページ	指定したジョブの次の実行時間を変更します。
「INSTANCE プロシージャ」 20-9 ページ	インスタンスでジョブが実行されるように割り当てます。
「INTERVAL プロシージャ」 20-10 ページ	指定したジョブの実行間隔を変更します。
「BROKEN プロシージャ」 20-11 ページ	ジョブの実行を禁止します。
「RUN プロシージャ」 20-11 ページ	指定したジョブを強制的に実行します。
「USER_EXPORT プロシージャ」 20-12 ページ	指定したジョブをエクスポート用に再作成します。
「USER_EXPORT プロシージャ」 20-13 ページ	指定したジョブをインスタンス親和性付きでエクスポート用に再作成します。

SUBMIT プロシージャ

このプロシージャは新規ジョブを送信します。順序 sys.jobseq からジョブを選択します。

構文

```
DBMS_JOB.SUBMIT (
   job       OUT BINARY_INTEGER,
   what      IN  VARCHAR2,
   next_date IN  DATE DEFAULT sysdate,
   interval  IN  VARCHAR2 DEFAULT 'null',
   no_parse  IN  BOOLEAN DEFAULT FALSE,
   instance  IN  BINARY_INTEGER DEFAULT any_instance,
   force     IN  BOOLEAN DEFAULT FALSE);
```

パラメータ

表 20-2 SUBMIT プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
what	実行する PL/SQL プロシージャ。
next_date	ジョブを次回実行する日付。
interval	ジョブを次回実行する時間を計算する日付ファンクション。デフォルトは NULL です。このファンクションは、将来の日時または NULL に設定される必要があります。
no_parse	フラグ。デフォルトは FALSE です。FALSE に設定すると、ジョブに関連付けられているプロシージャが解析されます。TRUE に設定すると、ジョブに関連付けられているプロシージャがそのジョブの初回実行時に解析されます。 たとえば、ジョブに関連付けられている表を作成する前にそのジョブを送信する場合は、この値を TRUE に設定します。
instance	ジョブが送信されたときに、そのジョブを実行できるインスタンスを指定します。
force	TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。FALSE（デフォルト）の場合は、指定したインスタンスで実行する必要があり、そうでない場合は、例外が発生します。

使用上の注意

パラメータ instance および force がジョブ・キュー親和性のために追加されています。ジョブ・キュー親和性によって、ユーザーは、送信されたジョブを特定のインスタンスで実行するか、またはどのインスタンスでも実行可能とするかを指示できます。

例

新規ジョブをジョブ・キューに送る例です。このジョブは、プロシージャ DBMS_DDL.ANALYZE_OBJECT をコールし、表 DQUON.ACCOUNTS に関するオプティマイザの統計情報を生成します。統計情報は、ACCOUNTS 表にある行の半分をサンプルとして使用します。このジョブは 24 時間ごとに実行されます。

```
VARIABLE jobno number;
BEGIN
   DBMS_JOB.SUBMIT(:jobno,
      'dbms_ddl.analyze_object(''TABLE'',
      ''DQUON'', ''ACCOUNTS'',
      ''ESTIMATE'', NULL, 50);'
      SYSDATE, 'SYSDATE + 1');
   commit;
END;
/
Statement processed.
print jobno
JOBNO
----------
14144
```

REMOVE プロシージャ

このプロシージャは、ジョブ・キューから既存のジョブを削除します。実行中のジョブを停止する機能は、現在はありません。

構文

```
DBMS_JOB.REMOVE (
   job       IN  BINARY_INTEGER );
```

パラメータ

表 20-3 REMOVE プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。

例

```
EXECUTE DBMS_JOB.REMOVE(14144);
```

CHANGE プロシージャ

このプロシージャは、ジョブ内のユーザー設定可能フィールドを変更します。

構文

```
DBMS_JOB.CHANGE (
   job       IN  BINARY_INTEGER,
   what      IN  VARCHAR2,
   next_date IN  DATE,
   interval  IN  VARCHAR2,
   instance  IN  BINARY_INTEGER DEFAULT NULL,
   force     IN  BOOLEAN DEFAULT FALSE);
```

パラメータ

表 20-4 CHANGE プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
what	実行する PL/SQL プロシージャ。
next_date	次回のリフレッシュ日付。
interval	日付ファンクション。ジョブ実行の直前に評価されます。
instance	ジョブが送信されたときに、そのジョブを実行できるインスタンスを指定します。デフォルトは NULL で、インスタンス親和性が変更されないことを示します。
force	FALSE の場合、（インスタンス番号を変更する対象の）指定インスタンスで実行する必要があります。そうでない場合は、例外が発生します。 TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。

使用上の注意

パラメータ instance および force がジョブ・キュー親和性のために追加されています。ジョブ・キュー親和性によって、ユーザーは、送信されたジョブを特定のインスタンスで実行するか、またはどのインスタンスでも実行可能とするかを指示できます。

パラメータ what、next_date または interval が NULL の場合、現在の値は変更されません。

例

```
EXECUTE DBMS_JOB.CHANGE(14144, null, null, 'sysdate+3');
```

WHAT プロシージャ

このプロシージャは、既存のジョブが実行する内容を変更し、その環境を置き換えます。

構文

```
DBMS_JOB.WHAT (
   job       IN  BINARY_INTEGER,
   what      IN  VARCHAR2);
```

パラメータ

表 20-5 WHAT プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
what	実行する PL/SQL プロシージャ。

次に正しい what の値の例（ルーチンが存在していると仮定）を示します。

- 'myproc(''10-JAN-82'', next_date, broken);'
- 'scott.emppackage.give_raise(''JENKINS'', 30000.00);'
- 'dbms_job.remove(job);'

NEXT_DATE プロシージャ

このプロシージャは、既存のジョブの次回実行時間を変更します。

構文

```
DBMS_JOB.NEXT_DATE (
   job       IN  BINARY_INTEGER,
   next_date IN  DATE);
```

パラメータ

表 20-6 NEXT_DATE プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
next_date	次回のリフレッシュ日付。ジョブはこの日付に自動的に実行されます。ただし、ジョブを実行するバックグラウンド・プロセスがあることが前提です。

INSTANCE プロシージャ

このプロシージャは、ジョブ・インスタンス親和性を変更します。

構文

```
DBMS_JOB.INSTANCE (
   job        IN BINARY_INTEGER,
   instance   IN BINARY_INTEGER,
   force      IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 20-7 INSTANCE プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
instance	ジョブを送信するときに、ユーザーはジョブを実行できるインスタンスを指定できます。
force	TRUE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。FALSE（デフォルト）の場合は、指定したインスタンスで実行する必要があり、そうでない場合は、例外が発生します。

INTERVAL プロシージャ

このプロシージャは、ジョブの実行間隔を変更します。

構文

```
DBMS_JOB.INTERVAL (
   job       IN  BINARY_INTEGER,
   interval  IN  VARCHAR2);
```

パラメータ

表 20-8 INTERVAL プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
interval	日付ファンクション。ジョブの実行直前に評価されます。

使用上の注意

ジョブが正常に完了すると、next_date にこの新しい日付が設定されます。interval は、その日付を select interval into next_date 文に接続することによって評価されます。

interval パラメータの評価結果は将来の日時に設定される必要があります。有効な間隔の例は次のとおりです。

間隔	説明
'sysdate + 7'	毎週 1 回実行。
'next_day(sysdate,''TUESDAY'')'	毎週火曜日に実行。
'null'	1 回のみ実行。

interval の評価結果が NULL で、ジョブが正常に完了した場合、そのジョブはキューから自動的に削除されます。

BROKEN プロシージャ

このプロシージャは中断フラグを設定します。中断状態のジョブはその後実行されません。

構文

```
DBMS_JOB.BROKEN (
   job       IN  BINARY_INTEGER,
   broken    IN  BOOLEAN,
   next_date IN  DATE DEFAULT SYSDATE);
```

パラメータ

表 20-9 Broken プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
broken	ジョブ中断 : IN の値は FALSE です。
next_date	次回のリフレッシュ日付。

> **注意 :** 実行中のジョブに中断を指定すると、ジョブの完了後、ジョブのステータスは normal にリセットされます。したがって、このプロシージャは、実行中でないジョブに対してのみ実行してください。

RUN プロシージャ

このプロシージャは、ジョブ JOB をすぐに実行します。そのジョブが中断されている場合でも実行します。

ジョブが実行されると、next_date が再計算されます。user_jobs ビューを参照してください。

構文

```
DBMS_JOB.RUN (
   job       IN  BINARY_INTEGER,
   force     IN  BOOLEAN DEFAULT FALSE);
```

パラメータ

表 20-10 Run プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
force	TRUE の場合、フォアグラウンド・プロセスでのジョブの実行にインスタンス親和性は無関係です。FALSE の場合は、指定したインスタンスでのみフォアグラウンドでジョブを実行できます。

例

```
EXECUTE DBMS_JOB.RUN(14144);
```

注意： これは、現行セッションのパッケージを再初期化します。

例外

force が FALSE で、接続インスタンスが正しくない場合は、例外が発生します。

USER_EXPORT プロシージャ

このプロシージャは、指定したジョブを再作成するコールのテキストを生成します。

構文

```
DBMS_JOB.USER_EXPORT (
   job    IN     BINARY_INTEGER,
   mycall IN OUT VARCHAR2);
```

パラメータ

表 20-11 USER_EXPORT プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
mycall	指定したジョブを再作成するコールのテキスト。

USER_EXPORT プロシージャ

このプロシージャは、インスタンス親和性（8*i* 以上）を変更し、互換性を保ちます。

構文

```
DBMS_JOB.USER_EXPORT (
   job      IN     BINARY_INTEGER,
   mycall   IN OUT VARCHAR2,
   myinst   IN OUT VARCHAR2);
```

パラメータ

表 20-12 USER_EXPORT プロシージャのパラメータ

パラメータ	説明
job	実行するジョブの番号。
mycall	指定したジョブを再作成するコールのテキスト。
myinst	インスタンス親和性を変更するコールのテキスト。

21

DBMS_LDAP

DBMS_LDAP は、LDAP サーバーからデータにアクセスする機能および手順を提供します。DBMS_LDAP を使用するには、まずデータベースにロードする必要があります。$ORACLE_HOME/rdbms/admin ディレクトリの catldap.sql スクリプトを使用してください。

> **関連項目：** DBMS_LDAP の使用方法の詳細は、『Oracle Internet Directory アプリケーション開発者ガイド』を参照してください。

この章では、次の項目について説明します。

例外の要約

表 21-1 に、DBMS_LDAP により生成される例外を示します。

表 21-1 DBMS_LDAP 例外の要約

例外名	Oracle エラー	例外の原因
general_error	31202	特定の PL/SQL 例外が関連付けられていないエラーが発生した場合、常に呼び出されます。エラー文字列には、ユーザーのローカル言語で問題の説明が表示されます。
init_failed	31203	なんらかの問題が発生した場合、DBMS_LDAP.init により呼び出されます。
invalid_session	31204	無効なセッション・ハンドルが渡された場合、DBMS_LDAP パッケージのすべてのファンクションおよびプロシージャにより呼び出されます。
invalid_auth_method	31205	要求された認証方式がサポートされていない場合、DBMS_LDAP.bind_s により呼び出されます。
invalid_search_scope	31206	検索範囲が無効である場合に、すべての検索ファンクションにより呼び出されます。
invalid_search_time_val	31207	制限時間の値が無効である場合、時間ベースの検索ファンクション（DBMS_LDAP.search_st）により呼び出されます。
invalid_message	31208	検索処理からエントリを取得するために指定されたメッセージ・ハンドルが無効である場合、結果セットを使用して繰り返されるすべてのファンクションにより呼び出されます。
count_entry_error	31209	指定された結果セットにおいてエントリをカウントできない場合、DBMS_LDAP.count_entries により呼び出されます。
get_dn_error	31210	取り出すエントリの DN が NULL である場合、DBMS_LDAP.get_dn により呼び出されます。
invalid_entry_dn	31211	無効なエントリ DN が提示されている場合、エントリを変更、追加または改名するすべてのファンクションにより呼び出されます。
invalid_mod_array	31212	無効な変更配列が指定された場合に、変更配列を引数として使用するすべてのファンクションによって呼び出されます。
invalid_mod_option	31213	MOD_ADD、MOD_DELETE または MOD_REPLACE 以外の変更オプションが指定された場合、DBMS_LDAP.populate_mod_array により呼び出されます。
invalid_mod_type	31214	変更される属性のタイプが NULL である場合、DBMS_LDAP.populate_mod_array により呼び出されます。
invalid_mod_value	31215	指定された属性の変更する値パラメータが NULL である場合、DBMS_LDAP.populate_mod_array により呼び出されます。
invalid_rdn	31216	有効な RDN を予測するすべてのファンクションおよびプロシージャにおいて、RDN の値が NULL である場合に発生します。

表 21-1 DBMS_LDAP 例外の要約（続き）

例外名	Oracle エラー	例外の原因
invalid_ newparent	31217	改名されるエントリの新しい親が NULL である場合、DBMS_LDAP.rename_s により呼び出されます。
invalid_ deleteoldrdn	31218	deleteoldrdn パラメータが無効である場合、DBMS_LDAP.rename_s により呼び出されます。
invalid_ notypes	31219	notypes パラメータが無効である場合、DBMS_LDAP.explode_dn により呼び出されます。
invalid_ssl_ wallet_loc	31220	Wallet の場所が NULL だが、SSL 認証モードで有効な Wallet が要求されている場合、DBMS_LDAP.open_ssl により呼び出されます。
invalid_ssl_ wallet_ password	31221	指定された Wallet のパスワードが NULL である場合、DBMS_LDAP.open_ssl により呼び出されます。
invalid_ssl_ auth_mode	31222	SSL 認証モードが 1、2 または 3 のいずれにも該当しない場合、DBMS_LDAP.open_ssl により呼び出されます。
mts_mode_not_ supported	31398	init、bind_s または simple_bind_s ファンクションが MTS モードで起動されたことがある場合、これらのファンクションにより呼び出されます。

データ・タイプの要約

DBMS_LDAP パッケージが使用するデータ・タイプを表 21-2 に示します。

表 21-2 DBMS_LDAP データ・タイプの要約

データ・タイプ	用途
SESSION	LDAP セッションのハンドルを保持します。API のファンクションを処理するには、ほとんどすべての場合、有効な LDAP セッションが必要です。
MESSAGE	結果セットから取り出されたメッセージへのハンドルを保持します。エントリ、属性および値を処理するすべてのファンクションにおいて使用されます。
MOD_ARRAY	modify_s または add_s のいずれかに渡される変更配列へのハンドルを保持します。
TIMEVAL	制限時間を必要とする LDAP API ファンクションに制限時間情報を渡します。

表 21-2 DBMS_LDAP データ・タイプの要約（続き）

データ・タイプ	用途
BER_ELEMENT	渡されたメッセージのデコードに使用される BER データ構造へのハンドルを保持します。
STRING_COLLECTION	LDAP サーバーに渡せる VARCHAR2 文字列のリストを保持します。
BINVAL_COLLECTION	バイナリ・データを表す RAW データのリストを保持します。
BERVAL_COLLECTION	変更配列の移入に使用される BERVAL 値のリストを保持します。

DBMS_LDAP サブプログラムの要約

表 21-3 DBMS_LDAP のサブプログラム

ファンクションまたはプロシージャ	説明
「init ファンクション」21-6 ページ	LDAP サーバーとのセッションを初期化します。これにより、LDAP サーバーとの接続が実際に確立されます。
「simple_bind_s ファンクション」21-8 ページ	ディレクトリ・サーバーに対し、ユーザー名およびパスワードに基づいた単純な認証を実行します。
「bind_s ファンクション」21-9 ページ	ディレクトリ・サーバーに対し、複雑な認証を実行します。
「unbind_s ファンクション」21-11 ページ	アクティブな LDAP セッションをクローズします。
「compare_s ファンクション」21-12 ページ	特定のエントリにおける特定の属性が特定の値を持っているかどうかをテストします。
「search_s ファンクション」21-14 ページ	LDAP サーバーにおいて同期検索を実行します。すべての検索結果がサーバーにより送信された後または検索要求がサーバー側でタイムアウトした場合にのみ、PL/SQL 環境に制御が戻されます。
「search_st ファンクション」21-16 ページ	クライアント側をタイムアウトにして、LDAP サーバーにおいて同期検索を実行します。すべての検索結果がサーバーにより送信された後または検索要求がクライアントまたはサーバー側でタイムアウトした場合にのみ、PL/SQL 環境に制御が戻されます。
「first_entry ファンクション」21-18 ページ	search_s または search_st のいずれかにより戻された結果セットの最初のエントリを取り出します。
「next_entry ファンクション」21-19 ページ	検索処理の結果セット内の次のエントリを繰り返します。

表 21-3 DBMS_LDAP のサブプログラム（続き）

ファンクションまたはプロシージャ	説明
「count_entries ファンクション」21-21 ページ	結果セットのエントリ数をカウントします。first_entry および next_entry ファンクションを組み合せて使用して結果セットを走査する間、残存するエントリ数をカウントすることも可能です。
「first_attribute ファンクション」21-22 ページ	結果セットにおいて指定されたエントリの最初の属性をフェッチします。
「next_attribute ファンクション」21-24 ページ	結果セットにおいて指定されたエントリの次の属性をフェッチします。
「get_dn ファンクション」21-25 ページ	結果セットにおいて指定されたエントリの X.500 識別名を取り出します。
「get_values ファンクション」21-26 ページ	指定されたエントリにおける指定された属性に関連付けられたすべての値を取り出します。
「get_values_len ファンクション」21-28 ページ	バイナリ構文を持つ属性の値を取り出します。
「delete_s ファンクション」21-29 ページ	LDAP ディレクトリ情報ツリーのリーフ・エントリを削除します。
「modrdn2_s ファンクション」21-30 ページ	エントリの相対識別名を改名します。
「err2string ファンクション」21-32 ページ	LDAP エラー・コードを API が処理を行うローカル言語の文字列に変換します。
「create_mod_array ファンクション」21-33 ページ	modify_s ファンクションを使用してエントリに適用される配列変更エントリに、メモリーを割り当てます。
「populate_mod_array（文字列バージョン）プロシージャ」21-34 ページ	追加または変更処理を行う属性情報を 1 セット移入します。
「populate_mod_array（バイナリ・バージョン）プロシージャ」21-35 ページ	追加または変更処理を行う属性情報を 1 セット移入します。このプロシージャ・コールは、DBMS_LDAP.create_mod_array がコールされた後に発生させる必要があります。
「modify_s ファンクション」21-37 ページ	既存の LDAP ディレクトリ・エントリの同期変更を実行します。
「add_s ファンクション」21-38 ページ	LDAP ディレクトリに対し、新規エントリを同期式で追加します。add_s をコールする前に、DBMS_LDAP.creat_mod_array および DBMS_LDAP.populate_mod_array をコールする必要があります。

表 21-3 DBMS_LDAP のサブプログラム（続き）

ファンクションまたはプロシージャ	説明
「free_mod_array プロシージャ」21-40 ページ	DBMS_LDAP.create_mod_array により割り当てられたメモリーを解放します。
「count_values ファンクション」21-41 ページ	DBMS_LDAP.get_values により戻された値の数をカウントします。
「count_values_len ファンクション」21-42 ページ	DBMS_LDAP.get_values_len により戻された値の数をカウントします。
「rename_s ファンクション」21-43 ページ	LDAP エントリを同期式で改名します。
「explode_dn ファンクション」21-44 ページ	DN をコンポーネントに分割します。
「open_ssl ファンクション」21-45 ページ	既存の LDAP 接続において、Secure Sockets Layer（SSL）接続を確立します。

init ファンクション

このファンクションは、LDAP サーバーとのセッションを初期化します。これにより、LDAP サーバーとの接続が実際に確立されます。

構文

```
DBMS_LDAP.init (
   hostname  IN VARCHAR2,
   portnum   IN PLS_INTEGER )
RETURN SESSION;
```

パラメータ

表 21-4 init ファンクションのパラメータ

パラメータ	説明
hostname (IN)	スペース区切りのホスト名リストまたは LDAP サーバーを実行するホストの IP アドレスを表すドット区切りの文字列を含みます。リストの各ホスト名にはポート番号を含められます。ポート番号はコロン（:）で区切ります。ホストはリストに表示されている順序で試行され、最初に接続が確立されたホストで停止します。
portnum (IN)	接続先の TCP ポート番号を含みます。ホストがポート番号を含む場合、このパラメータは無視されます。このパラメータが指定されておらず、ホスト名にポート番号が含まれていない場合、デフォルトのポート番号は 389 とみなされます。

戻り値

表 21-5 init ファンクションの戻り値

戻り値	説明
SESSION	API へのその後のコールに使用できる LDAP セッションへのハンドル。

例外

表 21-6 init ファンクションの例外

例外	説明
init_failed	LDAP サーバーへの接続で問題が発生した場合に呼び出されます。
ts_mode_not_supported	MTS サービスを使用してデータベースにログインしたユーザー・セッションから DBMS_LDAP.init が起動された場合に呼び出されます。
general_error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

DBMS_LDAP.init は、LDAP サーバーへのセッションを確立するために最初にコールする必要があるファンクションです。DBMS_LDAP.init はセッション・ハンドルを戻します。セッション・ハンドルとは、セッションに関連する後続のコールに渡す必要のある、不透明なデータ構造へのポインタです。このルーチンは、セッションを初期化できない場合、NULL を戻し、INIT_FAILED の例外を呼び出します。init へのコールに引き続き、DBMS_LDAP.bind_s または DBMS_LDAP.simple_bind_s を使用して接続が認証される必要があります。

関連項目：

- 21-8 ページ「simple_bind_s ファンクション」
- 21-9 ページ「bind_s ファンクション」

simple_bind_s ファンクション

このファンクションは、ディレクトリ・サーバーに対し、ユーザー名およびパスワードに基づいた単純な認証を実行するために使用します。

構文

```
DBMS_LDAP.simple_bind_s (
   ld      IN SESSION,
   dn      IN VARCHAR2,
   passwd  IN VARCHAR2)
RETURN PLS_INTEGER;
```

パラメータ

表 21-7 simple_bind_s ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
dn (IN)	ログインを試行するユーザーの識別名。
passwd (IN)	パスワードを含むテキスト文字列。

戻り値

表 21-8 simple_bind_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	正常に完了した場合の DBMS_LDAP SUCCESS。問題が発生した場合、表 21-9 に示す例外のいずれかが呼び出されます。

例外

表 21-9 simple_bind_s ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
mts_mode_not_supported	MTS サービスとしてログインしたユーザー・セッションから DBMS_LDAP.init が起動された場合に呼び出されます。
general_error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

DBMS_LDAP.simple_bind_s を使用して、ディレクトリの識別名およびディレクトリのパスワードが既知であるユーザーを認証できます。ただし、コールできるのは、DBMS_LDAP.init へのコールから有効な LDAP セッション・ハンドルが獲得された後のみです。

bind_s ファンクション

このファンクションは、ディレクトリ・サーバーに対し、複雑な認証を実行します。

構文

```
DBMS_LDAP.bind_s (
   ld    IN SESSION,
   dn    IN VARCHAR2,
   cred  IN VARCHAR2,
   meth  IN PLS_INTEGER )
RETURN PLS_INTEGER;
```

パラメータ

表 21-10 bind_s ファンクションのパラメータ

パラメータ	説明
ld	有効な LDAP セッション・ハンドル。
dn	ログインを試行するユーザーの識別名。
cred	認証に使用される資格証明を含むテキスト文字列。
meth	認証方式。

戻り値

表 21-11 bind_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	正常に完了した場合の DBMS_LDAP.SUCCESS。問題が発生した場合、表 21-12 に示す例外のいずれかが呼び出されます。

例外

表 21-12 bind_s ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_auth_method	要求された認証方式がサポートされていない場合に呼び出されます。
mts_mode_not_supported	MTS サービスにログインしたユーザー・セッションから起動された場合に呼び出されます。
general_error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

DBMS_LDAP.bind_s を使用してユーザーを認証できます。ただし、コールできるのは、DBMS_LDAP.init へのコールから有効な LDAP セッション・ハンドルが獲得された後のみです。

関連項目：

- 21-6 ページ「init ファンクション」
- 21-8 ページ「simple_bind_s ファンクション」

unbind_s ファンクション

このファンクションは、アクティブな LDAP セッションをクローズします。

構文

```
DBMS_LDAP.unbind_s (
   ld  IN SESSION )
RETURN PLS_INTEGER;
```

パラメータ

表 21-13 unbind_s ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。

戻り値

表 21-14 unbind_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	正常終了した場合の DBMS_LDAP.SUCCESS。問題が発生した場合、表 21-15 に示す例外のいずれかが呼び出されます。

例外

表 21-15 unbind_s ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
general error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

unbind_s ファンクションは、バインドされていない要求をサーバーに送信し、LDAP セッションに関連するオープン接続をすべてクローズします。さらにセッション・ハンドルに割り当てられたリソースを解放し、戻ります。このファンクションへコールした後、セッション・ハンドル ld は無効となり、それ以降に ld を使用して LDAP API をコールすることは不正となります。

関連項目：

- 21-8 ページ「simple_bind_s ファンクション」
- 21-9 ページ「bind_s ファンクション」

compare_s ファンクション

このファンクションは、特定のエントリにおける特定の属性が特定の値を持っているかどうかをテストします。

構文

```
DBMS_LDAP.compare_s (
   ld     IN SESSION,
   dn     IN VARCHAR2,
   attr   IN VARCHAR2,
   value  IN VARCHAR2)
RETURN PLS_INTEGER;
```

パラメータ

表 21-16 compare_s ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
dn (IN)	比較対象のエントリ名。
attr (IN)	比較対象の属性。
value (IN)	比較対象の文字列属性値。

戻り値

表 21-17 compare_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	COMPARE_TRUE は、一致する値を持つ、指定された属性です。 属性の値が指定された値に一致しない場合は、COMPARE_FALSE です。

例外

表 21-18 compare_s ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
general_error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

compare_s ファンクションは、ディレクトリ・サーバーに格納された任意の属性が一定の値と一致する場合のアサーションに使用できます。この処理を実行できるのは、構文の定義において比較が許可されている属性のみです。compare_s ファンクションをコールできるのは、有効な LDAP セッション・ハンドルが init ファンクションから獲得され、bind_s または simple_bind_s ファンクションを使用して認証された後のみです。

関連項目： 21-9 ページ「bind_s ファンクション」

search_s ファンクション

このファンクションは、LDAP サーバーにおいて同期検索を実行します。すべての検索結果がサーバーにより送信された後または検索要求がサーバー側でタイムアウトした場合にのみ、PL/SQL 環境に制御が戻されます。

構文

```
FUNCTION search_s  (
   ld         IN SESSION,
   base       IN VARCHAR2,
   scope      IN PLS_INTEGER,
   filter     IN VARCHAR2,
   attrs      IN STRING_COLLECTION,
   attronly   IN PLS_INTEGER,
   res        OUT MESSAGE)
RETURN PLS_INTEGER;
```

パラメータ

表 21-19 search_s ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
base (IN)	検索を開始するエントリの dn。
scope (IN)	SCOPE_BASE (0 × 00)、SCOPE_ONELEVEL (0 × 01) または SCOPE_SUBTREE (0 × 02) のいずれか。検索範囲を示します。
filter (IN)	検索フィルタを表す文字列。値を NULL に指定すると、すべてのエントリに一致するフィルタ（objectclass=*）が使用されます。
attrs (IN)	一致する各エントリに対して戻す属性を指定する文字列のコレクション。このパラメータに NULL を渡した場合、使用可能なすべてのユーザー属性が取り出されます。特殊な定数文字列 NO_ATTRS (1.1) を配列における唯一の文字列として使用することで、サーバーから属性タイプが戻らないことを示します。特殊な定数文字列 ALL_USER_ATTRS（*） を attrs 配列において一定の操作属性の名前とともに使用し、すべてのユーザー属性および表示された操作属性が戻されることを示します。
attrsonly (IN)	ブール値。属性のタイプおよび値が両方戻される場合には 0（ゼロ）、タイプのみが必要である場合には 0（ゼロ）以外の値を指定する必要があります。
res (OUT)	コールの完了時における検索結果を含む結果パラメータ。結果が戻されない場合、*res は NULL に設定されます。

戻り値

表 21-20 search_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	検索処理が成功した場合は、DBMS_LDAP.SUCCESS。その他の場合はすべて、例外が呼び出されます。
res （OUT パラメータ）	検索が成功してエントリが存在する場合、このパラメータは、結果セットを繰り返し使用できる非 NULL 値に設定されます。

例外

表 21-21 search_s ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_search_scope	検索範囲が SCOPE_BASE、SCOPE_ONELEVEL または SCOPE_SUBTREE のいずれにも該当しない場合に呼び出されます。
general_error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

このファンクションは検索処理を発行します。すべての結果がサーバーから返されるまでユーザー環境に制御が戻されません。検索から戻されたエントリは、res パラメータに含まれます。このパラメータは、コール元には不透明です。次の項に示す解析ルーチンをコールすることにより、エントリ、属性、値などを抽出できます。

関連項目：

- 21-16 ページ「search_st ファンクション」
- 21-18 ページ「first_entry ファンクション」
- 21-19 ページ「next_entry ファンクション」

search_st ファンクション

このファンクションは、クライアント側をタイムアウトにして、LDAP サーバーにおいて同期検索を実行します。すべての検索結果がサーバーにより送信された後または検索要求がクライアントまたはサーバー側でタイムアウトした場合にのみ、PL/SQL 環境に制御が戻されます。

構文

```
DBMS_LDAP.search_st (
   ld         IN  SESSION,
   base       IN  VARCHAR2,
   scope      IN  PLS_INTEGER,
   filter     IN  VARCHAR2,
   attrs      IN  STRING_COLLECTION,
   attronly   IN  PLS_INTEGER,
   tv         IN  TIMEVAL,
   res        OUT MESSAGE)
RETURN PLS_INTEGER;
```

パラメータ

表 21-22 search_st ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
base (IN)	検索を開始するエントリの dn。
scope (IN)	SCOPE_BASE (0 × 00)、SCOPE_ONELEVEL (0 × 01) または SCOPE_SUBTREE (0 × 02) のいずれか。検索範囲を示します。
filter (IN)	検索フィルタを表す文字列。値を NULL に指定すると、すべてのエントリに一致するフィルタ（objectclass=*）が使用されます。
attrs (IN)	一致する各エントリに対して戻す属性を指定する文字列のコレクション。このパラメータに NULL を渡した場合、使用可能なすべてのユーザー属性が取り出されます。特殊な定数文字列 NO_ATTRS (1.1) を配列における唯一の文字列として使用することで、サーバーから属性タイプが戻らないことを示します。特殊な定数文字列 ALL_USER_ATTRS (*) を attrs 配列において一定の操作属性の名前とともに使用し、すべてのユーザー属性および表示された操作属性が戻されることを示します。
attrsonly (IN)	ブール値。属性のタイプおよび値が両方戻される場合には 0（ゼロ）、タイプのみが必要である場合には 0（ゼロ）以外の値を指定する必要があります。

表 21-22 search_st ファンクションのパラメータ（続き）

パラメータ	説明
tv （IN）	この検索に使用する必要があるタイムアウト値。秒およびミリ秒で表されます。
res （OUT）	コールの完了時における検索結果を含む結果パラメータ。結果が戻されない場合、*res は NULL に設定されます。

戻り値

表 21-23 search_st ファンクションの戻り値

戻り値	説明
PLS_INTEGER	検索処理が成功した場合は、DBMS_LDAP.SUCCESS。その他の場合はすべて、例外が呼び出されます。
res （OUT パラメータ）	検索が成功してエントリが存在する場合、このパラメータは、結果セットを繰り返し使用できる非 NULL 値に設定されます。

例外

表 21-24 search_st ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_search_scope	検索範囲が SCOPE_BASE、SCOPE_ONELEVEL または SCOPE_SUBTREE のいずれにも該当しない場合に呼び出されます。
invalid_search_time_value	タイムアウトに指定された時間の値が無効である場合に呼び出されます。
general_error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

このファンクションは、タイムアウト値を必要とする点を除き、DBMS_LDAP.search_s に非常に類似しています。

関連項目：

- 21-14 ページ「search_s ファンクション」
- 21-18 ページ「first_entry ファンクション」
- 21-19 ページ「next_entry ファンクション」

first_entry ファンクション

このファンクションは、search_s または search_st のいずれかにより戻された結果セットの最初のエントリを取り出します。

構文

```
DBMS_LDAP.first_entry (
   ld   IN SESSION,
   msg  IN MESSAGE )
RETURN MESSAGE;
```

パラメータ

表 21-25 first_entry ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
msg (IN)	同期検索ルーチンのいずれかへのコールにより獲得された検索結果。

戻り値

表 21-26 first_entry の戻り値

戻り値	説明
MESSAGE	LDAP サーバーから戻されたエントリ・リストにおける最初のエントリへのハンドル。エラーが発生し、例外が呼び出された場合には、NULL に設定されます。

例外

表 21-27 first_entry の例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_message	渡されたメッセージ・ハンドルが無効である場合に呼び出されます。

使用上の注意

first_entry ファンクションは、常に、検索処理から結果を取り出す場合に使用する最初のファンクションに指定する必要があります。

関連項目：

- 21-19 ページ「next_entry ファンクション」
- 21-14 ページ「search_s ファンクション」
- 21-16 ページ「search_st ファンクション」

next_entry ファンクション

このファンクションは、検索処理において、結果セットの次のエントリを繰り返します。

構文

```
DBMS_LDAP.next_entry (
   ld   IN SESSION,
   msg  IN MESSAGE )
RETURN MESSAGE;
```

パラメータ

表 21-28 next_entry ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
msg (IN)	同期検索ルーチンのいずれかへのコールにより獲得された検索結果。

戻り値

表 21-29 next_entry ファンクションの戻り値

戻り値	説明
MESSAGE	LDAP サーバーから戻されたエントリ・リストにおける次のエントリへのハンドル。エラーが発生し、例外が呼び出された場合には、NULL に設定されます。

例外

表 21-30 next_entry ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_message	渡されたメッセージ・ハンドルが無効である場合に呼び出されます。

使用上の注意

next_entry ファンクションは、常に、first_entry へのコールの後にコールされる必要があります。また、next_entry への正常なコールの戻り値を msg 引数として使用する必要があります。これは、後続の next_entry へのコールにおいてリストの次のエントリをフェッチするために使用されます。

関連項目：

- 21-14 ページ「search_s ファンクション」
- 21-16 ページ「search_st ファンクション」
- 21-18 ページ「first_entry ファンクション」

count_entries ファンクション

このファンクションは、結果セットのエントリ数をカウントします。first_entry および next_entry ファンクションを組み合せて使用して結果セットを走査する間、残存するエントリ数をカウントすることも可能です。

構文

```
DBMS_LDAP.count_entries (
   ld   IN SESSION,
   msg  IN MESSAGE )
RETURN PLS_INTEGER;
```

パラメータ

表 21-31 count_entry ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
msg (IN)	同期検索ルーチンのいずれかへのコールにより獲得された検索結果。

戻り値

表 21-32 count_entry ファンクションの戻り値

戻り値	説明
PLS INTEGER	結果セットにエントリがある場合は、0（ゼロ）以外の数値。 問題がある場合には、-1 です。

例外

表 21-33 count_entry ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_message	渡されたメッセージ・ハンドルが無効である場合に呼び出されます。
count_entry_error	エントリのカウントにおいて問題がある場合に呼び出されます。

使用上の注意

count_entries ファンクションは、連鎖するエントリに含まれるエントリの数を戻します。res パラメータが無効であるなどのエラーが発生した場合、-1 が戻されます。また、count_entries コールが first_message、next_message、first_entry、next_entry、first_reference および next_reference により戻されたメッセージ、エントリまたはリファレンスを使用してコールされた場合、連鎖に残存するエントリの数をカウントできます。

関連項目：

- 21-18 ページ「first_entry ファンクション」
- 21-19 ページ「next_entry ファンクション」

first_attribute ファンクション

このファンクションは、結果セットにおいて指定されたエントリの最初の属性をフェッチします。

構文

```
DBMS_LDAP.first_attribute (
   ld         IN  SESSION,
   msg        IN  MESSAGE,
   ber_elem   OUT BER_ELEMENT)
RETURN VARCHAR2;
```

パラメータ

表 21-34 first_attribute ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
msg (IN)	属性が参照されるエントリ。first_entry または next_entry により戻されます。
ber_elem (OUT)	BER_ELEMENT へのハンドル。エントリにおいて読み取られた属性を追跡するために使用されます。

戻り値

表 21-35 first_attribute ファンクションの戻り値

戻り値	説明
VARCHAR2	属性の名前（存在する場合）。 属性が存在しない場合またはエラーが発生した場合は、NULL です。
ber_elem	すべての属性を繰り返し行うために DBMS_LDAP.next_attribute により使用されるハンドル。

例外

表 21-36 first_attribute ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_message	渡されたメッセージ・ハンドルが無効である場合に呼び出されます。

使用上の注意

first_attribute へのファンクション・パラメータとして戻された BER_ELEMENT へのハンドルは、エントリの多様な属性を繰り返しするために next_attribute への次のコールで使用する必要があります。first_attribute へのコールから戻された属性の名前は、get_values ファンクションまたは get_values_len ファンクションへのコールで使用し、特定の属性の値を取得することも可能です。

関連項目：

- 21-18 ページ「first_entry ファンクション」
- 21-19 ページ「next_entry ファンクション」
- 21-24 ページ「next_attribute ファンクション」
- 21-26 ページ「get_values ファンクション」
- 21-28 ページ「get_values_len ファンクション」

next_attribute ファンクション

このファンクションは、結果セットにおいて指定されたエントリの次の属性をフェッチします。

構文

```
DBMS_LDAP.next_attribute (
   ld        IN SESSION,
   msg       IN MESSAGE,
   ber_elem  IN BER_ELEMENT)
RETURN VARCHAR2;
```

パラメータ

表 21-37 next_attribute ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
msg (IN)	属性が参照されるエントリ。first_entry または next_entry により戻されます。
ber_elem (IN)	BER_ELEMENT へのハンドル。エントリにおいて読み取られた属性を追跡するために使用されます。

戻り値

表 21-38 next_attribute ファンクションの戻り値

戻り値	説明
VARCHAR2	属性の名前（存在する場合)。

例外

表 21-39 next_attribute ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_message	渡されたメッセージ・ハンドルが無効である場合に呼び出されます。

使用上の注意

first_attribute へのファンクション・パラメータとして戻された BER_ELEMENT へのハンドルは、エントリの多様な属性を繰り返しするために next_attribute への次のコールで使用する必要があります。next_attribute へのコールから戻された属性の名前は、get_values または get_values_len へのコールで使用し、特定の属性の値を取得することも可能です。

関連項目：

- 21-18 ページ「first_entry ファンクション」
- 21-19 ページ「next_entry ファンクション」
- 21-22 ページ「first_attribute ファンクション」
- 21-26 ページ「get_values ファンクション」
- 21-28 ページ「get_values_len ファンクション」

get_dn ファンクション

このファンクションは、結果セットにおいて指定されたエントリの X.500 識別名を取り出します。

first_attribute ファンクションは、結果セットにおいて指定されたエントリの最初の属性をフェッチします。

構文

```
DBMS_LDAP.get_dn (
   ld   IN SESSION,
   msg  IN MESSAGE)
RETURN VARCHAR2;
```

パラメータ

表 21-40 get_dn ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
msg (IN)	DN が戻されるエントリ。

戻り値

表 21-41 get_dn ファンクションの戻り値

戻り値	説明
VARCHAR2	PL/SQL 文字列であるエントリの X.500 識別名。 問題がある場合は、NULL です。

例外

表 21-42 get_dn ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_message	渡されたメッセージ・ハンドルが無効である場合に呼び出されます。
get_dn_error	DN の判断において問題がある場合に呼び出されます。

使用上の注意

get_dn ファンクションは、プログラム・ロジックが結果セットを繰り返しする際にエントリの DN を取り出すために使用できます。これは、explode_dn への入力として使用され、DN の各コンポーネントを取り出します。

関連項目： 21-44 ページ「explode_dn ファンクション」

get_values ファンクション

このファンクションは、任意のエントリにおける任意の属性に関連付けられたすべての値を取り出します。

構文

```
DBMS_LDAP.get_values (
   ld       IN SESSION,
   ldapentry  IN MESSAGE,
   attr       IN VARCHAR2)
RETURN STRING_COLLECTION;
```

パラメータ

表 21-43 get_values ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
ldapentry (IN)	検索結果から戻されたエントリへの有効なハンドル。
attr (IN)	値が検索対象となっている属性の名前。

戻り値

表 21-44 get_values ファンクションの戻り値

戻り値	説明
STRING_COLLECTION	指定された属性の値すべてを含む PL/SQL 文字列のコレクション。 指定された属性に関連付けられた値がない場合は、NULL です。

例外

表 21-45 get_values ファンクションの例外

例外	説明
invalid session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid message	渡されたエントリ・ハンドルが無効である場合に呼び出されます。

使用上の注意

get_values ファンクションをコールできるのは、first_entry または next_entry のいずれかに対するコールにより、エントリへのハンドルが最初に取り出された後です。属性の名前は事前に知ることができ、first_attribute または next_attribute へのコールによる判別も可能です。get_values ファンクションは、常に、取り出す属性のデータ・タイプが文字列であるとみなします。バイナリのデータ・タイプを取り出す場合は、get_values_len を使用してください。

関連項目：

- 21-18 ページ「first_entry ファンクション」
- 21-19 ページ「next_entry ファンクション」
- 21-28 ページ「get_values_len ファンクション」
- 21-41 ページ「count_values ファンクション」

get_values_len ファンクション

このファンクションは、バイナリ構文を持つ属性の値を取り出します。

構文

```
DBMS_LDAP.get_values_len (
   ld         IN SESSION,
   ldapentry  IN MESSAGE,
   attr       IN VARCHAR2)
RETURN BINVAL_COLLECTION;
```

パラメータ

表 21-46 get_values_len ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
ldapentrymsg (IN)	検索結果から戻されたエントリへの有効なハンドル。
attr (IN)	値が検索対象となっている文字列の名前。

戻り値

表 21-47 get_values_len ファンクションの戻り値

戻り値	説明
BINVAL_COLLECTION	指定された属性の値すべてを含む PL/SQL RAW コレクション。 指定された属性に関連付けられた値がない場合は、NULL です。

例外

表 21-48 get_values_len ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_message	渡されたエントリ・ハンドルが無効である場合に呼び出されます。

使用上の注意

get_values_len ファンクションをコールできるのは、first_entry または next_entry のいずれかに対するコールにより、エントリへのハンドルが最初に取り出された後です。属性の名前は事前に知ることができ、first_attribute または next_attribute へのコールによる判別も可能です。このファンクションを使用して、バイナリおよびバイナリ以外の属性値を両方取り出すことも可能です。

関連項目：

- 21-18 ページ「first_entry ファンクション」
- 21-19 ページ「next_entry ファンクション」
- 21-26 ページ「get_values ファンクション」
- 21-42 ページ「count_values_len ファンクション」

delete_s ファンクション

このファンクションは、LDAP ディレクトリ情報ツリーのリーフ・エントリを削除します。

構文

```
DBMS_LDAP.delete_s (
   ld       IN SESSION,
   entrydn  IN VARCHAR2)
RETURN PLS_INTEGER;
```

パラメータ

表 21-49 delete_s ファンクションのパラメータ

パラメータ名	説明
ld (IN)	有効な LDAP セッション。
entrydn (IN)	削除するエントリの X.500 識別名。

戻り値

表 21-50 delete_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	削除処理が成功した場合は、DBMS_LDAP.SUCCESS。それ以外の場合は、例外が呼び出されます。

例外

表 21-51 delete_s ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_entry_dn	エントリの識別名が無効である場合に呼び出されます。
general_error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

delete_s ファンクションを使用して、LDAP DIT のリーフ・レベル・エントリのみを削除できます。リーフ・レベル・エントリとは、子または LDAP エントリを下位に持たないエントリです。リーフ・エントリ以外のエントリの削除には使用できません。

関連項目： 21-30 ページ「modrdn2_s ファンクション」

modrdn2_s ファンクション

modrdn2_s ファンクションを使用して、エントリの相対識別名を改名できます。

構文

```
DBMS_LDAP.modrdn2_s (
   ld            IN SESSION,
   entrydn       IN VARCHAR2
   newrdn        IN VARCHAR2
   deleteoldrdn  IN PLS_INTEGER)
RETURN PLS_INTEGER;
```

パラメータ

表 21-52 modrdn2_s ファンクションのパラメータ

パラメータ	説明
ld (IN)	有効な LDAP セッション・ハンドル。
entrydn (IN)	エントリの識別名（このエントリは、DIT においてリーフ・ノードである必要があります)。
newrdn (IN)	エントリの新しい相対識別名。
deleteoldrdn (IN)	ブール値。値が 0（ゼロ）以外の場合は、元の名前の属性値をエントリから削除する必要があることを示しています。

戻り値

表 21-53 modrdn2_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	処理が成功した場合は、DBMS_LDAP.SUCCESS。それ以外の場合は、例外が呼び出されます。

例外

表 21-54 modrdn2_s ファンクションの例外

例外	説明
invalid_session	セッション・ハンドル ld が無効である場合に呼び出されます。
invalid_entry_dn	エントリの識別名が無効である場合に呼び出されます。
invalid_rdn	無効な LDAP RDN。
invalid_deleteoldrdn	無効な LDAP deleteoldrdn。
general error	その他すべてのエラー。例外に関連付けられたエラーの文字列では、エラーの内容が詳細に記述されます。

使用上の注意

このファンクションを使用して、DIT のリーフ・ノードを改名できます。その場合、単に既知の相対識別名を変更します。このファンクションは、LDAP バージョン 3 の標準では使用できません。同じ機能を持つ rename_s を使用してください。

関連項目： 21-43 ページ「rename_s ファンクション」

err2string ファンクション

このファンクションは、LDAP エラー・コードを API が処理を行うローカル言語の文字列に変換します。

構文

```
DBMS_LDAP.err2string (
   ldap_err  IN PLS_INTEGER )
RETURN VARCHAR2;
```

パラメータ

表 21-55 err2string ファンクションのパラメータ

パラメータ	説明
ldap_err (IN)	いずれかの API コールから戻されるエラー番号。

戻り値

表 21-56 err2string ファンクションの戻り値

戻り値	説明
VARCHAR2	エラーを詳細に説明するローカル言語に変換された文字列。

例外

表 21-57 err2string ファンクションの例外

例外	説明
N/A	なし。

使用上の注意

このリリースでは、API コールのいずれかでエラーが発生した場合、例外処理メカニズムがこのファンクションを自動的に起動します。

create_mod_array ファンクション

このファンクションは、modify_s ファンクションまたは add_s ファンクションを使用するエントリに適用される変更配列エントリにメモリーを割り当てます。

構文

```
DBMS_LDAP.create_mod_array (
   num  IN PLS_INTEGER)
RETURN MOD_ARRAY;
```

パラメータ

表 21-58 create_mod_array ファンクションのパラメータ

パラメータ	説明
num (IN)	追加または変更する属性の番号。

戻り値

表 21-59 create_mod_array ファンクションの戻り値

戻り値	説明
MOD_ARRAY	LDAP 変更配列へのポインタを保持するデータ構造。 問題がある場合は、NULL です。

例外

表 21-60 create_mod_array ファンクションの例外

例外	説明
N/A	LDAP 特有の例外は呼び出されません。

使用上の注意

このファンクションは、DBMS_LDAP.add_s および DBMS_LDAP.modify_s の準備ステップの 1 つです。add_s または modify_s へのコールが完了した後、DBMS_LDAP.free_mod_array をコールしてメモリーを解放する必要があります。

関連項目：

- 21-34 ページ「populate_mod_array（文字列バージョン）プロシージャ」
- 21-37 ページ「modify_s ファンクション」
- 21-38 ページ「add_s ファンクション」
- 21-40 ページ「free_mod_array プロシージャ」

populate_mod_array（文字列バージョン）プロシージャ

このプロシージャは、追加または変更処理を行う属性情報を 1 セット移入します。

構文

```
DBMS_LDAP.populate_mod_array (
   modptr    IN DBMS_LDAP.MOD_ARRAY,
   mod_op    IN PLS_INTEGER,
   mod_type  IN VARCHAR2,
   modval    IN DBMS_LDAP.STRING_COLLECTION);
```

パラメータ

表 21-61 populate_mod_array（文字列バージョン）プロシージャのパラメータ

パラメータ	説明
modptr （IN）	LDAP 変更配列へのポインタを保持するデータ構造。
Mod_op （IN）	このフィールドは、実行する変更のタイプを指定します。
Mod_type （IN）	このフィールドは、変更を適用する属性のタイプの名前を示します。
Modval （IN）	このフィールドは、追加、削除または置換する属性の値を指定します。文字列の値でのみ使用できます。

戻り値

表 21-62 populate_mod_array（文字列バージョン）プロシージャの戻り値

戻り値	説明
N/A	—

例外

表 21-63 populate_mod_array（文字列バージョン）プロシージャの例外

例外	説明
invalid_mod_array	無効な LDAP 変更配列。
invalid_mod_option	無効な LDAP 変更オプション。
invalid_mod_type	無効な LDAP 変更タイプ。
invalid_mod_value	無効な LDAP 変更値。

使用上の注意

このファンクションは、DBMS_LDAP.add_s および DBMS_LDAP.modify_s の準備ステップの 1 つです。DBMS_LDAP.create_mod_array がコールされた後で発生させる必要があります。

関連項目：

- 21-33 ページ「create_mod_array ファンクション」
- 21-37 ページ「modify_s ファンクション」
- 21-38 ページ「add_s ファンクション」
- 21-40 ページ「free_mod_array プロシージャ」

populate_mod_array（バイナリ・バージョン）プロシージャ

このプロシージャは、追加または変更処理を行う属性情報を 1 セット移入します。このプロシージャ・コールは、DBMS_LDAP.create_mod_array がコールされた後で発生させる必要があります。

構文

```
PROCEDURE populate_mod_array
    (modptr   IN DBMS_LDAP.MOD_ARRAY,
     mod_op   IN PLS_INTEGER,
     mod_type IN VARCHAR2,
     modval   IN DBMS_LDAP.BERVAL_COLLECTION);
```

パラメータ

表 21-64 populate_mod_array（バイナリ・バージョン）プロシージャのパラメータ

パラメータ	説明
modptr (IN)	LDAP 変更配列へのポインタを保持するデータ構造。
Mod_op (IN)	このフィールドは、実行する変更のタイプを指定します。
Mod_typ (IN)	このフィールドは、変更を適用する属性のタイプの名前を示します。
Modval (IN)	このフィールドは、追加、削除または置換する属性の値を指定します。バイナリの値でのみ使用できます。

戻り値

表 21-65 populate_mod_array（バイナリ・バージョン）プロシージャの戻り値

戻り値	説明
N/A	—

例外

表 21-66 populate_mod_array（バイナリ・バージョン）プロシージャの例外

例外	説明
invalid_mod_array	無効な LDAP 変更配列。
invalid_mod_option	無効な LDAP 変更オプション。
invalid_mod_type	無効な LDAP 変更タイプ。
invalid_mod_value	無効な LDAP 変更値。

使用上の注意

このファンクションは、DBMS_LDAP.add_s および DBMS_LDAP.modify_s の準備段階の 1 つです。DBMS_LDAP.create_mod_array がコールされた後で発生させる必要があります。

関連項目：

- 21-33 ページ「create_mod_array ファンクション」
- 21-37 ページ「modify_s ファンクション」
- 21-38 ページ「add_s ファンクション」
- 21-40 ページ「free_mod_array プロシージャ」

modify_s ファンクション

このファンクションは、既存の LDAP ディレクトリ・エントリの同期変更を実行します。

構文

```
DBMS_LDAP.modify_s (
   ld       IN DBMS_LDAP.SESSION,
   entrydn  IN VARCHAR2,
   modptr   IN DBMS_LDAP.MOD_ARRAY)
RETURN PLS_INTEGER;
```

パラメータ

表 21-67 modify_s ファンクションのパラメータ

パラメータ	説明
ld (IN)	LDAP セッションへのハンドル。DBMS_LDAP.init へのコールが成功した場合に戻されます。
entrydn (IN)	内容を変更するディレクトリ・エントリの名前を指定します。
modptr (IN)	LDAP 変更データ構造へのハンドル。DBMS_LDAP.create_mod_array へのコールが成功した場合に戻されます。

戻り値

表 21-68 modify_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	変更処理が成功したかどうかを示します。

例外

表 21-69 modify_s ファンクションの例外

例外	説明
invalid_session	無効な LDAP セッション。
invalid_entry_dn	無効な LDAP エントリ dn。
invalid_mod_array	無効な LDAP 変更配列。

使用上の注意

このファンクションは、DBMS_LDAP.create_mod_array および DBMS_LDAP.populate_mod_array のコールが成功した後に使用する必要があります。

関連項目：

- 21-33 ページ「create_mod_array ファンクション」
- 21-34 ページ「populate_mod_array（文字列バージョン）プロシージャ」
- 21-38 ページ「add_s ファンクション」
- 21-40 ページ「free_mod_array プロシージャ」

add_s ファンクション

このファンクションは、LDAP ディレクトリに対し、新規エントリを同期的に追加します。add_s をコールする前に、DBMS_LDAP.create_mod_array および DBMS_LDAP.populate_mod_array をコールする必要があります。

構文

```
DBMS_LDAP.add_s (
   ld       IN DBMS_LDAP.SESSION,
   entrydn  IN VARCHAR2,
   modptr   IN DBMS_LDAP.MOD_ARRAY)
RETURN PLS_INTEGER;
```

パラメータ

表 21-70 add_s ファンクションのパラメータ

パラメータ	説明
ld (IN)	LDAP セッションへのハンドル。DBMS_LDAP.init へのコールが成功した場合に戻されます。
Entrydn (IN)	作成するディレクトリ・エントリの名前を指定します。
Modptr (IN)	LDAP 変更データ構造へのハンドル。DBMS_LDAP.create_mod_array へのコールが成功した場合に戻されます。

戻り値

表 21-71 add_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	変更処理が成功したかどうかを示します。

例外

表 21-72 add_s ファンクションの例外

例外	説明
invalid_session	無効な LDAP セッション。
invalid_entry_dn	無効な LDAP エントリ dn。
invalid_mod_array	無効な LDAP 変更配列。

使用上の注意

追加するエントリの親エントリがすでにディレクトリ内に存在している必要があります。このファンクションは、DBMS_LDAP.create_mod_array および DBMS_LDAP.populate_mod_array のコールが成功した後に使用する必要があります。

関連項目：

- 21-33 ページ「create_mod_array ファンクション」
- 21-34 ページ「populate_mod_array（文字列バージョン）プロシージャ」
- 21-37 ページ「modify_s ファンクション」
- 21-40 ページ「free_mod_array プロシージャ」

free_mod_array プロシージャ

このプロシージャは、DBMS_LDAP.create_mod_array により割り当てられたメモリーを解放します。

構文

```
DBMS_LDAP.free_mod_array (
   modptr  IN DBMS_LDAP.MOD_ARRAY);
```

パラメータ

表 21-73 free_mod_array プロシージャのパラメータ

パラメータ	説明
modptr (in)	LDAP 変更データ構造へのハンドル。DBMS_LDAP.create_mod_array へのコールが成功した場合に戻されます。

戻り値

表 21-74 free_mod_array プロシージャの戻り値

戻り値	説明
N/A	—

例外

表 21-75 free_mod_array プロシージャの例外

例外	説明
N/A	LDAP 特有の例外は呼び出されません。

関連項目：

- 21-33 ページ「create_mod_array ファンクション」
- 21-34 ページ「populate_mod_array（文字列バージョン）プロシージャ」
- 21-37 ページ「modify_s ファンクション」
- 21-38 ページ「add_s ファンクション」

count_values ファンクション

このファンクションは、DBMS_LDAP.get_values により戻された値の数をカウントします。

構文

```
DBMS_LDAP.count_values (
   values  IN DBMS_LDAP.STRING_COLLECTION)
RETURN PLS_INTEGER;
```

パラメータ

表 21-76 count_values ファンクションのパラメータ

パラメータ	説明
values（IN）	文字列の値のコレクション。

戻り値

表 21-77 count_values ファンクションの戻り値

戻り値	説明
PLS_INTEGER	処理が成功したかどうかを示します。

例外

表 21-78 count_values ファンクションの例外

例外	説明
N/A	LDAP 特有の例外は呼び出されません。

関連項目：

- 21-26 ページ「get_values ファンクション」
- 21-42 ページ「count_values_len ファンクション」

count_values_len ファンクション

このファンクションは、DBMS_LDAP.get_values_len により戻された値の数をカウントします。

構文

```
DBMS_LDAP.count_values_len (
   values  IN DBMS_LDAP.BINVAL_COLLECTION)
RETURN PLS_INTEGER;
```

パラメータ

表 21-79 count_values_len ファンクションのパラメータ

パラメータ	説明
values（IN）	バイナリの値のコレクション。

戻り値

表 21-80 count_values_len ファンクションの戻り値

戻り値	説明
PLS_INTEGER	処理が成功したかどうかを示します。

例外

表 21-81 count_values_len ファンクションの例外

例外	説明
N/A	LDAP 特有の例外は呼び出されません。

関連項目：

- 21-28 ページ「get_values_len ファンクション」
- 21-41 ページ「count_values ファンクション」

rename_s ファンクション

このファンクションは、LDAP エントリを同期的に改名します。

構文

```
DBMS_LDAP.rename_s (
   ld            IN SESSION,
   dn            IN VARCHAR2,
   newrdn        IN VARCHAR2,
   newparent     IN VARCHAR2,
   deleteoldrdn  IN PLS_INTEGER,
   serverctrls   IN LDAPCONTROL,
   clientctrls   IN LDAPCONTROL)
RETURN PLS_INTEGER;
```

パラメータ

表 21-82 rename_s ファンクションのパラメータ

パラメータ	説明
ld (IN)	LDAP セッションへのハンドル。DBMS_LDAP.init へのコールが成功した場合に戻されます。
Dn (IN)	改名または移動するディレクトリ・エントリの名前を指定します。
newrdn (IN)	新しい RDN を指定します。
Newparent (IN)	新しい親の DN を指定します。
Deleteoldrdn (IN)	元の RDN を保持する必要があるかどうか指定します。値を 1 に設定すると、元の RDN が削除されます。
Serverctrls (IN)	現在サポートされていません。
Clientctrls (IN)	現在サポートされていません。

戻り値

表 21-83 rename_s ファンクションの戻り値

戻り値	説明
PLS_INTEGER	処理が成功したかどうかを示します。

例外

表 21-84 rename_s ファンクションの例外

例外	説明
invalid_session	無効な LDAP セッション。
invalid_entry_dn	無効な LDAP DN。
invalid_rdn	無効な LDAP RDN。
invalid_newparent	無効な LDAP newparent。
invalid_deleteoldrdn	無効な LDAP deleteoldrdn。

関連項目： 21-30 ページ「modrdn2_s ファンクション」

explode_dn ファンクション

このファンクションは、DN をコンポーネントに分割します。

構文

```
DBMS_LDAP.explode_dn (
   dn       IN VARCHAR2,
   notypes  IN PLS_INTEGER)
RETURN STRING_COLLECTION;
```

パラメータ

表 21-85 explode_dn ファンクションのパラメータ

パラメータ	説明
dn （IN）	分割するディレクトリ・エントリの名前を指定します。
Notypes （IN）	属性タグが戻されるかどうか指定します。値が 0（ゼロ）以外の場合、属性タグは戻されません。

戻り値

表 21-86 explode_dn ファンクションの戻り値

戻り値	説明
STRING_COLLECTION	文字列の配列。DN を分割できない場合は、NULL が戻されます。

例外

表 21-87 explode_dn ファンクションの例外

例外	説明
invalid_entry_dn	無効な LDAP DN。
invalid_notypes	無効な LDAP notypes 値。

関連項目： 21-25 ページ「get_dn ファンクション」

open_ssl ファンクション

このファンクションは、既存の LDAP 接続において、Secure Sockets Layer（SSL）接続を確立します。

構文

```
DBMS_LDAP.open_ssl (
   ld               IN SESSION,
   sslwrl           IN VARCHAR2,
   sslwalletpasswd  IN VARCHAR2,
   sslauth          IN PLS_INTEGER)
RETURN PLS_INTEGER;
```

パラメータ

表 21-88 open_ssl ファンクションのパラメータ

パラメータ	説明
ld (IN)	LDAP セッションへのハンドル。DBMS_LDAP.init へのコールが成功した場合に戻されます。
Sslwrl (IN)	Wallet の場所を指定します（単方向または双方向の SSL 接続に必要です）。
sslwalletpasswd (IN)	Wallet のパスワードを指定します（単方向または双方向の SSL 接続に必要です）。
sslauth (IN)	SSL の認証モードを指定します。認証不要の場合は 1、単方向の認証が必要な場合は 2、双方向の認証が必要な場合は 3 です。

戻り値

表 21-89 open_ssl ファンクションの戻り値

戻り値	説明
PLS_INTEGER	処理が成功したかどうかを示します。

例外

表 21-90 open_ssl ファンクションの例外

例外	説明
invalid_session	無効な LDAP セッション。
invalid_ssl_wallet_loc	無効な LDAP SSL Wallet の場所。
invalid_ssl_wallet_passwd	無効な LDAP SSL Wallet のパスワード。
invalid_ssl_auth_mode	無効な LDAP SSL 認証モード。

使用上の注意

有効な LDAP セッションを取得するには、最初に DBMS_LDAP.init をコールしてください。

関連項目： 21-6 ページ「init ファンクション」

22

DBMS_LIBCACHE

DBMS_LIBCACHE は、リモート・インスタンスから SQL および PL/SQL を抽出し、この SQL を実行せずにローカルでコンパイルすることにより、Oracle インスタンスにライブラリ・キャッシュを準備します。インスタンスのキャッシュをコンパイルする値を使用して、フェイルオーバーまたはスイッチオーバーの前に実行するためにアプリケーションが必要とする情報を準備します。

共有カーソルのコンパイルは、オープン、解析およびバインド操作で構成され、これに加えて最初の実行時にはタイプ・チェックおよび実行計画ファンクションが実行されます。これらの手順はすべて、SELECT 文の DBMS_LIBCACHE パッケージにより事前に実行されます。オープンおよび解析ファンクションは、PL/SQL および DML に対して事前に実行されます。PL/SQL の場合、解析フェーズを実行すると、MCODE 以外のすべてのライブラリ・キャッシュ・ヒープのロードに影響を与えます。

この章では、次の項目について説明します。

要件

DBMS_LIBCACHE を実行するには、SQL 文の場合と同じオブジェクトに直接アクセスする必要があります。そのためには、リモート・システム上で元のシステムと同じユーザー ID を利用することをお薦めします。複数のスキーマ・ユーザーが存在する場合は、個々のユーザーに対し DBMS_LIBCACHE をコールしてください。また、DBMS_LIBCACHE は、一般的なユーザー PARSER を使用してコールすることも可能です。ただし、このユーザーのロールに付与されたアクセス権ではオブジェクトを使用する SQL を解析できません。これは、標準の PL/SQL セキュリティの制限です。

DBMS_LIBCACHE サブプログラムの要約

表 22-1 DBMS_LIBCACHE のサブプログラム

サブプログラム	説明
「COMPILE_CURSORS_FROM_REMOTE プロシージャ」22-2 ページ	ソース・インスタンスからバッチで SQL を抽出し、ターゲット・インスタンスで SQL をコンパイルします。

COMPILE_CURSORS_FROM_REMOTE プロシージャ

このプロシージャは、ソース・インスタンスからバッチで SQL を抽出し、ターゲット・インスタンスで SQL をコンパイルします。

構文

```
DBMS_LIBCACHE.COMPILE_CURSORS_FROM_REMOTE('LIBC_LINK', {MY_USER}, 1, 1024000);
```

パラメータ

表 22-2 COMPILE_CURSORS_FROM_REMOTE プロシージャのパラメータ

パラメータ	説明
Database Link Name	SQL 文の抽出に使用されるインスタンスを指すデータベース・リンク。
Source username	抽出された SQL 文の解析ユーザー名。
Execution threshold	実行回数の下限。この値より下では、カーソルはコンパイルでは選択されません。
Sharable memory threshold	ソース・インスタンスのコンテキスト領域により消費される共有メモリーの下限サイズ。この値より下では、カーソルはコンパイルでは選択されません。

使用上の注意

次の点に注意してください。

- Database link name および Source User name は必須パラメータであるため、指定する必要があります。構文では、1MB を超える SQL すべてを解析する 2 つの追加オプション・パラメータが記されています。
- Database link name －接続ではパスワード・ファイルまたは LDAP 認証のいずれかを使用できます。デフォルトのデータベース・リンク libc_link は、カタログ・プログラム catlibc.sql が実行されたときに作成されます。このパラメータは、dbms_libcache$def.ACCESS_METHOD = DB_LINK_METHOD を使用するリリースでは必須であるため、実際のデフォルト値は設定されていません。
- Source user name －このパラメータを使用すると、一致するローカルの解析ユーザー ID でパッケージを実行できます。このパラメータを使用する場合は通常、ローカルで同じユーザー名に接続します。ユーザー名が指定されている場合、有効な値である必要があります。大文字と小文字は区別されません。
- Execution threshold －カーソル値の実行件数は、カーソルが再ロードされると常にリセットされます。このパラメータを使用すると、アプリケーションにおいて任意の実行回数（4 回以上など）の文を抽出し、コンパイルできます。デフォルト値は 1 です。つまり、無効な SQL 文も含め、実行されていない SQL 文は抽出されません。
- Sharable memory threshold －このパラメータを使用すると、指定したサイズ（たとえば 1024000 バイト）を超える共有メモリーを使用している文を抜き出して、コンパイルできます。デフォルト値（1000）を使用すると、無効で実行されていないカーソルをスキップできます。

23

DBMS_LOB

DBMS_LOB パッケージは、BLOB、CLOB、NCLOB、BFILE および一時 LOB を操作するサブプログラムを提供します。DBMS_LOB を使用すると、各 LOB の特定の部分または LOB 全体に対するアクセスおよび操作ができます。

このパッケージは、SYS によって作成される必要があります。このパッケージが提供する操作は、パッケージ所有者 SYS ではなく、現行のコール・ユーザーのもとで実行されます。

DBMS_LOB では、BLOB、CLOB および NCLOB の読取りおよび変更ができます。BFILE に対しては、読取り専用操作を実行できます。LOB 操作の大部分が、このパッケージによって提供されます。

> **関連項目：**『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』

この章では、次の項目について説明します。

DBMS_LOB の LOB ロケータ

すべての DBMS_LOB サブプログラムは、LOB ロケータをベースにして動作します。DBMS_LOB サブプログラムを正常に実行するためには、データベース表領域または外部ファイル・システムにすでに存在している LOB を示す入力ロケータを用意する必要があります。『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』の第 1 章も参照してください。

データベースで LOB を使用するには、最初に SQL データ定義言語（DDL）を使用して、LOB 列が含まれる表を定義する必要があります。

内部 LOB

表に LOB 列を定義した後、表に内部 LOB を移入するには、SQL データ操作言語（DML）を使用して、LOB 列のロケータを初期化するか、または移入します。

外部 LOB

外部 LOB が LOB ロケータで示されるようにするには、次のことを実行する必要があります。

- 有効な既存の物理ディレクトリを示す DIRECTORY オブジェクトが定義済みであること、および Oracle に対する読取り権限を伴った物理ファイル（追加対象の LOB）が存在していることを確認する必要があります。オペレーティング・システムでパス名の大文字と小文字が区別される場合は、必ず正しい形式でディレクトリを指定してください。
- 追加する外部 LOB の DIRECTORY オブジェクトとファイル名を BFILENAME() ファンクションに渡して、外部 LOB の LOB ロケータを作成する必要があります。

これらの作業が終了すると、特定の LOB ロケータを使用して、LOB 列が含まれている行を挿入または更新できます。

LOB の定義および作成が終了すると、SELECT を実行して LOB ロケータをローカルの PL/SQL LOB 変数に割り当て、この変数を DBMS_LOB の入力パラメータとして使用して LOB 値にアクセスできます。

これらの作業を行うための別の方法については、『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』の「外部 LOB（BFILE）へのアクセス」の項を参照してください。

一時 LOB

一時 LOB の場合は、OCI、PL/SQL またはその他のプログラム・インタフェースを使用して作成または操作する必要があります。一時 LOB は、BLOB、CLOB または NCLOB のいずれにもできます。

DBMS_LOB のデータ・タイプ、定数および例外

データ・タイプ

DBMS_LOB サブプログラムに対するパラメータには、次のデータ・タイプが使用されます。

表 23-1 DBMS_LOB のデータ・タイプ

タイプ	説明
BLOB	ソースまたは宛先のバイナリ LOB。
RAW	ソースまたは宛先の RAW バッファ（BLOB とともに使用されます）。
CLOB	ソースまたは宛先の文字 LOB（NCLOB を含みます）。
VARCHAR2	ソースまたは宛先の文字バッファ（CLOB および NCLOB とともに使用されます）。
INTEGER	バッファまたは LOB のサイズ、LOB へのオフセットまたはアクセス量を指定します。
BFILE	データベースの外部に格納されているラージ・バイナリ・オブジェクト。

DBMS_LOB パッケージでは特別なタイプを定義しません。NCLOB は、CLOB の特別なケースで、固定幅および可変幅のマルチバイト各国語キャラクタ・セットに使用されます。CLOB 用の DBMS_LOB サブプログラム仕様部にある句 ANY_CS によって、CLOB または NCLOB ロケータ変数を入力として受け入れることができます。

定数

DBMS_LOB では、次の定数が定義されます。

```
file_readonly CONSTANT BINARY_INTEGER := 0;
lob_readonly  CONSTANT BINARY_INTEGER := 0;
lob_readwrite CONSTANT BINARY_INTEGER := 1;
lobmaxsize    CONSTANT INTEGER        := 4294967295;
call          CONSTANT PLS_INTEGER    := 12;
session       CONSTANT PLS_INTEGER    := 10;
```

Oracle は、最大 4GB（2^{32}）の LOB をサポートします。ただし、パッケージの amount および offset パラメータで設定できるのは、1 ～ 4294967295（2^{32}-1）の範囲の値です。

PL/SQL 3.0 言語では、RAW または VARCHAR2 変数の最大サイズが 32767 バイトに指定されます。

注意： 値 32767 バイトは、以降の項では maxbufsize で表されます。

例外

表 23-2 DBMS_LOB の例外

例外	コード	説明
invalid_argval	21560	引数は NULL 以外の有効な値であることが必要ですが、渡された引数値は NULL、無効または範囲外です。
access_error	22925	LOB に書き込むデータが多すぎます。LOB サイズは最大 4GB です。
noexist_directory	22285	ファイルに指定されているディレクトリが存在しません。
nopriv_directory	22286	ユーザーに、そのディレクトリ別名またはファイルの操作に必要なアクセス権限がありません。
invalid_directory	22287	現行の操作に使用しているディレクトリ別名は、それが初めてアクセスされた別名か、または前回のアクセス以降に DBA が変更した別名である場合は無効です。
operation_failed	22288	ファイル操作に失敗しました。
unopened_file	22289	要求された操作の実行に使用するファイルがオープンされていません。
open_toomany	22290	オープン・ファイル数が最大値に達しました。

DBMS_LOB に対するセキュリティ

無名 PL/SQL ブロックからコールされた DBMS_LOB サブプログラムは、カレント・ユーザーの権限を使用して実行されます。ストアド・プロシージャからコールされた DBMS_LOB サブプログラムは、そのストアド・プロシージャの所有者の権限を使用して実行されます。

Oracle9*i* では、ユーザーはプロシージャの作成時に、AUTHID を設定して定義者の権限または実行者の権限のどちらを使用するかを示せます。たとえば、次のようにします。

```
CREATE PROCEDURE proc1 authid definer ...
```

または

```
CREATE PROCEDURE proc1 authid current_user ....
```

> **関連項目：** AUTHID および権限の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

DIRECTORY 機能を使用すると、BFILE に安全にアクセスできます。この機能は、『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』および『Oracle9*i* SQL リファレンス』の「BFILENAME 関数」で説明されています。

DBMS_LOB での規則および制限事項

- このパッケージにあるサブプログラムの仕様部に適用される規則は次のとおりです。
  - BLOB および BFILE を操作するサブプログラムの length および offset パラメータは、バイト単位で指定する必要があります。
  - CLOB を操作するサブプログラムの length および offset パラメータは、文字単位で指定する必要があります。
  - offset および amount パラメータは常に、CLOB/NCLOB の場合は文字単位、BLOB/BFILE の場合はバイト単位です。
- パラメータ値の指定時に、次の制限事項に従わなかった場合（または指定しなかった場合）は、INVALID_ARGVAL 例外が発生します。
  1. LOB データの開始位置からの正の絶対オフセットのみ許可されています。LOB の終了位置からの負のオフセットは許可されていません。
  2. amount、offset、newlen、nth など、サイズおよび位置を表すパラメータには、0（ゼロ）以外の正の値のみ許可されています。Oracle SQL 文字列ファンクションおよび演算子で算出された負のオフセットおよび範囲は許可されていません。
  3. offset、amount、newlen、nth の値は、どの DBMS_LOB サブプログラムの場合でも、最大値は lobmaxsize（4GB-1）です。
  4. 固定幅のマルチバイト・キャラクタで構成される CLOB の場合、これらのパラメータの最大値は（lobmaxsize/character_width_in_bytes）文字です。

たとえば、CLOB が次のような 2 バイト文字で構成されているとします。

```
JA16SJISFIXED
```

この場合の amount の最大値は次のとおりです。

```
4294967295/2 = 2147483647 characters.
```

- PL/SQL 言語仕様では、DBMS_LOB サブプログラムで使用される RAW および VARCHAR2 パラメータの上限は 32767 バイト（文字数ではありません）に規定されています。たとえば、変数を次のように宣言するとします。

  ```
  charbuf VARCHAR2(3000)
  ```

  この場合、charbuf にはシングルバイト文字の場合は 3000 文字、2 バイトの固定幅文字の場合は 1500 文字格納できます。これは、CLOB および NCLOB に対して DBMS_LOB サブプログラムを使用するときに特に注意してください。

- %CHARSET 句は、%CHARSET を使用するパラメータの形式が、参照先である ANY_CS パラメータの形式と一致している必要があることを示します。

  たとえば、VARCHAR2 バッファ・パラメータを使用する DBMS_LOB サブプログラムでは、VARCHAR2 バッファの形式は CLOB パラメータの形式と一致している必要があります。入力 LOB パラメータのタイプが NCLOB の場合、バッファには NCHAR データが含まれている必要があります。これに対して、入力 LOB のパラメータのタイプが CLOB の場合、バッファには CHAR データが含まれる必要があります。

  2 つの CLOB パラメータを使用する DBMS_LOB サブプログラムの場合は、2 つの CLOB パラメータの形式が同じであることが必要です。つまり、両方とも NCLOB であるか、または CLOB であることが必要です。

- 更新サブプログラム（APPEND、COPY、TRIM、WRITE および WRITEAPPEND サブプログラム）のコールで、amount および offset を加算した値が、BLOB と BFILE の場合で 4GB（lobmaxsize+1）、CLOB の場合で（lobmaxsize/character_width_in_bytes）+1 を超えると、アクセス例外が発生します。

  この入力条件のもとでは、READ、COMPARE、INSTR および SUBSTR などの読込みサブプログラムでは、End of Lob/File に達するまでデータが読み込まれます。たとえば、BLOB または BFILE での READ 操作で、ユーザーが offset 値に 3GB、amount 値に 2GB を指定すると、READ では（(4GB-1)-3GB）バイトのみが読み込まれます。

- パラメータに NULL または無効な値が入力されると、ファンクションは NULL を戻します。宛先 LOB のパラメータに NULL 値が指定されると、プロシージャでは例外が発生します。

- COMPARE、INSTR および SUBSTR など、パラメータとしてパターンが含まれている操作では、pattern パラメータまたは副文字列に、標準の式または特殊一致文字（例 : SQL の LIKE 演算子の %）はサポートされていません。

- End Of LOB 状態は、READ プロシージャで NO_DATA_FOUND 例外を使用して示されます。この例外が発生するのは、ユーザーが LOB/FILE の終了位置を超えてデータを読み込もうとしたときのみです。最後に読み込んだ READ バッファは 0（ゼロ）バイトです。
- LOB 更新の一貫性を保つために、LOB データを変更するプロシージャ（ミューテータ）をコールする前に、宛先 LOB が含まれている行をロックする必要があります。
- 特に注記がないかぎり、offset パラメータのデフォルト値は 1 です。これは、BLOB または BFILE データの最初のバイト、および CLOB または NCLOB 値の最初の文字を示します。amount パラメータはデフォルト値が指定されていないため、値を明示的に入力する必要があります。
- LOB を変更する任意のサブプログラム（例：APPEND、COPY、ERASE、TRIM または WRITE）をコールする前に、宛先内部 LOB が含まれている行をロックする必要があります。これらのサブプログラムでは、LOB を含んだ行のロックは暗黙的には行われません。

BFILE 特有の規則および制限事項

- サブプログラム COMPARE、INSTR、READ、SUBSTR、FILECLOSE、FILECLOSEALL および LOADFROMFILE は、オープン済みの BFILE ロケータでのみ動作します。つまり、これらのサブプログラムのコールの前に、FILEOPEN コールが正常に完了している必要があります。
- ファンクション FILEEXISTS、FILEGETNAME および GETLENGTH に関しては、ファイルのオープン / クローズ状態はあまり重要ではありません。ただし、ファイルが物理的に必ず存在し、ユーザーに DIRECTORY オブジェクトとそのファイルに関する適切な権限があることが必要です。
- DBMS_LOB は、BFILE 操作に対する並行性制御メカニズムをサポートしません。
- クローズ処理が正しく行われていない複数のオープン・ファイルがセッションで使用されている場合は、FILECLOSEALL サブプログラムを使用してセッションでオープンされたファイルをすべてクローズし、ファイル操作を最初からやり直すことができます。
- DIRECTORY の作成者である場合、またはシステム権限がある場合、SQL の CREATE OR REPLACE、DROP および REVOKE 文の使用には特に注意してください。

  ユーザーまたは特定のディレクトリ・オブジェクトの権限受領者がセッション内に複数のファイルをオープンしている場合は、前述のコマンドが誤ってファイル操作に影響を与える場合があります。この状況で異常終了した場合は、必ず FILECLOSEALL をコールするプログラムまたは無名ブロックを起動し、ファイルを再オープンしてファイル操作を再開してください。

- ユーザー・セッションの間にオープンされたファイルはすべて、セッション終了時に暗黙的にクローズされます。ただし、BFILE の操作の正常終了および異常終了いずれの場合も、操作終了後にファイルをクローズすることをお薦めします。

  プログラムが正常に終了した場合、ファイルを適切にクローズすることによって、セッションで同時にオープンしているファイル数は、必ず SESSION_MAX_OPEN_FILES より少なくなります。

  PL/SQL プログラムが異常終了した場合は、その PL/SQL プログラム内でオープン中のすべてのファイルをクローズする例外ハンドラを使用する必要があります。例外が発生すると、最新の状態の BFILE 変数にアクセスできるのは例外ハンドラのみであるため、この処理が必要です。

  例外によってプログラム制御が PL/SQL プログラム・ブロック外に転送されると、オープン中の BFILE への参照はすべて失われます。この結果、オープン・ファイル・カウントが増加し、SESSION_MAX_OPEN_FILES 値を超える場合があります。

  たとえば、BFILE 値の終了位置を超えて READ 操作を行い、NO_DATA_FOUND 例外が発生したと想定します。

```
DECLARE
       fil BFILE;
       pos INTEGER;
       amt BINARY_INTEGER;
       buf RAW(40);
BEGIN
       SELECT f_lob INTO fil FROM lob_table WHERE key_value = 21;
       dbms_lob.open(fil, dbms_lob.lob_readonly);
       amt := 40; pos := 1 + dbms_lob.getlength(fil); buf := '';
       dbms_lob.read(fil, amt, pos, buf);
       dbms_output.put_line('Read F1 past EOF: '||
           utl_raw.cast_to_varchar2(buf));
       dbms_lob.close(fil);
END;

ORA-01403: データが見つかりません。
ORA-06512:"SYS.DBMS_LOB" 行 373
ORA-06512: 行 10
```

例外が発生した後、BFILE ロケータの変数ファイルは有効範囲外となり、その変数を使用したファイル操作は実行できません。したがって、解決方法として、次のような例外ハンドラを使用します。

```
DECLARE
     fil BFILE;
     pos INTEGER;
     amt BINARY_INTEGER;
     buf RAW(40);
BEGIN
     SELECT f_lob INTO fil FROM lob_table WHERE key_value = 21;
     dbms_lob.open(fil, dbms_lob.lob_readonly);
     amt := 40; pos := 1 + dbms_lob.getlength(fil); buf := '';
     dbms_lob.read(fil, amt, pos, buf);
     dbms_output.put_line('Read F1 past EOF: '||
          utl_raw.cast_to_varchar2(buf));
     dbms_lob.close(fil);
     exception
     WHEN no_data_found
     THEN
       BEGIN
         dbms_output.put_line('End of File reached. Closing file');
         dbms_lob.fileclose(fil);
         -- or dbms_lob.filecloseall if appropriate
       END;
END;
     /

Statement processed.
End of File reached. Closing file
```

通常、DBMS_LOB を使用して PL/SQL ブロック内でオープンされたファイルは、ブロックの正常終了または異常終了の前にクローズしてください。

一時 LOB

Oracle9*i* では、一時 LOB の定義、作成、削除、アクセスおよび更新がサポートされています。一時 LOB データは、ユーザーの一時表領域に格納されています。一時 LOB はデータベースに永続的には格納されません。この目的は、主に LOB データの変換の実行です。

一時 LOB は空の状態で作成されます。デフォルトでは一時 LOB は、それが作成されたセッションの終了時にすべて削除されます。処理が途中で停止したり、データベースがクラッシュした場合、一時 LOB は削除され、その領域は解放されます。

Oracle9*i* には、一時 LOB を論理バケットにまとめてグループ化するインタフェースもあります。一時 LOB 用のこの論理記憶域は期間で表されます。各一時 LOB には、CACHE/NOCACHE など、別々の記憶特性があります。各セッションごとにデフォルトの記憶域があり、ユーザーが特定の期間を指定しない場合、一時 LOB はこの記憶域に格納されます。さらに、期間ごとに解放操作を実行して、期間内のすべてのコンテンツを解放できます。

一時 LOB に関する、読込み一貫性（CR）、取消し、バックアップ、パラレル処理またはトランザクション管理はサポートされていません。一時 LOB には、CR およびロールバックがサポートされていないため、エラーが発生した場合、ユーザーは一時 LOB を解放して、最初から操作をやり直す必要があります。

CR、取消しおよびバージョンが、一時 LOB に対しては生成されないため、同じ一時 LOB に複数のロケータを割り当てると、パフォーマンスに影響する可能性があります。具体的には、各ロケータが一時 LOB の独自のコピーを所有するためです。

別のロケータが一時 LOB を指しているときに、その内容を変更する場合は、一時 LOB のコピーが作成されます。この場合、変更操作が行われたロケータは、一時 LOB の新しいコピーを指します。これ以降、他のロケータは、変更操作が行われたロケータと同じデータは参照しません。このような状況の場合、永続 LOB でディープ・コピーは行われません。これは、CR スナップショットおよびバージョン・ページによって、ユーザーは独自のバージョンの LOB を簡単に参照できるためです。

必要な場合は、OCI のロケータへのポインタを使用し、ロケータへの複数のポインタが同じ一時 LOB ロケータを指すように設定することによって、擬似 REF 構文を取得できます。PL/SQL では、1 つの一時 LOB に対して複数のロケータは使用しないでください。一時 LOB ロケータは、参照によって別のプロシージャに渡される可能性があります。

一時 LOB は表スキーマと関連付けられていないため、行内および行外の一時 LOB という概念はありません。ユーザーが一時 LOB インスタンスを作成すると、エンジンによって LOB データへのロケータが作成され戻されます。PL/SQL の DBMS_LOB パッケージ、PRO*C、OCI およびその他のプログラム・インタフェースは、このロケータを介して、永続 LOB に対する操作と同様に一時 LOB を操作します。

クライアント側の一時 LOB はサポートされていません。一時 LOB はすべてサーバーに常駐します。

一時 LOB は、永続 LOB がサポートしている EMPTY_BLOB または EMPTY_CLOB ファンクションをサポートしません。EMPTY_BLOB ファンクションは、LOB の初期化を指定しますが、データの移入は行いません。

一時 LOB インスタンスは、適切な FREETEMPORARY または OCIDurationEnd 文で OCI または DBMS_LOB パッケージを使用したときのみ破棄できます。

一時 LOB インスタンスは、適切な OCI および DBMS_LOB 文を使用することによって、標準の永続内部 LOB と同様にアクセスおよび変更できます。一時 LOB を永続 LOB に変更するには、OCI または DBMS_LOB の COPY コマンドを明示的に使用して、一時 LOB を永続 LOB にコピーする必要があります。

セキュリティは、LOB ロケータを介して提供されます。一時 LOB は、その作成ユーザーのみ参照できます。ロケータは、あるユーザーのセッションから別のユーザーのセッションに渡すことはできません。あるセッションから別のセッションにロケータを渡しても、元のセッションの一時 LOB にはアクセスできません。一時 LOB データ参照は、各ユーザー固有のセッションに限定されます。別の場所のロケータを使用するユーザーは、同じ LOB ID を持つ自分のセッション内でのみ LOB にアクセスできます。ユーザーがこれを試行することはお薦めしませんが、試行した場合でも他のユーザーのデータに影響を与えることはありません。

Oracle は、V$TEMPORARY_LOBS と呼ばれる v$ ビューにセッションごとの一時 LOB を記録します。この中には、セッションごとに存在している一時 LOB 数に関する情報が含まれています。V$ ビューは、DBA が使用するためのものです。セッションから、Oracle はどのユーザーがその一時 LOB を所有しているかを判断できます。V$TEMPORARY_LOBS を DBA_SEGMENTS と組み合せて使用すると、DBA は、セッションで一時 LOB 用に使用されている領域の量を把握できます。これらの表を使用すると、DBA は一時 LOB が使用している一時領域の緊急クリーンアップを監視して指示できます。

一時 LOB の使用上の注意

1. 入力パラメータのいずれかが NULL の場合、DBMS_LOB のファンクションはすべて NULL を戻します。LOB ロケータが NULL で入力されると、DBMS_LOB のすべてのプロシージャで例外が発生します。
2. CLOB に基づく操作では、パラメータ（CLOB パラメータや VARCHAR2 のバッファやパターンなど）のキャラクタ・セット ID が一致しているかどうかは検証されません。この確認はユーザーが各自で行ってください。
3. データ記憶域リソースは、DBA が異なるテンポラリ表領域を作成して制御します。必要に応じて、DBA はユーザーごとに別々のテンポラリ表領域を定義できます。

4. 一時 LOB は値構文に準拠しています。これは、永続 LOB との一貫性を保ち、LOB 用の ANSI 規格に従うためです。このため、OCILobLocatatorAssign または PL/SQL で同じ割当てが実行されるたびに、データベースでは一時 LOB のコピーが作成されます。

   各ロケータは、それぞれ独自の LOB 値を指します。一時 LOB を作成するためにあるロケータが使用され、そのロケータが、OCI の OCILobLOcatorAssign を使用して、または PL/SQL の割当て操作によって別の LOB ロケータに割り当てられた場合、データベースは元の一時 LOB のコピーを作成し、2 番目のロケータがそのコピーを指すようにします。

   複数のユーザーが同じ LOB を変更できるようにするには、同じロケータを使用する必要があります。OCI では、ロケータへのポインタを使用して、そのポインタが同じロケータを指すように割り当てることによって、比較的簡単にこの処理を実行できます。PL/SQL では、同様の効果を得るには、同じ LOB 変数を使用して LOB を更新する必要があります。

   次の例は、コピーが作成される場合、または最低 1 回サーバーへの特別なラウンドトリップが行われる場合を示しています。

```
DECLARE
  a blob;
  b blob;
BEGIN
  dbms_lob.createtemporary(b, TRUE);
  -- the following assignment results in a deep copy
  a := b;
END;
```

   PL/SQL コンパイラは、OUT または IN OUT パラメータにバインドされる実引数のテンポラリ・コピーを作成します。実パラメータが一時 LOB の場合、テンポラリ・コピーはディープ（値）・コピーです。

   次の PL/SQL ブロックは、一時 LOB を IN OUT パラメータとして渡すことによって、ディープ・コピーが行われる例を示しています。

```
DECLARE
  a blob;
  procedure foo(parm IN OUT blob) is
  BEGIN
   ...
  END;
BEGIN
  dbms_lob.createtemporary(a, TRUE);
  -- the following call results in a deep copy of the blob a
  foo(a);
END;
```

PL/SQL パラメータの受渡しのためのディープ・コピーを最小限にするためには、可能であれば NOCOPY のコンパイラ・ヒントを使用します。

dbms_lob.createtemporary() に渡される継続期間パラメータがヒントです。新しい一時 LOB の継続期間は、PL/SQL のロケータ変数の継続期間と同じです。たとえば、前述のプログラム・ブロックでは、プログラム変数 a には常駐フレームの継続期間が設定されます。したがって、ブロックの終わりでは、ファンクションの終了時に a のメモリーが解放されます。

PL/SQL パッケージ変数を使用して一時 LOB が作成された場合、SESSION の継続期間を持つパッケージ変数の継続期間が設定されます。

```
BEGIN
   y clob;
  END;
/
BEGIN
   dbms_lob.createtemporary(package.y, TRUE);
END;
```

関連項目： NOCOPY 構文の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

例外

表 23-3 DBMS_LOB の例外

例外	コード	説明
INVALID_ARGVAL	21560	引数 %s の値が無効です。
ACCESS_ERROR	22925	%s で LOB の最大サイズを超えて、読込みまたは書込みを実行しようとしました。
NO_DATA_FOUND		ループ読取り操作用の EndofLob のインジケータ。ハード・エラーではありません。
VALUE_ERROR	6502	サブプログラムのパラメータの値が無効のため、PL/SQL エラーが発生しました。

DBMS_LOB サブプログラムの要約

表 23-4 DBMS_LOB のサブプログラム

サブプログラム	説明
「APPEND プロシージャ」 23-16 ページ	ソース LOB の内容を宛先 LOB に追加します。
「CLOSE プロシージャ」 23-18 ページ	オープンしている内部または外部 LOB をクローズします。
「COMPARE ファンクション」 23-19 ページ	2 つの LOB 全体、または 2 つの LOB の一部を比較します。
「COPY プロシージャ」 23-22 ページ	ソース LOB 全体または一部を宛先 LOB にコピーします。
「CREATETEMPORARY プロシージャ」 23-25 ページ	一時 BLOB または CLOB およびそれに対応する索引をユーザーのデフォルト一時表領域に作成します。
「ERASE プロシージャ」 23-26 ページ	LOB 全体または一部を消去します。
「FILECLOSE プロシージャ」 23-28 ページ	ファイルをクローズします。
「FILECLOSEALL プロシージャ」 23-29 ページ	オープンしているファイルをすべてクローズします。
「FILEEXISTS ファンクション」 23-30 ページ	ファイルがサーバー上に存在しているかどうかをチェックします。
「FILEGETNAME プロシージャ」 23-31 ページ	ディレクトリ別名およびファイル名を取得します。
「FILEISOPEN ファンクション」 23-32 ページ	入力 BFILE ロケータを使用してファイルがオープンされたかどうかをチェックします。
「FILEOPEN プロシージャ」 23-34 ページ	ファイルをオープンします。
「FREETEMPORARY プロシージャ」 23-35 ページ	ユーザーのデフォルト一時表領域にある一時 BLOB または CLOB を解放します。
「GETCHUNKSIZE ファンクション」 23-36 ページ	LOB 値を格納する LOB チャンクの使用領域容量を戻します。
「GETLENGTH ファンクション」 23-37 ページ	LOB 値の長さを取得します。

表 23-4 DBMS_LOB のサブプログラム（続き）

サブプログラム	説明
「INSTR ファンクション」 23-39 ページ	LOB にあるパターンの *n* 番目の一致のマッチング位置を戻します。
「ISOPEN ファンクション」 23-42 ページ	LOB が入力ロケータを使用して、すでにオープンされたかどうかをチェックします。
「ISTEMPORARY ファンクション」 23-43 ページ	ロケータが一時 LOB を指しているかどうかをチェックします。
「LOADFROMFILE プロシージャ」 23-43 ページ	BFILE データを内部 LOB にロードします。
「LOADBLOBFROMFILE プロシージャ」 23-46 ページ	BFILE データを内部 BLOB にロードします。
「LOADCLOBFROMFILE プロシージャ」 23-48 ページ	BFILE データを内部 CLOB にロードします。
「OPEN プロシージャ」 23-52 ページ	指定されたモードで LOB（内部、外部またはテンポラリ）をオープンします。
「READ プロシージャ」 23-53 ページ	指定されたオフセット以降の LOB データを読み込みます。
「SUBSTR ファンクション」 23-57 ページ	指定されたオフセット以降の LOB 値の一部を戻します。
「TRIM プロシージャ」 23-59 ページ	LOB 値を指定された長さに切り捨てます。
「WRITE プロシージャ」 23-61 ページ	LOB の指定されたオフセット以降にデータを書き込みます。
「WRITEAPPEND プロシージャ」 23-64 ページ	LOB の終わり以降にバッファを書き込みます。

APPEND プロシージャ

このプロシージャは、ソース内部 LOB の内容を宛先 LOB に追加します。ソース LOB 全体を追加します。

APPEND プロシージャは 2 種類、オーバーロードされています。

構文

```
DBMS_LOB.APPEND (
   dest_lob IN OUT  NOCOPY BLOB,
   src_lob  IN             BLOB);

DBMS_LOB.APPEND (
   dest_lob IN OUT  NOCOPY CLOB  CHARACTER SET ANY_CS,
   src_lob  IN             CLOB  CHARACTER SET dest_lob%CHARSET);
```

パラメータ

表 23-5 APPEND プロシージャのパラメータ

パラメータ	説明
dest_lob	データを追加する内部 LOB のロケータ。
src_lob	データを読み取る内部 LOB のロケータ。

例外

表 23-6 APPEND プロシージャの例外

例外	説明
VALUE_ERROR	ソースまたは宛先 LOB のいずれかが NULL です。

使用上の注意

オープン / クローズ API 内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

オープン / クローズ API 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

例

```
CREATE OR REPLACE PROCEDURE Example_1a IS
    dest_lob BLOB;
    src_lob  BLOB;
BEGIN
    -- get the LOB locators
    -- note that the FOR UPDATE clause locks the row
    SELECT b_lob INTO dest_lob
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
    SELECT b_lob INTO src_lob
        FROM lob_table
        WHERE key_value = 21;
    DBMS_LOB.APPEND(dest_lob, src_lob);
    COMMIT;
EXCEPTION
    WHEN some_exception
    THEN handle_exception;
END;

CREATE OR REPLACE PROCEDURE Example_1b IS
    dest_lob, src_lob  BLOB;
BEGIN
    -- get the LOB locators
    -- note that the FOR UPDATE clause locks the row
    SELECT b_lob INTO dest_lob
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
    SELECT b_lob INTO src_lob
        FROM lob_table
        WHERE key_value = 12;
    DBMS_LOB.APPEND(dest_lob, src_lob);
    COMMIT;
EXCEPTION
    WHEN some_exception
    THEN handle_exception;
END;
```

CLOSE プロシージャ

このプロシージャは、オープンしている内部または外部 LOB をクローズします。

構文

```
DBMS_LOB.CLOSE (
   lob_loc    IN OUT NOCOPY BLOB);

DBMS_LOB.CLOSE (
   lob_loc    IN OUT NOCOPY CLOB CHARACTER SET ANY_CS);

DBMS_LOB.CLOSE (
   file_loc   IN OUT NOCOPY BFILE);
```

エラー

BFILE が存在していてもオープンしていない場合、エラーは戻されません。エラーは、LOB がオープンしていない場合に戻されます。

使用上の注意

CLOSE では、内部または外部 LOB のいずれの場合もサーバーへのラウンドトリップが必要です。内部 LOB の場合、CLOSE はクローズ・コールに依存するその他のコードをトリガーし、外部 LOB（BFILE）の場合は、サーバー側のオペレーティング・システム・ファイルを実際にクローズします。

オープン / クローズ API 内のすべての LOB 操作のラップは必須ではありません。ただし、LOB をオープンしている場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。クローズしない場合は、エラーが発生します。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

トランザクションによりオープンされた LOB をすべてクローズする前にトランザクションをコミットした場合、エラーが発生します。エラーが戻された場合 LOB のオープンは廃棄されますが、トランザクションのコミットは正常に行われます。したがって、そのトランザクションにおける LOB および非 LOB データの変更はすべてコミットされますが、ドメイン索引およびファンクションベースの索引は更新されません。このような場合は、LOB 列のファンクション索引およびドメイン索引を再構築する必要があります。

COMPARE ファンクション

このファンクションは、2 つの LOB 全体、または 2 つの LOB の一部を比較します。同じデータ・タイプの LOB のみ比較できます（BLOB タイプの LOB とその他の BLOB、CLOB と CLOB、および BFILE と BFILE を比較できます）。BFILE の場合は、この操作を行う前に、FILEOPEN 操作でファイルをあらかじめオープンしておく必要があります。

offset および amount パラメータによって指定した範囲のデータがすべて完全に一致すると 0（ゼロ）が戻されます。一致しない場合は 0（ゼロ）以外の INTEGER が戻されます。

固定幅の *n* バイトの CLOB の場合は、COMPARE に対する入力 amount の指定が（4294967295/*n*）を超えると、（4294967295/*n*）または Max（length(clob1), length(clob2)）のサイズのうち、小さい方の範囲で文字を比較します。

構文

```
DBMS_LOB.COMPARE (
   lob_1            IN BLOB,
   lob_2            IN BLOB,
   amount           IN INTEGER := 4294967295,
   offset_1         IN INTEGER := 1,
   offset_2         IN INTEGER := 1)
  RETURN INTEGER;

DBMS_LOB.COMPARE (
   lob_1            IN CLOB  CHARACTER SET ANY_CS,
   lob_2            IN CLOB  CHARACTER SET lob_1%CHARSET,
   amount           IN INTEGER := 4294967295,
   offset_1         IN INTEGER := 1,
   offset_2         IN INTEGER := 1)
  RETURN INTEGER;

DBMS_LOB.COMPARE (
   lob_1            IN BFILE,
   lob_2            IN BFILE,
   amount           IN INTEGER,
   offset_1         IN INTEGER := 1,
   offset_2         IN INTEGER := 1)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(COMPARE, WNDS, WNPS, RNDS, RNPS);
```

パラメータ

表 23-7 COMPARE ファンクションのパラメータ

パラメータ	説明
lob_1	最初の比較対象の LOB ロケータ。
lob_2	2 番目の比較対象の LOB ロケータ。
amount	比較するバイト数（BLOB の場合）または文字数（CLOB の場合）。
offset_1	最初の LOB の比較開始位置を示すオフセット（起点 : 1）。バイト数または文字数で指定します。
offset_2	2 番目の LOB の比較開始位置を示すオフセット（起点 : 1）。バイト数または文字数で指定します。

戻り値

- INTEGER: 比較が正常に行われた場合は 0（ゼロ）、それ以外の場合は 0（ゼロ）以外の整数が戻されます。
- 次の場合に NULL が戻されます。
  - amount < 1
  - amount > LOBMAXSIZE
  - offset_1 または offset_2 < 1
    - offset_1 または offset_2 > LOBMAXSIZE

例外

表 23-8 BFILE 操作に関する COMPARE ファンクションの例外

例外	説明
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```
CREATE OR REPLACE PROCEDURE Example2a IS
    lob_1, lob_2      BLOB;
    retval            INTEGER;
BEGIN
    SELECT b_col INTO lob_1 FROM lob_table
        WHERE key_value = 45;
    SELECT b_col INTO lob_2 FROM lob_table
        WHERE key_value = 54;
    retval := dbms_lob.compare(lob_1, lob_2, 5600, 33482,
         128);
    IF retval = 0 THEN
      ;    -- process compared code
    ELSE
      ;    -- process not compared code
    END IF;
END;

CREATE OR REPLACE PROCEDURE Example_2b IS
    fil_1, fil_2       BFILE;
    retval             INTEGER;
BEGIN

    SELECT f_lob INTO fil_1 FROM lob_table WHERE key_value = 45;
    SELECT f_lob INTO fil_2 FROM lob_table WHERE key_value = 54;
    dbms_lob.fileopen(fil_1, dbms_lob.file_readonly);
    dbms_lob.fileopen(fil_2, dbms_lob.file_readonly);
    retval := dbms_lob.compare(fil_1, fil_2, 5600,
                                3348276, 2765612);
    IF (retval = 0)
    THEN
        ; -- process compared code
    ELSE
        ; -- process not compared code
    END IF;
    dbms_lob.fileclose(fil_1);
    dbms_lob.fileclose(fil_2);
END;
```

COPY プロシージャ

このプロシージャは、ソース内部 LOB の全体または一部を宛先内部 LOB にコピーします。ソースおよび宛先 LOB の両方に対するオフセット、およびコピーするバイト数または文字数を指定できます。

宛先 LOB に指定したオフセットが、現在その LOB に格納されているデータの終わりを超えている場合は、宛先 BLOB または CLOB に、0（ゼロ）バイトの FILLER または空白がそれぞれ挿入されます。オフセットが宛先 LOB の現行の長さより小さい場合、既存のデータは上書きされます。

ソース LOB のデータ長を超える量を指定してもエラーにはなりません。したがって、ソース LOB のデータを、src_offset からソース LOB の終わりまでコピーする大量のコピーを指定できます。

構文

```
DBMS_LOB.COPY (
  dest_lob    IN OUT NOCOPY BLOB,
  src_lob     IN            BLOB,
  amount      IN            INTEGER,
  dest_offset IN            INTEGER := 1,
  src_offset  IN            INTEGER := 1);

DBMS_LOB.COPY (
  dest_lob    IN OUT NOCOPY CLOB  CHARACTER SET ANY_CS,
  src_lob     IN            CLOB  CHARACTER SET dest_lob%CHARSET,
  amount      IN            INTEGER,
  dest_offset IN            INTEGER := 1,
  src_offset  IN            INTEGER := 1);
```

パラメータ

表 23-9 COPY プロシージャのパラメータ

パラメータ	説明
dest_lob	コピー先の LOB ロケータ。
src_lob	コピー元の LOB ロケータ。
amount	コピーするバイト数（BLOB の場合）または文字数（CLOB の場合）。
dest_offset	宛先 LOB のコピー開始位置を示すオフセット（起点 : 1）。バイト数または文字数で指定します。
src_offset	ソース LOB のコピー開始位置を示すオフセット（起点 : 1）。バイト数または文字数で指定します。

例外

表 23-10 COPY プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL または無効です。
INVALID_ARGVAL	次のいずれかです。 − src_offset または dest_offset < 1 − src_offset または dest_offset > LOBMAXSIZE − amount < 1 − amount > LOBMAXSIZE

使用上の注意

オープン / クローズ API 内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

オープン / クローズ API 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

例

```
CREATE OR REPLACE PROCEDURE Example_3a IS
    lobd, lobs     BLOB;
    dest_offset    INTEGER := 1
    src_offset     INTEGER := 1
    amt            INTEGER := 3000;
BEGIN
    SELECT b_col INTO lobd
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
    SELECT b_col INTO lobs
        FROM lob_table
        WHERE key_value = 21;
    DBMS_LOB.COPY(lobd, lobs, amt, dest_offset, src_offset);
    COMMIT;
   EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;

CREATE OR REPLACE PROCEDURE Example_3b IS
    lobd, lobs     BLOB;
    dest_offset    INTEGER := 1
    src_offset     INTEGER := 1
    amt            INTEGER := 3000;
BEGIN
    SELECT b_col INTO lobd
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
    SELECT b_col INTO lobs
        FROM lob_table
        WHERE key_value = 12;
    DBMS_LOB.COPY(lobd, lobs, amt, dest_offset, src_offset);
    COMMIT;
   EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

CREATETEMPORARY プロシージャ

このプロシージャは、一時 BLOB または CLOB およびそれに対応する索引をユーザーのデフォルト一時表領域に作成します。

構文

```
DBMS_LOB.CREATETEMPORARY (
   lob_loc IN OUT NOCOPY BLOB,
   cache   IN            BOOLEAN,
   dur     IN            PLS_INTEGER := 10);

DBMS_LOB.CREATETEMPORARY (
   lob_loc IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
   cache   IN            BOOLEAN,
   dur     IN            PLS_INTEGER := 10);
```

パラメータ

表 23-11 CREATETEMPORARY プロシージャのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ。
cache	LOB をバッファ・キャッシュに読み込むかどうかを指定します。
dur	2 つの事前定義の継続時間（SESSION または CALL）のうちの 1 つ。一時 LOB のクリーンアップをセッション終了時に行うか、コール終了時に行うかを指定します。 dur が省略された場合は、セッション継続時間が使用されます。

例

```
DBMS_LOB.CREATETEMPORARY(Dest_Loc, TRUE)
```

関連項目： NOCOPY と一時 LOB をパラメータとして渡す方法は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

ERASE プロシージャ

このプロシージャは、内部 LOB 全体、または内部 LOB の一部を消去します。

> **注意：** LOB の長さは、LOB のセクションを消去しても減少しません。LOB 値の長さを減らす方法は、23-59 ページの「TRIM プロシージャ」を参照してください。

LOB の中央部のデータが消去されると、BLOB または CLOB には、0（ゼロ）バイトの FILLER または空白がそれぞれ書き込まれます。

指定した数を消去する前に LOB の終わりに達した場合、実際に消去されたバイト数または文字数は、amount パラメータで指定した数と異なる場合があります。実際に消去された文字数またはバイト数は、amount パラメータに戻されます。

構文

```
DBMS_LOB.ERASE (
   lob_loc          IN OUT   NOCOPY   BLOB,
   amount           IN OUT   NOCOPY   INTEGER,
   offset           IN                INTEGER := 1);

DBMS_LOB.ERASE (
   lob_loc          IN OUT   NOCOPY   CLOB CHARACTER SET ANY_CS,
   amount           IN OUT   NOCOPY   INTEGER,
   offset           IN                INTEGER := 1);
```

パラメータ

表 23-12 ERASE プロシージャのパラメータ

パラメータ	説明
lob_loc	消去する LOB のロケータ。
amount	消去するバイト数（BLOB または BFILES の場合）または文字数（CLOB または NCLOB の場合）。
offset	LOB の先頭からの絶対オフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。

例外

表 23-13 ERASE プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL です。
INVALID_ARGVAL	次のいずれかです。 − amount < 1 または amount > LOBMAXSIZE − offset < 1 または offset > LOBMAXSIZE

使用上の注意

オープン / クローズ API 内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

オープン / クローズ API 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

例

```
CREATE OR REPLACE PROCEDURE Example_4 IS
    lobd        BLOB;
    amt         INTEGER := 3000;
BEGIN
    SELECT b_col INTO lobd
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
    dbms_lob.erase(dest_lob, amt, 2000);
    COMMIT;
END;
```

関連項目： 23-59 ページ「TRIM プロシージャ」

FILECLOSE プロシージャ

このプロシージャは、入力ロケータによってすでにオープンされている BFILE をクローズします。

> **注意：** Oracle には、BFILE に対する読取り専用アクセスしかありません。つまり、Oracle を介して BFILE に書き込むことはできません。

構文

```
DBMS_LOB.FILECLOSE (
    file_loc IN OUT NOCOPY BFILE);
```

パラメータ

表 23-14 FILECLOSE プロシージャのパラメータ

パラメータ	説明
file_loc	クローズする BFILE のロケータ。

例外

表 23-15 FILECLOSE プロシージャの例外

例外	説明
VALUE_ERROR	file_loc の入力値が NULL です。
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```
CREATE OR REPLACE PROCEDURE Example_5 IS
    fil BFILE;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 99;
    dbms_lob.fileopen(fil);
    -- file operations
```

```
    dbms_lob.fileclose(fil);
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

関連項目：

- 23-34 ページ「FILEOPEN プロシージャ」
- 23-29 ページ「FILECLOSEALL プロシージャ」

FILECLOSEALL プロシージャ

このプロシージャは、セッションでオープンされたすべての BFILE をクローズします。

構文

```
DBMS_LOB.FILECLOSEALL;
```

例外

表 23-16 FILECLOSEALL プロシージャの例外

例外	説明
UNOPENED_FILE	セッションでオープンされたファイルはありません。

例

```
CREATE OR REPLACE PROCEDURE Example_6 IS
    fil BFILE;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 99;
    dbms_lob.fileopen(fil);
    -- file operations
    dbms_lob.filecloseall;
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

関連項目：

- 23-34 ページ「FILEOPEN プロシージャ」
- 23-28 ページ「FILECLOSE プロシージャ」

FILEEXISTS ファンクション

このファンクションは、指定した BFILE ロケータが、サーバーのファイル・システムに実際に存在しているファイルを指しているかどうかを検証します。

構文

```
DBMS_LOB.FILEEXISTS (
   file_loc     IN    BFILE)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(FILEEXISTS, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 23-17 FILEEXISTS ファンクションのパラメータ

パラメータ	説明
file_loc	BFILE のロケータ。

戻り値

表 23-18 FILEEXISTS ファンクションの戻り値

戻り値	説明
0	物理ファイルが存在しません。
1	物理ファイルが存在します。

例外

表 23-19 FILEEXISTS ファンクションの例外

例外	説明
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。

例

```
CREATE OR REPLACE PROCEDURE Example_7 IS
    fil BFILE;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 12;
    IF (dbms_lob.fileexists(fil))
    THEN
        ; -- file exists code
    ELSE
        ; -- file does not exist code
    END IF;
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

関連項目： 23-32 ページ 「FILEISOPEN ファンクション」

FILEGETNAME プロシージャ

このプロシージャは、指定した BFILE ロケータのディレクトリ別名とファイル名を判別します。ロケータに割り当てられたディレクトリの別名とファイル名を示すのみで、物理ファイルまたはディレクトリが実際に存在しているかどうかは判別しません。

dir_alias バッファの最大値は 30 で、パス名全体の最大値は 2000 です。

構文

```
DBMS_LOB.FILEGETNAME (
   file_loc   IN    BFILE,
   dir_alias  OUT   VARCHAR2,
   filename   OUT   VARCHAR2);
```

パラメータ

表 23-20 FILEGETNAME プロシージャのパラメータ

パラメータ	説明
file_loc	BFILE のロケータ。
dir_alias	ディレクトリ別名。
filename	BFILE 名。

例外

表 23-21 FILEGETNAME プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL または INVALID です。
INVALID_ARGVAL	dir_alias またはファイル名が NULL です。

例

```
CREATE OR REPLACE PROCEDURE Example_8 IS
    fil BFILE;
    dir_alias VARCHAR2(30);
    name VARCHAR2(2000);
BEGIN
    IF (dbms_lob.fileexists(fil))
    THEN
        dbms_lob.filegetname(fil, dir_alias, name);
        dbms_output.put_line("Opening " || dir_alias || name);
        dbms_lob.fileopen(fil, dbms_lob.file_readonly);
        -- file operations
        dbms_output.fileclose(fil);
    END IF;
END;
```

FILEISOPEN ファンクション

このファンクションは、BFILE が特定の FILE ロケータでオープンされたかどうかを検証します。

入力 FILE ロケータが FILEOPEN プロシージャに渡されていない場合、そのファイルはこのロケータによってオープンされていないとみなされます。ただし、別のロケータがこのファイルをオープンしている可能性はあります。つまり、オープンされているかどうかは特定のロケータと関連付けられています。

構文

```
DBMS_LOB.FILEISOPEN (
   file_loc   IN    BFILE)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(FILEISOPEN, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 23-22 FILEISOPEN ファンクションのパラメータ

パラメータ	説明
file_loc	BFILE のロケータ。

戻り値

INTEGER: 0 = ファイルはオープンされていません。1 = ファイルはオープンされています。

例外

表 23-23 FILEISOPEN ファンクションの例外

例外	説明
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。

例

```
CREATE OR REPLACE PROCEDURE Example_9 IS
DECLARE
    fil      BFILE;
    pos      INTEGER;
    pattern  VARCHAR2(20);
BEGIN
    SELECT f_lob INTO fil FROM lob_table
        WHERE key_value = 12;
    -- open the file
    IF (dbms_lob.fileisopen(fil))
    THEN
        pos := dbms_lob.instr(fil, pattern, 1025, 6);
         -- more file operations
         dbms_lob.fileclose(fil);
   ELSE
        ; -- return error
    END IF;
END;
```

関連項目： 23-30 ページ「FILEEXISTS ファンクション」

FILEOPEN プロシージャ

このプロシージャは、BFILE を読取り専用アクセスでオープンします。BFILE は、Oracle を介して書き込むことはできません。

構文

```
DBMS_LOB.FILEOPEN (
   file_loc   IN OUT NOCOPY  BFILE,
   open_mode  IN             BINARY_INTEGER := file_readonly);
```

パラメータ

表 23-24 FILEOPEN プロシージャのパラメータ

パラメータ	説明
file_loc	BFILE のロケータ。
open_mode	ファイル・アクセスは読取り専用です。

例外

表 23-25 FILEOPEN プロシージャの例外

例外	説明
VALUE_ERROR	file_loc または open_mode が NULL です。
INVALID_ARGVAL	open_mode が FILE_READONLY ではありません。
OPEN_TOOMANY	セッション内のオープン・ファイル数が session_max_open_files を超えています。
NOEXIST_DIRECTORY	file_loc に関連付けられたディレクトリが存在しません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```
CREATE OR REPLACE PROCEDURE Example_10 IS
    fil BFILE;
BEGIN
    -- open BFILE
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 99;
    IF (dbms_lob.fileexists(fil))
    THEN
        dbms_lob.fileopen(fil, dbms_lob.file_readonly);
        -- file operation
        dbms_lob.fileclose(fil);
    END IF;
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

関連項目：

- 23-28 ページ「FILECLOSE プロシージャ」
- 23-29 ページ「FILECLOSEALL プロシージャ」

FREETEMPORARY プロシージャ

このプロシージャは、ユーザーのデフォルト一時表領域内の一時 BLOB または CLOB を解放します。FREETEMPORARY へのコール後、解放された LOB ロケータには無効のマークが設定されます。

無効の LOB ロケータが、OCI の OCILobLocatorAssign を使用して、または PL/SQL の割当て操作によって別の LOB ロケータに割り当てられている場合、割当て先も解放され、無効のマークが設定されます。

構文

```
DBMS_LOB.FREETEMPORARY (
   lob_loc  IN OUT  NOCOPY BLOB);

DBMS_LOB.FREETEMPORARY (
   lob_loc  IN OUT  NOCOPY CLOB CHARACTER SET ANY_CS);
```

パラメータ

表 23-26 FREETEMPORARY プロシージャのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ。

例

```
DECLARE
  a blob;
  b blob;
BEGIN
  dbms_lob.createtemporary(a, TRUE);
  dbms_lob.createtemporary(b, TRUE);
  ...
  -- the following call frees lob a
  dbms_lob.freetemporary(a);
  -- at this point lob locator a is marked as invalid
  -- the following assignment frees the lob b and marks it as invalid
also
  b := a;
END;
```

GETCHUNKSIZE ファンクション

表の作成時に、Oracle ブロックの倍数でチャンク・ファクタを指定できます。これは、LOB 値のアクセスまたは変更時に LOB データ・レイヤーによって使用されるチャンク・サイズに対応します。チャンクの一部はシステム関連情報の格納に使用され、それ以外の部分に LOB 値が格納されます。

このファンクションは、LOB 値を格納する LOB チャンクで使用される領域容量を戻します。

構文

```
DBMS_LOB.GETCHUNKSIZE (
   lob_loc IN BLOB)
  RETURN INTEGER;

DBMS_LOB.GETCHUNKSIZE (
   lob_loc IN CLOB CHARACTER SET ANY_CS)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(GETCHUNKSIZE, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 23-27 GETCHUNKSIZE ファンクションのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ。

戻り値

BLOB の場合に戻される値はバイト単位です。CLOB の場合に戻される値は文字単位です。

使用上の注意

このチャンク・サイズの倍数を使用して読取り / 書込み要求を入力するとパフォーマンスが改善されます。LOB チャンクはバージョン化されるため、書込みの場合にはさらに利点があります。すべての書込みがチャンクを基準に行われると、余分なバージョン分割が行われず、重複もしません。同じチャンクに対して WRITE コールを複数回送信するかわりに、1 つのチャンクのサイズ内で WRITE コールをまとめることができます。

GETLENGTH ファンクション

このファンクションは、指定された LOB 値の長さを取得します。長さは、バイト数または文字数で戻されます。

BFILE の場合に戻される長さには、EOF（存在している場合）が含まれます。以前に行った ERASE または WRITE 操作で LOB に挿入された 0（ゼロ）バイトまたは空白の FILLER も長さに含まれます。空の内部 LOB の長さは 0（ゼロ）です。

構文

```
DBMS_LOB.GETLENGTH (
   lob_loc    IN  BLOB)
  RETURN INTEGER;

DBMS_LOB.GETLENGTH (
   lob_loc    IN  CLOB   CHARACTER SET ANY_CS)
  RETURN INTEGER;

DBMS_LOB.GETLENGTH (
   file_loc    IN  BFILE)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(GETLENGTH, WNDS, WNPS, RNDS, RNPS);
```

パラメータ

表 23-28 GETLENGTH ファンクションのパラメータ

パラメータ	説明
file_loc	長さが戻される LOB のファイル・ロケータ。

戻り値

LOB の長さが、INTEGER としてバイト数または文字数で戻されます。入力 LOB が NULL であるか、または入力 lob_loc が NULL の場合は、NULL が戻されます。BFILE の場合は、次の場合にエラーが戻されます。

- lob_loc に必要なディレクトリ権限とオペレーティング・システム権限がない場合
- オペレーティング・システム読込みエラーのために、lob_loc を読み込むことができない場合

例

```
CREATE OR REPLACE PROCEDURE Example_11a IS
    lobd        BLOB;
    length      INTEGER;
BEGIN
    -- get the LOB locator
    SELECT b_lob INTO lobd FROM lob_table
        WHERE key_value = 42;
    length := dbms_lob.getlength(lobd);
    IF length IS NULL THEN
        dbms_output.put_line('LOB is null.');
    ELSE
        dbms_output.put_line('The length is '
            || length);
    END IF;
END;

CREATE OR REPLACE PROCEDURE Example_11b IS
DECLARE
    len INTEGER;
    fil BFILE;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 12;
    len := dbms_lob.length(fil);
END;
```

INSTR ファンクション

このファンクションは、指定されたオフセットを開始位置として、LOB におけるパターンの *n* 番目に一致した位置を戻します。

VARCHAR2 バッファ（pattern パラメータ）の形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータのタイプが NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータのタイプが CLOB の場合、バッファには CHAR データが含まれる必要があります。

BFILE の場合は、この操作を行う前に、FILEOPEN 操作でファイルをあらかじめオープンしておく必要があります。

INSTR など、パターン・マッチング用に RAW または VARCHAR2 パラメータを受け入れる操作では、パターン・パラメータまたは副文字列において、標準の式または特殊一致文字（例：SQL の LIKE）はサポートされていません。

構文

```
DBMS_LOB.INSTR (
   lob_loc    IN   BLOB,
   pattern    IN   RAW,
   offset     IN   INTEGER := 1,
   nth        IN   INTEGER := 1)
  RETURN INTEGER;

DBMS_LOB.INSTR (
   lob_loc    IN   CLOB      CHARACTER SET ANY_CS,
   pattern    IN   VARCHAR2  CHARACTER SET lob_loc%CHARSET,
   offset     IN   INTEGER := 1,
   nth        IN   INTEGER := 1)
  RETURN INTEGER;

DBMS_LOB.INSTR (
   file_loc   IN   BFILE,
   pattern    IN   RAW,
   offset     IN   INTEGER := 1,
   nth        IN   INTEGER := 1)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(INSTR, WNDS, WNPS, RNDS, RNPS);
```

パラメータ

表 23-29 INSTR ファンクションのパラメータ

パラメータ	説明
lob_loc	検査する LOB のロケータ。
file_loc	検査する LOB のファイル・ロケータ。
pattern	テスト対象のパターン。パターンは、BLOB の場合は RAW バイトのグループ、CLOB の場合は文字列（VARCHAR2）です。パターンのサイズは、最大 16383 バイトです。
offset	パターン・マッチングの開始位置を示す絶対オフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。
nth	出現番号。1 から始まります。

戻り値

表 23-30 INSTR ファンクションの戻り値

戻り値	説明
INTEGER	一致したパターンの先頭のオフセット。バイト数または文字数で示されます。 パターンが見つからない場合は 0（ゼロ）が戻されます。
NULL	次のいずれかです。 － IN パラメータのいずれかが NULL または INVALID の場合 － offset < 1 または offset > LOBMAXSIZE － nth < 1 － nth > LOBMAXSIZE

例外

表 23-31 BFILES に関する INSTR ファンクションの例外

例外	説明
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。

表 23-31 BFILES に関する INSTR ファンクションの例外（続き）

例外	説明
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

例

```
CREATE OR REPLACE PROCEDURE Example_12a IS
    lobd        CLOB;
    pattern     VARCHAR2 := 'abcde';
    position    INTEGER  := 10000;
BEGIN
-- get the LOB locator
    SELECT b_col INTO lobd
        FROM lob_table
        WHERE key_value = 21;
    position := DBMS_LOB.INSTR(lobd,
                        pattern, 1025, 6);
    IF position = 0 THEN
        dbms_output.put_line('Pattern not found');
    ELSE
        dbms_output.put_line('The pattern occurs at '
                || position);
    END IF;
END;

CREATE OR REPLACE PROCEDURE Example_12b IS
DECLARE
    fil BFILE;
    pattern VARCHAR2;
    pos INTEGER;
BEGIN
    -- initialize pattern
    -- check for the 6th occurrence starting from 1025th byte
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 12;
    dbms_lob.fileopen(fil, dbms_lob.file_readonly);
    pos := dbms_lob.instr(fil, pattern, 1025, 6);
    dbms_lob.fileclose(fil);
END;
```

関連項目： 23-57 ページ「SUBSTR ファンクション」

ISOPEN ファンクション

このファンクションは、LOB が入力ロケータを使用してすでにオープンされたかどうかをチェックします。このサブプログラムは、内部および外部 LOB 用です。

構文

```
DBMS_LOB.ISOPEN (
   lob_loc IN BLOB)
  RETURN INTEGER;

DBMS_LOB.ISOPEN (
   lob_loc IN CLOB CHARACTER SET ANY_CS)
  RETURN INTEGER;

DBMS_LOB.ISOPEN (
   file_loc IN BFILE)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(ISOPEN, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 23-32 ISOPEN ファンクションのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ。
file_loc	ファイル・ロケータ。

使用上の注意

BFILES の場合、オープンされているかどうかはロケータと関連付けられています。入力ロケータが OPEN に渡されていない場合、その BFILE はこのロケータによってオープンされていないとみなされます。ただし、別のロケータが BFILE をオープンしている可能性はあります。異なるロケータを使用すると、同じ BFILE 上で複数の OPEN を実行できます。

内部 LOB の場合、オープンされているかどうかは、ロケータではなくその LOB に関連付けられています。ロケータ 1 が LOB をオープンした場合、ロケータ 2 もその LOB はオープンしているとみなします。内部 LOB の場合は、LOB が実際にオープンしているかどうかを調べるにはサーバー上の状態をチェックするため、ISOPEN でラウンドトリップが必要です。

外部 LOB（BFILE）の場合も、サーバーに状態が保持されているため、ISOPEN でラウンドトリップが必要です。

ISTEMPORARY ファンクション

構文

```
DBMS_LOB.ISTEMPORARY (
   lob_loc IN BLOB)
  RETURN INTEGER;

DBMS_LOB.ISTEMPORARY (
   lob_loc IN CLOB CHARACTER SET ANY_CS)
  RETURN INTEGER;
```

プラグマ

```
PRAGMA RESTRICT_REFERENCES(istemporary, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 23-33 ISTEMPORARY プロシージャのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ。
temporary	ブール値。LOB がテンポラリであるかどうかを示します。

戻り値

ロケータが一時 LOB を指している場合は、temporary に TRUE が戻されます。そうでない場合は FALSE が戻されます。

LOADFROMFILE プロシージャ

このプロシージャは、ソース外部 LOB（BFILE）の全体または一部を宛先内部 LOB にコピーします。

ソースおよび宛先 LOB の両方に対するオフセット、およびソース BFILE からコピーするバイト数を指定できます。amount および src_offset は BFILE を参照するため、バイト単位で、dest_offset は BLOB の場合はバイト、CLOB の場合は文字が単位です。

注意： 入力 BFILE は、このプロシージャを使用する前にオープンしている必要があります。バイナリ BFILE データが CLOB にロードされる場合、キャラクタ・セット変換は暗黙的には実行されません。BFILE データは、データベース内の CLOB と同じキャラクタ・セットであることが必要です。これを検証するためのエラー・チェックは実行されません。

宛先 LOB に指定したオフセットが、現在その LOB に格納されているデータの終わりを超えている場合は、宛先 BLOB または CLOB に、0（ゼロ）バイトの FILLER または空白がそれぞれ挿入されます。オフセットが宛先 LOB の現行の長さより小さい場合、既存のデータは上書きされます。

入力 amount および offset を加算した値が BFILE 内のデータの長さを超えた場合はエラーが発生します。

注意： UTF-8 などのようにキャラクタ・セットの幅が変化する場合、LOB 値は固定幅形式の UCS2 で格納されます。したがって、DBMS_LOB.LOADFROMFILE を使用している場合、BFILE のデータは UTF-8 キャラクタ・セットではなく UCS2 キャラクタ・セットにする必要があります。ただし、CLOB または NCLOB にデータをロードするには、LOADFROMFILE ではなく sql*loader を使用してください。sql*loader は、必要なキャラクタ・セット変換を提供します。

構文

```
DBMS_LOB.LOADFROMFILE (
   dest_lob    IN OUT NOCOPY BLOB,
   src_file    IN            BFILE,
   amount      IN            INTEGER,
   dest_offset IN            INTEGER  := 1,
   src_offset  IN            INTEGER  := 1);
```

パラメータ

表 23-34 LOADFROMFILE プロシージャのパラメータ

パラメータ	説明
dest_lob	ロード先の LOB ロケータ。
src_file	ロード元の BFILE ロケータ。
amount	BFILE からロードするバイト数。
dest_offset	宛先 LOB のロード開始位置を示すオフセット（起点 : 1）。バイト数または文字数で指定します。
src_offset	ソース BFILE のロード開始位置を示すオフセット（起点 : 1）。バイト数で指定します。

使用要件

オープン / クローズ API 内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

オープン / クローズ API 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

例外

表 23-35 LOADFROMFILE プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL または INVALID です。
INVALID_ARGVAL	次のいずれかです。 − src_offset または dest_offset < 1 − src_offset または dest_offset > LOBMAXSIZE − amount < 1 − amount > LOBMAXSIZE

例

```
CREATE OR REPLACE PROCEDURE Example_l2f IS
  lobd       BLOB;
  fils       BFILE   := BFILENAME('SOME_DIR_OBJ','some_file');
  amt        INTEGER := 4000;
BEGIN
  SELECT b_lob INTO lobd FROM lob_table WHERE key_value = 42 FOR UPDATE;
  dbms_lob.fileopen(fils, dbms_lob.file_readonly);
  dbms_lob.loadfromfile(lobd, fils, amt);
  COMMIT;
  dbms_lob.fileclose(fils);
```

LOADBLOBFROMFILE プロシージャ

このプロシージャは、BFILE のデータを内部 BLOB にロードします。結果は LOADFROMFILE と同じで、新しいオフセットを戻します。

ソースおよび宛先 LOB 両方に対するオフセット、およびソース BFILE からコピーするバイト数を指定できます。amount および src_offset は BFILE を参照するため、バイト単位で、dest_offset は BLOB の場合はバイト単位です。

宛先 LOB に指定したオフセットが、現在その LOB に格納されているデータの終わりを超えている場合は、宛先 BLOB に、0（ゼロ）バイトの FILLER または空白が挿入されます。オフセットが宛先 LOB の現行の長さより小さい場合、既存のデータは上書きされます。

入力 amount および offset を加算した値が BFILE 内のデータの長さを超えた場合はエラーが発生します（ただし、指定されている amount が、BFILE の終わりに達するまでロードするように指定できる LOBMAXSIZE の場合を除きます）。

構文

```
DBMS_LOB.LOADBLOBFROMFILE (
   dest_lob    IN OUT NOCOPY BLOB,
   src_bfile   IN            BFILE,
   amount      IN            INTEGER,
   dest_offset IN OUT        INTEGER,
   src_offset  IN OUT        INTEGER);
```

パラメータ

表 23-36 LOADBLOBFROMFILE プロシージャのパラメータ

パラメータ	説明
dest_lob	ロード先の BLOB ロケータ。
src_bfile	ロード元の BFILE ロケータ。
amount	BFILE からロードするバイト数。BFILE の終わりまでロードする DBMS_LOB.LOBMAXSIZE を使用することもできます。
dest_offset	(IN) 宛先 BLOB の書込み開始位置を示すオフセット（起点 : 1）。バイト数で指定します。(OUT) この書込み終了直後の宛先 BLOB の新規オフセット。バイト数で示されます。次の書込みの開始位置でもあります。
src_offset	(IN) ソース BFILE の読込み開始位置を示すオフセット（起点 : 1）。バイト数で指定します。(OUT) この読込み終了直後のソース BFILE のオフセット。バイト数で示されます。次の読込みの開始位置でもあります。

使用要件

OPEN/CLOSE 操作内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

OPEN/CLOSE 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

定数およびデフォルト

パラメータを省略する簡単な方法はありません。IN/OUT パラメータに変数を宣言するか、または IN パラメータにデフォルト値を指定します。使用可能な定数およびデフォルトを次に示します。

表 23-37 パラメータのデフォルト値

パラメータ	デフォルト値	説明
amount	DBMSLOB.LOBMAXSIZE (IN)	ファイル全体をロードします。
dest_offset	1 (IN)	先頭から開始します。
src_offset	1 (IN)	先頭から開始します。

DBMSLOB.SQL で定義される定数

```
lobmaxsize                      CONSTANT INTEGER        := 4294967295;
```

例外

表 23-38 LOADBLOBFROMFILE プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL または INVALID です。
INVALID_ARGVAL	次のいずれかです。 − src_offset または dest_offset < 1 − src_offset または dest_offset > LOBMAXSIZE − amount < 1 − amount > LOBMAXSIZE

例

```
TBD
;
```

LOADCLOBFROMFILE プロシージャ

このプロシージャは、BFILE のデータを必要なキャラクタ・セット変換を行った後、内部 CLOB/NCLOB にロードし、新しいオフセットを戻します。

ソースおよび宛先 LOB の両方に対するオフセット、およびソース BFILE からコピーするバイト数を指定できます。amount および src_offset は BFILE を参照するため、バイト単位で、dest_offset は CLOB の場合は文字単位です。

宛先 LOB に指定したオフセットが、現在その LOB に格納されているデータの終わりを超えている場合は、宛先 CLOB に、0（ゼロ）バイトの FILLER または空白が挿入されます。オフセットが宛先 LOB の現行の長さより小さい場合、既存のデータは上書きされます。

入力 amount および offset を加算した値が BFILE 内のデータの長さを超えた場合はエラーが発生します（ただし、指定されている amount が、BFILE の終わりに達するまでロードするように指定できる LOBMAXSIZE の場合を除きます）。

構文

```
DBMS_LOB.LOADCLOBFROMFILE (
   dest_lob       IN OUT NOCOPY   BLOB,
   src_bfile      IN              BFILE,
   amount         IN              INTEGER,
   dest_offset    IN OUT          INTEGER,
   src_offset     IN OUT          INTEGER,
   src_csid       IN              NUMBER,
   lang_context   IN OUT          INTEGER,
   warning        OUT             INTEGER);
```

パラメータ

表 23-39 LOADCLOBFROMFILE プロシージャのパラメータ

パラメータ	説明
dest_lob	ロード先の CLOB/NCLOB ロケータ。
src_bfile	ロード元の BFILE ロケータ。
amount	BFILE からロードするバイト数 BFILE の終わりまでロードする DBMS_LOB.LOBMAXSIZE を使用することもできます。
dest_offset	(IN) 宛先 CLOB の書込み開始位置を示すオフセット（起点 : 1)。文字数で指定します。(OUT) このロード終了直後の新規オフセット。文字数で示されます。次のロードの開始位置でもあります。常に、ロード終了後の最初の完全な文字の開始位置を示します。最後の文字が完全でない場合は、部分文字の先頭までオフセットが戻ります。
src_offset	(IN) ソース BFILE の読込み開始位置を示すオフセット（起点 : 1)。バイト数で指定します。(OUT) この読込み終了直後のソース BFILE のオフセット。バイト数で示されます。次の読込みの開始位置でもあります。
src_csid	ソース（BFILE）ファイルのキャラクタ・セット ID。
lang_context	(IN) 現在のロードに関する、シフト・ステータスなどの言語コンテキスト。(OUT) 現在のロードが停止した時点の言語コンテキスト。同じソースから継続してロードする場合に、次のロードで使用されます。この情報は、ユーザーがソース・データを失ったり、誤って解釈することなく継続してロードできるように戻されます。一番最初のロードの場合、または特に必要ない場合は、デフォルトの 0（ゼロ）を使用してください。この言語コンテキストの詳細は、ユーザーには隠されています。コールするときに、その内容を認識している必要はありません。

表 23-39 LOADCLOBFROMFILE プロシージャのパラメータ（続き）

パラメータ	説明
warning	(OUT) 警告メッセージ。ロード時になんらかの異常が発生したことを示します。ユーザーのミスが原因の場合と、それ以外の場合があります。ロードは要求に応じて完了され、警告メッセージをチェックするかどうかはユーザーが判断します。現在、発生する可能性があるのは、変換不可能な文字に関する警告のみです。この警告は、ソース内の文字が宛先の文字に正しく変換できず、かわりに、デフォルトの置換文字（例：'?'）が使用された場合に発生します。メッセージは、DBMSLOB で warn_inconvertable_char として定義されます。

使用要件

- 宛先のキャラクタ・セットは常に、CLOB の場合はデータベース・キャラクタ・セットおよび NCLOB の場合は各国語キャラクタ・セットと同じです。
- csid=0 はデフォルトの動作を示します。ソースの csid のかわりに、CLOB の場合はデータベース csid、NCLOB の場合は各国語 csid を使用します。この場合でも、可変幅の場合は変換が必要です。
- OPEN/CLOSE 操作内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

  OPEN/CLOSE 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

定数およびデフォルト

パラメータを省略する簡単な方法はありません。IN/OUT パラメータに変数を宣言するか、または IN パラメータにデフォルト値を指定します。使用可能な定数およびデフォルトを次に示します。

表 23-40 パラメータのデフォルト値

パラメータ	デフォルト値	説明
amount	DBMSLOB.LOBMAXSIZE (IN)	ファイル全体をロードします。
dest_offset	1 (IN)	先頭から開始します。
src_offset	1 (IN)	先頭から開始します。
csid	0 (IN)	デフォルトの csid。宛先の csid を使用します。
lang_context	0 (IN)	デフォルトの言語コンテキスト。
warning	0 (OUT)	警告メッセージは発生しません。

DBMSLOB.SQL で定義される定数

```
lobmaxsize                CONSTANT INTEGER       := 4294967295;
warn_inconvertible_char   CONSTANT INTEGER       := 1;
default_csid              CONSTANT INTEGER       := 0;
default_lang_ctx          CONSTANT INTEGER       := 0;
no_warning                CONSTANT INTEGER       := 0;
```

例外

表 23-41 LOADCLOBFROMFILE プロシージャの例外

例外	説明
VALUE_ERROR	入力パラメータのいずれかが NULL または INVALID です。
INVALID_ARGVAL	次のいずれかです。 − src_offset または dest_offset < 1 − src_offset または dest_offset > LOBMAXSIZE − amount < 1 − amount > LOBMAXSIZE

例

```
TBD
;
```

OPEN プロシージャ

このプロシージャは、指定されたモードで、（内部または外部）LOB をオープンします。有効なモードは読取り専用と読取り / 書込みです。同じ LOB を 2 回オープンするとエラーになります。

> **注意：** LOB が読取り専用モードでオープンされた場合は、その LOB に書込みを行おうとすると、エラーが戻されます。BFILE は、読取り専用モードでのみオープンできます。

Oracle8.0 では、定数 file_readonly は BFILE をオープンするときのみ有効なモードです。Oracle8*i* では、lob_readonly および lob_readwrite の 2 つの新規定数が DBMS_LOB パッケージに追加されました。

構文

```
DBMS_LOB.OPEN (
   lob_loc   IN OUT NOCOPY BLOB,
   open_mode IN            BINARY_INTEGER);

DBMS_LOB.OPEN (
   lob_loc   IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
   open_mode IN            BINARY_INTEGER);

DBMS_LOB.OPEN (
   file_loc  IN OUT NOCOPY BFILE,
   open_mode IN            BINARY_INTEGER := file_readonly);
```

パラメータ

表 23-42 OPEN プロシージャのパラメータ

パラメータ	説明
lob_loc	LOB ロケータ。
open_mode	オープンするモード。

使用上の注意

OPEN では、内部または外部 LOB のいずれの場合もサーバーへのラウンドトリップが必要です。内部 LOB の場合、OPEN は OPEN コールに依存しているその他のコードのトリガーとなります。外部 LOB（BFILE）の場合は、サーバー側の実際のオペレーティング・システム・ファイルがオープンされるため、ラウンドトリップが必要です。

オープン / クローズ API 内のすべての LOB 操作のラップは必須ではありません。ただし、LOB をオープンしている場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。クローズしない場合は、エラーが発生します。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

トランザクションによりオープンされた LOB をすべてクローズする前にトランザクションをコミットした場合、エラーが発生します。エラーが戻された場合 LOB のオープンは廃棄されますが、トランザクションのコミットは正常に行われます。したがって、そのトランザクションにおける LOB および非 LOB データの変更はすべてコミットされますが、ドメイン索引およびファンクションベースの索引は更新されません。このような場合は、LOB 列のファンクション索引およびドメイン索引を再構築する必要があります。

READ プロシージャ

このプロシージャは、LOB の一部を読み込み、LOB の先頭からの絶対オフセットから開始し、指定された量のデータを buffer パラメータに戻します。

実際に読み込まれたバイト数または文字数は、amount パラメータに戻されます。入力 offset が LOB の終わりを超えた位置を指している場合、amount は 0（ゼロ）に設定され、NO_DATA_FOUND 例外が発生します。

構文

```
DBMS_LOB.READ (
   lob_loc   IN             BLOB,
   amount    IN OUT  NOCOPY BINARY_INTEGER,
   offset    IN             INTEGER,
   buffer    OUT            RAW);

DBMS_LOB.READ (
   lob_loc   IN             CLOB CHARACTER SET ANY_CS,
   amount    IN OUT  NOCOPY BINARY_INTEGER,
   offset    IN             INTEGER,
   buffer    OUT            VARCHAR2 CHARACTER SET lob_loc%CHARSET);
```

```
DBMS_LOB.READ (
   file_loc   IN              BFILE,
   amount     IN OUT   NOCOPY BINARY_INTEGER,
   offset     IN              INTEGER,
   buffer     OUT             RAW);
```

パラメータ

表 23-43 READ プロシージャのパラメータ

パラメータ	説明
lob_loc	読み込む LOB のロケータ。
file_loc	検査する LOB のファイル・ロケータ。
amount	読み込む、または読み込まれたバイト数（BLOB の場合）または文字数（CLOB の場合）。
offset	LOB の先頭からのオフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。
buffer	読取り操作用の出力バッファ。

例外

表 23-44 READ プロシージャの例外

例外	説明
VALUE_ERROR	lob_loc、amount または offset パラメータのいずれかが NULL です。
INVALID_ARGVAL	次のいずれかです。 − amount < 1 − amount > MAXBUFSIZE − offset < 1 − offset > LOBMAXSIZE − amount（バイト数または文字数）が buffer の容量を超えています。
NO_DATA_FOUND	LOB の終わりに達し、LOB から読み込むバイトまたは文字がありません。amount の値は 0（ゼロ）です。

例外

表 23-45 BFILE に関する READ プロシージャの例外

例外	説明
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

使用上の注意

VARCHAR2 バッファの形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータのタイプが NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータのタイプが CLOB の場合、バッファには CHAR データが含まれる必要があります。

クライアントから DBMS_LOB.READ をコールすると（例 : SQL*Plus 内からの BEGIN/END ブロックでのコール）、戻されるバッファにはクライアントのキャラクタ・セットのデータが含まれます。Oracle は、バッファをユーザーに戻す前に、サーバーのキャラクタ・セットからクライアントのキャラクタ・セットに LOB 値を変換します。

例

```
CREATE OR REPLACE PROCEDURE Example_13a IS
    src_lob        BLOB;
    buffer         RAW(32767);
    amt            BINARY_INTEGER := 32767;
    pos            INTEGER := 2147483647;
BEGIN
    SELECT b_col INTO src_lob
        FROM lob_table
        WHERE key_value = 21;
    LOOP
        dbms_lob.read (src_lob, amt, pos, buffer);
        -- process the buffer
        pos := pos + amt;
    END LOOP;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('End of data');
END;
```

```
CREATE OR REPLACE PROCEDURE Example_13b IS
    fil BFILE;
    buf RAW(32767);
    amt BINARY_INTEGER := 32767;
    pos INTEGER := 2147483647;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 21;
    dbms_lob.fileopen(fil, dbms_lob.file_readonly);
    LOOP
        dbms_lob.read(fil, amt, pos, buf);
        -- process contents of buf
        pos := pos + amt;
    END LOOP;
    EXCEPTION
      WHEN NO_DATA_FOUND
      THEN
      BEGIN
        dbms_output.putline ('End of LOB value reached');
        dbms_lob.fileclose(fil);
      END;
END;
```

ストリーム I/O よりもブロック I/O の方がオペレーティング・システム上で効率的に実行される I/O の例

```
CREATE OR REPLACE PROCEDURE Example_13c IS
    fil BFILE;
    amt BINARY_INTEGER := 1024; -- or n x 1024 for reading n
    buf RAW(1024); -- blocks at a time
    tmpamt BINARY_INTEGER;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 99;
    dbms_lob.fileopen(fil, dbms_lob.file_readonly);
    LOOP
        dbms_lob.read(fil, amt, pos, buf);
        -- process contents of buf
        pos := pos + amt;
    END LOOP;
    EXCEPTION
        WHEN NO_DATA_FOUND
        THEN
            BEGIN
                dbms_output.putline ('End of data reached');
                dbms_lob.fileclose(fil);
            END;
END;
```

SUBSTR ファンクション

このファンクションは、LOB の先頭からの絶対 offset から開始し、LOB の amount バイトまたは文字を戻します。

固定幅の n バイトの CLOB の場合、SUBSTR に対する入力 amount の指定が（32767/n）を超えると、長さ（32767/n）の文字バッファまたは CLOB の長さのうち、小さい方が戻されます。可変幅キャラクタ・セットの CLOB の場合、n は 2 です。

構文

```
DBMS_LOB.SUBSTR (
   lob_loc     IN    BLOB,
   amount      IN    INTEGER := 32767,
   offset      IN    INTEGER := 1)
  RETURN RAW;

DBMS_LOB.SUBSTR (
   lob_loc     IN    CLOB   CHARACTER SET ANY_CS,
   amount      IN    INTEGER := 32767,
   offset      IN    INTEGER := 1)
  RETURN VARCHAR2 CHARACTER SET lob_loc%CHARSET;

DBMS_LOB.SUBSTR (
   file_loc     IN    BFILE,
   amount      IN    INTEGER := 32767,
   offset      IN    INTEGER := 1)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(SUBSTR, WNDS, WNPS, RNDS, RNPS);
```

パラメータ

表 23-46 SUBSTR ファンクションのパラメータ

パラメータ	説明
lob_loc	読み込む LOB のロケータ。
file_loc	検査する LOB のファイル・ロケータ。
amount	読み込むバイト数（BLOB の場合）または文字数（CLOB の場合）。
offset	LOB の先頭からのオフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。

戻り値

表 23-47 SUBSTR ファンクションの戻り値

戻り値	説明
RAW	ファンクション・オーバーロード。パラメータに BLOB または BFILE を使用します。
VARCHAR2	CLOB のバージョン。
NULL	次のいずれかです。 − 入力パラメータのいずれかが NULL です。 − amount < 1 − amount > 32767 − offset < 1 − offset > LOBMAXSIZE

例外

表 23-48 BFILE 操作に関する SUBSTR ファンクションの例外

例外	説明
UNOPENED_FILE	ファイルが入力ロケータを使用してオープンされていません。
NOEXIST_DIRECTORY	ディレクトリが存在しません。
NOPRIV_DIRECTORY	ディレクトリに対する権限がありません。
INVALID_DIRECTORY	ディレクトリがファイルのオープン後に無効となりました。
INVALID_OPERATION	ファイルが存在しないか、またはファイルのアクセス権限がありません。

使用上の注意

VARCHAR2 バッファの形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータのタイプが NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータのタイプが CLOB の場合、バッファには CHAR データが含まれる必要があります。

クライアントから DBMS_LOB.SUBSTR をコールすると（例 : SQL*Plus 内からの BEGIN/END ブロックでのコール）、戻されるバッファにはクライアントのキャラクタ・セットのデータが含まれます。Oracle は、バッファをユーザーに戻す前に、サーバーのキャラクタ・セットからクライアントのキャラクタ・セットに LOB 値を変換します。

例

```
CREATE OR REPLACE PROCEDURE Example_14a IS
    src_lob        CLOB;
    pos            INTEGER := 2147483647;
    buf            VARCHAR2(32000);
BEGIN
    SELECT c_lob INTO src_lob FROM lob_table
        WHERE key_value = 21;
    buf := DBMS_LOB.SUBSTR(src_lob, 32767, pos);
    -- process the data
END;

CREATE OR REPLACE PROCEDURE Example_14b IS
    fil BFILE;
    pos INTEGER := 2147483647;
    pattern RAW;
BEGIN
    SELECT f_lob INTO fil FROM lob_table WHERE key_value = 21;
    dbms_lob.fileopen(fil, dbms_lob.file_readonly);
    pattern := dbms_lob.substr(fil, 255, pos);
    dbms_lob.fileclose(fil);
END;
```

関連項目：

- 23-39 ページ「INSTR ファンクション」
- 23-53 ページ「READ プロシージャ」

TRIM プロシージャ

このプロシージャは、内部 LOB の値を newlen パラメータで指定された長さに切り捨てます。BLOB の場合はバイト数、CLOB の場合は文字数で長さを指定します。

注意： TRIM プロシージャは、LOB の長さを newlen パラメータで指定された値に減らします。

空の LOB に対して TRIM を実行すると、何も処理は行われずエラーは戻されません。newlen で指定した新しい長さが LOB のサイズよりも大きい場合は、例外が発生します。

構文

```
DBMS_LOB.TRIM (
   lob_loc         IN OUT  NOCOPY BLOB,
   newlen          IN              INTEGER);

DBMS_LOB.TRIM (
   lob_loc         IN OUT  NOCOPY CLOB CHARACTER SET ANY_CS,
   newlen          IN              INTEGER);
```

パラメータ

表 23-49 TRIM プロシージャのパラメータ

パラメータ	説明
lob_loc	長さを切り捨てる内部 LOB のロケータ。
newlen	切り捨て後の新しい LOB 値の長さ。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。

例外

表 23-50 TRIM プロシージャの例外

例外	説明
VALUE_ERROR	lob_loc が NULL です。
INVALID_ARGVAL	次のいずれかです。 − new_len < 0 − new_len > LOBMAXSIZE

使用上の注意

オープン / クローズ API 内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

オープン / クローズ API 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

例

```
CREATE OR REPLACE PROCEDURE Example_15 IS
    lob_loc        BLOB;
BEGIN
-- get the LOB locator
    SELECT b_col INTO lob_loc
        FROM lob_table
        WHERE key_value = 42 FOR UPDATE;
    dbms_lob.trim(lob_loc, 4000);
    COMMIT;
END;
```

関連項目：

- 23-26 ページ「ERASE プロシージャ」
- 23-64 ページ「WRITEAPPEND プロシージャ」

WRITE プロシージャ

このプロシージャは、LOB の先頭からの絶対オフセットから開始し、指定された量のデータを内部 LOB に書き込みます。データは、buffer パラメータから書き込まれます。

WRITE は、オフセット位置以降 LOB にすでに存在しているデータを、指定した長さ分置換（上書き）します。

入力 amount がバッファのデータより多い場合はエラーが発生します。入力 amount がバッファのデータより少ない場合は、その量のバイト数または文字数のみバッファから LOB に書き込まれます。指定したオフセットが現在その LOB に格納されているデータの終わりを超えている場合は、BLOB または CLOB に、0（ゼロ）バイトの FILLER または空白がそれぞれ挿入されます。

構文

```
DBMS_LOB.WRITE (
   lob_loc  IN OUT NOCOPY  BLOB,
   amount   IN             BINARY_INTEGER,
   offset   IN             INTEGER,
   buffer   IN             RAW);

DBMS_LOB.WRITE (
   lob_loc  IN OUT  NOCOPY CLOB   CHARACTER SET ANY_CS,
   amount   IN             BINARY_INTEGER,
   offset   IN             INTEGER,
   buffer   IN             VARCHAR2 CHARACTER SET lob_loc%CHARSET);
```

パラメータ

表 23-51 WRITE プロシージャのパラメータ

パラメータ	説明
lob_loc	書込み先の内部 LOB のロケータ。
amount	書き込む、または書き込まれたバイト数（BLOB の場合）または文字数（CLOB の場合）。
offset	LOB の先頭からの書込み操作のオフセット（起点 : 1）。BLOB の場合はバイト数、CLOB の場合は文字数で指定します。
buffer	書込み用の入力バッファ。

例外

表 23-52 WRITE プロシージャの例外

例外	説明
VALUE_ERROR	lob_loc、amount または offset パラメータのいずれかが NULL、範囲外または INVALID です。
INVALID_ARGVAL	次のいずれかです。 − amount < 1 − amount > MAXBUFSIZE − offset < 1 − offset > LOBMAXSIZE

使用上の注意

VARCHAR2 バッファの形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータのタイプが NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータのタイプが CLOB の場合、バッファには CHAR データが含まれる必要があります。

クライアントから DBMS_LOB.WRITE をコールするときは（例 : SQL*Plus 内からの BEGIN/END ブロック内でコール）、バッファにクライアントのキャラクタ・セットのデータが含まれている必要があります。Oracle は、バッファ・データを LOB に書き込む前に、クライアント側のキャラクタ・セットをサーバー側のキャラクタ・セットに変換します。

オープン / クローズ API 内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。

内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

オープン / クローズ API 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

例

```
CREATE OR REPLACE PROCEDURE Example_16 IS
    lob_loc         BLOB;
    buffer          RAW;
    amt             BINARY_INTEGER := 32767;
    pos             INTEGER := 2147483647;
    i               INTEGER;
BEGIN
    SELECT b_col INTO lob_loc
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
    FOR i IN 1..3 LOOP
        dbms_lob.write (lob_loc, amt, pos, buffer);
        -- fill in more data
        pos := pos + amt;
    END LOOP;
    EXCEPTION
        WHEN some_exception
        THEN handle_exception;
END;
```

関連項目：

- 23-16 ページ「APPEND プロシージャ」
- 23-22 ページ「COPY プロシージャ」

WRITEAPPEND プロシージャ

このプロシージャは、指定された量のデータを内部 LOB の後ろに書き込みます。データは、buffer パラメータから書き込まれます。

入力 amount がバッファのデータより多い場合はエラーが発生します。入力 amount がバッファのデータより少ない場合は、その量のバイト数または文字数のみバッファから LOB の終わりに書き込まれます。

構文

```
DBMS_LOB.WRITEAPPEND (
   lob_loc IN OUT NOCOPY BLOB,
   amount  IN            BINARY_INTEGER,
   buffer  IN            RAW);

DBMS_LOB.WRITEAPPEND (
   lob_loc IN OUT NOCOPY CLOB CHARACTER SET ANY_CS,
   amount  IN            BINARY_INTEGER,
   buffer  IN            VARCHAR2 CHARACTER SET lob_loc%CHARSET);
```

パラメータ

表 23-53 WRITEAPPEND プロシージャのパラメータ

パラメータ	説明
lob_loc	書込み先の内部 LOB のロケータ。
amount	書き込む、または書き込まれたバイト数（BLOB の場合）または文字数（CLOB の場合）。
buffer	書込み用の入力バッファ。

例外

表 23-54 WRITEAPPEND プロシージャの例外

例外	説明
VALUE_ERROR	lob_loc、amount または offset パラメータのいずれかが NULL、範囲外または INVALID です。
INVALID_ARGVAL	次のいずれかです。 − amount < 1 − amount > MAXBUFSIZE

使用上の注意

VARCHAR2 バッファの形式は、CLOB パラメータの形式と一致する必要があります。つまり、入力 LOB のパラメータのタイプが NCLOB の場合、バッファには NCHAR データが含まれる必要があります。これに対して、入力 LOB のパラメータのタイプが CLOB の場合、バッファには CHAR データが含まれる必要があります。

クライアントから DBMS_LOB.WRITEAPPEND をコールするときは（例 : SQL*Plus 内からの BEGIN/END ブロック内でコール）、バッファにクライアントのキャラクタ・セットのデータが含まれている必要があります。Oracle は、バッファ・データを LOB に書き込む前に、クライアント側のキャラクタ・セットをサーバー側のキャラクタ・セットに変換します。

オープン / クローズ API 内の LOB 操作のラップは必須ではありません。操作を実行する前に LOB をオープンしなかった場合、LOB 列のファンクション索引およびドメイン索引は、コール中に更新されます。ただし、操作を実行する前に LOB をオープンした場合は、トランザクションのコミットまたはロールバックを実行する前にクローズする必要があります。内部 LOB がクローズされるときに、LOB 列のファンクション索引およびドメイン索引が更新されます。

オープン / クローズ API 内の LOB 操作をラップしなかった場合、ファンクション索引およびドメイン索引は LOB へ書き込むたびに更新されます。これにより、パフォーマンスに悪影響が与えられる場合があります。したがって、LOB への書込み操作を OPEN または CLOSE 文の中に囲むことをお薦めします。

例

```
CREATE OR REPLACE PROCEDURE Example_17 IS
    lob_loc    BLOB;
    buffer     RAW;
    amt        BINARY_INTEGER := 32767;
    i          INTEGER;
BEGIN
    SELECT b_col INTO lob_loc
        FROM lob_table
        WHERE key_value = 12 FOR UPDATE;
    FOR i IN 1..3 LOOP
        -- fill the buffer with data to be written to the lob
        dbms_lob.writeappend (lob_loc, amt, buffer);
    END LOOP;
END;
```

関連項目 :

- 23-16 ページ「APPEND プロシージャ」
- 23-22 ページ「COPY プロシージャ」
- 23-61 ページ「WRITE プロシージャ」

24

DBMS_LOCK

アプリケーションに対する Oracle Lock Management サービスは、DBMS_LOCK パッケージにあるプロシージャを介して使用できます。特定モードのロックを要求したり、同一または別のインスタンスにある別のプロシージャ内で識別できる一意の名前をロックに付けたり、ロック・モードの変更およびロックの解放を行うことができます。

確保されているユーザー・ロックは Oracle Lock と同一であるため、デッドロック検出など、Oracle Lock の全機能を備えています。分散トランザクションで使用するユーザー・ロックが COMMIT 時に解放されることを確認してください。解放されない場合、デッドロックが検出されない可能性があります。

ユーザー・ロックは、接頭辞 UL で識別されているため、Oracle Lock と競合することはありません。これらのロックは、Oracle Enterprise Manager Lock Monitor Screen または適切な固定ビューを使用して表示できます。ユーザー・ロックは、セッションの終了時に自動的に解放されます。

ロック識別子は、0 ～ 1073741823 の番号です。

ユーザー・ロックを使用して、次のことができます。

- 端末などのデバイスに排他的アクセスを提供します。
- アプリケーションレベルの読込みロックを施行します。
- ロックの解放時期およびアプリケーション終了後のクリーンアップを検出します。
- アプリケーションを同期化し、順次処理を施行します。

この章では、次の項目について説明します。

DBMS_LOCK の要件、セキュリティおよび定数

要件

DBMS_LOCK は、セッション当たりのロック数を 200 ～ 300 に制限すると、最も効率的です。プロシージャ間で同一のロックを使用して競合が発生しないように、ロックの使用について標準規則を作成することをお薦めします。たとえば、ロック名の一部に会社名を含める方法があります。

セキュリティ

使用可能なロックの最大数について、オペレーティング・システム固有の制限がある場合があります。これは、ロックの使用時またはこのパッケージを他のユーザーに対して使用可能にするときに考慮する必要があります。特定のユーザーまたはロールに対してのみ EXECUTE 権限の付与を検討してください。

使用するロック数を制限するカバー・パッケージを作成してから特定のユーザーに EXECUTE 権限を付与することをお薦めします。カバー・パッケージの例は、DBMSLOCK.SQL パッケージ仕様部ファイルに記載されています。

定数

```
nl_mode  constant integer := 1;
ss_mode  constant integer := 2;       -- Also called 'Intended Share'
sx_mode  constant integer := 3;       -- Also called 'Intended Exclusive'
s_mode   constant integer := 4;
ssx_mode constant integer := 5;
x_mode   constant integer := 6;
```

いくつかのロック・モードがあります（NL -> 「NULL」、SS -> 「半共有（Sub Shared）」、SX -> 「半排他（Sub eXclusive）」、S -> 「共有（Shared）」、SSX -> 「共有半排他（Shared Sub eXclusive）」、X -> 「排他（eXclusive）」）。

半共有ロックを集計オブジェクトで使用して、共有ロックがオブジェクトのサブパーツに対して作動中であることを示すことができます。同様に、半排他ロックを集計オブジェクトで使用して、排他ロックがオブジェクトのサブパーツに対して作動中であることを示すことができます。共有半排他ロックは、集計オブジェクト全体で共有ロックを使用している一方、一部のサブパーツではさらに排他ロックを使用していることを示します。

ロックの互換性ルール

別のプロセスが保持状態のときに、取得を試みると次のようになります。

表 24-1 ロックの互換性

保持モード	NL の取得	SS の取得	SX の取得	S の取得	SSX の取得	X の取得
NL	成功	成功	成功	成功	成功	成功
SS	成功	成功	成功	成功	成功	失敗
SX	成功	成功	成功	失敗	失敗	失敗
S	成功	成功	失敗	成功	失敗	失敗
SSX	成功	成功	失敗	失敗	失敗	失敗
X	成功	失敗	失敗	失敗	失敗	失敗

```
maxwait  constant integer := 32767;
```

定数 maxwait は、無期限に待機します。

DBMS_LOCK サブプログラムの要約

表 24-2 DBMS_LOCK パッケージのサブプログラム

サブプログラム	説明
「ALLOCATE_UNIQUE プロシージャ」24-4 ページ	名前付きロックに一意のロック ID を割り当てます。
「REQUEST ファンクション」24-6 ページ	特定のモードのロックを要求します。
「CONVERT ファンクション」24-7 ページ	ロックを別のモードに変換します。
「RELEASE ファンクション」24-9 ページ	ロックを解放します。
「SLEEP プロシージャ」24-10 ページ	プロシージャを指定時間だけスリープさせます。

ALLOCATE_UNIQUE プロシージャ

このプロシージャは、一意のロック識別子（1073741824 ～ 1999999999 の範囲内）を指定のロック名に割り当てます。アプリケーションは、ロック識別子を使用してロックの使用方法を調整できます。これは、アプリケーションでロックの使用方法を調整するには、ロック番号よりロック名に基づいた方が容易になるためです。

ロックを名前で識別する場合は、ALLOCATE_UNIQUE を使用して、名前付きロックに一意のロック識別番号を生成できます。

新規ロック名で ALLOCATE_UNIQUE をコールする最初のセッションで、一意のロック ID が生成され、dbms_lock_allocated 表に格納されます。次のコール（通常は他のセッションによる）では、最初のセッションで生成されたロック ID が戻ります。

ロック名は、指定のロック名で ALLOCATE_UNIQUE を最後にコールした後、少なくとも expiration_secs（デフォルトは 10 日間）の間は、戻されたロック ID に関連付けられています。この期間を経過すると、このロック名の dbms_lock_allocated 表にある行は、領域リカバリのために削除される場合があります。ALLOCATE_UNIQUE プロシージャはコミットを実行します。

注意： Oracle では、SQL を使用して指定の名前に関連付けられたロックを判断するため、名前付きユーザー・ロックを使用すると効率が悪くなる場合があります。

構文

```
DBMS_LOCK.ALLOCATE_UNIQUE (
   lockname         IN  VARCHAR2,
   lockhandle       OUT VARCHAR2,
   expiration_secs  IN  INTEGER   DEFAULT 864000);
```

パラメータ

表 24-3 ALLOCATE_UNIQUE プロシージャのパラメータ

パラメータ	説明
lockname	一意の ID を生成するロックの名前。 ORA$ で始まるロック名は使用しないでください。これはオラクル社の製品用に予約されています。
lockhandle	ALLOCATE_UNIQUE が生成したロック ID にハンドルを戻します。 このハンドルは、REQUEST、CONVERT および RELEASE への次のコールで使用できます。 ハンドルは実際のロック ID のかわりに戻され、プログラミング・エラーにより不適切であるが有効なロック ID が作成されないようにします。これにより、このパッケージを使用するアプリケーション間の独立性が高まります。 LOCKHANDLE は、VARCHAR2（128）まで可能です。 ALLOCATE_UNIQUE が同じロック名で戻したロック・ハンドルを使用しているすべてのセッションは、同じロックを参照します。したがって、ロック・ハンドルを別のセッションに渡さないでください。
expiration_specs	指定のロックに最後の ALLOCATE_UNIQUE を実行した後、DBMS_LOCK_ALLOCATED 表からロックの削除を許可するまで待機する秒数。 デフォルトの待機期間は 10 日間です。この表からロックを削除しないでください。ALLOCATE_UNIQUE への次のコールで、期限切れのロックを削除し、領域をリカバリできます。

エラー

ORA-20000,ORU-10003:dbms_lock_allocated カタログに <lockname> ロックを挿入または検出できません。

REQUEST ファンクション

このファンクションは、指定のモードのロックを要求します。REQUEST はオーバーロードされたファンクションで、ユーザー定義のロック識別子または ALLOCATE_UNIQUE プロシージャが戻すロック・ハンドルのいずれかを受け入れます。

構文

```
DBMS_LOCK.REQUEST(
   id                 IN  INTEGER ||
   lockhandle         IN  VARCHAR2,
   lockmode           IN  INTEGER DEFAULT X_MODE,
   timeout            IN  INTEGER DEFAULT MAXWAIT,
   release_on_commit  IN  BOOLEAN DEFAULT FALSE,
  RETURN INTEGER;
```

X_MODE や MAXWAIT などの現行のデフォルト値は、DBMS_LOCK パッケージ仕様部で定義されています。

パラメータ

表 24-4 REQUEST ファンクションのパラメータ

パラメータ	説明
id または lockhandle	変更するロック・モードのユーザー割当てロック識別子（0 ～ 1073741823）または ALLOCATE_UNIQUE が戻すロック・ハンドル。
lockmode	要求するロックのモード。 使用可能なモードおよび関連する整数の識別子は次のとおりです。カッコ内の略称は、V$ ビューおよび Oracle Enterprise Manager Monitors に表示されるときのロックの略称です。 1 − NULL モード 2 −行共有モード（ULRS） 3 −行排他モード（ULRX） 4 −共有モード（ULS） 5 −共有行排他モード（ULRSX） 6 −排他モード（ULX）
timeout	ロックの付与を待つ秒数。 この時間内にロックが付与できない場合、コールは値 1（timeout）を戻します。

表 24-4 REQUEST ファンクションのパラメータ（続き）

パラメータ	説明
release_on_commit	このパラメータを TRUE に設定すると、ロックをコミットまたはロールバック時に解放します。 そうでない場合は、ロックが明示的に解放されるかセッションが終了するまで、ロックは解放されません。

戻り値

表 24-5 REQUEST ファンクションの戻り値

戻り値	説明
0	成功。
1	タイムアウト。
2	デッドロック。
3	パラメータ・エラー。
4	id または lockhandle で指定されたロックをすでに所有しています。
5	不正なロック・ハンドル。

CONVERT ファンクション

このファンクションは、ロックを別のモードに変換します。CONVERT はオーバーロードされたファンクションで、ユーザー定義のロック識別子または ALLOCATE_UNIQUE プロシージャが戻すロック・ハンドルのいずれかを受け入れます。

構文

```
DBMS_LOCK.CONVERT(
   id         IN INTEGER ||
   lockhandle IN VARCHAR2,
   lockmode   IN INTEGER,
   timeout    IN NUMBER DEFAULT MAXWAIT)
  RETURN INTEGER;
```

パラメータ

表 24-6 CONVERT ファンクションのパラメータ

パラメータ	説明
id または lockhandle	変更するロック・モードのユーザー割当てロック識別子（0 ～ 1073741823）または ALLOCATE_UNIQUE が戻すロック・ハンドル。
lockmode	指定のロックに割り当てる新規モード。 使用可能なモードおよび関連する整数の識別子は次のとおりです。カッコ内の略称は、V$ ビューおよび Oracle Enterprise Manager Monitors に表示されるときのロックの略称です。 1 － NULL モード 2 －行共有モード（ULRS） 3 －行排他モード（ULRX） 4 －共有モード（ULS） 5 －共有行排他モード（ULRSX） 6 －排他モード（ULX）
timeout	ロック・モードの変更を待つ秒数。 この時間内にロックを変換できない場合、コールは値 1（timeout）を戻します。

戻り値

表 24-7 CONVERT ファンクションの戻り値

戻り値	説明
0	成功。
1	タイムアウト。
2	デッドロック。
3	パラメータ・エラー。
4	id または lockhandle が指定するロックを所有していません。
5	不正なロック・ハンドル。

RELEASE ファンクション

このファンクションは、REQUEST ファンクションを使用して前に取得したロックを明示的に解放します。ロックは、セッションの終了時に自動的に解放されます。RELEASE はオーバーロードされたファンクションで、ユーザー定義のロック識別子または ALLOCATE_UNIQUE プロシージャが戻すロック・ハンドルのいずれかを受け入れます。

構文

```
DBMS_LOCK.RELEASE (
   id          IN INTEGER)
  RETURN INTEGER;

DBMS_LOCK.RELEASE (
   lockhandle IN VARCHAR2)
  RETURN INTEGER;
```

パラメータ

表 24-8 RELEASE ファンクションのパラメータ

パラメータ	説明
id または lockhandle	変更するロック・モードのユーザー割当てロック識別子（0 ～ 1073741823）または ALLOCATE_UNIQUE が戻すロック・ハンドル。

戻り値

表 24-9 RELEASE ファンクションの戻り値

戻り値	説明
0	成功。
3	パラメータ・エラー。
4	id または lockhandle が指定するロックを所有していません。
5	不正なロック・ハンドル。

SLEEP プロシージャ

このプロシージャは、指定時間だけセッションを中断します。

構文

```
DBMS_LOCK.SLEEP (
   seconds  IN NUMBER);
```

パラメータ

表 24-10 SLEEP プロシージャのパラメータ

パラメータ	説明
seconds	セッションを中断する時間（秒）。 入力できる最小増分値は 100 分の 1 秒です。たとえば、1.95 は有効な時間値です。

例 : 小切手の印刷

次の Pro*COBOL プリコンパイラの例では、複数のユーザーが単一のデバイスにアクセスするとき、ロックを使用して競合が発生しないようにする方法を示します。DBMS_LOCK パッケージを使用して、排他的アクセスを確保します。

レジ係が顧客の返品に対して返金を行うとします。50 ドル未満の返金は現金で、それ以上の場合は小切手で返金します。次のコードにより、小切手を印刷します。小切手の印刷のたびにプリンタをオープンしてクローズする負担を避けるため、1 台のプリンタはすべてのレジ係に対してオープンしています。このため、プリンタへの排他的アクセスを確保しないと、複数のレジ係からの出力ラインがインターリーブとなる可能性があります。

CHECK-PRINT

プリンタ・ロックに対するロック「ハンドル」を取得します。

```
MOVE "CHECKPRINT" TO LOCKNAME-ARR.
MOVE 10 TO LOCKNAME-LEN.
EXEC SQL EXECUTE
   BEGIN DBMS_LOCK.ALLOCATE_UNIQUE ( :LOCKNAME, :LOCKHANDLE );
   END; END-EXEC.
```

プリンタを排他モード（デフォルトのモード）にロックします。

```
EXEC SQL EXECUTE
   BEGIN DBMS_LOCK.REQUEST ( :LOCKHANDLE );
   END; END-EXEC.
```

これでプリンタを排他的に使用できるため、小切手を印刷します。

```
...
```

プリンタのロックを解除して、他のユーザーが使用できるようにします。

```
EXEC SQL EXECUTE
   BEGIN DBMS_LOCK.RELEASE ( :LOCKHANDLE );

   END; END-EXEC.
```

25

DBMS_LOGMNR

LogMiner を使用すると、実際のデータ値に基づいた問合せを実行できます。たとえば、表 scott.emp に対するすべての更新、またはユーザー scott が実行したすべての削除を検索する問合せを発行できます。また、sal が一定量より増加した scott.emp に対する更新をすべて表示するような問合せを実行することも可能です。このようなデータを使用して、システムの動作を分析、および監査タスクを実行できます。

DBMS_LOGMNR パッケージには、LogMiner ツールの初期化に使用するプロシージャが含まれています。これらのプロシージャを使用して、分析対象の REDO ログのリストを作成し、目的の SCN または時間範囲を指定します。このプロシージャを完了すると、サーバーは、V$LOGMNR_CONTENTS ビューに対して SQL SELECT を処理する用意ができます。

> **関連項目：** LogMiner の使用方法は、『Oracle9*i* データベース管理者ガイド』を参照してください。

この章では、次の項目について説明します。

DBMS_LOGMNR の定数

表 25-1 に、DBMS_LOGMNR パッケージの ADD_LOGFILE オプション・フラグの定数を示します。

表 25-1 ADD_LOGFILE オプション・フラグの定数

定数	説明
NEW	DBMS_LOGMNR.NEW は、REDO ログの既存リストがある場合、これをパージします。指定された REDO ログを分析対象の REDO ログ・リストに配置します。
ADDFILE	DBMS_LOGMNR.ADDFILE は、指定された REDO ログを分析対象の REDO ログ・リストに追加します。重複するファイルを追加しようとすると、例外（ORA-1289）が発生します。
REMOVEFILE	DBMS_LOGMNR.REMOVEFILE は、指定した REDO ログを分析対象の REDO ログ・リストから削除します。事前に追加されていないファイルを削除しようとすると、例外（ORA-1290）が発生します。

表 25-2 に、DBMS_LOGMNR パッケージの START_LOGMNR オプション・フラグの定数を示します。

表 25-2 START_LOGMNR オプション・フラグの定数

定数	説明
COMMITTED_DATA_ONLY	このオプションを設定すると、コミットされたトランザクションに対応する DML のみが戻されます。コミットされたトランザクションに対応する DML は、グループ化されます。トランザクションは、コミットされた順序で戻されます。このオプションが設定されていない場合は、全トランザクション（コミット、ロールバックおよび処理中）のすべての行が戻されます。
SKIP_CORRUPTION	V$LOGMNR_CONTENTS からの SELECT 操作で、分析中の REDO ログの破損をスキップして処理を続行するように指示します。このオプションは、REDO ログのブロック（REDO ログのヘッダーは除く）が破損している場合のみ機能します。コール元は、V$LOGMNR_CONTENTS ビューの INFO 列をチェックして、LogMiner がスキップした破損ブロックを確認する必要があります。
DDL_DICT_TRACKING	使用中のディクショナリがフラット・ファイルであるか、REDO ログに格納されている場合、LogMiner では、DDL イベントが発生すると内部ディクショナリが更新されることが確認されます。これにより、LogMiner ディクショナリが作成された後に変更されたオブジェクトに対し、SQL_REDO および SQL_UNDO の正しい情報が保持されます。 このオプションは、DICT_FROM_ONLINE_CATALOG オプションとともに使用できません。

表 25-2 START_LOGMNR オプション・フラグの定数（続き）

定数	説明
DICT_FROM_ONLINE_CATALOG	フラット・ファイルまたは REDO ログに含まれたディクショナリのスナップショットではなく、現行のデータベース・ディクショナリを使用するように LogMiner に指示します。 このオプションは、DDL_DICT_TRACKING オプションと一緒に指定できません。
DICT_FROM_REDO_LOGS	このオプションを設定すると、LogMiner は DBMS_LOGMNR.ADD_LOGFILE プロシージャで指定された REDO ログからディクショナリを検索しようとします。
NO_SQL_DELIMITER	このオプションを設定すると、再構築された SQL 文の終わりに SQL デリミタ（セミコロン）が付加されません。
PRINT_PRETTY_SQL	このオプションを設定すると、LogMiner は再構築された SQL 文を読み易いようにフォーマットします。
CONTINUOUS_MINE	このオプションを設定すると、1 つのアーカイブ REDO ログのみ登録する必要があります。LogMiner は、これ以降のアーカイブ REDO ログおよびオンライン・カタログについて、追加および取出しを自動的に行います。これは、REDO ログを生成している同じインスタンスで取出しを行う場合に便利です。

REDO ログからのデータ値の抽出

LogMiner における REDO ログからのデータ抽出は、DBMS_LOGMNR.MINE_VALUE および DBMS_LOGMNR.COLUMN_PRESENT の 2 つのファンクションを使用して実行されます。これらのファンクションについては、この章の後半で説明します。

DBMS_LOGMNR の使用例

次の例では、DBMS_LOGMNR プロシージャを使用して LogMiner セッションに REDO ログを追加する方法、LogMiner を（フラット・ファイル・ディクショナリで）起動する方法、V$LOGMNR_CONTENTS からの選択操作を実行する方法および LogMiner セッションを終了する方法を示します。DBMS_LOGMNR プロシージャの詳細な説明は、25-5 ページの「DBMS_LOGMNR サブプログラムの要約」を参照してください。

```
SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
  2 LogFileName => '/oracle/logs/log1.f', -
  3 Options => dbms_logmnr.NEW);

SQL> EXECUTE DBMS_LOGMNR.ADD_LOGFILE( -
  2 LogFileName => '/oracle/logs/log2.f', -
  3 Options => dbms_logmnr.ADDFILE);

SQL> EXECUTE DBMS_LOGMNR.START_LOGMNR( -
  2 DictFileName =>'/oracle/dictionary.ora');

SQL> SELECT sql_redo
  2 FROM V$LOGMNR_CONTENTS

SQL> EXECUTE DBMS_LOGMNR.END_LOGMNR();
```

DBMS_LOGMNR サブプログラムの要約

表 25-3 に、DBMS_LOGMNR パッケージのプロシージャを説明します。

表 25-3 DBMS_LOGMNR パッケージのサブプログラム

サブプログラム	説明
「ADD_LOGFILE プロシージャ」25-5 ページ	ファイルを既存または新規作成した処理対象のアーカイブ・ファイルのリストに追加します。
「START_LOGMNR プロシージャ」25-7 ページ	LogMiner ユーティリティを初期化します。
「END_LOGMNR プロシージャ」25-9 ページ	LogMiner セッションを終了します。
「MINE_VALUE ファンクション」25-10 ページ	このファンクションを V$LOGMNR_CONTENTS から戻された行に対してコールし、このファンクションに対する column_name 入力パラメータにより指定された列の UNDO または REDO 列値を取得できます。
「COLUMN_PRESENT ファンクション」25-11 ページ	このファンクションを V$LOGMNR_CONTENTS から戻された行に対してコールし、このファンクションに対する column_name 入力パラメータにより指定された列の UNDO または REDO 列値が存在するかどうかを判別できます。

ADD_LOGFILE プロシージャ

このプロシージャは、ファイルを既存または新規作成した処理対象のアーカイブ・ファイルのリストに追加します。

V$LOGMNR_CONTENTS ビューから情報を選択するために、LogMiner セッションは分析対象の REDO ログに関する情報を使用して設定する必要があります。ADD_LOGFILE プロシージャを使用して、分析する REDO ログのリストを指定してください。

> **注意：** 複数の REDO ログを分析する場合は、REDO ログごとに ADD_LOGFILE プロシージャを別々にコールする必要があります。

構文

```
DBMS_LOGMNR.ADD_LOGFILE(
   LogFileName     IN VARCHAR2,
   Options         IN BINARY_INTEGER default ADDFILE );
```

パラメータ

表 25-4 に、ADD_LOGFILE プロシージャのパラメータを示します。

表 25-4 ADD_LOGFILE プロシージャのパラメータ

パラメータ	説明
LogFileName	このセッションで分析する REDO ログ・リストに追加する必要のある REDO ログの名前。
Options	次のいずれかです。 － 新規リストを開始します（DBMS_LOGMNR.NEW）。 － 既存のリストにファイルを追加します（DBMS_LOGMNR.ADDFILE）。 － REDO ログを削除します（DBMS_LOGMNR.REMOVEFILE）。 表 25-1「ADD_LOGFILE オプション・フラグの定数」を参照してください。

例外

- ORA-1284: 指定した REDO ログ・ファイルをオープンできません。ログ・ファイルまたはディレクトリが存在していないか、またはアクセスできません。
- ORA-1285: REDO ログ・ファイルのヘッダーの読込みエラーです。
- ORA-1286: 指定した REDO ログ・ファイルは、分析用に追加した他のログ・ファイルを生成したデータベースからのファイルではありません。
- ORA-1287: 指定した REDO ログ・ファイルは、別のデータベース・インカネーションによるファイルです。
- ORA-1289: 指定した REDO ログ・ファイルは、以前に指定したログ・ファイルと重複しています。
- ORA-1290: 削除するように指定した REDO ログ・ファイルは、登録されていません。
- ORA-1337: 指定された REDO ログ・ファイルの互換性バージョンは、追加された他のログ・ファイルと異なります。

START_LOGMNR プロシージャ

このプロシージャは、LogMiner セッションを開始します。

> **注意：** ADD_LOGFILE プロシージャを使用して、分析する REDO ログのリストをあらかじめ指定していない場合、このプロシージャは失敗します。

構文

```
DBMS_LOGMNR.START_LOGMNR(
   startScn            IN NUMBER default 0,
   endScn              IN NUMBER default 0,
   startTime           IN DATE default '01-jan-1988',
   endTime             IN DATE default '01-jan-2988',
   DictFileName        IN VARCHAR2 default '',
   Options             IN BINARY_INTEGER default 0 );
```

パラメータ

表 25-5 に、DBMS_LOGMNR.START_LOGMNR プロシージャのパラメータを示します。

表 25-5 START_LOGMNR プロシージャのパラメータ

パラメータ	説明
startScn	指定した startSCN 以上の SCN を持つ REDO レコードのみを処理対象にします。SCN 範囲（V$LOGMNR_LOGS ビューに表示される REDO ログに関連する LOW_SCN および NEXT_SCN）に startScn を含む REDO ログがない場合、このプロシージャは失敗します。
endScn	指定した endSCN 以下の SCN を持つ REDO レコードのみを処理対象にします。SCN 範囲（V$LOGMNR_LOGS ビューに表示される REDO ログに関連する LOW_SCN および NEXT_SCN）に endScn を含む REDO ログがない場合、このプロシージャは失敗します。
startTime	指定した startTime 以上のタイムスタンプを持つ REDO レコードのみを処理対象にします。時間範囲（V$LOGMNR_LOGS ビューに表示される REDO ログに関連する LOW_TIME および HIGH_TIME）に startTime を含む REDO ログがない場合、このプロシージャは失敗します。startScn が指定されている場合、このパラメータは無視されます。

表 25-5 START_LOGMNR プロシージャのパラメータ（続き）

パラメータ	説明
endTime	指定した endTime 以下のタイムスタンプを持つ REDO レコードのみを処理対象にします。時間範囲（V$LOGMNR_LOGS ビューに表示される REDO ログに関連する LOW_TIME および HIGH_TIME）に endTime を含む REDO ログがない場合、このプロシージャは失敗します。endScn が指定されている場合、このパラメータは無視されます。
DictFileName	このフラット・ファイルには、データベース・カタログのスナップショットが含まれています。V$LOGMNR_CONTENTS の SQL_REDO および SQL_UNDO 列の再構築に使用されます。また、SEG_NAME、SEG_OWNER、SEG_TYPE_NAME および TABLE_SPACE 列の変換にも使用されます。ディクショナリ・ファイルの絶対パス名を指定する必要があります（このファイルは、DBMS_LOGMNR_D.BUILD プロシージャですでに作成されている必要があります）。 DICT_FROM_REDO_LOGS または DICT_FROM_ONLINE_CATALOG のいずれも指定されていない場合、指定する必要があるのはこのパラメータのみです。
Options	表 25-2「START_LOGMNR オプション・フラグの定数」を参照してください。

START_LOGMNR プロシージャを実行した後、次のビューを使用できるようになります。

- V$LOGMNR_CONTENTS － REDO ログの情報の履歴を含みます。
- V$LOGMNR_DICTIONARY －ディクショナリ・ファイルに関する現在の情報を含みます。
- V$LOGMNR_LOGS －分析する REDO ログに関する情報を含みます。
- V$LOGMNR_PARAMETERS － LogMiner セッションに関する情報を含みます。

例外

- ORA-1280: プロシージャは、LogMiner で内部エラーが発生すると、この例外で失敗します。
- ORA-1281: endScn が startScn 未満です。
- ORA-1282: endDate が startDate より前の日付です。
- ORA-1283: 指定したオプションは無効です。
- ORA-1292: REDO ログ・ファイルが LogMiner に登録されていません。
- ORA-1293: このプロシージャは、次の理由により例外が発生して失敗します。

1. 指定した startScn を含む範囲（LOW_SCN、NEXT_SCN）を持つログ・ファイルがない場合。
2. 指定した endScn を含む範囲（LOW_SCN、NEXT_SCN）を持つログ・ファイルがない場合。
3. 指定した startTime を含む範囲（LOW_TIME、HIGH_TIME）を持つログ・ファイルがない場合。
4. 指定した endTime を含む範囲（LOW_TIME、HIGH_TIME）を持つログ・ファイルがない場合。

- ORA-1294: 指定したディクショナリ・ファイルが破損しています。
- ORA-1295: 指定したディクショナリが、分析対象のログ・ファイルを生成した同じデータベースに対応していません。
- ORA-1296: データ・ディクショナリに指定されているキャラクタ・セットが、マイニング・データベースのキャラクタ・セットと一致せず、互換性がありません。
- ORA-1297: ディクショナリと登録 REDO ログ・ファイルで REDO バージョンが一致しません。
- ORA-1299: 指定したディクショナリは、別のデータベース・インカネーションによるファイルです。
- ORA-1300: ディクショナリの使用可能なスレッド・ビット・ベクトルが、REDO ログ・ファイルと一致しません。すべての REDO スレッドが LogMiner に登録されているわけではありません。

END_LOGMNR プロシージャ

このプロシージャは、LogMiner セッションを終了します。このプロシージャは他の方法で実行できないクリーンアップ操作を実行するため、このプロシージャにより LogMiner セッションを適切に終了する必要があります。

構文

```
DBMS_LOGMNR.END_LOGMNR;
```

パラメータ

なし

例外

- ORA-1307: Logminer セッションはアクティブではありません。END_LOGMNR プロシージャが、ログ・ファイルの追加なしにコールされました。

MINE_VALUE ファンクション

MINE_VALUE ファンクションは、2 つの引数を受け入れます。最初の引数は、データの REDO（REDO_VALUE）部分を取り出すか、または UNDO（UNDO_VALUE）部分を取り出すかを指定します。2 番目の引数は、取り出す列の完全修飾名を指定する文字列です。MINE_VALUE ファンクションは常に、元のデータ・タイプに再変換できる文字列を戻します。

構文

```
dbms_logmnr.mine_value(
     sql_redo_undo      IN  RAW,
     column_name        IN  VARCHAR2 default '') RETURN VARCHAR2;
```

パラメータ

表 25-6 に、MINE_VALUE ファンクションのパラメータを示します。

表 25-6 MINE_VALUE ファンクションのパラメータ

パラメータ	説明
sql_redo_undo	データ値を抽出する V$LOGMNR_CONTENTS の列。このパラメータは、表の複数の列に対応する値を含む、自己記述型のレコードとみなすこともできます。
column_name	このファンクションが情報を戻す対象となる列の完全修飾名（schema.table.column）。

戻り値

表 25-7 に、MINE_VALUE ファンクションの戻り値を示します。

表 25-7 MINE_VALUE ファンクションの戻り値

戻り値	説明
NULL	列は自己記述型のレコードに含まれていないか、列値が NULL です。
非 NULL	列は自己記述型のレコードに含まれています。値は文字列形式で戻されます。

例外

- ORA-1302: 指定した表または列が存在しません。

使用上の注意

- MINE_VALUE ファンクションを使用するには、LogMiner が正常に開始されている必要があります。
- MINE_VALUE ファンクションは、V$LOGMNR_CONTENTS ビューの選択操作のコンテキストで起動する必要があります。
- MINE_VALUE ファンクションは、LONG、LOB、ADT または COLLECTION データ・タイプをサポートしていません。
- 列引数のタイプが DATE の場合、戻される文字列は、現行のセッションの日付書式に関係なく、標準的なフォーム（DD-MON-YYYY HH24:MI:SS.SS）でフォーマットされます。

COLUMN_PRESENT ファンクション

このファンクションは、MINE_VALUE ファンクションとともに使用します。

MINE_VALUE ファンクションが NULL 値を戻した場合は、次のいずれかの可能性があります。

- 指定した列が、データの REDO または UNDO 部分に存在していません。
- 指定列は存在しますが、NULL 値です。

この 2 つを区別するには、COLUMN_PRESENT ファンクションを使用します。データの REDO または UNDO 部分に列が存在する場合は、このファンクションで 1 が戻されます。それ以外の場合は 0（ゼロ）が戻されます。

構文

```
dbms_logmnr.column_present(
     sql_redo_undo      IN  RAW,
     column_name        IN  VARCHAR2 default '') RETURN NUMBER;
```

パラメータ

表 25-8 に、COLUMN_PRESENT ファンクションのパラメータを示します。

表 25-8 COLUMN_PRESENT ファンクションのパラメータ

パラメータ	説明
sql_redo_undo	データ値を抽出する V$LOGMNR_CONTENTS の列。このパラメータは、テーブルの複数の列に対応する値を含む、自己記述型のレコードとみなすこともできます。
column_name	このファンクションが情報を戻す対象となる列の完全修飾名（schema.table.column）。

戻り値

表 25-9 に、COLUMN_PRESENT ファンクションの戻り値を示します。

表 25-9 COLUMN_PRESENT ファンクションの戻り値

戻り値	説明
0	指定された列が V$LOGMNR_CONTENTS のこの行にありません。
1	列が V$LOGMNR_CONTENTS のこの行にあります。 自己記述型のレコード（最初のパラメータ）が 2 番目のパラメータで指定した列を含んでいる場合は、1 を戻します。これを使用して、DBMS_LOGMNR.MINE_VALUE ファンクションからの NULL の戻りを識別できます。

例外

- ORA-1302: 指定した表または列が存在しません。

使用上の注意

- COLUMN_PRESENT ファンクションを使用するには、LogMiner が正常に開始されている必要があります。
- COLUMN_PRESENT ファンクションは、V$LOGMNR_CONTENTS ビューの選択操作のコンテキストで起動する必要があります。
- COLUMN_PRESENT ファンクションは、LONG、LOB、ADT または COLLECTION データ・タイプをサポートしていません。
- 列引数のタイプが DATE の場合、戻される文字列は、現行のセッションの日付書式に関係なく、標準的なフォーム（DD-MON-YYYY HH24:MI:SS.SS）でフォーマットされます。

26

DBMS_LOGMNR_CDC_PUBLISH

Oracle チェンジ・データ・キャプチャは、リレーショナル表に追加、変更または削除され、データを識別し、アプリケーションによる使用が可能な形式で変更後のデータを公開します。

この章では、DBMS_LOGMNR_CDC_PUBLISH パッケージを使用して Oracle チェンジ・データ・キャプチャ・システムを設定し、1 つ以上の Oracle のリレーショナル・ソース表からデータを獲得および公開する方法を説明します。チェンジ・データ・キャプチャが獲得および公開するのは、コミットされたデータのみです。

通常、チェンジ・データ・キャプチャ・システムには 1 人の発行者が設定され、Oracle のソース（リレーショナル）表に対する変更の獲得および発行はすべて、この発行者が行います。発行者はサブスクライバ（通常はアプリケーション）に対し、発行されたデータへのアクセス権を付与します。

> **関連項目：** Oracle チェンジ・データ・キャプチャにおけるモデルの発行およびサブスクライブの情報は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

この章では、次の項目について説明します。

変更データの公開

発行者（通常はデータベース管理者）は主に、データのソースおよびチェンジ・データ・キャプチャ・システムの構造（変更ソース、変更セットおよび変更表）を記述するスキーマ・オブジェクトの作成に関与します。

大部分のチェンジ・データ・キャプチャ・システムには、1 人の発行者および多数のサブスクライバが設定されています。発行者は、次の主な目的を実現します。

1. 発行が必要なソース表の変更を決定します。
2. DBMS_LOGMNR_CDC_PUBLISH パッケージのプロシージャを使用して変更ソース、変更セットおよび変更表オブジェクトを作成および管理することにより、変更データを獲得してソース表から使用できるようにします。
3. SQL GRANT および REVOKE 文を使用してサブスクライバへユーザーおよびロールの変更表に対する SELECT 権限の付与および取消しを行うことによりアクセスを管理します。

   これは、サブスクライバ（通常はアプリケーション）に対し、DBMS_LOGMNR_CDC_SUBSCRIBE プロシージャの使用を許可するために必要です。このプロシージャは、変更データをサブスクライブするために使用します。

DBMS_LOGMNR_CDC_PUBLISH サブプログラムの要約

発行者は DBMS_LOGMNR_CDC_PUBLISH パッケージを使用して、変更ソース、変更セットおよび変更表を作成およびメンテナンスし、それらが不要になった場合は削除します。

注意： DBMS_LOGMNR_CDC_PUBLISH パッケージを使用するには、EXECUTE_CATALOG_ROLE 権限を付与されている必要があります。また、すべてのビューを参照するには、SELECT_CATALOG_ROLE 権限が必要です。

表 26-1 に、DBMS_LOGMNR_CDC_PUBLISH パッケージのプロシージャを説明します。

表 26-1 DBMS_LOGMNR_CDC_PUBLISH パッケージのサブプログラム

サブプログラム	説明
「CREATE_CHANGE_TABLE プロシージャ」26-3 ページ	指定されたスキーマに変更表を作成し、対応するチェンジ・データ・キャプチャのメタデータを作成します。
「ALTER_CHANGE_TABLE プロシージャ」26-8 ページ	既存の変更テーブルの列を追加または削除します。または、既存の変更表のプロパティを変更します。
「DROP_SUBSCRIBER_VIEW プロシージャ」26-12 ページ	発行者がサブスクライバのスキーマからサブスクライバ・ビューを削除します。削除するビューは、事前に PREPARE_SUBSCRIBER_VIEW プロシージャをコールして作成されている必要があります。
「DROP_SUBSCRIPTION プロシージャ」26-14 ページ	発行者が事前に GET_SUBSCRIPTION_HANDLE プロシージャへのコールを使用して作成されたサブスクリプションを削除します。
「DROP_CHANGE_TABLE プロシージャ」26-15 ページ	既存の変更表においてアクティビティが発生していない場合、その表を削除します。
「PURGE プロシージャ」26-16 ページ	すべてのサブスクリプションにより使用を監視し、サブスクリプションにおいて不要な行を判別した上で、その行を削除して変更表のサイズが無制限に拡大することを防ぎます。

CREATE_CHANGE_TABLE プロシージャ

このプロシージャは、指定したスキーマに変更表を作成します。

構文

次の構文は、カンマで区切られた文字列を使用して列およびデータ・タイプを指定します。

```
DBMS_LOGMNR_CDC_PUBLISH.CREATE_CHANGE_TABLE (
     owner                  IN VARCHAR2,
     change_table_name      IN VARCHAR2,
     change_set_name        IN VARCHAR2,
     source_schema          IN VARCHAR2,
     source_table           IN VARCHAR2,
     column_type_list       IN VARCHAR2,
     capture_values         IN VARCHAR2,
     rs_id                  IN CHAR,
     row_id                 IN CHAR,
     user_id                IN CHAR,
     timestamp              IN CHAR,
     object_id              IN CHAR,
     source_colmap          IN CHAR,
     target_colmap          IN CHAR,
     options_string         IN VARCHAR2);
```

パラメータ

表 26-2 CREATE_CHANGE_TABLE プロシージャのパラメータ

パラメータ	説明
owner	変更表を所有するスキーマの名前。
change_table_name	作成する変更表の名前。
change_set_name	この変更表が関連付けられた既存の変更セットの名前。変更表の同期は SYNC_SET を指定する必要があります。
source_schema	ソース表が置かれているスキーマ。
source_table	変更レコードが獲得されるソース表。
column_type_list	追跡する列およびデータ・タイプのカンマで区切られたリスト。
capture_values	更新操作を行うには、次の獲得値のいずれかに対しこのパラメータを設定します。 ■ OLD: ソース表から元の値を獲得します。 ■ NEW: ソース表から変更された値を獲得します。 ■ BOTH: ソース表から元の値および変更された値を獲得します。
rs_id	行順序番号を含む変更表に列を追加します。このパラメータは、データベースでコミットされた順番でトランザクションの操作を処理します。行順序番号（rs_id）パラメータは、同期モードのオプションです。 **注意：**同期モードの場合、rs_id パラメータはトランザクションにおける操作の獲得順序を反映しますが、rs_id パラメータを使用し、コミットされた処理をトランザクション間で順序付けることはできません。 このパラメータを、次のように Y または N に設定します。 Y: 変更表に行の変更順序を含む列を追加することを示します。 N: rs_id 列を追跡しないことを示します。
row_id	次のように、変更表にソース表において変更された行の ID を含む列を追加します。 Y: 変更表にソース表において変更された行の行 ID を含む列を追加することを示します。 N: row_id 列を追跡しないことを示します。

表 26-2 CREATE_CHANGE_TABLE プロシージャのパラメータ（続き）

パラメータ	説明
user_id	次のように、変更表に DML 文を入力したユーザーのユーザー名を含む列を追加します。 Y: 変更表に DML 文を入力したユーザーのユーザーの名前を含む列を追加することを示します。 N: ユーザーを追跡しないことを示します。
timestamp	次のように、変更表に変更レコードの獲得タイムスタンプを含む列を追加します。 Y: 変更表に変更レコードの獲得タイムスタンプを含む列を追加することを示します。 N: タイムスタンプを追跡しないことを示します。
object_id	変更表に変更レコードのオブジェクト ID を含む列を追加します。これは、オブジェクト・サポート用の制御列です。次のように Y または N を指定してください。 Y: 変更表にこの変更レコードのオブジェクト ID を含む列を追加することを示します。 N: オブジェクト ID を追跡しないことを示します。
source_colmap	実際に変更されたソース列を示す変更列ベクトルとして変更表に列を追加します。次のように Y または N を指定してください。 Y: 変更されたソース列を追跡するために変更表に列を追加することを示します。 N: 変更されたソース列を追跡しないことを示します。
target_colmap	実際に変更された変更表のユーザー列を示す列ベクトルとして変更表に列を追加します。次のように Y または N を指定してください。 Y: 変更された変更表のユーザー列を追跡するために変更表に列を追加することを示します。 N: 変更された変更表のユーザー列を追跡しないことを示します。
options_string	CREATE TABLE DDL 文に渡される、構文的に正しいオプションを含む文字列。このオプション文字列は、生成された CREATE TABLE DDL 文の、表の列を定義するカッコの後に追加されます。詳細は、「使用上の注意」を参照してください。

例外

表 26-3 CREATE_CHANGE_TABLE プロシージャの例外

例外	説明
ORA-31409	CREATE_CHANGE_TABLE プロシージャに対する 1 つ以上の入力パラメータに無効な値が設定されています。不適切なパラメータを判別し、正しい値を設定してください。
ORA-31416	source_colmap パラメータに対して指定した値が無効です。同期モードの場合、Y または N を指定してください。
ORA-31417	列リストまたは列タイプ・パラメータで予約済みの列名が指定されています。指定した名前が予約済みの列名と競合しないことを確認してください。
ORA-31418	同期変更表の作成時に、ソース・スキーマの名前がデータベースにおける既存のスキーマ名と一致しませんでした。
ORA-31419	同期変更表の作成時に、プロシージャがコールされた元のソース表が存在しませんでした。
ORA-31420	最初の変更表の作成時に、パージ・ジョブがジョブ・キューに送られます。このパージ・ジョブの送信が失敗しました。
ORA-31421	指定した変更表が存在しません。指定した変更表の名前をチェックし、既存の変更表の名前と一致していることを確認してください。
ORA-31422	所有者のスキーマが存在しません。
ORA-31438	変更表を複製します。一意の名前で変更表を再作成してください。
ORA-31450	change_table_name に対して無効な値が指定されました。
ORA-31451	capture_value に対して無効な値が指定されました。OLD、NEW または BOTH のいずれかを指定してください。
ORA-31452	無効な値が指定されました。Y または N を指定してください。
ORA-31459	DBMS_LOGMRN_CDC_PUBLISH パッケージのシステム・トリガーがインストールされていません。
ORA-31467	ソース表に列がありません。CREATE_CHANGE_TABLE へのコールおよび同期変更セットに属する変更表で OBJECT_ID フラグが Y に設定されました。対応するオブジェクトの列がソース表で検出されませんでした。

使用上の注意

- 変更表はソース表に対して作成された DML 文（INSERT、UPDATE および DELETE）から発生した変更データを含むデータベース・オブジェクトです。指定した変更表は、単一のソース表からのみ変更を獲得できます。
- 同期変更表は、SYNC_SET 変更セットに属している必要があります。
- 変更表は、次の 2 種類の列にある変更データをメンテナンスするデータベース表です。
  - ソース列は、ソース表から獲得する列を識別します。ソース列は、変更表に常駐する実際のソース表の列のコピーです。
  - 制御列は、コンテナ表の各変更行に対する特殊なメタデータをメンテナンスします。制御列の例として、実行された DML 操作などの情報、獲得時間（タイムスタンプ）および変更された列ベクトルなどがあげられます。
- 発行者は、options_string パラメータを指定することにより、変更表の物理的なプロパティ、表領域のプロパティなどを制御できます。options_string パラメータを使用すると、CREATE TABLE DDL 文に対して有効なオプションをすべて設定できます。
- 変更表のパーティション・プロパティを制御しないでください。チェンジ・データ・キャプチャはパージ操作を実行して変更セットから行を削除して、変更表のパーティションを自動的に管理します。

注意： options_string パラメータ定義は、チェンジ・データ・キャプチャ・システムのパフォーマンスおよび操作に影響を与えます。たとえば、発行者が複数の制約をオプション列に設定した場合、パフォーマンスに著しく影響する可能性があります。また、発行者が NOT NULL 制約を使用し、獲得した変更行において特定の列が変更されていない場合、その制約により INSERT 操作全体が失敗する可能性があります。

例

```
execute DBMS_CDC_PUBLISH.CREATE_CHANGE_TABLE(OWNER => 'cdc1', \
   CHANGE_TABLE_NAME => 'emp_ct', \
   CHANGE_SET_NAME => 'SYNC_SET', \
   SOURCE_SCHEMA => 'scott', \
   SOURCE_TABLE => 'emp', \
   COLUMN_TYPE_LIST => 'empno number, ename varchar2(10), job varchar2(9), mgr
number, hiredate date, deptno number', \
   CAPTURE_VALUES => 'both', \
   RS_ID => 'y', \
   ROW_ID => 'n', \
   USER_ID => 'n', \
   TIMESTAMP => 'n', \
   OBJECT_ID => 'n',\
   SOURCE_COLMAP => 'n', \
   TARGET_COLMAP => 'y', \
   OPTIONS_STRING => NULL);
```

ALTER_CHANGE_TABLE プロシージャ

このプロシージャは、既存の変更表に列を追加または削除します。

構文

次の構文は、カンマで区切られた文字列を使用して列およびデータ・タイプを指定します。

```
DBMS_LOGMNR_CDC_PUBLISH.ALTER_CHANGE_TABLE (
     owner                  IN VARCHAR2,
     change_table_name      IN VARCHAR2,
     operation              IN VARCHAR2,
     column_list            IN VARCHAR2,
     rs_id                  IN CHAR,
     row_id                 IN CHAR,
     user_id                IN CHAR,
     timestamp              IN CHAR,
     object_id              IN CHAR,
     source_colmap          IN CHAR,
     target_colmap          IN CHAR);
```

パラメータ

表 26-4 ALTER_CHANGE_TABLE プロシージャのパラメータ

パラメータ	説明
owner	変更表を所有するスキーマの名前。
change_table_name	変更する変更表の名前。
operation	DROP または ADD のいずれかの値を指定し、column_table フィールドまたは column_list フィールドの列を追加または削除するかどうかを指定します。
column_list	変更表に追加または変更表から削除する必要があるソース表の各列に対する列名およびデータ・タイプのカンマで区切られたリスト。
rs_id	行の順序（rs_id）を追跡する制御列を追加または削除します。このパラメータを、次のように Y または N に設定します。 Y: 変更表に行の順序（rs_id）を含む列を追加または削除します。 N: rs_id 制御列は、変更表では変更されません。
row_id	次のように、row_id 列を追加または削除します。 Y: 変更表の row_id 制御列を追加または削除します。 N: row_id 列は、変更表では変更されません。
user_id	ユーザー名の制御列を追加または削除します。次のように Y または N を指定してください。 Y: 変更表にユーザー名（user_id）を含む列を追加または削除します。 N: user_id 列は、変更表では変更されません。
timestamp	次のようにタイムスタンプ制御列を変更表に追加または削除します。 Y: 変更表にタイムスタンプを含む列を追加または削除します。 N: タイムスタンプ制御列は、変更表では変更されません。
object_id	次のように、object_id 列を追加または削除します。 Y: 変更表に object_id を含む列を追加または削除します。 N: object_id 制御列は、変更表では変更されません。

表 26-4 ALTER_CHANGE_TABLE プロシージャのパラメータ（続き）

パラメータ	説明
source_colmap	次のように、source_colmap 制御列を変更表に追加または削除します。 Y: 変更表にソース列（source_colmap）を含む列を追加または削除します。 N: source_colmap 列は、変更表では変更されません。
target_colmap	次のように、target_colmap 制御列を変更表に追加または削除します。 Y: 変更表にターゲット列（target_colmap）を含む列を追加または削除します。 N: target_colmap 列は、変更表では変更されません。

例外

表 26-5 ALTER_CHANGE_TABLE プロシージャの例外

例外	説明
ORA-31403	ADD 操作を使用して ALTER_CHANGE_TABLE プロシージャを発行しましたが、この名前を持つ列が指定した表にすでに存在しています。
ORA-31409	ALTER_CHANGE_SET プロシージャに対する 1 つ以上の入力パラメータに無効な値が設定されています。不適切なパラメータを判別し、正しい値を設定してください。
ORA-31417	列リスト・パラメータで予約済みの列名が指定されています。指定した名前が予約済みの列名と競合しないことを確認してください。
ORA-31421	指定した変更表が存在しません。指定した変更表の名前をチェックし、既存の変更表の名前と一致していることを確認してください。
ORA-31423	DROP 操作を使用して ALTER_CHANGE_TABLE を発行しましたが、指定した列が変更表にありません。
ORA-31454	操作パラメータに指定された値が不正です。ADD または DROP を指定してください。
ORA-31455	変更できません。指定した列リストは NULL であり、オプションの制御列はすべて N です。
ORA-31456	DBMS_CDC_UTILITY パッケージ内のプロシージャの内部コールが失敗しました。詳細は、トレース・ログをチェックしてください。
ORA-31459	1 つ以上の必須システム・トリガーがインストールされていません。

使用上の注意

- ALTER_CHANGE_TABLE プロシージャに対するコールにおいてユーザー列を同時に追加および削除できません。このようなスキーマの変更は、別々のコールで行ってください。
- ユーザー列リストで制御列の名前を指定しないでください。
- 次の表で、変更表に列を追加した場合の状態を説明します。

発行者が追加するもの . . .	状況 . . .	結果 . .
ユーザー列	新しいサブスクリプションがこの列を含みます。	列が追加された時点でサブスクリプション・ウィンドウが起動します。
ユーザー列	新しいサブスクリプションは、新しく追加された列を含みません。	変更表の低水位標でサブスクリプション・ウィンドウが起動し、サブスクライバが表全体を確認できるようになります。
ユーザー列	元のサブスクリプションが存在します。	サブスクリプション・ウィンドウは変更されず、表全体が表示されます。
制御列	新しいサブスクリプションが存在します。	変更表の低水位標でサブスクリプション・ウィンドウが起動します。サブスクリプションは、制御列を即時に確認できます。制御列を追加する前に変更表に存在していた行の値はすべて、新しく追加された制御列のフィールドに対して NULL となります。
制御列	―	ウィンドウが拡張された場合（DBMS_LOGMNR_CDC_PUBLISH.EXTEND_WINDOW プロシージャ）、制御列が追加された時点でウィンドウの低水位標がポイントを越えるように、既存のサブスクリプションは新しい制御列を表示できます。

例

```
EXECUTE DBMS_LOGMNR_CDC_PUBLISH.ALTER_CHANGE_TABLE (OWNER => 'cdc1') \
   CHANGE_TABLE_NAME => 'emp_ct' \
   OPERATION => ADD \
   ADD_COLUMN_LIST => '' \
   RS_ID => 'Y' \
   ROW_ID => 'N' \
   USER_ID => 'N' \
   TIMESTAMP => 'N' \
   OBJECT_ID => 'N' \
   SOURCE_COLMAP => 'N' \
   TARGET_COLMAP => 'N');
```

DROP_SUBSCRIBER_VIEW プロシージャ

このプロシージャは、発行者がサブスクライバのスキーマからサブスクライバ・ビューを削除できるようにします。

> **注意：** このプロシージャは、DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIBER_VIEW プロシージャと同様に動作します。

構文

```
DBMS_LOGMNR_CDC_PUBLISH.DROP_SUBSCRIBER_VIEW (
     subscription_handle    IN NUMBER,
     source_schema          IN VARCHAR2,
     source_table           IN VARCHAR2)
```

パラメータ

表 26-6 DROP_SUBSCRIBER_VIEW プロシージャのパラメータ

パラメータ	説明
subscription_handle	事前の DBMS_LOGMNR_CDC_SUBSCRIBE.GET_SUBSCRIPTION_HANDLE プロシージャへのコールにより戻されたサブスクリプション・ハンドルの一意の番号。
source_schema	ソース表が常駐するスキーマの名前。
source_table	発行されたソース表の名前。

例外

表 26-7 DROP_SUBSCRIBER_VIEW プロシージャの例外

例外	説明
ORA-31425	サブスクリプション・ハンドルが存在しないか、ハンドルがこのユーザーに属しません。有効なサブスクリプション・ハンドルを使用し、再度ファンクションをコールしてください。
ORA-31429	サブスクリプションがアクティブにされていません。サブスクリプション・ハンドルをチェックし、必要に応じて修正してください。このサブスクリプション・ハンドルに対し DBMS_LOGMNR_CDC_SUBSCRIBE.ACTIVATE_SUBSCRIPTION プロシージャをコールし、元のコマンドを再度試行してください。
ORA-31432	schema_name.source_table が存在しないか、このサブスクリプションに属しません。schema_name および source_table パラメータの綴りをチェックしてください。指定した表が指定したスキーマに存在し、サブスクリプション・ハンドルによりサブスクライブされていることを確認してください。
ORA-31433	サブスクライバ・ビューが存在しません。不適切なサブスクライバ・ビューを指定したか、ビューがすでに削除されています。名前をチェックし、存在するサブスクライバ・ビューの名前を指定してください。

使用上の注意

- このプロシージャは、サブスクライバにより削除されなかったビューをクリーンアップする方法を発行者に提供します。通常、サブスクライバは、DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIBER_VIEW プロシージャを使用してサブスクライバ・ビューを削除します。
- 削除するサブスクライバ・ビューは、DBMS_LOGMNR_CDC_SUBSCRIBE.PREPARE_SUBSCRIBER_VIEW プロシージャへの事前のコールにより作成されている必要があります。
- DBMS_LOGMNR_CDC_PUBLISH.DROP_SUBSCRIPTION プロシージャを使用するサブスクリプションを削除するには、このプロシージャを使用してサブスクライバ・ビューをすべて削除しておく必要があります。

例

```
EXECUTE sys.DBMS_CDC_SUBSCRIBE.DROP_SUBSCRIBER_VIEW( \
   SUBSCRIPTION_HANDLE =>:subhandle, \
   SOURCE_SCHEMA =>'scott', \
   SOURCE_TABLE => 'emp');
```

DROP_SUBSCRIPTION プロシージャ

このプロシージャは、発行者が DBMS_LOGMNR_CDC_SUBSCRIBE.GET_SUBSCRIPTION_HANDLE プロシージャへの事前のコールを使用して作成されたサブスクリプションを削除できるようにします。

注意： このプロシージャは、DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIPTION プロシージャと同様に動作します。

構文

```
DBMS_LOGMNR_CDC_PUBLISH.DROP_SUBSCRIPTION (
     subscription_handle  IN NUMBER)
```

パラメータ

表 26-8 DROP_SUBSCRIPTION プロシージャのパラメータ

パラメータ	説明
subscription_handle	DBMS_LOGMNR_CDC_SUBSCRIBE.GET_SUBSCRIPTION_HANDLE プロシージャへの事前のコールにより戻されたサブスクリプション・ハンドルの一意の番号。

例外

表 26-9 DROP_SUBSCRIPTION プロシージャの例外

例外	説明
ORA-31425	サブスクリプション・ハンドルが存在しないか、ハンドルがこのユーザーに属しません。有効なサブスクリプション・ハンドルを使用し、再度ファンクションをコールしてください。
ORA-31430	このコールの前に、サブスクライバ・ビューが削除されていませんでした。DBMS_LOGMNR_CDC_PUBLISH.DROP_SUBSCRIBER_VIEW プロシージャをコールし、元のコマンドを再度試行してください。

使用上の注意

- このプロシージャは、サブスクライバにより削除されなかったサブスクリプションを削除する方法を発行者に提供します（通常、サブスクライバは、DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIPTION プロシージャを使用してサブスクリプションを削除します）。
- サブスクリプションを削除するには、DBMS_LOGMNR_CDC_PUBLISH.DROP_SUBSCRIBER_VIEW プロシージャを使用してサブスクライバ・ビューを削除しておく必要があります。

例

```
EXECUTE DBMS_LOGMNR_CDC_PUBLISH.DROP_SUBSCRIPTION ( \
   SUBSCRIPTION_HANDLE => :subhandle);
```

DROP_CHANGE_TABLE プロシージャ

このプロシージャは、既存の変更表を削除します。

構文

```
DBMS_LOGMNR_CDC_PUBLISH.DROP_CHANGE_TABLE (
     owner              IN VARCHAR2,
     change_table_name  IN VARCHAR2,
     force_flag         IN CHAR)
```

パラメータ

表 26-10 DROP_CHANGE_TABLE プロシージャのパラメータ

パラメータ	説明
owner	変更表を所有するスキーマの名前。
change_table_name	削除する変更表の名前。
force_flag	次のように、参照するサブスクリプションがあるかどうかに基づいて変更表を削除します。 Y: 参照するサブスクリプションがある場合でも変更表を削除します。 N: 参照するサブスクリプションがない場合にのみ変更表を削除します。

例外

表 26-11 DROP_CHANGE_TABLE プロシージャの例外

例外	説明
ORA-31421	指定した変更表が存在しません。指定した変更表の名前をチェックし、既存の変更表の名前と一致していることを確認してください。
ORA-31422	所有者のスキーマが存在しません。
ORA-31424	指定した変更表にはアクティブなサブスクリプションがあるため、削除できません。表を削除するには、force_flag パラメータを使用してすべてのサブスクライバからただちに変更表を削除します。
ORA-31441	表が変更表ではありません。変更表でない表に対し、DROP_CHANGE_TABLE プロシージャを実行しようとしました。

例

```
EXECUTE DBMS_LOGMNR_CDC_PUBLISH.DROP_CHANGE_TABLE ( \
   OWNER => 'cdc1', \
   CHANGE_TABLE_NAME => 'emp_ct' \
   FORCE_FLAG => 'N')
```

PURGE プロシージャ

このプロシージャは、すべてのサブスクリプションにより変更表の使用を監視し、サブスクリプションにおいて不要な行を判別した上で、その行を削除して変更表のサイズが無制限に拡大することを防ぎます。

構文

```
DBMS_LOGMNR_CDC_PUBLISH.PURGE ( )
```

例外

パージ操作では、標準の Oracle 例外（権限違反など）のみが戻されます。

使用上の注意

- このプロシージャは手動または自動のどちらでも実行できます。
  - 変更表のデータをパージする場合は、コマンドラインから随時手動でこのプロシージャを実行します。
  - このプロシージャをスクリプトで実行して定期的にパージ操作を実行し、変更表の拡大を事前に制御します。パージ操作が自動的に実行されないようにするために、いつでも操作を削除または無効（または中断）できます。
- このプロシージャを使用して、変更表の拡大を制御します。
- 変更表のパーティション・プロパティを制御しないでください。
  DBMS_LOGMNR_CDC_PUBLISH.PURGE プロシージャを実行すると、チェンジ・データ・キャプチャが自動的にパーティションのメンテナンスを実行します。

例

```
EXECUTE DBMS_LOGMNR_CDC_PUBLISH.PURGE
```

27

DBMS_LOGMNR_CDC_SUBSCRIBE

この章では、DBMS_LOGMNR_CDC_SUBSCRIBE パッケージを使用して、DBMS_LOGMNR_CDC_PUBLISH パッケージで獲得および発行された変更データを表示および問い合せる方法を説明します。

通常、チェンジ・データ・キャプチャ・システムには 1 人の発行者が設定され、Oracle ソース（リレーショナル）表に対する変更の獲得および発行はすべて、この発行者が行います。その他、多数のサブスクライバが設定されます。サブスクライバ（通常はアプリケーション）は、オラクル社が提供するパッケージ（DBMS_LOGMNR_CDC_SUBSCRIBE）を使用して発行されたデータにアクセスします。

> **関連項目：** Oracle チェンジ・データ・キャプチャにおけるモデルの発行およびサブスクライブの情報は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

この章では、次の項目について説明します。

変更データのサブスクライブ

発行者が変更表にデータを獲得しアクセス権を付与するようにシステムを設定すると、サブスクライバは目的とするソース表に対して発行された変更データにアクセスし、問い合せることが可能になります。サブスクライバは、DBMS_LOGMNR_CDC_SUBSCRIBE パッケージのプロシージャを使用して、次の主な目的を実現します。

1. 発行されたソース表およびソース列に対してサブスクリプションを作成し、目的の変更データを表示します。
2. サブスクライバが一連の変更データを受け取る準備が完了したとき、サブスクリプション・ウィンドウを拡張し、新しいサブスクライバ・ビューを作成します。
3. SELECT 文を使用して、サブスクライバ・ビューから変更データを取り出します。
4. 変更ブロックの処理を完了したとき、サブスクライバ・ビューを削除し、サブスクリプション・ウィンドウをパージします。
5. サブスクライバが変更データを必要としなくなった場合、サブスクリプションを削除します。

DBMS_LOGMNR_CDC_SUBSCRIBE サブプログラムの要約

サブスクライバの主な目的は、変更データを使用することです。DBMS_LOGMNR_CDC_SUBSCRIBE パッケージを使用して、各サブスクライバは一連のソース表にサブスクライブすることにより登録します。

表 27-1 に、DBMS_LOGMNR_CDC_SUBSCRIBE パッケージのプロシージャを示します。

表 27-1 DBMS_LOGMNR_CDC_SUBSCRIBE パッケージのサブプログラム

サブプログラム	説明
「GET_SUBSCRIPTION_HANDLE プロシージャ」27-5 ページ	サブスクリプションを 1 つの変更セットと関連付けるサブスクリプション・ハンドルを作成します。
「SUBSCRIBE プロシージャ」27-6 ページ	サブスクライバが変更データにアクセスするソース表およびソース列を指定します。
「ACTIVATE_SUBSCRIPTION プロシージャ」27-8 ページ	サブスクリプションが変更データへのアクセス開始の準備を完了していることを示します。
「EXTEND_WINDOW プロシージャ」27-9 ページ	新しい変更データが表示されるようにサブスクリプション・ウィンドウの境界（低および高水位標）を設定します。
「PREPARE_SUBSCRIBER_VIEW プロシージャ」27-11 ページ	現行のサブスクリプション・ウィンドウに包含された変更データを問い合せられるサブスクライバのスキーマに、サブスクライバ・ビューを作成します。
「DROP_SUBSCRIBER_VIEW プロシージャ」27-13 ページ	サブスクライバのスキーマからサブスクライバ・ビューを削除します。
「PURGE_WINDOW プロシージャ」27-14 ページ	サブスクリプション・ウィンドウの低水位標を設定し、サブスクライバが一連の変更データの処理を完了したチェンジ・データ・キャプチャ・システムを通知します。
「DROP_SUBSCRIPTION プロシージャ」27-14 ページ	GET_SUBSCRIPTION_HANDLE プロシージャへの事前のコールを使用して作成されたサブスクリプションを削除します。

サブスクライバは、エラーが発生しないかぎり表 27-1 に示された順序でプロシージャをコールします。エラーが発生した場合は、その時点でサブスクライバが終了します。図 27-1 に、DBMS_LOGMNR_CDC_SUBSCRIBE パッケージのプロシージャの使用における最も一般的なステップを示します。

図 27-1 サブスクリプション・フロー

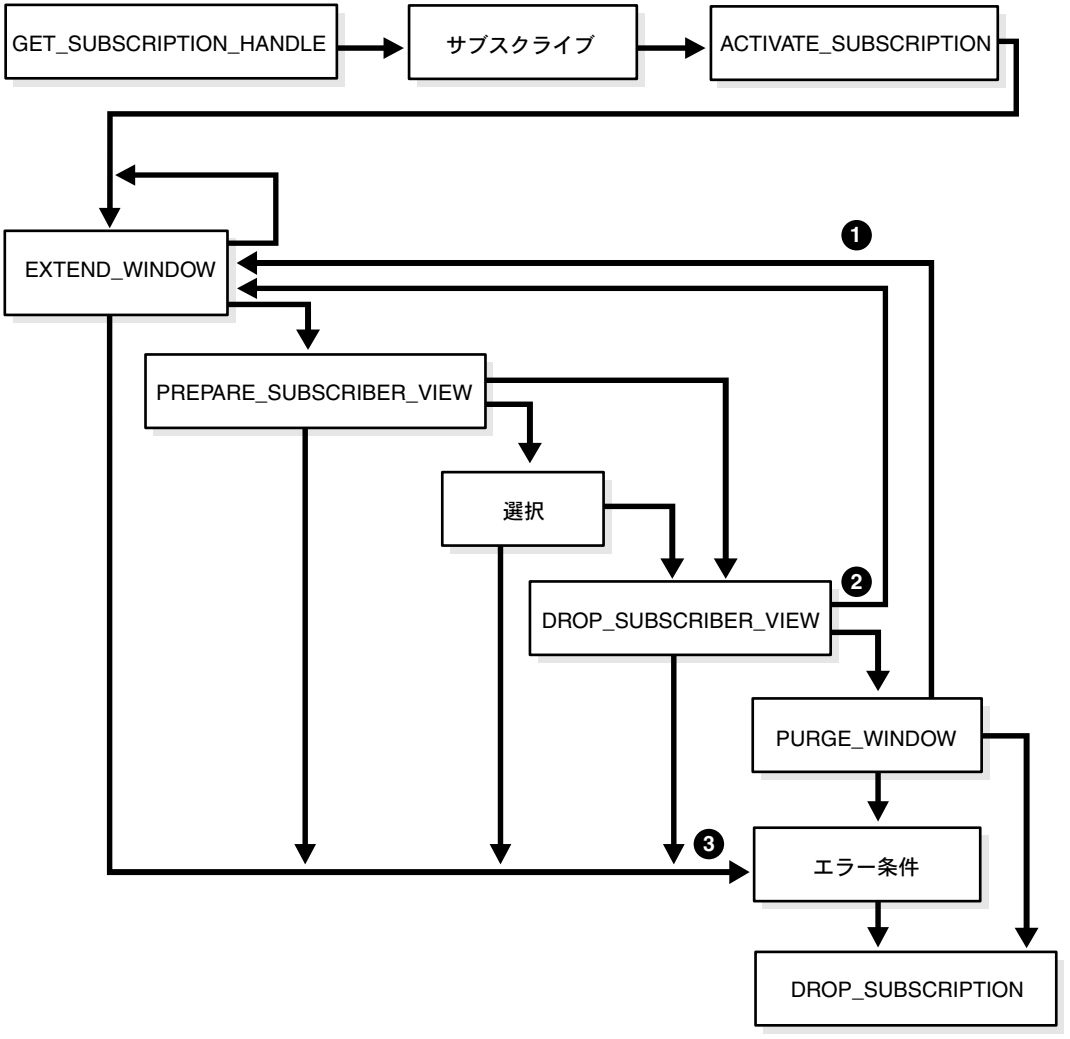


図 27-1 では、

1. EXTEND_WINDOW プロシージャを使用した直後に PURGE_WINDOW プロシージャを使用した場合、変更データが処理されずに消失します。
2. DROP_SUBSCRIBER_VIEW プロシージャを使用した直後に EXTEND_WINDOW プロシージャを使用した場合、処理されたデータが再び表示されます。また、新しいデータが表示される場合もあります。

3. 処理中のプロセスでエラーが発生した場合、DBMS_LOGMNR_CDC_SUBSCRIBE プロシージャをコールするアプリケーション・プログラムがエラーを検出し、終了する必要があります。たとえば、PREPARE_SUBSCRIBER_VIEW プロシージャがなんらかの理由で失敗し、アプリケーションがこのエラーを無視して処理を続行した場合、PURGE_WINDOW プロシージャは表示されなかったデータまたはサブスクライバに選択されたデータを削除します。

GET_SUBSCRIPTION_HANDLE プロシージャ

このプロシージャは、サブスクリプションを 1 つの変更セットと関連付けるサブスクリプション・ハンドルを作成します。サブスクリプション・ハンドルの作成は、サブスクリプション獲得における最初のステップです。

構文

```
DBMS_LOGMNR_CDC_SUBSCRIBE.GET_SUBSCRIPTION_HANDLE(
   change_set           IN  VARCHAR2,
   description          IN  VARCHAR2 := NULL,
   subscription_handle  OUT NUMBER);
```

パラメータ

表 27-2 GET_SUBSCRIPTION_HANDLE プロシージャのパラメータ

パラメータ	説明
change_set	アプリケーションがサブスクライブする既存の変更セットの名前。値を SYNC_SET に設定する必要があります。
description	サブスクリプション・ハンドルおよび使用目的を記述します。
subscription_handle	このサブスクリプションに対するサブスクリプション・ハンドルの一意の番号。

例外

表 27-3 GET_SUBSCRIPTION_HANDLE プロシージャの例外

例外	説明
ORA-31415	既存の変更セットの中に、この名前がありませんでした。
ORA-31457	説明フィールドで許可された最大文字数を超えました。
ORA-31458	内部エラーです。オラクル社カスタマ・サポート・センターに連絡して、エラーを報告してください。

使用上の注意

- GET_SUBSCRIPTION_HANDLE プロシージャを使用すると、サブスクライバは目的のソース表に関連付けられた変更セットを登録できます。
- サブスクライバが権限を付与された発行済みソース表をすべて表示するには、ALL_PUBLICATIONS ビューを問い合せてください。
- サブスクライバは後で単一のサブスクリプション・ハンドルを使用し、サブスクリプションにおける複数の変更表にアクセスできます。
- サブスクリプション・ハンドル:
  - DROP_SUBSCRIPTION プロシージャを使用して削除されないかぎり、作成時から再使用されることも追跡されることもありません。
  - サブスクライバ間で共有されません。むしろ、各サブスクリプション・ハンドルはサブスクライバのログイン ID に対して有効になります。

例

```
EXECUTE sys.DBMS_CDC_SUBSCRIBE.GET_SUBSCRIPTION_HANDLE(\
   CHANGE_SET=>'SYNC_SET', \
   DESCRIPTION=>'Change data for emp',\
   SUBSCRIPTION_HANDLE=>:subhandle);
```

SUBSCRIBE プロシージャ

このプロシージャは、サブスクライバが変更データにアクセスするソース表およびソース列を指定します。

構文

SUBSCRIBE プロシージャには 2 つのバージョンの構文があります。各構文は、サブスクライバの列およびデータ・タイプを指定します。サブスクライバが目的のソース列を含む発行を認識している場合、発行 ID を含むプロシージャのバージョンを使用できます。発行 ID が不明な場合、チェンジ・データ・キャプチャ・システムは、提供されたソース・スキーマおよびソース表に基づいて発行を選択します。

次の構文は、チェンジ・データ・キャプチャを使用して目的のソース列すべてを含む発行を選択できるようにすることにより、ソース表を識別します。

```
DBMS_LOGMNR_CDC_SUBSCRIBE.SUBSCRIBE (
   subscription_handle  IN NUMBER,
   source_schema        IN VARCHAR2,
   source_table         IN VARCHAR2,
   column_list          IN VARCHAR2);
```

次の構文は、目的のソース列を含む特定の発行の発行 ID を指定します。

```
DBMS_LOGMNR_CDC_SUBSCRIBE.SUBSCRIBE (
   subscription_handle  IN NUMBER,
   publication_id       IN NUMBER,
   column_list          IN VARCHAR2);
```

パラメータ

表 27-4 SUBSCRIBE プロシージャのパラメータ

パラメータ	説明
subscription_handle	GET_SUBSCRIPTION_HANDLE プロシージャへの前のコールにより戻されたサブスクリプション・ハンドルの一意の番号。
source_schema	ソース表が常駐するスキーマの名前。
source_table	発行されたソース表の名前。
column_list	発行されたソース表の列のカンマで区切られたリスト。
publication_id	ALL_PUBLISHED_COLUMNS ビューから取得できる有効な publication_id。

例外

表 27-5 SUBSCRIBE プロシージャの例外

例外	説明
ORA-31425	指定されたサブスクリプション・ハンドルが存在しないか、このユーザーまたはアプリケーションに属しません。
ORA-31426	サブスクリプション・ハンドルがアクティブにされており、SUBSCRIBE プロシージャをコールできません。サブスクリプションをアクティブにするには、目的の表および列のすべてをサブスクライブする必要があります。適切なサブスクリプション・ハンドルが指定されていることを確認してください。
ORA-31427	サブスクリプション・ハンドルにより示されたサブスクリプションには、スキーマ名およびソース表がすでに含まれています。subscription_handle、source_schema および source_table パラメータの値をチェックしてください。同じサブスクリプション・ハンドルを使用して、同じ表を複数回サブスクライブしないでください。
ORA-31428	指定した列すべてを含む発行がありません。1 つ以上の指定した列が単一の発行で見つかりません。ALL_PUBLISHED_COLUMNS ビューを調べて現在の発行を確認し、サブスクリプション要求を変更して同じ発行にある列のみを選択してください。

使用上の注意

- 有効な publication_id はいずれもサブスクライブできます。有効な発行は ALL_PUBLISHED_COLUMNS ビューで確認できます。
- SUBSCRIBE プロシージャを使用すると、アプリケーションにおいて、1 つ以上の発行済みソース表および各ソース表の特定の列に対しサブスクライブを実行できます。
- サブスクライバが権限を付与された発行済みソース表の列をすべて表示するには、ALL_PUBLISHED_COLUMNS ビューを問い合せてください。
- アプリケーションが実際にデータを必要とする前に、サブスクリプションが作成されている必要があります。チェンジ・データ・キャプチャ・システムは、サブスクリプションの作成時に変更データが使用可能であることは保証しません。
- サブスクライバは、ソース表から発行された列に対してのみサブスクライブを実行できます。また、列はすべて同じ発行のものである必要があります。元の変更表に関連付けられた制御列は、サブスクリプションに自動的に追加されます。

例

```
EXECUTE sys.DBMS_CDC_SUBSCRIBE.SUBSCRIBE(\
   SUBSCRIPTION_HANDLE=>:subhandle, \
   SOURCE_SCHEMA=>'scott', \
   SOURCE_TABLE=>'emp', \
   COLUMN_LIST=>'empno, ename, hiredate');
```

ACTIVATE_SUBSCRIPTION プロシージャ

ACTIVATE_SUBSCRIPTION プロシージャは、サブスクリプションが変更データへのアクセス開始の準備を完了していることを示します。

構文

```
DBMS_CDC_SUBSCRIBE.ACTIVATE_SUBSCRIPTION (
   subscription_handle  IN NUMBER);
```

パラメータ

表 27-6 ACTIVATE_SUBSCRIPTION プロシージャのパラメータ

パラメータ	説明
subscription_handle	GET_SUBSCRIPTION_HANDLE プロシージャへの前のコールにより戻されたサブスクリプション・ハンドルの一意の番号。

例外

表 27-7 ACTIVATE_SUBSCRIPTION プロシージャの例外

例外	説明
ORA-31425	指定されたサブスクリプション・ハンドルが存在しないか、このユーザー ID またはアプリケーションに属しません。
ORA-31439	サブスクリプションはすでにアクティブです。サブスクリプションをアクティブにできるのは 1 回のみです。

使用上の注意

- ACTIVATE_SUBSCRIPTION プロシージャは、表へのサブスクライブが完了したこと、およびサブスクリプションがデータへのアクセス開始の準備を完了していることを示します。
- サブスクライバがサブスクリプションをアクティブにすると、次のようになります。
  - それ以降、サブスクリプションにソース表を追加できません。
  - チェンジ・データ・キャプチャ・システムがソース表で使用できるデータを保持し、サブスクリプション・ウィンドウを空にします。
  - サブスクライバは EXTEND_WINDOW プロシージャを使用し、変更データの初期設定を確認する必要があります。
  - サブスクリプションは、再度アクティブにはできません。

例

```
EXECUTE sys.DBMS_CDC_SUBSCRIBE.ACTIVATE_SUBSCRIPTION( \
   SUBSCRIPTION_HANDLE=>:subhandle);
```

EXTEND_WINDOW プロシージャ

このプロシージャは、新しい変更データが表示されるように、サブスクリプション・ウィンドウの境界（低および高水位標）を設定します。

構文

```
DBMS_LOGMNR_CDC_SUBSCRIBE.EXTEND_WINDOW (
   subscription_handle  IN NUMBER);
```

パラメータ

表 27-8 EXTEND_WINDOW プロシージャのパラメータ

パラメータ	説明
subscription_handle	GET_SUBSCRIPTION_HANDLE プロシージャへの前のコールにより戻されたサブスクリプション・ハンドルの一意の番号。

例外

表 27-9 EXTEND_WINDOW プロシージャの例外

例外	説明
ORA-31425	指定されたサブスクリプション・ハンドルが存在しないか、このユーザーまたはアプリケーションに属しません。
ORA-31429	EXTEND_WINDOW プロシージャを使用する前に、サブスクリプション・ハンドルをアクティブにする必要があります。このサブスクリプション・ハンドルに対し ACTIVATE_SUBSCRIPTION プロシージャをコールし、元のコマンドを再度試行してください。
ORA-31430	このコールの前に、サブスクライバ・ビューが削除されていませんでした。DROP_SUBSCRIBER_VIEW プロシージャをコールし、元のコマンドを再度試行してください。

使用上の注意

- EXTEND_WINDOW プロシージャをコールして変更データの獲得を開始するまで、サブスクリプション・ウィンドウは空のままです。
  - EXTEND_WINDOW プロシージャを初めてコールすると、サブスクリプション・ウィンドウの初期境界が確立されます。
  - EXTEND_WINDOW プロシージャへの後続のコールが、新しい変更データが表示されるようにサブスクリプション・ウィンドウの高水位標を拡張します。

例

```
EXECUTE sys.DBMS_CDC_SUBSCRIBE.EXTEND_WINDOW( \
subscription_handle=>:subhandle);
```

PREPARE_SUBSCRIBER_VIEW プロシージャ

このプロシージャは、現行のサブスクリプション・ウィンドウに包含された変更データを問い合せることができるサブスクライバのスキーマに、サブスクライバ・ビューを作成します。

構文

```
DBMS_LOGMNR_CDC_SUBSCRIBE.PREPARE_SUBSCRIBER_VIEW (
   subscription_handle  IN  NUMBER,
   source_schema        IN  VARCHAR2,
   source_table         IN  VARCHAR2,
   view_name            OUT VARCHAR2);
```

パラメータ

表 27-10 PREPARE_SUBSCRIBER_VIEW プロシージャのパラメータ

パラメータ	説明
subscription_handle	GET_SUBSCRIPTION_HANDLE プロシージャへの前のコールにより戻されたサブスクリプション・ハンドルの一意の番号。
source_schema	ソース表が常駐するスキーマの名前。
source_table	サブスクリプション・ハンドルに属する発行済みソース表の名前。
view_name	ソース表に変更データを戻す、新しく作成されたビューの名前。

例外

表 27-11 PREPARE_SUBSCRIBER_VIEW プロシージャの例外

例外	説明
ORA-31425	指定されたサブスクリプション・ハンドルが存在しないか、このユーザーまたはアプリケーションに属しません。
ORA-31429	サブスクリプションがアクティブにされていません。PREPARE_SUBSCRIBER_VIEW プロシージャを使用する前に、サブスクリプション・ハンドルをアクティブにする必要があります。このサブスクリプション・ハンドルに対し ACTIVATE_SUBSCRIPTION プロシージャをコールし、元のコマンドを再度試行してください。
ORA-31430	このコールの前に、以前のサブスクライバ・ビューが削除されていませんでした。DROP_SUBSCRIBER_VIEW プロシージャをコールし、元のコマンドを再度試行してください。

表 27-11 PREPARE_SUBSCRIBER_VIEW プロシージャの例外（続き）

例外	説明
ORA-31432	スキーマ名またはソース表が存在しないか、このサブスクリプションに属しません。schema_name および source_table パラメータの綴りをチェックしてください。指定した表が指定したスキーマに存在し、サブスクリプション・ハンドルによりサブスクライブされていることを確認してください。

使用上の注意

- このプロシージャは、変更データを表示するサブスクライバのスキーマにサブスクライバ・ビューを作成します。サブスクライバ・ビューが作成された後、サブスクライバは EXTEND_WINDOW プロシージャによりサブスクリプション・ウィンドウに対して定義された境界内にある変更データを選択できます。
- チェンジ・データ・キャプチャ・システムはサブスクライバ・ビューの名前を判別し、サブスクライバに対してその名前を戻します。サブスクライバ・ビューの名前は、そのサブスクリプションが存続するかぎり一定です。変更データにアクセスするには、サブスクリプション内に各ソース表のビューがあることが必要です。アプリケーションはこれらのビューから SELECT 文を使用し、変更データを取り出します。次の例では、sys.sub9view が PREPARE_SUBSCRIBER_VIEW プロシージャにより戻されたビューの名前であると想定します。

```
SELECT * FROM sys.sub9view;
.
.
.
```

- 同じ view_name を持つビューが存在する場合（前のビューが DROP VIEW DDL 文を使用して削除されていない場合など）、例外が発生します。PREPARE_SUBSCRIBER_VIEW プロシージャは、元の変更表がまだ存在しているかどうかをチェックします。

例

```
EXECUTE sys.DBMS_CDC_SUBSCRIBE.PREPARE_SUBSCRIBER_VIEW( \
   SUBSCRIPTION_HANDLE =>:subhandle, \
   SOURCE_SCHEMA =>'scott', \
   SOURCE_TABLE => 'emp', \
   VIEW_NAME => :viewname);
```

DROP_SUBSCRIBER_VIEW プロシージャ

このプロシージャは、サブスクライバのスキーマからサブスクライバ・ビューを削除します。

構文

```
DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIBER_VIEW (
   subscription_handle  IN NUMBER,
   source_schema        IN VARCHAR2,
   source_table         IN VARCHAR2);
```

パラメータ

表 27-12 DROP_SUBSCRIBER_VIEW プロシージャのパラメータ

パラメータ	説明
subscription_handle	GET_SUBSCRIPTION_HANDLE プロシージャへの前のコールにより戻されたサブスクリプション・ハンドルの一意の番号。
source_schema	ソース表が常駐するスキーマの名前。
source_table	サブスクリプション・ハンドルに属する発行済みソース表の名前。

例外

表 27-13 DROP_SUBSCRIBER_VIEW プロシージャの例外

例外	説明
ORA-31425	サブスクリプション・ハンドルが存在しないか、ハンドルがこのユーザーに属しません。有効なサブスクリプション・ハンドルを使用し、再度ファンクションをコールしてください。
ORA-31429	サブスクリプションがアクティブにされていません。サブスクリプション・ハンドルをチェックし、必要に応じて修正してください。このサブスクリプション・ハンドルに対し ACTIVATE_SUBSCRIPTION プロシージャをコールし、元のコマンドを再度試行してください。
ORA-31432	schema_name.source_table が存在しないか、このサブスクリプションに属しません。schema_name および source_table パラメータの綴りをチェックしてください。指定した表が指定したスキーマに存在し、サブスクリプション・ハンドルによりサブスクライブされていることを確認してください。
ORA-31433	サブスクライバ・ビューが存在しません。不適切なソース表を指定したか、ビューがすでに削除されています。

使用上の注意

- 削除するサブスクライバ・ビューは、DBMS_LOGMNR_CDC_SUBSCRIBE.PREPARE_SUBSCRIBER_VIEW プロシージャへの事前のコールにより作成されている必要があります。
- DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIPTION プロシージャを使用するサブスクリプションを削除するには、このプロシージャを使用してサブスクライバ・ビューを削除しておく必要があります。

例

```
EXECUTE sys.DBMS_CDC_SUBSCRIBE.DROP_SUBSCRIBER_VIEW( \
   SUBSCRIPTION_HANDLE =>:subhandle, \
   SOURCE_SCHEMA =>'scott', \
   SOURCE_TABLE => 'emp');
```

PURGE_WINDOW プロシージャ

サブスクライバはこのプロシージャをコールし、チェンジ・データ・キャプチャ・システムに対し変更ブロックの処理を完了したことを通知します。PURGE_WINDOW プロシージャは、サブスクリプションが今後一切データを表示しないように低水位標を設定します。実質上、サブスクリプション・ウィンドウは空になります。

構文

```
DBMS_CDC_SUBSCRIBE.PURGE_WINDOW(
   subscription_handle    IN NUMBER);
```

パラメータ

表 27-14 PURGE_WINDOW プロシージャのパラメータ

パラメータ	説明
subscription_handle	GET_SUBSCRIPTION_HANDLE プロシージャへの前のコールにより戻されたサブスクリプション・ハンドルの一意の番号。

例外

表 27-15 PURGE_WINDOW プロシージャの例外

例外	説明
ORA-31425	サブスクリプション・ハンドルが存在しないか、ハンドルがこのユーザーに属しません。有効なサブスクリプション・ハンドルを使用し、再度ファンクションをコールしてください。
ORA-31429	EXTEND_WINDOW プロシージャを使用する前に、サブスクリプション・ハンドルをアクティブにする必要があります。このサブスクリプション・ハンドルに対し ACTIVATE_SUBSCRIPTION プロシージャをコールし、元のコマンドを再度試行してください。
ORA-31430	このコールの前に、サブスクライバ・ビューが削除されていませんでした。DROP_SUBSCRIBER_VIEW プロシージャをコールし、元のコマンドを再度試行してください。

使用上の注意

- 一連の変更を完了すると、サブスクライバは PURGE_WINDOW プロシージャを使用してサブスクリプション・ウィンドウをパージします。このアクションにより、サブスクライバは次のファンクションを実行します。
  - サブスクライバが変更データの次のバッチを受け取る準備を完了していることをチェンジ・データ・キャプチャ・システムに知らせます。
  - どのサブスクライバからも必要とされていない変更データをシステムで削除できるようにします。

  チェンジ・データ・キャプチャ・システムは、変更データを必要とするサブスクライバがいるかぎり、そのデータを使用できるように管理します。

例

```
EXECUTE sys.DBMS_CDC_SUBSCRIBE.PURGE_WINDOW ( \
SUBSCRIPTION_HANDLE=>:subhandle);
```

DROP_SUBSCRIPTION プロシージャ

このプロシージャは、GET_SUBSCRIPTION_HANDLE プロシージャへの事前のコールを使用して作成されたサブスクリプションを削除します。

構文

```
DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIPTION (
   subscription_handle  IN NUMBER);
```

パラメータ

表 27-16 DROP_SUBSCRIPTION プロシージャのパラメータ

パラメータ	説明
subscription_handle	GET_SUBSCRIPTION_HANDLE プロシージャへの事前のコールにより戻されたサブスクリプション・ハンドルの一意の番号。

例外

表 27-17 DROP_SUBSCRIPTION プロシージャの例外

例外	説明
ORA-31425	サブスクリプション・ハンドルが存在しないか、ハンドルがこのユーザーに属しません。有効なサブスクリプション・ハンドルを使用し、再度ファンクションをコールしてください。
ORA-31430	このコールの前に、サブスクライバ・ビューが削除されていませんでした。DROP_SUBSCRIBER_VIEW プロシージャをコールし、元のコマンドを再度試行してください。

使用上の注意

- サブスクリプションを削除するには、DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIBER_VIEW プロシージャを使用してサブスクライバ・ビューを削除しておく必要があります。

例

```
EXECUTE DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIPTION (\
   SUBSCRIPTION_HANDLE => :subhandle);
```

28

DBMS_LOGMNR_D

DBMS_LOGMNR_D パッケージには、LogMiner のプロシージャ DBMS_LOGMNR_D.BUILD および DBMS_LOGMNR_D.SET_TABLESPACE が含まれています。DBMS_LOGMNR_D.BUILD プロシージャは、REDO ログ・ファイルまたはフラット・ファイルのいずれかにディクショナリを抽出します。この情報は、将来的に LogMiner ツールを使用して REDO ログを分析する場合の準備として保存されます。DBMS_LOGMNR_D.SET_TABLESPACE プロシージャは、すべての LogMiner 表を代替の表領域に再作成します。

> **関連項目：** LogMiner の使用方法は、『Oracle9*i* データベース管理者ガイド』を参照してください。

この章では、次の項目について説明します。

DBMS_LOGMNR_D サブプログラムの要約

表 28-1 に、DBMS_LOGMNR_D パッケージのプロシージャを示します。

表 28-1 DBMS_LOGMNR_D パッケージのサブプログラム

サブプログラム	説明
「BUILD プロシージャ」 28-2 ページ	フラット・ファイルまたは REDO ログ内のファイルのいずれかにデータベース・ディクショナリを抽出します。
「SET_TABLESPACE プロシージャ」 28-5 ページ	すべての LogMiner 表を代替の表領域に再作成します。

BUILD プロシージャ

DBMS_LOGMNR_D.BUILD プロシージャの構文は次のとおりです。

構文

```
DBMS_LOGMNR_D.BUILD (
dictionary_filename IN VARCHAR2,
dictionary_location IN VARCHAR2,
options IN NUMBER);
```

パラメータ

表 28-2 に、BUILD プロシージャのパラメータを示します。

表 28-2 BUILD プロシージャのパラメータ

パラメータ	説明
dictionary_filename	ディクショナリ・ファイルの名前。
dictionary_location	ディクショナリ・ファイルへのパス。
options	フラット・ファイル（STORE_IN_FLAT_FILE）または REDO ログ（STORE_IN_REDO_LOGS）の宛先のいずれかにディクショナリが書き込まれることを指定します。

フラット・ファイルにディクショナリを抽出するには、ファイル名および場所を指定する必要があります。

REDO ログにディクショナリを抽出するには、STORE_IN_REDO_LOGS オプションのみを指定してください。ディクショナリのサイズによっては、ディクショナリが複数の REDO ログに含まれる場合があります。

要約すると、使用パラメータの組合せにより次のような動作が発生します。

- いずれのパラメータも指定しない場合、エラー・メッセージが戻されます。
- オプションを指定せずにファイル名および場所を指定した場合、ディクショナリはその名前でフラット・ファイルに抽出されます。
- ファイル名および場所を指定し、DBMS_LOGMNR_D.STORE_IN_FLAT_FILE オプションも指定した場合、ディクショナリは指定した名前でフラット・ファイルに抽出されます。
- ファイル名および場所を指定せずに、DBMS_LOGMNR_D.STORE_IN_REDO_LOGS オプションを指定した場合、ディクショナリは REDO ログに抽出されます。
- ファイル名および場所を指定して STORE_IN_REDO_LOGS オプションを指定した場合は、エラーが戻されます。

例外

- ORA-1308: 初期化パラメータ UTL_FILE_DIR が設定されていません。
- ORA-1336 －このエラーは次の状況で戻されます。
  1. dictionary_location が存在しない場合。
  2. UTL_FILE_DIR に dictionary_location へのアクセス権限が設定されていない場合。
  3. ディクショナリ・ファイルが読取り専用の場合。

使用上の注意

- 原則的には、ディクショナリ・ファイルは、データベースに対するすべてのディクショナリ変更を行った後、分析する REDO ログを作成する前に作成されます。Oracle9*i* リリース 1（9.0.1）では、LogMiner を使用してディクショナリを REDO ログにダンプし、DDL 操作を実行し、その変更を LogMiner ディクショナリに動的に適用できます。
- 進行中の DDL 操作がある場合、DBMS_LOGMNR_D.BUILD プロシージャは実行されません。
- DBMS_LOGMNR_D.BUILD プロシージャを使用するには、分析するファイルが格納されたデータベースをマウントおよびオープンする必要があります。
- ディクショナリ作成の進行状況を監視するには、SET SERVEROUTPUT ON コマンドを発行します。
- フラット・ファイルにディクショナリを抽出する場合、プロシージャが現行のデータベースのディクショナリ表を問い合せ、表のコンテンツを含むテキストベースのファイルを作成します。ディクショナリ・ファイルをフラット・ファイルに抽出するには、次の条件を満たしている必要があります。
  - ディクショナリ・ファイルは、分析する REDO ログを生成した同じデータベースから作成する必要があります。

  - PL/SQL プロシージャで使用するためにディレクトリを指定する必要があります。この処理を行うには、init.ora ファイルの初期化パラメータ UTL_FILE_DIR を設定します。たとえば、次のようにします。

    ```
    UTL_FILE_DIR = /oracle/dictionary
    ```

    このパラメータを設定しない場合、プロシージャは失敗します。

  - 作成されたディクショナリを実行している間、DDL 操作が発生しないようにする必要があります。そうしないと、ディクショナリ・ファイルにデータ・ディクショナリの一貫したスナップショットを含めることはできません。

- ディクショナリ・ファイルを REDO ログに抽出するには、次の条件を満たしている必要があります。

  - REDO ログに有効な情報が含まれていることを確認するために、（少なくとも最小レベルの）サプリメンタル・ロギングが使用可能であることが必要です。LogMiner でのサプリメンタル・ロギングの使用方法は、『Oracle9*i* データベース管理者ガイド』を参照してください。

  - Oracle9*i* 以降を実行しているシステム上で DBMS_LOGMNR_D.BUILD プロシージャが実行されている必要があります。

  - 使用可能な REDO を生成するために、アーカイブ・モードが有効化されている必要があります。

  - Oracle9*i* の互換性が指定されている必要があります。

  - Mining System が Oracle9*i* 以降である必要があります。

  - ディクショナリの REDO ファイルは、分析する REDO ログを生成した同じデータベースから作成する必要があります。

例 1: ディクショナリのフラット・ファイルへの抽出

次の例では、指定したパス（/oracle/database）にある dictionary.ora という名前のフラット・ファイルにディクショナリ・ファイルを抽出します。

```
SQL> EXECUTE dbms_logmnr_d.build('dictionary.ora', -
  2 '/oracle/database/', -
  3 options => dbms_logmnr_d.store_in_flat_file);
```

例 2: ディクショナリの REDO ログへの抽出

次の例では、REDO ログにディクショナリを抽出します。

```
SQL> EXECUTE dbms_logmnr_d.build ( -
  2 options => dbms_logmnr_d.store_in_redo_logs);
```

SET_TABLESPACE プロシージャ

デフォルトでは、すべての LogMiner 表は、SYSTEM 表領域を使用するように作成されます。ただし、代替の表領域を使用するように LogMiner 表を変更することをお薦めします。このルーチンを使用すると、すべての LogMiner 表を代替の表領域に再作成できます。

パラメータ

表 28-3 に、SET_TABLESPACE プロシージャのパラメータを示します。

表 28-3 SET_TABLESPACE パラメータ

パラメータ	説明
new_tablespace	すでに存在している表領域を示す文字列。この表領域を使用するようにすべての LogMiner 表を再作成する場合は、このパラメータのみを指定します。
dictionary_tablespace	すでに存在している表領域を示す文字列。このパラメータは、LogMiner ディクショナリ・データを、LogMiner のスピル・データが書き込まれる表領域とは異なる表領域に格納します。このパラメータは、LogMiner ディクショナリ表に関して、new_tablespace パラメータを上書きします。
spill_tablespace	すでに存在している表領域を示す文字列。このパラメータは、LogMiner のスピル・データを、LogMiner ディクショナリ・データが書き込まれる表領域とは異なる表領域に格納します。このパラメータは、LogMiner のスピル表に関して、new_tablespace パラメータを上書きします。

使用上の注意

- このプロシージャの実行時に LogMiner セッションを実行することはできず、このプロシージャの実行前に LogMiner を異常終了させることもできません。これらの状況では、予測できない結果になる場合があります。
- このルーチンは、他の製品で LogMiner を使用するように構成するために、1 回のみ実行することを目的としていますが、使用される表領域を再定義する必要がある場合は、複数回実行できます。ただし、前述の使用上の注意は守る必要があります。レイヤー化された製品で、その LogMiner セッションを強制的に終了するには高度な技術が必要であるため、このルーチンを 2 回以上使用することはお薦めしません。
- 特定のレイヤー化された製品の場合は、このルーチンを使用して、その製品が動作する前にすべての LogMiner 表の表領域を変更する必要があります。

- LogMiner 表で SYSTEM 表領域を使用しない場合は、一定のパフォーマンス最適化が行われる場合があります。特に、メモリー・スピル、LogMiner ディクショナリのロードおよび索引作成など、簡単に繰り返すことができる操作はログに記録されません。これは、データベース・リカバリの際の SYSTEM 表領域に関して容認できない影響を与えることになります。
- このルーチンのユーザーは、既存の表領域を指定する必要があります。表領域およびその作成方法に関する情報は、『Oracle9*i* データベース概要』および『Oracle9*i* SQL リファレンス』を参照してください。

例 : DBMS_LOGMNR_D.SET_TABLESPACE プロシージャの使用方法

次に、代替表領域の作成例および DBMS_LOGMNR_D.SET_TABLESPACE プロシージャの実行例を示します。

```
SQL> CREATE TABLESPACE  logmnrts$ datafile '/usr/oracle/dbs/logmnrts'
  2 SIZE 25 M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;

SQL> EXECUTE dbms_logmnr_d.set_tablespace('logmnrts$');
```

29

DBMS_LOGSTDBY

DBMS_LOGSTDBY パッケージは、ロジカル・スタンバイ・データベース環境を構成および管理するためのプロシージャを提供します。

> **関連項目：** ロジカル・スタンバイ・データベースの詳細は、『Oracle9*i* Data Guard 概要および管理』を参照してください。

この章では、次の項目について説明します。

ロジカル・スタンバイ環境の構成および管理

DBMS_LOGSTDBY パッケージは、ロジカル・スタンバイ環境の管理に役立つプロシージャを提供します。DBMS_LOGSTDBY パッケージのプロシージャを使用すると、次の主な目的を実現できます。

- メンテナンスが必要なスタンバイ・データベース内の表へのアクセス制御
- スタンバイ・データベース内の選択した表またはスキーマ全体に対するアーカイブ REDO ログの適用をスキップする方法の提供、および例外の処理方法の記述
- ログ適用サービスで使用される初期化パラメータの管理
- サプリメンタル・ロギングが正しく有効化されていることの確認、および LogMiner ディクショナリの作成

DBMS_LOGSTDBY サブプログラムの要約

表 29-1 に、DBMS_LOGSTDBY パッケージのプロシージャを示します。

表 29-1 DBMS_LOGSTDBY パッケージのサブプログラム

サブプログラム	説明
「APPLY_SET プロシージャ」29-3 ページ	ログ適用サービスを構成およびメンテナンスする特定の初期化パラメータの値を設定できます。
「APPLY_UNSET プロシージャ」29-8 ページ	特定の初期化パラメータの値をシステム・デフォルト値にリセットします。
「BUILD プロシージャ」29-9 ページ	サプリメンタル・ロギングが正しく有効化されていることを確認し、LogMiner ディクショナリを作成します。
「GUARD_BYPASS_OFF プロシージャ」29-9 ページ	GUARD_BYPASS_ON プロシージャを使用して以前にバイパスしたデータベース・ガードを再び使用可能にします。
「GUARD_BYPASS_ON プロシージャ」29-10 ページ	ロジカル・スタンバイ・データベース内の表を変更できるように、現行のセッションでデータベース・ガードをバイパスできるようにします。
「INSTANTIATE_TABLE プロシージャ」29-11 ページ	プライマリ・データベース内の対応する表から、スタンバイ・データベース内に表を作成し、データを移入します。
「SKIP プロシージャ」29-13 ページ	プライマリ・データベースで行われたデータベース処理の中で、ロジカル・スタンバイ・データベースに適用しない処理を指定できます。

表 29-1 DBMS_LOGSTDBY パッケージのサブプログラム（続き）

サブプログラム	説明
「SKIP_ERROR プロシージャ」29-19 ページ	エラーが発生した場合に従う基準を指定します。ログ適用サービスを停止するか、またはエラーを無視することを選択できます。
「SKIP_TRANSACTION プロシージャ」29-21 ページ	トランザクション識別情報を指定して、ロジカル・スタンバイ・データベースに対する特定トランザクションの適用をスキップします（処理しません）。
「UNSKIP プロシージャ」29-22 ページ	SKIP プロシージャで設定されたオプションを変更します。
「UNSKIP_ERROR プロシージャ」29-23 ページ	SKIP_ERROR プロシージャで設定されたオプションを変更します。
「UNSKIP_TRANSACTION プロシージャ」29-23 ページ	SKIP_TRANSACTION プロシージャで設定されたオプションを変更します。

APPLY_SET プロシージャ

このプロシージャを使用して、ロジカル・スタンバイ・データベース環境でログ適用サービスを構成および管理する初期化パラメータの値を設定および変更します。このプロシージャを使用しているときは、ログ適用サービスは実行できません。

構文

```
DBMS_LOGSTDBY.APPLY_SET (
     parameter          IN VARCHAR,
     value              IN VARCHAR);
```

パラメータ

表 29-2 に、APPLY_SET プロシージャのパラメータを示します。

表 29-2 DBMS_LOGSTDBY.APPLY_SET プロシージャのパラメータ

パラメータ	説明
APPLY_DELAY	スタンバイ・データベースでの管理リカバリ操作の適用遅延間隔（分）を指定します。 フェイルオーバー処理後にプライマリ・データベースを元に戻さない場合は、APPLY_UNSET プロシージャで APPLY_DELAY パラメータを使用します。
MAX_SGA	ログ適用サービスのキャッシュに対するシステム・グローバル領域（SGA）割当ての MB 数。デフォルト値は、SHARED_POOL_SIZE 初期化パラメータに設定された値の 4 分の 1 です。
MAX_SERVERS	ログ適用サービス用に確保されたパラレル問合せサーバーの数。デフォルトでは、ログ適用サービスは使用可能なすべてのパラレル問合せサーバーを使用して、ログ・ファイルの読込みおよび変更の適用を行います。パラレル問合せサーバーの詳細は、『Oracle9*i* データベース・リファレンス』を参照してください。
MAX_EVENTS_RECORDED	ロジカル・スタンバイのイベント情報が格納される DBA_LOGSTDBY_EVENTS 表に格納されるイベントの数。

表 29-2 DBMS_LOGSTDBY.APPLY_SET プロシージャのパラメータ（続き）

パラメータ	説明
TRANSACTION_CONSISTENCY	プライマリ・データベースとスタンバイ・データベース間でメンテナンスされるトランザクション一貫性のレベル。次のいずれかの値を指定します。 FULL: トランザクションは、プライマリ・データベースでコミットされた順序と同じ順序でロジカル・スタンバイ・データベースに適用されます （トランザクションは、コミット SCN 順に適用されます)。このオプションを選択すると、パフォーマンスが最も低くなります。これがデフォルトのパラメータ設定です。 NONE: トランザクションは、順序に関係なく適用されます。このオプションを選択すると、3 つのモードの中でパフォーマンスが最も高くなります。ただし、この設定では、スタンバイ・データベースの一貫性が失われる可能性があります。ロジカル・スタンバイ・データベースを読み込んでいるアプリケーションがトランザクション順に関して前提事項がない場合、このオプションは正常に動作します。たとえば、プライマリ・データベースで、あるトランザクションで新規顧客を追加し、その次のトランザクションでその顧客に新しいオーダーを追加したとします。スタンバイ・データベースでは、これらのトランザクションが逆に適用される場合があります。新規顧客に対するオーダーが先に追加される可能性があります。この場合に、スタンバイ・データベース上で、その新しいオーダーに対する顧客が見つかることを前提としているレポート作成アプリケーションを実行すると、そのアプリケーションは、制約がチェックされず、トリガーが起動されないために失敗します。 READ_ONLY: トランザクションは順序に関係なくコミットされます（パフォーマンスが向上します）が、順序の適用は定期的に実施されます。スタンバイ・データベースで実行される SQL SELECT 文は、適用エンジンが認識している一貫性のある最終 SCN に基づいて、常に一貫性のある結果を戻します。適用エンジンは、SGA でメンテナンスされている SCN を定期的にリフレッシュします。この SCN は一貫性のある状態を示します。スタンバイ・データベースで実行される問合せは、メンテナンスされている SCN に対して Oracle フラッシュバックを自動的に使用します。これは、ロジカル・スタンバイ・データベースがレポートの生成に使用されているときに有効です。このモードでは、Oracle フラッシュバックの制限がすべて適用されます。

表 29-2 DBMS_LOGSTDBY.APPLY_SET プロシージャのパラメータ（続き）

パラメータ	説明
RECORD_SKIP_ERRORS	スキップしたエラー（SKIP_ERROR プロシージャを参照）を DBA_LOGSTDBY_EVENTS 表に記録するかどうかを制御します。次のいずれかの値を指定します。 TRUE: スキップしたエラーは DBA_LOGSTDBY_EVENTS 表に記録されます。これがデフォルトのパラメータ設定です。 FALSE: スキップしたエラーは DBA_LOGSTDBY_EVENTS 表に記録されません。
RECORD_SKIP_DDL	スキップした DDL 文を DBA_LOGSTDBY_EVENTS 表に記録するかどうかを制御します。次のいずれかの値を指定します。 TRUE: スキップした DDL 文は DBA_LOGSTDBY_EVENTS 表に記録されます。これがデフォルトのパラメータ設定です。 FALSE: スキップした DDL 文は DBA_LOGSTDBY_EVENTS 表に記録されません。
RECORD_APPLIED_DDL	ロジカル・スタンバイ・データベースに適用された DDL 文を DBA_LOGSTDBY_EVENTS 表に記録するかどうかを制御します。次のいずれかの値を指定します。 TRUE: ロジカル・スタンバイ・データベースに適用された DDL 文が DBA_LOGSTDBY_EVENTS 表に記録されることを示します。これがデフォルトのパラメータ設定です。 FALSE: 適用した DDL 文は記録されないことを示します。

例外

表 29-3 に、APPLY_SET プロシージャの例外を示します。

表 29-3 DBMS_LOGSTDBY.APPLY_SET プロシージャの例外

例外	説明
ORA-16104	ロジカル・スタンバイ・オプションの要求が無効です。
ORA-16103	この操作を許可するにはロジカル・スタンバイ適用を停止する必要があります。

使用上の注意

- システムによって提供されている初期化パラメータのデフォルト値は、ほとんどのアプリケーションに適していますが、チューニングおよびメンテナンス・タスクを実行する必要があるときは、APPLY_SET プロシージャを使用することもできます。たとえば、ログ適用サービスをチューニングするために初期化パラメータのデフォルト値を上書きするときに、APPLY_SET プロシージャを使用します。
- APPLY_SET プロシージャを使用して初期化パラメータを変更したときは、ログ適用サービスで、スタンバイ・データベースにアーカイブ REDO ログ・データを適用しないでください。このプロシージャを使用して設定した初期化パラメータ値は、ログ適用サービスを起動するまで有効になりません。
- フェイルオーバー後、プライマリ・データベースが使用できなくなった場合は、DBMS_LOGSTDBY.APPLY_UNSET('APPLY_DELAY') プロシージャを使用して、初期化パラメータ・ファイルで指定されている設定を削除してください。
- APPLY_SET プロシージャの処理を元に戻す（UNDO する）には、APPLY_UNSET プロシージャを使用します。

例

アプリケーションによってパラレル問合せが定期的に実行される場合は、これらの問合せに対して一定数のパラレル・サーバーを確保する必要があります。ロジカル・スタンバイのログ適用サービスに 30 のパラレル問合せサーバーを割り当てるには、次の文を入力します。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_SET('MAX_SERVERS', 30);
```

この結果、PARALLEL_MAX_SERVERS パラメータが 50 に設定されている場合は、ロジカル・スタンバイ処理に 30 のサーバーが使用可能で、パラレル問合せ処理に 20 のパラレル問合せサーバーが割り当てられます。

> **注意：** パラレル問合せの実行中にログ適用サービスを起動すると、エラーが発生する場合があります。

APPLY_UNSET プロシージャ

APPLY_UNSET プロシージャを使用して、APPLY_SET プロシージャで行った設定を元に戻します（UNDO します）。APPLY_UNSET プロシージャは、指定した初期化パラメータ値をシステム・デフォルト値にリセットします。初期化パラメータのデフォルト値は、ログ適用サービスが起動されるまで有効になりません。

構文

```
DBMS_LOGSTDBY.APPLY_UNSET (
     parameter          IN VARCHAR);
```

パラメータ

APPLY_UNSET プロシージャは、APPLY_SET プロシージャで示した同じ初期化パラメータをサポートします。

> **関連項目：** APPLY_SET プロシージャのパラメータは、表 29-2 を参照してください。

使用上の注意

- APPLY_UNSET プロシージャを使用して初期化パラメータを変更したときは、ログ適用サービスで、スタンバイ・データベースにアーカイブ REDO ログ・データを適用しないでください。
- 初期化パラメータの値を設定するには、APPLY_SET プロシージャを使用します。

例

ログ適用サービスのパラレル問合せサーバー数の設定を元に戻すには、次の文を入力します。

```
SQL> EXECUTE DBMS_LOGSTDBY.APPLY_UNSET('MAX_SERVERS');
```

PARALLEL_MAX_SERVERS 初期化パラメータが 50 に設定されているとすると、この文は、パラレル問合せ処理に 50 のパラレル問合せサーバーを使用できるようにします。これは、デフォルトで、ログ適用サービスは使用可能なすべてのパラレル問合せサーバーを使用して、ログ・ファイルの読込みおよび変更の適用を行うためです。

> **注意：** パラレル問合せの実行中にログ適用サービスを起動すると、エラーが発生する場合があります。

BUILD プロシージャ

このプロシージャをプライマリ・データベースで使用して、REDO ログ内の重要なメタデータ（LogMiner ディクショナリ）情報を保存します。サプリメンタル・ロギングが正しく設定されていない場合、このプロシージャはその設定を行い、自動的に使用可能にします。

構文

```
DBMS_LOGSTDBY.BUILD;
```

パラメータ

なし

例外

なし

使用上の注意

- サプリメンタル・ログ情報には、ログ適用サービスでロジカル・スタンバイ・データベース内の表を一意に識別し、正しくメンテナンスできるように、アーカイブ REDO ログ内の特別な情報が含まれています。
- LogMiner ディクショナリ情報を使用すると、ログ適用サービスで REDO ログ内のデータを解析できます。

GUARD_BYPASS_OFF プロシージャ

GUARD_BYPASS_OFF プロシージャを使用して、GUARD_BYPASS_ON プロシージャで以前にバイパスしたデータベース・ガードを再び使用可能にします。

構文

```
DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
```

パラメータ

なし

例外

なし

使用上の注意

- ロジカル・スタンバイ・データベース内の表でデータベース・ガードをバイパスし、メンテナンスを実行する方法は、「GUARD_BYPASS_ON プロシージャ」を参照してください。

例

現行のセッションを GUARD_BYPASS_ON プロシージャが実行される前の状態に戻すには、次の文を入力します。

```
SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_OFF;
```

通常、このコマンドを使用する必要があるのは、データベース・ガードをバイパスするために GUARD_BYPASS_ON プロシージャを実行した後のみです。

GUARD_BYPASS_ON プロシージャ

デフォルトでは、ロジカル・スタンバイ・データベース内の表は変更できないように保護されています。しかし、GUARD_BYPASS_ON プロシージャを使用すると、データベース・ガードをバイパスし、ロジカル・スタンバイ・データベースを変更できます。たとえば、ロジカル・スタンバイ・データベース内の表でメンテナンスを実行したり、問題を修正することができます。このプロシージャを使用しているときは、トリガーが実行されず、制約がチェックされないため、データベースに対するトランザクションをアプリケーションで実行しないでください。

構文

```
DBMS_LOGSTDBY.GUARD_BYPASS_ON;
```

パラメータ

なし

例外

なし

使用上の注意

- このプロシージャは、現行のセッションに対してのみ有効です。
- GUARD_BYPASS_ON プロシージャでデータベース・ガードをバイパスしたときは、トリガーは実行されず、制約はチェックされません。
- GUARD_BYPASS_ON プロシージャを使用してデータベース・ガードをバイパスしているときは、アプリケーションが実行されないようにしてください。この環境は、問題の修正やメンテナンスの実行（例 : 索引の再作成またはマテリアライズド・ビューのリフレッシュ）などのメンテナンス作業のみを目的としています。

例

ロジカル・スタンバイ・データベース内の表を変更できるようにするには、次の文を入力します。

```
SQL> EXECUTE DBMS_LOGSTDBY.GUARD_BYPASS_ON;
```

INSTANTIATE_TABLE プロシージャ

このプロシージャは、プライマリ・データベース内の対応する表から、スタンバイ・データベース内に表を作成し、データを移入します。表には、入力パラメータとしてデータベース・リンク（dblink）名が必要です。

INSTANTIATE_TABLE プロシージャを使用すると、次のことを実行できます。

- スタンバイ・データベースへの表の追加
- スタンバイ・データベース内の表の再作成

構文

```
DBMS_LOGSTDBY.INSTANTIATE_TABLE (
     schema_name         IN VARCHAR2,
     table_name          IN VARCHAR2,
     dblink              IN VARCHAR2);
```

パラメータ

表 29-4 に、INSTANTIATE_TABLE プロシージャのパラメータを示します。

表 29-4 DBMS_LOGSTDBY.INSTANTIATE_TABLE プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマの名前。
table_name	スタンバイ・データベースに作成または再作成する表の名前。
dblink	プライマリ・データベース内の表の読込み権限およびロック権限があるデータベース・リンク・アカウントの名前。

例外

なし

使用上の注意

- このプロシージャを使用すると、プライマリ・データベースとトランザクション的に一貫性のある状態にスタンバイ・データベースのデータを保つように、表が作成され、データが移入されます。
- このプロシージャは、メタデータが正しくメンテナンスされていることを前提としています。
- この表は、実行時点でプライマリ・データベースで最新だった REDO ログがスタンバイ・データベースに適用されるまでは、信頼できる状態ではありません。
- このプロシージャは、BLOB データ・タイプがロジカル・スタンバイ・データベースでサポートされていても、BLOB データ・タイプをサポートしません。

例

スタンバイ・データベースに新しい表を作成し、データを移入するにはこの文を入力します。

```
SQL> EXECUTE DBMS_LOGSTDBY.INSTANTIATE_TABLE ('mytable','myschema' , 'mydblink');
```

SKIP プロシージャ

デフォルトでは、プライマリ・データベースで実行された SQL 文はすべて、ロジカル・スタンバイ・データベースに適用されます。プライマリ・データベース上の一部の処理のみがレプリケーションの対象である場合、SKIP プロシージャは、ロジカル・スタンバイ・データベースで SQL 文の適用を行わないようにするフィルタを定義します。SQL 文をスキップする（処理しない）ことがフィルタの主な目的ですが、ストアド・プロシージャをフィルタと関連付けると、文をスキップする、その文を実行する、または代替文を実行するかどうかを実行時に決定することができます。

このプロシージャをコールする前に、ログ適用サービスを停止する必要があります。停止するには、ALTER DATABASE STOP LOGICAL STANDBY APPLY 文を発行します。必要なフィルタをすべて指定した後、ALTER DATABASE START LOGICAL STANDBY APPLY 文を発行して、新しいフィルタ設定を使用してログ適用サービスを起動します。

構文

```
DBMS_LOGSTDBY.SKIP (
     statement_option          IN VARCHAR2,
     schema_name               IN VARCHAR2,
     object_name               IN VARCHAR2,
     proc_name                 IN VARCHAR2);
```

パラメータ

表 29-5 に、SKIP プロシージャのパラメータを示します。

表 29-5 DBMS_LOGSTDBY.SKIP プロシージャのパラメータ

パラメータ	説明
statement_option	SQL 文のセットまたは特定の SQL 文を識別するキーワード。キーワードを使用すると構成が簡単になります。これは、指定オブジェクトで動作するすべての SQL 文をキーワード（通常はデータベース・オブジェクトで定義）が識別するためです。表 29-6 に、キーワードとそれに対応する SQL 文を示します。いずれもこのパラメータに有効な値です。
schema_name	statement_option パラメータによって識別される SQL 文と関連付けられる 1 つ以上のスキーマの名前（ワイルド・カードの使用可）。必要でない場合は、この値を NULL に設定してください。
object_name	statement_option パラメータによって識別される SQL 文と関連付けられる 1 つ以上のオブジェクトの名前（ワイルド・カードの使用可）。必要でない場合は、この値を NULL に設定してください。

表 29-5 DBMS_LOGSTDBY.SKIP プロシージャのパラメータ（続き）

パラメータ	説明
proc_name	特定の文が、statement_option、schema_name および object_name パラメータによって定義されたフィルタと一致するかどうかをログ適用サービスが判断するときにコールするストアド・プロシージャの名前。次のフォーマットで指定します。 '"schema"."package"."procedure"' このプロシージャは、ログ適用サービスに対して、文の実行、文のスキップまたは代替文の実行のいずれかを行うように指示する値を戻します。 ログ適用サービスは、次のコール識別記号を指定してストアド・プロシージャをコールします。 ■ IN STATEMENT VARCHAR2 -- フィルタと一致する SQL 文。 ■ IN STATEMENT_TYPE VARCHAR2 -- フィルタの statement_option。 ■ IN SCHEMA VARCHAR2 -- フィルタの schema_name（該当する場合）。 ■ IN NAME VARCHAR2 -- フィルタの object_name（該当する場合）。 ■ IN XIDUSN NUMBER -- トランザクション ID パート 1。 ■ IN XIDSLT NUMBER -- トランザクション ID パート 2。 ■ IN XIDSQN NUMBER -- トランザクション ID パート 3。 ■ OUT SKIP_ACTION NUMBER -- このルーチンの完了時にログ適用サービスが実行する処理。有効な値は次のとおりです。 SKIP_ACTION_APPLY -- 文の実行 SKIP_ACTION_SKIP -- 文のスキップ SKIP_ACTION_REPLACE -- NEW_STATEMENT 出力パラメータで提供された代替文の実行 ■ OUT NEW_STATEMENT VARCHAR2 -- 元の文のかわりに実行する文。このオプションを使用するには、SKIP_ACTION が SKIP_ACTION_REPLACE に設定されている必要があります。それ以外の場合、このオプションは NULL に設定します。

注意： ハードウェアまたはソフトウェアの障害によってログ適用サービスが停止した場合、原子性を保つ実行は保証されません。障害状況では、文は複数回実行される可能性があります。

これらのストアド・プロシージャでは、DBMS_LOGSTDBY.SKIP('DML'...) はサポートされません。複数のワイルド・カードが、statement_option パラメータによって定義されたデータベース文オブジェクトと一致する場合、一致するストアド・プロシージャのうち 1 つのみがコールされます（プロシージャのアルファベット順）。

スキップ文のオプション

表 29-6 に、SKIP プロシージャの statement_option パラメータに対してサポートされる値を示します。表の左の列は、その右側の SQL 文のセットを識別するために使用されるキーワードです。ただし、右の列の SQL 文も有効な値です。キーワードは通常、データベース・オブジェクトによって定義されることに注意してください。

表 29-6 statement_option パラメータに対してサポートされる値

キーワード	対応する SQL 文
NON_SCHEMA_DDL	特定のスキーマに関係しないすべての DDL
SCHEMA_DLL	特定のスキーマに関係するすべての DDL
DML	*sequence.nextval* などのシーケンス操作
CLUSTER	CREATE CLUSTER AUDIT CLUSTER DROP CLUSTER TRUNCATE CLUSTER
CONTEXT	CREATE CONTEXT DROP CONTEXT
DATABASE LINK	CREATE DATABASE LINK DROP DATABASE LINK
DIMENSION	CREATE DIMENSION ALTER DIMENSION DROP DIMENSION
DIRECTORY	CREATE DIRECTORY DROP DIRECTORY
INDEX	CREATE INDEX ALTER INDEX DROP INDEX

表 29-6 statement_option パラメータに対してサポートされる値（続き）

キーワード	対応する SQL 文
PROCEDURE[1]	CREATE FUNCTION CREATE LIBRARY CREATE PACKAGE CREATE PACKAGE BODY CREATE PROCEDURE DROP FUNCTION DROP LIBRARY DROP PACKAGE DROP PROCEDURE
PROFILE	CREATE PROFILE ALTER PROFILE DROP PROFILE
PUBLIC DATABASE LINK	CREATE PUBLIC DATABASE LINK DROP PUBLIC DATABASE LINK
PUBLIC SYNONYM	CREATE PUBLIC SYNONYM DROP PUBLIC SYNONYM
ROLE	CREATE ROLE ALTER ROLE DROP ROLE SET ROLE
ROLLBACK STATEMENT	CREATE ROLLBACK SEGMENT ALTER ROLLBACK SEGMENT DROP ROLLBACK SEGMENT
SEQUENCE	CREATE SEQUENCE DROP SEQUENCE
SESSION	Logons
SYNONYM	CREATE SYNONYM DROP SYNONYM
SYSTEM AUDIT	AUDIT *SQL_statements* NOAUDIT *SQL_statements*
SYSTEM GRANT	GRANT *system_privileges_and_roles* REVOKE *system_privileges_and_roles*
TABLE	CREATE TABLE DROP TABLE TRUNCATE TABLE
TABLESPACE	CREATE TABLESPACE DROP TABLESPACE TRUNCATE TABLESPACE

表 29-6 statement_option パラメータに対してサポートされる値（続き）

キーワード	対応する SQL 文
TRIGGER	CREATE TRIGGER ENABLE および DISABLE 句を伴う ALTER TRIGGER DROP TRIGGER ENABLE ALL TRIGGERS 句を伴う ALTER TABLE DISABLE ALL TRIGGERS 句を伴う ALTER TABLE
TYPE	CREATE TYPE CREATE TYPE BODY ALTER TYPE DROP TYPE DROP TYPE BODY
USER	CREATE USER ALTER USER DROP USER
VIEW	CREATE VIEW DROP VIEW

[1] Java スキーマ・オブジェクト（ソース、クラスおよびリソース）は、SQL 文をスキップする（処理しない）という目的に関してはプロシージャと同様であるとみなされます。

例外

表 29-7 に、SKIP プロシージャの例外を示します。

表 29-7 DBMS_LOGSTDBY.SKIP プロシージャの例外

例外	説明
ORA-16203	" スキップ・プロシージャの戻り値を解釈できません " SKIP プロシージャで例外が発生したか、または戻り値が明確でないことを示します。DBA_LOGSTDBY_EVENTS ビューを調べると、例外が発生したプロシージャを識別できます。

使用上の注意

- SKIP プロシージャは、DDL 文をスキップする場合は特に注意して使用してください。たとえば、CREATE TABLE 文をスキップする場合は、その表を参照する他の DDL 文も SKIP プロシージャで指定する必要があります。このように処理しないと、文が失敗し、例外が発生する原因となります。例外が発生すると、ログ適用サービスの実行が停止します。
- SKIP プロシージャの設定を元に戻す（UNDO する）方法は、「UNSKIP プロシージャ」を参照してください。

例

次の例は、SKIP プロシージャを使用して、ロジカル・スタンバイ・データベースでスキーマをスキップする（処理しない）方法を示しています。

例 1 スキーマのスキップ

指定したスキーマに対する変更をスキップするには、ログ適用サービスで、スキーマでの新規オブジェクトの作成およびスキーマ内の既存オブジェクトの変更が行われないようにする必要があります。また、スキーマをサポートする表領域も変更されないようにする必要があります。次は、この処理を SKIP プロシージャを使用して行う例を示しています。この例では、処理しない表領域 *bones* に、スキーマ *smith* の表がいくつか定義されています。

```
BEGIN
DBMS_LOGSTDBY.SKIP('SCHEMA_DDL', 'SMITH', '%', null);
DBMS_LOGSTDBY.SKIP('DML', 'SMITH', '%', null);
DBMS_LOGSTDBY.SKIP('TABLESPACE', null, null, 'SMITH.PROTECT_BONES');

END;
```

前述の例では、object_name パラメータにワイルド・カードが使用され、フィルタがすべてのオブジェクトに適用されることを示しています。SKIP プロシージャの最後のコールで、PROTECT_BONES プロシージャが提供されているため、TABLESPACE の指定によって、BONES での表領域操作を防止できます。次は、PROTECT_BONES プロシージャの定義例です。

```
CREATE OR REPLACE PROCEDURE PROTECT_BONES (statement      IN  VARCHAR2,
                                           statement_type IN  VARCHAR2,
                                           schema         IN  VARCHAR2,
                                           name           IN  VARCHAR2,
                                           xidusn         IN  NUMBER,
                                           xidslt         IN  NUMBER,
                                           xidsqn         IN  NUMBER,
                                           skip_action    OUT NUMBER,
                                           new_statement  OUT VARCHAR2) AS
    BEGIN
       -- Init
       new_statement := NULL;

       -- Guaranteed to be either CREATE, DROP, or TRUNCATE TABLESPACE
       IF statement LIKE '%TABLESPACE BONES%'
       THEN
           -- Skip the statement
           skip_action := DBMS_LOGSTDBY.SKIP_ACTION_SKIP;
       ELSE
           -- Apply the statement
           skip_action := DBMS_LOGSTDBY.SKIP_ACTION_APPLY;
```

```
    END IF;
END protect_bones;
```

SKIP_ERROR プロシージャ

エラーが発生したときに、ロジカル・スタンバイ機能は、このプロシージャに含まれている基準を使用して、ログ適用サービスを停止するかどうかを判断します。スキップされるエラーはすべて、例外の処理方法を記述するシステム表に格納されます。

構文

```
DBMS_LOGSTDBY.SKIP_ERROR (
     statement_option          IN VARCHAR2,
     schema_name               IN VARCHAR2,
     object_name               IN VARCHAR2,
     proc_name                 IN VARCHAR2);
```

パラメータ

表 29-8 に、SKIP_ERROR プロシージャのパラメータを示します。

表 29-8 DBMS_LOGSTDBY.SKIP_ERROR プロシージャのパラメータ

パラメータ	説明
statement_option	SQL 文のセットまたは特定の SQL 文を識別するキーワード。キーワードを使用すると構成が簡単になります。これは、指定オブジェクトで動作するすべての SQL 文をキーワード（通常はデータベース・オブジェクトで定義）が識別するためです。表 29-6 に、キーワードとそれに対応する SQL 文を示します。いずれもこのパラメータに有効な値です。
schema_name	statement_option パラメータによって識別される SQL 文と関連付けられる 1 つ以上のスキーマの名前（ワイルド・カードの使用可）。必要でない場合は、この値を NULL に設定してください。
object_name	statement_option パラメータによって識別される SQL 文と関連付けられる 1 つ以上のオブジェクトの名前（ワイルド・カードの使用可）。必要でない場合は、この値を NULL に設定してください。

表 29-8 DBMS_LOGSTDBY.SKIP_ERROR プロシージャのパラメータ（続き）

パラメータ	説明
proc_name	特定の文が、statement_option、schema_name および object_name パラメータによって定義されたフィルタと一致するかどうかをログ適用サービスが判断するときにコールするストアド・プロシージャの名前。次のフォーマットで指定します。 'schema.package.procedure' このプロシージャは、ログ適用サービスに対して、文の実行、文のスキップまたは代替文の実行のいずれかを行うように指示する値を戻します。 ログ適用サービスは、次のコール識別記号を指定してストアド・プロシージャをコールします。 ■ IN STATEMENT VARCHAR(4000) -- 文の先頭 4K。 ■ IN STATEMENT_TYPE VARCHAR2 -- フィルタの statement_option。 ■ IN SCHEMA VARCHAR2 -- フィルタの schema_name（該当する場合）。 ■ IN NAME VARCHAR2 -- フィルタの object_name（該当する場合）。 ■ IN XIDUSN NUMBER -- トランザクション ID パート 1。 ■ IN XIDSLT NUMBER -- トランザクション ID パート 2。 ■ IN XIDSQN NUMBER -- トランザクション ID パート 3。 ■ IN ERROR VARCHAR(4000) -- 記録されるエラー・テキスト（オプション）。 ■ OUT NEW_ERROR VARCHAR(4000) -- NULL または修正されたエラー・テキスト。

例外

なし

使用上の注意

- SKIP_ERROR プロシージャに提供されているストアド・プロシージャは、ログ適用サービスで、スタンバイ・データベースへの REDO ログの適用を停止する可能性のあるエラーが発生した場合にコールされます。

  このストアド・プロシージャの実行は、DBA_LOGSTDBY_EVENTS 表の STATUS 列に書き込まれるエラーに作用します。STATUS_CODE 列は変更されません。ストアド・プロシージャの効果がない、つまり適用が停止された場合は、NEW_ERROR がイベント表に書き込まれます。まったく効果がないようにするには、そのプロシージャで NEW_ERROR を ERROR に設定します。

  ストアド・プロシージャで停止の回避が必要な場合は、NEW_ERROR を NULL に設定します。

例

```
DBMS_LOGSTDBY.SKIP_ERROR('DDL', 'joe', 'apptemp', null);
```

SKIP_TRANSACTION プロシージャ

このプロシージャは、ロジカル・スタンバイ・データベースへのトランザクションの適用をスキップする（処理しない）方法を提供します。トランザクション識別情報を指定することによって、特定のトランザクションをスキップできます。

SKIP_TRANSACTION プロシージャを使用すると、次のことを実行できます。

- すでに失敗したトランザクションで、ログ適用サービスの停止の原因となる可能性があるトランザクションのスキップ
- 論理的にデータを破壊する可能性があるトランザクションのスキップ

特定のトランザクション（例 : DDL トランザクション）が原因でログ適用サービスが停止した場合、そのトランザクション ID を指定して、適用を継続できます。このプロシージャは、ログ適用サービスで処理しないトランザクションの数と同じ回数コールできます。

> **注意 :** トランザクションをスキップしたときに、プライマリ・データベースとロジカル・スタンバイ・データベースのデータが分岐しないようにしてください。可能な場合は、スキップしたトランザクションのかわりに、補足トランザクションを手動で実行してください。

構文

```
DBMS_LOGSTDBY.SKIP_TRANSACTION (
     XIDUSN NUMBER          STRING,
     XIDSLT NUMBER          STRING,
     XIDSQN NUMBER          STRING);
```

パラメータ

表 29-9 に、SKIP_TRANSACTION プロシージャのパラメータを示します。

表 29-9 DBMS_LOGSTDBY.SKIP_TRANSACTION プロシージャのパラメータ

パラメータ	説明
XIDUSN NUMBER	スキップ対象トランザクションのトランザクション ID UNDO セグメント番号。
XIDSLT NUMBER	スキップ対象トランザクションのトランザクション ID スロット番号。
XIDSQN NUMBER	スキップ対象トランザクションのトランザクション ID 順序番号。

使用上の注意

- ログ適用サービスがロジカル・スタンバイ・データベースに対するトランザクション処理を停止した理由を判断するには、DBA_LOGSTDBY_EVENTS の最後の文を参照してください。提供されている文とエラー条件を検証してください。
- DBA_LOGSTDBY_SKIP_TRANSACTION ビューを使用すると、ログ適用サービスでスキップされる予定のトランザクションが示されます。

例外

なし

UNSKIP プロシージャ

このプロシージャは、レコードを検索してすべてのパラメータを照合し、システム表からレコードを削除して、SKIP プロシージャの処理を元に戻します。照合は正確に行う必要があり、複数の skip 処理は、同じ数の unskip 処理によってのみ元に戻すことができます。複数のスキップ処理をワイルド・カード文字を使用して元に戻すことはできません。

構文

```
DBMS_LOGSTDBY.UNSKIP (
     statement_option          IN VARCHAR2,
     schema_name               IN VARCHAR2,
     object_name               IN VARCHAR2);
```

パラメータ

UNSKIP プロシージャのパラメータ情報は、SKIP プロシージャで説明した情報と同じです。パラメータ情報の詳細は、表 29-5 を参照してください。

例外

なし

UNSKIP_ERROR プロシージャ

このプロシージャは、レコードを検索してすべてのパラメータを照合し、システム表からレコードを削除して、SKIP_ERROR プロシージャの処理を元に戻します（UNDO します）。照合は正確に行う必要があり、複数の skip 処理は、同じ数の unskip 処理によってのみ元に戻すことができます。1 回の unskip プロシージャ・コールで複数の skip 処理を元に戻すことはできません。

構文

```
DBMS_LOGSTDBY.UNSKIP_ERROR (
     statement_option          IN VARCHAR2,
     schema_name               IN VARCHAR2,
     object_name               IN VARCHAR2);
```

パラメータ

UNSKIP_ERROR プロシージャのパラメータ情報は、SKIP_ERROR プロシージャで説明した情報と同じです。パラメータ情報の詳細は、表 29-8 を参照してください。

例外

なし

例

```
DBMS_LOGSTDBY.UNSKIP_ERROR;
```

UNSKIP_TRANSACTION プロシージャ

このプロシージャは、SKIP_TRANSACTION プロシージャの処理を元に戻します。照合は正確に行う必要があり、複数の skip transaction 処理は、同じ数の unskip transaction 処理によってのみ元に戻すことができます。複数のスキップ・トランザクション処理をワイルド・カード文字を使用して元に戻すことはできません。

構文

```
DBMS_LOGSTDBY.UNSKIP_TRANSACTION (
     XIDUSN NUMBER          STRING,
     XIDSLT NUMBER          STRING,
     XIDSQN NUMBER          STRING);
```

パラメータ

表 29-10 に、UNSKIP_TRANSACTION プロシージャのパラメータを示します。

表 29-10 DBMS_LOGSTDBY.UNSKIP_TRANSACTION プロシージャのパラメータ

パラメータ	説明
XIDUSN NUMBER	スキップ対象トランザクションのトランザクション ID UNDO セグメント番号。
XIDSLT NUMBER	スキップ対象トランザクションのトランザクション ID スロット番号。
XIDSQN NUMBER	スキップ対象トランザクションのトランザクション ID 順序番号。

使用上の注意

- DBA_LOGSTDBY_SKIP_TRANSACTION ビューを使用すると、ログ適用サービスでスキップされる予定のトランザクションが示されます。

例外

なし

30

DBMS_METADATA

DBMS_METADATA を使用して、次の指定をすることにより、ディクショナリから完全なデータベース・オブジェクト定義（メタデータ）を取り出せます。

- オブジェクトのタイプ（表、索引またはプロシージャなど）。
- 所有者または名前などの選択条件（オプション）。
- 解析項目（解析され、個別に戻される戻りオブジェクトの属性）。
- 出力時の変換（オプション）。デフォルトでは、出力は XML 形式で表されますが、コール元は変換（SQL DDL など）の指定ができます。変換は、データベース内または外部に格納された XSLT（eXtensible Stylesheet Language Transformation）スタイルシートにより実装されます。

DBMS_METADATA に備わっている取出しインタフェースは次のとおりです。

- プログラミング用 : OPEN、SET_FILTER、SET_COUNT、GET_QUERY、SET_PARSE_ITEM、ADD_TRANSFORM、SET_TRANSFORM_PARAM、FETCH_xxx および CLOSE は複数のオブジェクトを取り出します。
- SQL 問合せおよびブラウズ用 : GET_XML および GET_DDL は、単一の名前付きオブジェクトのメタデータを戻します。GET_DEPENDENT_XML、GET_DEPENDENT_DDL、GET_GRANTED_XML および GET_GRANTED_DDL インタフェースは、1 つ以上の依存オブジェクトまたは権限付与オブジェクトのメタデータを戻します。

この章では、次の項目について説明します。

DBMS_METADATA サブプログラムの要約

表 30-1 は、DBMS_METADATA サブプログラムの要約です。

表 30-1 DBMS_METADATA のサブプログラム

サブプログラム	説明
「OPEN プロシージャ」30-3 ページ	取り出すオブジェクトのタイプ、メタデータのバージョンおよびオブジェクト・モデルを指定します。
「SET_FILTER プロシージャ」30-7 ページ	取り出すオブジェクトに関する制限事項（オブジェクト名またはスキーマなど）を指定します。
「SET_COUNT プロシージャ」30-13 ページ	単一の FETCH_xxx コールで取り出されるオブジェクトの最大数を指定します。
「GET_QUERY プロシージャ」30-14 ページ	FETCH_xxx により使用される問合せのテキストを戻します。
「SET_PARSE_ITEM プロシージャ」30-14 ページ	解析して戻されるオブジェクト属性を指定することにより、出力を解析できるようにします。
「ADD_TRANSFORM プロシージャ」30-16 ページ	FETCH_xxx が取り出されたオブジェクトの XML 表示に適用する変換を指定します。
「SET_TRANSFORM_PARAM プロシージャ」30-18 ページ	transform_handle により識別された XSLT スタイルシートにパラメータを指定します。
「FETCH_xxx プロシージャ」30-22 ページ	OPEN、SET_FILTER、SET_COUNT、ADD_TRANSFORM などにより確立された条件を満たすオブジェクトのメタデータを戻します。
「CLOSE プロシージャ」30-25 ページ	OPEN により戻されるハンドルを無効化し、関連する状態をクリーンアップします。
「GET_XML および GET_DDL ファンクション」30-30 ページ	XML または DDL として指定されたオブジェクトのメタデータを戻します。
「GET_DEPENDENT_XML および GET_DEPENDENT_DDL ファンクション」30-32 ページ	XML または DDL として指定された 1 つ以上の依存オブジェクトのメタデータを戻します。
「GET_GRANTED_XML および GET_GRANTED_DDL ファンクション」30-34 ページ	XML または DDL として指定された 1 つ以上の権限付与オブジェクトのメタデータを戻します。

OPEN プロシージャ

OPEN は、取り出すオブジェクトのタイプ、メタデータのバージョンおよびオブジェクト・モデルを指定します。戻り値は、後続のコールで使用されるオブジェクトの不透明なコンテキスト・ハンドルです。

構文

```
DBMS_METADATA.OPEN (
   object_type  IN VARCHAR2,
   version      IN VARCHAR2 DEFAULT 'COMPATIBLE',
   model        IN VARCHAR2 DEFAULT 'ORACLE', )
RETURN NUMBER;
```

パラメータ

表 30-2 に、OPEN プロシージャのパラメータを示します。

表 30-2 Open() パラメータ

パラメータ	説明
object_type	取り出されるオブジェクトのタイプ。表 30-3 に、有効なタイプ名およびその意味を示します。Oracle9*i* では、これらのオブジェクト・タイプはメタデータの Oracle モデル向けにサポートされます（この表の「model」を参照）。将来のモデルでは、別のオブジェクト・タイプもサポートされる可能性があります。 「属性」列は、いくつかのオブジェクト・タイプ属性を示します。表などのスキーマ・オブジェクトはスキーマに属しています。名前付きオブジェクトには一意の名前があります（スキーマ・オブジェクトの場合、名前はスキーマ内で一意です）。索引などの依存オブジェクトは、ベース・スキーマ・オブジェクトに関して定義されます。権限付与オブジェクトは、ユーザーまたはロールに付与される（割り当てられる）ため、名前付きの権限受領者を持っています。 オブジェクトの選定基準を選択する場合、このような差異が関連します。詳細は、30-7 ページの「SET_FILTER プロシージャ」を参照してください。

表 30-2 Open() パラメータ（続き）

パラメータ	説明
version	抽出されるメタデータのバージョン。バージョンと互換性のないデータベース・オブジェクトまたは属性は抽出されません。このパラメータに対する適正値は次のとおりです。 COMPATIBLE（デフォルト）－データベースの互換性レベルに対応するメタデータのバージョン。データベースの互換性は 9.0.1 以上に設定する必要があります。 LATEST －データベース・バージョンに対応するメタデータのバージョン。 9.0.1 など特定のデータベース・バージョン。
model	使用するビューを指定します。API はメタデータに対し複数のビューをサポートするためです。Oracle9*i* でサポートされているのは ORACLE モデルのみです。

表 30-3 に、DBMS_METADATA パッケージのオブジェクト・タイプの名前、意味、属性および注意点を示します。属性列の S はスキーマ・オブジェクト、N は名前付きオブジェクト、D は依存オブジェクトおよび G は権限付与オブジェクトを表します。

表 30-3 DBMS_METADATA: オブジェクト・タイプ

タイプ名	意味	属性	注意
ASSOCIATION	関連統計情報	D	
AUDIT	SQL 文の監査	DG	依存オブジェクト、権限付与オブジェクトとしてモデル化されます。ベース・オブジェクト名は、文監査オプション名（例 : ALTER SYSTEM）です。ベース・オブジェクト・スキーマはありません。権限受領者は、その文が監査されるユーザーまたはプロキシです。
AUDIT_OBJ	スキーマ・オブジェクトの監査	D	なし
CLUSTER	クラスタ	SN	なし
COMMENT	コメント	D	なし

表 30-3 DBMS_METADATA: オブジェクト・タイプ（続き）

タイプ名	意味	属性	注意
CONSTRAINT	制約	SND	次のものは含まれません。 ■ IOT の主キー制約 ■ 列の NOT NULL 制約 ■ REF 列が含まれている表に対する特定の REF SCOPE および WITH ROWID 制約
CONTEXT	アプリケーション・コンテキスト	N	なし
DB_LINK	データベース・リンク	SN	データベース・リンクには所有者がいるため、スキーマ・オブジェクトとしてモデル化されます。パブリック・リンクの場合、所有者は PUBLIC です。プライベート・リンクの場合は、作成者が所有者です。
DEFAULT_ROLE	デフォルト・ロール	G	ALTER USER によってユーザーに付与されます。
DIMENSION	ディメンション	SN	なし
DIRECTORY	ディレクトリ	N	なし
FUNCTION	ストアド・ファンクション	SN	なし
INDEX	索引	SND	なし
INDEXTYPE	索引タイプ	SN	なし
JAVA_SOURCE	Java ソース	SN	なし
LIBRARY	外部プロシージャ・ライブラリ	SN	なし
MATERIALIZED_VIEW	マテリアライズド・ビュー	SN	なし
MATERIALIZED_VIEW_LOG	マテリアライズド・ビュー・ログ	D	なし
OBJECT_GRANT	オブジェクト権限付与	DG	なし
OPERATOR	演算子	SN	なし
OUTLINE	ストアド・アウトライン	N	なし
PACKAGE	ストアド・パッケージ	SN	デフォルトでは、パッケージ仕様部およびパッケージ本体の両方が取り出されます。30-7 ページの「SET_FILTER プロシージャ」を参照してください。

表 30-3 DBMS_METADATA: オブジェクト・タイプ（続き）

タイプ名	意味	属性	注意
PACKAGE_SPEC	パッケージ仕様部	SN	なし
PACKAGE_BODY	パッケージ本体	SN	なし
PROCEDURE	ストアド・プロシージャ	SN	なし
PROFILE	プロファイル	N	なし
PROXY	プロキシ認証	G	ALTER USER によってユーザーに付与されます。
REF_CONSTRAINT	参照制約	SND	なし
ROLE	ロール	N	なし
ROLE_GRANT	ロール権限付与	G	なし
ROLLBACK_SEGMENT	ロールバック・セグメント	N	なし
SEQUENCE	順序	SN	なし
SYNONYM	シノニム	注意を参照。	プライベート・シノニムはスキーマ・オブジェクトです。パブリック・シノニムはスキーマ・オブジェクトではありませんが、この API でのスキーマ名は PUBLIC です。シノニム名はシノニム自体であるとみなされます。たとえば、CREATE PUBLIC SYNONYM FOO FOR BAR では、結果として生じるオブジェクトに FOO という名前および PUBLIC というスキーマがあるとみなされます。
SYSTEM_GRANT	システム権限付与	G	なし
TABLE	表	SN	なし
TABLESPACE	表領域	N	なし
TABLESPACE_QUOTA	表領域割当て制限	G	ALTER USER を使用して付与されます。
TRIGGER	トリガー	SND	なし
TRUSTED_DB_LINK	信頼されているリンク	N	なし
TYPE	ユーザー定義型	SN	デフォルトでは、タイプおよびタイプ本体の両方が取り出されます。30-7 ページの「SET_FILTER プロシージャ」を参照してください。
TYPE_SPEC	タイプ指定	SN	なし
TYPE_BODY	タイプ本体	SN	なし

表 30-3 DBMS_METADATA: オブジェクト・タイプ（続き）

タイプ名	意味	属性	注意
USER	ユーザー	N	なし
VIEW	ビュー	SN	なし
XMLSCHEMA	XML Schema	SN	オブジェクト名はその URL です（30 文字を超える可能性があります）。そのスキーマは、登録したユーザーです。

戻り値

オブジェクトのクラスに対する不透明なハンドル。このハンドルは、SET_FILTER、SET_COUNT、ADD_TRANSFORM、GET_QUERY、SET_PARSE_ITEM、FETCH_xxx および CLOSE への入力として使用されます。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- INVALID_OBJECT_PARAM。version または model パラメータが object_type に対して有効でありません。

SET_FILTER プロシージャ

SET_FILTER は、取り出すオブジェクトに関する制限事項（オブジェクト名またはスキーマなど）を指定します。

構文

```
DBMS_METADATA.SET_FILTER (
   handle  IN NUMBER,
   name    IN VARCHAR2,
   value   IN VARCHAR2);
DBMS_METADATA.SET_FILTER (
   handle  IN NUMBER,
   name    IN VARCHAR2,
   value   IN BOOLEAN DEFAULT TRUE);
```

パラメータ

表 30-4 に、SET_FILTER プロシージャのパラメータを示します。

表 30-4 SET_FILTER パラメータ

パラメータ	説明
handle	OPEN から戻されたハンドル。
name	フィルタ名。表 30-5 に、各フィルタが適用する object_type、名前、データ・タイプ（テキストまたはブール）およびその意味または効果（設定されている場合はデフォルト値を含む）を示します。
value	フィルタの値。

表 30-5 では、SET_FILTER プロシージャで使用可能なフィルタのオブジェクト・タイプ、名前、データ・タイプおよび意味について説明します。

表 30-5 SET_FILTER: フィルタ

オブジェクト・タイプ	名前	データ・タイプ	意味
名前付きオブジェクト	NAME	テキスト	この名前のオブジェクトが選択されています。
	NAME_EXPR	テキスト	フィルタの値は SQL 比較の右側、つまり、SQL 比較演算子（=、!= など）と比較対象の値の右側にあります。値には、適切な場所にカッコおよび引用符が含まれている必要があります。PL/SQL と SQL*Plus では、アポストロフィを表現するには 2 つの引用符（二重引用符とは異なります）が必要です。たとえば、次のようにします。 'IN (''DEPT'',''EMP'')' フィルタの値がオブジェクト名に対応するオブジェクト属性と組み合され、オブジェクトをフェッチする問合せにおいて WHERE 条件を生成しています。前述の例では、DEPT および EMP という名前のオブジェクトが取り出されます。 デフォルトでは、object_type の名前付きオブジェクトがすべて選択されます。

表 30-5 SET_FILTER: フィルタ（続き）

オブジェクト・タイプ	名前	データ・タイプ	意味
スキーマ・オブジェクト	SCHEMA	テキスト	このスキーマのオブジェクトが選択されています。
	SCHEMA_EXPR	テキスト	フィルタの値は、SQL 比較の右側です。フィルタの値がオブジェクト・スキーマに対応するオブジェクト属性と組み合され、オブジェクトをフェッチする問合せにおいて WHERE 条件を生成しています。構文の詳細は、NAME_EXPR を参照してください。 デフォルト： － BASE_OBJECT_SCHEMA が指定されている場合、そのスキーマのオブジェクトが選択されています。 － そうでない場合は、現行スキーマのオブジェクトが選択されています。 30-12 ページの「セキュリティ」を参照してください。
PACKAGE、TYPE	SPECIFICATION	ブール	TRUE の場合、パッケージまたはタイプの仕様部を取り出します。デフォルトは TRUE です。
	BODY	ブール	TRUE の場合、パッケージまたはタイプの本体を取り出します。デフォルトは TRUE です。
TABLE	TABLESPACE	テキスト	この表領域のオブジェクト（または、この表領域にパーティションがあるオブジェクト）が選択されています。
	TABLESPACE_EXPR	テキスト	フィルタの値は、SQL 比較の右側です。フィルタの値がオブジェクトの表領域（またはパーティション表の場合はそのパーティションの表領域）に対応する属性と組み合され、オブジェクトをフェッチする問合せにおいて WHERE 条件を生成しています。構文の詳細は、NAME_EXPR を参照してください。デフォルトでは、すべての表領域のオブジェクトが選択されています。

表 30-5 SET_FILTER: フィルタ（続き）

オブジェクト・タイプ	名前	データ・タイプ	意味
依存オブジェクト	BASE_OBJECT_NAME	テキスト	この名前のオブジェクトに定義または付与されたオブジェクトが選択されています。スキーマのトリガーの場合は、SCHEMA を指定してください。データベースのトリガーの場合は、DATABASE を指定してください。列レベルのコメントを列名で選択することはできません。ベース・オブジェクト名は、列が含まれている表、ビューまたはマテリアライズド・ビューの名前であることが必要です。
	BASE_OBJECT_SCHEMA	テキスト	このスキーマ内のオブジェクトに定義または付与されたオブジェクトが選択されています。BASE_OBJECT_NAME に SCHEMA または DATABASE 以外の値が指定されている場合、その値が現行スキーマのデフォルトです。
INDEX、TRIGGER	SYSTEM_GENERATED	ブール	TRUE の場合、システム生成の場合でも索引またはトリガーを選択します。FALSE の場合、システム生成の索引またはトリガーを省略します。デフォルトは TRUE です。
権限付与オブジェクト	GRANTEE	テキスト	ユーザーまたはロールに付与されたオブジェクトが選択されています。PUBLIC への付与の場合は、PUBLIC を指定します。
OBJECT_GRANT	GRANTOR	テキスト	このユーザーにより付与されたオブジェクト権限付与が選択されています。

表 30-5 SET_FILTER: フィルタ（続き）

オブジェクト・タイプ	名前	データ・タイプ	意味
SYNONYM、JAVA_SOURCE、XMLSCHEMA	LONGNAME	テキスト	30 文字を超える名前。この名前のオブジェクトが選択されています。オブジェクト名が 30 文字以内である場合は、NAME フィルタを使用する必要があります。
	LONGNAME_EXPR	テキスト	フィルタの値は、SQL 比較の右側です。フィルタの値がオブジェクトの長い名前に対応する属性と組み合され、オブジェクトをフェッチする問合せにおいて WHERE 条件を生成しています。構文の詳細は、NAME_EXPR を参照してください。デフォルトでは、オブジェクトの長い名前に対してフィルタ処理は行われません。
全オブジェクト	CUSTOM_FILTER	テキスト	WHERE 条件のテキスト。オブジェクトをフェッチする問合せに条件が追加されています。デフォルトでは、カスタム・フィルタは使用されません。その他のフィルタは、多数のユーザーのニーズを満たすことを目的としています。目的に合ったフィルタが定義されていない場合は、CUSTOM_FILTER を使用してください。フィルタの必要性は、UDT の詳細な構造および admin/catmeta.sql で定義された問合せで使用されるビューにより異なります。フィルタはバージョンごとに変更される可能性があるため、上位互換性は保証されません。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- INVALID_OPERATION。OPEN コンテキストの FETCH_xxx に対する最初のコールの後、SET_FILTER がコールされました。FETCH_xxx への最初のコールの後、現行の OPEN コンテキストに対し、それ以上 SET_FILTER へのコールを行うことはできません。
- INCONSISTENT_ARGS。name フィルタが OPEN コンテキストに関連付けられたオブジェクト・タイプに対し有効でないか、フィルタの値が不適切なデータ・タイプです。

セキュリティ

SET_FILTER を使用すると、取り出されるオブジェクトのスキーマを指定できますが、セキュリティの考慮事項がこの仕様部を上書きする可能性があります。コール元が SYS であるか、または SELECT_CATALOG_ROLE を付与されている場合、どのようなオブジェクトでも取り出せます。そうでない場合は、次のオブジェクトのみを取り出せます。

- コール元が所有するスキーマ・オブジェクト
- パブリック・シノニム
- コール元または PUBLIC に付与されたシステム権限
- コール元が所有者、付与者または被付与者（明示的または PUBLIC）であるオブジェクトへの付与

取出しの権限が付与されていないオブジェクトを要求した場合、例外は発生しません。ただし、オブジェクトは存在しないかのようになり、取り出されません。

使用上の注意

- 同じテキスト式のフィルタを、異なる値で複数回使用できます。すべてのフィルタ条件が問合せに適用されます。たとえば、Felix と Oscar の間の名前を持つオブジェクトを取得するには、次のように記述します。

```
dbms_metadata.set_filter(handle,'NAME_EXPR','>=''FELIX''');
dbms_metadata.set_filter(handle,'NAME_EXPR','<=''OSCAR''');
```

- トリガー、権限付与および索引などの依存オブジェクトに対しては、次の条件が適用されます。
  - 非権限ユーザーとして接続する場合－ BASE_OBJECT_NAME がフィルタとして指定された場合、BASE_OBJECT_SCHEMA が現行スキーマのデフォルトです。

    ```
    dbms_metadata.set_filter(h,'BASE_OBJECT_NAME','EMP');
    ```

  - SELECT_CATALOG_ROLE の権限があるユーザーとして接続する場合－ BASE_OBJECT_SCHEMA が指定されている場合は、これがスキーマのデフォルトです。そうでない場合は現行のスキーマがデフォルトです。たとえば、SCOTT.EMP に定義された SCOTT の索引すべてを参照するには、次のようなフィルタになります。

    ```
    dbms_metadata.set_filter(h,'BASE_OBJECT_NAME','EMP');
    dbms_metadata.set_filter(h,'BASE_OBJECT_SCHEMA','SCOTT');
    ```

    その他のスキーマの索引を参照するには、次のようなフィルタになります。

    ```
    dbms_metadata.set_filter(h,'SCHEMA_EXPR','LIKE ''%''');
    ```

索引およびトリガーにはシステム生成されたものもあります（一意の制約事項を実行するために使用される索引などが該当します）。これらを抽出しないようにするには、SYSTEM_GENERATED フィルタを FALSE に設定します。

SET_COUNT プロシージャ

SET_COUNT は、単一の FETCH_xxx コールで取り出されるオブジェクトの最大数を指定します。デフォルトでは、FETCH_xxx へのコールで戻されるオブジェクトは 1 つです。SET_COUNT を使用すると、このデフォルトを上書きできます。FETCH_xxx がクライアントからコールされた場合、2 以上のカウント値を指定するとサーバー・ラウンドトリップが減少するため、パフォーマンスが改善されます。NULL が戻されるとプロシージャは停止しますが、オブジェクトの最大数未満の値が戻された場合は停止しません。

構文

```
DBMS_METADATA.SET_COUNT (
   handle IN NUMBER,
   value  IN NUMBER);
```

パラメータ

表 30-6 に、SET_COUNT プロシージャのパラメータを示します。

表 30-6 SET_COUNT パラメータ

パラメータ	説明
handle	OPEN から戻されたハンドル。
value	取り出すオブジェクトの数。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- INVALID_OPERATION。OPEN コンテキストの FETCH_xxx に対する最初のコールの後、SET_COUNT がコールされました。FETCH_xxx への最初のコールの後、現行の OPEN コンテキストに対し、それ以上 SET_COUNT へのコールを行うことはできません。

GET_QUERY プロシージャ

GET_QUERY は、FETCH_xxx により使用される問合せのテキストを戻します。このファンクションは、デバッグにおいて役立ちます。

構文

```
DBMS_METADATA.GET_QUERY (
   handle  IN NUMBER)
RETURN VARCHAR2;
```

パラメータ

表 30-7 に、GET_QUERY プロシージャのパラメータを示します。

表 30-7 GET_QUERY パラメータ

パラメータ	説明
handle	OPEN から戻されたハンドル。

戻り値

FETCH_xxx により使用される問合せのテキスト。

例外

- INVALID_ARGVAL。handle パラメータに NULL または無効な値が指定されています。

SET_PARSE_ITEM プロシージャ

SET_PARSE_ITEM を使用すると、解析して戻されるオブジェクト属性を指定することにより、出力を解析できます。FETCH_DDL とともに使用します。単独では使用できません。

構文

```
DBMS_METADATA.SET_PARSE_ITEM (
   handle   IN NUMBER,
   name     IN VARCHAR2);
```

パラメータ

表 30-8 に、SET_PARSE_ITEM プロシージャのパラメータを示します。

表 30-8 SET_PARSE_ITEM パラメータ

パラメータ	説明
handle	OPEN から戻されたハンドル。
name	解析して戻されるオブジェクト属性の名前。オブジェクト・タイプ、名前および意味は、表 30-9 を参照してください。

表 30-9 では、SET_PARSE_ITEM プロシージャで使用可能な項目のオブジェクト・タイプ、名前および意味について説明します。

表 30-9 SET_PARSE_ITEM: 解析項目

オブジェクト・タイプ	名前	意味
全オブジェクト	VERB	fetch_ddl により戻されるネストした表 sys.ku$_ddls の各行に対し、対応する ddlText の verb が戻されます。ddlText が SQL DDL 文の場合は、SQL verb（例 : CREATE、GRANT、AUDIT）が戻されます。ddlText がプロシージャ・コール（例 : DBMS_RLS.ADD_POLICY_CONTEXT）の場合は、package.procedure-name が戻されます。
	OBJECT_TYPE	ddlText が SQL DDL 文で、その verb が CREATE または ALTER の場合は、DDL 文で使用されているとおりのオブジェクト・タイプ（例 : TABLE、PACKAGE BODY）が戻されます。それ以外の場合は、表 30-3「DBMS_METADATA: オブジェクト・タイプ」のオブジェクト・タイプ名が戻されます。
	SCHEMA	オブジェクト・スキーマが戻されます。オブジェクトがスキーマ・オブジェクトでない場合、NULL が戻されます。
	NAME	オブジェクト名が戻されます。オブジェクトが名前付きオブジェクトでない場合、NULL が戻されます。
TABLE、INDEX	TABLESPACE	表または索引の表領域名が戻されます。
TRIGGER	ENABLE	トリガーが有効にされている場合、ENABLE が戻されます。トリガーが無効にされている場合、DISABLE が戻されます。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- INVALID_OPERATION。OPEN コンテキストの FETCH_xxx に対する最初のコールの後、SET_PARSE_ITEM がコールされました。FETCH_xxx への最初のコールの後、それ以上 SET_PARSE_ITEM へのコールを行うことはできません。
- INCONSISTENT_ARGS。属性 name が OPEN コンテキストに関連付けられたオブジェクト・タイプに対し有効でありません。

使用上の注意

デフォルトでは、fetch_ddl は DDL の作成時にオブジェクトのメタデータを戻します。SET_PARSE_ITEM をコールすることにより、個々のオブジェクトの属性を戻すように要求し、SQL テキストの解析プロセスが冗長になることを回避できます。これは、戻された表の索引のフェッチなど、戻されたオブジェクトの値に基づいてオブジェクトをフェッチする場合に便利です。

SET_PARSE_ITEM を複数回コールして、複数の項目を解析して戻すように要求できます。解析された項目は、ネストした表 sys.ku$_parsed_items に戻されます。sys.ku$_parsed_items の使用例は、30-26 ページの「例 : PAYROLL 表およびその索引を DDL として取出し」を参照してください。

関連項目 :

- 30-22 ページ「FETCH_xxx プロシージャ」
- メタデータ API の使用方法の詳細は、『Oracle9*i* データベース・ユーティリティ』を参照してください。

ADD_TRANSFORM プロシージャ

ADD_TRANSFORM は、FETCH_xxx が取り出されたオブジェクトの XML 表示に適用する変換を指定します。複数の変換を追加することが可能です。

構文

```
DBMS_METADATA.ADD_TRANSFORM (
   handle    IN NUMBER,
   name      IN VARCHAR2,
   encoding  IN VARCHAR2 DEFAULT NULL)
RETURN NUMBER;
```

パラメータ

表 30-10 に、ADD_TRANSFORM プロシージャのパラメータを示します。

表 30-10 ADD_TRANSFORM パラメータ

パラメータ	説明
handle	OPEN から戻されたハンドル。
name	変換名。名前が DDL である場合、DDL を作成すると、Oracle ディクショナリ内に格納された XSLT スタイルシートを使用して生成されます。名前にピリオド（.）、コロン（:）またはスラッシュ（/）が含まれている場合、ユーザー提供の XSLT スタイルシートの URL として解釈されます。『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』を参照してください。
encoding	名前に指されたスタイルシートがエンコードされている NLS キャラクタ・セット名（『National Language Support Guide』を参照）。これは、名前が URL である場合にのみ有効です。NULL の値が残存し、URL がデータベース外部（/usr/williams/xsl/mystylesheet.xsl）である場合、UTF-8 エンコーディングとみなされます。NULL の値が残存し、URL が /oradb/ で始まるデータベース内部である場合、データベース・キャラクタ・セットはエンコード対象であるとみなされます。

戻り値

変換に対する不透明なハンドル。このハンドルは、SET_TRANSFORM_PARAM への入力として使用されます。このハンドルは OPEN により戻されるハンドルとは異なり、取り出されたオブジェクトではなく変換を参照します。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- INVALID_OPERATION。OPEN コンテキストの FETCH_xxx に対する最初のコールの後、ADD_TRANSFORM がコールされました。FETCH_xxx への最初のコールの後、現行の OPEN コンテキストに対し、それ以上 ADD_TRANSFORM へのコールを行うことはできません。

使用上の注意

変換が追加されていない場合、デフォルトでは XML 文書としてオブジェクトが戻されます。ADD_TRANSFORM をコールして XSLT スタイルシートを指定し、戻された文書を変換します。

ADD_TRANSFORM を複数回コールし、戻された XML 文書に対し複数の変換を適用できます。FETCH_xxx は、指定された順序で変換を適用します。たとえば、使用中の最初の変換の出力は 2 番目の入力となります。

エンコーディング・パラメータは、次のいずれかが該当する必要があります。

- 外部 URL に指された XSL スタイルシートは、UTF-8 のサブセットでないキャラクタ・セットでエンコードされます。
- データベース内部の URL に指された XSL スタイルシートは、データベース・キャラクタ・セットのサブセットでないキャラクタ・セットでエンコードされます。

後者は、NCLOB または NVARCHAR 列を指したデータベース内部の URL である場合もあります。US7ASCII（該当する場合）に対して明示的な設定を行うと、XML 解析のパフォーマンスがわずかに改善されますが、通常これを指定する必要はありません。

注意： DDL 変換の出力は XML 文書ではありません。したがって、DDL 変換の後に変換を追加しないでください。

SET_TRANSFORM_PARAM プロシージャ

SET_TRANSFORM_PARAM は、transform_handle により識別された XSLT スタイルシートに対し、パラメータを指定します。これを使用して、変換の出力を変更またはカスタマイズします。

構文

```
DBMS_METADATA.SET_TRANSFORM_PARAM (
   transform_handle  IN NUMBER,
   name                    IN VARCHAR2,
   value                   IN VARCHAR2);
DBMS_METADATA.SET_TRANSFORM_PARAM (
   transform_handle  IN NUMBER,
   name                    IN VARCHAR2,
   value                   IN BOOLEAN DEFAULT TRUE);
```

パラメータ

表 30-11 に、SET_TRANSFORM_PARAM プロシージャのパラメータを示します。

表 30-11 SET_TRANSFORM_PARAM パラメータ

パラメータ	説明
transform_handle	(1) ADD_TRANSFORM から戻されたハンドル、または (2) セッション全体の DDL 変換を指定する列挙定数 SESSION_TRANSFORM。OPEN により戻されたハンドルは、有効な変換ハンドルではありません。
name	パラメータ名。表 30-12 に、DDL 変換に対して定義された変換パラメータを示します。適用する object_type、データ・タイプ（この場合は常にブール）およびその意味または効果（設定されている場合はデフォルト値を含む）を示します。
value	変換の値。

表 30-12 では、SET_TRANSFORM_PARAM プロシージャの DDL 変換のパラメータのオブジェクト・タイプ、名前、データ・タイプおよび意味について説明します。

表 30-12 SET_TRANSFORM_PARAM: DDL 変換の変換パラメータ

オブジェクト・タイプ	名前	データ・タイプ	意味
全オブジェクト	PRETTY	ブール	TRUE である場合、インデントおよび改行が設定された出力をフォーマットします。デフォルトは TRUE です。
	SQLTERMINATOR	ブール	TRUE の場合、各 DDL 文に SQL 終了記号（; または /）を追加します。デフォルトは FALSE です。
TABLE	SEGMENT_ATTRIBUTES	ブール	TRUE の場合、セグメント属性（物理的な属性、記憶域の属性、表領域、ロギング）を発行します。デフォルトは TRUE です。
	STORAGE	ブール	TRUE の場合、記憶域句を発行します（SEGMENT_ATTRIBUTES が FALSE である場合には無視されます）。デフォルトは TRUE です。
	TABLESPACE	ブール	TRUE の場合、表領域を発行します（SEGMENT_ATTRIBUTES が FALSE である場合には無視されます）。デフォルトは TRUE です。

表 30-12 SET_TRANSFORM_PARAM: DDL 変換の変換パラメータ（続き）

オブジェクト・タイプ	名前	データ・タイプ	意味
TABLE	CONSTRAINTS	ブール	TRUE の場合、非参照表制約をすべて発行します。デフォルトは TRUE です。
	REF_CONSTRAINTS	ブール	TRUE の場合、参照制約（外部キーおよび有効範囲付 REF）をすべて発行します。デフォルトは TRUE です。
	CONSTRAINTS_AS_ALTER	ブール	TRUE の場合、別の ALTER TABLE 文（必要に応じて CREATE INDEX）として表制約を発行します。FALSE の場合、CREATE TABLE 文の一部として表制約を指定します。デフォルトは FALSE です。CONSTRAINTS が TRUE である必要があります。
	OID	ブール	TRUE の場合、オブジェクト表に対し OID 句を発行します。デフォルトは FALSE です。
	SIZE_BYTE_KEYWORD	ブール	TRUE の場合、バイト・セマンティクスを使用する CHAR および VARCHAR2 列のサイズ仕様の一部として、BYTE キーワードを発行します。FALSE の場合、キーワードを省略します。デフォルトは FALSE です。
INDEX	SEGMENT_ATTRIBUTES	ブール	TRUE の場合、セグメント属性（物理的な属性、記憶域の属性、表領域、ロギング）を発行します。デフォルトは TRUE です。
	STORAGE	ブール	TRUE の場合、記憶域句を発行します（SEGMENT_ATTRIBUTES が FALSE である場合には無視されます）。デフォルトは TRUE です。
	TABLESPACE	ブール	TRUE の場合、表領域を発行します（SEGMENT_ATTRIBUTES が FALSE である場合には無視されます）。デフォルトは TRUE です。
TYPE	SPECIFICATION	ブール	TRUE の場合、タイプ仕様部を発行します。デフォルトは TRUE です。
	BODY	ブール	TRUE の場合、タイプ本体を発行します。デフォルトは TRUE です。
PACKAGE	SPECIFICATION	ブール	TRUE の場合、パッケージ仕様部を発行します。デフォルトは TRUE です。
	BODY	ブール	TRUE の場合、パッケージ本体を発行します。デフォルトは TRUE です。

表 30-12 SET_TRANSFORM_PARAM: DDL 変換の変換パラメータ（続き）

オブジェクト・タイプ	名前	データ・タイプ	意味
VIEW	FORCE	ブール	TRUE の場合、CREATE VIEW 文で FORCE キーワードを使用します。デフォルトは TRUE です。
全オブジェクト	DEFAULT	ブール	TRUE に設定されたこのパラメータを使用して SET_TRANSFORM_PARAM をコールすると、変換のパラメータがすべてデフォルトにリセットされます。FALSE に設定した場合、影響はありません。デフォルトはありません。
	INHERIT	ブール	TRUE の場合、セッションレベルのパラメータを継承します。デフォルトは FALSE です。アプリケーションが ADD_TRANSFORM をコールして DDL 変換を追加した場合、デフォルトでは適用する変換パラメータのみがその変換ハンドルの明示的なセットとなります。変換ハンドルがセッション変換ハンドルである場合には、影響はありません。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- INVALID_OPERATION。OPEN コンテキストの FETCH_xxx に対する最初のコールの後、SET_TRANSFORM_PARAM がコールされました。FETCH_xxx への最初のコールの後、それ以上 SET_TRANSFORM_PARAM へのコールを行うことはできません。
- INCONSISTENT_ARGS。この変換パラメータの name が OPEN コンテキストに関連付けられたオブジェクト・タイプに対し有効でありません。

使用上の注意

XSLT を使用すると、パラメータをスタイルシートに渡すことができます。SET_TRANSFORM_PARAM をコールして、transform_handle により識別されたスタイルシートに渡されるパラメータの値を指定します。スタイルシートのパラメータ値を指定する場合、テキスト文字列が最も一般的な方法です。ただし、DDL 変換の場合、一定のパラメータをブールとして発行すると便利です。結果として、2 種類のプロシージャが提供されることになります。

GET_DDL ファンクションを使用すると、日常的に使用しているブラウザでオブジェクトに対し DDL 作成を抽出できます。変換パラメータを指定できるようにするために、このパッケージは、DDL 変換のハンドルとしてセッション・レベルで列挙定数 SESSION_TRANSFORM を定義します。DBMS_METADATA.SESSION_TRANSFORM を変換ハンドルとして使用して SET_TRANSFORM_PARAM をコールし、セッション全体の変換パラ

メータを設定できます。GET_DDL が DDL 変換を起動する場合、これらのパラメータを継承します。

注意： 列挙定数にはパッケージ名 DBMS_METADATA.SESSION_TRANSFORM を接頭辞として付加する必要があります。

FETCH_xxx プロシージャ

FETCH_xxx は、OPEN、SET_FILTER、SET_COUNT、ADD_TRANSFORM などにより確立された条件を満たすオブジェクトのメタデータを戻します。異形については、30-23 ページの「使用上の注意」を参照してください。

構文

FETCH のファンクションおよびプロシージャは次のとおりです。

```
DBMS_METADATA.FETCH_XML (
   handle  IN NUMBER)
RETURN sys.XMLType;
```

関連項目： XMLType の説明は、『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』を参照してください。

```
DBMS_METADATA.FETCH_DDL (
   handle  IN NUMBER)
RETURN sys.ku$_ddls;
```

次のタイプは、ネストした表タイプ sys.ku$_ddls を導出します。

```
TYPE sys.ku$_parsed_item AS OBJECT (
  item       VARCHAR2(30),
  value      VARCHAR2(4000),
  object-row NUMBER );
TYPE sys.ku$_parsed_items IS TABLE OF sys.ku$_parsed_item;
TYPE sys.ku$_ddl AS OBJECT (
  ddlText CLOB,
  parsedItems sys.ku$_parsed_items );
TYPE sys.ku$_ddls IS TABLE OF sys.ku$_ddl;

DBMS_METADATA.FETCH_CLOB (
   handle  IN NUMBER)
RETURN CLOB;
DBMS_METADATA.FETCH_CLOB (
   handle  IN NUMBER,
   doc     IN OUT NOCOPY CLOB);
```

パラメータ

表 30-13 に、FETCH_xxx プロシージャのパラメータを示します。

表 30-13 FETCH_xxx パラメータ

パラメータ	説明
handle	OPEN から戻されたハンドル。
doc（プロシージャ fetch_clob）	すべてのオブジェクトが戻された場合、オブジェクトのメタデータまたは NULL。

戻り値

すべてのオブジェクトが戻された場合、オブジェクトのメタデータまたは NULL。

例外

問合せの実行中に発生した例外の大部分は、コール元に伝播されます。また、次の例外が発生する場合があります。

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- INCONSISTENT_OPERATION。(1) DDL 変換が指定された後で FETCH_XML がコールされたか、(2) DDL 変換が指定されていない時点で FETCH_DDL がコールされました。

使用上の注意

これらのファンクションおよびプロシージャは、OPEN、SET_FILTER、SET_COUNT、ADD_TRANSFORM などにより確立された条件を満たすオブジェクトのメタデータを戻します。FETCH_xxx への各コールは、全オブジェクトが戻されるまで SET_COUNT により指定されたオブジェクトの数（元のカーソルに残存するオブジェクトが少ない場合はそれ以下）を戻します。最後のオブジェクトが戻された後、FETCH_xxx への後続のコールが NULL を戻し、透過的にクローズされた OPEN によりストリームが作成されます。

FETCH_xxx のファンクションおよびプロシージャは複数あります。

- FETCH_XML は、XMLType としてオブジェクトの XML メタデータを戻します。変換が指定された場合、変換は XML 文書を生成すると想定します。特に、DDL 変換が指定されていないと想定します。
- FETCH_DDL は、ネストした表 sys.ku$_ddls の作成 DDL を戻します。DDL 変換が指定されたと想定します。ネストした表 sys.ku$_ddls の各行は、ddlText 列に単一の DDL 文を含みます。必要であれば、DDL 文に対して解析された項目が parsedItems 列に戻されます。次の状況では、複数の DDL 文が戻されます。
  - SET_COUNT をコールして 1 を超える件数を指定した場合。

- オブジェクトが複数の DDL 文に変換された場合。たとえば、TYPE オブジェクトを CREATE TYPE および CREATE TYPE BODY 文の両方に変換できます。TABLE オブジェクトを CREATE TABLE、0（ゼロ）以上の CREATE INDEX 文および 0（ゼロ）以上の ALTER TABLE 文に変換できます。

- FETCH_CLOB は、変換の有無にかかわらず、CLOB としてオブジェクトを戻します。

FETCH_CLOB は、ファンクションおよびプロシージャの両方の異形で使用されます。プロシージャの異形は、IN OUT NOCOPY パラメータの参照によりオブジェクトを戻します。

FETCH_xxx により戻される LOB はすべて一時 LOB です。LOB を解放する必要があります。XMLType オブジェクトに同じものが適用されます。

SET_PARSE_ITEM がコールされた場合、FETCH_DDL は、ネストした表 sys.ku$_parsed_items の DDL 文の属性を戻します。この表は、戻されたネストした表 sys.ku$_ddls の列です。ネストした表 sys.ku$_parsed_items の各行は、SET_PARSE_ITEM に指定された項目に対応し、次の列を含みます。

- item － SET_PARSE_ITEM に対し、name パラメータで指定した属性の名前。
- value －属性が DDL 文に存在しない場合、属性値または NULL。
- object-row －将来使用。

行の順序は決定されていません。特定の項目を探すには、item が一致する表を検索する必要があります。

SET_PARSE_ITEM がコールされなかった場合、ネストした表 sys.ku$_parsed_items の値として NULL が戻されます。

FETCH_xxx の異形がコールされた場合

OPEN により選択された全オブジェクトに対し、同形の FETCH_xxx がコールされると予測されています。つまり、プログラムは、同じ OPEN ハンドルを使用する FETCH_XML、FETCH_DDL および FETCH_CLOB へのコールを混合しません。異形をコールした場合の影響は定義されていません。予測と異なる可能性があります。

CLOSE プロシージャ

CLOSE は、OPEN により戻されるハンドルを無効化し、関連する状態をクリーンアップします。

構文

```
DBMS_METADATA.CLOSE (
   handle  IN NUMBER);
```

パラメータ

表 30-14 に、CLOSE プロシージャのパラメータを示します。

表 30-14 CLOSE パラメータ

パラメータ	説明
handle	OPEN から戻されたハンドル。

例外

- INVALID_ARGVAL。handle パラメータに対して指定した値が NULL または無効です。

使用上の注意

OPEN により確立されたオブジェクトのストリームを早期終了できます。

- FETCH_xxx へのコールがオブジェクトが存在しないことを示す NULL を戻した場合、CLOSE へのコールが透過的に行われます。この場合、ハンドルの CLOSE をコールでき、例外は発生しません（CLOSE へのコールは必須ではありません）。
- 1 つの特定オブジェクトのみが戻されるとわかっている場合、FETCH_xxx を一度コールしてハンドルに保持されたリソースを解放した後、CLOSE を明示的にコールしてください。

例 : PAYROLL 表およびその索引を DDL として取出し

この例は、名前が PAYROLL で始まる現行のスキーマのすべての表に対する作成 DDL を取り出します。また、各表に対し、表で定義された索引の作成 DDL を戻します。戻された DDL は出力ファイルに書き込まれます。

```
CREATE OR REPLACE PACKAGE dbms_metadata_example AS

   PROCEDURE get_payroll_tables;
END;
/
CREATE OR REPLACE PACKAGE BODY dbms_metadata_example AS

-- Global Variables

fileHandle   UTL_FILE.FILE_TYPE;

-- Exception initialization

file_not_found EXCEPTION;
PRAGMA EXCEPTION_INIT(file_not_found, -1309);

-- Package-private routine to write a CLOB to an output file.

  PROCEDURE write_lob(doc IN CLOB) IS

     outString     varchar2(32760);
     cloblen       number;
     offset        number := 1;
     amount        number;

BEGIN
  cloblen := dbms_lob.getlength(doc);
  WHILE cloblen > 0
  LOOP
    IF cloblen > 32760 THEN
      amount := 32760;
    ELSE
      amount := cloblen;
    END IF;
    outString := dbms_lob.substr(doc, amount, offset);
    utl_file.put(fileHandle, outString);
    utl_file.fflush(fileHandle);
    offset  := offset  + amount;
    cloblen := cloblen - amount;
  END LOOP;
  RETURN;
```

```
END;

-- Public routines

-- GET_PAYROLL_TABLES: Fetch DDL for payroll tables and their indexes.

PROCEDURE  get_payroll_tables IS

tableOpenHandle       NUMBER;
indexOpenHandle       NUMBER;
tableTransHandle      NUMBER;
indexTransHandle      NUMBER;
schemaName            VARCHAR2(30);
tableName             VARCHAR2(30);
tableDDLs             sys.ku$_ddls;
tableDDL              sys.ku$_ddl;
parsedItems           sys.ku$_parsed_items;
indexDDL              CLOB;

BEGIN

--  open the output file... note that the 1st param. (dir. path) must be
--  included in the database's UTL_FILE_DIR init. parameter.
--
  BEGIN
    fileHandle := utl_file.fopen('/private/xml', 'ddl.out', 'w', 32760);
  EXCEPTION
    WHEN OTHERS THEN
        RAISE file_not_found;
  END;

-- Open a handle for tables in the current schema.
  tableOpenHandle := dbms_metadata.open('TABLE');

-- Call 'set_count' to request retrieval of one table at a time.
-- This call is not actually necessary because 1 is the default.
  dbms_metadata.set_count(tableOpenHandle, 1);

-- Retrieve tables whose name starts with 'PAYROLL'. When the filter is
-- 'NAME_EXPR', the filter value string must include the SQL operator. This
-- gives the caller flexibility to use LIKE, IN, NOT IN, subqueries, and so on.
  dbms_metadata.set_filter(tableOpenHandle, 'NAME_EXPR', 'LIKE ''PAYROLL%''');

-- Tell Metadata API to parse out each table's schema and name separately
-- so we can use them to set up the calls to retrieve its indexes.
  dbms_metadata.set_parse_item(tableOpenHandle, 'SCHEMA');
  dbms_metadata.set_parse_item(tableOpenHandle, 'NAME');
```

```
-- Add the DDL transform so we get SQL creation DDL
  tableTransHandle := dbms_metadata.add_transform(tableOpenHandle, 'DDL');

-- Tell the XSL stylesheet we don't want physical storage information (storage,
-- tablespace, etc), and that we want a SQL terminator on each DDL. Notice that
-- these calls use the transform handle, not the open handle.
  dbms_metadata.set_transform_param(tableTransHandle,
      'SEGMENT_ATTRIBUTES', FALSE);
  dbms_metadata.set_transform_param(tableTransHandle,
      'SQLTERMINATOR', TRUE);

-- Ready to start fetching tables. We use the FETCH_DDL interface (rather than
-- FETCH_XML or FETCH_CLOB). This interface returns a SYS.KU$_DDLS; a table of
-- SYS.KU$_DDL objects. This is a table because some object types return
-- multiple DDL statements (like types / pkgs which have create header and
-- body statements). Each KU$_DDL has a CLOB containing the 'CREATE TABLE'
-- statement plus a nested table of the parse items specified. In our case,
-- we asked for two parse items; Schema and Name.

  LOOP
    tableDDLs := dbms_metadata.fetch_ddl(tableOpenHandle);
    EXIT WHEN tableDDLs IS NULL;   -- Get out when no more payroll tables

-- In our case, we know there is only one row in tableDDLs (a KU$_DDLS tbl obj)
-- for the current table. Sometimes tables have multiple DDL statements,
-- for example, if constraints are applied as ALTER TABLE statements,
-- but we didn't ask for that option.
-- So, rather than writing code to loop through tableDDLs,
-- we'll just work with the 1st row.
--
-- First, write the CREATE TABLE text to our output file, then retrieve the
-- parsed schema and table names.
    tableDDL := tableDDLs(1);
    write_lob(tableDDL.ddltext);
    parsedItems := tableDDL.parsedItems;

-- Must check the name of the returned parse items as ordering isn't guaranteed
    FOR i IN 1..2 LOOP
      IF parsedItems(i).item = 'SCHEMA'
      THEN
        schemaName := parsedItems(i).value;
      ELSE
        tableName  := parsedItems(i).value;
      END IF;
    END LOOP;
```

```
-- Then use the schema and table names to set up a 2nd stream for retrieval of
-- the current table's indexes.
-- (Note that we don't have to specify a SCHEMA filter for the indexes,
-- Because SCHEMA defaults to the value of BASE_OBJECT_SCHEMA.)
    indexOpenHandle := dbms_metadata.open('INDEX');
    dbms_metadata.set_filter(indexOpenHandle,'BASE_OBJECT_SCHEMA',schemaName);
    dbms_metadata.set_filter(indexOpenHandle,'BASE_OBJECT_NAME',tableName);

-- Add the DDL transform and set the same transform options we did for tables
    indexTransHandle := dbms_metadata.add_transform(indexOpenHandle, 'DDL');
    dbms_metadata.set_transform_param(indexTransHandle,
                      'SEGMENT_ATTRIBUTES', FALSE);
    dbms_metadata.set_transform_param(indexTransHandle,
                      'SQLTERMINATOR', TRUE);

-- Retrieve index DDLs as CLOBs and write them to the output file.
    LOOP
      indexDDL := dbms_metadata.fetch_clob(indexOpenHandle);
      EXIT WHEN indexDDL IS NULL;
      write_lob(indexDDL);
    END LOOP;

-- Free resources allocated for index stream.
    dbms_metadata.close(indexOpenHandle);

  END LOOP;

-- Free resources allocated for table stream and close output file.
  dbms_metadata.close(tableOpenHandle);
  utl_file.fclose(fileHandle);
  RETURN;

END;  -- of procedure get_payroll_tables

END dbms_metadata_example;
/
```

GET_XML および GET_DDL ファンクション

GET_XML および GET_DDL は、XML または DDL として指定されたオブジェクトのメタデータを戻します。

構文

```
DBMS_METADATA.GET_XML (
   object_type  IN VARCHAR2,
   name         IN VARCHAR2,
   schema       IN VARCHAR2 DEFAULT NULL,
   version      IN VARCHAR2 DEFAULT 'COMPATIBLE',
   model        IN VARCHAR2 DEFAULT 'ORACLE',
   transform    IN VARCHAR2 DEFAULT NULL)
RETURN CLOB;

DBMS_METADATA.GET_DDL (
   object_type  IN VARCHAR2,
   name         IN VARCHAR2,
   schema       IN VARCHAR2 DEFAULT NULL,
   version      IN VARCHAR2 DEFAULT 'COMPATIBLE',
   model        IN VARCHAR2 DEFAULT 'ORACLE',
   transform    IN VARCHAR2 DEFAULT 'DDL')
RETURN CLOB;
```

パラメータ

表 30-15 に、GET_xxx ファンクションのパラメータを示します。

表 30-15 GET_xxx パラメータ

パラメータ	説明
object_type	取り出すオブジェクト・タイプ。このパラメータは、OPEN object_type パラメータと同じ値を使用します。
name	オブジェクトの名前（大 / 小文字区別）。object_type が SYNONYM で、name が 30 文字を超える場合、name は LONGNAME フィルタとして処理されます。表 30-5 を参照してください。
schema	スキーマ名（大 / 小文字区別）。object_type がスキーマ・オブジェクトを参照する場合は、現行のスキーマがデフォルトです。そうでない場合は、NULL がデフォルトです。
version	抽出されるメタデータのバージョン。このパラメータは、OPEN バージョン・パラメータと同じ値を使用します。

表 30-15 GET_xxx パラメータ（続き）

パラメータ	説明
model	使用するオブジェクト・モデル。このパラメータは、OPEN モデル・パラメータと同じ値を使用します。
transform	出力時の変換名。このパラメータは、ADD_TRANSFORM 名前パラメータと同じ値を使用します。GET_XML の場合、DDL 以外の値を設定してください。

戻り値

XML または DDL として指定されたオブジェクトのメタデータ。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- OBJECT_NOT_FOUND。指定したオブジェクトがデータベースにありません。

使用上の注意

これらのファンクションは、単一のオブジェクトに対しメタデータを簡単に戻す方法を提供します。概念的に、各 GET_xxx コールは、1 つの OPEN、1 つまたは 2 つの SET_FILTER コール、1 つの ADD_TRANSFORM（オプション）、1 つの FETCH_xxx および 1 つの CLOSE で構成されます。object_type パラメータのセマンティクスは、OPEN と同じです。schema および name パラメータはフィルタ処理に使用されます。変換が指定された場合、スキーマレベルの変換フラグが継承されます。

このファンクションは、名前付きオブジェクトのフェッチにのみ使用できます。OBJECT_GRANT タイプまたは SYSTEM_GRANT タイプのオブジェクトはフェッチできません。このようなオブジェクトをフェッチするには、プログラム・インタフェースを使用してください。

例 1. SCOTT.EMP の XML 表示のフェッチ

完全に連続した出力を生成するには、問合せを実行する前に、次に示すように PAGESIZE に 0（ゼロ）を設定し、LONG に大きい数字を設定します。

```
set pagesize 0
set long 90000
SELECT DBMS_METADATA.GET_XML
  (
   'TABLE','EMP','SCOTT')
    FROM DUAL;
```

例 2. ネストした表およびオーバーフロー・セグメントをフィルタにかけ、現行のスキーマ内のすべての Complete 表に対する DDL のフェッチ

この例は、ネストした表およびオーバーフロー・セグメントをフィルタにかけ、現行のスキーマ内のすべての Complete 表に対する DDL をフェッチします。例では、SET_TRANSFORM_PARAM（ハンドル値 = DBMS_METADATA.SESSION_TRANSFORM。現行のセッションを示す）を使用して、記憶域句が SQL DDL に戻されないように指定しています。この後、セッション・レベルのパラメータをデフォルトにリセットします（完全に連続した出力を生成するには、問合せを実行する前に、次に示すように PAGESIZE に 0（ゼロ）を設定し、LONG に大きい数字を設定します）。

```
set pagesize 0
set long 90000
execute DBMS_METADATA.SET_TRANSFORM_PARAM(
  DBMS_METADATA.SESSION_TRANSFORM,'STORAGE',false);
SELECT DBMS_METADATA.GET_DDL('TABLE',u.table_name)
     FROM USER_ALL_TABLES u
     WHERE u.nested='NO'
     AND (u.iot_type is null or u.iot_type='IOT');
execute DBMS_METADATA.SET_TRANSFORM_PARAM(
   DBMS_METADATA.SESSION_TRANSFORM,'DEFAULT');
```

GET_DEPENDENT_XML および GET_DEPENDENT_DDL ファンクション

GET_DEPENDENT_XML および GET_DEPENDENT_DDL ファンクションは、1 つ以上の依存オブジェクトのメタデータを戻します。

構文

```
DBMS_METADATA.GET_DEPENDENT_XML (
   object_type           IN VARCHAR2,
   base_object_name      IN VARCHAR2,
   base_object_schema    IN VARCHAR2 DEFAULT NULL,
   version               IN VARCHAR2 DEFAULT 'COMPATIBLE',
   model                 IN VARCHAR2 DEFAULT 'ORACLE',
   transform             IN VARCHAR2 DEFAULT NULL,
   object_count          IN NUMBER   DEFAULT 10000)
RETURN CLOB;

DBMS_METADATA.GET_DEPENDENT_DDL (
   object_type           IN VARCHAR2,
   base_object_name      IN VARCHAR2,
   base_object_schema    IN VARCHAR2 DEFAULT NULL,
   version               IN VARCHAR2 DEFAULT 'COMPATIBLE',
   model                 IN VARCHAR2 DEFAULT 'ORACLE',
   transform             IN VARCHAR2 DEFAULT DDL,
   object_count          IN NUMBER   DEFAULT 10000)
RETURN CLOB;
```

パラメータ

表 30-16 に、GET_DEPENDENT_xxx ファンクションのパラメータを示します。

表 30-16 GET_DEPENDENT_xxx パラメータ

パラメータ	説明
object_type	取り出すオブジェクト・タイプ。このパラメータは、OPEN object_type パラメータと同じ値を使用します。表 30-2「Open() パラメータ」を参照してください。オブジェクト・タイプの属性は、ファンクションに適した属性であることが必要です。GET_DEPENDENT_xxx の場合は、依存オブジェクトです。
base_object_name	ベース・オブジェクト名。BASE_OBJECT_NAME フィルタで内部的に使用されます。
base_object_schema	ベース・オブジェクト・スキーマ。BASE_OBJECT_SCHEMA フィルタで内部的に使用されます。デフォルトはカレント・ユーザーです。
version	抽出されるメタデータのバージョン。このパラメータは、OPEN バージョン・パラメータと同じ値を使用します。
model	使用するオブジェクト・モデル。このパラメータは、OPEN モデル・パラメータと同じ値を使用します。
transform	出力時の変換名。このパラメータは、ADD_TRANSFORM 名前パラメータと同じ値を使用します。GET_DEPENDENT_XML の場合は、DDL 以外であることが必要です。
object_count	戻すオブジェクトの最大数。

戻り値

XML または DDL として指定されたオブジェクトのメタデータ。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- OBJECT_NOT_FOUND。指定したオブジェクトがデータベースにありません。

使用上の注意

GET_DEPENDENT_xxx ファンクションを使用すると、1 回のコールで依存オブジェクトのメタデータをフェッチできます。オブジェクト・タイプによっては、複数のファンクションを使用できます。たとえば GET_xxx を使用すると、索引をその名前でフェッチでき、GET_DEPENDENT_xxx を使用すると、同じ索引をその索引が定義されている表を指定してフェッチできます。

不特定多数の依存オブジェクトが GET_DEPENDENT_xxx の入力基準と一致する可能性があります。これらのオブジェクトをフェッチするときに、オブジェクト件数を指定できます。ただし、通常はデフォルト件数の 10000 が適切です。

DDL 変換が指定された場合は、セッションレベルの変換パラメータが継承されます。

これらのファンクションを SQL*Plus から起動する場合は、SET LONG および SET PAGESIZE コマンドを使用して、完全に連続した出力を生成する必要があります。

例 : SCOTT.EMP のすべてのオブジェクト権限付与に対する DDL のフェッチ

```
SQL> SET PAGESIZE 0
SQL> SET LONG 90000
SQL> SELECT DBMS_METADATA.GET_DEPENDENT_DDL('OBJECT_GRANT',
> 'EMP', 'SCOTT') FROM DUAL;
```

GET_GRANTED_XML および GET_GRANTED_DDL ファンクション

GET_GRANTED_XML および GET_GRANTED_DDL ファンクションは、1 つ以上の権限付与オブジェクトのメタデータを戻します。

構文

```
DBMS_METADATA.GET_GRANTED_XML (
   object_type          IN VARCHAR2,
   grantee              IN VARCHAR2 DEFAULT NULL,
   version              IN VARCHAR2 DEFAULT 'COMPATIBLE',
   model                IN VARCHAR2 DEFAULT 'ORACLE',
   transform            IN VARCHAR2 DEFAULT NULL,
   object_count         IN NUMBER   DEFAULT 10000)
RETURN CLOB;

DBMS_METADATA.GET_GRANTED_DDL (
   object_type          IN VARCHAR2,
   grantee              IN VARCHAR2 DEFAULT NULL,
   version              IN VARCHAR2 DEFAULT 'COMPATIBLE',
   model                IN VARCHAR2 DEFAULT 'ORACLE',
   transform            IN VARCHAR2 DEFAULT DDL,
   object_count         IN NUMBER   DEFAULT 10000)
RETURN CLOB;
```

パラメータ

表 30-17 に、GET_GRANTED_xxx ファンクションのパラメータを示します。

表 30-17 GET_GRANTED_xxx パラメータ

パラメータ	説明
object_type	取り出すオブジェクト・タイプ。このパラメータは、OPEN object_type パラメータと同じ値を使用します。表 30-2「Open() パラメータ」を参照してください。オブジェクト・タイプの属性は、ファンクションに適した属性であることが必要です。GET_GRANTED_xxx の場合は、権限付与オブジェクトです。
grantee	権限受領者。GRANTEE フィルタで内部的に使用されます。デフォルトはカレント・ユーザーです。
version	抽出されるメタデータのバージョン。このパラメータは、OPEN バージョン・パラメータと同じ値を使用します。
model	使用するオブジェクト・モデル。このパラメータは、OPEN モデル・パラメータと同じ値を使用します。
transform	出力時の変換名。このパラメータは、ADD_TRANSFORM 名前パラメータと同じ値を使用します。GET_GRANTED_XML の場合は、DDL 以外であることが必要です。
object_count	戻すオブジェクトの最大数。

戻り値

XML または DDL として指定されたオブジェクトのメタデータ。

例外

- INVALID_ARGVAL。入力パラメータに NULL または無効な値が指定されています。エラー・メッセージ・テキストに不適切なパラメータが表示されます。
- OBJECT_NOT_FOUND。指定したオブジェクトがデータベースにありません。

使用上の注意

GET_GRANTED_xxx ファンクションを使用すると、1 回のコールで権限付与オブジェクトのメタデータをフェッチできます。

不特定多数の権限付与オブジェクトが GET_GRANTED_xxx の入力基準と一致する可能性があります。これらのオブジェクトをフェッチするときに、オブジェクト件数を指定できます。ただし、通常はデフォルト件数の 10000 が適切です。

DDL 変換が指定された場合は、セッションレベルの変換パラメータが継承されます。

これらのファンクションを SQL*Plus から起動する場合は、SET LONG および SET PAGESIZE コマンドを使用して、完全に連続した出力を生成する必要があります。

例 : SCOTT に付与されたすべてのシステム権限付与に対する DDL のフェッチ

```
SQL> SET PAGESIZE 0
SQL> SET LONG 90000
SQL> SELECT DBMS_METADATA.GET_GRANTED_DDL('SYSTEM_GRANT','SCOTT')
> FROM DUAL;
```

31

DBMS_MGWADM

DBMS_MGWADM は、メッセージ・ゲートウェイの管理インタフェースを定義します。パッケージとオブジェクト・タイプの所有者は SYS です。

注意： catmgw.sql スクリプトを起動して、メッセージ・ゲートウェイのパッケージとタイプをデータベースにロードする必要があります。データベース・オブジェクトのロード方法の詳細は、『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。

関連項目： 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』に、DBMS_MGWADM の使用に関する情報が含まれています。

この章では、次の項目について説明します。

DBMS_MGWADM のオブジェクト・タイプとメソッドの要約

表 31-1 DBMS_MGWADM オブジェクト・タイプ

オブジェクト・タイプ	説明
「MGW_PROPERTY タイプ」 31-3 ページ	名前付きのプロパティを指定します。
「MGW_PROPERTY.CONSTRUCT メソッド」31-3 ページ	新規 MGW_PROPERTY インスタンスを構成します。
「MGW_PROPERTY.CONSTRUCT メソッド」31-3 ページ	パラメータを使用して初期化される新規 MGW_PROPERTY インスタンスを構成します。
「MGW_PROPERTIES タイプ」31-4 ページ	プロパティの配列を指定します。
「MGW_MQSERIES_PROPERTIES タイプ」 31-5 ページ	MQSeries メッセージ・システム・リンクの基本プロパティを指定します。
「MGW_MQSERIES_PROPERTIES.CONSTRUCT メソッド」31-7 ページ	新規 MGW_MQSERIES_PROPERTIES インスタンスを構成します。
「MGW_MQSERIES_PROPERTIES.ALTER_CONSTRUCT メソッド」 31-7 ページ	既存のメッセージ・リンクのプロパティを変更するための新規 MGW_MQSERIES_PROPERTIES インスタンスを構成します。

MGW_PROPERTY タイプ

このタイプは、名前付きのプロパティを指定します。MGW_PROPERTY は、メッセージ・リンクおよび外部キューのオプションのプロパティを指定するために使用されます。

構文

```
TYPE SYS.MGW_PROPERTY IS OBJECT(
   name VARCHAR2(100),
   value VARCHAR2(1000));
```

属性

表 31-2 MGW_PROPERTY の属性

属性	説明
name	プロパティ名
value	プロパティ値

MGW_PROPERTY.CONSTRUCT メソッド

このメソッドは、新規 MGW_PROPERTY インスタンスを構成します。すべての属性に NULL が割り当てられます。

構文

```
STATIC FUNCTION CONSTRUCT
RETURN SYS.MGW_PROPERTY;
```

MGW_PROPERTY.CONSTRUCT メソッド

このメソッドは、特定のパラメータを使用して初期化される新規 MGW_PROPERTY インスタンスを構成します。

構文

```
STATIC FUNCTION CONSTRUCT(
   p_name   IN VARCHAR2,
   p_value  IN VARCHAR2)
RETURN SYS.MGW_PROPERTY;
```

パラメータ

表 31-3 MGW_PROPERTY.CONSTRUCT のパラメータ

パラメータ	説明
p_name	プロパティ名
p_value	プロパティ値

MGW_PROPERTIES タイプ

このタイプは、プロパティの配列を指定します。

構文

```
TYPE SYS.MGW_PROPERTIES AS VARRAY (100) OF SYS.MGW_PROPERTY;
```

使用上の注意

特に注記がないかぎり、メッセージ・ゲートウェイは、名前付きのプロパティを次のように使用します。

- 'MGWPROP$_' 接頭辞が付いた名前は予約されています。これらは、特別な目的で使用されており、通常のプロパティ名として使用された場合は無効です。
- プロパティ名は、プロパティ・リストに 1 回のみ記述できます。つまり、リストでは、特定の名前に対して 1 つの値のみ指定できます。名前の大文字と小文字は区別されません。
- 通常、プロパティ・リストは順序に依存せず、プロパティ名は任意の順序で表示される場合があります。変更プロパティ・リストは例外です。
- 既存のプロパティ・リストを変更するために、新規プロパティ・リストを使用できます。この場合、各新規プロパティによって、新規プロパティの追加、プロパティの変更、プロパティの削除または全プロパティの削除のいずれかの方法で元のリストが変更されます。

変更リストは、最初の要素から最後の要素まで順に処理されます。したがって、変更リストの要素の順序は重要です（特に既存のリストからプロパティを削除するために使用する場合）。

その要素が元のリストにどのように作用するかを判別するために、プロパティ名と値が使用されます。次のルールが適用されます。

- プロパティの追加 / 変更

```
MGW_PROPERTY.NAME = <property name>
MGW_PROPERTY.VALUE = <property value>
```

指定した名前のプロパティがすでに存在している場合、現行の値は新しい値で置き換えられます。プロパティが存在していない場合は、新規プロパティがリストの最後に追加されます。

- プロパティの削除

```
MGW_PROPERTY.NAME = 'MGWPROP$_REMOVE'
MGW_PROPERTY.VALUE = <name of property to remove>
```

プロパティ名が元のリストに存在しない場合、処理は何も行われません。

- すべてのプロパティの削除

```
MGW_PROPERTY.NAME = 'MGWPROP$_REMOVE_ALL'
MGW_PROPERTY.VALUE = not used
```

DBMS_MGWADM パッケージは、予約済みのプロパティ名を表す定数を定義しています。MGWPROP_< > 定数を参照してください。

MGW_MQSERIES_PROPERTIES タイプ

このタイプは、MQSeries メッセージ・システム・リンクの基本プロパティを指定します。

構文

```
TYPE SYS.MGW_MQSERIES_PROPERTIES  IS OBJECT (
   queue_manager       VARCHAR2(64),
   hostname            VARCHAR2(64),
   port                INTEGER,
   channel             VARCHAR2(64),
   interface_type      INTEGER,
   max_connections     INTEGER,
   username            VARCHAR2(64),
   password            VARCHAR2(64),
   inbound_log_queue   VARCHAR2(64),
   outbound_log_queue  VARCHAR2(64));
```

属性

表 31-4 MGW_MQSERIES_PROPERTIES の属性

属性	説明
queue_manager	MQSeries キュー・マネージャの名前。
hostname	MQSeries メッセージ・システムが常駐しているホスト。ホスト名が NULL の場合は、MQSeries バインド接続が使用されます。NULL 以外の場合は、クライアント接続が使用され、ポートとチャネルを指定する必要があります。
port	ポート番号。クライアント接続の場合のみ、つまり hostname が NULL 以外の場合に使用されます。
channel	キュー・マネージャへの接続の確立時に使用されるチャネル。クライアント接続の場合のみ、つまり hostname が NULL 以外の場合に使用されます。
interface_type	使用するメッセージ・インタフェースのタイプ。値は、DBMS_MGWADM.MQSERIES_BASE_JAVA_INTERFACE（MQSeries Base Java インタフェースの場合）です。
max_connections	MQSeries メッセージ・システムへのメッセージ接続の最大数。
username	MQSeries メッセージ・システムに対する認証に使用されるユーザー名。
password	MQSeries メッセージ・システムに対する認証に使用されるパスワード。
inbound_log_queue	メッセージ・リンクがインバウンド伝播に使用される場合、つまり、このリンクと関連付けられたキューが伝播ソースとして機能する場合に、伝播リカバリの目的で使用される MQSeries キューのメッセージ・プロバイダ（ネイティブ）名。キューは、MQSeries 管理ツールを使用して作成される必要があります。
outbound_log_queue	メッセージ・リンクがアウトバウンド伝播に使用される場合、つまり、このリンクと関連付けられたキューが伝播宛先として機能する場合に、伝播リカバリの目的で使用される MQSeries キューのメッセージ・プロバイダ（ネイティブ）名。キューは、MQSeries 管理ツールを使用して作成される必要があります。

MGW_MQSERIES_PROPERTIES.CONSTRUCT メソッド

このメソッドは、新規 MGW_MQSERIES_PROPERTIES インスタンスを構成します。すべての属性に NULL が割り当てられます。

構文

```
STATIC FUNCTION CONSTRUCT
RETURN SYS.MGW_MQSERIES_PROPERTIES ;
```

MGW_MQSERIES_PROPERTIES.ALTER_CONSTRUCT メソッド

このメソッドは、既存のメッセージ・リンクのプロパティを変更するための新規 MGW_MQSERIES_PROPERTIES インスタンスを構成します。VARCHAR2 データ・タイプのすべての属性に、DBMS_MGWADM.NO_CHANGE が割り当てられます。他のデータ・タイプの属性には、NULL が割り当てられます。

構文

```
STATIC FUNCTION ALTER_CONSTRUCT
RETURN SYS.MGW_MQSERIES_PROPERTIES ;
```

DBMS_MGWADM 定数

表 31-5 DBMS_MGWADM 定数－伝播タイプ

名前	タイプ	説明
OUTBOUND_PROPAGATION	CONSTANT BINARY_INTEGER;	AQ から Oracle 以外への伝播用の伝播タイプを表します。伝播ソースはローカル AQ キューで、宛先は外部（Oracle 以外の）メッセージ・システムのキューです。
INBOUND_PROPAGATION	CONSTANT BINARY_INTEGER;	Oracle 以外から AQ への伝播用の伝播タイプを表します。伝播ソースは外部（Oracle 以外の）メッセージ・システムのキューで、宛先はローカル AQ キューです。

表 31-6 DBMS_MGWADM 定数－キュー・ドメイン・タイプ

名前	タイプ	説明
DOMAIN_QUEUE	CONSTANT BINARY_INTEGER;	キューの宛先を表します。JMS キュー（Point-to-Point モデル）はキューとして分類されます。
DOMAIN_TOPIC	CONSTANT BINARY_INTEGER;	トピックの宛先を表します。JMS トピック（パブリッシュ・サブスクライブ・モデル）はトピックとして分類されます。

表 31-7 DBMS_MGWADM 定数－強制値

名前	タイプ	説明
NO_FORCE	CONSTANT BINARY_INTEGER;	通常の非強制処理を表します。
FORCE	CONSTANT BINARY_INTEGER;	強制処理を表します。

表 31-8 DBMS_MGWADM 定数－停止モード

名前	タイプ	説明
SHUTDOWN_NORMAL	CONSTANT BINARY_INTEGER;	通常の停止モードを表します。
SHUTDOWN_IMMEDIATE	CONSTANT BINARY_INTEGER;	即時停止モードを表します。

表 31-9 DBMS_MGWADM 定数ークリーンアップ処理

名前	タイプ	説明
CLEAN_STARTUP_STATE	CONSTANT BINARY_INTEGER;	ゲートウェイ起動状態のリカバリ用のクリーンアップ処理を表します。

表 31-10 DBMS_MGWADM 定数ーロギング・レベル

名前	タイプ	説明
BASIC_LOGGING	CONSTANT BINARY_INTEGER;	ログ・ファイルに書き込まれるログ情報の詳細のレベルを表します。ロギング・レベルの範囲は、標準（最小限の）情報の BASIC_LOGGING から最大限の情報の TRACE_DEBUG_LOGGING までです。
TRACE_LITE_LOGGING	CONSTANT BINARY_INTEGER;	
TRACE_HIGH_LOGGING	CONSTANT BINARY_INTEGER;	
TRACE_DEBUG_LOGGING	CONSTANT BINARY_INTEGER;	

表 31-11 DBMS_MGWADM 定数ー MQSeries インタフェース・タイプ

名前	タイプ	説明
MQSERIES_BASE_JAVA_INTERFACE	CONSTANT BINARY_INTEGER;	MQSeries メッセージ・システム用の Base Java インタフェースを表します。
MQSERIES_JMS_INTERFACE	CONSTANT BINARY_INTEGER;	MQSeries メッセージ・システム用の JMS インタフェースを表します。

表 31-12 DBMS_MGWADM 定数ー名前付きのプロパティの定数

名前	タイプ	説明
MGWPROP_PREFIX	CONSTANT VARCHAR2;	予約済みのプロパティ名接頭辞の定数（MGWPROP$_）。
MGWPROP_REMOVE	CONSTANT VARCHAR2;	既存のプロパティを削除するために使用される予約済みのプロパティ名の定数（MGWPROP$_REMOVE）。
MGWPROP_REMOVE_ALL	CONSTANT VARCHAR2;	すべてのプロパティを削除するために使用される予約済みのプロパティ名の定数（MGWPROP$_REMOVE_ALL）。

表 31-13 DBMS_MGWADM 定数－その他の定数

名前	タイプ	説明
NO_CHANGE	CONSTANT VARCHAR2;	既存の値を保持する（変更しない）ことを示します。これは、1 つ以上のパラメータを変更し、その他のパラメータは変更しない場合に、特定の API に使用されます。

MQSeries システム・プロパティ

次の項では、リンクとキューに関連する MQSeries のプロパティについて説明します。詳細は、IBM MQSeries のドキュメントを参照してください。

基本リンク・プロパティ（MGW_MQSERIES_PROPERTIES）

表 31-14 は、MQSeries メッセージ・リンクの基本構成プロパティの一覧です（カッコ内の数字の説明は、31-11 ページの「表 31-14 の注」を参照してください）。表には、プロパティがオプション（NULL が設定可能）かどうか、変更可能かどうか、および変更可能な場合は動的に変更可能かどうかが示されています。

表 31-14 MQSeries リンク・プロパティ

属性	NULL 値設定可能？	値変更可能？	動的変更可能？
queue_manager	いいえ	いいえ	--
hostname	はい（1）	いいえ	--
port	はい（1）	いいえ	--
channel	はい（1）	いいえ	--
interface_type	はい（2）	いいえ	--
max_connections	はい（3）	はい	はい
username	はい	はい	はい
password	はい	はい	はい
inbound_log_queue	はい（4）	はい（4）	はい
outbound_log_queue	はい（5）	はい（5）	はい

表 31-14 の注

1. hostname が NULL の場合、port と channel には NULL を設定する必要があります。hostname が NULL 以外の場合、port と channel には NULL 以外の値を設定する必要があります。hostname が NULL の場合は MQSeries バインド接続が使用され、NULL 以外の場合はクライアント接続が使用されます。
2. NULL の場合は、デフォルト値 DBMS_MGWADM.MQSERIES_BASE_JAVA_INTERFACE が使用されます。
3. NULL の場合は、デフォルト値 1 が使用されます。
4. リンクがインバウンド伝播に使用されない場合、インバウンド・ログ・キューは NULL にできます。ログ・キューは、そのリンクを参照するインバウンド伝播サブスクライバがない場合にのみ変更できます。
5. リンクがアウトバウンド伝播に使用されない場合、アウトバウンド・ログ・キューは NULL にできます。ログ・キューは、そのリンクを参照するアウトバウンド伝播サブスクライバがない場合にのみ変更できます。

オプションのリンク・プロパティ

この項では、MQSeries メッセージ・リンクに対してサポートされるオプションの構成プロパティについて説明します。これらのプロパティは、DBMS_MGWADM.CREATE_MSGSYSTEM_LINK および DBMS_MGWADM.ALTER_MSGSYSTEM_LINK の options パラメータを使用して指定されます。

MQ_ccsid

このプロパティは、使用するキャラクタ・セット識別子を指定します。説明的な文字列ではなく、キャラクタ・セットの整数値（例 : 819）を指定する必要があります。設定されない場合は、MQSeries のデフォルト・キャラクタ・セット 819 が使用されます。

デフォルト : 819

変更可能 : はい

動的変更可能 : いいえ

MQ_ReceiveExit

このプロパティは、MQReceiveExit インタフェースを実装するクラスの完全修飾 Java クラス名を指定します。設定されない場合、デフォルトは使用されません。このクラスは、メッセージ・ゲートウェイ・エージェントの CLASSPATH 内にあることが必要です。

デフォルト : なし

変更可能 : はい

動的変更可能 : いいえ

MQ_SendExit

このプロパティは、MQSendExit インタフェースを実装するクラスの完全修飾 Java クラス名を指定します。設定されない場合、デフォルトは使用されません。このクラスは、メッセージ・ゲートウェイ・エージェントの CLASSPATH 内にあることが必要です。

デフォルト : なし

変更可能 : はい

動的変更可能 : いいえ

MQ_SecurityExit

このプロパティは、MQSecurityExit インタフェースを実装するクラスの完全修飾 Java クラス名を指定します。設定されない場合、デフォルトは使用されません。このクラスは、メッセージ・ゲートウェイ・エージェントの CLASSPATH 内にあることが必要です。

デフォルト : なし

変更可能 : はい

動的変更可能 : いいえ

オプションのキュー・プロパティ

この項では、MQSeries メッセージ・リンクの登録済みキューに対してサポートされるオプションの構成プロパティについて説明します。これらのプロパティは、DBMS_MGWADM.REGISTER_FOREIGN_QUEUE の options パラメータを使用して指定されます。

MQ_openOptions

このプロパティは、MQSeries Base Java の MQQueueManager.accessQueue メソッドの openOptions 引数に使用される値を指定します。必須ではありませんが、値を 1 つ指定すると、メッセージ・ゲートウェイ・エージェントは、エンキュー（put）操作のために、MQOO_OUTPUT を指定値に追加します。デキュー（get）操作の場合は、MQOO_INPUT_SHARED が追加されます。

デフォルト : エンキュー /put 操作の場合は MQOO_OUTPUT、デキュー /get 操作の場合は MQOO_INPUT_SHARED

変更可能 : いいえ

動的変更可能 : いいえ

DBMS_MGWADM サブプログラムの要約

表 31-15 DBMS_MGWADM のサブプログラム

サブプログラム	説明
「ALTER_AGENT プロシージャ」 31-15 ページ	メッセージ・ゲートウェイ・エージェントのパラメータを変更します。
「DB_CONNECT_INFO プロシージャ」 31-16 ページ	Oracle データベースへの接続のためにメッセージ・ゲートウェイ・エージェントが使用する接続情報を構成します。
「STARTUP プロシージャ」 31-17 ページ	メッセージ・ゲートウェイ・エージェントを起動します。
「SHUTDOWN プロシージャ」 31-18 ページ	メッセージ・ゲートウェイ・エージェントを停止します。
「CLEANUP_GATEWAY プロシージャ」 31-18 ページ	メッセージ・ゲートウェイをクリーンアップします。
「SET_LOG_LEVEL プロシージャ」 31-20 ページ	メッセージ・ゲートウェイ・エージェントのロギング・レベルを動的に変更します。

表 31-15 DBMS_MGWADM のサブプログラム（続き）

サブプログラム	説明
「CREATE_MSGSYSTEM_LINK プロシージャ」31-20 ページ	MQSeries メッセージ・システムへのメッセージ・システム・リンクを作成します。
「ALTER_MSGSYSTEM_LINK プロシージャ」31-21 ページ	MQSeries メッセージ・システム・リンクのプロパティを変更します。
「REMOVE_MSGSYSTEM_LINK プロシージャ」31-23 ページ	Oracle 以外のメッセージ・システムに対するメッセージ・システム・リンクを削除します。
「REGISTER_FOREIGN_QUEUE プロシージャ」31-23 ページ	Oracle 以外のキュー・エンティティをメッセージ・ゲートウェイに登録します。
「UNREGISTER_FOREIGN_QUEUE プロシージャ」31-25 ページ	メッセージ・ゲートウェイ内の Oracle 以外のキュー・エンティティを削除します。
「ADD_SUBSCRIBER プロシージャ」31-25 ページ	伝播用のソース・キューから宛先にメッセージをコンシュームするために使用されるサブスクライバを追加します。
「ALTER_SUBSCRIBER プロシージャ」31-28 ページ	伝播用のソース・キューから宛先にメッセージをコンシュームするために使用されるサブスクライバのパラメータを変更します。
「REMOVE_SUBSCRIBER プロシージャ」31-31 ページ	伝播用のソース・キューから宛先にメッセージをコンシュームするために使用されるサブスクライバを削除します。
「RESET_SUBSCRIBER プロシージャ」31-32 ページ	サブスクライバの伝播エラー状態をリセットします。
「SCHEDULE_PROPAGATION プロシージャ」31-32 ページ	ソースから宛先へのメッセージ伝播をスケジュールします。
「UNSCHEDULE_PROPAGATION プロシージャ」31-34 ページ	伝播スケジュールを削除します。
「ALTER_PROPAGATION_SCHEDULE プロシージャ」31-35 ページ	伝播スケジュールを変更します。
「ENABLE_PROPAGATION_SCHEDULE プロシージャ」31-36 ページ	伝播スケジュールを使用可能にします。
「DISABLE_PROPAGATION_SCHEDULE プロシージャ」31-36 ページ	伝播スケジュールを使用禁止にします。

ALTER_AGENT プロシージャ

このプロシージャは、メッセージ・ゲートウェイ・エージェントのパラメータを変更します。

構文

```
DBMS_MGWADM.ALTER_AGENT (
   max_connections  IN BINARY_INTEGER  DEFAULT NULL,
   max_memory       IN BINARY_INTEGER  DEFAULT NULL);
```

パラメータ

表 31-16 ALTER_AGENT プロシージャのパラメータ

パラメータ	説明
max_connections	ゲートウェイ・エージェントが使用する Oracle データベースへのメッセージ接続の最大数。NULL の場合、現行の値は変更されません。NULL 以外の場合は、1 以上の値を設定する必要があります。
max_memory	ゲートウェイ・エージェントが使用する最大ヒープ・サイズ（MB）。NULL の場合、現行の値は変更されません。NULL 以外の場合は、64 以上の値を設定する必要があります。

使用上の注意

構成パラメータのデフォルト値は、メッセージ・ゲートウェイのインストール時に設定されます。

max_memory パラメータの変更は、ゲートウェイ・エージェントが次回アクティブになったときに有効になります。エージェントが現在アクティブの場合、変更を有効にするには、ゲートウェイを停止し、再起動する必要があります。

max_connections パラメータは、AQ ドライバによって作成および使用される JDBC メッセージ接続の最大数を指定します。このパラメータは、大きい値に変更した場合のみ動的に変更されます。リリース 2（9.2）では、小さい値への変更を有効にするには、ゲートウェイ・エージェントを停止し、再起動する必要があります。

DB_CONNECT_INFO プロシージャ

このプロシージャは、Oracle データベースへの接続のためにメッセージ・ゲートウェイ・エージェントが使用する接続情報を構成します。

構文

```
DBMS_MGWADM.DB_CONNECT_INFO (
   username      IN VARCHAR2,
   password      IN VARCHAR2,
   database      IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 31-17 DB_CONNECT_INFO プロシージャのパラメータ

パラメータ	説明
username	Oracle データベースへの接続に使用されるユーザー名。NULL は設定できません。
password	Oracle データベースへの接続に使用されるパスワード。NULL は設定できません。
database	ゲートウェイ・エージェントが使用するデータベース接続文字列。NULL は、ローカル接続を使用することを示します。

使用上の注意

ゲートウェイ・エージェントは、この API によって構成されるユーザーで Oracle データベースに接続します。Oracle の管理者は、ユーザーを作成し、そのユーザーにロール MGW_AGENT_ROLE を付与した後、このプロシージャをコールしてメッセージ・ゲートウェイを構成する必要があります。MGW_AGENT_ROLE は、データベースに格納されているゲートウェイ構成情報へのアクセス、Oracle キューとのメッセージのデキューとエンキュー、および特定の AQ 管理タスクの実行に必要となる特別な権限をこのユーザーに付与するために使用されます。

STARTUP プロシージャ

このプロシージャは、メッセージ・ゲートウェイ・エージェントを起動します。伝播アクティビティを行うには、このプロシージャをコールする必要があります。

構文

```
DBMS_MGWADM.STARTUP(
   instance  IN  BINARY_INTEGER DEFAULT 0,
   force     IN  BINARY_INTEGER DEFAULT dbms_mgwadm.NO_FORCE);
```

パラメータ

表 31-18 STARTUP プロシージャのパラメータ

パラメータ	説明
instance	メッセージ・ゲートウェイ・エージェントの起動に使用されるジョブ・キューのジョブを実行できるインスタンスを指定します。0（ゼロ）の場合は、どのインスタンスでもジョブを実行できます。
force	dbms_mgwadm.FORCE の場合は、ジョブ・インスタンスとして正の整数がすべて受け入れられます。dbms_mgwadm.NO_FORCE（デフォルト）の場合は、指定したインスタンスで実行する必要があり、そうでない場合は、例外が発生します。

使用上の注意

メッセージ・ゲートウェイ・エージェントは、DB_CONNECT_INFO を使用してエージェント・ユーザーが構成されるまで起動できません。

このプロシージャは、ジョブ・キューのジョブを発行し、実行時にメッセージ・ゲートウェイ・エージェントを起動します。instance および force パラメータは、ジョブ・キュー親和性に使用されます。この親和性を使用して、発行されたジョブを実行できるのは、特定のインスタンスか、または任意のインスタンスかを示します。

SHUTDOWN プロシージャ

このプロシージャは、メッセージ・ゲートウェイ・エージェントを停止します。ゲートウェイが起動されるまで、伝播アクティビティは行われません。

構文

```
DBMS_MGWADM.SHUTDOWN (
   sdmode  IN BINARY_INTEGER DEFAULT DBMS_MGWADM.SHUTDOWN_NORMAL);
```

パラメータ

表 31-19 SHUTDOWN プロシージャのパラメータ

パラメータ	説明
sdmode	停止モード。値は次のとおりです。 ■ SHUTDOWN_NORMAL: 通常の停止。ゲートウェイ・エージェントは、現在進行中の伝播作業を完了しようとします。 ■ SHUTDOWN_IMMEDIATE: 即時停止。ゲートウェイは、現在進行中のすべての伝播作業を終了し、即時に停止します。

使用上の注意

リリース 2（9.2）では、sdmode パラメータは無視され、停止モードはすべて同様に動作します。

CLEANUP_GATEWAY プロシージャ

このプロシージャは、メッセージ・ゲートウェイをクリーンアップします。ゲートウェイがなんらかの異常がある状態または予期しない状態のままのときに必要なクリーンアップまたはリカバリ処理を実行します。MGW_GATEWAY ビューには、クリーンアップ処理に関係するゲートウェイ・ステータスおよび構成情報がリストされます。

構文

```
DBMS_MGWADM.CLEANUP_GATEWAY(
   action  IN BINARY_INTEGER);
```

パラメータ

表 31-20 CLEANUP_GATWAY プロシージャのパラメータ

パラメータ	説明
action	実行するクリーンアップ処理。値は次のとおりです。 CLEAN_STARTUP_STATE: ゲートウェイ起動状態のリカバリの場合。

使用上の注意

CLEAN_STARTUP_STATE 処理には、リカバリ・タスクが含まれます。このタスクは、ゲートウェイ・エージェントがクラッシュしたか、またはゲートウェイ・エージェントを起動できないような他のなんらかの異常なイベントが発生したときに、ゲートウェイを既知の状態に設定します。この処理は、ゲートウェイ・エージェントが起動されたが、クラッシュしたと思われるか、長時間応答がない場合にのみ実行してください。

この処理が必要と想定される状態または現象

- MGW_GATEWAY ビューで、AGENT_STATUS の値が NOT_STARTED または START_SCHEDULED 以外であり、AGENT_PING の値が長時間 UNREACHABLE である場合。

クリーンアップ・タスクでは次の処理が行われます。

- 外部ゲートウェイ・エージェント・プロセスの起動に使用されるキューイングされたジョブを削除します。
- 特定の構成情報を既知の状態に設定します。たとえば、エージェント・ステータスを NOT_STARTED に設定します。

次の点を考慮してください。

- エージェント・ステータスが NOT_STARTED または START_SCHEDULED の場合は失敗します。
- エージェント・ステータスが STARTING の場合を除いて、このプロシージャのコール前に停止が試みられない場合は失敗します。
- ゲートウェイ・エージェントと通信（ping）しようとします。成功した場合は、エージェントはアクティブで、このプロシージャは失敗したと考えられます。数回試行してもエージェントが応答しない場合は、クリーンアップ・タスクが実行されます。
- このプロシージャは、ゲートウェイ・エージェントが ping に応答しない場合は、数秒（場合によっては 1 分程度まで）かかります。これは、この特定のクリーンアップ処理が適切で必要な状態での予想される動作です。

SET_LOG_LEVEL プロシージャ

このプロシージャは、メッセージ・ゲートウェイ・エージェントのロギング・レベルを動的に変更します。メッセージ・ゲートウェイ・エージェントが実行されている必要があります。

構文

```
DBMS_MGWADM.SET_LOG_LEVEL (
   log_level  IN BINARY_INTEGER);
```

パラメータ

表 31-21 SET_LOG_LEVEL プロシージャのパラメータ

パラメータ	説明
log_level	メッセージ・ゲートウェイ・エージェントが情報をログに記録するレベル。DBMS_MGWADM.<>_LOGGING 定数を参照してください。BASIC_LOGGING の場合は最小限の情報が生成され、TRACE_DEBUG_LOGGING の場合は最大限の情報が生成されます。

CREATE_MSGSYSTEM_LINK プロシージャ

このプロシージャは、MQSeries メッセージ・システムへのメッセージ・システム・リンクを作成します。

構文

```
DBMS_MGWADM.CREATE_MSGSYSTEM_LINK(
   linkname    IN VARCHAR2,
   properties  IN sys.mgw_mqseries_properties,
   options     IN sys.mgw_properties DEFAULT NULL,
   comment     IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 31-22 CREATE_MSGSYSTEM_LINK プロシージャのパラメータ

パラメータ	説明
linkname	メッセージ・システム・リンクを識別するためのユーザー定義名。
properties	MQSeries メッセージ・システム・リンクの基本プロパティ。
options	オプションのリンク・プロパティ。オプションのプロパティがない場合は NULL を設定します。これらは、メッセージ・システムでサポートされている、あまり使用されない構成プロパティです。
comment	ユーザーが指定した説明。必要ない場合は NULL を設定します。

使用上の注意

メッセージ・リンク・プロパティの詳細は、31-10 ページの「基本リンク・プロパティ(MGW_MQSERIES_PROPERTIES)」を参照してください。

ALTER_MSGSYSTEM_LINK プロシージャ

このプロシージャは、MQSeries メッセージ・システム・リンクのプロパティを変更します。

構文

```
DBMS_MGWADM.ALTER_MSGSYSTEM_LINK (
   linkname   IN  VARCHAR2,
   properties IN  SYS.MGW_MQSERIES_PROPERTIES,
   options    IN  SYS.MGW_PROPERTIES DEFAULT NULL,
   comment    IN  VARCHAR2 DEFAULT DBMS_MGWADM.NO_CHANGE);
```

パラメータ

表 31-23 ALTER_MSGSYSTEM_LINK プロシージャのパラメータ

パラメータ	説明
linkname	メッセージ・システム・リンク名。
properties	MQSeries メッセージ・システム・リンクの基本プロパティ。NULL の場合は、どのリンク・プロパティも変更されません。
options	オプションのリンク・プロパティ。どのオプションも変更しない場合は NULL を設定します。NULL 以外の場合、このリストで指定されたプロパティは、現行のオプション・プロパティと組み合され、新しいリンク・オプションのセットを構成します。
comment	オプションの説明。必要ない場合は NULL を設定します。DBMS_MGWADM.NO_CHANGE が指定された場合、現行の値は変更されません。

使用上の注意

リリース 2（9.2）では、MGW_MQSERIES_PROPERTIES.MAX_CONNECTIONS パラメータは、そのメッセージ・リンク用に作成および使用されるメッセージ接続の最大数を指定します。このパラメータは、大きい値に変更された場合のみ動的に変更されます。小さい値への変更を有効にするには、ゲートウェイ・エージェントを停止し、再起動する必要があります。

VARCHAR2 データ・タイプのメッセージ・リンク・プロパティの既存値を保持するには、その特定のプロパティに DBMS_MGWADM.NO_CHANGE を指定します。別のデータ・タイプのプロパティの既存値を保持するには、そのプロパティに NULL を指定します。

options パラメータは、現行のオプション・プロパティを変更するために使用されるプロパティのセットを指定します。各プロパティによって、新規プロパティの追加、既存プロパティの置換、既存プロパティの削除または全プロパティの削除のいずれかの方法で現行のプロパティ・リストが変更されます。

一部のプロパティは変更できず、それらを変更しようとするとこのプロシージャは失敗します。その他のプロパティは、現行の構成によって特定の条件下（たとえば、リンクに関連付けられたソースまたは宛先を持つ伝播サブスクライバまたはスケジュールがない場合）でのみ変更可能です。

変更可能なプロパティの中で動的に変更できるものは少数で、その他のプロパティは、変更を有効にするために、メッセージ・ゲートウェイを停止し、再起動する必要があります。

メッセージ・リンク・プロパティの詳細は、31-10 ページの「基本リンク・プロパティ（MGW_MQSERIES_PROPERTIES）」を参照してください。

REMOVE_MSGSYSTEM_LINK プロシージャ

このプロシージャは、Oracle 以外のメッセージ・システムに対するメッセージ・システム・リンクを削除します。

構文

```
DBMS_MGWADM.REMOVE_MSGSYSTEM_LINK(
   linkname  IN VARCHAR2);
```

パラメータ

表 31-24 REMOVE_MSGSYSTEM_LINK プロシージャのパラメータ

パラメータ	説明
linkname	メッセージ・システム・リンク名。

使用上の注意

メッセージ・システム・リンクを削除するには、このリンクと関連付けられた登録済みのすべてのキューを削除する必要があります。このリンクを参照する登録済みの外部（Oracle 以外の）キューがある場合は失敗します。

REGISTER_FOREIGN_QUEUE プロシージャ

このプロシージャは、メッセージ・ゲートウェイに Oracle 以外のキュー・エンティティを登録します。

構文

```
DBMS_MGWADM.REGISTER_FOREIGN_QUEUE(
   name            IN VARCHAR2,
   linkname        IN VARCHAR2,
   provider_queue  IN VARCHAR2 DEFAULT NULL,
   domain          IN INTEGER DEFAULT NULL,
   options         IN sys.mgw_properties DEFAULT NULL,
   comment         IN VARCHAR2  DEFAULT NULL);
```

パラメータ

表 31-25 REGISTER_FOREIGN_QUEUE プロシージャのパラメータ

パラメータ	説明
name	登録されるキュー名。この名前は、メッセージ・ゲートウェイ内で外部キューを識別します。外部メッセージ・システムのキュー名と一致する必要はありません。
linkname	このキューが存在するメッセージ・システムに対するリンク名。
provider_queue	メッセージ・プロバイダ（ネイティブ）のキュー名。NULL の場合は、name パラメータに提供された値がプロバイダ・キュー名として使用されます。
domain	キューのドメイン・タイプ。値は次のとおりです。 ■ NULL: ドメイン・タイプがキューのメッセージ・システムに基づいて自動的に判断される場合 ■ DOMAIN_QUEUE: キュー（Point-to-Point モデル）の場合 ■ DOMAIN_TOPIC: トピック（パブリッシュ・サブスクライブ・モデル）の場合
options	オプションのキュー・プロパティ。
comment	ユーザーが指定した説明。NULL を設定することもできます。

使用上の注意

このプロシージャは、Oracle 以外のメッセージ・システムに物理的なキューを作成しません。Oracle 以外のキューは、そのメッセージ・システムの管理ツールを使用して作成する必要があります。

リリース 2（9.2）では、domain は使用されないため、NULL を設定する必要があります。これは、現在サポートされているメッセージ・システムに対してはドメイン・タイプを自動的に判断できるためです。

メッセージ・リンク・プロパティの詳細は、31-10 ページの「基本リンク・プロパティ（MGW_MQSERIES_PROPERTIES）」を参照してください。

UNREGISTER_FOREIGN_QUEUE プロシージャ

このプロシージャは、メッセージ・ゲートウェイ内の Oracle 以外のキュー・エンティティを削除します。

構文

```
DBMS_MGWADM.UNREGISTER_FOREIGN_QUEUE(
   name         IN VARCHAR2,
   linkname     IN VARCHAR2);
```

パラメータ

表 31-26 UNREGISTER_FOREIGN_QUEUE プロシージャのパラメータ

パラメータ	説明
name	キュー名。
linkname	キューが存在するメッセージ・システムに対するリンク名。

使用上の注意

このプロシージャは、Oracle 以外のメッセージ・システム内の物理的なキューは削除しません。

キューの登録を解除するには、このキューを参照しているすべてのサブスクライバおよびスケジュールの登録を解除する必要があります。サブスクライバまたは伝播スケジュールが Oracle 以外のキューを参照している場合は失敗します。

ADD_SUBSCRIBER プロシージャ

このプロシージャは、伝播用のソース・キューから宛先にメッセージをコンシュームするために使用されるサブスクライバを追加します。

構文

```
DBMS_MGWADM.ADD_SUBSCRIBER(
   subscriber_id     IN VARCHAR2,
   propagation_type  IN BINARY_INTEGER,
   queue_name        IN VARCHAR2,
   destination       IN VARCHAR2,
   rule              IN VARCHAR2 DEFAULT NULL,
   transformation    IN VARCHAR2 DEFAULT NULL,
   exception_queue   IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 31-27 ADD_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
subscriber_id	このサブスクライバを識別するユーザー定義名を指定します。
propagation_type	メッセージ伝播のタイプを指定します。値は次のとおりです。 ■ DBMS_MGWADM.OUTBOUND_PROPAGATION: AQ から Oracle 以外への伝播の場合 ■ DBMS_MGWADM.INBOUND_PROPAGATION: Oracle 以外から AQ への伝播の場合
queue_name	このサブスクライバが追加されるソース・キューを指定します。このパラメータの構文および解釈は、propagation_type に指定された値によって決まります。
destination	このサブスクライバによってコンシュームされるメッセージの伝播先の宛先キューを指定します。このパラメータの構文および解釈は、propagation_type に指定された値によって決まります。
rule	ソース・キューからメッセージをデキューするために、サブスクライバが使用するオプションのサブスクリプション・ルールを指定します。ルールが必要ない場合は NULL を設定します。このパラメータの構文および解釈は、propagation_type に指定された値によって決まります。
transformation	AQ ペイロードとゲートウェイ定義のユーザー定義型間の変換に必要な変換機能を指定します。必要な変換機能のタイプは、propagation_type に指定された値によって決まります。 変換機能が提供されない（NULL 値が指定された）場合、ゲートウェイは、AQ ペイロード・タイプと Oracle 以外のメッセージ・システムの機能に基づいて、最大限メッセージを伝播しようとします。たとえば、ゲートウェイは、RAW ペイロードを持つ AQ キュー、およびバイトのメッセージ本体をサポートする Oracle 以外のメッセージ・システムに対するメッセージを自動的に伝播します。
exception_queue	例外メッセージ・ロギングの目的で使用されるキューを指定します。このキューは、伝播ソースと同じメッセージ・システム上にあることが必要です。NULL の場合、問題発生時に例外キューは使用されず、伝播は停止します。このパラメータの構文および解釈は、propagation_type に指定された値によって決まります。 ソース・キューと例外キューを同じキューにはできません。

使用上の注意

OUTBOUND_PROPAGATION の場合、パラメータは次のように解釈されます。

- queue_name ― 伝播ソースであるローカル AQ キューを指定します。これは、schema.queue の構文で指定する必要があります。
- destination ― メッセージの伝播先の外部キューを指定します。これは、registered_queue@message_link の構文で指定する必要があります。
- rule ― オプションの AQ サブスクライバ・ルールを指定します。ルールが必要ない場合は NULL を設定します。
- transformation ― AQ ペイロードをゲートウェイ定義のユーザー定義型に変換するために使用される変換機能を指定します。

  ゲートウェイ伝播は、AQ ペイロードを既知のゲートウェイ定義のユーザー定義型に変換する変換機能を使用して、AQ キューからメッセージをデキューします。その後、メッセージは、ゲートウェイのユーザー定義型に基づいて外部メッセージ・システムにエンキューされます。
- exception_queue ― 例外が発生した場合にメッセージを移動するローカル AQ キューの名前を指定します。これは、schema.queue の構文で指定する必要があります。

INBOUND_PROPAGATION の場合、パラメータは次のように解釈されます。

- queue_name ― 伝播ソースである外部キューを指定します。これは、registered_queue@message_link の構文で指定する必要があります。
- destination ― メッセージの伝播先のローカル AQ キューを指定します。これは、schema.queue の構文で指定する必要があります。
- rule ― 外部メッセージ・システムに対して有効であるオプションのサブスクライバ・ルールを指定します。ルールが必要ない場合は NULL を設定します。
- transformation ― ゲートウェイ定義のユーザー定義型を AQ ペイロード・タイプに変換するために使用される変換機能を指定します。

  ゲートウェイ伝播は、外部メッセージ・システムからメッセージをデキューし、メッセージ本体を既知のゲートウェイ定義のユーザー定義型に変換します。変換機能は、メッセージが AQ キューにエンキューされるときに、ゲートウェイ・ユーザー定義型を AQ ペイロード・タイプに変換するために使用されます。
- exception_queue ― 例外が発生した場合にメッセージを移動する外部キューの名前を指定します。これは、registered_queue@message_link の構文で指定する必要があります。

OUTBOUND_PROPAGATION の場合は、ローカル・サブスクライバが AQ キューに追加されます。サブスクライバの構成は、aq$_agent('MGW_<subscriber_id>',NULL,NULL) です。

INBOUND_PROPAGATION の場合、サブスクライバが必要かどうかは、Oracle 以外のメッセージ・システムの要件によって決まります。

OUTBOUND_PROPAGATION の場合は、例外キューに関して次の点を考慮してください。

- 例外キューとして使用される AQ キューは、ユーザーが作成する必要があります。
- ソース・キューと例外キューのペイロード・タイプは一致している必要があります。
- 例外キューは、キュー・タイプ EXCEPTION_QUEUE ではなく、NORMAL_QUEUE で作成する必要があります。エンキューの制限事項によって、ゲートウェイ伝播は、ゲートウェイの例外キューとしてタイプ EXCEPTION_QUEUE の AQ キューを使用できません。

INBOUND_PROPAGATION の場合は、例外キューに関して次の点を考慮してください。

- 例外キューは、登録済みの Oracle 以外のキューであることが必要です。
- ソース・キューと例外キューは、同じメッセージ・システム・リンクを使用する必要があります。

ALTER_SUBSCRIBER プロシージャ

このプロシージャは、伝播用のソース・キューから宛先にメッセージをコンシュームするために使用されるサブスクライバのパラメータを変更します。

構文

```
DBMS_MGWADM.ALTER_SUBSCRIBER (
   subscriber_id    IN VARCHAR2,
   rule             IN VARCHAR2 DEFAULT DBMS_MGWADM.NO_CHANGE,
   transformation   IN VARCHAR2 DEFAULT DBMS_MGWADM.NO_CHANGE,
   exception_queue  IN VARCHAR2 DEFAULT DBMS_MGWADM.NO_CHANGE );
```

パラメータ

表 31-28 ALTER_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
subscriber_id	変更するサブスクライバを識別します。
rule	ソース・キューからメッセージをデキューするために、サブスクライバが使用するオプションのサブスクリプション・ルールを指定します。このパラメータの構文および解釈は、サブスクライバの伝播タイプによって決まります。 NULL 値は、サブスクリプション・ルールが必要ないことを示します。DBMS_MGWADM.NO_CHANGE の場合、現行の値は変更されません。
transformation	AQ ペイロードとゲートウェイ定義のユーザー定義型間の変換に必要な変換機能を指定します。必要な変換機能のタイプは、サブスクライバの伝播タイプによって決まります。 NULL 値は、変換機能が必要ないことを示します。DBMS_MGWADM.NO_CHANGE の場合、現行の値は変更されません。
exception_queue	例外メッセージ・ロギングの目的で使用されるキューを指定します。このキューは、伝播ソースと同じメッセージ・システム上にあることが必要です。サブスクライバと関連付けられた例外キューがない場合、問題発生時に伝播は停止します。このパラメータの構文および解釈は、サブスクライバの伝播タイプによって決まります。 NULL 値は、例外キューが使用されないことを示します。DBMS_MGWADM.NO_CHANGE の場合、現行の値は変更されません。 ソース・キューと例外キューを同じキューにはできません。

使用上の注意

サブスクライバの伝播タイプが OUTBOUND_PROPAGATION の場合、パラメータは次のように解釈されます。

- rule ― オプションの AQ サブスクライバ・ルールを指定します。
- transformation ― AQ ペイロードをゲートウェイ定義のユーザー定義型に変換するために使用される変換機能を指定します。

  ゲートウェイ伝播は、AQ ペイロードを既知のゲートウェイ定義のユーザー定義型に変換する変換機能を使用して、AQ キューからメッセージをデキューします。その後、メッセージは、ゲートウェイのユーザー定義型に基づいて外部メッセージ・システムにエンキューされます。
- exception_queue ― 例外が発生した場合にメッセージを移動するローカル AQ キューの名前を指定します。これは、schema.queue の構文で指定する必要があります。

サブスクライバの伝播タイプが INBOUND_PROPAGATION の場合、パラメータは次のように解釈されます。

- rule ― 外部メッセージ・システムに対して有効であるオプションのサブスクライバ・ルールを指定します。
- transformation ― ゲートウェイ定義のユーザー定義型を AQ ペイロード・タイプに変換するために使用される変換機能を指定します。

  ゲートウェイ伝播は、外部メッセージ・システムからメッセージをデキューし、メッセージ本体を既知のゲートウェイ定義のユーザー定義型に変換します。変換機能は、メッセージが AQ キューにエンキューされるときに、ゲートウェイ・ユーザー定義型を AQ ペイロード・タイプに変換するために使用されます。
- exception_queue ― 例外が発生した場合にメッセージを移動する外部キューの名前を指定します。これは、registered_queue@message_link の構文で指定する必要があります。

OUTBOUND_PROPAGATION の場合は、例外キューに関して次の点を考慮してください。

- 例外キューとして使用される AQ キューは、ユーザーが作成する必要があります。
- ソース・キューと例外キューのペイロード・タイプは一致している必要があります。
- 例外キューは、キュー・タイプ EXCEPTION_QUEUE ではなく、NORMAL_QUEUE で作成する必要があります。エンキューの制限事項によって、ゲートウェイ伝播は、ゲートウェイ例外キューとしてタイプ EXCEPTION_QUEUE の AQ キューを使用できません。

INBOUND_PROPAGATION の場合は、例外キューに関して次の点を考慮してください。

- 例外キューは、登録済みの Oracle 以外のキューであることが必要です。
- ソース・キューと例外キューは、同じメッセージ・システム・リンクを使用する必要があります。

REMOVE_SUBSCRIBER プロシージャ

このプロシージャは、伝播用のソース・キューから宛先にメッセージをコンシュームするために使用されるサブスクライバを削除します。

構文

```
DBMS_MGWADM.REMOVE_SUBSCRIBER (
   subscriber_id  IN VARCHAR2,
   force          IN BINARY_INTEGER DEFAULT DBMS_MGWADM.NO_FORCE );
```

パラメータ

表 31-29 REMOVE_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
subscriber_id	削除するサブスクライバを識別します。
force	ゲートウェイがこのサブスクライバに関連するすべてのクリーンアップ処理を実行できない場合でも、このプロシージャを正常終了させるかどうかを指定します。値は次のとおりです。 ■ NO_FORCE (0) クリーンアップを正常に完了できない場合は失敗します。 ■ FORCE (1) すべてのクリーンアップ処理を完了できない場合でも正常終了します。 ゲートウェイ・エージェントは、その伝播作業に、Oracle データベースおよび Oracle 以外のメッセージ・システムの様々なリソースを使用します。たとえば、メッセージをログ・キューにエンキューする、サブスクライバを作成するなどの作業があります。これらのリソースは通常、各サブスクライバと関連付けられ、サブスクライバが不要になったときに解放される必要があります。一般的に、このプロシージャは、ゲートウェイ・エージェントが実行中で、このサブスクライバに関連付けられた Oracle 以外のメッセージ・システムにアクセスできるときにのみコールしてください。

使用上の注意

アウトバウンド伝播の場合、ローカル・サブスクライバが AQ キューから削除されます。

RESET_SUBSCRIBER プロシージャ

このプロシージャは、サブスクライバの伝播エラー状態をリセットします。

構文

```
DBMS_MGWADM.RESET_SUBSCRIBER (
   subscriber_id   IN VARCHAR2 );
```

パラメータ

表 31-30 RESET_SUBSCRIBER プロシージャのパラメータ

パラメータ	説明
subscriber_id	サブスクライバを識別します。

SCHEDULE_PROPAGATION プロシージャ

このプロシージャは、ソースから宛先へのメッセージ伝播をスケジュールします。スケジュールが使用可能で、メッセージが伝播されるようにゲートウェイが適切に起動されている必要があります。

構文

```
DBMS_MGWADM.SCHEDULE_PROPAGATION (
   schedule_id       IN VARCHAR2,
   propagation_type  IN BINARY_INTEGER,
   source            IN VARCHAR2,
   destination       IN VARCHAR2,
   start_time        IN DATE DEFAULT SYSDATE,
   duration          IN NUMBER DEFAULT NULL,
   next_time         IN VARCHAR2 DEFAULT NULL,
   latency           IN NUMBER DEFAULT 60 );
```

パラメータ

表 31-31 SCHEDULE_PROPAGATION プロシージャのパラメータ

パラメータ	説明
schedule_id	スケジュールを識別するユーザー定義名を指定します。
propagation_type	メッセージ伝播のタイプを指定します。値は次のとおりです。 ■ DBMS_MGWADM.OUTBOUND_PROPAGATION: AQ から Oracle 以外への伝播の場合 ■ DBMS_MGWADM.INBOUND_PROPAGATION: Oracle 以外から AQ への伝播の場合
source	メッセージの伝播元であるソース・キューを指定します。このパラメータの構文および解釈は、propagation_type に指定された値によって決まります。
destination	メッセージの伝播先である宛先キューを指定します。このパラメータの構文および解釈は、propagation_type に指定された値によって決まります。
start_time	ソース・キューから宛先へのメッセージに対する伝播ウィンドウの初期開始時間を指定します。
duration	伝播ウィンドウの継続期間を指定します（秒数）。NULL 値の場合、伝播ウィンドウは無期限または伝播スケジュールが取り消されるまで継続します。
next_time	現在の伝播ウィンドウの終了から次の伝播ウィンドウの開始を計算する日付ファンクションを指定します。NULL 値の場合は、現在の伝播ウィンドウの終了時に伝播が終了します。
latency	伝播ウィンドウにおいて、メッセージのエンキュー後、メッセージが伝播されるまでの最大待機時間を指定します（秒数）。ただし、たとえば待機時間が 60 秒で、伝播を待機しているメッセージがない場合、伝播ウィンドウ内において、最低 60 秒間、ソースから宛先にメッセージは伝播されません。 待機時間が 0（ゼロ）の場合、メッセージはエンキュー後すぐに伝播されます。

使用上の注意

リリース 2（9.2）では、すべてのウィンドウ・パラメータが無視されます。

OUTBOUND_PROPAGATION の場合、パラメータは次のように解釈されます。

- source － 伝播ソースであるローカル AQ キューを指定します。これは、schema.queue の構文で指定する必要があります。
- destination － メッセージの伝播先の外部キューを指定します。これは、registered_queue@message_link の構文で指定する必要があります。

INBOUND_PROPAGATION の場合、パラメータは次のように解釈されます。

- source － 伝播ソースである外部キューを指定します。これは、registered_queue@message_link の構文で指定する必要があります。
- destination － メッセージの伝播先のローカル AQ キューを指定します。これは、schema.queue の構文で指定する必要があります。

スケジュールは、作成時に使用可能な状態に設定されます。

UNSCHEDULE_PROPAGATION プロシージャ

このプロシージャは、伝播スケジュールを削除します。

構文

```
DBMS_MGWADM.UNSCHEDULE_PROPAGATION (
   schedule_id   IN VARCHAR2 );
```

パラメータ

表 31-32 UNSCHEDULE_PROPAGATION プロシージャのパラメータ

パラメータ	説明
schedule_id	削除する伝播スケジュールを識別します。

ALTER_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、伝播スケジュールを変更します。

構文

```
DBMS_MGWADM.ALTER_PROPAGATION_SCHEDULE (
   schedule_id  IN VARCHAR2,
   duration     IN NUMBER DEFAULT NULL,
   next_time    IN VARCHAR2 DEFAULT NULL,
   latency      IN NUMBER DEFAULT 60 );
```

パラメータ

表 31-33 ALTER_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
schedule_id	変更する伝播スケジュールを識別します。
duration	伝播ウィンドウの継続期間を指定します（秒数）。NULL 値の場合、伝播ウィンドウは無期限または伝播スケジュールが取り消されるまで継続します。
next_time	現在の伝播ウィンドウの終了から次の伝播ウィンドウの開始を計算する日付ファンクションを指定します。NULL 値の場合は、現在の伝播ウィンドウの終了時に伝播が終了します。
latency	伝播ウィンドウにおいて、メッセージのエンキュー後、メッセージが伝播されるまでの最大待機時間を指定します（秒数）。ただし、たとえば待機時間が 60 秒で、伝播を待機しているメッセージがない場合、伝播ウィンドウ内において、最低 60 秒間、ソースから宛先にメッセージは伝播されません。 待機時間が 0（ゼロ）の場合、メッセージはエンキュー後すぐに伝播されます。

使用上の注意

リリース 2（9.2）では、伝播ウィンドウ・パラメータは無視されます。

注意： このプロシージャは、各パラメータの既存値を常に上書きします。特定のパラメータが指定されない場合、既存値はデフォルト値で上書きされます。

ENABLE_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、伝播スケジュールを使用可能にします。

構文

```
DBMS_MGWADM.ENABLE_PROPAGATION_SCHEDULE (
   schedule_id  IN VARCHAR2 );
```

パラメータ

表 31-34 ENABLE_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
schedule_id	使用可能にする伝播スケジュールを識別します。

DISABLE_PROPAGATION_SCHEDULE プロシージャ

このプロシージャは、伝播スケジュールを無効にします。

構文

```
DBMS_MGWADM.DISABLE_PROPAGATION_SCHEDULE (
   schedule_id  IN VARCHAR2 );
```

パラメータ

表 31-35 DISABLE_PROPAGATION_SCHEDULE プロシージャのパラメータ

パラメータ	説明
schedule_id	使用禁止にする伝播スケジュールを識別します。

データベース・ビューの要約

表 31-36 にリストしたビューは、メッセージ・ゲートウェイの構成、ステータスおよび統計情報を提供します。特に指示がないかぎり、SELECT 権限は MGW_ADMINISTRATOR_ROLE に付与されるため、メッセージ・ゲートウェイの管理者のみがビューにアクセスできます。すべてのビューの所有者は SYS です。

表 31-36 データベース・ビュー

名前	説明
MGW_GATEWAY ビュー	メッセージ・ゲートウェイの構成およびステータス情報をリストします。
MGW_LINKS ビュー	現在作成されているメッセージ・システム・リンクの名前とタイプをリストします。
MGW_MQSERIES_LINKS ビュー	MQSeries リンクのメッセージ・システム・プロパティをリストします。
MGW_FOREIGN_QUEUES ビュー	登録済みのキューのキュー・プロパティをリストします。
MGW_SUBSCRIBERS ビュー	サブスクライバのプロパティ、ステータスおよび統計情報をリストします。
MGW_SCHEDULES ビュー	スケジュールのプロパティおよびステータスをリストします。

MGW_GATEWAY ビュー

このビューは、表 31-37 に示されているように、メッセージ・ゲートウェイの構成およびステータス情報をリストします。

表 31-37 MGW_GATEWAY のプロパティ

名前	タイプ	説明
AGENT_STATUS	VARCHAR2	ゲートウェイ・エージェントのステータス。値は次のとおりです。 ■ NOT_STARTED: ゲートウェイ・エージェントは起動されていません。 ■ START_SCHEDULED: ゲートウェイ・エージェントは起動するようにスケジュールされています。これは、ゲートウェイが DBMS_MGWADM.STARTUP を使用して起動されているが、ゲートウェイ・エージェントの起動に使用されるキューイングされたジョブが実行されていないことを示します。 ■ STARTING: ゲートウェイ・エージェントは起動中です。これは、キューイングされたジョブが実行され、ゲートウェイ・エージェントが起動されていることを示します。 ■ INITIALIZING: ゲートウェイ・エージェントは起動され、初期化中です。 ■ RUNNING: ゲートウェイ・エージェントは実行中です。 ■ SHUTTING_DOWN: ゲートウェイ・エージェントは停止中です。
AGENT_PING	VARCHAR2	ゲートウェイ・エージェントの ping ステータス。値は次のとおりです。 ■ NULL: ping は行われていません。 ■ REACHABLE: ping は正常終了しました。 ■ UNREACHABLE: ping は失敗しました。 AGENT_PING はゲートウェイ・エージェントとの通信を試みます。ping が失敗した場合は、多少遅延（最大 5 秒）します。AGENT_STATUS が NOT_STARTED または START_SCHEDULED の場合、ping は行われません。
AGENT_JOB	NUMBER	メッセージ・ゲートウェイ・エージェント・プロセスの起動に使用されるキューイングされたジョブのジョブ番号。ジョブ番号は、メッセージ・ゲートウェイの起動時に設定され、停止時にクリアされます。
AGENT_USER	VARCHAR2	ゲートウェイ・エージェントがデータベースに接続するために使用するデータベース・ユーザー名。
AGENT_DATABASE	VARCHAR2	ゲートウェイ・エージェントが使用するデータベース接続文字列。NULL は、ローカル接続を使用することを示します。
LAST_ERROR_DATE	DATE	メッセージ・ゲートウェイ・エージェントの最終エラーの日付。最終エラー情報は、メッセージ・ゲートウェイの起動時にクリアされます。メッセージ・ゲートウェイ・エージェントが、なんらかの異常によって起動または終了に失敗した場合に設定されます。
LAST_ERROR_TIME	VARCHAR2	メッセージ・ゲートウェイ・エージェントの最終エラーの時間。

表 31-37 MGW_GATEWAY のプロパティ（続き）

名前	タイプ	説明
LAST_ERROR_MSG	VARCHAR2	メッセージ・ゲートウェイ・エージェントの最終エラーに関するメッセージ。
MAX_CONNECTIONS	NUMBER	Oracle データベースへのメッセージ接続の最大数。
MAX_MEMORY	NUMBER	ゲートウェイ・エージェントによって使用される最大ヒープ・サイズ（MB）。

MGW_LINKS ビュー

このビューは、現在定義されているメッセージ・システム・リンクの名前とタイプをリストします。

表 31-38 MGW_LINKS のプロパティ

名前	タイプ	説明
LINK_NAME	VARCHAR2	メッセージ・システム・リンクの名前。
LINK_TYPE	VARCHAR2	メッセージ・システム・リンクのタイプ。値は MQSERIES です。
LINK_COMMENT	VARCHAR2	リンクに関するユーザーのコメント。

MGW_MQSERIES_LINKS ビュー

このビューは、MQSeries メッセージ・システム・リンクの情報をリストします。ビューには、リンクの作成時に指定されるメッセージ・システム・プロパティの大部分が含まれています。

表 31-39 MGW_MQSERIES_LINKS のプロパティ

名前	タイプ	説明
LINK_NAME	VARCHAR2	メッセージ・システム・リンクの名前。
QUEUE_MANAGER	VARCHAR2	MQSeries キュー・マネージャの名前。
HOSTNAME	VARCHAR2	MQSeries ホストの名前。
PORT	NUMBER	ポート番号。
CHANNEL	VARCHAR2	接続チャネル。

表 31-39 MGW_MQSERIES_LINKS のプロパティ（続き）

名前	タイプ	説明
INTERFACE_TYPE	VARCHAR2	メッセージ・インタフェース・タイプ。値は、BASE_JAVA（MQSeries Base Java インタフェースの場合）です。
MAX_CONNECTIONS	NUMBER	メッセージ接続の最大数。
INBOUND_LOG_QUEUE	VARCHAR2	インバウンド伝播のログ・キュー。
OUTBOUND_LOG_QUEUE	VARCHAR2	アウトバウンド伝播のログ・キュー。
OPTIONS	SYS.MGW.PROPERTIES	リンク・オプション。
LINK_COMMENT	VARCHAR2	リンクに関するユーザーのコメント。

MGW_FOREIGN_QUEUES ビュー

このビューは、外部キューに関する情報をリストします。ビューには、キューの登録時に指定されるキュー・プロパティの大部分が含まれています。

表 31-40 MGW_FOREIGN_QUEUES のプロパティ

名前	タイプ	説明
NAME	VARCHAR2	登録されたキューの名前。
LINK_NAME	VARCHAR2	メッセージ・システム・リンクの名前。
PROVIDER_QUEUE	VARCHAR2	メッセージ・プロバイダ（ネイティブ）のキュー名。
DOMAIN	VARCHAR2	キュー・ドメイン・タイプ。値は次のとおりです。 ■ NULL: メッセージ・システムによって自動的に判断される場合 ■ QUEUE: キュー（Point-to-Point）モデルの場合 ■ TOPIC: トピック（パブリッシュ・サブスクライブ）モデルの場合
OPTIONS	SYS.MGW.PROPERTIES	オプションのキュー・プロパティ。
QUEUE_COMMENT	VARCHAR2	外部キューに関するユーザーのコメント。

MGW_SUBSCRIBERS ビュー

このビューは、メッセージ・ゲートウェイ・サブスクライバの構成およびステータス情報をリストします。ビューには、サブスクライバの追加時に指定されるサブスクライバ・プロパティの大部分が含まれる他、その他のステータスおよび統計情報が含まれます。

表 31-41 MGW_SUBSCRIBERS のプロパティ

名前	タイプ	説明
SUBSCRIBER_ID	VARCHAR2	伝播サブスクライバの識別子。
PROPAGATION_TYPE	VARCHAR2	伝播タイプ。値は次のとおりです。 ■ OUTBOUND: AQ から Oracle 以外への伝播の場合 ■ INBOUND: Oracle 以外から AQ への伝播の場合
QUEUE_NAME	VARCHAR2	サブスクライバのソース・キュー。
DESTINATION	VARCHAR2	メッセージの伝播先の宛先キュー。
RULE	VARCHAR2	サブスクリプション・ルール。
TRANSFORMATION	VARCHAR2	メッセージ変換に使用される変換機能。
EXCEPTION_QUEUE	VARCHAR2	ロギングの目的で使用される例外キュー。
STATUS	VARCHAR2	サブスクライバのステータス。値は次のとおりです。 ■ ENABLED: サブスクライバが使用可能な場合。 ■ DELETE_PENDING: サブスクライバの削除が保留中の場合。通常は、DBMS_MGWADM.REMOVE_SUBSCRIBER がコール済みで、このサブスクライバに関係する特定のクリーンアップ・タスクが未処理の場合です。
FAILURES	NUMBER	伝播の失敗数。
LAST_ERROR_DATE	DATE	伝播の最終エラーの日付。
LAST_ERROR_TIME	VARCHAR2	伝播の最終エラーの時間。
LAST_ERROR_MSG	VARCHAR2	伝播の最終エラーに関するメッセージ。
PROPAGATED_MSGS	NUMBER	最後にエージェントが起動されてから宛先キューに伝播されたメッセージの数。
EXCEPTIONQ_MSGS	NUMBER	最後にエージェントが起動されてから伝播例外キューに移動されたメッセージの数。

MGW_SCHEDULES ビュー

このビューは、メッセージ・ゲートウェイ・スケジュールの構成およびステータス情報をリストします。ビューには、スケジュールの作成時に指定されるスケジュール・プロパティの大部分が含まれる他、その他のステータス情報が含まれます。

表 31-42 MGW_SCHEDULES のプロパティ

名前	タイプ	説明
SCHEDULE_ID	VARCHAR2	伝播スケジュールの識別子。
PROPAGATION_TYPE	VARCHAR2	伝播タイプ。値は次のとおりです。 ■ OUTBOUND: AQ から Oracle 以外への伝播の場合 ■ INBOUND: Oracle 以外から AQ への伝播の場合
SOURCE	VARCHAR2	伝播ソース。
DESTINATION	VARCHAR2	伝播宛先。
START_DATE	DATE	スケジュール開始日付。
START_TIME	VARCHAR2	スケジュール開始時間。
PROPAGATION_WINDOW	NUMBER	伝播ウィンドウの継続期間（秒数）。
NEXT_TIME	VARCHAR2	次の伝播ウィンドウの開始を計算するために使用される日付ファンクション。
LATENCY	NUMBER	伝播ウィンドウ待機時間（秒数）。
SCHEDULE_DISABLED	VARCHAR2	スケジュールが使用禁止であるかどうかを示します。値は次のとおりです。 ■ Y: スケジュールが使用禁止の場合 ■ N: スケジュールが使用可能の場合

32

DBMS_MGWMSG

DBMS_MGWMSG は、オブジェクト・タイプ（メッセージ本体を変換するために標準的なメッセージ・タイプで使用）、およびメッセージ・ゲートウェイ・メッセージ・タイプを処理するためのヘルパー・メソッド、定数およびサブプログラムを提供します。パッケージとオブジェクト・タイプの所有者は SYS です。

注意： catmgw.sql スクリプトを起動して、メッセージ・ゲートウェイのパッケージとタイプをデータベースにロードする必要があります。データベース・オブジェクトのロードの詳細は、『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。

関連項目： 『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』に、DBMS_MGWMSG の使用に関する情報が含まれています。

この章では、次の項目について説明します。

DBMS_MGWMSG のオブジェクト・タイプとメソッドの要約

表 32-1 DBMS_MGWMSG のオブジェクト・タイプとメソッド

オブジェクト・タイプ	説明
MGW_NAME_VALUE_T タイプ	名前付きの値を指定します。
MGW_NAME_VALUE_T.CONSTRUCT メソッド	新規 MGW_NAME_VALUE_T インスタンスを構成します。
MGW_NAME_VALUE_T.CONSTRUCT_<TYPE> メソッド	特定のタイプの値で初期化される新規 MGW_NAME_VALUE_T インスタンスを構成します。
MGW_NAME_TYPE_ARRAY_T タイプ	名前 / 値ペアの配列を指定します。
MGW_TEXT_VALUE_T タイプ	TEXT 値を指定します。
MGW_TEXT_VALUE_T.CONSTRUCT メソッド	新規 MGW_TEXT_VALUE_T インスタンスを構成します。
MGW_RAW_VALUE_T タイプ	RAW 値を指定します。
MGW_RAW_VALUE_T.CONSTRUCT メソッド	新規 MGW_RAW_VALUE_T インスタンスを構成します。
MGW_BASIC_MSG_T タイプ	基本的な TEXT または RAW メッセージの標準的なタイプ。
MGW_BASIC_MSG_T.CONSTRUCT メソッド	新規 MGW_BASIC_MSG_T インスタンスを構成します。

MGW_NAME_VALUE_T タイプ

このタイプは、名前付きの値を指定します。name 属性と type 属性、および < >_value 属性の 1 つが、通常 NULL 以外の値です。

構文

```
TYPE SYS.MGW_NAME_VALUE_T IS OBJECT
   name                VARCHAR2(250),
   type                INTEGER,
   integer_value       INTEGER,
   number_value        NUMBER,
   text_value          VARCHAR2(4000),
   raw_value           RAW(2000),
   date_value          DATE);
```

属性

表 32-2 MGW_NAME_VALUE_T の属性

属性	説明
name	値に関連付けられた名前。
type	値のタイプ。表 32-7 の DBMS_MGWMSG.< >_VALUE 定数を参照してください。これは、どの Java データ・タイプとクラスが値と関連付けられているかを示します。また、どの属性に値が格納されているかも示します。
integer_value	整数値を格納します。
number_value	float タイプまたはラージ・タイプの整数値を格納します。
text_value	TEXT 値を格納します。
raw_value	RAW（バイト）値を格納します。
date_value	日付値を格納します。

タイプと属性のマッピング

表 32-3 に、値のタイプと、値の格納に使用される属性との間のマッピングを示します。

表 32-3 タイプと属性のマッピング

タイプ	属性に格納される値
DBMS_MGWMSG.TEXT_VALUE	text_value
DBMS_MGWMSG.RAW_VALUE	raw_value
DBMS_MGWMSG.BOOLEAN_VALUE	integer_value
DBMS_MGWMSG.BYTE_VALUE	integer_value
DBMS_MGWMSG.SHORT_VALUE	integer_value
DBMS_MGWMSG.INTEGER_VALUE	integer_value
DBMS_MGWMSG.LONG_VALUE	number_value
DBMS_MGWMSG.FLOAT_VALUE	number_value
DBMS_MGWMSG.DOUBLE_VALUE	number_value
DBMS_MGWMSG.DATE_VALUE	date_value

MGW_NAME_VALUE_T.CONSTRUCT メソッド

このメソッドは、新規 MGW_NAME_VALUE_T インスタンスを構成します。すべての属性に NULL が割り当てられます。

構文

```
STATIC FUNCTION CONSTRUCT
RETURN SYS.MGW_NAME_VALUE_T;
```

MGW_NAME_VALUE_T.CONSTRUCT_<TYPE> メソッド

これらのメソッドは、特定のタイプの値で初期化される新規 MGW_NAME_VALUE_T インスタンスを構成します。各メソッドは、表 32-3 のマッピングに示されているように、name 属性と type 属性、および < >_value 属性の 1 つを設定します。

構文

```
STATIC FUNCTION CONSTRUCT_BOOLEAN (
   name   IN VARCHAR2,
   value  IN INTEGER )
RETURN SYS.MGW_NAME_VALUE_T,

STATIC FUNCTION CONSTRUCT_BYTE (
   name   IN VARCHAR2,
   value  IN INTEGER )
RETURN SYS.MGW_NAME_VALUE_T,

STATIC FUNCTION CONSTRUCT_SHORT (
   name   IN VARCHAR2,
   value  IN INTEGER )
RETURN SYS.MGW_NAME_VALUE_T,

STATIC FUNCTION CONSTRUCT_INTEGER (
   name   IN VARCHAR2,
   value  IN INTEGER )
RETURN SYS.MGW_NAME_VALUE_T,

STATIC FUNCTION CONSTRUCT_LONG (
   name   IN VARCHAR2,
   value  IN NUMBER )
RETURN SYS.MGW_NAME_VALUE_T,
```

```
STATIC FUNCTION CONSTRUCT_FLOAT (
   name   IN VARCHAR2,
   value  IN NUMBER )
RETURN SYS.MGW_NAME_VALUE_T,

STATIC FUNCTION CONSTRUCT_DOUBLE (
   name   IN VARCHAR2,
   value  IN NUMBER )
RETURN SYS.MGW_NAME_VALUE_T,

STATIC FUNCTION CONSTRUCT_TEXT (
   name   IN VARCHAR2,
   value  IN VARCHAR2 )
RETURN SYS.MGW_NAME_VALUE_T,

STATIC FUNCTION CONSTRUCT_RAW (
   name  IN VARCHAR2,
   value  IN RAW )
RETURN SYS.MGW_NAME_VALUE_T,

STATIC FUNCTION CONSTRUCT_DATE (
   name   IN VARCHAR2,
   value  IN DATE )
RETURN SYS.MGW_NAME_VALUE_T );
```

使用上の注意

construct_boolean メソッドは、値を DBMS_MGWMSG.BOOLEAN_TRUE または DBMS_MGWMSG.BOOLEAN_FALSE のいずれかに設定します。

MGW_NAME_TYPE_ARRAY_T タイプ

このタイプは、名前 / 値ペアの配列を指定します。MGW_NAME_VALUE_ARRAY_T タイプのオブジェクトには、要素を 1024 個まで設定できます。

構文

```
TYPE SYS.MGW_NAME_VALUE_ARRAY_T AS VARRAY (1024) OF SYS.MGW_NAME_VALUE_T;
```

MGW_TEXT_VALUE_T タイプ

このタイプは、TEXT 値を指定します。大きい値は CLOB で、比較的小さい値（サイズ <= 4000）は VARCHAR2 で格納できます。< >_ value 属性は、その中の 1 つのみ設定してください。

構文

```
TYPE SYS.MGW_TEXT_VALUE_T IS OBJECT
   small_value VARCHAR2(4000),
   large_value CLOB);
```

属性

表 32-4 MGW_TEXT_VALUE_T の属性

属性	説明
small_value	小さい TEXT 値。値 <= 4000 の場合に使用されます。
large_value	大きい TEXT 値。small_value 属性を使用するには値が大きすぎる場合に使用されます。

MGW_TEXT_VALUE_T.CONSTRUCT メソッド

このメソッドは、新規 MGW_TEXT_VALUE_T インスタンスを構成します。すべての属性に NULL が割り当てられます。

構文

```
STATIC FUNCTION CONSTRUCT
RETURN SYS.MGW_TEXT_VALUE_T;
```

MGW_RAW_VALUE_T タイプ

このタイプは、RAW 値を指定します。大きい値は BLOB で、比較的小さい値（サイズ <= 2000）は RAW で格納できます。< >_value 属性は、その中の 1 つのみ設定してください。

構文

```
TYPE SYS.MGW_RAW_VALUE_T IS OBJECT(
   small_value RAW(2000),
   large_value BLOB);
```

属性

表 32-5 MGW_RAW_VALUE_T の属性

属性	説明
small_value	小さい RAW（バイト）値 <= 2000。
large_value	大きい RAW 値。small_value 属性を使用するには値が大きすぎる場合に使用されます。

MGW_RAW_VALUE_T.CONSTRUCT メソッド

このメソッドは、新規 MGW_RAW_VALUE_T インスタンスを構成します。すべての属性に NULL が割り当てられます。

構文

```
STATIC FUNCTION CONSTRUCT
RETURN SYS.MGW_RAW_VALUE_T;
```

MGW_BASIC_MSG_T タイプ

基本的な TEXT または RAW メッセージの標準的なタイプ。通常、単一の TEXT または RAW の値のみが設定されます。このタイプのオブジェクトでは、TEXT と RAW の両方に NULL 以外の値を同時に設定しないでください。

構文

```
TYPE SYS.MGW_BASIC_MSG_T IS OBJECT
   header     SYS.MGW_NAME_VALUE_ARRAY_T,
   text_body  SYS.MGW_TEXT_VALUE_T,
   raw_body   SYS.MGW_RAW_VALUE_T);
```

属性

表 32-6 MGW_BASIC_MSG_T の属性

属性	説明
header	名前 / 値ペアの配列としてのメッセージ・ヘッダー情報。
text_body	TEXT メッセージのメッセージ本体。
raw_body	RAW（バイト）メッセージのメッセージ本体。

MGW_BASIC_MSG_T.CONSTRUCT メソッド

このメソッドは、新規 MGW_BASIC_MSG_T インスタンスを構成します。すべての属性に NULL が割り当てられます。

構文

```
STATIC FUNCTION CONSTRUCT
RETURN SYS.MGW_BASIC_MSG_T;
```

DBMS_MGWMSG 定数

表 32-7 DBMS_MGWMSG 定数 : 値のタイプー SYS.MGW_NAME_VALUE_T オブジェクトの値のタイプを表す定数

値	定数
TEXT_VALUE	CONSTANT BINARY_INTEGER := 1;
RAW_VALUE	CONSTANT BINARY_INTEGER := 2;
BOOLEAN_VALUE	CONSTANT BINARY_INTEGER := 3;
BYTE_VALUE	CONSTANT BINARY_INTEGER := 4;
SHORT_VALUE	CONSTANT BINARY_INTEGER := 5;
INTEGER_VALUE	CONSTANT BINARY_INTEGER := 6;
LONG_VALUE	CONSTANT BINARY_INTEGER := 7;
FLOAT_VALUE	CONSTANT BINARY_INTEGER := 8;
DOUBLE_VALUE	CONSTANT BINARY_INTEGER := 9;
DATE_VALUE	CONSTANT BINARY_INTEGER := 10;

表 32-8 DBMS_MGWMSG 定数 : ブール値ーブール値を数値で表す定数

値	定数
BOOLEAN_FALSE	CONSTANT BINARY_INTEGER := 0;
BOOLEAN_TRUE	CONSTANT BINARY_INTEGER := 1;

表 32-9 DBMS_MGWMSG 定数 : 大文字と小文字の比較

値	定数
CASE_SENSITIVE	CONSTANT BINARY_INTEGER := 0;
CASE_INSENSITIVE	CONSTANT BINARY_INTEGER := 1;

DBMS_MGWMSG サブプログラムの要約

表 32-10 DBMS_MGWMSG のサブプログラム

サブプログラム	説明
「NVARRAY_ADD プロシージャ」32-11 ページ	名前 / 値の配列の終わりに名前 / 値の要素を追加します。
「NVARRAY_GET ファンクション」32-11 ページ	p_name に指定した名前の名前 / 値の要素を名前 / 値配列から取得します。
「NVARRAY_GET_BOOLEAN ファンクション」32-12 ページ	p_name に指定した名前 / 値配列の要素で、BOOLEAN_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_BYTE ファンクション」32-13 ページ	p_name に指定した名前 / 値配列の要素で、BYTE_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_SHORT ファンクション」32-13 ページ	p_name に指定した名前 / 値配列の要素で、SHORT_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_INTEGER ファンクション」32-14 ページ	p_name に指定した名前 / 値配列の要素で、INTEGER_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_LONG ファンクション」32-15 ページ	p_name に指定した名前 / 値配列の要素で、LONG_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_FLOAT ファンクション」32-15 ページ	p_name に指定した名前 / 値配列の要素で、FLOAT_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_DOUBLE ファンクション」32-16 ページ	p_name に指定した名前 / 値配列の要素で、DOUBLE_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_TEXT ファンクション」32-17 ページ	p_name に指定した名前 / 値配列の要素で、TEXT_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_RAW ファンクション」32-17 ページ	p_name に指定した名前 / 値配列の要素で、RAW_VALUE 値タイプの要素の値を取得します。
「NVARRAY_GET_DATE ファンクション」32-18 ページ	p_name に指定した名前 / 値配列の要素で、DATE_VALUE 値タイプの要素の値を取得します。
「NVARRAY_FIND_NAME ファンクション」32-19 ページ	名前 / 値配列を検索し、p_name に指定した名前を持つ要素を探します。
「NVARRAY_FIND_NAME_TYPE ファンクション」32-20 ページ	名前 / 値配列を検索し、指定した名前と値タイプを持つ要素を探します。

NVARRAY_ADD プロシージャ

このプロシージャは、名前 / 値の配列の終わりに名前 / 値の要素を追加します。

構文

```
DBMS_MGWMSG.NVARRAY_ADD (
   p_array  IN OUT SYS.MGW_NAME_VALUE_ARRAY_T,
   p_value  IN     SYS.MGW_NAME_VALUE_T );
```

パラメータ

表 32-11 NVARRAY_ADD プロシージャのパラメータ

パラメータ	説明
p_array	名前 / 値配列のインスタンス。入力時は変更する配列です。NULL の場合は新しい配列が作成されます。出力時は変更後の配列です。
p_value	追加する値。NULL の場合、p_array は変更されません。

NVARRAY_GET ファンクション

このファンクションは、p_name に指定した名前の名前 / 値の要素を名前 / 値配列から取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN SYS.MGW_NAME_VALUE_T;
```

パラメータ

表 32-12 NVARAAY_GET ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

一致要素、または指定した名前が見つからない場合は NULL。

NVARRAY_GET_BOOLEAN ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、BOOLEAN_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_BOOLEAN (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN INTEGER;
```

パラメータ

表 32-13 NVARRAY_GET_BOOLEAN ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_BYTE ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、BYTE_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_BYTE (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN INTEGER;
```

パラメータ

表 32-14 NVARRAY_GET_BYTE ファンクション

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_SHORT ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、SHORT_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_SHORT (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN INTEGER;
```

パラメータ

表 32-15 NVARRAY_GET_SHORT ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_INTEGER ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、INTEGER_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_INTEGER (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN INTEGER;
```

パラメータ

表 32-16 NVARRAY_GET_INTEGER ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_LONG ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、LONG_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_LONG (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN NUMBER;
```

パラメータ

表 32-17 NVARRAY_GET_LONG ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_FLOAT ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、FLOAT_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_FLOAT (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN NUMBER;
```

パラメータ

表 32-18 NVARRAY_GET_FLOAT ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_DOUBLE ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、DOUBLE_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_DOUBLE (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN NUMBER;
```

パラメータ

表 32-19 NVARRAY_GET_DOUBLE ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_TEXT ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、TEXT_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_TEXT (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN VARCHAR2;
```

パラメータ

表 32-20 NVARRAY_GET_TEXT ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_RAW ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、RAW_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_RAW (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN RAW;
```

パラメータ

表 32-21 NVARRAY_GET_RAW ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_GET_DATE ファンクション

このファンクションは、p_name に指定した名前 / 値配列の要素で、DATE_VALUE 値タイプの要素の値を取得します。

構文

```
DBMS_MGWMSG.NVARRAY_GET_DATE (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN DATE;
```

パラメータ

表 32-22 NVARRAY_GET_DATE ファンクションのパラメータ

パラメータ	説明
p_array	名前 / 値配列。
p_name	値名。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

値。または指定した名前が見つからないか、タイプの不一致がある場合は NULL。

NVARRAY_FIND_NAME ファンクション

このファンクションは、名前 / 値配列を検索し、p_name に指定した名前を持つ要素を探します。

構文

```
DBMS_MGWMSG.NVARRAY_FIND_NAME (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN BINARY_INTEGER;
```

パラメータ

表 32-23 NVARRAY_FIND_NAME ファンクションのパラメータ

パラメータ	説明
p_array	検索する名前 / 値配列。
p_name	検索する名前。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

- 一致要素の配列の索引である正の整数
- 0: 指定した名前が見つからない場合

NVARRAY_FIND_NAME_TYPE ファンクション

このファンクションは、名前 / 値配列を検索し、指定した名前と値タイプを持つ要素を探します。

構文

```
DBMS_MGWMSG.NVARRAY_FIND_NAME_TYPE (
   p_array    IN SYS.MGW_NAME_VALUE_ARRAY_T,
   p_name     IN VARCHAR2,
   p_type     IN BINARY_INTEGER
   p_compare  IN BINARY_INTEGER DEFAULT CASE_SENSITIVE )
RETURN BINARY_INTEGER;
```

パラメータ

表 32-24 NVARRAY_FIND_NAME_TYPE ファンクションのパラメータ

パラメータ	説明
p_array	検索する名前 / 値配列。
p_name	検索する名前。
p_type	値のタイプ。32-9 ページの表 32-7 にリストされている値タイプの定数を参照してください。
p_compare	名前比較方法。値は、CASE_SENSITIVE、CASE_INSENSITIVE です。

戻り値

- 一致要素の配列の索引である正の整数
- 0: 指定した名前が見つからない場合
- -1: 指定した名前は見つかったが、タイプの不一致がある場合

33

DBMS_MVIEW

DBMS_MVIEW を使用すると、リライトの可用性の他、マテリアライズド・ビューおよび潜在的なマテリアライズド・ビューの機能を理解できます。また、同じリフレッシュ・グループおよびパージ・ログの一部ではないマテリアライズド・ビューをリフレッシュできます。

この章では、次の項目について説明します。

注意： DBMS_SNAPSHOT は、DBMS_MVIEW のシノニムです。

関連項目：

- レプリケーション環境におけるマテリアライズド・ビューの使用の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。
- ウェアハウス環境におけるマテリアライズド・ビューの使用の詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

DBMS_MVIEW サブプログラムの要約

表 33-1 DBMS_MVIEW パッケージのサブプログラム

サブプログラム	説明
「BEGIN_TABLE_REORGANIZATION プロシージャ」33-3 ページ	リフレッシュに必要なマテリアライズド・ビュー・データを保存するプロセスを実行します。
「END_TABLE_REORGANIZATION プロシージャ」33-4 ページ	マスター表のマテリアライズド・ビュー・データが有効であり、マスター表が適切な状態であることを確認します。
「EXPLAIN_MVIEW プロシージャ」33-4 ページ	マテリアライズド・ビューまたは潜在的なマテリアライズド・ビューを使用して実行できる事項を説明します。
「EXPLAIN_REWRITE プロシージャ」33-6 ページ	クエリー・リライトが失敗した理由を説明します。
「I_AM_A_REFRESH ファンクション」33-7 ページ	I_AM_REFRESH パッケージの状態の値を戻します。
「PMARKER ファンクション」33-7 ページ	ROWID からパーティション作成者を戻します。このファンクションは、パーティション変更追跡（PCT）に使用されます。
「PURGE_DIRECT_LOAD_LOG プロシージャ」33-7 ページ	マテリアライズド・ビューで行が不要になると、ダイレクト・ローダー・ログからその行を削除します（データ・ウェアハウスで使用）。
「PURGE_LOG プロシージャ」33-8 ページ	マテリアライズド・ビュー・ログから行をパージします。
「PURGE_MVIEW_FROM_LOG プロシージャ」33-9 ページ	マテリアライズド・ビュー・ログから行をパージします。
「REFRESH プロシージャ」33-11 ページ	同じリフレッシュ・グループのメンバーではない 1 つ以上のマテリアライズド・ビューをリフレッシュします。
「REFRESH_ALL_MVIEWS プロシージャ」33-14 ページ	マスター表またはマスター・マテリアライズド・ビューに変更を反映していないマテリアライズド・ビューをすべてリフレッシュします。
「REFRESH_DEPENDENT プロシージャ」33-15 ページ	指定したマスター表またはマスター・マテリアライズド・ビューに依存する表ベースのマテリアライズド・ビュー、またはマスター表またはマスター・マテリアライズド・ビューをリフレッシュします。

表 33-1 DBMS_MVIEW パッケージのサブプログラム（続き）

サブプログラム	説明
「REGISTER_MVIEW プロシージャ」33-18 ページ	個々のマテリアライズド・ビューの管理を可能にします。
「UNREGISTER_MVIEW プロシージャ」33-20 ページ	個々のマテリアライズド・ビューの管理を可能にします。マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで起動し、マテリアライズド・ビューを登録解除します。

注意： 問合せが 256 文字未満の場合は、SQL*PLUS から EXECUTE コマンドを使用して EXPLAIN_REWRITE() を起動できます。それ以外の場合は、/rdbms/demo/smxrw.sql の例に示されているように、PL/SQL の BEGIN..END ブロックを使用する方法をお薦めします。EXPLAIN_REWRITE() API は、32627 文字を超える問合せを受け入れることはできません。これらの制限は、マテリアライズド・ビューの定義問合せを EXPLAIN_MVIEW プロシージャに渡すときにも適用されます。

BEGIN_TABLE_REORGANIZATION プロシージャ

このプロシージャは、リフレッシュに必要なマテリアライズド・ビュー・データを保存するプロセスを実行します。このプロシージャは、マスター表の再編成前にコールする必要があります。

構文

```
DBMS_MVIEW.BEGIN_TABLE_REORGANIZATION (
   tabowner     IN   VARCHAR2,
   tabname      IN   VARCHAR2);
```

パラメータ

表 33-2 BEGIN_TABLE_REORGANIZATION プロシージャのパラメータ

パラメータ	説明
tabowner	再編成する表の所有者。
tabname	再編成する表の名前。

END_TABLE_REORGANIZATION プロシージャ

このプロシージャは、マスター表のマテリアライズド・ビュー・データが有効であり、マスター表が適切な状態であることを確認します。このプロシージャは、マスター表の再編成後にコールする必要があります。

構文

```
DBMS_MVIEW.END_TABLE_REORGANIZATION (
   tabowner    IN   VARCHAR2,
   tabname     IN   VARCHAR2);
```

パラメータ

表 33-3 END_TABLE_REORGANIZATION プロシージャのパラメータ

パラメータ	説明
tabowner	再編成する表の所有者。
tabname	再編成する表の名前。

EXPLAIN_MVIEW プロシージャ

このプロシージャは、マテリアライズド・ビューまたは潜在的なマテリアライズド・ビューを使用して実行できる事項を説明します。たとえば、マテリアライズド・ビューが高速リフレッシュ可能であるかどうか、また、特定のマテリアライズド・ビューを使用して実行できるクエリー・リライトのタイプなどを判別します。

このプロシージャの使用方法は簡潔です。DBMS_MVIEW.EXPLAIN_MVIEW をコールします。既存のマテリアライズド・ビューにパラメータとしてスキーマおよびマテリアライズド・ビュー名を渡します。また、潜在的なマテリアライズド・ビューには、SELECT 文字列を指定できます。マテリアライズド・ビューまたは潜在的なマテリアライズド・ビューは分析され、その結果が MV_CAPABILITIES_TABLE と呼ばれる表（デフォルト）または MSG_ARRAY と呼ばれる配列に書き込まれます。

EXPLAIN_MVIEW をコールする前に utlxmv.sql スクリプトを実行する必要があります。ただし、VARRAY に出力を送る場合を除きます。スクリプトは admin ディレクトリに格納されています。さらに、現行のスキーマにおいて MV_CAPABILITIES_TABLE を作成する必要があります。

構文

次の PL/SQL 宣言は、ユーザーのために DBMS_MVIEW パッケージに作成されたもので、既存のマテリアライズド・ビューおよび潜在的なマテリアライズド・ビューを説明するパラメータの順序およびデータ・タイプを、表および VARRAY に出力して示します。

既存または潜在的なマテリアライズド・ビューを MV_CAPABILITIES_TABLE に出力して説明するには、次の手順を実行します。

```
DBMS_MVIEW.EXPLAIN_MVIEW (
 mv             IN VARCHAR2,
 statement_id   IN VARCHAR2:= NULL);
```

既存または潜在的なマテリアライズド・ビューを VARRAY に出力して説明するには、次の手順を実行します。

```
DBMS_MVIEW.EXPLAIN_MVIEW (
 mv           IN VARCHAR2,
 msg_array    OUT SYS.ExplainMVArrayType);
```

パラメータ

表 33-4 EXPLAIN_MVIEW プロシージャのパラメータ

パラメータ	説明
mv	既存のマテリアライズド・ビューの名前（オプションで「.（ピリオド）」で区切られた所有者の名前を使用して修飾）または潜在的なマテリアライズド・ビューの SELECT 文。
statement_id	クライアントが提供する一意の識別子。出力行を特定の EXPLAIN_MVIEW の起動と関連付けます。
msg_array	出力を受け取る PL/SQL VARRAY。このパラメータを使用して、MV_CAPABILITIES_TABLE ではなく EXPLAIN_MVIEW の出力を PL/SQL VARRAY に送ります。

EXPLAIN_REWRITE プロシージャ

このプロシージャを使用すると、クエリー・リライトに失敗した理由や、リライトした場合は使用されるマテリアライズド・ビューを確認できます。プロシージャから得た結果を使用することで、クエリー・リライトに必要な適切なアクションを可能なかぎり実行できます。EXPLAIN_REWRITE 文で指定された問合せが、実際に実行されることはありません。

出力を表に取得するには、EXPLAIN_REWRITE をコールする前に utlxrw.sql スクリプトを実行する必要があります。このスクリプトは、現行のスキーマに REWRITE_TABLE という名前の表を作成します。

構文

EXPLAIN_REWRITE から出力を取得する方法は 2 通りあります。1 つは表を使用する方法です。もう 1 つは、VARRAY を作成する方法です。次に、出力表を使用する場合の基本的な構文を示します。

```
DBMS_MVIEW.EXPLAIN_REWRITE (
    query           IN VARCHAR2,
    mv              IN VARCHAR2,
    statement_id    IN VARCHAR2;
```

表のかわりに EXPLAIN_REWRITE の出力を VARRAY に送る場合は、プロシージャを次のようにコールする必要があります。

```
DBMS_MVIEW.EXPLAIN_REWRITE (
    query           IN VARCHAR2(2000),
    mv              IN VARCHAR2(30),
    msg_array       IN OUT SYS.RewriteArrayType);
```

パラメータ

表 33-5 EXPLAIN_REWRITE プロシージャのパラメータ

パラメータ	説明
query	説明する SQL SELECT 文。
mv	既存マテリアライズド・ビューの SCHEMA.MV 形式の完全修飾名。
statement_id	出力メッセージを区別するためにクライアントが提供する一意の識別子。
msg_array	出力を受け取る PL/SQL VARRAY。このパラメータを使用して、EXPLAIN_REWRITE の出力を PL/SQL VARRAY に送ります。

I_AM_A_REFRESH ファンクション

このファンクションは、I_AM_REFRESH パッケージの状態の値を戻します。戻り値が TRUE の場合は、各レプリケーション・トリガーが最初にこの状態をチェックするため、このセッションではマテリアライズド・ビューのすべてのローカル・レプリケーション・トリガーが完全に使用禁止になります。戻り値が FALSE の場合は、そのトリガーを使用できます。

構文

```
DBMS_MVIEW.I_AM_A_REFRESH()
  RETURN BOOLEAN;
```

PMARKER ファンクション

このファンクションは、ROWID からパーティション作成者を戻します。パーティション変更追跡（PCT）に使用されます。

構文

```
DBMS_MVIEW.PMARKER(rid IN ROWID)
  RETURN NUMBER;
```

パラメータ

表 33-6 PMARKER プロシージャのパラメータ

パラメータ	説明
rid	マスター表における行エントリの ROWID。

PURGE_DIRECT_LOAD_LOG プロシージャ

このプロシージャは、既知のマテリアライズド・ビューでエントリが不要になると、ダイレクト・ローダー・ログからそのエントリを削除します。このプロシージャは通常、Oracle のデータ・ウェアハウス・テクノロジを使用する環境で使用されます。

> **関連項目：** 詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

構文

```
DBMS_MVIEW.PURGE_DIRECT_LOAD_LOG();
```

PURGE_LOG プロシージャ

このプロシージャは、マテリアライズド・ビュー・ログから行を削除します。

構文

```
DBMS_MVIEW.PURGE_LOG (
   master        IN   VARCHAR2,
   num           IN   BINARY_INTEGER := 1,
   flag          IN   VARCHAR2       := 'NOP');
```

パラメータ

表 33-7 PURGE_LOG プロシージャのパラメータ

パラメータ	説明
master	マスター表またはマスター・マテリアライズド・ビューの名前。

num	マテリアライズド・ビュー・ログから行を削除する行の中で、リフレッシュ日付が最も古いマテリアライズド・ビューの数。たとえば、次の文は、リフレッシュ日付が最も古い 2 つのマテリアライズド・ビューをリフレッシュするために必要な行を削除します。
	`DBMS_MVIEW.PURGE_LOG('master_table', 2);`
	マテリアライズド・ビュー・ログにあるすべての行を削除するには、次の例のように、削除するマテリアライズド・ビューについて大きい数を指定します。
	`DBMS_MVIEW.PURGE_LOG('master_table',9999);`
	この文は、MASTER_TABLE に基づくマテリアライズド・ビューの数が 9999 未満の場合、MASTER_TABLE に対応するマテリアライズド・ビュー・ログを完全に削除します。行がマテリアライズド・ビューからすでに削除されている単純マテリアライズド・ビューは、次回リフレッシュ時に完全にリフレッシュする必要があります。

flag	delete を指定して、少なくとも 1 つのマテリアライズド・ビューに対し、マテリアライズド・ビューから行を削除することを保証します。このパラメータは、パラメータ num の設定を上書きできます。たとえば、次の文は、リフレッシュ日付が最も古いマテリアライズド・ビューに従属行を持つマテリアライズド・ビュー・ログから行を削除します。 `DBMS_MVIEW.PURGE_LOG('master_table',1,'delete');`

PURGE_MVIEW_FROM_LOG プロシージャ

このプロシージャは、マテリアライズド・ビューのリフレッシュに関連したデータ・ディクショナリ表にある行を削除するためにマスター・サイトまたはマスター・マテリアライズド・ビューのサイトでコールされます。この表は mview_id または mviewowner、mviewname および mviewsite の組合せで識別される指定マテリアライズド・ビューについて、マスター・サイトでメンテナンスされている表です。指定したマテリアライズド・ビューが、任意のマスター表からリフレッシュされた最も古いマテリアライズド・ビューの場合、またはマスター・マテリアライズド・ビューの場合は、そのマテリアライズド・ビュー・ログもパージされます。このプロシージャは、マテリアライズド・ビューの登録解除は行いません。

マテリアライズド・ビュー・ログの 1 つを削除している間にエラーが発生した場合、それ以前に正常終了したマテリアライズド・ビュー・ログの削除処理はロールバックされません。これは、マテリアライズド・ビュー・ログのサイズを最小限にするためです。このプロシージャは、エラーが発生した場合でも、すべてのマテリアライズド・ビュー・ログが削除されるまで再起動できます。

構文

```
DBMS_MVIEW.PURGE_MVIEW_FROM_LOG (
   mview_id       IN   BINARY_INTEGER  |
   mviewowner     IN   VARCHAR2,
   mviewname      IN   VARCHAR2,
   mviewsite      IN   VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。mview_id には、同時に指定することのできないパラメータが 3 つあります。mviewowner、mviewname および mviewsite です。

パラメータ

表 33-8 PURGE_MVIEW_FROM_LOG プロシージャのパラメータ

パラメータ	説明
mview_id	ターゲット・マテリアライズド・ビューの識別に基づいてこのプロシージャを実行するには、mview_id パラメータを使用してマテリアライズド・ビューの識別を指定します。マテリアライズド・ビュー・ログ・サイトで、マテリアライズド・ビュー ID のリスト表示について DBA_BASE_TABLE_MVIEWS ビューを問い合せます。 登録済みマテリアライズド・ビューのリスト（DBA_REGISTERED_MVIEWS）にターゲット・マテリアライズド・ビューがない場合は、マテリアライズド・ビュー ID に基づいてこのプロシージャを実行すると便利です。
mviewowner	mview_id を指定しない場合は、mviewowner パラメータを使用してターゲット・マテリアライズド・ビューの所有者を入力します。マテリアライズド・ビュー・ログ・サイトで DBA_REGISTERED_MVIEWS ビューを問い合せて、マテリアライズド・ビューの所有者を表示します。
mviewname	mview_id を指定しない場合は、mviewname パラメータを使用してターゲット・マテリアライズド・ビューの名前を入力します。マテリアライズド・ビュー・ログ・サイトで DBA_REGISTERED_MVIEWS ビューを問い合せて、マテリアライズド・ビューの名前を表示します。
mviewsite	mview_id を指定しない場合は、mviewsite パラメータを使用してターゲット・マテリアライズド・ビューを入力します。マテリアライズド・ビュー・ログ・サイトで DBA_REGISTERED_MVIEWS ビューを問い合せて、マテリアライズド・ビュー・サイトを表示します。

REFRESH プロシージャ

このプロシージャは、マテリアライズド・ビューのリストをリフレッシュします。

構文

```
DBMS_MVIEW.REFRESH (
   { list                 IN     VARCHAR2,
   | tab                  IN OUT DBMS_UTILITY.UNCL_ARRAY,}
   method                 IN     VARCHAR2        := NULL,
   rollback_seg           IN     VARCHAR2        := NULL,
   push_deferred_rpc      IN     BOOLEAN         := true,
   refresh_after_errors   IN     BOOLEAN         := false,
   purge_option           IN     BINARY_INTEGER  := 1,
   parallelism            IN     BINARY_INTEGER  := 0,
   heap_size              IN     BINARY_INTEGER  := 0,
   atomic_refresh         IN     BOOLEAN         := true);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 33-9 REFRESH プロシージャのパラメータ

パラメータ	説明
list | tab	リフレッシュするマテリアライズド・ビューのカンマで区切られたリスト（シノニムはサポートされていません）。このようなマテリアライズド・ビューを異なるスキーマに配置し、異なるマスター表またはマスター・マテリアライズド・ビューを設定できます。ただし、リストされているすべてのマテリアライズド・ビューは、ローカル・データベースに存在している必要があります。 他の方法として、DBMS_UTILITY.UNCL_ARRAY タイプの PL/SQL 索引付き表を渡せます。この場合、各要素がマテリアライズド・ビューの名前です。
method	リストされているマテリアライズド・ビューのリフレッシュ方法を示す文字列。f は高速リフレッシュ、? は強制リフレッシュ、C または c は完全リフレッシュ、A または a は常にリフレッシュを示します。A および C は同等です。 マテリアライズド・ビューに対応するリフレッシュ方法がない場合（つまり、リフレッシュ方法より多くのマテリアライズド・ビューが指定された場合）、そのマテリアライズド・ビューはデフォルトのリフレッシュ方法に従ってリフレッシュされます。たとえば、SQL*Plus 内の次の EXECUTE 文を実行します。 `DBMS_MVIEW.REFRESH ('countries_mv,regions_mv,hr.employees_mv','cf');` この文は、countries_mv マテリアライズド・ビューの完全なリフレッシュ、regions_mv マテリアライズド・ビューの高速リフレッシュおよび hr.employees マテリアライズド・ビューのデフォルトのリフレッシュを実行します。
rollback_seg	マテリアライズド・ビューのリフレッシュ中に使用する、マテリアライズド・ビュー・サイトのロールバック・セグメント名。
push_deferred_rpc	更新可能なマテリアライズド・ビューでのみ使用します。マテリアライズド・ビューをリフレッシュする前に、マテリアライズド・ビューから関連するマスター表またはマスター・マテリアライズド・ビューに変更を送信する場合、このパラメータを TRUE に設定します。そうでない場合は、変更が一時的に失われたように表示される場合があります。
refresh_after_errors	このパラメータを TRUE に設定すると、マテリアライズド・ビューのマスター表またはマスター・マテリアライズド・ビューの DEFERROR ビューに未解決の競合が記録されていても、更新可能なマテリアライズド・ビューのリフレッシュは続行します。このパラメータが TRUE で、atomic_refresh が FALSE の場合、このプロシージャは、マテリアライズド・ビューのリフレッシュに失敗しても他のマテリアライズド・ビューのリフレッシュを続行します。

表 33-9 REFRESH プロシージャのパラメータ（続き）

パラメータ	説明
purge_option	パラレル伝播メカニズムを使用する場合（つまり、パラレル化に 1 以上を設定）は、次のように指定します。0 = パージなし、1 = レイジー・パージ、2 = aggressive パージ。ほとんどの場合、レイジー・パージが最適な設定です。複数のマスター・レプリケーション・グループが別々のターゲット・サイトに送信され、1 つ以上のレプリケーション・グループへの更新や送信がまれな場合は、aggressive パージに設定してキューを減らします。すべてのレプリケーション・グループへの更新と送信がまれな場合は、このパラメータを 0 に設定し、キューを減らすためにこのパラメータを時々 2 に設定して PUSH を実行してください。
parallelism	0（ゼロ）はシリアル伝播を指定します。 *n* > 1 は *n* のパラレル処理を使用するパラレル伝播を指定します。 1 は単一のパラレル処理を使用するパラレル伝播を指定します。
heap_size	パラレル伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。 **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。
atomic_refresh	このパラメータを TRUE に設定すると、マテリアライズド・ビューのリストは単一のトランザクションでリフレッシュされます。リフレッシュされたすべてのマテリアライズド・ビューは、単一のポイントへ同時に更新されます。マテリアライズド・ビューのいずれかでリフレッシュに失敗すると、すべてのマテリアライズド・ビューが更新されません。 このパラメータを FALSE に設定すると、マテリアライズド・ビューのリストは別のトランザクションでリフレッシュされます。このパラメータが FALSE の場合、ジョブ・キュー・プロセスの数は 1 以上に設定する必要があります。

REFRESH_ALL_MVIEWS プロシージャ

このプロシージャは、次のプロパティを持つすべてのマテリアライズド・ビューをリフレッシュします。

- マテリアライズド・ビューが依存するマスター表またはマスター・マテリアライズド・ビューに対する最も新しい変更が行われてから、マテリアライズド・ビューがリフレッシュされていない場合。
- マテリアライズド・ビューおよびそれが依存しているすべてのマスター表またはマスター・マテリアライズド・ビューがローカルな場合。
- マテリアライズド・ビューが DBA_MVIEWS ビューにある場合。

これは、データ・ウェアハウスで使用するためのプロシージャです。

構文

```
DBMS_MVIEW.REFRESH_ALL_MVIEWS (
   number_of_failures     OUT   BINARY_INTEGER,
   method                 IN    VARCHAR2         := NULL,
   rollback_seg           IN    VARCHAR2         := NULL,
   refresh_after_errors   IN    BOOLEAN          := false,
   atomic_refresh         IN    BOOLEAN          := true);
```

パラメータ

表 33-10 REFRESH_ALL_MVIEWS プロシージャのパラメータ

パラメータ	説明
number_of_failures	処理中に発生した失敗の件数を戻します。
method	リフレッシュされる各マテリアライズド・ビューに対して実行するリフレッシュのタイプを示す単一のリフレッシュ方法。F または f は高速リフレッシュ、? は強制リフレッシュ、C または c は完全リフレッシュ、A または a は常にリフレッシュを示します。A および C は同等です。方法が指定されていない場合、マテリアライズド・ビューはデフォルトのリフレッシュ方法に従ってリフレッシュされます。
rollback_seg	マテリアライズド・ビューのリフレッシュ中に使用する、マテリアライズド・ビュー・サイトのロールバック・セグメント名。

表 33-10 REFRESH_ALL_MVIEWS プロシージャのパラメータ（続き）

パラメータ	説明
refresh_after_errors	このパラメータを TRUE に設定すると、マテリアライズド・ビューのマスター表またはマスター・マテリアライズド・ビューの DEFERROR ビューに未解決の競合が記録されていても、更新可能なマテリアライズド・ビューのリフレッシュは続行します。このパラメータが TRUE で、atomic_refresh が FALSE の場合、このプロシージャは、マテリアライズド・ビューのリフレッシュに失敗しても他のマテリアライズド・ビューのリフレッシュを続行します。
atomic_refresh	このパラメータを TRUE に設定すると、リフレッシュされたマテリアライズド・ビューは単一のトランザクションでリフレッシュされます。リフレッシュされたすべてのマテリアライズド・ビューは、単一のポイントへ同時に更新されます。マテリアライズド・ビューのいずれかでリフレッシュに失敗すると、すべてのマテリアライズド・ビューが更新されません。 このパラメータを FALSE に設定すると、各マテリアライズド・ビューは別のトランザクションでリフレッシュされます。このパラメータが FALSE の場合、ジョブ・キュー・プロセスの数は 1 以上に設定する必要があります。

REFRESH_DEPENDENT プロシージャ

このプロシージャは、次のプロパティを持つすべてのマテリアライズド・ビューをリフレッシュします。

- マテリアライズド・ビューが、指定されたマスターのリストにあるマスター表またはマスター・マテリアライズド・ビューに依存している場合。
- マテリアライズド・ビューが依存するマスター表またはマスター・マテリアライズド・ビューに対する最も新しい変更が行われてから、マテリアライズド・ビューがリフレッシュされていない場合。
- マテリアライズド・ビューおよびそれが依存しているすべてのマスター表またはマスター・マテリアライズド・ビューがローカルな場合。
- マテリアライズド・ビューが DBA_MVIEWS ビューにある場合。

これは、データ・ウェアハウスで使用するためのプロシージャです。

構文

```
DBMS_MVIEW.REFRESH_DEPENDENT (
   number_of_failures     OUT    BINARY_INTEGER,
   { list                 IN     VARCHAR2,
   | tab                  IN OUT DBMS_UTILITY.UNCL_ARRAY,}
   method                 IN     VARCHAR2    := NULL,
   rollback_seg           IN     VARCHAR2    := NULL,
   refresh_after_errors   IN     BOOLEAN     := false,
   atomic_refresh         IN     BOOLEAN     := true);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 33-11 REFRESH_DEPENDENT プロシージャのパラメータ

パラメータ	説明
number_of_failures	処理中に発生した失敗の件数を戻します。
list | tab	マテリアライズド・ビューが依存できるマスター表またはマスター・マテリアライズド・ビューのカンマで区切られたリスト（シノニムはサポートされていません）。これらの表およびそれらに依存するマテリアライズド・ビューは、別々のスキーマに配置できます。ただし、すべての表とマテリアライズド・ビューは、ユーザーのローカル・データベースに存在している必要があります。 他の方法として、DBMS_UTILITY.UNCL_ARRAY タイプの PL/SQL 索引付き表を渡せます。この場合、各要素が表の名前です。

表 33-11 REFRESH_DEPENDENT プロシージャのパラメータ（続き）

パラメータ	説明
method	依存するマテリアライズド・ビューのリフレッシュ方法を示す文字列。特定の表に依存しているすべてのマテリアライズド・ビューは、その表に関連付けられたリフレッシュ方法に従ってリフレッシュされます。F または f は高速リフレッシュ、? は強制リフレッシュ、C または c は完全リフレッシュ、A または a は常にリフレッシュを示します。A および C は同等です。 表に対応するリフレッシュ方法がない場合（つまり、リフレッシュ方法より多くの表が指定された場合）、その表に依存するマテリアライズド・ビューはデフォルトのリフレッシュ方法に従ってリフレッシュされます。たとえば、SQL*Plus 内の次の EXECUTE 文を実行します。 `DBMS_MVIEW.REFRESH_DEPENDENT ('employees,deptartments,hr.regions','cf');` employees 表に依存するマテリアライズド・ビューの完全リフレッシュ、departments 表に依存するマテリアライズド・ビューの高速リフレッシュおよび hr.regions 表に依存するマテリアライズド・ビューのデフォルトのリフレッシュを実行します。
rollback_seg	マテリアライズド・ビューのリフレッシュ中に使用する、マテリアライズド・ビュー・サイトのロールバック・セグメント名。
refresh_after_errors	このパラメータを TRUE に設定すると、マテリアライズド・ビューのマスター表またはマスター・マテリアライズド・ビューの DEFERROR ビューに未解決の競合が記録されていても、更新可能なマテリアライズド・ビューのリフレッシュは続行します。このパラメータが TRUE で、atomic_refresh が FALSE の場合、このプロシージャは、マテリアライズド・ビューのリフレッシュに失敗しても他のマテリアライズド・ビューのリフレッシュを続行します。
atomic_refresh	このパラメータを TRUE に設定すると、リフレッシュされたマテリアライズド・ビューは単一のトランザクションでリフレッシュされます。リフレッシュされたすべてのマテリアライズド・ビューは、単一のポイントへ同時に更新されます。マテリアライズド・ビューのいずれかでリフレッシュに失敗すると、すべてのマテリアライズド・ビューが更新されません。 このパラメータを FALSE に設定すると、各マテリアライズド・ビューは別のトランザクションでリフレッシュされます。このパラメータが FALSE の場合、ジョブ・キュー・プロセスの数は 1 以上に設定する必要があります。

REGISTER_MVIEW プロシージャ

このプロシージャは、個々のマテリアライズド・ビューの管理を可能にします。マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで起動し、マテリアライズド・ビューを登録します。

> **注意：** 通常、マテリアライズド・ビューは、作成されている間に自動的に登録されます。自動登録に失敗した場合または登録情報が削除された場合は、このプロシージャを実行することでマテリアライズド・ビューは手動で登録されます。

構文

```
DBMS_MVIEW.REGISTER_MVIEW (
   mviewowner  IN   VARCHAR2,
   mviewname   IN   VARCHAR2,
   mviewsite   IN   VARCHAR2,
   mview_id    IN   DATE | BINARY_INTEGER,
   flag        IN   BINARY_INTEGER,
   qry_txt     IN   VARCHAR2,
   rep_type    IN   BINARY_INTEGER := DBMS_MVIEW.REG_UNKNOWN);
```

パラメータ

表 33-12 REGISTER_MVIEW プロシージャのパラメータ

パラメータ	説明
mviewowner	マテリアライズド・ビューの所有者。
mviewname	マテリアライズド・ビューの名前。
mviewsite	Oracle8 以上のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで登録するマテリアライズド・ビューのマテリアライズド・ビュー・サイトの名前。この名前に二重引用符を含めることはできません。
mview_id	マテリアライズド・ビューの識別番号。BINARY_INTEGER として Oracle8 以上のマテリアライズド・ビューを指定します。Oracle8 以上のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで、Oracle7 のマテリアライズド・ビューを DATE として指定します。

表 33-12 REGISTER_MVIEW プロシージャのパラメータ（続き）

パラメータ	説明
flag	登録されているマテリアライズド・ビューのプロパティを記述する定数。割当て可能な定数は次のとおりです。 ■ ROWID マテリアライズド・ビューの場合は dbms_mview.reg_rowid_mview。 ■ 主キー・マテリアライズド・ビューの場合は dbms_mview.reg_primary_key_mview。 ■ オブジェクト ID マテリアライズド・ビューの場合は dbms_mview.reg_object_id_mview。 ■ 高速リフレッシュが可能なマテリアライズド・ビューの場合は dbms_mview.reg_fast_refreshable_mview。 ■ 更新可能なマテリアライズド・ビューの場合は dbms_mview.reg_updatable_mview。 マテリアライズド・ビューにはこれらのプロパティのうち複数を設定できます。複数のプロパティを指定するには、プラス記号（+）を使用します。たとえば、主キーのマテリアライズド・ビューに高速リフレッシュを実行できる場合、このパラメータに次のように入力できます。 dbms_mview.reg_primary_key_mview + dbms_mview.reg_fast_refreshable_mview ALL_MVIEWS データ・ディクショナリ・ビューを問い合せると、マテリアライズド・ビューのプロパティを判別できます。
qry_txt	マテリアライズド・ビュー定義問合せの最初の 32,000 バイト。
rep_type	マテリアライズド・ビューのバージョン。割当て可能な定数は次のとおりです。 ■ マテリアライズド・ビューが Oracle7 のサイトにある場合、dbms_mview.reg_v7_snapshot。 ■ マテリアライズド・ビューが Oracle8 以上のサイトにある場合、dbms_mview.reg_v8_snapshot。 ■ マテリアライズド・ビューが Oracle7 のサイトまたは Oracle8 以上のサイトのどちらにあるか不明な場合、dbms_mview.reg_unknown（デフォルト）。

使用上の注意

このプロシージャは、リモート・プロシージャ・コールを使用して、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトでリモート・マテリアライズド・ビューにより起動します。REGISTER_MVIEW が同じ mviewowner、mviewname および mviewsite を使用して複数回コールされている場合、mview_id、flag および qry_txt に対する最も新しい値が格納されます。問合せが VARCHAR2 の最大サイズを超える場合は、最初の 32000 文字が query_txt に格納され、残りは切り捨てられます。手動で起動した場合、プロシージャをコールしたユーザーがマテリアライズド・ビューのデータ・ディクショナリ・ビューで mview_id の値を参照する必要があります。

UNREGISTER_MVIEW プロシージャ

このプロシージャは、個々のマテリアライズド・ビューの管理を可能にします。マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで起動し、マテリアライズド・ビューを登録解除します。

構文

```
DBMS_MVIEW.UNREGISTER_MVIEW (
   mviewowner      IN   VARCHAR2,
   mviewname       IN   VARCHAR2,
   mviewsite       IN   VARCHAR2);
```

パラメータ

表 33-13 UNREGISTER_MVIEW プロシージャのパラメータ

パラメータ	説明
mviewowner	マテリアライズド・ビューの所有者。
mviewname	マテリアライズド・ビューの名前。
mviewsite	マテリアライズド・ビューの名前。

34

DBMS_OBFUSCATION_TOOLKIT

DBMS_OBFUSCATION_TOOLKIT を使用すると、データ暗号化規格（DES）またはトリプル DES アルゴリズムを使用して、アプリケーションでデータを暗号化できます。

データ暗号化規格（DES）は、米国規格協会（ANSI）ではデータ暗号化アルゴリズム（DEA）、国際標準化機構（ISO）では DEA-1 とも呼ばれ、20 年以上にわたり世界的な暗号化規格の地位を保っています。銀行業界では、民間の金融機関どうしのトランザクションや、金融機関と個人とのトランザクションに DES ベースの規格も採用しています。DES は最終的に新しい Advanced Encryption Standard（AES）に置き換えられることになります。

DES は対称鍵暗号です。つまり、データの暗号化と復合化で同じ鍵が使用されます。データの暗号化は 64 ビットのブロックで、56 ビットのキーを使用して行われます。DES アルゴリズムでは提供される 64 ビット・キーの 8 ビットは無視されます。ただし、開発者は 64 ビット・キーを DES アルゴリズムに提供する必要があります。

トリプル DES（3DES）は DES よりもさらに強力な暗号です。生成された暗号文（暗号化データ）は、総当たり探索（2**56 の試行のかわりに 2**112 または 2**168 を試行）を使用しても、より一層破られにくくなっています。また、一定のタイプの暗号分析に対しても DES ほど弱くありません。DES プロシージャは次のとおりです。

- DESEncrypt プロシージャ
- DESDecrypt プロシージャ

このパッケージは SYS スキーマにインストールされます。必要に応じて、既存のユーザーおよびロールに、パッケージ・アクセス権限を付与できます。このパッケージでは PUBLIC ロールへのアクセスも付与されるため、明示的な付与は必要ありません。

この章では、次の項目について説明します。

- キー管理の概要
- DBMS_OBFUSCATION サブプログラムの要約

キー管理の概要

キー管理（暗号化キーの生成および同キーの安全な格納を含む）は、暗号化の重要な要素の 1 つです。キーの選択が適切でなかったり格納方法が誤っていると、悪意のあるユーザーによって暗号化が容易に破られます。暗号分析者は、総当たりキー探索（つまり、正しい復号化キーを見つけようとして、可能なすべてのキーを調べる）の攻撃を行うのではなく、通常はキーの選択やキーの格納方法の弱点を探します。

キー生成は暗号化の重要な要素です。通常、キーは乱数ジェネレータで自動的に生成されます。乱数生成が暗号面で安全であれば、キー生成の 1 つの形態として許容できます。ただし、乱数が暗号面で安全ではなく、しかも予測可能な要素があると、暗号化のセキュリティが容易に損なわれる可能性があります。

DBMS_OBFUSCATION_TOOLKIT パッケージでは、暗号化キーの生成もメンテナンスも行いません。アプリケーション開発者は、このパッケージで使用される暗号化キーの生成および格納が安全に行われるように注意する必要があります。また、DBMS_OBFUSCATION_TOOLKIT による暗号化および復号化はクライアントではなくサーバーで行われます。クライアントとサーバーの接続でキーを渡す場合、その接続は Oracle Advanced Security で保護する必要があります。保護しないと、回線で傍受されやすくなります。

キー格納は暗号化の最も重要な要素の 1 つですが、同時に難しく、正しく管理することが困難な要素です。対称鍵で暗号化したデータをリカバリするには、その対称鍵が、データを復号化しようとしているアプリケーションまたはユーザーによってアクセス可能である必要があります。対称鍵は、パフォーマンスを大幅に劣化させずに、ユーザーが必要なときに暗号化データにアクセスできるように、取出しが簡単なものにする必要があります。また、暗号化データにアクセスしようとする未許可のユーザーによって容易にリカバリされないように、十分に安全である必要があります。

開発者が選択できる 3 つのオプションを次に示します。

- キーをデータベースに格納する
- キーをオペレーティング・システムに格納する
- ユーザーによるキーの管理を可能にする

データベースへのキーの格納

DBA による暗号化データへのアクセスからデータを保護しようとする場合、キーをデータベースに格納しても、必ずしも完全なセキュリティを保つことはできません（すべての権限を付与された DBA は暗号化キーが含まれた表にアクセスできるためです）。ただし、一時的なスヌーパーやオペレーティング・システム上のデータベース・ファイルを悪用するユーザーに対しては効果があります。また、データベースにキーを格納して獲得するセキュリティは、完全でなくても有効です。

たとえば、従業員の社会保障番号（表 EMP の列の 1 つ）を暗号化するとします。その場合は、EMP の別の列に格納されているキーを使用して各従業員の社会保障番号を暗号化します。ただし、EMP 表で SELECT アクセスを持つユーザーは、暗号化キーを取得し一致する

社会保障番号を復号化できます。また、暗号化キーを別の表に格納し、パッケージを使用して、表どうしの主キーと外部キーの関係に基づき暗号化データ項目の正しいキーを取得する方法もあります。

開発者は、DBMS_OBFUSCATION_TOOLKIT パッケージと、そのパッケージに提供された暗号化キーを取り出すプロシージャの両方をエンベロープできます。また、暗号化キー自体をなんらかの方法（たとえば、EMP 表の外部キーで XOR するなど）で変換して、容易にリカバリできる形態で格納されないようにすることもできます。

PL/SQL のラップ・ユーティリティを使用して、暗号化を行う PL/ SQL パッケージ自体の内部にあるコードをわかりにくくすることをお薦めします。これにより、第三者が、キー、暗号化ルーチンのコールなどを処理する PL/SQL を見て暗号化を破ることができなくなります。つまり、ラップ・ユーティリティを使用して PL/SQL パッケージ自体をわかりにくくします。このスキームは、EMP への SELECT アクセスを持つユーザーが、暗号化されていない重要データを読み取ったり、DBA が暗号化キーを容易に取り出し、その暗号化キーで EMP 表のデータを復号化することが防止される点で安全です。暗号化キーを定期的に変更したり、キー格納アルゴリズムを改良する（たとえば、キー自体が暗号化されるようにするなど）とより安全になります。

オペレーティング・システムへのキーの格納

もう 1 つのオプションは、キーをオペレーティング・システム（つまり、フラット・ファイル）に格納することです。Oracle9*i* では、PL/SQL からのコールアウトが可能なため、これを使用して暗号化キーを取り出すことができます。オペレーティング・システムにキーを格納し、コールアウトを行ってそのキーを取り出す場合、暗号化データのセキュリティは、オペレーティング・システム上のキー・ファイルに対する保護と同程度にしかなりません。その場合は、オペレーティング・システムからキーを取り出すユーザーが（暗号化データを復号化するために）、Oracle データベース・ファイルにアクセスできるようにするか、暗号化データが正当なユーザーとして格納されている表へのアクセスを取得できるようにする必要があります。

ユーザー指定キー

ユーザーがキーを指定する場合は、Oracle Advanced Security などのネットワーク暗号化を使用して、そのキーがクライアントからサーバーにそのままの状態で渡されないようにすることが非常に重要です。ユーザーがキーを覚えていないと、データがリカバリ不能になります。

DBMS_OBFUSCATION サブプログラムの要約

表 34-1 DBMS_OBFUSCATION サブプログラム

サブプログラム	説明
「DESEncrypt プロシージャ」34-4 ページ	入力データの暗号化フォームを生成します。
「DESDecrypt プロシージャ」34-6 ページ	入力データの復号化フォームを生成します。
「DES3Encrypt プロシージャ」34-9 ページ	トリプル DES（3DES）暗号化アルゴリズムを介して渡すことにより、入力データの暗号化フォームを生成します。
「DES3Decrypt プロシージャ」34-11 ページ	入力データの復号化フォームを生成します。

DESEncrypt プロシージャ

DESEncrypt プロシージャは、入力データの暗号化フォームを生成します。DESEncrypt プロシージャの例は、この章の最後に記載されています。

DES アルゴリズムでは、56 ビットのキーを使用して暗号化が 64 ビットのブロックで行われます。また、提供されたキーの 8 ビットが送信されます（具体的な送信ビットについては、このマニュアルの対象外です）。ただし、このアルゴリズムを使用する開発者は、64 ビットのキーを提供しないと、エラーが発生します。

パラメータ

表 34-2 ロー・データ用の DESEncrypt パラメータ

パラメータ名	モード	タイプ	説明
input	IN	RAW	暗号化するデータ
key	IN	RAW	暗号キー
encrypted_data	OUT	RAW	暗号化されたデータ

表 34-3 文字列データ用の DESEncrypt パラメータ

パラメータ名	モード	タイプ	説明
input_string	IN	VARCHAR2	暗号化する文字列
key_string	IN	VARCHAR2	暗号キー文字列
encrypted_string	OUT	VARCHAR2	暗号化された文字列

PL/SQL DESEncrypt プロシージャに渡された入力データまたはキーに値が指定されていない場合は、エラー ORA-28231「dbms_obfuscation_toolkit にデータが渡されていません。」が発生します。

DESEncrypt プロシージャに渡された入力データが 8 バイトの倍数でない場合は、エラー ORA-28232「dbms_obfuscation_toolkit にデータが渡されていません。」が発生します。

ユーザーが DESEncrypt プロシージャを使用して、データを二重暗号化しようとすると、エラー ORA-28233「二重暗号化はサポートされません。」が発生します。

キーの長さが欠落していたり、7 バイト以下である場合は、エラー ORA-28234「キーの長さが短かすぎます。」が発生します。より長いキーを使用した場合、余分なバイトは無視されます。したがって、9 バイトのキーでは例外は発生しません。

制限事項

DESEncryption プロシージャには 2 つの制限事項があります。1 つは、暗号化用の DES キーの長さが 56 ビットに固定されていることです。このキーの長さは変更できません。

もう 1 つは、暗号化の多重パスを実行できないことです。つまり、ファンクションを 2 回コールして、すでに暗号化されているデータを再度暗号化することはできません。

注意： キーの長さの制限および多重暗号パスの防止は、暗号化製品の輸出を統括している米国の条項における要件です。

DESDecrypt プロシージャ

DESDecrypt プロシージャの目的は、入力データの復号化フォームを生成することにあります。DESDecrypt プロシージャの例は、この章の最後に記載されています。

パラメータ

表 34-4 ロー・データ用の DESDecrypt パラメータ

パラメータ名	モード	タイプ	説明
input	IN	RAW	復号化するデータ
key	IN	RAW	復号化キー
decrypted_data	OUT	RAW	復号化されたデータ

表 34-5 文字列データ用の DESDecrypt パラメータ

パラメータ名	モード	タイプ	説明
input_string	IN	VARCHAR2	復号化する文字列
key_string	IN	VARCHAR2	復号化キー文字列
decrypted_string	OUT	VARCHAR2	復号化された文字列

PL/SQL DESDecrypt ファンクションに渡された入力データまたはキーに値が指定されていない場合は、ORA エラー 28231「dbms_obfuscation_toolkit にデータが渡されていません。」が発生します。

DESDecrypt ファンクションに渡された入力データが 8 バイトの倍数でない場合は、ORA エラー 28232「dbms_obfuscation_toolkit にデータが渡されていません。」が発生します。

キーの長さが欠落していたり、7 バイト以下である場合は、エラー ORA-28234「キーの長さが短かすぎます。」が発生します。より長いキーを使用した場合、余分なバイトは無視されます。したがって、9 バイトのキーでは例外は発生しません。

注意： ORA-28233 は、DESDecrypt ファンクションには適用されません。

制限事項

暗号化用の DES キーの長さは 64 ビット（この内 56 ビットが使用される）に固定されています。このキーの長さは変更できません。

> **注意：** キーの長さの制限事項は、暗号化製品の輸出を統括している米国の条項における要件です。

例

次にサンプル PL/SQL プログラムを示します。コードのセグメントに番号を付け、そのコードの部分を説明するテキストを挿入しています。

```
DECLARE
   input_string        VARCHAR2(16) := 'tigertigertigert';
   raw_input           RAW(128) := UTL_RAW.CAST_TO_RAW(input_string);
   key_string          VARCHAR2(8)  := 'scottsco';
   raw_key             RAW(128) := UTL_RAW.CAST_TO_RAW(key_string);
   encrypted_raw               RAW(2048);
   encrypted_string            VARCHAR2(2048);
   decrypted_raw               RAW(2048);
   decrypted_string            VARCHAR2(2048);
   error_in_input_buffer_length EXCEPTION;
   PRAGMA EXCEPTION_INIT(error_in_input_buffer_length, -28232);
   INPUT_BUFFER_LENGTH_ERR_MSG VARCHAR2(100) :=
    '*** DES INPUT BUFFER NOT A MULTIPLE OF 8 BYTES - IGNORING
EXCEPTION ***';
   double_encrypt_not_permitted EXCEPTION;
   PRAGMA EXCEPTION_INIT(double_encrypt_not_permitted, -28233);
   DOUBLE_ENCRYPTION_ERR_MSG VARCHAR2(100) :=
    '*** CANNOT DOUBLE ENCRYPT DATA - IGNORING EXCEPTION ***';

-- 1. Begin testing raw data encryption and decryption
   BEGIN
   dbms_output.put_line('> ========= BEGIN TEST RAW DATA =========');
   dbms_output.put_line('> Raw input                          : ' ||
                 UTL_RAW.CAST_TO_VARCHAR2(raw_input));
   BEGIN
      dbms_obfuscation_toolkit.DESEncrypt(input => raw_input,
               key => raw_key, encrypted_data => encrypted_raw );
      dbms_output.put_line('> encrypted hex value              : ' ||
               rawtohex(encrypted_raw));
      dbms_obfuscation_toolkit.DESDecrypt(input => encrypted_raw,
               key => raw_key, decrypted_data => decrypted_raw);
      dbms_output.put_line('> Decrypted raw output             : ' ||
                  UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw));
```

```
      dbms_output.put_line('>  ');
      if UTL_RAW.CAST_TO_VARCHAR2(raw_input) =
                   UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw) THEN
         dbms_output.put_line('> Raw DES Encyption and Decryption successful');
      END if;
   EXCEPTION
      WHEN error_in_input_buffer_length THEN
            dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
   END;
   dbms_output.put_line('>  ');


-- 2. Begin testing string data encryption and decryption
   dbms_output.put_line('> ========= BEGIN TEST STRING DATA =========');

   BEGIN
      dbms_output.put_line('> input string                    : '
                           || input_string);
      dbms_obfuscation_toolkit.DESEncrypt(
               input_string => input_string,
               key_string => key_string,
               encrypted_string => encrypted_string );
      dbms_output.put_line('> encrypted hex value             : ' ||
                  rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_string)));
      dbms_obfuscation_toolkit.DESDecrypt(
               input_string => encrypted_string,
               key_string => key_string,
               decrypted_string => decrypted_string );
      dbms_output.put_line('> decrypted string output         : ' ||
                decrypted_string);
      if input_string = decrypted_string THEN
         dbms_output.put_line('> String DES Encyption and Decryption successful');
      END if;
   EXCEPTION
      WHEN error_in_input_buffer_length THEN
            dbms_output.put_line(' ' || INPUT_BUFFER_LENGTH_ERR_MSG);
   END;
   dbms_output.put_line('>  ');
END;
```

DES3Encrypt プロシージャ

DES3Encrypt プロシージャは、トリプル DES（3DES）暗号化アルゴリズムを使用して渡すことにより、入力データの暗号化フォームを生成します。DESEncrypt プロシージャの例は、この章の最後に記載されています。

Oracle の実装 3DES では、外部暗号ブロック連鎖（CBC）モードでの、2 キーまたは 3 キーの実装をサポートしています。

2 キーを実装した Oracle の 3DES インタフェースを使用する開発者は、DES3Encrypt プロシージャに対する引数に、128 ビットの単一キーを指定する必要があります。3 キーを実装した場合は、192 ビットの単一キーを提供する必要があります。提供されたキーは、2 つの 64 ビットのキーに分割されます。DES の場合と同様に、3DES アルゴリズムでは、各導出キーの 8 ビットが送信されます。ただし、2 キーの 3DES を実装した場合は単一の 128 ビットのキーを、3 キーの 3DES を実装した場合は単一の 192 ビットのキーを提供しないとエラーが発生します。DES3Encrypt プロシージャでは、デフォルトで 2 キーの実装が使用されます。

パラメータ

表 34-6 ロー・データ用の DES3Encrypt パラメータ

パラメータ名	モード	タイプ	説明
input	IN	RAW	暗号化するデータ
key	IN	RAW	暗号キー
encrypted_data	OUT	RAW	暗号化されたデータ
which	IN	PLS_INTEGER	0（デフォルト）の場合は TwoKeyMode が使用されます。1 の場合は ThreeKeyMode が使用されます。

表 34-7 文字列データ用の DES3Encrypt パラメータ

パラメータ名	モード	タイプ	説明
input_string	IN	VARCHAR2	暗号化する文字列
key_string	IN	VARCHAR2	暗号キー文字列
encrypted_string	OUT	VARCHAR2	暗号化された文字列
which	IN	PLS_INTEGER	0（デフォルト）の場合は TwoKeyMode が使用されます。1 の場合は ThreeKeyMode が使用されます。

PL/SQL DES3Encrypt プロシージャに渡された入力データまたはキーに値が指定されていない場合は、エラー ORA-28231「dbms_obfuscation_toolkit にデータが渡されていません。」が発生します。

DES3Encrypt プロシージャに渡された入力データが 8 バイトの倍数でない場合は、エラー ORA-28232「dbms_obfuscation_toolkit にデータが渡されていません。」が発生します。

ユーザーが DES3Encrypt プロシージャを使用して、データを二重暗号化しようとすると、エラー ORA-28233「二重暗号化はサポートされません。」が発生します。

キーの長さが欠落していたり、7 バイト以下である場合は、エラー ORA-28234「キーの長さが短かすぎます。」が発生します。より長いキーを使用した場合、余分なバイトは無視されます。したがって、9 バイトのキーでは例外は発生しません。

WHICH パラメータに不適切な値を指定すると、ORA-28236「三重 DES モードが無効です。」が生成されます。有効な値は 0（TwoKeyMode）および 1（ThreeKeyMode）です。

制限事項

DES3Encryption プロシージャには 2 つの制限事項があります。1 つは、暗号化用の DES キーの長さが 128 ビット（2 キー DES）または 192 ビット（3 キー DES）に固定されていることです。これらのキーの長さは変更できません。

もう 1 つは、3DES を使用して暗号化の多重パスを実行できないことです。（**注意 :** 3DES アルゴリズム自体はデータを複数回暗号化します。ただし、3DESencrypt ファンクション自体を複数回コールして、3DES で同じデータを暗号化することはできません。）

注意 : キーの長さの制限および多重暗号パスの防止は、暗号化製品の輸出を統括している米国の条項における要件です。

DES3Decrypt プロシージャ

DES3Decrypt プロシージャの目的は、入力データの復号化フォームを生成することにあります。DES3Decrypt プロシージャの例は、この章の最後に記載されています。

パラメータ

表 34-8 ロー・データ用の DES3Decrypt パラメータ

パラメータ名	モード	タイプ	説明
input	IN	RAW	復号化するデータ
key	IN	RAW	復号化キー
decrypted_data	OUT	RAW	復号化されたデータ
which	IN	PLS_INTEGER	0（デフォルト）の場合は TwoKeyMode が使用されます。1 の場合は ThreeKeyMode が使用されます。

表 34-9 文字列データ用の DES3Decrypt パラメータ

パラメータ名	モード	タイプ	説明
input_string	IN	VARCHAR2	復号化する文字列
key_string	IN	VARCHAR2	復号化キー文字列
decrypted_string	OUT	VARCHAR2	復号化された文字列
which	IN	PLS_INTEGER	0（デフォルト）の場合は TwoKeyMode が使用されます。1 の場合は ThreeKeyMode が使用されます。

DES3Decrypt プロシージャに渡された入力データまたはキーに値が指定されていない場合は、エラー ORA-28231「dbms_obfuscation_toolkit にデータが渡されていません。」が発生します。

DES3Decrypt プロシージャに渡された入力データが 8 バイトの倍数でない場合は、エラー ORA-28232「dbms_obfuscation_toolkit にデータが渡されていません。」が発生します。ORA-28233 は、DES3Decrypt ファンクションには適用されません。

キーの長さが欠落していたり、7 バイト以下である場合は、エラー ORA-28234「キーの長さが短かすぎます。」が発生します。より長いキーを使用した場合、余分なバイトは無視されます。したがって、9 バイトのキーでは例外は発生しません。

WHICH パラメータに不適切な値を指定すると、ORA-28236「三重 DES モードが無効です。」が生成されます。有効な値は 0（TwoKeyMode）および 1（ThreeKeyMode）です。

制限事項

開発者は、2 キー実装の場合は 128 ビットの単一キー（使用されるのは 112 ビットのみ）、3 キー実装の場合は 192 ビット（使用されるのは 168 ビット）の単一キーを指定する必要があります。指定したキーは、自動的に復号化用に 56 ビットの長さに切り捨てられます。このキーの長さは固定されていて変更できません。

注意： キーの長さの制限および多重暗号パスの防止は、暗号化製品の輸出を統括している米国の条項における要件です。

例

次のコードは、参照用のサンプル PL/SQL プログラムです。コードのセグメントに番号を付け、そのコードの部分を説明するテキストを挿入しています。

```
DECLARE
   input_string        VARCHAR2(16) := 'tigertigertigert';
   raw_input           RAW(128) := UTL_RAW.CAST_TO_RAW(input_string);
   key_string          VARCHAR2(16)  := 'scottscottscotts';
   raw_key             RAW(128) := UTL_RAW.CAST_TO_RAW(key_string);
encrypted_raw               RAW(2048);
   encrypted_string            VARCHAR2(2048);
decrypted_raw               RAW(2048);
   decrypted_string            VARCHAR2(2048);
   error_in_input_buffer_length EXCEPTION;
   PRAGMA EXCEPTION_INIT(error_in_input_buffer_length, -28232);
   INPUT_BUFFER_LENGTH_ERR_MSG VARCHAR2(100) :=
    '*** DES INPUT BUFFER NOT A MULTIPLE OF 8 BYTES - IGNORING EXCEPTION ***';
   double_encrypt_not_permitted EXCEPTION;
   PRAGMA EXCEPTION_INIT(double_encrypt_not_permitted, -28233);
   DOUBLE_ENCRYPTION_ERR_MSG VARCHAR2(100) :=
    '*** CANNOT DOUBLE ENCRYPT DATA - IGNORING EXCEPTION ***';

-- 1. Begin testing raw data encryption and decryption
   BEGIN
   dbms_output.put_line('> ========= BEGIN TEST RAW DATA =========');
   dbms_output.put_line('> Raw input                          : ' ||
                UTL_RAW.CAST_TO_VARCHAR2(raw_input));
   BEGIN
      dbms_obfuscation_toolkit.DES3Encrypt(input => raw_input,
               key => raw_key, encrypted_data => encrypted_raw );
```

```
      dbms_output.put_line('> encrypted hex value              : ' ||
               rawtohex(encrypted_raw));
      dbms_obfuscation_toolkit.DES3Decrypt(input => encrypted_raw,
               key => raw_key, decrypted_data => decrypted_raw);
      dbms_output.put_line('> Decrypted raw output             : ' ||
                  UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw));
      dbms_output.put_line('>  ');
      if UTL_RAW.CAST_TO_VARCHAR2(raw_input) =
                  UTL_RAW.CAST_TO_VARCHAR2(decrypted_raw) THEN
         dbms_output.put_line('> Raw DES3 Encyption and Decryption successful');
      END if;
   EXCEPTION
      WHEN error_in_input_buffer_length THEN
            dbms_output.put_line('> ' || INPUT_BUFFER_LENGTH_ERR_MSG);
   END;
   dbms_output.put_line('>  ');
END;

-- 2. Begin testing string data encryption and decryption
   dbms_output.put_line('> ========= BEGIN TEST STRING DATA =========');

   BEGIN
      dbms_output.put_line('> input string                     : '
                           || input_string);
      dbms_obfuscation_toolkit.DES3Encrypt(
               input_string => input_string,
               key_string => key_string,
               encrypted_string => encrypted_string );
      dbms_output.put_line('> encrypted hex value              : ' ||
                  rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_string)));
      dbms_obfuscation_toolkit.DES3Decrypt(
               input_string => encrypted_string,
               key_string => key_string,
               decrypted_string => decrypted_string );
      dbms_output.put_line('> decrypted string output          : ' ||
                 decrypted_string);
      if input_string = decrypted_string THEN
         dbms_output.put_line('> String DES3 Encyption and Decryption successful');
      END if;
   EXCEPTION
      WHEN error_in_input_buffer_length THEN
            dbms_output.put_line(' ' || INPUT_BUFFER_LENGTH_ERR_MSG);
   END;
   dbms_output.put_line('>  ');
END;
```

35

DBMS_ODCI

DBMS_ODCI は、機能の経過時間に基づき、ユーザー・ファンクションの CPU コストを戻します。CPU コストは拡張可能オプティマイザ・ルーチンで使用されます。

この章では、次の項目について説明します。

DBMS_ODCI サブプログラムの要約

表 35-1 DBMS_ODCI サブプログラム

サブプログラム	説明
「ESTIMATE_CPU_UNITS ファンクション」 35-2 ページ	指定の期間（秒）に対応する CPU 命令の概数（単位 : 千）を戻します。

ESTIMATE_CPU_UNITS ファンクション

ESTIMATE_CPU_UNITS は、指定の期間（秒）に対応する CPU 命令の概数（単位 : 千）を戻します。この情報を使用すると、拡張可能オプティマイザのユーザー定義ファンクションと CPU コストとを関連付けることができます。

このファンクションは、ユーザー・ファンクションの経過時間を入力として使用し、その経過時間とマシンのプロセッサ速度を乗算して CPU 単位を測定したのち、ユーザー・ファンクションに関連付ける CPU の概数を戻します（マルチプロセッサの場合、ESTIMATE_CPU_UNITS ではプロセッサ 1 つ当たりの速度で計算されます）。

構文

```
DBMS_ODCI.ESTIMATE_CPU_UNITS (
   elapsed_time NUMBER)
RETURN NUMBER;
```

パラメータ

表 35-2 ESTIMATE_CPU_UNITS ファンクションのパラメータ

パラメータ	説明
elapsed_time	ファンクション実行の秒単位の経過時間

使用上の注意

CPU コストをユーザー定義ファンクションと関連付ける場合は、ESTIMATE_CPU_UNITS から戻される千単位の CPU 単位ではなく、完全な数の CPU 単位を使用します。つまり、ESTIMATE_CPU_UNITS から戻された数に 1,000 を乗算します。

例

あるマシン上で 10 秒を要するファンクションに使用される CPU 単位の数を判別するには次のようにします。

```
DECLARE
  a INTEGER;
BEGIN
  a := DBMS_ODCI.ESTIMATE_CPU_UNITS(10);
  DBMS_OUTPUT.PUT_LINE('CPU units = '|| a*1000);
END;
```

36

DBMS_OFFLINE_OG

DBMS_OFFLINE_OG パッケージには、マスター・グループのオフライン・インスタンス化のためのパブリック API が含まれています。

この章では、次の項目について説明します。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンス化を実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（マテリアライズド・ビューのオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

DBMS_OFFLINE_OG サブプログラムの要約

表 36-1 DBMS_OFFLINE_OG パッケージのサブプログラム

サブプログラム	説明
「BEGIN_INSTANTIATION プロシージャ」36-2 ページ	マスター・グループのオフライン・インスタンス化を開始します。
「BEGIN_LOAD プロシージャ」36-4 ページ	オフライン・インスタンス化の一部としてデータを新規マスター・サイトにインポートする間、トリガーを使用禁止にします。
「END_INSTANTIATION プロシージャ」36-5 ページ	マスター・グループのオフライン・インスタンス化を完了します。
「END_LOAD プロシージャ」36-7 ページ	オフライン・インスタンス化の一部としてデータを新規マスター・サイトにインポートした後、トリガーを再び使用可能にします。
「RESUME_SUBSET_OF_MASTERS プロシージャ」36-8 ページ	マスター・グループのオフライン・インスタンス化中に、新規サイトを除く既存のすべてのサイトでレプリケーション・アクティビティを再開します。

BEGIN_INSTANTIATION プロシージャ

このプロシージャは、マスター・グループのオフライン・インスタンス化を開始します。このプロシージャは、マスター定義サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンス化を実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（マテリアライズド・ビューのオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.BEGIN_INSTANTIATION (
   gname     IN   VARCHAR2,
   new_site  IN   VARCHAR2
   fname     IN   VARCHAR2);
```

パラメータ

表 36-2 BEGIN_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
gname	新規サイトにレプリケートするレプリケーション・グループの名前
new_site	レプリケーション・グループをレプリケートする新規サイトの完全修飾データベース名
fname	内部使用のためのパラメータ **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

例外

表 36-3 BEGIN_INSTANTIATION プロシージャの例外

例外	説明
badargument	レプリケーション・グループまたは新規マスター・サイト名が NULL または指定されていません。
dbms_repcat.nonmasterdef	このプロシージャは、マスター定義サイトからコールする必要があります。
sitealreadyexists	指定のサイトは、すでにこのレプリケーション・グループのマスター・サイトです。
wrongstate	マスター定義サイトのステータスは、停止中である必要があります。
dbms_repcat.missingrepgroup	gname がマスター・グループとして存在しません。
dbms_repcat.missing_flavor	この例外が発生した場合は、オラクル社カスタマ・サポート・センターに連絡してください。

BEGIN_LOAD プロシージャ

このプロシージャは、オフライン・インスタンス化の一部としてデータを新規マスター・サイトにインポートする間、トリガーを使用禁止にします。このプロシージャは、新規マスター・サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンス化を実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（マテリアライズド・ビューのオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.BEGIN_LOAD (
   gname     IN   VARCHAR2,
   new_site  IN   VARCHAR2);
```

パラメータ

表 36-4 BEGIN_LOAD プロシージャのパラメータ

パラメータ	説明
gname	インポートするメンバーのレプリケーション・グループ名。
new_site	レプリケーション・グループのメンバーをインポートする新規サイトの完全修飾データベース名。

例外

表 36-5 BEGIN_LOAD プロシージャの例外

例外	説明
badargument	レプリケーション・グループまたは新規マスター・サイト名が NULL または指定されていません。
wrongsite	このプロシージャは、新規マスター・サイトからコールする必要があります。
unknownsite	指定のサイトがレプリケーション・グループで認識されません。
wrongstate	新規マスター・サイトのステータスは、停止中である必要があります。
dbms_repcat.missingrepgroup	gname がマスター・グループとして存在しません。

END_INSTANTIATION プロシージャ

このプロシージャは、マスター・グループのオフライン・インスタンス化を完了します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **注意：** このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンス化を実行するときに使用します。
>
> DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（マテリアライズド・ビューのオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.END_INSTANTIATION (
  gname     IN  VARCHAR2,
  new_site  IN  VARCHAR2);
```

パラメータ

表 36-6 END_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
gname	新規サイトにレプリケートするレプリケーション・グループの名前。
new_site	レプリケーション・グループをレプリケートする新規サイトの完全修飾データベース名。

例外

表 36-7 END_INSTANTIATION プロシージャの例外

例外	説明
badargument	レプリケーション・グループまたは新規マスター・サイト名が NULL または指定されていません。
dbms_repcat.nonmasterdef	このプロシージャは、マスター定義サイトからコールする必要があります。
unknownsite	指定のサイトがレプリケーション・グループで認識されません。
wrongstate	マスター定義サイトのステータスは、停止中である必要があります。
dbms_repcat.missingrepgroup	gname がマスター・グループとして存在しません。

END_LOAD プロシージャ

このプロシージャは、オフライン・インスタンス化の一部としてデータを新規マスター・サイトにインポートした後、トリガーを再び使用可能にします。このプロシージャは、新規マスター・サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンス化を実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（マテリアライズド・ビューのオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.END_LOAD (
   gname     IN   VARCHAR2,
   new_site  IN   VARCHAR2
   fname     IN   VARCHAR2);
```

パラメータ

表 36-8 END_LOAD プロシージャのパラメータ

パラメータ	説明
gname	インポートを終了したメンバーのレプリケーション・グループの名前。
new_site	レプリケーション・グループのメンバーをインポートした新規サイトの完全修飾データベース名。
fname	内部使用のためのパラメータ。 **注意：**オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

例外

表 36-9 END_LOAD プロシージャの例外

例外	説明
badargument	レプリケーション・グループまたは新規マスター・サイト名が NULL または指定されていません。
wrongsite	このプロシージャは、新規マスター・サイトからコールする必要があります。
unknownsite	指定のサイトがレプリケーション・グループで認識されません。
wrongstate	新規マスター・サイトのステータスは、停止中である必要があります。
dbms_repcat.missingrepgroup	gname がマスター・グループとして存在しません。
dbms_repcat.flavor_noobject	この例外が発生した場合は、オラクル社カスタマ・サポート・センターに連絡してください。
dbms_repcat.flavor_contains	この例外が発生した場合は、オラクル社カスタマ・サポート・センターに連絡してください。

RESUME_SUBSET_OF_MASTERS プロシージャ

マスター・サイトのオフライン・インスタンス化を実行して新規マスター・サイトをマスター・グループに追加する場合、オフライン・インスタンス化プロセスの完了までに時間がかかることがあります。このプロシージャは、マスター・グループのオフライン・インスタンス化中に、新規サイトを除く既存のすべてのサイトでレプリケーション・アクティビティを再開します。通常このプロシージャは、DBMS_OFFLINE_OG.BEGIN_INSTANTIATION の後に実行します。このプロシージャは、マスター定義サイトからコールする必要があります。

注意： このプロシージャは、マルチマスター・レプリケーション環境でマスター表のオフライン・インスタンス化を実行するときに使用します。

DBMS_OFFLINE_SNAPSHOT パッケージにあるプロシージャ（マテリアライズド・ビューのオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_OG.RESUME_SUBSET_OF_MASTERS (
   gname     IN  VARCHAR2,
   new_site  IN  VARCHAR2
   override  IN  BOOLEAN := false);
```

パラメータ

表 36-10 RESUME_SUBSET_OF_MASTERS プロシージャのパラメータ

パラメータ	説明
gname	新規サイトにレプリケートするレプリケーション・グループの名前。
new_site	レプリケーション・グループをレプリケートする新規サイトの完全修飾データベース名。
override	TRUE の場合、保留中の RepCat 管理要求は無視され、通常のレプリケーション・アクティビティが各マスターで可能なかぎり早くリストアされます。override パラメータは、緊急の状況でのみ TRUE に指定してください。 このパラメータが FALSE の場合は、各マスターの gname に対する保留中の RepCat 管理要求がない場合のみ、各マスターで通常のレプリケーション・アクティビティがリストアされます。

例外

表 36-11 RESUME_SUBSET_OF_MASTERS プロシージャの例外

例外	説明
badargument	レプリケーション・グループまたは新規マスター・サイト名が NULL または指定されていません。
dbms_repcat.nonmasterdef	このプロシージャは、マスター定義サイトからコールする必要があります。
unknownsite	指定のサイトがレプリケーション・グループで認識されません。
wrongstate	マスター定義サイトのステータスは、停止中である必要があります。
dbms_ repcat.missingrepgroup	gname がマスター・グループとして存在しません。

37

DBMS_OFFLINE_SNAPSHOT

DBMS_OFFLINE_SNAPSHOT パッケージには、マテリアライズド・ビューのオフライン・インスタンス化のためのパブリック API が含まれています。

この章では、次の項目について説明します。

注意： このプロシージャは、マテリアライズド・ビューのオフライン・インスタンス化の実行時に使用されます。

DBMS_OFFLINE_OG パッケージにあるプロシージャ（マスター表のオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

DBMS_OFFLINE_SNAPSHOT サブプログラムの要約

表 37-1 DBMS_OFFLINE_SNAPSHOT パッケージのサブプログラム

サブプログラム	説明
「BEGIN_LOAD プロシージャ」37-2 ページ	オフライン・インスタンス化の一部として新規マテリアライズド・ビューをインポートするために、マテリアライズド・ビュー・サイトを準備します。
「END_LOAD プロシージャ」37-4 ページ	マテリアライズド・ビューのオフライン・インスタンス化を完了します。

BEGIN_LOAD プロシージャ

このプロシージャは、オフライン・インスタンス化の一部として新規マテリアライズド・ビューをインポートするために、マテリアライズド・ビュー・サイトを準備します。このプロシージャは、新規マテリアライズド・ビューのマテリアライズド・ビュー・サイトからコールする必要があります。

注意： このプロシージャは、マテリアライズド・ビューのオフライン・インスタンス化の実行時に使用されます。

DBMS_OFFLINE_OG パッケージにあるプロシージャ（マスター表のオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_SNAPSHOT.BEGIN_LOAD (
   gname                IN   VARCHAR2,
   sname                IN   VARCHAR2,
   master_site          IN   VARCHAR2,
   snapshot_oname       IN   VARCHAR2,
   storage_c            IN   VARCHAR2 := '',
   comment              IN   VARCHAR2 := '',
   min_communication    IN   BOOLEAN  := true);
```

パラメータ

表 37-2 BEGIN_LOAD プロシージャのパラメータ

パラメータ	説明
gname	オフライン・インスタンス化を使用して作成するマテリアライズド・ビューのレプリケーション・グループ名。
sname	新規マテリアライズド・ビューに対するスキーマの名前。
master_site	マテリアライズド・ビューのマスター・サイトの完全修飾データベース名。
snapshot_oname	マスター・サイトで作成される一時マテリアライズド・ビュー名。
storage_c	マテリアライズド・ビュー・サイトで新規マテリアライズド・ビューを作成するときに使用する記憶域オプション。
comment	ユーザー・コメント。
min_communication	TRUE の場合は、更新文で列を変更する場合にかぎり、更新トリガーによって列の新しい値が送信されます。また、その列がキー列または変更された列グループ内の列の場合、更新トリガーはその列の元の値も送信します。

例外

表 37-3 BEGIN_LOAD プロシージャの例外

例外	説明
badargument	レプリケーション・グループ、スキーマ、マスター・サイトまたはマテリアライズド・ビュー名が NULL または指定されていません。
dbms_repcat.missingrepgroup	gname がレプリケーション・グループとして存在しません。
missingremotemview	指定のマテリアライズド・ビューが指定のマスター・サイトで見つかりません。
dbms_repcat.missingschema	指定したスキーマが存在しません。
mviewtabmismatch	マスターにあるマテリアライズド・ビューのベース表名とマテリアライズド・ビューが一致しません。

END_LOAD プロシージャ

このプロシージャは、マテリアライズド・ビューのオフライン・インスタンス化を完了します。このプロシージャは、新規マテリアライズド・ビューのマテリアライズド・ビュー・サイトからコールする必要があります。

注意： このプロシージャは、マテリアライズド・ビューのオフライン・インスタンス化の実行時に使用されます。

DBMS_OFFLINE_OG パッケージにあるプロシージャ（マスター表のオフライン・インスタンス化の実行に使用）または DBMS_REPCAT_INSTANTIATE パッケージにあるプロシージャ（配置テンプレートのインスタンス化に使用）とこのプロシージャを混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_OFFLINE_SNAPSHOT.END_LOAD (
   gname           IN  VARCHAR2,
   sname           IN  VARCHAR2,
   snapshot_oname  IN  VARCHAR2);
```

パラメータ

表 37-4 END_LOAD プロシージャのパラメータ

パラメータ	説明
gname	オフライン・インスタンス化を使用して作成するマテリアライズド・ビューのレプリケーション・グループ名。
sname	新規マテリアライズド・ビューに対するスキーマの名前。
snapshot_oname	マテリアライズド・ビューの名前。

例外

表 37-5 END_LOAD プロシージャの例外

例外	説明
badargument	レプリケーション・グループ、スキーマまたはマテリアライズド・ビュー名が NULL または指定されていません。
dbms_repcat.missingrepgroup	gname がレプリケーション・グループとして存在しません。
dbms_repcat.nonmview	このプロシージャは、マテリアライズド・ビュー・サイトからコールする必要があります。

38

DBMS_OLAP

DBMS_OLAP パッケージは、マテリアライズド・ビュー分析の収集および任意の PL/SQL プログラムからコールできるアドバイザ機能を提供します。一部のファンクションは出力表を生成します。

> **関連項目：** DBMS_OLAP およびその出力表の使用方法の詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

この章では、次の項目について説明します。

要件

DBMS_OLAP は、7 つの主要機能、つまりマテリアライズド・ビューの計画の推奨、マテリアライズド・ビューの計画の評価、レポートおよびスクリプトの生成、リポジトリ管理、ワークロード管理、フィルタ管理およびディメンションの妥当性チェックを実行します。

マテリアライズド・ビューの計画の推奨および評価のファンクションを実行する場合、ワークロード情報はユーザーから提供することもアドバイザ・エンジンで合成することもできます。ワークロード情報をユーザーから提供する場合は、そのワークロードで参照するすべての表およびマテリアライズド・ビューのカーディナリティ情報が必要です。アドバイザ・エンジンで合成する場合は、ディメンション・オブジェクトの他、すべてのディメンション表、ファクト表およびマテリアライズド・ビューのカーディナリティ情報が必要です。カーディナリティ情報の収集には、DBMS_STATS.GATHER_TABLE_STATS プロシージャを使用してください。これらの機能が完了すると、レポートおよびスクリプト生成ファンクションで分析結果を表示できます。

ワークロード管理機能は 3 種類のワークロード、つまりユーザー定義ワークロード、SQL キャッシュ・ワークロードおよび Oracle Trace ワークロードを処理します。ユーザー指定のワークロードを処理するには、そのユーザーのスキーマにユーザー定義のワークロード表が必要です。Oracle Trace ワークロードを処理するには、収集したワークロード統計を、Oracle Trace フォーマッタを実行して前処理し、ユーザーのスキーマのデフォルトの V 表に組み込む必要があります。

エラー・メッセージ

表 38-1 に、DBMS_OLAP の主なエラー・メッセージを示します。

表 38-1 DBMS_OLAP エラー・メッセージ

エラー・コード	説明
ORA-30442	フィルタ <NUMBER> の定義が見つかりません。
ORA-30443	フィルタ <NUMBER> の項目 <NUMBER> の定義が無効です。
ORA-30444	SQL アナライザによってリライトが終了されました。
ORA-30445	ワークロード問合せが見つかりません。
ORA-30446	有効なワークロード問合せが見つかりません。
ORA-30447	実行番号 <NUMBER> の内部データに一貫性がありません。
ORA-30448	アドバイザ・リポジトリの内部データに一貫性がありません。
ORA-30449	パラメータ <NUMBER> に構文エラーがあります。
ORA-30465	指定した run_id は無効です : <NUMBER>
ORA-30466	指定したワークロード <NUMBER> が見つかりません

表 38-1 DBMS_OLAP エラー・メッセージ（続き）

エラー・コード	説明
ORA-30477	SELECT_CLAUSE の指定が正しくありません。
ORA-30478	指定されたディメンションは存在しません。
ORA-30479	サマリー・アドバイザ・エラーが発生しました <string>
QSM-00501	サマリー・アドバイザ環境の初期化ができません。
QSM-00502	OCI エラー
QSM-00503	メモリー不足です。
QSM-00504	内部エラー
QSM-00505	<parse_entity_name> - <error_description> 構文エラー
QSM-00506	ファクト表が見つかりません
QSM-00507	ディメンションが見つかりません。
QSM-00508	統計表示が表 / 列にありません。
QSM-00509	無効なパラメータ <parameter_name>
QSM-00510	統計表示がサマリーにありません。
QSM-00511	無効なファクト表がファクト・フィルタで指定されています
QSM-00512	無効なサマリーがリテンション・リストに指定されています。
QSM-00513	ワークロード表が 1 つ以上不足しています
QSM-00550	フィルタ項目型 <NAME> に必須のデータがありません。
QSM-00551	ファイル <NAME> が見つかりませんでした。
QSM-00552	ワークロード・ソースが定義あるいは認識されていません。
QSM-00553	フィルタ項目 <NAME> の文字列値の最大長は <NUMBER> 文字です
QSM-00554	必須の表名がありません。
QSM-00555	表 <NAME> がアクセスできないか存在しません。
QSM-00556	ファイル <NAME> をオープンできませんでした。
QSM-00557	所有者 <NAME> は、アクセスできないか、または存在しません
QSM-00558	ファイル <NAME> の読込み中にエラーが発生しました。
QSM-00559	指定したワークロード ID のワークロードがすでに存在します。
QSM-00560	文字 <NAME> は行 <LINE_NUMBER>、列 <COLUMN_NUMBER で無効です。

表 38-1 DBMS_OLAP エラー・メッセージ（続き）

エラー・コード	説明
QSM-00561	<TOKEN> が行 <NUMBER>、列 <NUMBER> で見つかりました。次の項目の 1 つが必要です <ITEMS>
QSM-00562	要求された Advisor タスクが見つかりませんでした
QSM-00563	<TOKEN> がファイル <NAME> の行 <NUMBER>、列 <NUMBER> で見つかりました。次の項目の 1 つが必要です : <ITEMS>
QSM-00564	内部字句エラーが発生しました : < エラー内容のテキスト >
QSM-00565	行 <NUMBER>、列 <NUMBER> で <TABLE または COLUMN> を検査中に <NAME> が見つかりませんでした。
QSM-00566	行 <NUMBER>、列 <NUMBER> で <TABLE または COLUMN> を検査中に <TOKEN> があいまいでした。
QSM-00567	ランタイム・エラーが発生しました : < エラー内容のテキスト >
QSM-00568	ファイル終りが見つかりました。
QSM-00569	必須の列 <NAME> が表 <NAME> で見つかりませんでした。
QSM-00570	ジョブはエラーで終了しました。ステータスの変更はできません。
QSM-00571	ジョブはすでに完了しています。ステータスの変更は不要です。
QSM-00572	リポジトリ接続は確立されていません。
QSM-00573	日付 <VALUE> の形式は、'DD/MM/YYYY HH24:MI:SS' または 'DD/MM/YYYY' である必要があります
QSM-00574	セキュリティ違反のため、ファイル <NAME> にアクセスできませんでした
QSM-00575	文字列 <VALUE> は数字に変換できません。
QSM-00576	使用可能な Oracle Trace コレクションがスキーマ <NAME> に見つかりませんでした
QSM-00577	現行の操作はユーザーに取り消されました。
QSM-00578	仕様 <FILE_NAME> を使用して一時ファイルは作成できません。
QSM-00579	ジョブはすでに完了しています。取消は不要です。
QSM-00580	ジョブはエラーで終了しました。取消はできません。
QSM-00581	内部エラー : < エラー内容のテキスト >
QSM-00582	データベース・エラーが発生しました。< エラー内容のテキスト >
QSM-00583	フィルタ項目の型 <NAME> が無効です。

表 38-1 DBMS_OLAP エラー・メッセージ（続き）

エラー・コード	説明
QSM-00584	ユーザー <NAME> は SQL キャッシュにアクセスできません
QSM-00585	ワークロード ID <NUMBER> のワークロードが見つかりません
QSM-00586	フィルタ ID <NUMBER> のフィルタが見つかりません
QSM-00587	実行 ID <NUMBER> の分析データが見つかりません
QSM-00588	カレント・ユーザーは、要求したワークロードにアクセスする権限がありません。このワークロードはユーザー <NAME> が所有しています
QSM-00589	カレント・ユーザーは、要求したワークロード・フィルタにアクセスする権限がありません。このワークロード・フィルタはユーザー <NAME> が所有しています
QSM-00590	カレント・ユーザーは、要求したアドバイザ項目にアクセスする権限がありません。この項目はユーザー <NAME> が所有しています
QSM-00591	指定されたレポート・スタイル <NAME> が見つかりませんでした。
QSM-00592	指定されたレポート・フィールド <NAME> はすでに存在します。
QSM-00593	指定されたレポート・フィールド <NAME> が見つかりませんでした。
QSM-00594	指定された ID 番号は他のユーザーが使用しています
QSM-00595	指定された ID 番号は Advisor <NAME> オブジェクトで使用しているためこの処理には使用できません
QSM-00596	NULL またはゼロの ID 番号は指定できません
QSM-00597	<TOKEN> が行 <NUMBER>、列 <NUMBER> で見つかりました
QSM-00598	フィルタ項目 <NAME> の範囲の最小値が最大値より大きい値です。
QSM-00599	指定したワークロード・フィルタには、要求したワークロード操作：<OPERATION> に対してサポートされない項目が含まれています
QSM-00602	フィルタ項目 <NUMBER> に無効な名前リスト： <VALUE> が含まれています。
QSM-00601	サマリー・アドバイザ詳細レポートの <NUMBER> のフラッグ値が無効です

DBMS_OLAP サブプログラムの要約

表 38-2 に、DBMS_OLAP で使用可能なサブプログラムを示します。

表 38-2 DBMS_OLAP パッケージのサブプログラム

サブプログラム	説明
「ADD_FILTER_ITEM プロシージャ」 38-7 ページ	推奨プロセスで使用中のコンテンツをフィルタにかけます。
「CREATE_ID プロシージャ」 38-9 ページ	新しいワークロード・コレクション、フィルタまたはアドバイザ実行で使用する内部 ID を生成します。
「ESTIMATE_MVIEW_SIZE プロシージャ」 38-10 ページ	作成するマテリアライズド・ビューのサイズを、バイトと行で見積ります。
「EVALUATE_MVIEW_STRATEGY プロシージャ」 38-10 ページ	既存の各マテリアライズド・ビューの使用率を測定します。
「GENERATE_MVIEW_REPORT プロシージャ」 38-11 ページ	指定されたアドバイザ実行で HTML ベースのレポートを生成します。
「GENERATE_MVIEW_SCRIPT プロシージャ」 38-13 ページ	サマリー・アドバイザ推奨を実装するための SQL コマンドが含まれた簡単なスクリプトを生成します。
「LOAD_WORKLOAD_CACHE プロシージャ」 38-14 ページ	SQL キャッシュ・ワークロードを取得します。
「LOAD_WORKLOAD_TRACE プロシージャ」 38-15 ページ	Oracle Trace が収集したワークロードをロードします。
「LOAD_WORKLOAD_USER プロシージャ」 38-16 ページ	ユーザー定義ワークロードをロードします。
「PURGE_FILTER プロシージャ」 38-17 ページ	特定のフィルタまたはすべてのフィルタを削除します。
「PURGE_RESULTS プロシージャ」 38-18 ページ	すべての結果または特定の実行の結果を削除します。
「PURGE_WORKLOAD プロシージャ」 38-18 ページ	すべてのワークロードまたは特定のコレクションを削除します。
「RECOMMEND_MVIEW_STRATEGY プロシージャ」 38-19 ページ	作成、保持または削除するマテリアライズド・ビューについて、リコメンデーション・セットを生成します。
「SET_CANCELLED プロシージャ」 38-21 ページ	結果が戻されるまでの時間が長すぎる場合にアドバイザを停止します。

表 38-2 DBMS_OLAP パッケージのサブプログラム （続き）

サブプログラム	説明
「VALIDATE_DIMENSION プロシージャ」38-21 ページ	ディメンションで指定した関連が正しいことを検証します。
「VALIDATE_WORKLOAD_CACHE プロシージャ」38-22 ページ	ロード処理の実行前に SQL キャッシュ・ワークロードを検証します。
「VALIDATE_WORKLOAD_TRACE プロシージャ」38-23 ページ	ロード処理の実行前に Oracle Trace ワークロードを検証します。
「VALIDATE_WORKLOAD_USER プロシージャ」38-23 ページ	ロード処理の実行前にユーザー指定ワークロードを検証します。

ADD_FILTER_ITEM プロシージャ

このプロシージャは、既存のフィルタに新しいフィルタ項目を追加して制限を強化します。ワークロードの分析対象を制限するフィルタも作成します。

構文

```
ADD_FILTER_ITEM (
   filter_id    IN NUMBER,
   filter_name  IN VARCHAR2,
   string_list  IN VARCHAR2,
   number_min   IN NUMBER,
   number_max   IN NUMBER,
   date_min     IN VARCHAR2,
   date_max     IN VARCHAR2);
```

表 38-3 ADD_FILTER_ITEM プロシージャのパラメータ

パラメータ	データ・タイプ	説明
filter_id	NUMBER	フィルタを一意に示す ID。DBMS_OLAP.CREATE_ID プロシージャで生成します。
filter_name	VARCHAR2	APPLICATION 文字列－ワークロードの APPLICATION 列。SQL キャッシュ・ワークロードのロード方法の例を次に示します。 BASETABLE 文字列－ワークロードの問合せの対象となる表。名前は所有者および表名を含めて完全修飾する必要があります（たとえば、SH.SALES）。 CARDINALITY 数値－問合せの対象となるカーディナリティの合計。 FREQUENCY 数値－ワークロードの FREQUENCY 列。 LASTUSE 日付－ワークロードの LASTUSE 列。SQL キャッシュ・ワークロードでは使用しません。 OWNER 文字列－ワークロードの OWNER 列。所有者は、すべてを大文字で記述するわけではないことを明示的に定義しないかぎり、大文字にする必要があります。 PRIORITY 数値－ワークロードの PRIORITY 列。SQL キャッシュ・ワークロードでは使用しません。 RESPONSETIME 数値－ワークロードの RESPONSETIME 列。SQL キャッシュ・ワークロードでは使用しません。 SCHEMA 文字列－ワークロードの問合せの対象となるスキーマ。 TRACENAME 文字列－ Oracle Trace のコレクション名のリスト。トレース・ワークロードでのみ使用されます。
string_list	VARCHAR2	カンマで区切られた文字列のリスト。文字列タイプのフィルタ項目でのみ使用されます。

表 38-3 ADD_FILTER_ITEM プロシージャのパラメータ（続き）

パラメータ	データ・タイプ	説明
number_min	NUMBER	数値範囲の下限。NULL は可能最低値を示します。数値タイプのパラメータでのみ使用されます。
number_max	NUMBER	数値範囲の上限。NULL は上限ではありません。NULL は可能最高値を示します。数値タイプのパラメータでのみ使用されます。
date_min	VARCHAR2	日付範囲の下限。NULL は日付の可能最低値を示します。日付タイプのパラメータでのみ使用されます。
date_max	VARCHAR2	日付範囲の上限。NULL は日付の可能最高値を示します。日付タイプのパラメータでのみ使用されます。

CREATE_ID プロシージャ

アドバイザまたはディメンション検証の実行のフィルタ、ワークロードまたは結果を識別するのに使用する一意の識別子を作成します。

構文

```
CALL DBMS_OLAP.CREATE_ID (
  id        OUT NUMBER);
```

表 38-4 CREATE_ID プロシージャのパラメータ

パラメータ	データ・タイプ	説明
id	NUMBER	フィルタ、ワークロードまたはアドバイザ実行を識別するのに使用する一意の識別子。

ESTIMATE_MVIEW_SIZE プロシージャ

このプロシージャでは、作成するマテリアライズド・ビューのサイズをバイトおよび行数で見積ります。

構文

```
DBMS_OLAP.ESTIMATE_MVIEW_SIZE (
   stmt_id       IN  VARCHAR2,
   select_clause IN  VARCHAR2,
   num_rows      OUT NUMBER,
   num_bytes     OUT NUMBER);
```

パラメータ

表 38-5 ESTIMATE_MVIEW_SIZE プロシージャのパラメータ

パラメータ	データ・タイプ	説明
stmt_id	NUMBER	EXPLAIN PLAN で文を識別するために使用する任意の文字列。
select_clause	STRING	分析用の SELECT 文。
num_rows	NUMBER	推測カーディナリティ。
num_bytes	NUMBER	推測バイト数。

EVALUATE_MVIEW_STRATEGY プロシージャ

このプロシージャでは、ワークロードから収集したマテリアライズド・ビューの使用統計に基づいて、既存の各マテリアライズド・ビューの使用率を測定します。workload_id はオプションです。指定しない場合、EVALUATE_MVIEW_STRATEGY には仮定ワークロードが使用されます。

DBMS_OLAP.PURGE_RESULTS プロシージャをコールすることで、未使用の結果を定期的にシステムからパージできます。

関連項目： 38-24 ページ「DBMS_OLAP インタフェース・ビュー」

構文

```
DBMS_OLAP.EVALUATE_MVIEW_STRATEGY (
run_id        IN NUMBER,
workload_id   IN NUMBER,
filter_id     IN NUMBER);
```

パラメータ

表 38-6 EVALUATE_MVIEW_STRATEGY プロシージャのパラメータ

パラメータ	データ・タイプ	説明
run_id	NUMBER	DBMS_OLAP.CREATE_ID プロシージャで生成される ID で、実行結果の識別に使用されます。
workload_id	NUMBER	現行のリポジトリのワークロードにマップされるオプションのワークロード ID。すべてのワークロードを選択する場合は、DBMS_OLAP.WORKLOAD_ALL パラメータを使用します。
filter_id	NUMBER	ワークロードに使用するフィルタを指定します。値 DBMS_OLAP.FILTER_NONE はフィルタリングを行わないことを示します。

GENERATE_MVIEW_REPORT プロシージャ

指定されたアドバイザ実行で HTML ベースのレポートを生成します。

構文

```
DBMS_OLAP.GENERATE_MVIEW_REPORT (
filename     IN VARCHAR2,
id           IN NUMBER,
flags        IN NUMBER);
```

表 38-7 GENERATE_MVIEW_REPORT プロシージャのパラメータ

パラメータ	データ・タイプ	説明
filename	VARCHAR2	HTML データを受信するための完全修飾出力ファイル名。Oracle サーバーでは、ファイル・アクセスは Oracle ストアド・プロシージャ内に限定されます。ファイルのアクセス許可の詳細は、『Oracle9*i* Java Developer's Guide』を参照してください。
id	NUMBER	アドバイザ実行を識別する ID。すべてのアドバイザ実行をレポートすることを示すには、パラメータ DBMS_OLAP.RUNID_ALL を使用します。
flags	NUMBER	レポートするセクションを示すビット・マスク・フラグ。 DBMS_OLAP.RPT_ACTIVITY -- アクティビティ全般 DBMS_OLAP.RPT_JOURNAL -- 実行時ジャーナル DBMS_OLAP.RPT_WORKLOAD_FILTER -- フィルタ DBMS_OLAP.RPT_WORKLOAD_DETAIL -- ワークロード情報 DBMS_OLAP.RPT_WORKLOAD_QUERY -- ワークロード問合せ情報 DBMS_OLAP.RPT_RECOMMENDATION -- リコメンデーション DBMS_OLAP.RPT_USAGE -- マテリアライズド・ビューの使用方法 DBMS_OLAP.RPT_ALL -- すべてのセクション

GENERATE_MVIEW_SCRIPT プロシージャ

サマリー・アドバイザ推奨を実装するための SQL コマンドが含まれた簡単なスクリプトを生成します。

構文

```
DBMS_OLAP.GENERATE_MVIEW_SCRIPT(
filename     IN VARCHAR2,
id           IN NUMBER,
tspace        IN VARCHAR2);
```

表 38-8 GENERATE_MVIEW_SCRIPT プロシージャのパラメータ

パラメータ	データ・タイプ	説明
filename	VARCHAR2	HTML データを受信するための完全修飾出力ファイル名。Oracle サーバーでは、ファイル・アクセスは Oracle ストアド・プロシージャ内に限定されます。ファイルのアクセス許可の詳細は、『Oracle9*i* Java Developer's Guide』を参照してください。
id	NUMBER	アドバイザ実行を識別する ID。すべてのアドバイザ実行をレポートすることを示すには、DBMS_OLAP.RUNID_ALL パラメータを使用します。
tspace	VARCHAR2	マテリアライズド・ビューの作成時に使用するオプションの表領域名。

LOAD_WORKLOAD_CACHE プロシージャ

SQL キャッシュ・ワークロードをロードします。

構文

```
DBMS_OLAP.LOAD_WORKLOAD_CACHE (
workload_id  IN NUMBER,
flags        IN NUMBER,
filter_id    IN NUMBER,
application  IN VARCHAR2,
priority     IN NUMBER);
```

表 38-9 LOAD_WORKLOAD_CACHE プロシージャのパラメータ

パラメータ	データ・タイプ	説明
workload_id	NUMBER	HTML データを受信するための完全修飾出力ファイル名。Oracle サーバーでは、ファイル・アクセスは Oracle ストアド・プロシージャ内に限定されます。ファイルのアクセス許可の詳細は、『Oracle9*i* Java Developer's Guide』を参照してください。
flags	NUMBER	DBMS_OLAP.WORKLOAD_OVERWRITE 指定したコレクション ID が所有するワークロードから、既存の問合せを明示的に削除するロード・ルーチン。 DBMS_OLAP.WORKLOAD_APPEND ワークロードの既存の問合せを保存するロード・ルーチン。ロード処理で収集した問合せは指定のワークロードの終わりに追加されます。 DBMS_OLAP.WORKLOAD_NEW ワークロードに既存の問合せがないものと仮定するロード・ルーチン。既存のワークロード要素が見つかると、コールが失敗してエラーが発生します。 **注意：**各フラグの動作は、LOAD_WORKLOAD 処理に関係なく同じです。
filter_id	NUMBER	ロードするワークロードにフィルタを指定します。
application	VARCHAR2	デフォルトのビジネス・アプリケーション名。この値はターゲット・ワークロードに見つからない場合、問合せに使用されます。
priority	NUMBER	ターゲット・ワークロードのすべての問合せに割り当てられる、デフォルトのビジネス優先順位。

LOAD_WORKLOAD_TRACE プロシージャ

Oracle Trace ワークロードをロードします。

構文

```
DBMS_OLAP.LOAD_WORKLOAD_TRACE (
workload_id  IN NUMBER,
flags        IN NUMBER,
filter_id    IN NUMBER,
application  IN VARCHAR2,
priority     IN NUMBER,
owner_name   IN VARCHAR2);
```

表 38-10 LOAD_WORKLOAD_TRACE プロシージャのパラメータ

パラメータ	データ・タイプ	説明
workload_id	NUMBER	HTML データを受信するための完全修飾出力ファイル名。Oracle サーバーでは、ファイル・アクセスは Oracle ストアド・プロシージャ内に限定されます。ファイルのアクセス許可の詳細は、『Oracle9*i* Java Developer's Guide』を参照してください。
flags	NUMBER	DBMS_OLAP.WORKLOAD_OVERWRITE 指定したコレクション ID が所有するワークロードから、既存の問合せを明示的に削除するロード・ルーチン。 DBMS_OLAP.WORKLOAD_APPEND ワークロードの既存の問合せを保存するロード・ルーチン。ロード処理で収集した問合せは指定のワークロードの終わりに追加されます。 DBMS_OLAP.WORKLOAD_NEW ワークロードに既存の問合せがないものと仮定するロード・ルーチン。既存のワークロード要素が見つかると、コールが失敗してエラーが発生します。 **注意 :** 各フラグの動作は、LOAD_WORKLOAD 処理に関係なく同じです。

表 38-10 LOAD_WORKLOAD_TRACE プロシージャのパラメータ（続き）

パラメータ	データ・タイプ	説明
filter_id	NUMBER	ロードするワークロードにフィルタを指定します。
application	VARCHAR2	デフォルトのビジネス・アプリケーション名。この値はターゲット・ワークロードに見つからない場合、問合せに使用されます。
priority	NUMBER	ターゲット・ワークロードのすべての問合せに割り当てられる、デフォルトのビジネス優先順位。
owner_name	VARCHAR2	Oracle Trace データを含むスキーマ。省略した場合はカレント・ユーザーが使用されます。

LOAD_WORKLOAD_USER プロシージャ

LOAD_WORKLOAD_USER プロシージャを使用してロードするユーザー定義ワークロードです。

構文

```
DBMS_OLAP.LOAD_WORKLOAD_USER (
workload_id  IN NUMBER,
flags        IN NUMBER,
filter_id    IN NUMBER,
owner_name   IN VARCHAR2,
table_name   IN VARCHAR2);
```

表 38-11 LOAD_WORKLOAD_USER プロシージャのパラメータ

パラメータ	データ・タイプ	説明
workload_id	NUMBER	DBMS_OLAP.CREATE_ID コールによって戻される必須 ID。
flags	NUMBER	DBMS_OLAP.WORKLOAD_OVERWRITE 指定したコレクション ID が所有するワークロードから、既存の問合せを明示的に削除するロード・ルーチン。 DBMS_OLAP.WORKLOAD_APPEND ワークロードの既存の問合せを保存するロード・ルーチン。ロード処理で収集した問合せは指定のワークロードの終わりに追加されます。 DBMS_OLAP.WORKLOAD_NEW ワークロードに既存の問合せがないものと仮定するロード・ルーチン。既存のワークロード要素が見つかると、コールが失敗してエラーが発生します。 **注意 :** 各フラグの動作は、LOAD_WORKLOAD 処理に関係なく同じです。
filter_id	NUMBER	ロードするワークロードにフィルタを指定します。
owner_name	VARCHAR2	ユーザー指定の表またはビューを含むスキーマ。
table_name	VARCHAR2	有効なワークロード・データを含む表名またはビュー名。

PURGE_FILTER プロシージャ

次に説明するプロシージャ PURGE_FILTER をコールすることで、任意の時点でフィルタを削除できます。特定のフィルタを削除することも、すべてのフィルタを削除することもできます。

構文

```
DBMS_OLAP.PURGE_FILTER (
filter_id    IN NUMBER);
```

表 38-12 PURGE_FILTER プロシージャのパラメータ

パラメータ	データ・タイプ	説明
filter_id	NUMBER	DBMS_OLAP.FILTER_ALL パラメータは、すべてのフィルタが削除されることを示します。

PURGE_RESULTS プロシージャ

DBMS_OLAP パッケージの多くのプロシージャは、DBMS_OLAP.RECOMMEND_MVIEW_STRATEGY の推奨結果、DBMS_OLAP.EVALUATE_MVIEW_STRATEGY の評価結果および DBMS_OLAP.VALIDATE_DIMENSION のディメンション検証結果などの出力をシステム表に生成します。これらの結果にアクセスするには、38-24 ページの「DBMS_OLAP インタフェース・ビュー」に示した一連のインタフェース・ビューを使用します。必要なくなった場合は、PURGE_RESULTS プロシージャで削除します。すべての結果を削除することも、特定の実行の結果を削除することもできます。

構文

```
DBMS_OLAP.PURGE_RESULTS (
run_id IN NUMBER);
```

パラメータ

表 38-13 PURGE_RESULTS プロシージャのパラメータ

パラメータ	データ・タイプ	説明
run_id	NUMBER	DBMS_OLAP.CREATE_ID プロシージャで生成する ID。DBMS_OLAP.RECOMMEND_MVIEW_STRATEGY、DBMS_OLAP.EVALUATE_MVIEW_STRATEGY または DBMS_OLAP.VALIDATE_DIMENSION のいずれかの実行に関連付けます。このような実行をすべて指定するには値 DBMS_OLAP.RUNID_ALL を使用します。

PURGE_WORKLOAD プロシージャ

必要なくなったワークロードは、PURGE_WORKLOAD プロシージャで削除します。すべてのワークロードを削除することも、特定のコレクションを削除することもできます。

構文

```
DBMS_OLAP.PURGE_WORKLOAD (
workload_id IN NUMBER);
```

表 38-14 DBMS_OLAP.PURGE_WORKLOAD プロシージャのパラメータ

パラメータ	データ・タイプ	説明
workload_id	NUMBER	create_id コールによって最初に割り当てられる ID 番号。ワークロードの値を DBMS_OLAP.WORKLOAD_ALL に設定すると、カレント・ユーザーのすべてのワークロードが削除されます。

RECOMMEND_MVIEW_STRATEGY プロシージャ

このプロシージャは、作成、保持または削除するマテリアライズド・ビューについてリコメンデーション・セットを生成します。これは、ワークロード（Oracle Trace、ユーザー・ワークロードまたは SQL キャッシュで収集）の情報および DBMS_STATS.GATHER_TABLE_STATS で収集された表および列のカーディナリティ統計の分析に基づいて行われます。

RECOMMEND_MVIEW_STRATEGY では、DBMS_STATS.GATHER_TABLE_STATS プロシージャを実行して表および列のカーディナリティ統計を収集し、ワークロード統計を収集および書式化する必要があります。

ワークロードはそのワークロードでの各要求件数を判断するために集計されます。この件数は最適化プロセスで重み付け要因として使用されます。workload_id を指定しない場合は、ディメンション定義およびその他の埋込み統計に基づいた仮定ワークロードが使用されます。

指定されたファクト表を含むすべてのディメンショナル・マテリアライズド・ビューの領域は、ワークロード全体のパフォーマンスを最適化するマテリアライズド・ビューのセットを示します。推奨結果はシステム表に格納されます。アクセスするにはビュー SYSTEM.MVIEW_RECOMMENDATIONS を使用します。

DBMS_OLAP.PURGE_RESULTS プロシージャをコールすることで、未使用の結果を定期的にシステムからパージできます。

関連項目： 38-24 ページ「DBMS_OLAP インタフェース・ビュー」

構文

```
DBMS_OLAP.RECOMMEND_MVIEW_STRATEGY (
   run_id              IN  NUMBER,
   workload_id         IN  NUMBER,
   filter_id           IN  NUMBER,
   storage_in_bytes    IN  NUMBER,
   retention_pct       IN  NUMBER,
   retention_list      IN  VARCHAR2,
   fact_table_filter   IN  VARCHAR2);
```

パラメータ

表 38-15 RECOMMEND_MVIEW_STRATEGY プロシージャのパラメータ

パラメータ	説明
run_id	DBMS_OLAP.CREATE_ID プロシージャで生成される ID で、実行結果を一意に識別するのに使用されます。
workload_id	現行のリポジトリのワークロードにマップされるオプションのワークロード ID。すべてのワークロードを選択する場合はパラメータ DBMS_OLAP.WORKLOAD_ALL を使用します。 workload_id を NULL に設定すると、仮定ワークロードが使用されます。
filter_id	一連のユーザー指定フィルタ項目にマップされるオプションのフィルタ ID。フィルタリングを行わない場合は DBMS_OLAP.FILTER_NONE パラメータを使用します。
storage_in_bytes	マテリアライズド・ビューの格納に使用できる記憶域の最大数(バイト)。負の数は指定できません。
retention_pct	実際または仮定されるワークロードに基づき、保持する必要がある既存のマテリアライズド・ビュー記憶域を 0 ～ 100 パーセントで指定します。 使用率で順位付けした累積領域が指定の保存しきい値内の場合(または retention_list に明示的にリストされている場合)、マテリアライズド・ビューは保持されます。使用率が NULL のマテリアライズド・ビュー（たとえば、非ディメンショナルのマテリアライズド・ビュー）は、常に保持されます。
retention_list	マテリアライズド・ビュー表名のカンマで区切られたリスト。このリストに表示されるマテリアライズド・ビューに対して、削除の推奨は行いません。
fact_table_filter	実際のワークロードまたは理想的なワークロードをフィルタにかける際に使用する、ファクト表のオプションのリスト。

SET_CANCELLED プロシージャ

RECOMMEND_MVIEW_STRATEGY プロシージャを使用したリコメンデーション作成に時間がかかりすぎる場合は、SET_CANCELLED プロシージャをコールし、該当する推奨プロセスの run_id を渡すことでサマリー・アドバイザを停止できます。

構文

```
DBMS_OLAP.SET_CANCELLED (
   run_id      IN  NUMBER);
```

表 38-16 DBMS_OLAP.SET_CANCELLED プロシージャのパラメータ

パラメータ	データ・タイプ	説明
run_id	NUMBER	アドバイザの分析処理を一意に識別する ID。このコールは、長時間実行のワークロード・コレクションやアドバイザの分析セッションを取り消す場合に使用します。

VALIDATE_DIMENSION プロシージャ

このプロシージャでは、既存のディメンション・オブジェクトで指定されている階層関係と属性関係、および結合関係が正しいことを検証します。これにより、参照整合性が保守されていることを迅速に確認できます。

検証結果はシステム表に格納されます。アクセスするには SYSTEM.MVIEW_EXCEPTIONS ビューを使用します。

DBMS_OLAP.PURGE_RESULTS プロシージャをコールすることで、未使用の結果を定期的にシステムからパージできます。

関連項目： 38-24 ページ「DBMS_OLAP インタフェース・ビュー」

構文

```
DBMS_OLAP.VALIDATE_DIMENSION (
   dimension_name     IN VARCHAR2,
   dimension_owner    IN VARCHAR2,
   incremental        IN BOOLEAN,
   check_nulls        IN BOOLEAN,
   run_id             IN NUMBER);
```

パラメータ

表 38-17 VALIDATE_DIMENSION プロシージャのパラメータ

パラメータ	説明
dimension_name	分析するディメンション名。
dimension_owner	ディメンション所有者の名前。
incremental	TRUE の場合は、このディメンションの表の sumdelta$ 表で指定された行に対してのみ、テストが実行されます。そうでない場合は、すべての行をチェックします。
check_nulls	TRUE の場合は、すべてのレベルの列が NULL でないことを検証します。それ以外の場合、このチェックは省略されます。 NOT NULL 制約などの別の方法で NULL でないことが保証されている場合は、FALSE を指定します。
run_id	DBMS_OLAP.CREATE_ID プロシージャで生成される ID で、実行の識別に使用されます。

VALIDATE_WORKLOAD_CACHE プロシージャ

このプロシージャでは、ロード処理の実行前に SQL キャッシュ・ワークロードを検証します。

構文

```
DBMS_OLAP.VALIDATE_WORKLOAD_CACHE (
   valid            OUT NUMBER,
   error            OUT VARCHAR2);
```

パラメータ

表 38-18 VALIDATE_WORKLOAD_CACHE プロシージャのパラメータ

パラメータ	説明
valid	DBMS_OLAP.VALID または DBMS_OLAP.INVALID を戻します。ワークロードが有効であるかどうかを示します。
error	VARCHAR2、エラーのセットを戻します。

VALIDATE_WORKLOAD_TRACE プロシージャ

ロード処理の実行前に Oracle Trace ワークロードを検証します。

構文

```
DBMS_OLAP.VALIDATE_WORKLOAD_TRACE (
   owner_name       IN  VARCHAR2,
   valid            OUT NUMBER,
   error            OUT VARCHAR2);
```

パラメータ

表 38-19 VALIDATE_WORKLOAD_TRACE プロシージャのパラメータ

パラメータ	説明
owner_name	トレース・ワークロード表の所有者。
valid	DBMS_OLAP.VALID または DBMS_OLAP.INVALID を戻します。ワークロードが有効であるかどうかを示します。
error	VARCHAR2、エラー・テキストを戻します。

VALIDATE_WORKLOAD_USER プロシージャ

このプロシージャでは、ロード処理の実行前にユーザー指定ワークロードを検証します。

構文

```
DBMS_OLAP.VALIDATE_WORKLOAD_USER (
   owner_name       IN  VARCHAR2,
   table_name       IN  VARCHAR2,
   valid            OUT NUMBER,
   error            OUT VARCHAR2);
```

パラメータ

表 38-20 VALIDATE_WORKLOAD_USER プロシージャのパラメータ

パラメータ	説明
owner_name	ユーザー・ワークロード表の所有者。
table_name	ユーザー・ワークロード表名。
valid	DBMS_OLAP.VALID または DBMS_OLAP.INVALID を戻します。ワークロードが有効であるかどうかを示します。
error	VARCHAR2、エラーのセットを戻します。

DBMS_OLAP インタフェース・ビュー

いくつかのビューは、DBMS_OLAP で作成します。いずれのビューも SYSTEM スキーマに存在します。これらのビューにアクセスするには、DBA ロールを取得する必要があります。

関連項目： DBMS_OLAP の使用方法の詳細は、『Oracle9*i* データ・ウェアハウス・ガイド』を参照してください。

SYSTEM.MVIEW_EVALUATIONS

表 38-21 SYSTEM.MVIEW_EVALUATIONS

列	NULL?	データ・タイプ	説明
RUNID	NOT NULL	NUMBER	一意のアドバイザ・コールを識別する実行 ID。
MVIEW_OWNER	—	VARCHAR2(30)	マテリアライズド・ビューの所有者。
MVIEW_NAME	—	VARCHAR2(30)	このデータベースにある既存のマテリアライズド・ビュー名。
RANK	NOT NULL	NUMBER	benefit_to_cost_ratio の降順で示される、このマテリアライズド・ビューの順位。
STORAGE_IN_BYTES	—	NUMBER	マテリアライズド・ビューのサイズ（バイト）。

表 38-21 SYSTEM.MVIEW_EVALUATIONS（続き）

列	NULL?	データ・タイプ	説明
FREQUENCY	—	NUMBER	このマテリアライズド・ビューがワークロードに記述される回数。
CUMULATIVE_BENEFIT	—	NUMBER	マテリアライズド・ビューの累積利点。
BENEFIT_TO_COST_RATIO	NOT NULL	NUMBER	cumulative_benefit と storage_in_bytes の比率。

SYSTEM.MVIEW_EXCEPTIONS

表 38-22 SYSTEM.MVIEW_EXCEPTIONS

列	NULL?	データ・タイプ	説明
RUNID	—	NUMBER	一意のアドバイザ・コールを識別する実行 ID
OWNER	—	VARCHAR2(30)	所有者名
TABLE_NAME	—	VARCHAR2(30)	表名
DIMENSION_NAME	—	VARCHAR2(30)	ディメンション名
RELATIONSHIP	—	VARCHAR2(11)	違反リレーション名
BAD_ROWID	—	ROWID	違反エントリの場所

SYSTEM.MVIEW_FILTER

表 38-23 SYSTEM.MVIEW_FILTER

列	NULL?	データ・タイプ	説明
FILTERID	NOT NULL	NUMBER	このフィルタを使用した処理を識別するのに使用する一意の番号。
SUBFILTERNUM	NOT NULL	NUMBER	すべてのフィルタ項目をまとめてグループ化する一意の ID 番号。対応するフィルタ・ヘッダー・レコードは、MVIEW_LOG 表にあります。
SUBFILTERTYPE	—	VARCHAR2(12)	フィルタ項目番号。
STR_VALUE	—	VARCHAR2(1028)	文字列が必要な項目の文字列属性。

表 38-23 SYSTEM.MVIEW_FILTER（続き）

列	NULL?	データ・タイプ	説明
NUM_VALUE1	—	NUMBER	番号が必要な項目の数値（下限）。
NUM_VALUE2	—	NUMBER	番号が必要な項目の数値（上限）。
DATE_VALUE1	—	DATE	日付が必要な項目の日付（下限）。
DATE_VALUE2	—	DATE	日付が必要な項目の日付（上限）。

SYSTEM.MVIEW_FILTERINSTANCE

表 38-24 SYSTEM.MVIEW_FILTER

列	NULL?	データ・タイプ	説明
RUNID	NOT NULL	NUMBER	このフィルタを使用した処理を識別するのに使用する一意の番号。
FILTERID	—	NUMBER	すべてのフィルタ項目をまとめてグループ化する一意の ID 番号。対応するフィルタ・ヘッダー・レコードは、MVIEW_LOG 表にあります。
SUBFILTERNUM	—	NUMBER	フィルタ項目番号。
SUBFILTERTYPE	—	VARCHAR2(12)	フィルタ項目タイプ。
STR_VALUE	—	VARCHAR2(1028)	文字列が必要な項目の文字列属性。
NUM_VALUE1	—	NUMBER	番号が必要な項目の数値（下限）。
NUM_VALUE2	—	NUMBER	番号が必要な項目の数値（上限）。
DATE_VALUE1	—	DATE	日付が必要な項目の日付（下限）。
DATE_VALUE2	—	DATE	日付が必要な項目の日付（上限）。

SYSTEM.MVIEW_LOG

表 38-25 SYSTEM.MVIEW_LOG

列	NULL?	データ・タイプ	説明
ID	NOT NULL	NUMBER	表エントリの識別に使用する一意の名前。この番号を作成するには CREATE_ID ルーチンを使用します。
FILTERID	—	NUMBER	オプションのフィルタ ID。0（ゼロ）はユーザー指定のフィルタが処理に割り当てられていないことを示します。
RUN_BEGIN	—	DATE	処理が開始された日付。
RUN_END	—	DATE	処理が終了した日付。
TYPE	—	VARCHAR2(11)	処理のタイプを識別する名前。
STATUS	—	VARCHAR2(11)	現行の処理ステータス。
MESSAGE	—	VARCHAR2(2000)	現行の処理または条件を示す情報メッセージ。
COMPLETED	—	NUMBER	操作の完了ステップ数。
TOTAL	—	NUMBER	実行するステップの合計数。
ERROR_CODE	—	VARCHAR2(20)	エラー発生時の Oracle エラー・コード。

SYSTEM.MVIEW_RECOMMENDATIONS

表 38-26 SYSTEM.MVIEW_RECOMMENDATIONS

列	NULL?	データ・タイプ	説明
RUNID	—	NUMBER	一意のアドバイザ・コールを識別する実行 ID。
ALL_TABLES	—	VARCHAR2(2000)	構造化リコメンデーションの完全修飾表名のカンマで区切られたリスト。
FACT_TABLES	—	VARCHAR2(1000)	構造化リコメンデーションのグループ・レベル（存在する場合）のカンマで区切られたリスト。
GROUPING_LEVELS	—	VARCHAR2(2000)	—
QUERY_TEXT	—	LONG	RECOMMENDED_ACTION が CREATE の場合のマテリアライズド・ビューの問合せテキスト。それ以外の場合は NULL になります。

表 38-26 SYSTEM.MVIEW_RECOMMENDATIONS（続き）

列	NULL?	データ・タイプ	説明
RECOMMENDATION_NUMBER	NOT NULL	NUMBER	このリコメンデーションに対する一意の識別子。
RECOMMENDED_ACTION	—	VARCHAR2(6)	CREATE、RETAIN または DROP。Retain、Create または Drop。
MVIEW_OWNER	—	VARCHAR2(30)	RECOMMENDED_ACTION が RETAIN または DROP の場合のマテリアライズド・ビューの所有者。それ以外の場合は NULL になります。
MVIEW_NAME	—	VARCHAR2(30)	RECOMMENDED_ACTION が RETAIN または DROP の場合のマテリアライズド・ビューの名前。それ以外の場合は NULL になります。
STORAGE_IN_BYTES	—	NUMBER	実際または見積りした記憶域（バイト）。
PCT_PERFORMANCE_GAIN	—	NUMBER	以前のリコメンデーションをすべて受け入れたと仮定し、当初の状態と比較して、このリコメンデーションを受け入れることによって期待できるパフォーマンスの改善率。不明な場合は NULL です。
BENEFIT_TO_COST_RATIO	NOT NULL	NUMBER	マテリアライズド・ビューのサイズ（バイト）について、パフォーマンスの改善率。不明な場合は NULL です。

SYSTEM.MVIEW_WORKLOAD

表 38-27 SYSTEM.MVIEW_WORKLOAD

列	NULL?	データ・タイプ	説明
APPLICATION	—	VARCHAR2(30)	問合せのオプションのアプリケーション名。
CARDINALITY	—	NUMBER	問合せのすべての表の合計カーディナリティ。
WORKLOADID	—	NUMBER	一意のサンプリングを識別するワークロード ID。
FREQUENCY	—	NUMBER	問合せが実行された回数。
IMPORT_TIME	—	DATE	項目が収集された日付。
LASTUSE	—	DATE	最終実行の日付。
OWNER	—	VARCHAR2(30)	最後に問合せを実行したユーザー。
PRIORITY	—	NUMBER	ユーザー指定の問合せランキング。
QUERY	—	LONG	問合せテキスト。

表 38-27 SYSTEM.MVIEW_WORKLOAD（続き）

列	NULL?	データ・タイプ	説明
QUERYID	—	NUMBER	一意の問合せを識別する ID 番号。
RESPONSETIME	—	NUMBER	実行時間（秒）。
RESULTSIZE	—	NUMBER	問合せで選択した合計バイト数。

39

DBMS_ORACLE_TRACE_AGENT

DBMS_ORACLE_TRACE_AGENT パッケージは、システム・レベルのユーティリティを提供します。

この章では、次の項目について説明します。

セキュリティ

このパッケージは、SYS 権限を持つユーザーのみアクセスできるようにデフォルト設定されています。ルーチンへのアクセスは、権限があるユーザーに実行を許可することにより制御できます。

注意： このパッケージは、DBA または Oracle Trace Collection Agent にのみ権限を付与してください。

DBMS_ORACLE_TRACE_AGENT サブプログラムの要約

このパッケージに含まれているサブプログラムは、SET_ORACLE_TRACE_IN_SESSION のみです。

SET_ORACLE_TRACE_IN_SESSION プロシージャ

このプロシージャでは、ユーザー所有以外のデータベース・セッションに関する Oracle Trace データを収集します。Oracle Trace は、（sid、serial#）で識別されるセッションで使用可能になります。これらの値は、v$session から取得されます。

構文

```
DBMS_ORACLE_TRACE_AGENT.SET_ORACLE_TRACE_IN_SESSION (
   sid                NUMBER   DEFAULT 0,
   serial#            NUMBER   DEFAULT 0,
   on_off IN          BOOLEAN  DEFAULT false,
   collection_name IN VARCHAR2 DEFAULT '',
   facility_name   IN VARCHAR2 DEFAULT '');
```

パラメータ

表 39-1 SET_ORACLE_TRACE_IN_SESSION プロシージャのパラメータ

パラメータ	説明
sid	セッション ID。
serial#	セッションのシリアル番号。
on_off	TRUE または FALSE。トレースをオンまたはオフにします。
collection_name	使用する Oracle Trace コレクション名。
facility_name	使用する Oracle Trace 機能名。

使用上の注意

コレクションが発生しない場合は、次の事項をチェックしてください。

- <facility_name> で識別されるサーバー・イベント・セット・ファイルの存在を確認してください。ファイルがフィールドに完全に指定されていない場合、ファイルは、初期化ファイルにある ORACLE_TRACE_FACILITY_PATH で識別されるディレクトリに配置されている必要があります。
- Oracle Trace 管理ディレクトリには、REGID.DAT、PROCESS.DAT および COLLECT.DAT の各ファイルが必要です。存在していない場合は、OTRCCREF 実行ファイルを実行してファイルを作成してください。

> **注意：** Oracle9*i* では、PROCESS.DAT が FACILITY.DAT に変更されました。

- ストアド・プロシージャ・パッケージがデータベースに存在している必要があります。存在していない場合は、OTRCSVR.SQL ファイル（Oracle Trace または RDBMS 管理ディレクトリ内にあります）を実行してパッケージを作成します。
- ストアド・プロシージャに対する EXECUTE 権限がユーザーにあるかどうかをチェックします。

例

```
EXECUTE DBMS_ORACLE_TRACE_AGENT.SET_ORACLE_TRACE_IN_SESSION
(8,12,TRUE,'NEWCOLL','oracled');
```

40

DBMS_ORACLE_TRACE_USER

DBMS_ORACLE_TRACE_USER は、Oracle Trace 機能へのパブリック・アクセスをコール元ユーザーに提供します。Oracle Trace ストアド・プロシージャを使用すると、ユーザー自身のセッションまたは別のセッションに対して Oracle Trace コレクションを起動できます。

この章では、次の項目について説明します。

DBMS_ORACLE_TRACE_USER サブプログラムの要約

このパッケージに含まれているサブプログラムは、SET_ORACLE_TRACE のみです。

SET_ORACLE_TRACE プロシージャ

このプロシージャは、ユーザー自身のデータベース・セッションに関する Oracle Trace データを収集します。

構文

```
DBMS_ORACLE_TRACE_USER.SET_ORACLE_TRACE (
   on_off          IN BOOLEAN  DEFAULT false,
   collection_name IN VARCHAR2 DEFAULT '',
   facility_name   IN VARCHAR2 DEFAULT '');
```

パラメータ

表 40-1 SET_ORACLE_TRACE プロシージャのパラメータ

パラメータ	説明
on_off	TRUE または FALSE。トレースをオンまたはオフにします。
collection_name	使用する Oracle Trace コレクション名。
facility_name	使用する Oracle Trace 機能名。

例

```
EXECUTE DBMS_ORACLE_TRACE_USER.SET_ORACLE_TRACE
(TRUE,'MYCOLL','oracle');
```

41

DBMS_OUTLN

DBMS_OUTLN パッケージ（OUTLN_PKG と同義）には、ストアド・アウトラインの管理に関連するサブプログラムのための機能インタフェースが含まれています。

ストアド・アウトラインは、指定した SQL 文についての実行計画に関連するストアド・データです。これによって、オプティマイザは、最初にアウトラインとともに生成された計画と同じ実行計画を繰り返し再作成できます。アウトラインに格納されたデータの一部は、プラン・スタビリティを保つために使用するヒントで構成されています。

この章では、次の項目について説明します。

DBMS_OUTLN の要件およびセキュリティ

要件

DBMS_OUTLN には、適切なユーザーのみ使用できる管理プロシージャが含まれています。EXECUTE 権限は、DBA が明示的に定めないかぎり、一般ユーザー・コミュニティにまで拡張されません。

セキュリティ

アウトラインの管理目的に使用可能な PL/SQL ファンクションは、そのプロシージャ（またはパッケージ）に対する EXECUTE 権限があるユーザーのみが実行できます。

DBMS_OUTLN サブプログラムの要約

表 41-1 DBMS_OUTLN パッケージのサブプログラム

サブプログラム	説明
「DROP_BY_CAT プロシージャ」 41-3 ページ	指定されたカテゴリに属しているアウトラインを削除します。
「DROP_COLLISION プロシージャ」 41-3 ページ	ol$.hintcount 値が、ol$hints のそのアウトラインのヒント数と一致しないアウトラインを削除します。
「DROP_EXTRAS プロシージャ」 41-4 ページ	ヒント数に算入されない余分なヒント・タプルを削除して、インポート後のクリーンアップを行います。
「DROP_UNREFD_HINTS プロシージャ」 41-4 ページ	対応するアウトラインが OL$ 表にないヒント・タプルを削除します。
「DROP_BY_CAT プロシージャ」 41-3 ページ	SQL 文のコンパイルで適用されなくなったアウトラインを削除します。
「UPDATE_BY_CAT プロシージャ」 41-5 ページ	あるカテゴリにあるアウトラインのカテゴリを、新規のカテゴリに変更します。
「GENERATE_SIGNATURE プロシージャ」 41-6 ページ	指定した SQL テキストの署名を生成します。

DROP_BY_CAT プロシージャ

このプロシージャは、指定されたカテゴリに属しているアウトラインを削除します。

構文

```
DBMS_OUTLN.DROP_BY_CAT
   cat VARCHAR2);
```

パラメータ

表 41-2 DROP_BY_CAT プロシージャのパラメータ

パラメータ	説明
cat	削除するアウトラインのカテゴリ

使用上の注意

このプロシージャでは、1 回のコールでアウトラインのカテゴリがパージされます。

例

この例では、DEFAULT カテゴリ内にあるすべてのアウトラインを削除します。

```
DBMS_OUTLN.DROP_BY_CAT('DEFAULT');
```

DROP_COLLISION プロシージャ

このプロシージャでは、ol$.hintcount 値が、ol$hints のそのアウトラインのヒント数と一致しないアウトラインが削除されます。

構文

```
DBMS_OUTLN.DROP_COLLISION;
```

使用上の注意

アウトラインの作成または変更をインポートと同時に行うと、同時実行性の問題が発生する可能性があります。アウトラインのインポートは、元の設計に従って行う必要があるため、同時操作によってインポートの最中に変更されてメタデータに一貫性がなくなると、アウトラインは信頼性を失って削除されます。

DROP_EXTRAS プロシージャ

このプロシージャでは、ヒント数に算入されない余分なヒント・タプルを削除して、インポート後のクリーンアップを行います。

構文

```
DBMS_OUTLN.DROP_EXTRAS;
```

使用上の注意

ターゲット・データベースに同名または同一署名のアウトラインがすでに存在すると、アウトラインの OL$-tuple は拒否されます。ヒント・タプルも、既存のアウトラインのヒント数まで拒否されます。したがって、拒否されたアウトラインのヒント・タプルが既存のアウトラインのヒント・タプルよりも多い場合は、誤ったタプルが OL$HINTS 表に挿入されます。このプロシージャ（ポスト・テーブル・アクションとして自動的に実行）では、誤って挿入されたヒント・タプルが削除されます。

DROP_UNREFD_HINTS プロシージャ

このプロシージャでは、対応するアウトラインが OL$ 表にないヒント・タプルが削除されます。

構文

```
DBMS_OUTLN.DROP_UNREFD_HINTS;
```

使用上の注意

このプロシージャは、ポスト・テーブル・アクションとして自動的に実行され、対応するエントリが OL$ 表にないヒント（アウトラインの削除とインポートが同時に行われた場合に生じる可能性があります）が削除されます。

DROP_UNUSED プロシージャ

このプロシージャは、SQL 文のコンパイルで適用されなくなったアウトラインを削除します。

構文

```
DBMS_OUTLN.DROP_UNUSED;
```

使用上の注意

DROP_UNUSED は、動的 SQL 文で作成したアプリケーションによって生成されたアウトラインに対して 1 回かぎり使用できます。このプロシージャが使用されたアウトラインは使用されなくなり、貴重なディスク領域が解放されます。

UPDATE_BY_CAT プロシージャ

このプロシージャは、あるカテゴリにあるすべてのアウトラインのカテゴリを、新規のカテゴリに変更します。アウトライン内の SQL テキストがターゲット・カテゴリ内ですでにアウトラインを持っている場合は、新規カテゴリにマージされません。

構文

```
DBMS.OUTLN.UPDATE_BY_CAT (
   oldcat VARCHAR2 DEFAULT 'DEFAULT',
   newcat VARCHAR2 DEFAULT 'DEFAULT');
```

パラメータ

表 41-3 UPDATE_BY_CAT プロシージャのパラメータ

パラメータ	説明
oldcat	変更する現行のカテゴリ
newcat	アウトラインを変更するターゲット・カテゴリ

使用上の注意

アウトラインのセットが希望通りならば、アウトラインを実験的なカテゴリから本番カテゴリに移動できます。同様に、あるカテゴリから別の既存のカテゴリに、アウトラインのセットをマージすることもできます。

例

次の例では、DEFAULT カテゴリ内のすべてのアウトラインを CAT1 カテゴリに変更します。

```
DBMS_OUTLN.UPDATE_BY_CAT('DEFAULT', 'CAT1');
```

GENERATE_SIGNATURE プロシージャ

このプロシージャでは、指定した SQL テキストの署名が生成されます。

構文

```
DBMS_OUTLN.GENERATE_SIGNATURE (
   sqltxt      IN   VARCHAR2,
   signature   OUT  RAW);
```

パラメータ

表 41-4 GENERATE_SIGNATURE プロシージャのパラメータ

パラメータ	説明
sqltxt	指定した SQL
signature	生成する署名

42

DBMS_OUTLN_EDIT

DBMS_OUTLN_EDIT パッケージは、実行者権限のパッケージです。

この章では、次の項目について説明します。

DBMS_OUTLN_EDIT サブプログラムの要約

表 42-1 DBMS_OUTLN_EDIT パッケージのサブプログラム

サブプログラム	説明
「CHANGE_JOIN_POS プロシージャ」42-2 ページ	アウトライン名およびヒント番号で識別するヒントの結合位置を、newpos で指定した位置に変更します。
「CREATE_EDIT_TABLES プロシージャ」42-3 ページ	ユーザーのスキーマのコール時にアウトライン編集表を作成します。
「DROP_EDIT_TABLES プロシージャ」42-3 ページ	ユーザーのスキーマのコール時にアウトライン編集表を削除します。
「REFRESH_PRIVATE_OUTLINE プロシージャ」42-3 ページ	アウトラインのメモリー内コピーをリフレッシュして、そのデータとアウトライン・ヒントに加えた編集内容を同期化します。

CHANGE_JOIN_POS プロシージャ

このプロシージャは、アウトライン名およびヒント番号で識別するヒントの結合位置を、newpos で指定した位置に変更します。

構文

```
DBMS_OUTLN_EDIT.CHANGE_JOIN_POS (
   name     VARCHAR2
   hintno   NUMBER
   newpos   NUMBER);
```

パラメータ

表 42-2 CHANGE_JOIN_POS プロシージャのパラメータ

パラメータ	説明
name	修正するプライベート・アウトラインの名前。
hintno	修正するヒント番号。
newpos	ターゲット・ヒントの新しい結合位置。

CREATE_EDIT_TABLES プロシージャ

このプロシージャは、ユーザーのスキーマのコール時にアウトライン編集表を作成します。

構文

```
DBMS_OUTLN_EDIT.CREATE_EDIT_TABLES;
```

DROP_EDIT_TABLES プロシージャ

このプロシージャは、ユーザーのスキーマのコール時にアウトライン編集表を削除します。

構文

```
DBMS_OUTLN_EDIT.DROP_EDIT_TABLES;
```

REFRESH_PRIVATE_OUTLINE プロシージャ

このプロシージャは、アウトラインのメモリー内コピーをリフレッシュして、そのデータとアウトライン・ヒントに加えた編集内容を同期化します。

構文

```
DBMS_OUTLN_EDIT.REFRESH_PRIVATE_OUTLINE (
   name  IN  VARCHAR2);
```

パラメータ

表 42-3 REFRESH_PRIVATE_OUTLINE プロシージャのパラメータ

パラメータ	説明
name	リフレッシュするプライベート・アウトラインの名前。

43

DBMS_OUTPUT

DBMS_OUTPUT パッケージによって、ストアド・プロシージャ、パッケージおよびトリガーからメッセージを送信できます。

このパッケージにある PUT および PUT_LINE プロシージャによって、別のトリガー、プロシージャまたはパッケージが読み込めるバッファに情報を設定できます。別の PL/SQL プロシージャまたは無名ブロックでは、GET_LINE プロシージャをコールして、バッファに入れた情報を表示できます。

GET_LINE をコールしない場合、または SQL*Plus や Oracle Enterprise Manager のスクリーンにメッセージを表示しない場合、バッファに入れたメッセージは無視されます。DBMS_OUTPUT パッケージは、PL/SQL デバッグ情報の表示に特に役立ちます。

> **注意：** DBMS_OUTPUT を使用して送信するメッセージは、送信サブプログラムまたはトリガーが完了するまでは実際に送信されません。プロシージャの実行中に出力をフラッシュするメカニズムはありません。

この章では、次の項目について説明します。

- DBMS_OUTPUT のセキュリティ、エラーおよびタイプ
- DBMS_OUTPUT の使用方法
- DBMS_OUTPUT サブプログラムの要約

DBMS_OUTPUT のセキュリティ、エラーおよびタイプ

セキュリティ

このスクリプトの最後にパブリック・シノニム（DBMS_OUTPUT）が作成され、このパッケージに対する EXECUTE 許可がパブリックに付与されます。

エラー

DBMS_OUTPUT サブプログラムでアプリケーション・エラー ORA-20000 が発生すると、出力プロシージャは次のエラーを戻すことができます。

表 43-1 DBMS_OUTPUT エラー

エラー	説明
ORU-10027:	Buffer overflow
ORU-10028:	Line length overflow

タイプ

CHARARR タイプは表のタイプです。

DBMS_OUTPUT の使用方法

トリガーによってデバッグ情報を出力する場合があります。これを行うには、次のようにトリガーを指定します。

```
DBMS_OUTPUT.PUT_LINE('I got here:'||:new.col||' is the new value');
```

DBMS_OUTPUT パッケージを使用可能にした場合、この PUT_LINE はバッファに入れられ、トリガー文（トリガーを起動させる INSERT、DELETE または UPDATE が想定できます）の実行後、情報行を戻すことができます。たとえば、次のようにします。

```
BEGIN
   DBMS_OUTPUT.GET_LINE(:buffer, :status);
END;
```

これでスクリーンにバッファを表示できます。ステータスが 0（ゼロ）以外で戻るまで、GET_LINE へのコールを繰り返します。パフォーマンス向上のためには、行の配列を戻すことができる GET_LINES へのコールを使用してください。

Oracle Enterprise Manager および SQL*Plus は SET SERVEROUTPUT ON コマンドを実装して、INSERT、UPDATE、DELETE または匿名の PL/SQL コール（これらのコールのみが、トリガーまたはストアド・プロシージャを実行させることができます）の発行後に、GET_LINE へのコールを行うかどうかを判断します。

DBMS_OUTPUT サブプログラムの要約

表 43-2 DBMS_OUTPUT パッケージのサブプログラム

サブプログラム	説明
「ENABLE プロシージャ」43-3 ページ	メッセージの出力を使用可能にします。
「DISABLE プロシージャ」43-4 ページ	メッセージの出力を使用禁止にします。
「PUT および PUT_LINE プロシージャ」43-4 ページ	PUT: 行をバッファに設定します。 PUT_LINE: 行の一部をバッファに設定します。
「NEW_LINE プロシージャ」43-6 ページ	PUT を使用して作成した行を終了します。
「GET_LINE および GET_LINES プロシージャ」43-6 ページ	バッファから 1 行、または行の配列を取り出します。

ENABLE プロシージャ

このプロシージャは、PUT、PUT_LINE、NEW_LINE、GET_LINE および GET_LINES へのコールを使用可能にします。DBMS_OUTPUT パッケージが使用可能でない場合は、これらのプロシージャへのコールは無視されます。

> **注意：** Oracle Enterprise Manager または SQL*Plus の SERVEROUTPUT オプションを使用する場合は、このプロシージャをコールする必要はありません。

ENABLE へのコールが複数の場合、buffer_size は最大値が指定されます。最大サイズは 1,000,000 で、最小サイズは 2,000 です。

構文

```
DBMS_OUTPUT.ENABLE (
   buffer_size IN INTEGER DEFAULT 20000);
```

パラメータ

表 43-3 ENABLE プロシージャのパラメータ

パラメータ	説明
buffer_size	バッファに設定する情報量（バイト）。

プラグマ

```
pragma restrict_references(enable,WNDS,RNDS);
```

エラー

表 43-4 ENABLE プロシージャのエラー

エラー	説明
ORA-20000:、 ORU-10027:	バッファのオーバーフロー。バッファは <buffer_limit> バイトに制限されています。

DISABLE プロシージャ

このプロシージャは、PUT、PUT_LINE、NEW_LINE、GET_LINE および GET_LINES へのコールを使用禁止にして、残っている情報のバッファをパージします。

ENABLE と同様に、Oracle Enterprise Manager または SQL*Plus の SERVEROUTPUT オプションを使用する場合は、このプロシージャをコールする必要はありません。

構文

```
DBMS_OUTPUT.DISABLE;
```

プラグマ

```
pragma restrict_references(disable,WNDS,RNDS);
```

PUT および PUT_LINE プロシージャ

PUT_LINE をコールして情報行全体をバッファに設定するか、または PUT を複数回コールして個別に情報行を作成することができます。両方のプロシージャはオーバーロードされていて、VARCHAR2、NUMBER または DATE タイプの項目を受け入れてバッファに設定します。

すべての項目は、取り出されるときに VARCHAR2 に変換されます。NUMBER または DATE タイプの項目を渡す場合は、その項目が取り出されるとき、デフォルトの書式を使用して TO_CHAR でフォーマットされます。別の書式を使用する場合は、その項目を VARCHAR2 として渡し、明示的にフォーマットする必要があります。

PUT_LINE をコールすると、指定した項目には行端マーカーが自動的に付きます。PUT をコールして行を作成する場合は、NEW_LINE をコールして行端マーカーを追加する必要があります。GET_LINE および GET_LINES は、改行文字で終了していない行は戻しません。

行がバッファ制限を超えた場合は、エラー・メッセージが発生します。

注意： PUT または PUT_LINE で作成した出力はバッファに入れられます。この出力は、それをバッファに入れた PL/SQL プログラム・ユニットがコール元に戻るまで取り出せません。

たとえば、Oracle Enterprise Manager または SQL*Plus は、PL/SQL プログラムが完了するまで DBMS_OUTPUT メッセージを表示しません。PL/SQL プログラム内で DBMS_OUTPUT バッファをフラッシュするメカニズムはありません。たとえば、次のようになります。

```
SQL> SET SERVER OUTPUT ON
SQL> BEGIN
     2 DBMS_OUTPUT.PUT_LINE ('hello');
     3 DBMS_LOCK.SLEEP (10);
     4 END;
```

構文

```
DBMS_OUTPUT.PUT      (item IN NUMBER);
DBMS_OUTPUT.PUT      (item IN VARCHAR2);
DBMS_OUTPUT.PUT      (item IN DATE);
DBMS_OUTPUT.PUT_LINE (item IN NUMBER);
DBMS_OUTPUT.PUT_LINE (item IN VARCHAR2);
DBMS_OUTPUT.PUT_LINE (item IN DATE);
DBMS_OUTPUT.NEW_LINE;
```

パラメータ

表 43-5 PUT および PUT_LINE プロシージャのパラメータ

パラメータ	説明
item	バッファに設定する項目。

エラー

表 43-6 PUT および PUT_LINE プロシージャのエラー

エラー	説明
ORA-20000、ORU-10027:	バッファのオーバーフロー。バッファは <buf_limit> バイトに制限されています。
ORA-20000、ORU-10028:	行の長さのオーバーフロー。1 行につき 255 バイトに制限されています。

NEW_LINE プロシージャ

このプロシージャは、行端マーカーを設定します。GET_LINE は、改行で区切られた行を戻します。PUT_LINE または NEW_LINE へのすべてのコールによって、GET_LINE で戻される行が生成されます。

構文

```
DBMS_OUTPUT.NEW_LINE;
```

エラー

表 43-7 NEW_LINE プロシージャのエラー

エラー	説明
ORA-20000、ORU-10027:	バッファのオーバーフロー。バッファは <buf_limit> バイトに制限されています。
ORA-20000、ORU-10028:	行の長さのオーバーフロー。1 行につき 255 バイトに制限されています。

GET_LINE および GET_LINES プロシージャ

単一行または行の配列をバッファから取り出すことができます。バッファに入れられた単一行の情報を取り出すには、GET_LINE プロシージャをコールします。サーバーへのコール数を減らすには、GET_LINES プロシージャをコールして、バッファから行の配列を取り出します。

Oracle Enterprise Manager または SQL*Plus を使用している場合は、特別の SET SERVEROUTPUT ON コマンドを使用して、この情報を自動的に表示できます。

GET_LINE または GET_LINES をコールしてから、次に PUT、PUT_LINE または NEW_LINE をコールする前に取り出されなかった行は、次のメッセージとの混同を避けるために廃棄されます。

構文

```
DBMS_OUTPUT.GET_LINE (
   line    OUT VARCHAR2,
   status  OUT INTEGER);
```

パラメータ

表 43-8 GET_LINE プロシージャのパラメータ

パラメータ	説明
line	最後の改行文字を除いて、バッファに入れられた単一行の情報を戻します。最大長は 255 バイトです。
status	コールが正常に完了すると、ステータス 0（ゼロ）が戻ります。 バッファにこれ以上行がない場合は、ステータス 1 が戻ります。

構文

```
DBMS_OUTPUT.GET_LINES (
   lines       OUT  CHARARR,
   numlines    IN OUT  INTEGER);
```

CHARARR は、VARCHAR2(255) の表です。

パラメータ

表 43-9 GET_LINES プロシージャのパラメータ

パラメータ	説明
lines	バッファに入れられた情報行の配列を戻します。 配列内の各行の最大長は 255 バイトです。
numlines	バッファから取り出す行数。 プロシージャは、指定の行数を取り出した後、実際に取り出した行数を戻します。この数が要求した行数より少ない場合は、バッファにそれ以上行がない場合です。

例 1: ストアド・プロシージャとトリガーのデバック

DBMS_OUTPUT パッケージは、一般的に、ストアド・プロシージャおよびトリガーをデバッグするために使用されます。また、43-9 ページの「例 2: オブジェクトに関する情報の取得」で示すように、このパッケージを使用すると、オブジェクトの情報を取り出し、その出力をフォーマットできます。

このファンクションは、従業員表を問い合せて、指定部門の給与合計を戻します。このファンクションには、次のように PUT_LINE プロシージャへのコールがいくつか含まれます。

```
CREATE FUNCTION dept_salary (dnum NUMBER) RETURN NUMBER IS
   CURSOR emp_cursor IS
      SELECT sal, comm FROM emp WHERE deptno = dnum;
   total_wages    NUMBER(11, 2) := 0;
   counter        NUMBER(10) := 1;
BEGIN

   FOR emp_record IN emp_cursor LOOP
      emp_record.comm := NVL(emp_record.comm, 0);
      total_wages := total_wages + emp_record.sal
         + emp_record.comm;
      DBMS_OUTPUT.PUT_LINE('Loop number = ' || counter ||
         '; Wages = '|| TO_CHAR(total_wages));    /* Debug line */
      counter := counter + 1; /* Increment debug counter */
   END LOOP;
   /* Debug line */
   DBMS_OUTPUT.PUT_LINE('Total wages = ' ||
     TO_CHAR(total_wages));
   RETURN total_wages;

END dept_salary;
```

EMP 表には、次の行が含まれているとします。

```
EMPNO          SAL     COMM     DEPT
-----        ------- -------- -------
1002           1500      500      20
1203           1000               30
1289           1000               10
1347           1000      250      20
```

ユーザーが、Oracle Enterprise Manager の SQL ワークシート入力ペインで次の文を実行するとします。

```
SET SERVEROUTPUT ON
VARIABLE salary NUMBER;
EXECUTE :salary := dept_salary(20);
```

出力ペインには、次の情報が表示されます。

```
Loop number = 1; Wages = 2000
Loop number = 2; Wages = 3250
Total wages = 3250


PL/SQL procedure successfully executed.
```

例 2: オブジェクトに関する情報の取得

この例では、ユーザーはすでに EXPLAIN PLAN コマンドを使用して、ある文の実行計画に関する情報を取り出し、その情報を PLAN_TABLE に格納してあります。また、ユーザーはこの文に文 ID を割り当ててあります。EXPLAIN_OUT プロシージャの例では、表から情報を取り出し、出力をネスト形式でフォーマットして、SQL 文を処理するステップの順序を厳密に記述しています。

```
 /*****************************************************************/
/* Create EXPLAIN_OUT procedure. User must pass STATEMENT_ID to */
/* to procedure, to uniquely identify statement.                 */
/*****************************************************************/
CREATE OR REPLACE PROCEDURE explain_out
   (statement_id IN VARCHAR2) AS

   -- Retrieve information from PLAN_TABLE into cursor EXPLAIN_ROWS.

   CURSOR explain_rows IS
      SELECT level, id, position, operation, options,
         object_name
      FROM plan_table
      WHERE statement_id = explain_out.statement_id
      CONNECT BY PRIOR id = parent_id
         AND statement_id = explain_out.statement_id
      START WITH id = 0
       ORDER BY id;

BEGIN


   -- Loop through information retrieved from PLAN_TABLE:

   FOR line IN explain_rows LOOP
```

```
      -- At start of output, include heading with estimated cost.

      IF line.id = 0 THEN
         DBMS_OUTPUT.PUT_LINE ('Plan for statement '
            || statement_id
            || ', estimated cost = ' || line.position);
      END IF;

      -- Output formatted information. LEVEL determines indention level.

      DBMS_OUTPUT.PUT_LINE (lpad(' ',2*(line.level-1)) ||
         line.operation || ' ' || line.options || ' ' ||
         line.object_name);
   END LOOP;

END;
```

関連項目： 第 95 章「UTL_FILE」

44

DBMS_PCLXUTIL

DBMS_PCLXUTIL パッケージは、パーティション単位でローカル索引を作成するためのパーティション内並列性を提供します。

> **関連項目：** パーティションおよび索引に関しては、いくつかのルールがあります。詳細は、『Oracle9*i* データベース概要』および『Oracle9*i* データベース管理者ガイド』を参照してください。

各パーティションには 1 つのスレーブ・プロセスしか使用できません。DBMS_PCLXUTIL は、ローカル索引の作成でパーティション数によって並列度が制限されることを回避します。

DBMS_PCLXUTIL は DBMS_JOB パッケージを使用して、パーティション表のローカル索引を作成するための高い並列度を提供します。これは、バックグラウンド・プロセスを使用した（DBMS_JOB を使用）非同期のパーティション間並列性と、パラレル問合せスレーブ処理を使用したパーティション内並列性の組合せで達成されます。

DBMS_PCLXUTIL は、レンジ・パーティション化およびレンジーハッシュ・コンポジット・パーティション化の両方で機能します。

注意： レンジ・パーティション化の最小互換モードは 8.0 で、レンジーハッシュ・コンポジット・パーティション化の最小互換モードは 8*i* です。

この章では、次の項目について説明します。

DBMS_PCLXUTIL の使用方法

DBMS_PCLXUTIL パッケージは、次の DBA タスク中に使用できます。

1. ローカル索引の作成

   BUILD_PART_INDEX プロシージャは、ローカル索引のディクショナリ情報がすでに存在していることを前提としています。これは、UNUSABLE オプションを持つ CREATE INDEX SQL コマンドを発行して行います。

   ```
   CREATE INDEX <idx_name> on <tab_name>(...) local(...) unusable;
   ```

   これにより、索引作成で時間がかかるディクショナリ・エントリの作成を、索引自体を作成せずに行うことができます。そして、BUILD_PART_INDEX プロシージャを起動して、指定の並列度でローカル索引の同時作成を行います。

   ```
   EXECUTE dbms_pclxutil.build_part_index(4,4,<tab_name>,<idx_name>,FALSE);
   ```

   コンポジット・パーティションについて、プロシージャは、そのコンポジット表のすべてのサブパーティションに対するローカル索引を自動的に作成します。

2. ローカル索引のメンテナンス

   目的のパーティションを使用可能または使用禁止にマークし、BUILD_PART_INDEX プロシージャもローカル索引の選択的な再作成を可能にします。force_opt パラメータは、これを上書きし、すべてのパーティションに対するローカル索引の作成方法を提供します。

   ```
   ALTER INDEX <idx_name> local(...) unusable;
   ```

   目的の（サブ）パーティション（UNUSABLE とマークされています）のみを再作成します。

   ```
   EXECUTE dbms_pclxutil.build_part_index(4,4,<tab_name>,<idx_name>,FALSE);
   ```

   force_opt = TRUE を使用して、すべての（サブ）パーティションを再作成します。

   ```
   EXECUTE dbms_pclxutil.build_part_index(4,4,<tab_name>,<idx_name>,TRUE);
   ```

   進捗レポートが作成され、プログラムの終了時に出力がスクリーンに表示されます（DBMS_OUTPUT パッケージは、最初にメッセージをバッファに書き込み、そのバッファをプログラムの終了時のみスクリーンにフラッシュするためです）。

制限事項

DBMS_PCLXUTIL パッケージは DBMS_JOB パッケージを使用しているため、DBMS_JOB に関する次の制限事項があることに注意してください。

- job_queue_processes 初期化パラメータの適切な値を決定する必要があります。BUILD_PART_INDEX() をコールする前にジョブ・プロセスが開始されていない場合、パッケージは正しく機能しません。バックグラウンド・プロセスは、次の init.ora パラメータで指定されます。

  ```
  job_queue_processes=n   #the number of background processes = n
  ```

- キューに入る同時ジョブの数に上限があり、この上限は、SNP[0..9] および SNP[A..Z] のマークが付けられたバックグラウンド・プロセスの数で決まります。上限は 36 です。

  関連項目：『Oracle9*i* データベース管理者ガイド』

- 障害の状態はトレース・ファイルにのみレポートされ（DBMS_JOB の制限事項）、ユーザーへの対話的なフィードバックはできません。このパッケージは、失敗時にメッセージを出力して未完了のジョブをキューから削除し、snp*.trc トレース・ファイルを調べるようにユーザーに要求します。

DBMS_PCLXUTTL サブプログラムの要約

DBMS_PCLXUTIL に含まれているプロシージャは、BUILD_PART_INDEX のみです。

BUILD_PART_INDEX プロシージャ

構文

```
DBMS_PCLXUTIL.build_part_index (
   procs_per_job   IN NUMBER   DEFAULT 1,
   tab_name        IN VARCHAR2 DEFAULT NULL,
   idx_name        IN VARCHAR2 DEFAULT NULL,
   force_opt       IN BOOLEAN  DEFAULT FALSE);
```

パラメータ

表 44-1 BUILD_PART_INDEX プロシージャのパラメータ

パラメータ	説明
procs_per_job	各ローカル索引作成に利用するパラレル問合せスレーブ数（1 <= procs_per_job <= max_slaves）。
tab_name	パーティション表の名前（表が存在しない、またはパーティション化されていない場合は例外が発生します）。
idx_name	ローカル索引に指定する名前（ローカル索引が tab_name 表に作成されていない場合は例外が発生します）。
force_opt	TRUE の場合は、すべてのパーティション索引の再作成を強制します。それ以外の場合は、UNUSABLE にマークされたパーティションのみを再作成します。

例

PROJ001 と PROJ002 の 2 つのパーティション、およびローカル索引 IDX を持つ PROJECT 表が作成されるとします。

BUILD_PART_INDEX(2,4,'PROJECT','IDX',TRUE) プロシージャへのコールによって、次の出力が作成されます。

```
SQLPLUS> EXECUTE dbms_pclxutil.build_part_index(2,4,'PROJECT','IDX',TRUE);
Statement processed.
INFO: Job #21 created for partition PROJ002 with 4 slaves
INFO: Job #22 created for partition PROJ001 with 4 slaves
```

45

DBMS_PIPE

DBMS_PIPE パッケージによって、同じインスタンスにある複数のセッションの通信を行います。Oracle パイプは、UNIX で使用するパイプと概念は似ていますが、オペレーティング・システムのパイプ・メカニズムを使用して実装されていません。

Oracle パイプを介して送信する情報は、システム・グローバル領域（SGA）にバッファリングされます。パイプ内のすべての情報は、インスタンスがシャットダウンすると失われます。

セキュリティ要件に応じて、パブリック・パイプまたはプライベート・パイプのいずれかを使用できます。

注意： パイプはトランザクションから独立しています。トランザクション制御に影響を与える可能性がある場合は、注意してパイプを使用してください。

この章では、次の項目について説明します。

- パブリック・パイプ、プライベート・パイプおよびパイプの使用
- セキュリティ、定数およびエラー
- DBMS_PIPE サブプログラムの要約

パブリック・パイプ、プライベート・パイプおよびパイプの使用

パブリック・パイプ

パブリック・パイプは、暗黙的または明示的に作成できます。暗黙的なパブリック・パイプは、そのパイプの最初の参照時に自動的に作成され、データがなくなると消去されます。パイプ記述子は SGA に格納されるため、空のパイプがキャッシュから削除されるまで、スペースを使用するオーバーヘッドが若干あります。

明示的なパブリック・パイプを作成するためには、private フラグに FALSE を設定した CREATE_PIPE ファンクションをコールします。明示的に作成したパイプは、REMOVE_PIPE ファンクションをコールして割当て解除する必要があります。

パブリック・パイプのドメインは、明示的または暗黙的にパイプが作成されたスキーマです。

パイプへの書込みおよび読込み

各パブリック・パイプは非同期に動作します。スキーマ・ユーザーが DBMS_PIPE パッケージに対する EXECUTE 許可を持ち、パブリック・パイプ名を知っている場合に限り、任意の数のスキーマ・ユーザーがパブリック・パイプへの書込みを行うことができます。ただし、バッファに入っている情報を 1 人のユーザーが読み込むと、その情報はバッファから消去されるため、同じパイプの他のユーザーは読み込むことができません。

送信側セッションでは、PACK_MESSAGE プロシージャに 1 つ以上コールを行ってメッセージを作成します。このプロシージャは、セッションのローカル・メッセージ・バッファにメッセージを追加します。このバッファ内の情報は、メッセージ送信に使用するパイプ名を指定した SEND_MESSAGE ファンクションをコールして送信されます。SEND_MESSAGE をコールすると、ローカル・バッファにスタックされたすべてのメッセージが送信されます。

メッセージを受信するプロセスは、メッセージを受信するパイプ名を指定した RECEIVE_MESSAGE ファンクションをコールします。次に、UNPACK_MESSAGE プロシージャをコールして、メッセージ内の各項目にアクセスします。

プライベート・パイプ

CREATE_PIPE ファンクションをコールして、プライベート・パイプを明示的に作成します。プライベート・パイプは、一度作成されると、REMOVE_PIPE ファンクションをコールして明示的に割当て解除するまで共有メモリーに存続します。プライベート・パイプは、データベース・インスタンスがシャットダウンしたときにも割当て解除されます。

メモリーに暗黙的なパイプが存在し、そのパイプと作成しようとするプライベート・パイプが同じ名前の場合、プライベート・パイプは作成できません。この場合は、CREATE_PIPE からエラーが戻されます。

プライベート・パイプへのアクセスは、次のように限定されます。

- パイプ作成者と同じユーザー ID で実行中のセッション
- パイプ作成者と同じユーザー ID 権限ドメインで実行中のストアド・サブプログラム
- SYSDBA として接続したユーザー

これ以外のユーザーがパイプ上のメッセージの送受信やパイプの削除を試みると、即時にエラーが発生します。別のユーザーが同じ名前のパイプを作成しようとしても、エラーが発生します。

パブリック・パイプと同様に、SEND_MESSAGE をコールする前に PACK_MESSAGE へのコールを使用して、最初にメッセージを作成する必要があります。同様に、RECEIVE_MESSAGE をコールしてメッセージを取り出してから、UNPACK_MESSAGE をコールしてメッセージ内の項目にアクセスする必要があります。

パイプの使用方法

パイプ機能には、次のような潜在的な応用例があります。

- 外部サービス・インタフェース : RDBMS 外部にあるユーザー記述のサービスと通信することができます。この通信は、共有サーバー・プロセスで効果的に行うことができるため、サービスの複数インスタンスが同時に実行されます。さらに、サービスは非同期で使用できます。サービスのリクエスタは、待機応答をブロックする必要がなく、（タイムアウトに関係なく）後でチェックできます。サービスは、Oracle がサポートする任意の 3GL 言語で記述できます。
- 独立したトランザクション : パイプは、操作を独立したトランザクション（トリガーで検出するセキュリティ違反のロギングなど）で実行できる個別のセッションと通信できます。
- アラート（トランザクション以外）: ポーリングするための待機プロセスを要求せずに、別のプロセスを転送できます。アプリケーションにアラートを通知する AFTER 行または AFTER 文トリガーがある場合、アプリケーションは、このアラートをデータが変更された可能性を示すためのアラートとして処理します。アプリケーションはそのデータを読み込み、現行の値を取得します。これは AFTER トリガーなので、アプリケーションは、SELECT FOR UPDATE を行って正しいデータを読み込んだことを確認します。
- デバッグ : トリガーおよびストアド・プロシージャは、デバッグ情報をパイプに送信できます。別のセッションは、パイプからの読込みを続行し、その内容をスクリーンに表示したりファイルに書き込むことができます。
- コンセントレータ : これは、少数のネットワーク接続に対して多数のユーザーがいる場合にユーザーを多重化したり、複数のユーザー・トランザクションを 1 つの DBMS トランザクションに集中化する場合のパフォーマンス改善に役立ちます。

セキュリティ、定数およびエラー

セキュリティ

セキュリティは、DBMS_PIPE パッケージの実行権限の付与を使用して達成できます。これは、CREATE_PIPE ファンクションの private パラメータを使用してパイプを作成し、特定の機能、特定のユーザーやロールへのパイプ名を表示するのみのカバー・パッケージを記述することによって行われます。

定数

```
maxwait   constant integer := 86400000; /* 1000 days */
```

メッセージの送受信を行うための最大待機時間です。

エラー

DBMS_PIPE パッケージ・サブプログラムは、次のエラーを戻すことができます。

表 45-1 DBMS_PIPE エラー

エラー	説明
ORA-23321:	パイプ名には NULL を指定できません。このエラーは、CREATE_PIPE ファンクションまたはパイプ名をパラメータとして使用しているサブプログラムによって戻されます。
ORA-23322:	パイプへのアクセス権限が不十分です。このエラーは、パラメータ・リストにあるプライベート・パイプを参照するサブプログラムによって戻されます。

DBMS_PIPE サブプログラムの要約

表 45-2 DBMS_PIPE パッケージのサブプログラム

サブプログラム	説明
「CREATE_PIPE ファンクション」45-6 ページ	パイプを作成します（プライベート・パイプの場合は必須です）。
「PACK_MESSAGE プロシージャ」45-8 ページ	ローカル・バッファにメッセージを作成します。
「SEND_MESSAGE ファンクション」45-9 ページ	名前付きパイプにメッセージを送信します。名前付きパイプが存在しない場合は、パブリック・パイプが暗黙的に作成されます。
「RECEIVE_MESSAGE ファンクション」45-12 ページ	名前付きパイプからローカル・バッファにメッセージをコピーします。
「NEXT_ITEM_TYPE ファンクション」45-13 ページ	バッファにある次の項目のデータ・タイプを戻します。
「UNPACK_MESSAGE プロシージャ」45-14 ページ	バッファにある次の項目にアクセスします。
「REMOVE_PIPE ファンクション」45-15 ページ	名前付きパイプを削除します。
「PURGE プロシージャ」45-16 ページ	名前付きパイプの内容をパージします。
「RESET_BUFFER プロシージャ」45-17 ページ	ローカル・バッファの内容をパージします。
「UNIQUE_SESSION_NAME ファンクション」45-18 ページ	一意のセッション名を戻します。

CREATE_PIPE ファンクション

このファンクションは、パブリック・パイプまたはプライベート・パイプを明示的に作成します。private フラグが TRUE の場合、パイプ作成者がそのプライベート・パイプの所有者として割り当てられます。

明示的に作成されたパイプは、REMOVE_PIPE をコールするか、またはインスタンスをシャットダウンすることによってのみ削除できます。

構文

```
DBMS_PIPE.CREATE_PIPE (
   pipename     IN VARCHAR2,
   maxpipesize  IN INTEGER DEFAULT 8192,
   private      IN BOOLEAN DEFAULT TRUE)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(create_pipe,WNDS,RNDS);
```

パラメータ

表 45-3 CREATE_PIPE ファンクションのパラメータ

パラメータ	説明
pipename	作成するパイプの名前。 SEND_MESSAGE および RECEIVE_MESSAGE をコールするときは、この名前を使用する必要があります。この名前は、インスタンス間で一意である必要があります。 **注意：** ORA$ で始まるパイプ名を使用しないでください。これは、オラクル社が提供する製品用に予約されています。パイプ名は 128 バイト以下で指定し、大 / 小文字区別があります。現時点では、名前に NLS 文字を含めることはできません。

表 45-3 CREATE_PIPE ファンクションのパラメータ（続き）

パラメータ	説明
maxpipesize	パイプの最大サイズ（バイト単位）。 パイプ上のすべてのメッセージの合計サイズは、この数を超えることはできません。この最大値を超えると、そのメッセージはブロックされます。デフォルトの maxpipesize は 8192 バイトです。 パイプの maxpipesize はパイプ特性の一部となり、パイプが存続するかぎり有効です。これより大きいパイプ・サイズで SEND_MESSAGE をコールすると、maxpipesize の値が大きくなります。これより小さいサイズでコールした場合は、より大きい既存の値を使用します。
private	デフォルトの TRUE を使用して、プライベート・パイプを作成します。 パブリック・パイプは、SEND_MESSAGE をコールして、暗黙的に作成できます。

戻り値

表 45-4 CREATE_PIPE ファンクションの戻り値

戻り値	説明
0	成功。 パイプが存在し、パイプを作成するユーザーにそのパイプの使用が認可されている場合、Oracle は 0（ゼロ）を戻して成功であることを示し、パイプ内のデータはそのまま残ります。 SYSDBA/SYSOPER として接続したユーザーがパイプを再作成した場合、Oracle はステータス 0（ゼロ）を戻しますが、そのパイプの所有者は変更されないままです。
ORA-23322	命名競合のために失敗。 同じ名前のパイプが存在し、別のユーザーがそのパイプを作成した場合、Oracle ではエラー ORA-23322 を通知して命名競合であることを示しています。

例外

表 45-5 CREATE_PIPE ファンクションの例外

例外	説明
パイプ名が NULL	アクセス権エラー。同名のパイプが存在するため使用できません。

PACK_MESSAGE プロシージャ

このプロシージャは、ユーザーのメッセージをローカル・メッセージ・バッファに作成します。

メッセージを送信するためには、最初に PACK_MESSAGE を 1 回以上コールします。次に、SEND_MESSAGE をコールして、ローカル・バッファにあるメッセージを名前付きパイプに送信します。

PACK_MESSAGE プロシージャは、VARCHAR2 タイプ、NUMBER タイプまたは DATE タイプの項目を受け入れるためにオーバーロードされています。バッファ内の各項目には、データ・バイトの他に、項目のタイプを示すための 1 バイト、および長さを格納するための 2 バイトが必要です。さらに、メッセージを終了するために 1 バイト必要です。VARCHAR 以外のすべてのタイプのオーバーヘッドは 4 バイトです。

Oracle8 では、キャラクタ・セット ID（2 バイト）およびキャラクタ・セット・フォーム（1 バイト）が各データ項目に格納されています。したがって、Oracle8 を使用するときのオーバーヘッドは 7 バイトです。

SEND_MESSAGE をコールしてメッセージを送信するときは、メッセージを送信するパイプの名前を示す必要があります。パイプがすでに存在する場合は、そのパイプにアクセスするための十分な権限が必要です。パイプが存在しない場合は、自動的に作成されます。

構文

```
DBMS_PIPE.PACK_MESSAGE       (item  IN  VARCHAR2);
DBMS_PIPE.PACK_MESSAGE       (item  IN  NCHAR);
DBMS_PIPE.PACK_MESSAGE       (item  IN  NUMBER);
DBMS_PIPE.PACK_MESSAGE       (item  IN  DATE);
DBMS_PIPE.PACK_MESSAGE_RAW   (item  IN  RAW);
DBMS_PIPE.PACK_MESSAGE_ROWID (item  IN  ROWID);
```

注意： PACK_MESSAGE プロシージャは、VARCHAR2、NCHAR、NUMBER または DATE タイプの項目を受け入れるためにオーバーロードされています。さらに、RAW および ROWID 項目をパックするための 2 つのプロシージャがあります。

プラグマ

```
pragma restrict_references(pack_message,WNDS,RNDS);
pragma restrict_references(pack_message_raw,WNDS,RNDS);
pragma restrict_references(pack_message_rowid,WNDS,RNDS);
```

パラメータ

表 45-6 PACK_MESSAGE プロシージャのパラメータ

パラメータ	説明
item	ローカル・メッセージ・バッファにパックする項目。

例外

メッセージ・バッファ（現在は 4096 バイト）がオーバーフローした場合は、ORA-06558 が発生します。バッファ内の各項目には、実際のデータに加えて、タイプを示すために 1 バイト、長さを示すために 2 バイトが必要です。さらに、メッセージを終了するために 1 バイトが必要です。

SEND_MESSAGE ファンクション

このファンクションは、メッセージを名前付きパイプに送信します。

PACK_MESSAGE へのコールで入力されたメッセージは、ローカル・メッセージ・バッファに格納されます。パイプは CREATE_PIPE を使用して明示的に作成されます。そうでない場合は、暗黙的に作成されます。

構文

```
DBMS_PIPE.SEND_MESSAGE (
    pipename     IN VARCHAR2,
    timeout      IN INTEGER DEFAULT MAXWAIT,
    maxpipesize  IN INTEGER DEFAULT 8192)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(send_message,WNDS,RNDS);
```

パラメータ

表 45-7 SEND_MESSAGE ファンクションのパラメータ

パラメータ	説明
pipename	メッセージを設定するパイプの名前。 明示的なパイプを使用している場合、この名前は、CREATE_PIPE をコールしたときに指定した名前です。 **注意 :** 'ORA$' で始まるパイプ名を使用しないでください。これらの名前は、オラクル社が提供する製品用に予約されています。パイプ名は 128 バイト以下で指定し、大 / 小文字区別があります。現時点では、名前に NLS 文字を含めることはできません。
timeout	パイプにメッセージを設定する間の待機時間（秒単位）。 デフォルト値は定数 MAXWAIT で、86400000（1000 日）に定義されています。
maxpipesize	パイプの最大サイズ（バイト単位）。 パイプ上のすべてのメッセージの合計サイズは、この数を超えることはできません。この最大値を超えると、そのメッセージはブロックされます。デフォルトは 8192 バイトです。 パイプの maxpipesize はパイプ特性の一部となり、パイプが存続するかぎり有効です。これより大きいパイプ・サイズで SEND_MESSAGE をコールすると、maxpipesize の値が大きくなります。これより小さいサイズでコールした場合は、より大きい既存の値を使用します。 SEND_MESSAGE プロシージャの一部として maxpipesize を指定すると、別のコールでパイプをオープンする必要がなくなります。明示的にパイプを作成した場合は、オプションの maxpipesize パラメータを使用して、作成したパイプのサイズを上書きできます。

戻り値

表 45-8 SEND_MESSAGE ファンクションの戻り値

戻り値	説明
0	成功。 パイプが存在し、パイプを作成するユーザーにそのパイプの使用が認可されている場合、Oracle は 0（ゼロ）を戻して成功であることを示し、パイプ内のデータはそのまま残ります。 SYSDBS/SYSOPER として接続したユーザーがパイプを再作成した場合、Oracle はステータス 0（ゼロ）を戻しますが、そのパイプの所有者は変更されないままです。
1	タイムアウト。 このプロシージャは、パイプでロックが取得できないか、またはパイプがいっぱいで使用できない理由でタイムアウトできます。暗黙的に作成されたパイプが空の場合、そのパイプは削除されます。
3	割込みが発生しました。 暗黙的に作成されたパイプが空の場合、そのパイプは削除されます。
ORA-23322	権限が不十分です。 同じ名前のパイプが存在し、別のユーザーがそのパイプを作成した場合、Oracle ではエラー ORA-23322 を通知して命名競合であることを示しています。

例外

表 45-9 SEND_MESSAGE ファンクションの例外

例外	説明
パイプ名が NULL	アクセス権エラー。パイプに書込みを行うための権限が不十分です。パイプはプライベートで、別のユーザーが所有しています。

RECEIVE_MESSAGE ファンクション

このファンクションは、メッセージをローカル・メッセージ・バッファにコピーします。

パイプからメッセージを受信するには、最初に RECEIVE_MESSAGE をコールします。メッセージを受信すると、メッセージはパイプから削除されます。したがって、メッセージは 1 回しか受信できません。暗黙的に作成されたパイプは、最後のレコードがそのパイプから削除された後に削除されます。

RECEIVE_MESSAGE のコール時に指定したパイプがすでに存在しない場合、Oracle はパイプを暗黙的に作成してメッセージの受信を待ちます。メッセージが指定したタイムアウト時間内に着信しなかった場合、そのコールは戻され、パイプは削除されます。

メッセージの受信後、1 つ以上の UNPACK_MESSAGE をコールして、メッセージ内の個別の項目にアクセスする必要があります。UNPACK_MESSAGE は、DATE、NUMBER、VARCHAR2 タイプの項目をアンパックするためにオーバーロードされています。さらに、RAW および ROWID 項目をアンパックするために 2 つのプロシージャがあります。アンパックするデータのタイプが不明の場合は、NEXT_ITEM_TYPE をコールして、バッファ内にある次の項目のタイプを判別します。

構文

```
DBMS_PIPE.RECEIVE_MESSAGE (
   pipename     IN VARCHAR2,
   timeout      IN INTEGER      DEFAULT maxwait)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(receive_message,WNDS,RNDS);
```

パラメータ

表 45-10 RECEIVE_MESSAGE ファンクションのパラメータ

パラメータ	説明
pipename	メッセージを受信するパイプ名。 ORA$ で始まる名前は、オラクル社で使用するために予約されています。
timeout	メッセージを待つ時間（秒単位）。 デフォルト値は定数 MAXWAIT で、86400000（1000 日）に定義されています。タイムアウトを 0（ゼロ）に指定すると、ブロックされずに読み込むことができます。

戻り値

表 45-11 RECEIVE_MESSAGE ファンクションの戻り値

戻り値	説明
0	成功。
1	タイムアウト。暗黙的に作成されたパイプが空の場合、そのパイプは削除されます。
2	パイプにあるレコードがバッファに対して大きすぎます（これは起こり得ない値です)。
3	割込みが発生しました。
ORA-23322	パイプから読み込むための十分な権限がユーザーにありません。

例外

表 45-12 RECEIVE_MESSAGE ファンクションの例外

例外	説明
パイプ名が NULL	アクセス権エラー。パイプからレコードを削除するための権限が不十分です。パイプは別のユーザーが所有しています。

NEXT_ITEM_TYPE ファンクション

このファンクションは、ローカル・メッセージ・バッファにある次の項目のデータ・タイプを判別します。

RECEIVE_MESSAGE をコールして、ローカル・バッファにパイプ情報を設定した後、NEXT_ITEM_TYPE をコールします。

構文

```
DBMS_PIPE.NEXT_ITEM_TYPE
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(next_item_type,WNDS,RNDS);
```

戻り値

表 45-13 NEXT_ITEM_TYPE ファンクションの戻り値

戻り値	説明
0	次の項目がありません。
6	NUMBER
9	VARCHAR2
11	ROWID
12	DATE
23	RAW

UNPACK_MESSAGE プロシージャ

このプロシージャは、バッファから項目を取り出します。

RECEIVE_MESSAGE をコールして、ローカル・バッファにパイプ情報を設定した後、UNPACK_MESSAGE をコールします。

構文

```
DBMS_PIPE.UNPACK_MESSAGE       (item  OUT VARCHAR2);
DBMS_PIPE.UNPACK_MESSAGE       (item  OUT NCHAR);
DBMS_PIPE.UNPACK_MESSAGE       (item  OUT NUMBER);
DBMS_PIPE.UNPACK_MESSAGE       (item  OUT DATE);
DBMS_PIPE.UNPACK_MESSAGE_RAW   (item  OUT RAW);
DBMS_PIPE.UNPACK_MESSAGE_ROWID (item  OUT ROWID);
```

注意： UNPACK_MESSAGE プロシージャは、VARCHAR2、NCHAR、NUMBER または DATE タイプの項目を戻すためにオーバーロードされています。さらに、RAW および ROWID 項目をアンパックするための 2 つのプロシージャがあります。

プラグマ

```
pragma restrict_references(unpack_message,WNDS,RNDS);
pragma restrict_references(unpack_message_raw,WNDS,RNDS);
pragma restrict_references(unpack_message_rowid,WNDS,RNDS);
```

パラメータ

表 45-14 UNPACK_MESSAGE プロシージャのパラメータ

パラメータ	説明
item	アンパックされた次の項目をローカル・メッセージ・バッファから受け取るための引数。

例外

バッファに次の項目がない場合、または項目が要求されたタイプでない場合は、ORA-06556 または 06559 が生成されます。

REMOVE_PIPE ファンクション

このファンクションは、明示的に作成されたパイプを削除します。

SEND_MESSAGE で暗黙的に作成されたパイプは、空になると自動的に削除されます。ただし、CREATE_PIPE で明示的に作成されたパイプは、REMOVE_PIPE をコールするか、またはインスタンスをシャットダウンした場合にのみ削除されます。パイプ内の未使用のレコードは、パイプが削除される前にすべて削除されます。

これは、暗黙的に作成されたパイプで PURGE をコールするのに似ています。

構文

```
DBMS_PIPE.REMOVE_PIPE (
   pipename  IN  VARCHAR2)
RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(remove_pipe,WNDS,RNDS);
```

パラメータ

表 45-15 REMOVE_PIPE ファンクションのパラメータ

パラメータ	説明
pipename	削除するパイプ名。

戻り値

表 45-16 REMOVE_PIPE ファンクションの戻り値

戻り値	説明
0	成功。 パイプが存在しない場合、またはパイプがすでに存在し、パイプを削除しようとするユーザーに削除が認可されている場合、Oracle は 0（ゼロ）を戻して成功であることを示し、パイプ内に残っているデータが削除されます。
ORA-23322	権限が不十分です。 パイプは存在するが、ユーザーにそのパイプへのアクセス権がない場合、Oracle はエラー ORA-23322 を通知して権限が不十分であることを示します。

例外

表 45-17 REMOVE_PIPE ファンクションの例外

例外	説明
パイプ名が NULL	アクセス権エラー。パイプを削除する権限が不十分です。パイプは作成されていて、別のユーザーが所有しています。

PURGE プロシージャ

このプロシージャは、名前付きパイプの内容を空にします。

暗黙的に作成された空のパイプは、LRU アルゴリズムに従って、共有グローバル領域から削除されます。したがって、PURGE をコールすると、暗黙的に作成したパイプに関連するメモリーを空にできます。

PURGE プロシージャは RECEIVE_MESSAGE をコールするため、メッセージがパイプからパージされるとき、ローカル・バッファはそのメッセージで上書きされる場合があります。また、不十分なアクセス権でパイプをパージしようとすると、ORA-23322 エラー（不十分な権限）を受け取ります。

構文

```
DBMS_PIPE.PURGE (
   pipename  IN  VARCHAR2);
```

プラグマ

```
pragma restrict_references(purge,WNDS,RNDS);
```

パラメータ

表 45-18 PURGE プロシージャのパラメータ

パラメータ	説明
pipename	すべてのメッセージを削除するパイプの名前。 メッセージが廃棄されると、ローカル・バッファがそのメッセージで上書きされる場合があります。パイプ名は 128 バイト以下で指定し、大 / 小文字区別があります。

例外

パイプを別のユーザーが所有している場合は、アクセス権エラーが発生します。

RESET_BUFFER プロシージャ

このプロシージャは、PACK_MESSAGE および UNPACK_MESSAGE の位置設定インジケータを 0（ゼロ）にリセットします。

すべてのパイプが 1 つのバッファを共有しているため、新規パイプを使用する前にバッファをリセットすると効果的です。これにより、初めてメッセージをパイプに送信するとき、バッファ内に残っていた期限切れのメッセージを誤って送信することがなくなります。

構文

```
DBMS_PIPE.RESET_BUFFER;
```

プラグマ

```
pragma restrict_references(reset_buffer,WNDS,RNDS);
```

UNIQUE_SESSION_NAME ファンクション

このファンクションは、現在データベースに接続しているすべてのセッション間での一意の名前を受け取ります。

同じセッションからこのファンクションを複数回コールしても、常に同じ値が戻されます。このファンクションは、SEND_MESSAGE および RECEIVE_MESSAGE のコールに PIPENAME パラメータを提供するのに役立ちます。

構文

```
DBMS_PIPE.UNIQUE_SESSION_NAME
  RETURN VARCHAR2;
```

プラグマ

```
pragma restrict_references(unique_session_name,WNDS,RNDS,WNPS);
```

戻り値

このファンクションは、一意の名前を戻します。戻される名前は、最大 30 バイトです。

例 1: デバッグ

この例は、デバッグ情報をパイプに設定するために、PL/SQL プログラムがコールできるプロシージャを示しています。

```
CREATE OR REPLACE PROCEDURE debug (msg VARCHAR2) AS
    status  NUMBER;
BEGIN
  DBMS_PIPE.PACK_MESSAGE(LENGTH(msg));
  DBMS_PIPE.PACK_MESSAGE(msg);
  status := DBMS_PIPE.SEND_MESSAGE('plsql_debug');
  IF status != 0 THEN
    raise_application_error(-20099, 'Debug error');
  END IF;
END debug;
```

次の Pro*C コードでは、「例 1: デバッグ」の PLSQL_DEBUG パイプから受信したメッセージが表示されます。Pro*C セッションが別のウィンドウで実行されている場合、このセッションは、別のセッションで実行中の PL/SQL プログラムからデバッグ・プロシージャに送信されるメッセージの表示に使用できます。

```
#include <stdio.h>
#include <string.h>

EXEC SQL BEGIN DECLARE SECTION;
   VARCHAR username[20];
   int     status;
   int     msg_length;
   char    retval[2000];
EXEC SQL END DECLARE SECTION;

EXEC SQL INCLUDE SQLCA;

void sql_error();

main()
{

-- Prepare username:
   strcpy(username.arr, "SCOTT/TIGER");
   username.len = strlen(username.arr);

   EXEC SQL WHENEVER SQLERROR DO sql_error();
   EXEC SQL CONNECT :username;

   printf("connected\n");

-- Start an endless loop to look for and print messages on the pipe:
   FOR (;;)
   {
      EXEC SQL EXECUTE
         DECLARE
            len INTEGER;
            typ INTEGER;
            sta INTEGER;
            chr VARCHAR2(2000);
         BEGIN
            chr := '';
            sta := dbms_pipe.receive_message('plsql_debug');
            IF sta = 0 THEN
               DBMS_PIPE.UNPACK_MESSAGE(len);
               DBMS_PIPE.UNPACK_MESSAGE(chr);
            END IF;
```

```
            :status := sta;
            :retval := chr;
            IF len IS NOT NULL THEN
               :msg_length := len;
            ELSE
               :msg_length := 2000;
            END IF;
         END;
      END-EXEC;
      IF (status == 0)
         printf("\n%.*s\n", msg_length, retval);
      ELSE
         printf("abnormal status, value is %d\n", status);
   }
}

void sql_error()
{
   char msg[1024];
   int rlen, len;
   len = sizeof(msg);
   sqlglm(msg, &len, &rlen);
   printf("ORACLE ERROR\n");
   printf("%.*s\n", rlen, msg);
   exit(1);
}
```

例 2: システム・コマンドの実行

この例は、PL/SQL と Pro*C コードによって、PL/SQL ストアド・プロシージャ（または無名ブロック）が PL/SQL プロシージャをコールし、コマンドをリスニングしている Pro*C プログラムにパイプを介してそのコマンドを送信する処理を示します。

Pro*C プログラムはスリープして、名前付きパイプにメッセージが着信するのを待ちます。メッセージが着信すると、Pro*C プログラムはそれを処理し、*system()* コールから UNIX コマンドを実行したり、埋込み SQL を使用して SQL コマンドを実行するなど、要求された処理を実行します。

DAEMON.SQL は、PL/SQL パッケージ用のソース・コードです。このパッケージには、DBMS_PIPE パッケージを使用して Pro*C デーモンに対してメッセージを送受信するプロシージャが含まれています。完全なハンドシェイクが使用されていることに注意してください。このデーモンは、常にメッセージをパッケージに返信します（STOP コマンドの場合は除きます）。これによって、PL/SQL プロシージャは Pro*C デーモンが動作していることを確認できるため、デーモンのこの機能は重要です。

SQL*Plus または Oracle Enterprise Manager を使用して、無名 PL/SQL ブロックから DAEMON パッケージ・プロシージャをコールできます。たとえば、次のようにします。

```
SQLPLUS> variable rv number
SQLPLUS> execute :rv := DAEMON.EXECUTE_SYSTEM('ls -la');
```

これにより、UNIX システムでは、Pro*C デーモンによってコマンド *system*("*ls -la*") が実行されます。

最初にデーモンを実行する必要があることに留意してください。デーモンは、バックグラウンドで実行したり、デーモンをコールした SQL*Plus または Oracle Enterprise Manager セッションの近くにある別のウィンドウで実行できます。

また、DAEMON.SQL は、DBMS_OUTPUT パッケージを使用して結果を表示します。この例を動作させるには、このパッケージに対する EXECUTE 権限が必要です。

DAEMON.SQL 例。次の例は、PL/SQL DAEMON パッケージのコード例です。

```
CREATE OR REPLACE PACKAGE daemon AS
  FUNCTION execute_sql(command VARCHAR2,
                       timeout NUMBER DEFAULT 10)
    RETURN NUMBER;

  FUNCTION execute_system(command VARCHAR2,
                          timeout NUMBER DEFAULT 10)
    RETURN NUMBER;

  PROCEDURE stop(timeout NUMBER DEFAULT 10);
END daemon;
/
CREATE OR REPLACE PACKAGE BODY daemon AS

  FUNCTION execute_system(command VARCHAR2,
                          timeout NUMBER DEFAULT 10)
  RETURN NUMBER IS

    status       NUMBER;
    result       VARCHAR2(20);
    command_code NUMBER;
    pipe_name    VARCHAR2(30);
  BEGIN
    pipe_name := DBMS_PIPE.UNIQUE_SESSION_NAME;

    DBMS_PIPE.PACK_MESSAGE('SYSTEM');
    DBMS_PIPE.PACK_MESSAGE(pipe_name);
    DBMS_PIPE.PACK_MESSAGE(command);
    status := DBMS_PIPE.SEND_MESSAGE('daemon', timeout);
    IF status <> 0 THEN
      RAISE_APPLICATION_ERROR(-20010,
        'Execute_system: Error while sending.  Status = ' ||
         status);
```

```
  END IF;

  status := DBMS_PIPE.RECEIVE_MESSAGE(pipe_name, timeout);
  IF status <> 0 THEN
    RAISE_APPLICATION_ERROR(-20011,
      'Execute_system: Error while receiving.
       Status = ' || status);
  END IF;

  DBMS_PIPE.UNPACK_MESSAGE(result);
  IF result <> 'done' THEN
    RAISE_APPLICATION_ERROR(-20012,
      'Execute_system: Done not received.');
  END IF;

  DBMS_PIPE.UNPACK_MESSAGE(command_code);
  DBMS_OUTPUT.PUT_LINE('System command executed.  result = ' ||
                       command_code);
  RETURN command_code;
END execute_system;

FUNCTION execute_sql(command VARCHAR2,
                     timeout NUMBER DEFAULT 10)
RETURN NUMBER IS

  status       NUMBER;
  result       VARCHAR2(20);
  command_code NUMBER;
  pipe_name    VARCHAR2(30);

BEGIN
  pipe_name := DBMS_PIPE.UNIQUE_SESSION_NAME;

  DBMS_PIPE.PACK_MESSAGE('SQL');
  DBMS_PIPE.PACK_MESSAGE(pipe_name);
  DBMS_PIPE.PACK_MESSAGE(command);
  status := DBMS_PIPE.SEND_MESSAGE('daemon', timeout);
  IF status <> 0 THEN
    RAISE_APPLICATION_ERROR(-20020,
      'Execute_sql: Error while sending.  Status = ' || status);
  END IF;

  status := DBMS_PIPE.RECEIVE_MESSAGE(pipe_name, timeout);

  IF status <> 0 THEN
    RAISE_APPLICATION_ERROR(-20021,
      'execute_sql: Error while receiving.
```

```
          Status = ' || status);
    END IF;

    DBMS_PIPE.UNPACK_MESSAGE(result);
    IF result <> 'done' THEN
      RAISE_APPLICATION_ERROR(-20022,
        'execute_sql: done not received.');
    END IF;

    DBMS_PIPE.UNPACK_MESSAGE(command_code);
    DBMS_OUTPUT.PUT_LINE
        ('SQL command executed.  sqlcode = ' || command_code);
    RETURN command_code;
  END execute_sql;

  PROCEDURE stop(timeout NUMBER DEFAULT 10) IS
    status NUMBER;
  BEGIN
    DBMS_PIPE.PACK_MESSAGE('STOP');
    status := DBMS_PIPE.SEND_MESSAGE('daemon', timeout);
    IF status <> 0 THEN
      RAISE_APPLICATION_ERROR(-20030,
        'stop: error while sending.  status = ' || status);
    END IF;
  END stop;
END daemon;
```

daemon.pc 例。次の例は、Pro*C デーモンのコード例です。バージョン 1.5.x 以降の Pro*C プリコンパイラを使用して、このコードをプリコンパイルする必要があります。この例には埋込み PL/SQL コードが含まれているため、USERID および SQLCHECK オプションも指定する必要があります。

> **注意：** PL/SQL ブロックで VARCHAR 出力ホスト変数を使用するには、そのブロックの入力前に長さの要素を初期化する必要があります。

```
proc iname=daemon userid=scott/tiger sqlcheck=semantics
```

次に、通常の方法で、C コンパイルしてリンクします。

```
#include <stdio.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;
```

```
EXEC SQL BEGIN DECLARE SECTION;
  char *uid = "scott/tiger";
  int status;
  VARCHAR command[20];
  VARCHAR value[2000];
  VARCHAR return_name[30];
EXEC SQL END DECLARE SECTION;

void
connect_error()
{
  char msg_buffer[512];
  int msg_length;
  int buffer_size = 512;

  EXEC SQL WHENEVER SQLERROR CONTINUE;
  sqlglm(msg_buffer, &buffer_size, &msg_length);
  printf("Daemon error while connecting:\n");
  printf("%.*s\n", msg_length, msg_buffer);
  printf("Daemon quitting.\n");
  exit(1);
}

void
sql_error()
{
  char msg_buffer[512];
  int msg_length;
  int buffer_size = 512;

  EXEC SQL WHENEVER SQLERROR CONTINUE;
  sqlglm(msg_buffer, &buffer_size, &msg_length);
  printf("Daemon error while executing:\n");
  printf("%.*s\n", msg_length, msg_buffer);
  printf("Daemon continuing.\n");
}
main()
{
command.len = 20; /*initialize length components*/
value.len = 2000;
return_name.len  = 30;
  EXEC SQL WHENEVER SQLERROR DO connect_error();
  EXEC SQL CONNECT :uid;
  printf("Daemon connected.\n");

  EXEC SQL WHENEVER SQLERROR DO sql_error();
  printf("Daemon waiting...\n");
```

```
while (1) {
  EXEC SQL EXECUTE
    BEGIN
      :status := DBMS_PIPE.RECEIVE_MESSAGE('daemon');
      IF :status = 0 THEN
        DBMS_PIPE.UNPACK_MESSAGE(:command);
      END IF;
    END;
  END-EXEC;
  IF (status == 0)
  {
    command.arr[command.len] = '\0';
    IF (!strcmp((char *) command.arr, "STOP"))
    {
      printf("Daemon exiting.\n");
      break;
    }

    ELSE IF (!strcmp((char *) command.arr, "SYSTEM"))
    {
      EXEC SQL EXECUTE
        BEGIN
          DBMS_PIPE.UNPACK_MESSAGE(:return_name);
          DBMS_PIPE.UNPACK_MESSAGE(:value);
        END;
      END-EXEC;
      value.arr[value.len] = '\0';
      printf("Will execute system command '%s'\n", value.arr);

      status = system(value.arr);
      EXEC SQL EXECUTE
        BEGIN
          DBMS_PIPE.PACK_MESSAGE('done');
          DBMS_PIPE.PACK_MESSAGE(:status);
          :status := DBMS_PIPE.SEND_MESSAGE(:return_name);
        END;
      END-EXEC;

      IF (status)
      {
        printf
         ("Daemon error while responding to system command.");
        printf("  status: %d\n", status);
      }
    }
    ELSE IF (!strcmp((char *) command.arr, "SQL")) {
      EXEC SQL EXECUTE
```

```
        BEGIN
          DBMS_PIPE.UNPACK_MESSAGE(:return_name);
          DBMS_PIPE.UNPACK_MESSAGE(:value);
        END;
      END-EXEC;
      value.arr[value.len] = '\0';
      printf("Will execute sql command '%s'\n", value.arr);

      EXEC SQL WHENEVER SQLERROR CONTINUE;
      EXEC SQL EXECUTE IMMEDIATE :value;
      status = sqlca.sqlcode;

      EXEC SQL WHENEVER SQLERROR DO sql_error();
      EXEC SQL EXECUTE
        BEGIN
          DBMS_PIPE.PACK_MESSAGE('done');
          DBMS_PIPE.PACK_MESSAGE(:status);
          :status := DBMS_PIPE.SEND_MESSAGE(:return_name);
        END;
      END-EXEC;

      IF (status)
      {
        printf("Daemon error while responding to sql command.");
        printf("  status: %d\n", status);
      }
    }
    ELSE
    {
      printf
        ("Daemon error: invalid command '%s' received.\n",
          command.arr);
    }
  }
  ELSE
  {
    printf("Daemon error while waiting for signal.");
    printf("  status = %d\n", status);
  }
}
EXEC SQL COMMIT WORK RELEASE;
exit(0);
```

例 3: 外部サービス・インタフェース

ユーザー作成の 3GL コードを、OCI またはプリコンパイラ・プログラムに設定します。プログラムは、データベースに接続して PL/SQL コードを実行し、パイプから要求を読み込み、結果を計算します。次に、PL/SQL コードを実行して、パイプ上の結果をリクエスタに返信します。

次に、株式サービス要求の例を示します。すべてのサービス要求についてパイプに渡す引数の推奨する順序を次に示します。

```
protocol_version      VARCHAR2        - '1', 10 bytes or less
returnpipe            VARCHAR2        - 30 bytes or less
service               VARCHAR2        - 30 bytes or less
arg1                  VARCHAR2/NUMBER/DATE
   ...
argn                  VARCHAR2/NUMBER/DATE
```

結果を戻すための推奨する書式を次に示します。

```
success               VARCHAR2        - 'SUCCESS' if OK,
                                        otherwise error message
arg1                  VARCHAR2/NUMBER/DATE
   ...
argn                  VARCHAR2/NUMBER/DATE
```

OCI または PRO*（疑似コードで）を使用して、株価要求サーバーを次のように設定します。

```
<loop forever>
  BEGIN dbms_stock_server.get_request(:stocksymbol); END;
  <figure out price based on stocksymbol (probably from some radio
       signal), set error if can't find such a stock>
  BEGIN dbms_stock_server.return_price(:error, :price); END;
```

クライアントは次のように設定します。

```
BEGIN :price := stock_request('YOURCOMPANY'); end;
```

前述の株価要求サーバーでコールしたストアド・プロシージャ dbms_stock_server は、次のように設定します。

```
CREATE OR REPLACE PACKAGE dbms_stock_server IS
  PROCEDURE get_request(symbol OUT VARCHAR2);
  PROCEDURE return_price(errormsg IN VARCHAR2, price IN VARCHAR2);
END;

CREATE OR REPLACE PACKAGE BODY dbms_stock_server IS
  returnpipe    VARCHAR2(30);

  PROCEDURE returnerror(reason VARCHAR2) IS
```

```
  s INTEGER;
BEGIN
  dbms_pipe.pack_message(reason);
  s := dbms_pipe.send_message(returnpipe);
  IF s <> 0 THEN
    raise_application_error(-20000, 'Error:' || to_char(s) ||
      ' sending on pipe');
  END IF;
END;

PROCEDURE get_request(symbol OUT VARCHAR2) IS
  protocol_version VARCHAR2(10);
  s                INTEGER;
  service          VARCHAR2(30);
BEGIN
  s := dbms_pipe.receive_message('stock_service');
  IF s <> 0 THEN
    raise_application_error(-20000, 'Error:' || to_char(s) ||
      'reading pipe');
  END IF;
  dbms_pipe.unpack_message(protocol_version);
  IF protocol_version <> '1' THEN
    raise_application_error(-20000, 'Bad protocol: ' ||
      protocol_version);
  END IF;
  dbms_pipe.unpack_message(returnpipe);
  dbms_pipe.unpack_message(service);
  IF service != 'getprice' THEN
    returnerror('Service ' || service || ' not supported');
  END IF;
  dbms_pipe.unpack_message(symbol);
END;

PROCEDURE return_price(errormsg in VARCHAR2, price in VARCHAR2) IS
  s INTEGER;
BEGIN
  IF errormsg is NULL THEN
    dbms_pipe.pack_message('SUCCESS');
    dbms_pipe.pack_message(price);
  ELSE
    dbms_pipe.pack_message(errormsg);
  END IF;
  s := dbms_pipe.send_message(returnpipe);
  IF s <> 0 THEN
    raise_application_error(-20000, 'Error:'||to_char(s)||
      ' sending on pipe');
  END IF;
```

```
  END;
END;
```

クライアントがコールするプロシージャは、次のように設定します。

```
CREATE OR REPLACE FUNCTION stock_request (symbol VARCHAR2)
    RETURN VARCHAR2 IS
  s        INTEGER;
  price    VARCHAR2(20);
  errormsg VARCHAR2(512);
BEGIN
  dbms_pipe.pack_message('1');  -- protocol version
  dbms_pipe.pack_message(dbms_pipe.unique_session_name); -- return pipe
  dbms_pipe.pack_message('getprice');
  dbms_pipe.pack_message(symbol);
  s := dbms_pipe.send_message('stock_service');
  IF s <> 0 THEN
    raise_application_error(-20000, 'Error:'||to_char(s)||
      ' sending on pipe');
  END IF;
  s := dbms_pipe.receive_message(dbms_pipe.unique_session_name);
  IF s <> 0 THEN
    raise_application_error(-20000, 'Error:'||to_char(s)||
      ' receiving on pipe');
  END IF;
  dbms_pipe.unpack_message(errormsg);
  IF errormsg <> 'SUCCESS' THEN
    raise_application_error(-20000, errormsg);
  END IF;
  dbms_pipe.unpack_message(price);
  RETURN price;
END;
```

一般的に、株式サービス・アプリケーション・サーバーに対してのみ dbms_stock_service の実行権限を付与し、このサービスを利用できるユーザーに対してのみ stock_request の実行権限を付与します。

関連項目： 第 2 章「DBMS_ALERT」

46

DBMS_PROFILER

Oracle9*i* は、既存の PL/SQL アプリケーションをプロファイルし、パフォーマンスのボトルネックを識別するためのプロファイラ API を提供します。収集されたプロファイラ（パフォーマンス）データは、パフォーマンスを向上させるため、または PL/SQL アプリケーションのコード・カバレージを決定するために使用できます。アプリケーション開発者は、コード・カバレージ・データを使用して、増分テストに集中できます。

プロファイラ API は、PL/SQL パッケージの DBMS_PROFILER として実装され、PL/SQL プロファイラ・データを収集し、継続的に格納するためのサービスを提供します。

> **注意：** DBMS_PROFILER は、NATIVE モードでコンパイルされたプログラム・ユニットを、ユーザーには CREATE 権限がないかのように処理します。つまり、ユーザーが出力を取得することはありません。

この章では、次の項目について説明します。

DBMS_PROFILER の使用方法

アプリケーションのパフォーマンス向上は、繰り返し行うプロセスです。この反復プロセスには、次の手順が伴います。

1. プロファイラ・データの収集を可能にし、1 つ以上のベンチマーク・テストを使用してアプリケーションを実行します。
2. プロファイラ・データを分析し、パフォーマンス上の問題を識別します。
3. 問題を解決します。

PL/SQL プロファイラは、「実行」という概念を使用してこのプロセスをサポートします。実行には、プロファイラ・データの収集を可能にし、アプリケーションをベンチマーク・テストを介して実行することが含まれます。実行の開始および終了は、START_PROFILER ファンクションおよび STOP_PROFILER ファンクションをコールして制御できます。

一般的な実行には、次の処理が含まれます。

- 実行でプロファイラ・データ収集を開始します。
- プロファイラ・データおよびコード・カバレージ・データが必要な PL/SQL コードを実行します。
- プロファイラ・データ収集を停止します。その実行に対して収集したデータは、データベース表に書き込まれます。

> **注意：** 収集したプロファイラ・データは、ユーザーの切断時に自動的には格納されません。セッションの終了時にデータを格納するためには、FLUSH_DATA または STOP_PROFILER ファンクションの明示的なコールを発行する必要があります。データ収集を停止すると、収集されたデータが格納されます。

アプリケーションの実行時、プロファイラ・データは、実行期間中存続するメモリー・データ構造に収集されます。FLUSH_DATA ファンクションを実行の途中でコールして、増分データを取得し、割り当てられているファイラ・データ構造用のメモリーを解放できます。

収集されたデータをフラッシュすると、その内容がデータベース表に格納されます。この表は、プロファイラ・ユーザーのスキーマにすでに存在している必要があります。PROFTAB.SQL スクリプトは、プロファイラ・データを継続的に格納するための表や他のデータ構造を作成します。

PROFTAB.SQL を実行すると、現行の表が削除されることに注意してください。PROFTAB.SQL スクリプトは、RDBMS/ADMIN ディレクトリにあります。PL/SQL ユニットの最初の実行など、一部の PL/SQL 操作には、実行中の PL/SQL ユニットにバイト・コードをロードするカタログ表への I/O が含まれる場合があります。また、パッケージ・プロシージャまたはファンクションが最初にコールされたとき、パッケージ初期化コードの実行に時間がかかる場合があります。

この時間的なオーバーヘッドを避けるためには、プロファイラ・データの収集前に、データベースのウォーム・アップを行います。ウォーム・アップを行うには、プロファイラ・データを収集せずにアプリケーションを 1 回実行します。

システム全体のプロファイル

システムの全ユーザーをプロファイルできます。たとえば、使用中か否かに関係なく、あるパッケージの全ユーザーをプロファイルできます。このような場合、SYSADMIN は、変更した PROFLOAD.SQL スクリプトで次の内容を実行します。

- プロファイラ表および順序を作成します。
- これらの表および順序の SELECT/INSERT/UPDATE 権限をすべてのユーザーに付与します。
- 表および順序のパブリック・シノニムを定義します。

注意： 表の実際のフィールドは変更しないでください。

関連項目： 46-9 ページ「FLUSH_DATA ファンクション」

要件

DBMS_PROFILER は、SYS としてインストールする必要があります。

PROFLOAD.SQL スクリプトを使用して、PL/SQL プロファイラ・パッケージをロードします。

収集されたデータ

プローブ・プロファイラ API を使用すると、セッションで実行されるすべての名前付きライブラリ・ユニットについて、プロファイル情報を生成できます。プロファイラは、PL/SQL 仮想マシン・レベルで情報を収集します。その情報には、各行の合計実行回数、その行の実行に要した合計時間、およびその行の特定の実行に要した最小時間と最大時間が含まれています。

注意： データが収集された PL/SQL ユニットに関するコード・カバレージ値を推論することは可能です。

プロファイル情報は、データベース表に格納されています。このため、データに対して問合せが可能です。ユーザーは、カスタマイズ可能なレポート（サマリー・レポート、最新行、コード・カバレージ・データなど）を作成できます。また、データの分析も可能です。

PROFTAB.SQL

PROFTAB.SQL スクリプトは、表 46-1、表 46-2 および表 46-3 にリストした列、データ・タイプおよび定義を持つ表を作成します。

表 46-1 表 PLSQL_PROFILER_RUNS の列

列	データ・タイプ	定義
runid	NUMBER	主キー。plsql_profiler_runnumber で作成される一意の実行識別子。
related_run	NUMBER	（クライアントとサーバーの相関関係に）関連する実行の実行 ID。
run_owner	VARCHAR2（32）	実行を開始したユーザー。
run_date	DATE	実行の開始時間。
run_comment	VARCHAR2（2047）	この実行に関してユーザーが指定したコメント。
run_total_time	NUMBER	この実行の経過時間（ナノ秒）。
run_system_info	VARCHAR2（2047）	現在は使用されていません。
run_comment1	VARCHAR2（2047）	追加のコメント。
spare1	VARCHAR2（256）	未使用。

表 46-2 表 PLSQL_PROFILER_UNITS の列

列	データ・タイプ	定義
runid	NUMBER	主キー。plsql_profiler_runs を参照します。
unit_number	NUMBER	主キー。内部的に生成されたライブラリ・ユニット番号。
unit_type	VARCHAR2（32）	ライブラリ・ユニットのタイプ。
unit_owner	VARCHAR2（32）	ライブラリ・ユニットの所有者名。
unit_name	VARCHAR2（32）	ライブラリ・ユニットにおけるライブラリ・ユニット名のタイムスタンプ。
unit_timestamp	DATE	ユニットに複数実行の間に発生した変更を検出するために、将来使用される予定です。
total_time	NUMBER	このユニットで経過した合計時間（ナノ秒）。プロファイラはこのフィールドを設定しませんが、分析ツールで利用するために用意されています。

表 46-2 表 PLSQL_PROFILER_UNITS の列（続き）

列	データ・タイプ	定義
spare1	NUMBER	未使用。
spare2	NUMBER	未使用。

表 46-3 表 PLSQL_PROFILER_DATA の列

列	データ・タイプ	定義
runid	NUMBER	主キー。一意の（生成された）実行識別子。
unit_number	NUMBER	主キー。内部的に生成されたライブラリ・ユニット番号。
line#	NUMBER	主キー。ユニット内の NULL 以外の行番号。
total_occur	NUMBER	行が実行された回数。
total_time	NUMBER	行の実行に要した合計時間（ナノ秒）。
min_time	NUMBER	この行の最小実行時間（ナノ秒）。
max_time	NUMBER	この行の最大実行時間（ナノ秒）。
spare1	NUMBER	未使用。
spare2	NUMBER	未使用。
spare3	NUMBER	未使用。
spare4	NUMBER	未使用。

Oracle8 には、PL/SQL デモ用スクリプトに、文脈依存のレポート・ライター（profrep.sql）のサンプルが準備されています。

セキュリティ

プロファイラは、ユーザーに CREATE 権限があるユニットのデータのみ収集します。EXECUTE ONLY アクセス権限が付与されているユニットを、パッケージを使用してプロファイルすることはできません。一般的に、ユニットをデバッグできるユーザーは、そのユニットをプロファイルできます。ただし、ユニットは DEBUG でコンパイルされているかどうかに関係なく、プロファイルできます。プロファイル対象のモジュールは、DEBUG でコンパイルすることをお薦めします。これによって、データベース内のユニットに関する追加情報が提供されます。

例外生成の 2 つの方法

このパッケージの各ルーチンには、エラーのレポート方法が 2 通りあります。

- 成功または失敗をステータス値として戻すファンクション。例外を呼び出すことはありません。
- 成功した場合は正常に戻り、失敗した場合は例外を呼び出すプロシージャ。

いずれの場合も、ファンクションおよびプロシージャのパラメータは同じです。エラーのレポート方法のみ異なります。エラーがある場合、ファンクションが戻すエラー・コードと、プロシージャが呼び出す例外は対応しています。

冗長性を避けるために、次の項では、ファンクションのフォームに関する詳細のみ提供します。

例外

表 46-4 DBMS_PROFILER の例外

例外	説明
version_mismatch	error_version に相当します。
profiler_error	「error_param」または「error_io」のいずれかに相当します。

エラー・コード

ファンクションからの戻り値が 0（ゼロ）の場合は、正常終了を示します。0（ゼロ）以外の戻り値は、エラー状態を示します。潜在的なエラーは、次のとおりです。

- サブプログラムは不正なパラメータでコールされました。

```
error_param constant binary_integer := 1;
```

- データ・フラッシュ操作に失敗しました。プロファイラ表が作成済でアクセス可能か、および十分な領域があるかどうかをチェックしてください。

```
error_io    constant binary_integer := 2;
```

- パッケージとデータベースの実装に不一致があります。不適切なバージョンの DBMS_PROFILER パッケージがインストールされ、このバージョンのプロファイラ・パッケージがそのデータベース・バージョンで動作できない場合に、このエラーが戻されます。リカバリするための唯一の方法は、適切なバージョンのパッケージをインストールすることです。

```
error_version constant binary_integer := -1;
```

DBMS_PROFILER サブプログラムの要約

表 46-5 DBMS_PROFILER サブプログラム

サブプログラム	説明
「START_PROFILER ファンクション」46-8 ページ	ユーザーのセッションでプロファイラ・データ収集を開始します。
「STOP_PROFILER ファンクション」46-9 ページ	ユーザーのセッションでプロファイラ・データ収集を停止します。
「FLUSH_DATA ファンクション」46-9 ページ	ユーザーのセッションでプロファイラ・データ収集をフラッシュします。
「PAUSE_PROFILER ファンクション」46-9 ページ	プロファイラ・データ収集を一時停止します。
「RESUME_PROFILER ファンクション」46-9 ページ	プロファイラ・データ収集を再開します。
「GET_VERSION プロシージャ」46-10 ページ	この API のバージョンを取得します。
「INTERNAL_VERSION_CHECK ファンクション」46-10 ページ	このバージョンの DBMS_PROFILER パッケージが、データベース内の実装で動作可能であることを検証します。

START_PROFILER ファンクション

このファンクションは、ユーザーのセッションでプロファイラ・データ収集を開始します。

構文

START_PROFILER ファンクションのフォームは 2 種類、オーバーロードされています。1 つはコール結果のみではなく、実行を開始した実行番号も戻します。もう 1 つは実行番号を戻しません。最初のフォームは、プロファイラを制御する GUI ベースのツールで使用することを目的としています。

1 番目のフォームは次のとおりです。

```
DBMS_PROFILER.START_PROFILER(run_comment IN VARCHAR2 := sysdate,
run_comment1 IN VARCHAR2 :='',
run_number OUT BINARY_INTEGER)
RETURN BINARY_INTEGER;
```

2 番目のフォームは次のとおりです。

```
DBMS_PROFILER.START_PROFILER(run_comment IN VARCHAR2 := sysdate,
run_comment1 IN VARCHAR2 :='')
RETURN BINARY_INTEGER;
```

パラメータ

表 46-6 START_PROFILER ファンクションのパラメータ

パラメータ	説明
run_comment	実行するプロファイラごとに、コメントを指定できます。たとえば、コメントによって、データ収集で使用したベンチマーク・テストの名前およびバージョンを提供できます。
run_number	実行番号を格納します。ユーザーは実行データを格納しておき、後で再コールできます。
run_comment1	実行に関するわかりやすいコメントを記述できます。

STOP_PROFILER ファンクション

このファンクションは、ユーザーのセッションでプロファイラ・データ収集を停止します。

このファンクションには、セッションでそれまでに収集したデータをフラッシュする副次効果があり、これが実行の終了を示します。

構文

```
DBMS_PROFILER.STOP_PROFILER
  RETURN BINARY_INTEGER;
```

FLUSH_DATA ファンクション

このファンクションは、ユーザーのセッションで収集したプロファイラ・データをフラッシュします。データは、以前から存在しているデータベース表にフラッシュされます。

> **注意：** PROFTAB.SQL スクリプトを使用して、プロファイラ・データを継続的に格納するための表や他のデータ構造を作成します。

構文

```
DBMS_PROFILER.FLUSH_DATA
  RETURN BINARY_INTEGER;
```

PAUSE_PROFILER ファンクション

このファンクションは、プロファイラ・データ収集を一時停止します。

RESUME_PROFILER ファンクション

このファンクションは、プロファイラ・データ収集を再開します。

GET_VERSION プロシージャ

このプロシージャは、この API のバージョンを取得します。

構文

```
DBMS_PROFILER.GET_VERSION (
   major  OUT BINARY_INTEGER,
   minor  OUT BINARY_INTEGER);
```

パラメータ

表 46-7 GET_VERSION プロシージャのパラメータ

パラメータ	説明
major	DBMS_PROFILER のバージョン番号。
minor	DBMS_PROFILER のリリース番号。

INTERNAL_VERSION_CHECK ファンクション

このファンクションは、このバージョンの DBMS_PROFILER パッケージが、データベース内の実装で動作可能であること検証します。

構文

```
DBMS_PROFILER.INTERNAL_VERSION_CHECK
  RETURN BINARY_INTEGER;
```

47

DBMS_PROPAGATION_ADM

DBMS_PROPAGATION_ADM パッケージは、ソース・キューから宛先キューへの伝播を構成するための管理プロシージャを提供します。

この章では、次の項目について説明します。

関連項目： Streams 環境での伝播に関する詳細は、『Oracle9*i* Streams』を参照してください。

DBMS_PROPAGATION_ADM サブプログラムの要約

表 47-1 DBMS_PROPAGATION_ADM サブプログラム

サブプログラム	説明
「ALTER_PROPAGATION プロシージャ」 47-3 ページ	伝播ジョブのルール・セットを追加、変更または削除します。
「CREATE_PROPAGATION プロシージャ」 47-4 ページ	伝播ジョブを作成し、その伝播ジョブのソース・キュー、宛先キューおよびルール・セットを指定します。
「DROP_PROPAGATION プロシージャ」 47-7 ページ	伝播ジョブを削除します。

注意： 特に指定がないかぎり、すべてのプロシージャがコミットされます。

ALTER_PROPAGATION プロシージャ

伝播ジョブのルール・セットを追加、変更または削除します。

関連項目： ルールおよびルール・セットの詳細は、『Oracle9*i* Streams』および第 64 章「DBMS_RULE_ADM」を参照してください。

構文

```
DBMS_PROPAGATION_ADM.ALTER_PROPAGATION(
   propagation_name   IN VARCHAR2,
   rule_set_name      IN VARCHAR2);
```

パラメータ

表 47-2 ALTER_PROPAGATION プロシージャのパラメータ

パラメータ	説明
propagation_name	変更する伝播ジョブの名前。既存の伝播ジョブ名を指定する必要があります。
rule_set_name	この伝播ジョブの伝播ルールが含まれているルール・セットの名前。伝播ジョブにルール・セットを使用する場合は、[*schema_name*.]*rule_set_name* の形式で既存のルール・セットを指定する必要があります。たとえば、ルール・セットに hr スキーマの prop_rules という名前を指定するには、hr.prop_rules と入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 指定したルール・セットが存在しない場合は、エラーが戻ります。ルール・セットを作成し、そのルール・セットにルールを追加するには、DBMS_RULE_ADM パッケージを使用します。 このパラメータに NULL を指定すると、伝播ジョブによって、そのキューのすべての LCR およびユーザー・メッセージが伝播されます。

CREATE_PROPAGATION プロシージャ

伝播ジョブを作成し、その伝播ジョブのソース・キュー、宛先キューおよびルール・セットを指定します。伝播ジョブによって、ローカル・ソース・キューのイベントが宛先キューに伝播されます。宛先キューは、ソース・キューと同じデータベース内にある場合と、ない場合があります。このプロシージャを実行するユーザーは、伝播ジョブの所有者となります。

また、このプロシージャは伝播を開始し、伝播ジョブのデフォルト・スケジュールを確立します。デフォルトのスケジュールには、次のプロパティがあります。

- 開始時刻は SYSDATE()。
- 継続時間は NULL。無限であることを意味します。
- 次回実行は NULL。現行の継続時間が終了次第、伝播が再起動されることを意味します。
- 待機時間は 5 秒。同じ宛先キューへの伝播が不要なメッセージのキューにエンキューされた後、メッセージが宛先キューへの伝播を待機する時間です。

伝播ジョブの作成後は、DBMS_AQADM パッケージの次のプロシージャを使用してジョブを管理できます。

- 伝播ジョブのデフォルト・スケジュールを変更するには、ALTER_PROPAGATION_SCHEDULE プロシージャを使用します。
- 伝播を停止するには、DISABLE_PROPAGATION_SCHEDULE プロシージャを使用し、queue_name パラメータにソース・キューを、destination パラメータにデータベース・リンクを指定します。
- 伝播を再起動するには、ENABLE_PROPAGATION_SCHEDULE プロシージャを使用し、queue_name パラメータにソース・キューを、destination パラメータにデータベース・リンクを指定します。エラーのために伝播ジョブが自動的に使用禁止になっている場合は、伝播の再起動が必要です。

このような変更は、ソース・キューのデータベース・リンクに対する伝播ジョブすべてに影響します。

ソース・キューを所有するユーザーが、イベントを伝播するユーザーです。このユーザーには、イベントを伝播するための権限が必要です。必要な権限は、次のとおりです。

- 伝播ジョブが使用するルール・セットの実行権限
- ルール・セットで使用される変換ファンクションすべての実行権限
- 宛先キューが同じデータベース上にある場合は、宛先キューでのエンキュー権限

伝播ジョブによって、イベントがリモート・データベースの宛先キューに伝播される場合、ソース・キューの所有者は、伝播ジョブのデータベース・リンクを使用できる必要があり、リモート・データベースでデータベース・リンクに接続しているユーザーには、宛先キューでのエンキュー権限が必要です。

注意：

- 現在は、データベース・リンクによってイベントが複数の宛先キューに伝播される場合でも、単一の伝播ジョブによって、特定のデータベース・リンクを使用するイベントすべてが伝播されます。
- ソース・キューの所有者は伝播を実行しますが、伝播ジョブの所有者はジョブを作成したユーザーです。これらのユーザーは、同一のユーザーまたは別のユーザーの場合があります。

関連項目：

- 第 64 章「DBMS_RULE_ADM」
- 『Oracle9*i* Streams』

構文

```
DBMS_PROPAGATION_ADM.CREATE_PROPAGATION(
   propagation_name         IN VARCHAR2,
   source_queue             IN VARCHAR2,
   destination_queue        IN VARCHAR2,
   destination_dblink       IN VARCHAR2  DEFAULT NULL,
   rule_set_name            IN VARCHAR2  DEFAULT NULL);
```

パラメータ

表 47-3 CREATE_PROPAGATION プロシージャのパラメータ

パラメータ	説明
propagation_name	作成する伝播ジョブの名前。NULL を設定することはできません。
source_queue	ソース・キューの名前。現行のデータベースにはソース・キューが含まれている必要があります。
destination_queue	宛先キューの名前。
destination_dblink	伝播ジョブで使用するデータベース・リンクの名前。データベース・リンクは、ソース・キューが含まれているデータベースから宛先キューが含まれているデータベースへのリンクです。 NULL を指定する場合は、ソース・キューおよび宛先キューが同一のデータベースであることが必要です。 **注意：**接続修飾子は使用できません。

表 47-3 CREATE_PROPAGATION プロシージャのパラメータ（続き）

パラメータ	説明
rule_set_name	この伝播ジョブの伝播ルールが含まれているルール・セットの名前。既存のルール・セットを [*schema_name*.]*rule_set_name* の形式で指定する必要があります。たとえば、ルール・セットに hr スキーマの prop_rules という名前を指定するには、hr.prop_rules と入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 指定したルール・セットが存在しない場合は、エラーが戻ります。ルール・セットを作成し、そのルール・セットにルールを追加するには、DBMS_RULE_ADM パッケージを使用します。 このパラメータに NULL を指定すると、伝播ジョブによって、そのキューのすべての LCR およびユーザー・メッセージが伝播されます。

DROP_PROPAGATION プロシージャ

伝播ジョブを削除し、ソース・キューにある宛先キューに関するすべてのメッセージを削除します。また、このプロシージャによって、ソース・キューから宛先キューへの伝播スケジュールも削除されます。

構文

```
DBMS_PROPAGATION_ADM.DROP_PROPAGATION(
   propagation_name      IN VARCHAR2);
```

パラメータ

表 47-4 DROP_PROPAGATION プロシージャのパラメータ

パラメータ	説明
propagation_name	削除する伝播ジョブの名前。既存の伝播ジョブ名を指定する必要があります。

48

DBMS_RANDOM

DBMS_RANDOM パッケージは、組込み式の乱数ジェネレータを提供します。このパッケージは、Oracle の内部乱数ジェネレータをコールするため、PL/SQL で記述したジェネレータより高速です。

この章では、次の項目について説明します。

要件

DBMS_RANDOM は、乱数ジェネレータをコールする前に初期化する必要があります。このジェネレータは 8 桁の整数を生成します。初期化サブプログラムがコールされないと、パッケージでは例外が発生します。

DBMS_RANDOM サブプログラムの要約

表 48-1 DBMS_RANDOM パッケージのサブプログラム

サブプログラム	説明
「INITIALIZE プロシージャ」48-2 ページ	シード値を使用して、パッケージを初期化します。
「SEED プロシージャ」48-3 ページ	シードをリセットします。
「RANDOM ファンクション」48-3 ページ	乱数を取得します。
「TERMINATE プロシージャ」48-3 ページ	パッケージをクローズします。

INITIALIZE プロシージャ

このパッケージを使用するためには、最初にシードを使用して初期化サブプログラムをコールします。

構文

```
DBMS_RANDOM.INITIALIZE (
   seed  IN  BINARY_INTEGER);
```

注意： 5 桁を超える十分な大きさのシードを使用してください。1 桁の値では、乱数を戻すために不十分な場合があります。また、時間などの変数値からシードを取得することも検討してください。

パラメータ

表 48-2 INITIALIZE プロシージャのパラメータ

パラメータ	説明
seed	乱数の生成に使用するシード番号。

SEED プロシージャ

このプロシージャは、シードをリセットします。

構文

```
DBMS_RANDOM.SEED (
   seed  IN  BINARY_INTEGER);
```

パラメータ

表 48-3 INITIALIZE プロシージャのパラメータ

パラメータ	説明
seed	乱数の生成に使用するシード番号。

RANDOM ファンクション

このファンクションは、乱数を取得します。

構文

```
DBMS_RANDOM.RANDOM
   RETURN BINARY_INTEGER;
```

TERMINATE プロシージャ

パッケージを終了するときは、TERMINATE プロシージャをコールします。

構文

```
DBMS_RANDOM.TERMINATE;
```

49

DBMS_RECTIFIER_DIFF

DBMS_RECTIFIER_DIFF パッケージには、2 つのレプリケート・サイト間のデータの不整合を検出して解決するための API が含まれています。

この章では、次の項目について説明します。

DBMS_RECTIFIER_DIFF サブプログラムの要約

表 49-1 DBMS_RECTIFIER_DIFF パッケージのサブプログラム

サブプログラム	説明
「DIFFERENCES プロシージャ」49-2 ページ	2 つの表の違いを判断します。
「RECTIFY プロシージャ」49-5 ページ	2 つの表の違いを解決します。

DIFFERENCES プロシージャ

このプロシージャは、2 つの表の違いを判断します。さらに、ネストした表の記憶表を受け入れます。

注意： このプロシージャは、LOB 列やユーザー定義型に基づく列では使用できません。

構文

```
DBMS_RECTIFIER_DIFF.DIFFERENCES (
   sname1                IN  VARCHAR2,
   oname1                IN  VARCHAR2,
   reference_site        IN  VARCHAR2 := '',
   sname2                IN  VARCHAR2,
   oname2                IN  VARCHAR2,
   comparison_site       IN  VARCHAR2 := '',
   where_clause          IN  VARCHAR2 := '',
   { column_list         IN  VARCHAR2 := '',
   | array_columns       IN  dbms_utility.name_array, }
   missing_rows_sname    IN  VARCHAR2,
   missing_rows_oname1   IN  VARCHAR2,
   missing_rows_oname2   IN  VARCHAR2,
   missing_rows_site     IN  VARCHAR2 := '',
   max_missing           IN  INTEGER,
   commit_rows           IN  INTEGER := 500);
```

注意： このプロシージャはオーバーロードされています。column_list パラメータと array_columns パラメータは、両方同時には指定できません。

パラメータ

表 49-2 DIFFERENCES プロシージャのパラメータ

パラメータ	説明
sname1	reference_site にあるスキーマの名前。
oname1	reference_site にある表の名前。
reference_site	参照データベース・サイトの名前。デフォルトの NULL は、現行のサイトを示します。
sname2	comparison_site にあるスキーマの名前。
oname2	comparison_site にある表の名前。
comparison_site	比較データベース・サイトの名前。デフォルトの NULL は、現行のサイトを示します。
where_clause	この句を満たす行のみが比較のために選択されます。デフォルトの NULL は、すべての行を比較することを示します。
column_list	2 つの表について比較される 1 つ以上の列名のカンマで区切られたリスト。カンマの前後に空白を入れないでください。デフォルトの NULL は、すべての列を比較することを示します。
array_columns	2 つの表について比較される列名の PL/SQL 索引付き表。索引は 1 から始まり、配列の最後の要素は NULL である必要があります。位置 1 が NULL の場合は、すべての列が使用されます。
missing_rows_sname	欠落行の情報がある表を含んでいるスキーマの名前。
missing_rows_ oname1	missing_rows_site にある既存の表の名前で、reference_site の表にあって comparison_site の表にはない行に関する情報、および comparison_site の表にあって reference_site の表にはない行に関する情報が格納されています。
missing_rows_ oname2	欠落行に関する情報が格納されている missing_rows_site にある表の名前。この表には、次の 3 つの列があります。 missing_rows_oname1 表にある行の ROWID を示す R_ID 列、行が存在するサイトの名前を示す PRESENT 列、および行が欠落しているサイトの名前を示す ABSENT 列です。

表 49-2 DIFFERENCES プロシージャのパラメータ（続き）

パラメータ	説明
missing_rows_site	missing_rows_oname1 表および missing_rows_oname2 表が配置されているサイトの名前。デフォルトの NULL は、これらの表が現行のサイトに配置されていることを示します。
max_missing	missing_rows_oname 表に挿入する必要のある最大行数を示す整数。max_missing の数値を超える行が欠落している場合は、その行数が missing_rows_oname に挿入され、ルーチンは、さらに行が欠落しているかどうかを判断することなく正常に戻ります。このパラメータは、断片部分が違いすぎるために欠落行の表の項目が多くなり、継続する必要がなくなった場合に役に立ちます。max_missing が 1 未満または NULL の場合は、例外の badnumber が発生します。
commit_rows	COMMIT が発生する前に参照表または比較表に挿入される、またはこれらの表から削除される最大行数。デフォルトでは、500 行挿入されるか、または 500 行削除されると COMMIT が発生します。空の文字列（''）または NULL は、1 つの表のすべての行が挿入または削除された後でのみ、COMMIT が発行されることを示します。

例外

表 49-3 DIFFERENCES プロシージャの例外

例外	説明
nosuchsite	データベース・サイトが見つかりません。
badnumber	commit_rows パラメータは 1 未満です。
missingprimarykey	列のリストには、主キー（または、SET_COLUMNS と等価のもの）を含める必要があります。
badname	表またはスキーマ名が NULL または空の文字列です。
cannotbenull	パラメータに NULL を指定できません。
notshapeequivalent	比較される表の形態が同じではありません。形態とは、列数、表の列名および列のデータ・タイプを指します。
unknowncolumn	列が存在しません。
unsupportedtype	サポートされていないタイプです。
dbms_repcat.commfailure	リモート・サイトにアクセスできません。
dbms_repcat.missingobject	表が存在しません。

制限事項

欠落行の表に一意キー制約または主キー制約がある場合は、エラー ORA-00001（一意制約の違反）が発行されます。

RECTIFY プロシージャ

このプロシージャは、2 つの表の違いを解決します。さらに、ネストした表の記憶表を受け入れます。

> **注意：** このプロシージャは、LOB 列やユーザー定義型に基づく列では使用できません。

構文

```
DBMS_RECTIFIER_DIFF.RECTIFY (
   sname1               IN  VARCHAR2,
   oname1               IN  VARCHAR2,
   reference_site       IN  VARCHAR2 := '',
   sname2               IN  VARCHAR2,
   oname2               IN  VARCHAR2,
   comparison_site      IN  VARCHAR2 := '',
   { column_list        IN  VARCHAR2 := '',
   | array_columns      IN  dbms_utility.name_array, }
   missing_rows_sname   IN  VARCHAR2,
   missing_rows_oname1  IN  VARCHAR2,
   missing_rows_oname2  IN  VARCHAR2,
   missing_rows_site    IN  VARCHAR2 := '',
   commit_rows          IN  INTEGER := 500);
```

> **注意：** このプロシージャはオーバーロードされています。column_list パラメータと array_columns パラメータは、両方同時には指定できません。

パラメータ

表 49-4 RECTIFY プロシージャのパラメータ

パラメータ	説明
sname1	reference_site にあるスキーマの名前。
oname1	reference_site にある表の名前。
reference_site	参照データベース・サイトの名前。デフォルトの NULL は、現行のサイトを示します。
sname2	comparison_site にあるスキーマの名前。
oname2	comparison_site にある表の名前。
comparison_site	比較データベース・サイトの名前。デフォルトの NULL は、現行のサイトを示します。
column_list	2 つの表について比較される 1 つ以上の列名のカンマで区切られたリスト。カンマの前後に空白を入れないでください。デフォルトの NULL は、すべての列を比較することを示します。
array_columns	2 つの表について比較される列名の PL/SQL 索引付き表。索引は 1 から始まり、配列の最後の要素は NULL である必要があります。位置 1 が NULL の場合は、すべての列が使用されます。
missing_rows_sname	欠落行の情報がある表を含んでいるスキーマの名前。
missing_rows_oname1	missing_rows_site にある表の名前で、reference_site の表にあって comparison_site の表にはない行に関する情報、および comparison_site の表にあって reference_site の表にはない行に関する情報が格納されています。
missing_rows_oname2	欠落行に関する情報が格納されている missing_rows_site にある表の名前。この表には、次の 3 つの列があります。 missing_rows_oname1 表にある行の ROWID、行が存在するサイトの名前、および行が欠落しているサイトの名前です。
missing_rows_site	missing_rows_oname1 表および missing_rows_oname2 表が配置されているサイトの名前。デフォルトの NULL は、これらの表が現行のサイトに配置されていることを示します。
commit_rows	COMMIT が発生する前に参照表または比較表に挿入される、またはこれらの表から削除される最大行数。デフォルトでは、500 行挿入されるか、または 500 行削除されると COMMIT が発生します。空の文字列（''）または NULL は、1 つの表のすべての行が挿入または削除された後でのみ、COMMIT が発行されることを示します。

例外

表 49-5 RECTIFY プロシージャの例外

例外	説明
nosuchsite	データベース・サイトが見つかりません。
badnumber	commit_rows パラメータは 1 未満です。
badname	表またはスキーマ名が NULL または空の文字列です。
dbms_repcat.commfailure	リモート・サイトにアクセスできません。
dbms_ repcat.missingobject	表が存在しません。

50

DBMS_REDEFINITION

DBMS_REDEFINITION を使用すると、表の再定義をオンラインで実行できます。オンラインで再定義を行うには、段階的に維持可能なローカル・マテリアライズド・ビューを使用します。段階的に維持可能なマテリアライズド・ビューをサポートするには、スナップショット・ログをマスター表に定義する必要があります。スナップショット・ログは、マスター表への変更を追跡し、リフレッシュ同期時にマテリアライズド・ビューで使用されます。オンラインで再定義可能な表の制限事項は次のとおりです。

- マテリアライズド・ビューおよびマテリアライズド・ビュー・ログで定義された表はオンラインで再定義できません。
- 表がマテリアライズド・ビュー・コンテナ表および AQ 表はオンラインで再定義できません。
- IOT 表のオーバーフロー表はオンラインで再定義できません。

> **関連項目：** 詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

この章では、次の項目について説明します。

DBMS_REDEFINITION の定数

このパッケージには、次の定数が定義されています。

- `cons_use_pk constant BINARY_INTEGER := 1;`
- `cons_use_rowid constant BINARY_INTEGER := 2;`

DBMS_REDEFINITION サブプログラムの要約

表 50-1 DBMS_REDEFINITION サブプログラム

サブプログラム	説明
「CAN_REDEF_TABLE プロシージャ」50-3 ページ	指定された表がオンラインで再定義可能かどうかを判断します。
「START_REDEF_TABLE プロシージャ」50-4 ページ	再定義プロセスを開始します。
「FINISH_REDEF_TABLE プロシージャ」50-5 ページ	再定義プロセスを完了します。
「SYNC_INTERIM_TABLE プロシージャ」50-5 ページ	一時表と元の表との同期を保ちます。
「ABORT_REDEF_TABLE プロシージャ」50-6 ページ	再定義プロセスで発生したエラーをクリーンアップします。

CAN_REDEF_TABLE プロシージャ

このプロシージャは、指定された表がオンラインで再定義可能かどうかを判断します。これはオンラインでの再定義プロセスの最初のステップです。その表がオンラインでの再定義の候補ではない場合は、エラー・メッセージが表示されます。

構文

```
DBMS_REDEFINITION.can_redef_table (
   uname        IN  VARCHAR2,
   tname        IN  VARCHAR2,
   options_flag IN  BINARY_INTEGER := 1);
```

例外

その表がオンラインでの再定義の候補ではない場合は、エラー・メッセージが表示されます。

パラメータ

表 50-2 CAN_REDEF_TABLE プロシージャのパラメータ

パラメータ	説明
uname	表のスキーマ名。
tname	再定義する表の名前。
options_flag	使用する再定義方法のタイプ。このフラグの値が dbms_redefinition.cons_use_pk の場合、再定義は主キーを使用して行われます。このフラグの値が dbms_redefinition.cons_use_rowid の場合、再定義は ROWID を使用して行われます。デフォルトの再定義方法は主キーを使用します。

START_REDEF_TABLE プロシージャ

このプロシージャは、再定義プロセスを開始します。表がオンラインで再定義可能であることを検証してから、空の一時表を（再定義する表と同じスキーマで）作成し、再定義後の表に必要な属性を指定します。

構文

```
DBMS_REDINITION.start_redef_table (
   uname        IN VARCHAR2,
   orig_table   IN VARCHAR2,
   int_table    IN VARCHAR2,
   col_mapping  IN VARCHAR2 := NULL,
   options_flag IN BINARY_INTEGER := 1);
```

パラメータ

表 50-3 START_REDEF_TABLE プロシージャのパラメータ

パラメータ	説明
uname	表のスキーマ名。
orig_table	再定義する表の名前。
int_table	一時表のスキーマ名。
col_mapping	一時表の列と元の表の列をマッピングするための情報（これは問合せの SELECT 句の列リストと同様です）。NULL の場合は、元の表の列が選択され、再定義後の名前は変更されません。
options_flag	使用する再定義方法のタイプ。このフラグの値が dbms_redefinition.cons_use_pk の場合、再定義は主キーを使用して行われます。このフラグの値が dbms_redefinition.cons_use_rowid の場合、再定義は ROWID を使用して行われます。デフォルトの再定義方法は主キーを使用します。

FINISH_REDEF_TABLE プロシージャ

このプロシージャは、再定義プロセスを完了します。このステップの前に、一時表に索引、トリガー、権限および制約を作成できます。一時表に関係する参照制約は無効にする必要があります。このステップが完了すると、一時表の属性およびデータで元の表が再定義されます。このプロシージャの際は、元の表が一時的にロックされます。

構文

```
DBMS_REDFINITION.finish_redef_table (
   uname       IN VARCHAR2,
   orig_table  IN VARCHAR2,
   int_table   IN VARCHAR2);
```

パラメータ

表 50-4 FINISH_REDEF_TABLE プロシージャのパラメータ

パラメータ	説明
uname	表のスキーマ名。
orig_table	再定義する表の名前。
int_table	一時表のスキーマ名。

SYNC_INTERIM_TABLE プロシージャ

このプロシージャは、一時表と元の表との同期を保ちます。このステップは、オンラインでの再定義完了前の finish_redef_table に必要な同期化量を最小化するのに役立ちます。このプロシージャは、一時表で長時間実行操作（索引の作成など）が行われているときにコールして元の表のデータと一時表を同期化し、以降の操作を高速化します。

構文

```
DBMS_REDFINITION.sync_interim_table (
   uname       IN VARCHAR2,
   orig_table  IN VARCHAR2,
   int_table   IN VARCHAR2);
```

パラメータ

表 50-5 SYNC_INTERIM_TABLE プロシージャのパラメータ

パラメータ	説明
uname	表のスキーマ名。
orig_table	再定義する表の名前。
int_table	一時表のスキーマ名。

ABORT_REDEF_TABLE プロシージャ

このプロシージャは、再定義プロセスで発生したエラーをクリーンアップします。このプロシージャを使用すると、start_redef_table のコール後で、finish_redef_table のコール前であれば、任意の時点で再定義プロセスを中断することもできます。

構文

```
DBMS_REDEFINITION.abort_redef_table (
   uname       IN VARCHAR2,
   orig_table  IN VARCHAR2,
   int_table   IN VARCHAR2);
```

パラメータ

表 50-6 ABORT_REDEF_TABLE プロシージャのパラメータ

パラメータ	説明
uname	表のスキーマ名。
orig_table	再定義する表の名前。
int_table	一時表のスキーマ名。

51

DBMS_REFRESH

DBMS_REFRESH によって、トランザクション的に一貫性を保つ時点にまとめてリフレッシュできるマテリアライズド・ビューのグループを作成できます。

この章では、次の項目について説明します。

DBMS_REFRESH サブプログラムの要約

表 51-1 DBMS_REFRESH パッケージのサブプログラム

サブプログラム	説明
「ADD プロシージャ」 51-2 ページ	リフレッシュ・グループにマテリアライズド・ビューを追加します。
「CHANGE プロシージャ」 51-3 ページ	リフレッシュ・グループのリフレッシュ間隔を変更します。
「DESTROY プロシージャ」 51-6 ページ	リフレッシュ・グループからすべてのマテリアライズド・ビューを削除して、そのリフレッシュ・グループを削除します。
「MAKE プロシージャ」 51-7 ページ	リフレッシュ・グループのメンバー、およびグループのメンバーをリフレッシュする時間間隔を指定します。
「REFRESH プロシージャ」 51-10 ページ	リフレッシュ・グループを手動でリフレッシュします。
「SUBTRACT プロシージャ」 51-10 ページ	リフレッシュ・グループからマテリアライズド・ビューを削除します。

ADD プロシージャ

このプロシージャは、リフレッシュ・グループにマテリアライズド・ビューを追加します。

関連項目： 詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REFRESH.ADD (
   name      IN VARCHAR2,
   { list    IN VARCHAR2,
   | tab     IN DBMS_UTILITY.UNCL_ARRAY, }
   lax       IN BOOLEAN := false);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 51-2 ADD プロシージャのパラメータ

パラメータ	説明
name	メンバーを追加するリフレッシュ・グループの名前。
list	リフレッシュ・グループに追加するマテリアライズド・ビューのカンマで区切られたリスト （シノニムはサポートされていません)。
tab	カンマで区切られたリストのかわりに、DBMS_UTILITY.UNCL_ARRAY タイプの PL/SQL 表を指定できます。この場合、各要素がマテリアライズド・ビュー名です。最初のマテリアライズド・ビューを位置 1 に設定し、最後の位置には NULL を設定する必要があります。
lax	1 つのマテリアライズド・ビューは、一度に 1 つのリフレッシュ・グループにのみ所属できます。マテリアライズド・ビューを 1 つのグループから別のグループに移動する場合は、lax フラグを TRUE に設定する必要があります。Oracle はマテリアライズド・ビューを他のリフレッシュ・グループから自動的に削除し、そのリフレッシュ間隔を新規グループのリフレッシュ間隔に更新します。そうでない場合は、ADD へのコールでエラー・メッセージが生成されます。

CHANGE プロシージャ

このプロシージャは、リフレッシュ・グループのリフレッシュ間隔を変更します。

関連項目： リフレッシュ・グループの詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REFRESH.CHANGE (
   name                 IN VARCHAR2,
   next_date            IN DATE           := NULL,
   interval             IN VARCHAR2       := NULL,
   implicit_destroy     IN BOOLEAN        := NULL,
   rollback_seg         IN VARCHAR2       := NULL,
   push_deferred_rpc    IN BOOLEAN        := NULL,
   refresh_after_errors IN BOOLEAN        := NULL,
   purge_option         IN BINARY_INTEGER := NULL,
   parallelism          IN BINARY_INTEGER := NULL,
   heap_size            IN BINARY_INTEGER := NULL);
```

パラメータ

表 51-3 CHANGE プロシージャのパラメータ

パラメータ	説明
name	リフレッシュ間隔を変更するリフレッシュ・グループの名前。
next_date	次にリフレッシュを実行する日付。デフォルトでは、この日付は変更されずにそのまま残ります。
interval	リフレッシュ・グループにあるマテリアライズド・ビューの次回のリフレッシュ時期を計算するためのファンクション。この間隔は、リフレッシュの直前に計算されます。このため、リフレッシュの実行時間より長い間隔を選択する必要があります。デフォルトでは、この間隔は変更されずにそのまま残ります。
implicit_destroy	implicit_destroy フラグの値をリセットします。このフラグを設定すると、グループにメンバーが含まれていない場合、Oracle は自動的にそのグループを削除します。デフォルトでは、このフラグは変更されずにそのまま残ります。
rollback_seg	使用するロールバック・セグメントを変更できます。デフォルトでは、ロールバック・セグメントは変更されずにそのまま残ります。このパラメータをリセットしてデフォルトのロールバック・セグメントを使用するためには、引用符も含めて NULL を指定します。引用符を付けずに NULL を指定すると、現在使用しているロールバック・セグメントを変更しないことを示します。
push_deferred_rpc	更新可能なマテリアライズド・ビューでのみ使用します。マテリアライズド・ビューをリフレッシュする前に、マテリアライズド・ビューから関連するマスター表またはマスター・マテリアライズド・ビューに変更を送信する場合、このパラメータを TRUE に設定します。そうでない場合は、変更が一時的に失われたように表示される場合があります。デフォルトでは、このフラグは変更されずにそのまま残ります。

表 51-3 CHANGE プロシージャのパラメータ（続き）

パラメータ	説明
refresh_after_errors	更新可能なマテリアライズド・ビューでのみ使用します。マテリアライズド・ビューのマスター表あるいはマスター・マテリアライズド・ビューの DEFERROR ビューに未解決の競合が記録されていても、リフレッシュを続行する場合は、このパラメータを TRUE に設定します。デフォルトでは、このフラグは変更されずにそのまま残ります。
purge_option	パラレル伝播メカニズムを使用する場合（つまり、parallelism に 1 以上を設定する場合）は、次のように指定します。 ■ 0 = パージなし ■ 1 = レイジー・パージ（デフォルト） ■ 2 = アグレッシブ・パージ ほとんどの場合、レイジー・パージが最適な設定です。複数のマスター・レプリケーション・グループが別々のターゲット・サイトに送信され、1 つ以上のレプリケーション・グループへの更新やその送信がまれな場合、アグレッシブ・パージに設定してキューを減らします。すべてのレプリケーション・グループへの更新と送信がまれな場合は、「パージなし」に設定し、キューを減らすために時々「アグレッシブ・パージ」に設定して PUSH を実行してください。
parallelism	0（ゼロ）はシリアル伝播を指定します。 *n* > 1 は *n* のパラレル処理を使用するパラレル伝播を指定します。 1 は単一のパラレル処理を使用するパラレル伝播を指定します。
heap_size	パラレル伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。 **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

DESTROY プロシージャ

このプロシージャは、リフレッシュ・グループからすべてのマテリアライズド・ビューを削除して、そのリフレッシュ・グループを削除します。

関連項目： リフレッシュ・グループの詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REFRESH.DESTROY (
   name    IN    VARCHAR2);
```

パラメータ

表 51-4 DESTROY プロシージャのパラメータ

パラメータ	説明
name	破棄するリフレッシュ・グループの名前。

MAKE プロシージャ

このプロシージャは、リフレッシュ・グループのメンバー、およびグループのメンバーをリフレッシュする時間間隔を指定します。

関連項目： 詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REFRESH.MAKE (
   name                 IN     VARCHAR2
   { list               IN     VARCHAR2,
   | tab                IN     DBMS_UTILITY.UNCL_ARRAY,}
   next_date            IN     DATE,
   interval             IN     VARCHAR2,
   implicit_destroy     IN     BOOLEAN         := false,
   lax                  IN     BOOLEAN         := false,
   job                  IN     BINARY INTEGER  := 0,
   rollback_seg         IN     VARCHAR2        := NULL,
   push_deferred_rpc    IN     BOOLEAN         := true,
   refresh_after_errors IN     BOOLEAN         := false)
   purge_option         IN     BINARY_INTEGER := NULL,
   parallelism          IN     BINARY_INTEGER := NULL,
   heap_size            IN     BINARY_INTEGER := NULL);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 51-5 MAKE プロシージャのパラメータ

パラメータ	説明
name	リフレッシュ・グループの識別に使用する一意の名前。リフレッシュ・グループは、表と同じ命名規則に従っている必要があります。
list	リフレッシュするマテリアライズド・ビューのカンマで区切られたリスト（シノニムはサポートされていません）。このようなマテリアライズド・ビューを異なるスキーマに配置し、異なるマスター表またはマスター・マテリアライズド・ビューを設定できます。ただし、リストされているすべてのマテリアライズド・ビューは、現行のデータベースに存在している必要があります。
tab	カンマで区切られたリストのかわりに、データ・タイプ DBMS_UTILITY.UNCL_ARRAY を使用して、リフレッシュするマテリアライズド・ビュー名の PL/SQL 索引付き表を指定できます。表に *n* 個のマテリアライズド・ビュー名が含まれている場合、最初のマテリアライズド・ビューを位置 1 に設定し、*n* + *1* の位置には NULL を設定する必要があります。
next_date	次にリフレッシュを実行する日付。
interval	グループにあるマテリアライズド・ビューの次回のリフレッシュ時期を計算するためのファンクション。このフィールドは、next_date 値とともに使用されます。 たとえば、間隔として NEXT_DAY(SYSDATE+1, "MONDAY") を指定した場合、next_date の値が月曜日になると、Oracle はマテリアライズド・ビューを月曜日ごとにリフレッシュします。この間隔は、リフレッシュの直前に計算されます。このため、リフレッシュの実行時間より長い間隔を選択する必要があります。
implicit_destroy	リフレッシュ・グループにメンバーがないときにそのグループを自動的に削除する場合は、このパラメータを TRUE に設定します。Oracle は、ユーザーが SUBTRACT プロシージャをコールしたときのみ、このフラグをチェックします。つまり、このフラグを設定しても、空のリフレッシュ・グループを作成できます。
lax	1 つのマテリアライズド・ビューは、一度に 1 つのリフレッシュ・グループにのみ所属できます。マテリアライズド・ビューを既存のグループから新規のリフレッシュ・グループに移動する場合は、このパラメータを TRUE に設定する必要があります。Oracle はマテリアライズド・ビューを他のリフレッシュ・グループから自動的に削除し、そのリフレッシュ間隔を新規グループのリフレッシュ間隔に更新します。そうでない場合は、MAKE へのコールでエラー・メッセージが生成されます。
job	インポート・ユーティリティで必要です。デフォルト値の 0（ゼロ）を使用してください。

表 51-5 MAKE プロシージャのパラメータ（続き）

パラメータ	説明
rollback_seg	マテリアライズド・ビューのリフレッシュ中に使用するロールバック・セグメントの名前。デフォルトの NULL では、デフォルト・ロールバック・セグメントが使用されます。
push_deferred_rpc	更新可能なマテリアライズド・ビューでのみ使用します。マテリアライズド・ビューをリフレッシュする前に、マテリアライズド・ビューから関連するマスター表またはマスター・マテリアライズド・ビューに変更を送信する場合、デフォルト値 TRUE を使用します。そうでない場合は、変更が一時的に失われたように表示される場合があります。
refresh_after_errors	更新可能なマテリアライズド・ビューでのみ使用します。マテリアライズド・ビューのマスター表あるいはマスター・マテリアライズド・ビューの DEFERROR ビューに未解決の競合が記録されていても、リフレッシュを続行する場合は、0（ゼロ）に設定します。
purge_option	パラレル伝播メカニズムを使用する場合（つまり、並列性に 1 以上を設定）は、次のように指定します。0 = パージなし、1 = レイジー・パージ（デフォルト）、2 = アグレッシブ・パージ。ほとんどの場合、レイジー・パージが最適な設定です。 複数のマスター・レプリケーション・グループが別々のターゲット・サイトに送信され、1 つ以上のレプリケーション・グループへの更新やその送信がまれな場合、アグレッシブ・パージに設定してキューを減らします。すべてのレプリケーション・グループへの更新と送信がまれな場合は、「パージなし」に設定し、キューを減らすために時々「アグレッシブ・パージ」に設定して PUSH を実行してください。
parallelism	0（ゼロ）はシリアル伝播を指定します。 *n* > 1 は *n* のパラレル処理を使用するパラレル伝播を指定します。 1 は単一のパラレル処理を使用するパラレル伝播を指定します。
heap_size	パラレル伝播スケジューリングで同時に検査されるトランザクションの最大数。最適なパフォーマンスのためのデフォルト設定は Oracle が自動的に計算します。 **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

REFRESH プロシージャ

このプロシージャでは、リフレッシュ・グループを手動でリフレッシュします。

関連項目： リフレッシュ・グループの詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REFRESH.REFRESH (
   name    IN    VARCHAR2);
```

パラメータ

表 51-6 REFRESH プロシージャのパラメータ

パラメータ	説明
name	手動でリフレッシュするリフレッシュ・グループの名前。

SUBTRACT プロシージャ

このプロシージャは、リフレッシュ・グループからマテリアライズド・ビューを削除します。

関連項目： リフレッシュ・グループの詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REFRESH.SUBTRACT (
   name       IN    VARCHAR2,
   { list     IN    VARCHAR2,
   | tab      IN    DBMS_UTILITY.UNCL_ARRAY, }
   lax        IN    BOOLEAN := false);
```

注意： このプロシージャはオーバーロードされています。list パラメータと tab パラメータは、両方同時には指定できません。

パラメータ

表 51-7 SUBTRACT プロシージャのパラメータ

パラメータ	説明
name	メンバーを削除するリフレッシュ・グループの名前。
list	リフレッシュ・グループから削除するマテリアライズド・ビューのカンマで区切られたリスト （シノニムはサポートされていません）。このようなマテリアライズド・ビューを異なるスキーマに配置し、異なるマスター表またはマスター・マテリアライズド・ビューを設定できます。ただし、リストされているすべてのマテリアライズド・ビューは、現行のデータベースに存在している必要があります。
tab	カンマで区切られたリストのかわりに、データ・タイプ DBMS_UTILITY.UNCL_ARRAY を使用して、リフレッシュするマテリアライズド・ビュー名の PL/SQL 索引付き表を指定できます。表に *n* 個のマテリアライズド・ビュー名が含まれている場合、最初のマテリアライズド・ビューを位置 1 に設定し、*n* + 1 の位置には NULL を設定する必要があります。
lax	削除するマテリアライズド・ビューがリフレッシュ・グループのメンバーでないときに、Oracle がエラー・メッセージを生成するようにする場合は、このパラメータを FALSE に設定します。

52

DBMS_REPAIR

DBMS_REPAIR には、データ破損修復プロシージャが含まれており、ユーザーは表および索引にある破損ブロックを検出して修復できます。可能な場合は破損をつき止め、再構築中または修復中にオブジェクトの使用を続行できます。

注意： DBMS_REPAIR パッケージは、データベース管理者のみの使用を目的としています。アプリケーション開発者を対象にした機能ではありません。

関連項目： DBMS_REPAIR パッケージの使用方法は、『Oracle9*i* データベース管理者ガイド』を参照してください。

この章では、次の項目について説明します。

セキュリティ、列挙タイプおよび例外

セキュリティ

このパッケージの所有者は SYS です。他のユーザーに実行権限は付与されていません。

列挙タイプ

DBMS_REPAIR パッケージは、パラメータ値の指定に使用するいくつかの列挙定数を定義します。列挙定数にはパッケージ名を接頭辞として付加する必要があります。たとえば、DBMS_REPAIR.TABLE_OBJECT と記述します。

表 52-1 は、パラメータおよび列挙定数の一覧です。

表 52-1 DBMS_REPAIR の列挙タイプ

パラメータ	定数
object_type	TABLE_OBJECT、INDEX_OBJECT、CLUSTER_OBJECT
action	CREATE_ACTION、DROP_ACTION、PURGE_ACTION
table_type	REPAIR_TABLE、ORPHAN_TABLE
flags	SKIP_FLAG、NOSKIP_FLAG

注意： デフォルトの table_name は、table_type が REPAIR_TABLE のときは REPAIR_TABLE で、table_type が ORPHAN_TABLE のときは ORPHAN_KEY_TABLE です。

例外

表 52-2 DBMS_REPAIR の例外

例外	説明	アクション
942	指定した表が存在していないと、DROP_ACTION 時に DBMS_REPAIR.ADMIN_TABLES によって戻されます。	
955	指定した表がすでに存在していると、DBMS_REPAIR. CREATE_ACTION によって戻されます。	
24120	指定した DBMS_REPAIR プロシージャに無効なパラメータが渡されました。	有効なパラメータ値を指定するか、またはパラメータのデフォルトを使用してください。
24122	ブロック範囲の指定に誤りがあります。	BLOCK_START および BLOCK_END パラメータに正しい値を指定してください。
24123	指定した機能を使用しようとしましたが、その機能はまだ実装されていません。	この機能は使用しないでください。
24124	ACTION パラメータに無効な値が指定されました。	ACTION パラメータには、CREATE_ACTION、PURGE_ACTION または DROP_ACTION のいずれかを指定してください。
24125	DBMS_REPAIR.CHECK_OBJECT の実行後に削除または切り捨てられたオブジェクトの破損ブロックを修正しようとしました。	DBMS_REPAIR.ADMIN_TABLES で修復表をパージし、DBMS_REPAIR.CHECK_OBJECT を実行して、修正対象の破損ブロックがあるかどうかを確認してください。
24127	CREATE_ACTION 以外の ACTION で TABLESPACE パラメータが指定されました。	CREATE_ACTION 以外のアクションの実行時には、TABLESPACE は指定しないでください。
24128	パーティション化されていないオブジェクトに対して、パーティション名が指定されました。	パーティション名は、オブジェクトがパーティション化されているときのみ指定してください。
24129	接頭辞を指定しないで、table_name パラメータを渡そうとしました。	有効な table_name パラメータを渡してください。

表 52-2 DBMS_REPAIR の例外（続き）

例外	説明	アクション
24130	存在しない修復表または親なし表を指定しようとしました。	table_name パラメータに有効な値を指定してください。
24131	誤った定義内容の修復表または親なし表を指定しようとしました。	正しく作成された表を参照する表名を指定してください。
24132	30 文字を超える表名を指定しようとしました。	table_name パラメータに有効な値を指定してください。

DBMS_REPAIR サブプログラムの要約

表 52-3 DBMS_REPAIR パッケージのサブプログラム

サブプログラム	説明
「ADMIN_TABLES プロシージャ」52-5 ページ	DBMS_REPAIR パッケージの修復表および孤立したキー表に対して、作成、パージおよび削除処理を実行する管理ファンクションを提供します。
「CHECK_OBJECT プロシージャ」52-6 ページ	表または索引の破損を検出し、レポートします。
「DUMP_ORPHAN_KEYS プロシージャ」52-8 ページ	破損データ・ブロック内の行を指す索引エントリをレポートします。
「FIX_CORRUPT_BLOCKS プロシージャ」52-9 ページ	CHECK_OBJECT によって破損が検出されたブロックにソフトウェア破損のマークを付けます。
「REBUILD_FREELISTS プロシージャ」52-10 ページ	オブジェクトの空きリストを再作成します。
「SKIP_CORRUPT_BLOCKS プロシージャ」52-11 ページ	表および索引のスキャン時に破損マークのあるブロックを無視するか、または破損マークのあるブロックが検出された場合に ORA-1578 をレポートするかを設定します。
「SEGMENT_FIX_STATUS プロシージャ」52-13 ページ	ビットマップ・エントリの破損状態を修正します。

ADMIN_TABLES プロシージャ

このプロシージャは、DBMS_REPAIR パッケージの修復表および孤立したキー表に対する管理ファンクションを提供します。

構文

```
DBMS_REPAIR.ADMIN_TABLES (
   table_name  IN   VARCHAR2,
   table_type  IN   BINARY_INTEGER,
   action      IN   BINARY_INTEGER,
   tablespace  IN   VARCHAR2        DEFAULT NULL);
```

パラメータ

表 52-4 ADMIN_TABLES プロシージャのパラメータ

パラメータ	説明
table_name	処理する表の名前。指定した table_type に基づいて、ORPHAN_KEY_TABLE または REPAIR_TABLE をデフォルト設定します。指定する場合は、表名に接頭辞 ORPHAN_ または REPAIR_ を付ける必要があります。
table_type	表のタイプ。ORPHAN_TABLE または REPAIR_TABLE のいずれかです。 52-2 ページの「列挙タイプ」を参照してください。
action	実行する管理アクションを示します。 CREATE_ACTION、PURGE_ACTION または DROP_ACTION のいずれかです。CREATE_ACTION の指定時に、表がすでに存在しているとエラーが戻されます。PURGE_ACTION を指定すると、存在しないオブジェクトに関連付けられている表内の行はすべて削除されます。DROP_ACTION の指定時に、表が存在していないとエラーが戻されます。 CREATE_ACTION および DROP_ACTION を指定すると、DBA_<table_name> という名前の関連ビューが、それぞれ作成または削除されます。このビューは、存在しないオブジェクトに関連付けられている行を排除するように定義されています。 SYS スキーマ内に作成されます。 52-2 ページの「列挙タイプ」を参照してください。

表 52-4 ADMIN_TABLES プロシージャのパラメータ（続き）

パラメータ	説明
tablespace	表の作成時に使用する表領域を示します。 デフォルトでは、SYS のデフォルト表領域が使用されます。CREATE_ACTION 以外のときに表領域を指定するとエラーが戻されます。

CHECK_OBJECT プロシージャ

このプロシージャは、指定したオブジェクトをチェックし、破損および修復指示に関する情報を修復表に移入します。

妥当性チェックでは、オブジェクト内のすべてのブロックがチェックされます。オブジェクトの一部をチェック対象とする場合は、オプションで、DBA 範囲、パーティション名またはサブパーティション名を指定することもできます。

構文

```
DBMS_REPAIR.CHECK_OBJECT (
   schema_name       IN  VARCHAR2,
   object_name       IN  VARCHAR2,
   partition_name    IN  VARCHAR2       DEFAULT NULL,
   object_type       IN  BINARY_INTEGER DEFAULT TABLE_OBJECT,
   repair_table_name IN  VARCHAR2       DEFAULT 'REPAIR_TABLE',
   flags             IN  BINARY_INTEGER DEFAULT NULL,
   relative_fno      IN  BINARY_INTEGER DEFAULT NULL,
   block_start       IN  BINARY_INTEGER DEFAULT NULL,
   block_end         IN  BINARY_INTEGER DEFAULT NULL,
   corrupt_count     OUT BINARY_INTEGER);
```

パラメータ

表 52-5 CHECK_OBJECT プロシージャのパラメータ

パラメータ	説明
schema_name	チェックするオブジェクトのスキーマ名。
object_name	チェックする表または索引の名前。
partition_name	チェックするパーティションまたはサブパーティションの名前。 パーティション・オブジェクトで、partition_name が指定されていない場合は、すべてのパーティションおよびサブパーティションがチェックされます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションがチェックされます。
object_type	処理するオブジェクトのタイプ。TABLE_OBJECT（デフォルト）または INDEX_OBJECT のいずれかです。 52-2 ページの「列挙タイプ」を参照してください。
repair_table_name	情報を移入する修復表の名前。 この表は、SYS スキーマに存在している必要があります。修復表を作成するには、admin_tables プロシージャを使用します。デフォルト名は REPAIR_TABLE です。
flags	将来の使用のために確保。
relative_fno	相対ファイル番号。ブロック範囲の指定時に使用します。
block_start	ブロック範囲を指定する場合に、最初に処理するブロックを指定します。オブジェクトが単一表、パーティションまたはサブパーティションの場合のみ指定できます。
block_end	ブロック範囲を指定する場合に、最後に処理するブロックを指定します。オブジェクトが単一表、パーティションまたはサブパーティションの場合のみ指定できます。block_start または block_end のいずれか一方のみ指定した場合、他方の値は、ファイル内の第 1 ブロックまたは最終ブロックにそれぞれデフォルト設定されます。
corrupt_count	レポートされた破損数。

DUMP_ORPHAN_KEYS プロシージャ

このプロシージャは、破損データ・ブロック内の行を指す索引エントリをレポートします。検出された該当索引エントリごとに、指定した親なし表に行が挿入されます。

修復表が指定されている場合は、ソフトウェア破損のマークがあるすべてのデータ・ブロックの他に、ベース表に関連付けられている破損ブロックが処理されます。修復表が指定されていない場合は、破損マークのあるブロックのみ処理されます。

この情報は、表内で失われた行を再構築する場合や診断の目的に使用されます。

構文

```
DBMS_REPAIR.DUMP_ORPHAN_KEYS (
   schema_name       IN  VARCHAR2,
   object_name       IN  VARCHAR2,
   partition_name    IN  VARCHAR2       DEFAULT NULL,
   object_type       IN  BINARY_INTEGER DEFAULT INDEX_OBJECT,
   repair_table_name IN  VARCHAR2       DEFAULT 'REPAIR_TABLE',
   orphan_table_name IN  VARCHAR2       DEFAULT 'ORPHAN_KEYS_TABLE',
   flags             IN  BINARY_INTEGER DEFAULT NULL,
   key_count         OUT BINARY_INTEGER);
```

パラメータ

表 52-6 DUMP_ORPHAN_KEYS プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマ名。
object_name	オブジェクト名。
partition_name	処理するパーティションまたはサブパーティションの名前。 パーティション・オブジェクトで、partition_name を指定しない場合は、すべてのパーティションおよびサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type	処理するオブジェクトのタイプ。デフォルトは INDEX_OBJECT です。 52-2 ページの「列挙タイプ」を参照してください。
repair_table_name	ベース表の破損ブロックに関する情報を含んだ修復表の名前。 指定した表は、SYS スキーマに存在している必要があります。表を作成するには、admin_tables プロシージャを使用します。

表 52-6 DUMP_ORPHAN_KEYS プロシージャのパラメータ（続き）

パラメータ	説明
orphan_table_name	破損データ・ブロック内の行を参照する各索引エントリに関する情報を移入する孤立したキー表の名前。 指定した表は、SYS スキーマに存在している必要があります。表を作成するには、admin_tables プロシージャを使用します。
flags	将来の使用のために確保。
key_count	処理された索引エントリ数。

FIX_CORRUPT_BLOCKS プロシージャ

このプロシージャは、check_object プロシージャによって事前に生成された修復表の情報に基づいて、指定したオブジェクト内の破損ブロックを修正します。

ブロックに変更を加える前に、そのブロックがまだ破損状態であることを確認するチェックが行われます。破損ブロックは、そのブロックにソフトウェア破損のマークを付けることによって修復されます。修復が有効になると、修復表内の関連行が修正タイムスタンプで更新されます。

構文

```
DBMS_REPAIR.FIX_CORRUPT_BLOCKS (
   schema_name       IN  VARCHAR2,
   object_name       IN  VARCHAR2,
   partition_name    IN  VARCHAR2       DEFAULT NULL,
   object_type       IN  BINARY_INTEGER DEFAULT TABLE_OBJECT,
   repair_table_name IN  VARCHAR2       DEFAULT 'REPAIR_TABLE',
   flags             IN  BINARY_INTEGER DEFAULT NULL,
   fix_count         OUT BINARY_INTEGER);
```

パラメータ

表 52-7 FIX_CORRUPT_BLOCKS プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマ名。
object_name	修正対象の破損ブロックがあるオブジェクトの名前。
partition_name	処理するパーティションまたはサブパーティションの名前。 パーティション・オブジェクトで、partition_name を指定しない場合は、すべてのパーティションおよびサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type	処理するオブジェクトのタイプ。TABLE_OBJECT（デフォルト）または INDEX_OBJECT のいずれかです。 52-2 ページの「列挙タイプ」を参照してください。
repair_table_name	修復指示を含んだ修復表の名前。 SYS スキーマに存在している必要があります。
flags	将来の使用のために確保。
fix_count	修正されたブロック数。

REBUILD_FREELISTS プロシージャ

このプロシージャは、指定したオブジェクトの空きリストを再作成します。すべての空きブロックは、マスター空きリストに格納されます。その他の空きリストはすべて 0（ゼロ）になります。

オブジェクトに複数の空きリスト・グループがある場合、空きブロックは、ラウンドロビン方式で異なるグループに割り当てられ、すべての空きリスト間に配分されます。

構文

```
DBMS_REPAIR.REBUILD_FREELISTS (
   schema_name    IN VARCHAR2,
   object_name    IN VARCHAR2,
   partition_name IN VARCHAR2       DEFAULT NULL,
   object_type    IN BINARY_INTEGER DEFAULT TABLE_OBJECT);
```

パラメータ

表 52-8 REBUILD_FREELISTS プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマ名。
object_name	空きリストを再作成するオブジェクトの名前。
object_type	処理するオブジェクトのタイプ。TABLE_OBJECT（デフォルト）または INDEX_OBJECT のいずれかです。 52-2 ページの「列挙タイプ」を参照してください。

SKIP_CORRUPT_BLOCKS プロシージャ

このプロシージャは、指定したオブジェクトの索引および表のスキャン時に、破損ブロックのスキップを使用可能または使用禁止にします。

オブジェクトが表のときは、スキップが表およびその索引に適用されます。オブジェクトがクラスタのときは、クラスタ内のすべての表およびその各索引に適用されます。

> **注意：** ベース表に対して DBMS_REPAIR.SKIP_CORRUPT_BLOCKS を設定した後、破損している索引で索引レンジ・スキャンを実行すると、破損ブランチ・ブロックおよび破損ルート・ブロックはスキップされません。ルート以外の破損しているリーフ・ブロックのみがスキップされます。

構文

```
DBMS_REPAIR.SKIP_CORRUPT_BLOCKS (
   schema_name  IN VARCHAR2,
   object_name  IN VARCHAR2,
   object_type  IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
   flags        IN BINARY_INTEGER DEFAULT SKIP_FLAG);
```

パラメータ

表 52-9 SKIP_CORRUPT_BLOCKS プロシージャのパラメータ

パラメータ	説明
schema_name	処理するオブジェクトのスキーマ名。
object_name	オブジェクト名。
partition_name (オプション)	処理するパーティションまたはサブパーティションの名前。 パーティション・オブジェクトで、partition_name を指定しない場合は、すべてのパーティションおよびサブパーティションが処理されます。パーティション・オブジェクトで、指定したパーティションにサブパーティションが含まれている場合は、すべてのサブパーティションが処理されます。
object_type	処理するオブジェクトのタイプ。TABLE_OBJECT（デフォルト）または CLUSTER_OBJECT のいずれかです。 52-2 ページの「列挙タイプ」を参照してください。
flags	SKIP_FLAG を指定すると、索引および表のスキャン時に、そのオブジェクトのソフトウェア破損ブロックのスキップがオンになります。NOSKIP_FLAG を指定すると、ソフトウェア破損ブロックが検出されたときに、ORA-1578 エラーが戻されます。 52-2 ページの「列挙タイプ」を参照してください。

SEGMENT_FIX_STATUS プロシージャ

このプロシージャで、ビットマップ・エントリの破損状態を修正します。このプロシージャでは、状態が対応ブロックの現行の内容に基づいて再計算されるか、特定の値に設定されます。

構文

```
DBMS_REPAIR.SEGMENT_FIX_STATUS (
   segment_owner   IN VARCHAR2,
   segment_name    IN VARCHAR2,
   segment_type    IN BINARY_INTEGER DEFAULT TABLE_OBJECT,
   file_number     IN BINARY_INTEGER DEFAULT NULL,
   block_number    IN BINARY_INTEGER DEFAULT NULL,
   status_value    IN BINARY_INTEGER DEFAULT NULL,
   partition_name  IN VARCHAR2 DEFAULT NULL,);
```

パラメータ

表 52-10 SEGMENT_FIX_STATUS プロシージャのパラメータ

パラメータ	説明
segment_owner	セグメントのスキーマ名。
segment_name	セグメント名。
partition_name	オプション。個々のパーティション名。NULL は非パーティション化オブジェクトを示します。デフォルトは NULL です。
segment_type	セグメントのオプション・タイプ（TABLE や INDEX など）。デフォルトは NULL です。
file_number	（オプション）ステータスを固定する必要のあるデータ・ブロックの、相対表領域ファイル番号。省略した場合、セグメント内の全ブロックで、状態が正しいかどうかチェックされ修正が行われます。
block_number	（オプション）ステータスを固定する必要のあるデータ・ブロックの、相対ファイル・ファイル番号。省略した場合、セグメント内の全ブロックで、状態が正しいかどうかチェックされ修正が行われます。

表 52-10 SEGMENT_FIX_STATUS プロシージャのパラメータ（続き）

パラメータ	説明
status_value	（オプション）file_number および block_number で示したブロック・ステータスが設定される値。省略すると、ブロックの現在の状態に基づいてステータスが設定されます。ほとんどの場合がこれに該当しますが、計算アルゴリズムにバグがある場合は手動で値を設定します。ステータス値は次のとおりです。 1 = ブロックがいっぱいです。 2 = ブロックに 0 ～ 25% の空きがあります。 3 = ブロックに 25 ～ 50% の空きがあります。 4 = ブロックに 50 ～ 75% の空きがあります。 5 = ブロックに 75 ～ 100% の空きがあります。 ビットマップ・ブロック、セグメント・ヘッダーおよびエクステント・マップ・ブロックのステータスは変更できません。固定ハッシュ領域のブロックのステータスは変更できません。索引ブロックのステータスは「1 = ブロックがいっぱいです」および「3 = ブロックに空きがあります」の 2 つのみです。

例

```
/* Fix the bitmap status for all the blocks in table mytab in schema sys */
execute dbms_repair.segment_fix_status('SYS', 'MYTAB');

/* Mark block number 45, filenumber 1 for table mytab in sys schema as FULL.*/
execute dbms_repair.segment_fix_status('SYS', 'MYTAB', 1,1, 45, 1);
```

53

DBMS_REPCAT

DBMS_REPCAT は、レプリケーション・カタログおよびレプリケーション環境を管理および更新するためのルーチンを提供します。

この章では、次の項目について説明します。

DBMS_REPCAT サブプログラムの要約

表 53-1 DBMS_REPCAT サブプログラム

サブプログラム	説明
「ADD_GROUPED_COLUMN プロシージャ」53-7 ページ	既存の列グループにメンバーを追加します。
「ADD_MASTER_DATABASE プロシージャ」53-8 ページ	レプリケーション環境に別のマスター・サイトを追加します。
「ADD_NEW_MASTERS プロシージャ」53-10 ページ	DBA_REPSITES_NEW データ・ディクショナリ・ビューのマスター・サイトを、使用可能なすべてのマスター・サイトのレプリケーション・カタログに追加します。
「ADD_PRIORITY_datatype プロシージャ」53-15 ページ	優先グループにメンバーを追加します。
「ADD_SITE_PRIORITY_SITE プロシージャ」53-17 ページ	サイト優先グループに新規サイトを追加します。
「ADD_conflicttype_RESOLUTION プロシージャ」53-18 ページ	更新、削除または一意性競合の解消方法を指定します。
「ALTER_CATCHUP_PARAMETERS プロシージャ」53-23 ページ	DBA_REPEXTENSIONS データ・ディクショナリ・ビューに格納されたパラメータの値を変更します。
「ALTER_MASTER_PROPAGATION プロシージャ」53-25 ページ	指定したマスター・サイトで指定したレプリケーション・グループの伝播方法を変更します。
「ALTER_MASTER_REPOBJECT プロシージャ」53-27 ページ	レプリケーション環境内のオブジェクトを変更します。
「ALTER_MVIEW_PROPAGATION プロシージャ」53-30 ページ	現行のマテリアライズド・ビュー・サイトで指定したレプリケーション・グループの伝播方法を変更します。
「ALTER_PRIORITY プロシージャ」53-32 ページ	指定した優先グループ・メンバーに関連付けられている優先順位レベルを変更します。
「ALTER_PRIORITY_datatype プロシージャ」53-33 ページ	優先グループ内のメンバーの値を変更します。
「ALTER_SITE_PRIORITY プロシージャ」53-35 ページ	指定したサイトに関連付けられている優先順位レベルを変更します。
「ALTER_SITE_PRIORITY_SITE プロシージャ」53-36 ページ	指定した優先順位レベルに関連付けられているサイトを変更します。

表 53-1 DBMS_REPCAT サブプログラム（続き）

サブプログラム	説明
「CANCEL_STATISTICS プロシージャ」53-38 ページ	表の更新競合、一意性競合および削除競合の正常な解消に関する統計の収集を停止します。
「COMMENT_ON_COLUMN_GROUP プロシージャ」53-39 ページ	列グループの ALL_REPCOLUMN_GROUP ビューのコメント・フィールドを更新します。
「COMMENT_ON_conflicttype_RESOLUTION プロシージャ」53-46 ページ	マテリアライズド・ビュー・サイトの ALL_REPGROUP ビューの SCHEMA_COMMENT フィールドを更新します。
「COMMENT_ON_PRIORITY_GROUP/COMMENT_ON_SITE_PRIORITY プロシージャ」53-41 ページ	（サイト）優先グループの ALL_REPPRIORITY_GROUP ビューのコメント・フィールドを更新します。
「COMMENT_ON_REPGROUP プロシージャ」53-42 ページ	マスター・グループの ALL_REPGROUP ビューのコメント・フィールドを更新します。
「COMMENT_ON_REPOBJECT プロシージャ」53-43 ページ	レプリケート・オブジェクトの ALL_REPOBJECT ビューのコメント・フィールドを更新します。
「COMMENT_ON_REPSITES プロシージャ」53-44 ページ	レプリケート・サイトの ALL_REPSITE ビューのコメント・フィールドを更新します。
「COMMENT_ON_conflicttype_RESOLUTION プロシージャ」53-46 ページ	競合解消ルーチンの ALL_REPRESOLUTION ビューのコメント・フィールドを更新します。
「COMPARE_OLD_VALUES プロシージャ」53-47 ページ	更新および削除のために、レプリケート表の各非キー列に対して、各マスター・サイトの元の列値を比較するかどうかを指定します。
「CREATE_MASTER_REPGROUP プロシージャ」53-50 ページ	新規に、空の停止中マスター・グループを作成します。
「CREATE_MASTER_REPOBJECT プロシージャ」53-51 ページ	オブジェクトをレプリケート・オブジェクトとして指定します。
「CREATE_MVIEW_REPGROUP プロシージャ」53-55 ページ	ローカル・データベース内に、新規で空のマテリアライズド・ビュー・グループを作成します。
「CREATE_MVIEW_REPOBJECT プロシージャ」53-56 ページ	マテリアライズド・ビュー・グループにレプリケート・オブジェクトを追加します。
「DEFINE_COLUMN_GROUP プロシージャ」53-60 ページ	空の列グループを作成します。
「DEFINE_PRIORITY_GROUP プロシージャ」53-61 ページ	マスター・グループに新規の優先グループを作成します。

表 53-1 DBMS_REPCAT サブプログラム（続き）

サブプログラム	説明
「DEFINE_SITE_PRIORITY プロシージャ」53-62 ページ	マスター・グループに新規のサイト優先グループを作成します。
「DO_DEFERRED_REPCAT_ADMIN プロシージャ」53-63 ページ	現行のマスター・サイトで指定したマスター・グループまたはすべてのマスター・サイトに対して未処理のローカル遅延管理プロシージャを実行します。
「DROP_COLUMN_GROUP プロシージャ」53-64 ページ	列グループを削除します。
「DROP_GROUPED_COLUMN プロシージャ」53-65 ページ	列グループからメンバーを削除します。
「DROP_MASTER_REPGROUP プロシージャ」53-67 ページ	現行のサイトからマスター・グループを削除します。
「DROP_MASTER_REPOBJECT プロシージャ」53-68 ページ	マスター・グループからレプリケート・オブジェクトを削除します。
「DROP_PRIORITY プロシージャ」53-72 ページ	マスター・グループからレプリケート・オブジェクトを削除します。
「DROP_MVIEW_REPGROUP プロシージャ」53-70 ページ	レプリケーション環境からマテリアライズド・ビュー・サイトを削除します。
「DROP_MVIEW_REPOBJECT プロシージャ」53-71 ページ	マテリアライズド・ビュー・サイトからレプリケート・オブジェクトを削除します。
「DROP_PRIORITY プロシージャ」53-72 ページ	優先順位レベルに従って優先グループのメンバーを削除します。
「DROP_PRIORITY_GROUP プロシージャ」53-73 ページ	指定したマスター・グループの優先グループを削除します。
「DROP_PRIORITY_datatype プロシージャ」53-74 ページ	値による優先グループのメンバーを削除します。
「DROP_SITE_PRIORITY プロシージャ」53-76 ページ	指定したマスター・グループのサイト優先グループを削除します。
「DROP_SITE_PRIORITY_SITE プロシージャ」53-77 ページ	サイト優先グループから、名前を指定してサイトを削除します。
「DROP_conflicttype_RESOLUTION プロシージャ」53-78 ページ	更新、削除または一意性競合解消方法を削除します。
「EXECUTE_DDL プロシージャ」53-80 ページ	各マスター・サイトで実行する DDL を提供します。

表 53-1 DBMS_REPCAT サブプログラム（続き）

サブプログラム	説明
「GENERATE_MVIEW_SUPPORT プロシージャ」53-81 ページ	トリガーを起動し、更新可能なマテリアライズド・ビューのレプリケーションまたはプロシージャ・レプリケーションのサポートに必要なパッケージを生成します。
「GENERATE_REPLICATION_SUPPORT プロシージャ」53-83 ページ	指定したオブジェクトのレプリケーションのサポートに必要なトリガー、パッケージおよびプロシージャを生成します。
「MAKE_COLUMN_GROUP プロシージャ」53-85 ページ	1 つ以上のメンバーを含んだ新しい列グループを作成します。
「PREPARE_INSTANTIATED_MASTER プロシージャ」53-86 ページ	マスター・グループに追加するデータベースのグローバル名を変更します。
「PURGE_MASTER_LOG プロシージャ」53-88 ページ	指定した識別番号、ソースまたはレプリケート・マスター・グループに関連している DBA_REPCATLOG 内のローカル・メッセージを削除します。
「PURGE_STATISTICS プロシージャ」53-89 ページ	ALL_REPRESOLUTION_STATISTICS ビューから情報を削除します。
「REFRESH_MVIEW_REPGROUP プロシージャ」53-90 ページ	マテリアライズド・ビュー・グループを、関連するマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトからの最新データでリフレッシュします。
「REGISTER_MVIEW_REPGROUP プロシージャ」53-92 ページ	DBA_REGISTERED_MVIEW_GROUPS への挿入、変更または削除を行うことによって、各マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのマテリアライズド・ビュー管理を容易にします。
「REGISTER_STATISTICS プロシージャ」53-94 ページ	表の更新競合、削除競合および一意性競合の正常な解消に関する情報を収集します。
「RELOCATE_MASTERDEF プロシージャ」53-95 ページ	マスター定義サイトをレプリケーション環境内の別のマスター・サイトに変更します。
「REMOVE_MASTER_DATABASES プロシージャ」53-97 ページ	レプリケーション環境から 1 つ以上のマスター・データベースを削除します。
「RENAME_SHADOW_COLUMN_GROUP プロシージャ」53-98 ページ	レプリケート表のシャドウ列グループを改名して、同グループを名前付きの列グループにします。

表 53-1 DBMS_REPCAT サブプログラム（続き）

サブプログラム	説明
「REPCAT_IMPORT_CHECK プロシージャ」53-99 ページ	レプリケート・オブジェクトまたはアドバンスト・レプリケーション機能で使用されたオブジェクトのエクスポートまたはインポートの実行後、マスター・グループ内のオブジェクトの識別子およびステータス値が適切かを確認します。
「RESUME_MASTER_ACTIVITY プロシージャ」53-100 ページ	レプリケーション環境の休止後、通常のレプリケーション・アクティビティを再開します。
「RESUME_PROPAGATION_TO_MDEF プロシージャ」53-101 ページ	エクスポートが効果的に実行され、マスター・サイトに存在する拡張レプリケーション・グループおよび影響を受けないレプリケーション・グループの両方の伝播が可能になることを示します。
「SEND_OLD_VALUES プロシージャ」53-102 ページ	更新および削除のために、レプリケート表の各非キー列の元の値を送信するかどうかを指定します。
「SET_COLUMNS プロシージャ」53-105 ページ	行レベル・レプリケーションの使用時に、主キーのかわりに使用する代替列または列グループを指定し、表のどの列を比較するかを判断します。
「SPECIFY_NEW_MASTERS プロシージャ」53-107 ページ	既存のグループに追加するマスター・サイトで、かつそのグループを休止させないものを指定します。
「SUSPEND_MASTER_ACTIVITY プロシージャ」53-109 ページ	マスター・グループのレプリケーション・アクティビティを中断します。
「SWITCH_MVIEW_MASTER プロシージャ」53-110 ページ	マテリアライズド・ビュー・グループのマスター・サイトを別のマスター・サイトに変更します。
「UNDO_ADD_NEW_MASTERS_REQUEST プロシージャ」53-111 ページ	指定した `extension_id` の `SPECIFY_NEW_MASTERS` プロシージャおよび `ADD_NEW_MASTERS` プロシージャで行った変更をすべて取り消します。
「UNREGISTER_MVIEW_REPGROUP プロシージャ」53-113 ページ	`DBA_REGISTERED_MVIEW_GROUPS` への挿入、変更または削除を行うことによって、各マスター・サイトおよびマスター・マテリアライズド・ビュー・サイトのマテリアライズド・ビュー管理を容易にします。
「VALIDATE ファンクション」53-114 ページ	マルチマスター・レプリケーション環境のキーのステータスが正しいかどうかを検証します。
「WAIT_MASTER_LOG プロシージャ」53-117 ページ	マスター・サイトに非同期で伝播された変更内容が適用されたかどうかを判断します。

ADD_GROUPED_COLUMN プロシージャ

このプロシージャは、既存の列グループにメンバーを追加します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.ADD_GROUPED_COLUMN (
   sname                 IN   VARCHAR2,
   oname                 IN   VARCHAR2,
   column_group          IN   VARCHAR2,
   list_of_column_names  IN   VARCHAR2 | DBMS_REPCAT.VARCHAR2s);
```

パラメータ

表 53-2 ADD_GROUPED_COLUMN プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	列グループが関連付けられているレプリケート表の名前。ネストした表の記憶表がこの表に該当します。
column_group	メンバーを追加する列グループの名前。
list_of_column_names	指定した列グループに追加する列の名前。列名は、カンマで区切られたリストまたは PL/SQL 索引付き表のいずれかで示します。PL/SQL 索引付き表は、DBMS_REPCAT.VARCHAR2S タイプにしてください。表内のすべての列を含んだ列グループを作成するには、単一の値 '*' を使用します。 列オブジェクトは指定できますが、列オブジェクトの属性は指定できません。 表がオブジェクトの場合、オブジェクト識別子列を列グループに追加するには SYS_NC_OID$ を指定します。オブジェクト識別子列では、各行オブジェクトのオブジェクト識別子が追跡されます。 表がネストした表の記憶表の場合、ネストした表の各行の識別子を追跡する列を追加するには NESTED_TABLE_ID を指定します。

表 53-3 ADD_GROUPED_COLUMN プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定した表が存在しません。
missinggroup	指定した列グループが存在しません。
missingcolumn	指定した列が、指定した表内に存在していません。
duplicatecolumn	指定した列は、すでに別の列グループのメンバーです。
missingschema	指定したスキーマが存在しません。
notquiesced	指定した表が属しているレプリケーション・グループが停止中ではありません。

ADD_MASTER_DATABASE プロシージャ

このプロシージャは、レプリケーション環境に別のマスター・サイトを追加します。また、すべてのトリガーと関連するパッケージを既存のマスター・サイトで再生成します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.ADD_MASTER_DATABASE (
   gname                 IN   VARCHAR2,
   master                IN   VARCHAR2,
   use_existing_objects  IN   BOOLEAN  := true,
   copy_rows             IN   BOOLEAN  := true,
   comment               IN   VARCHAR2 := '',
   propagation_mode      IN   VARCHAR2 := 'ASYNCHRONOUS',
   fname                 IN   VARCHAR2 := NULL);
```

パラメータ

表 53-4 ADD_MASTER_DATABASE プロシージャのパラメータ

パラメータ	説明
gname	レプリケートするレプリケーション・グループの名前。このレプリケーション・グループは、マスター定義サイトにすでに存在している必要があります。
master	新規マスター・データベースの完全修飾データベース名。
use_existing_objects	スキーマ内にすでに存在するオブジェクトと同じタイプ、同じ形式のオブジェクトを、新規マスター・サイトで再利用する場合は、TRUE を指定します。
copy_rows	新規マスター・サイトの表の初期の内容を、マスター定義サイトの表の内容と一致させる場合は TRUE を指定します。
comment	このコメントは DBA_REPSITES ビューの MASTER_COMMENT フィールドに追加されます。
propagation_mode	新規マスター・データベースとの間の変更内容の送受信方法を示します。受入れ値は synchronous および asynchronous です。
fname	内部使用のためのパラメータ。 **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

例外

表 53-5 ADD_MASTER_DATABASE プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	マスター・グループのレプリケーションは中断されていません。
missingrepgroup	レプリケーション・グループが、指定したデータベース・サイトに存在していません。
commfailure	新規マスターにアクセスできません。
typefailure	伝播モードの指定に誤りがあります。
notcompat	互換モードは 7.3.0.0 以上であることが必要です。
duplrepgrp	マスター・サイトがすでに存在しています。

ADD_NEW_MASTERS プロシージャ

このプロシージャでは、DBA_REPSITES_NEW データ・ディクショナリ・ビューのマスター・サイトが、SPECIFY_NEW_MASTERS プロシージャの実行時に指定したマスター・グループに追加されます。これらの新規マスター・サイトに関する情報が、使用可能なすべてのマスター・サイトのレプリケーション・カタログに追加されます。

オブジェクト・レベルのエクスポート / インポートでインスタンス化したマスター・サイトは、すべてこの時点でアクセスできるようにする必要があります。新しいレプリケーション・グループは、停止状態で追加されます。データベース全体のエクスポート / インポートまたは変更ベースのリカバリでインスタンス化したマスター・サイトを、アクセス可能にする必要はありません。

このプロシージャは、SPECIFY_NEW_MASTERS プロシージャの後に実行します。

注意： このプロシージャの実行後は、新規マスター・サイトを追加するまで遅延トランザクション・キューの伝播を使用禁止または使用可能にしないでください。伝播を使用禁止または使用可能にする場合は、あらかじめ DBA_REPEXTENSIONS データ・ディクショナリ・ビューを消去する必要があります。伝播を使用禁止または使用可能にするには、レプリケーション管理ツールまたは DBMS_DEFER_SYS パッケージの SET_DISABLED プロシージャを使用します。

関連項目： 53-107 ページ「SPECIFY_NEW_MASTERS プロシージャ」

構文

```
DBMS_REPCAT.ADD_NEW_MASTERS (
   export_required                IN    BOOLEAN,
   { available_master_list        IN    VARCHAR2,
   | available_master_table       IN    DBMS_UTILITY.DBLINK_ARRAY,}
   masterdef_flashback_scn        OUT   NUMBER,
   extension_id                   OUT   RAW,
   break_trans_to_masterdef       IN    BOOLEAN   := false,
   break_trans_to_new_masters     IN    BOOLEAN   := false,
   percentage_for_catchup_mdef    IN    BINARY_INTEGER  := 100,
   cycle_seconds_mdef             IN    BINARY_INTEGER  := 60,
   percentage_for_catchup_new     IN    BINARY_INTEGER  := 100,
   cycle_seconds_new              IN    BINARY_INTEGER  := 60);
```

注意： このプロシージャはオーバーロードされています。available_master_list パラメータと available_master_table パラメータは、両方同時には指定できません。

パラメータ

表 53-6 ADD_NEW_MASTERS プロシージャのパラメータ

パラメータ	説明
export_required	新規マスター・サイトの少なくとも 1 つで、オブジェクトレベルのデータベース・エクスポートまたはデータベース全体のエクスポートを行う必要がある場合は、TRUE に設定します。すべての新規マスター・サイトで変更ベースのリカバリを使用する場合は、FALSE に設定します。
available_master_list	オブジェクトレベルのエクスポート / インポートを使用してインスタンス化する新規マスター・サイトのカンマで区切られたリスト。リストに示されたサイトと SPECIFY_NEW_MASTERS プロシージャで指定したサイトは一致する必要があります。新規マスター・サイトのみをリストし、既存のマスター・サイトはリストしません。各サイト名の間には空白を入れないでください。 データベース全体のエクスポート / インポートまたは変更ベースのリカバリを使用してすべてのマスターをインスタンス化する場合は NULL を指定します。
available_master_table	オブジェクトレベルのエクスポート / インポートを使用してインスタンス化する新規マスター・サイトがリストされた表。表のサイトと SPECIFY_NEW_MASTERS プロシージャで指定したサイトは一致する必要があります。データベース全体のエクスポート / インポートまたは変更ベースのリカバリを使用してインスタンス化するマスターは指定しないでください。 オブジェクトレベルのエクスポート / インポートを使用してインスタンス化するマスター・サイトがリストされた表では、拡張対象のマスター・グループの新規マスター・サイトのみをリストします。拡張対象のマスター・グループの既存のマスター・サイトはリストしないでください。最初のマスター・サイトは位置 1、2 番目は位置 2、以下同様に設定されている必要があります。

表 53-6 ADD_NEW_MASTERS プロシージャのパラメータ（続き）

パラメータ	説明
masterdef_flashback_scn	この OUT パラメータは、エクスポート時または変更ベースのリカバリ時に使用するシステム変更番号（SCN）を戻します。このパラメータから戻された値は、エクスポートの実行時に FLASHBACK_SCN エクスポート・パラメータで使用します。flashback_scn 値を検索するには、DBA_REPEXTENSIONS データ・ディクショナリ・ビューに問い合せます。
extension_id	この OUT パラメータは、マスター・データベースを停止せずに追加するための現行の保留中の要求の識別子を戻します。extension_id 値を検索するには、DBA_REPSITES_NEW データ・ディクショナリ・ビューおよび DBA_REPEXTENSIONS データ・ディクショナリ・ビューに問い合せます。
break_trans_to_masterdef	このパラメータが意味を持つのは、export_required を TRUE に設定した場合のみです。 break_trans_to_masterdef を TRUE に設定すると、既存のマスターが、マスター・サイトを追加しないレプリケーション・グループのマスター定義サイトに、遅延トランザクションの伝播を継続する場合があります。マスター・サイトを追加するレプリケーション・グループの遅延トランザクションは、エクスポートが完了するまで伝播できません。 各遅延トランザクションは、1 つ以上のリモート・プロシージャ・コール（RPC）で構成されます。FALSE に設定し、影響を受けないマスター・グループおよび拡張対象のマスター・グループの両方のオブジェクトを参照するトランザクションが発生した場合、そのトランザクションが 2 分割されて 2 つのトランザクションとなり、異なるタイミングで宛先に送信されることがあります。このようなトランザクションを分割トランザクションと呼びます。分割トランザクションが使用可能な場合は、新規マスター・サイトを追加するまで、このような動作が違反となる可能性がある整合性制約を無効にする必要があります。 break_trans_to_masterdef を FALSE に設定した場合、既存のマスターは遅延トランザクションをマスター定義サイトに伝播できません。

表 53-6 ADD_NEW_MASTERS プロシージャのパラメータ（続き）

パラメータ	説明
break_trans_to_new_masters	break_trans_to_new_masters を TRUE に設定すると、既存のマスター・サイトが、マスター・サイトを追加しないレプリケーション・グループの新規マスター・サイトに、遅延トランザクションの伝播を継続する場合があります。 各遅延トランザクションは、1 つ以上のリモート・プロシージャ・コール（RPC）で構成されます。TRUE に設定し、影響を受けないマスター・グループおよび拡張対象のマスター・グループの両方のオブジェクトを参照するトランザクションが発生した場合、そのトランザクションが 2 分割されて 2 つのトランザクションとなり、異なるタイミングで宛先に送信されることがあります。このようなトランザクションを分割トランザクションと呼びます。分割トランザクションが使用可能な場合は、新規マスター・サイトを追加するまで、このような動作が違反となる可能性がある整合性制約を無効にする必要があります。 break_trans_to_new_masters を FALSE に設定した場合、新しいマスターへの遅延トランザクション・キューの伝播は無効となります。
percentage_for_catchup_mdef	このパラメータが意味を持つのは、export_required および break_trans_to_masterdef の両方を TRUE に設定した場合のみです。 マスター定義サイトへの伝播の捕捉に使用する伝播リソースのパーセント。10 の倍数で、かつ 0 ～ 100 の間に設定する必要があります。
cycle_seconds_mdef	このパラメータが意味を持つのは、percentage_for_catchup_mdef が意味を持ち、かつ値が 10 ～ 90 の範囲に設定されている場合です。その場合、masterdef への伝播は、拡張対象のレプリケーション・グループと拡張対象でないレプリケーション・グループとの間で、各サイクル 1 送信ずつ切り替わります。このパラメータは、そのサイクルの長さを秒で示します。
percentage_for_catchup_new	このパラメータが意味を持つのは、break_trans_to_masters を TRUE に設定した場合のみです。 新規マスター・サイトへの伝播の捕捉に使用する伝播リソースのパーセント。10 の倍数で、かつ 0 ～ 100 の間に設定する必要があります。

表 53-6 ADD_NEW_MASTERS プロシージャのパラメータ（続き）

パラメータ	説明
cycle_seconds_new	このパラメータが意味を持つのは、percentage_for_catchup_new が意味を持ち、かつ値が 10 ～ 90 の範囲に設定されている場合です。その場合、新規マスターへの伝播は、拡張対象のレプリケーション・グループと拡張対象でないレプリケーション・グループとの間で、各サイクル 1 送信ずつ切り替わります。このパラメータは、そのサイクルの長さを秒で示します。

例外

表 53-7 ADD_NEW_MASTERS プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
typefailure	パラメータに指定したパラメータ値が適切ではありません。
novalidextreq	有効な拡張要求がありません。extension_id は有効ではありません。
nonewsites	指定した拡張要求に追加する新規マスター・サイトがありません。
notanewsite	拡張要求の新規サイトではありません。SPECIFY_NEW_MASTERS プロシージャの実行時には指定していないサイトが指定されました。
dbnotcompatible	機能がデータベースのバージョンと互換性がありません。データベースはいずれも 9.0.1 以上の互換性レベルを持つものにする必要があります。

使用上の注意

変更ベースのリカバリまたはデータベース全体のエクスポート / インポートを使用してインスタンス化する新規マスター・サイトに対しては、次の条件が適用されます。

- 新規マスター・サイトに既存のレプリケーション・グループを含めることはできません。
- マスター定義サイトにマテリアライズド・ビュー・グループを含めることはできません。
- マスター定義サイトはすべてのマスター・グループで同じにする必要があります。1 つ以上のマスター・グループのマスター定義サイトが異なる場合は、変更ベースのリカバリおよびデータベース全体のエクスポート / インポートを使用せず、オブジェクトレベルのエクスポート / インポートを使用してください。

- 新規マスター・サイトには、拡張プロセスの完了時にマスター定義サイトの全レプリケーション・グループが含まれている必要があります。つまり、マスター定義サイトのマスター・グループのサブセットは新規マスター・サイトに追加できません。すべてのグループを追加する必要があります。

> **注意：** 変更ベースのリカバリを使用するには、同じオペレーティング・システムで既存のマスター・サイトおよび新規マスター・サイトを実行する必要があります。ただし、オペレーティング・システムのリリースは異なっていてもかまいません。

オブジェクトレベルのエクスポート / インポートの場合は、インポートの前に、拡張グループの DBA_REPCATLOG データ・ディクショナリ・ビューの全要求がエラーなしで処理されたことを確認してください。

ADD_PRIORITY_*datatype* プロシージャ

このプロシージャは、優先グループにメンバーを追加します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、priority 列のデータ・タイプによって決まります。このプロシージャは、priority 列の有効な値ごとに 1 回ずつコールしてください。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.ADD_PRIORITY_datatype (
   gname              IN   VARCHAR2,
   pgroup             IN   VARCHAR2,
   value              IN   datatype,
   priority           IN   NUMBER);
```

datatype には次のいずれかを指定します。

```
{ NUMBER
| VARCHAR2
| CHAR
| DATE
| RAW
| NCHAR
| NVARCHAR2 }
```

パラメータ

表 53-8 ADD_PRIORITY_*datatype* プロシージャのパラメータ

パラメータ	説明
gname	優先グループを作成するマスター・グループ。
pgroup	優先グループの名前。
value	優先グループ・メンバーの値。この値は、この優先グループを使用している表の priority 列に関連する候補値の 1 つです。
priority	この値の優先順位。数値が大きいほど優先順位は高くなります。

例外

表 53-9 ADD_PRIORITY_*datatype* プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
duplicatevalue	指定した値が、優先グループ内にすでに存在しています。
duplicatepriority	指定した優先順位が優先グループ内にすでに存在しています。
missingrepgroup	指定したマスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。
typefailure	優先グループに対して指定した値のデータ・タイプが正しくありません。
notquiesced	指定したマスター・グループが停止中ではありません。

ADD_SITE_PRIORITY_SITE プロシージャ

このプロシージャは、サイト優先グループに新規サイトを追加します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (
   gname          IN   VARCHAR2,
   name           IN   VARCHAR2
   site           IN   VARCHAR2,
   priority       IN   NUMBER);
```

パラメータ

表 53-10 ADD_SITE_PRIORITY_SITE プロシージャのパラメータ

パラメータ	説明
gname	サイトをグループに追加するマスター・グループ。
name	メンバーを追加するサイト優先グループの名前。
site	追加するサイトのグローバル・データベース名。
priority	追加するサイトの優先順位レベル。数値が大きいほど優先順位レベルは高くなります。

例外

表 53-11 ADD_SITE_PRIORITY_SITE プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
missingpriority	指定したサイト優先グループが存在しません。
duplicatepriority	指定した優先順位レベルが、グループ内の別のサイト用にすでに存在しています。
duplicatevalue	指定したサイトが、サイト優先グループ内にすでに存在しています。
notquiesced	マスター・グループが停止中ではありません。

ADD_*conflicttype*_RESOLUTION プロシージャ

このプロシージャは、更新、削除または一意性競合の解消方法を指定します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、ルーチンが解消する競合のタイプによって決まります。

表 53-12 ADD_conflicttype_RESOLUTION プロシージャ

競合のタイプ	プロシージャ名
update	ADD_UPDATE_RESOLUTION
uniqueness	ADD_UNIQUE_RESOLUTION
delete	ADD_DELETE_RESOLUTION

関連項目： 更新競合の解消方法の指定、一意性競合の解消方法の選択、および削除競合の解消方法の割当ての詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
   sname                  IN   VARCHAR2,
   oname                  IN   VARCHAR2,
   column_group           IN   VARCHAR2,
   sequence_no            IN   NUMBER,
   method                 IN   VARCHAR2,
   parameter_column_name  IN   VARCHAR2
                                | DBMS_REPCAT.VARCHAR2s
                                | DBMS_UTILITY.LNAME_ARRAY,
   priority_group         IN   VARCHAR2     := NULL,
   function_name          IN   VARCHAR2     := NULL,
   comment                IN   VARCHAR2     := NULL);

DBMS_REPCAT.ADD_DELETE_RESOLUTION (
   sname                  IN   VARCHAR2,
   oname                  IN   VARCHAR2,
   sequence_no            IN   NUMBER,
   parameter_column_name  IN   VARCHAR2 | DBMS_REPCAT.VARCHAR2s,
   function_name          IN   VARCHAR2,
   comment                IN   VARCHAR2     := NULL
   method                 IN   VARCHAR2     := 'USER FUNCTION');
```

```
DBMS_REPCAT.ADD_UNIQUE_RESOLUTION(
   sname                 IN   VARCHAR2,
   oname                 IN   VARCHAR2,
   constraint_name       IN   VARCHAR2,
   sequence_no           IN   NUMBER,
   method                IN   VARCHAR2,
   parameter_column_name IN   VARCHAR2
                              | DBMS_REPCAT.VARCHAR2s
                              | DBMS_UTILITY.LNAME_ARRAY,
   function_name         IN   VARCHAR2     := NULL,
   comment               IN   VARCHAR2     := NULL);
```

パラメータ

表 53-13 ADD_*conflicttype*_RESOLUTION プロシージャのパラメータ

パラメータ	説明
sname	レプリケートする表が含まれているスキーマの名前。
oname	競合解消ルーチンを追加する表の名前。ネストした表の記憶表がこの表に該当します。
column_group	競合解消ルーチンを追加する列グループの名前。更新競合解消ルーチンのみが列グループを必要とします。
constraint_name	競合解消ルーチンを追加する一意制約または一意索引の名前。一意索引の名前が関連する一意制約の名前と異なる場合は、一意索引の名前を使用します。一意性競合解消ルーチンのみ制約名を必要とします。
sequence_no	指定した競合解消方法を適用する順序。

表 53-13 ADD_*conflicttype*_RESOLUTION プロシージャのパラメータ（続き）

パラメータ	説明
method	作成する競合解消ルーチンのタイプ。アドバンスト・レプリケーションで提供されている標準ルーチンのいずれかの名前を指定できます。または、独自のルーチンを作成している場合は、USER FUNCTION を選択し、FUNCTION_NAME パラメータにそのメソッド名を指定してください。 このリリースでサポートされている更新競合の標準メソッドは次のとおりです。 ■ minimum ■ maximum ■ latest timestamp ■ earliest timestamp ■ additive, average ■ priority group ■ site priority ■ overwrite ■ discard このリリースでサポートされている一意性競合の標準メソッドは、append site name、append sequence および discard です。削除競合の（Oracle 提供の）組込み方法はありません。

表 53-13 ADD_*conflicttype*_RESOLUTION プロシージャのパラメータ（続き）

パラメータ	説明
parameter_column_name	競合の解消に使用する列の名前。標準メソッドでは、単一の列で操作が行われます。たとえば、列グループに対して latest timestamp メソッドを使用している場合は、このパラメータとしてタイムスタンプを含む列の名前を渡す必要があります。user function を使用している場合は、任意の数の列を使用して競合を解消できます。 更新競合または一意性競合の場合、このパラメータは、列名のカンマで区切られたリスト、または DBMS_REPCAT.VARCHAR2S タイプまたは DBMS_UTILITY.LNAME_ARRAY タイプの PL/SQL 索引付き表のいずれかを受け入れます。列オブジェクトの属性を指定すると、30 バイト以上の列名が発生することがありますが、その場合は DBMS_UTILITY.LNAME_ARRAY を使用します。 削除競合の場合、このパラメータは、列名のカンマで区切られたリスト、または DBMS_REPCAT.VARCHAR2S タイプの PL/SQL 索引付き表のいずれかを受け入れます。 値 '*' は、競合の解消に表内のすべての列（または更新競合の場合は列グループ）を使用することを意味します。'*' を指定すると、列はユーザー・ファンクションにアルファベット順で渡されます。 このパラメータには LOB 列は指定できません。 **関連項目：**列オブジェクトを使用する場合は、53-23 ページの「使用上の注意」を参照してください。
priority_group	priority group または site priority の更新競合解消のメソッドを使用している場合は、自分で作成した優先グループの名前を指定する必要があります。 詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。他のメソッドを使用している場合は、このパラメータのデフォルト値 NULL を使用できます。このパラメータは更新競合の場合のみ適用できます。
function_name	user function メソッドを選択した場合、または削除競合解消ルーチンを追加する場合は、自分で作成した競合解消ルーチンの名前を指定する必要があります。標準メソッドの 1 つを使用している場合は、このパラメータのデフォルト値 NULL を使用できます。
comment	DBA_REPRESOLUTION ビューに追加されるユーザー・コメント。

例外

表 53-14 ADD_*conflicttype*_RESOLUTION プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーションを使用する表として指定のスキーマ内に存在していません。
missingschema	指定したスキーマが存在しません。
missingcolumn	parameter_column_name パラメータの一部としてユーザーが指定した列が存在しません。
missinggroup	指定した列グループが存在しません。
missingprioritygroup	ユーザーが指定した優先グループがその表に存在していません。
invalidmethod	ユーザーが指定した解消メソッドは認識されていません。
invalidparameter	parameter_column_name パラメータに指定した列の数が正しくありません（標準ルーチンは列名を 1 つしか使用しません）。
missingfunction	ユーザーが指定したユーザー・ファンクションが存在しません。
missingconstraint	一意性競合に対してユーザーが指定した制約が存在しません。
notquiesced	指定した表が属しているレプリケーション・グループが停止中ではありません。
duplicateresolution	指定した競合解消のメソッドは、すでに登録されています。
duplicatesequence	指定オブジェクトには指定した順序番号がすでに存在しています。
invalidprioritygroup	指定した優先グループが存在しません。
paramtype	タイプが、優先グループに割り当てられたタイプと異なります。

使用上の注意

列オブジェクトを使用する場合、parameter_column_name パラメータに列オブジェクトの属性を指定できるかどうかは、競合解消方法が（Oracle 提供の）組込みのものであるかユーザーが作成したものであるかどうかによって決まります。

- 組込みの競合解消方法を使用する場合、オブジェクトの属性をこのパラメータに指定できます。たとえば、cust_address という名前の列オブジェクトが属性として street_address を持つ場合、このパラメータには cust_address.street_address を指定できます。
- 組込みの競合解消方法を使用する場合、列オブジェクトの LOB 属性、列オブジェクトのコレクションまたはコレクション属性、REF または列オブジェクト全体をこのパラメータに指定できません。
- ユーザー作成の競合解消方法を使用する場合は、列オブジェクト全体を指定する必要があります。列オブジェクトの属性を指定することはできません。たとえば、cust_address という名前の列オブジェクトが（他の属性間で）属性として street_address を持つ場合、このパラメータに指定できるのは cust_address のみです。

ALTER_CATCHUP_PARAMETERS プロシージャ

このプロシージャは、DBA_REPEXTENSIONS データ・ディクショナリ・ビューに格納された次のパラメータの値を変更します。

- percentage_for_catchup_mdef
- cycle_seconds_mdef
- percentage_for_catchup_new
- cycle_seconds_new

これらのパラメータは、最初に ADD_NEW_MASTERS プロシージャで設定されています。これらのパラメータに指定した新しい値は、新規マスター・サイトをマスター・グループに追加する残りのステップで使用されます。これらの変更は、実行するサイトにのみ適用されます。したがって、すべてのサイトでパラメータを変更する場合は、マスター定義サイトも含め各マスター・サイトで実行する必要があります。

関連項目： 53-10 ページ「ADD_NEW_MASTERS プロシージャ」

構文

```
DBMS_REPCAT.ALTER_CATCHUP_PARAMETERS (
   extension_id                  IN    RAW,
   percentage_for_catchup_mdef   IN    BINARY_INTEGER  := NULL,
   cycle_seconds_mdef            IN    BINARY_INTEGER  := NULL,
   percentage_for_catchup_new    IN    BINARY_INTEGER  := NULL,
   cycle_seconds_new             IN    BINARY_INTEGER  := NULL);
```

パラメータ

表 53-15 ALTER_CATCHUP_PARAMETERS プロシージャのパラメータ

パラメータ	説明
extension_id	マスター・データベースを停止せずに追加するための現行の保留中の要求の識別子。extension_id 値を検索するには、DBA_REPSITES_NEW データ・ディクショナリ・ビューおよび DBA_REPEXTENSIONS データ・ディクショナリ・ビューに問い合せます。
percentage_for_catchup_mdef	マスター定義サイトへの伝播の捕捉に使用する伝播リソースのパーセント。10 の倍数で、かつ 0 ～ 100 の間に設定する必要があります。
cycle_seconds_mdef	このパラメータが意味を持つのは、percentage_for_catchup_mdef が意味を持ち、かつ値が 10 ～ 90 の範囲に設定されている場合です。その場合、masterdef への伝播は、拡張対象のレプリケーション・グループと拡張対象でないレプリケーション・グループとの間で、各サイクル 1 送信ずつ切り替わります。このパラメータは、そのサイクルの長さを秒で示します。
percentage_for_catchup_new	新規マスター・サイトへの伝播の捕捉に使用する伝播リソースのパーセント。10 の倍数で、かつ 0 ～ 100 の間に設定する必要があります。
cycle_seconds_new	このパラメータが意味を持つのは、percentage_for_catchup_new が意味を持ち、かつ値が 10 ～ 90 の範囲に設定されている場合です。その場合、新規マスターへの伝播は、拡張対象のレプリケーション・グループと拡張対象でないレプリケーション・グループとの間で、各サイクル 1 送信ずつ切り替わります。このパラメータは、そのサイクルの長さを秒で示します。

例外

表 53-16 ALTER_CATCHUP_PARAMETERS プロシージャの例外

例外	説明
typefailure	パラメータに指定したパラメータ値が適切ではありません。
dbnotcompatible	機能がデータベースのバージョンと互換性がありません。データベースはいずれも 9.0.1 以上の互換性レベルを持つものにする必要があります。

ALTER_MASTER_PROPAGATION プロシージャ

このプロシージャは、指定したマスター・サイトで指定したレプリケーション・グループの伝播方法を変更します。レプリケーション・グループは停止中にしておいてください。このプロシージャは、マスター定義サイトからコールする必要があります。マスターが dblink_list または dblink_table に含まれている場合、そのデータベース・リンクは ALTER_MASTER_PROPAGATION によって無視されます。マスターから同じマスター自身への伝播モードは変更できません。

構文

```
DBMS_REPCAT.ALTER_MASTER_PROPAGATION (
   gname               IN   VARCHAR2,
   master              IN   VARCHAR2,
   { dblink_list       IN   VARCHAR2,
   | dblink_table      IN   dbms_utility.dblink_array,}
   propagation_mode    IN   VARCHAR2 : ='asynchronous',
   comment             IN   VARCHAR2 := '');
```

注意： このプロシージャはオーバーロードされています。dblink_list パラメータと dblink_table パラメータは、両方同時には指定できません。

パラメータ

表 53-17 ALTER_MASTER_PROPAGATION プロシージャのパラメータ

パラメータ	説明
gname	伝播モードを変更するレプリケーション・グループの名前。
master	伝播モードを変更するマスター・サイトの名前。
dblink_list	伝播方法を変更するデータベース・リンクのカンマで区切られたリスト。NULL の場合は、変更対象のマスター・サイト以外のすべてのマスターがデフォルトとして使用されます。
dblink_table	伝播を変更するデータベース・リンクの PL/SQL 索引付き表（位置 1 から索引付け）。
propagation_mode	指定したマスター・サイトの変更内容を、データベース・リンクのリストで指定したサイトに伝播する方法を指定します。適切な値は synchronous および asynchronous です。
comment	DBA_REPPROP ビューに追加されるコメント。

例外

表 53-18 ALTER_MASTER_PROPAGATION プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	起動サイトが停止中ではありません。
typefailure	指定した伝播モードは認識されませんでした。
nonmaster	データベース・リンクのリストに、マスター・サイト以外のサイトが含まれています。

ALTER_MASTER_REPOBJECT プロシージャ

このプロシージャは、レプリケーション環境内のオブジェクトを変更します。このプロシージャは、マスター定義サイトからコールする必要があります。

このプロシージャでは、次の条件のいずれかが真である場合、オブジェクトのマスター・グループを停止する必要があります。

- マルチマスター・レプリケーション環境で表を変更中である。
- safe_table_change パラメータが FALSE に設定された表を、単一マスター・レプリケーション環境で変更中である。

マスター・グループを停止せずに表以外のオブジェクトを変更するには、このプロシージャを使用します。

構文

```
DBMS_REPCAT.ALTER_MASTER_REPOBJECT (
   sname               IN   VARCHAR2,
   oname               IN   VARCHAR2,
   type                IN   VARCHAR2,
   ddl_text            IN   VARCHAR2,
   comment             IN   VARCHAR2     := '',
   retry               IN   BOOLEAN      := false
   safe_table_change   IN   BOOLEAN      := false);
```

パラメータ

表 53-19 ALTER_MASTER_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	変更するオブジェクトを含んだスキーマ。
oname	変更するオブジェクトの名前。ネストした表の記憶表はこのオブジェクトに該当しません。
type	変更するオブジェクトのタイプ。次のタイプがサポートされています。 FUNCTION　SYNONYM INDEX　TABLE INDEXTYPE　TRIGGER OPERATOR　TYPE PACKAGE　TYPE BODY PACKAGE BODY　VIEW PROCEDURE
ddl_text	オブジェクトの変更に使用する DDL テキスト。この DDL は、適用前には解析されません。したがって、変更するオブジェクトに対する適切なスキーマ名およびオブジェクト名を DDL テキストで使用していることを確認してください。 スキーマを指定しないで DDL を発行すると、レプリケーション管理者のスキーマがデフォルトのスキーマとなります。レプリケーション管理者のスキーマを使用しない場合は、スキーマを必ず指定してください。
comment	NULL 以外の場合、このコメントは DBA_REPOBJECT ビューの COMMENT フィールドに追加されます。
retry	retry が TRUE の場合は、オブジェクトの状態が VALID 以外のマスターにおいてのみ、オブジェクトが ALTER_MASTER_REPOBJECT によって変更されます。

表 53-19 ALTER_MASTER_REPOBJECT プロシージャのパラメータ（続き）

パラメータ	説明
safe_table_change	表の変更が安全に実行できる場合は TRUE を指定します。表の変更が安全にできない場合は FALSE を指定します。 単一のマスター・レプリケーション環境ではマスター表の変更を、その表を含むマスター・グループを停止せずに安全に実行できます。安全でない変更を実行する場合は、マスター・グループを停止する必要があります。 このパラメータは、単一のレプリケーション環境の表にのみ指定してください。マルチマスター・レプリケーション環境や、指定したオブジェクトが表でない場合、このパラメータは無視されます。マルチマスター・レプリケーション環境では、マスター・グループを停止して、表で ALTER_MASTER_REPOBJECT プロシージャを実行する必要があります。 安全な変更を次に示します。 ■ 記憶域およびエクステントの情報の変更 ■ VARCHAR2(20) 列を VARCHAR2(50) 列に変更するなど、既存の列の拡大 ■ 非主キー制約の追加 ■ 非主キー制約の変更 ■ 非主キー制約の有効化と無効化 安全でない変更を次に示します。 ■ キーの列を追加または削除することによる主キーの変更 ■ 列の追加または削除 ■ VARCHAR2(50) 列を VARCHAR2(20) 列に変更するなど、既存の列の縮小 ■ 主キー制約の無効化 ■ 既存の列のデータ・タイプの変更 ■ 既存の列の削除 変更が安全か安全でないか不明な場合は、マスター・グループを停止してから ALTER_MASTER_REPOBJECT プロシージャを実行します。

例外

表 53-20 ALTER_MASTER_REPOBJECT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	関連するレプリケーション・グループが中断されていません。
missingobject	sname および oname に該当するオブジェクトが存在しません。
typefailure	指定したタイプ・パラメータはサポートされていません。
ddlfailure	マスター定義サイトで DDL が成功しませんでした。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

ALTER_MVIEW_PROPAGATION プロシージャ

このプロシージャは、現行のマテリアライズド・ビュー・サイトで指定したレプリケーション・グループの伝播方法を変更します。また、マテリアライズド・ビュー・サイトで遅延トランザクション・キューを送信し、マテリアライズド・ビューのベース表をロックして、トリガーおよびトリガーと関連するパッケージを再生成します。このプロシージャは、マテリアライズド・ビュー・サイトからコールする必要があります。

構文

```
DBMS_REPCAT.ALTER_MVIEW_PROPAGATION (
   gname                IN  VARCHAR2,
   propagation_mode     IN  VARCHAR2,
   comment              IN  VARCHAR2   := ''
   gowner               IN  VARCHAR2   := 'PUBLIC');
```

パラメータ

表 53-21 ALTER_MVIEW_PROPAGATION プロシージャのパラメータ

パラメータ	説明
gname	伝播方法を変更するレプリケーション・グループの名前。
propagation_mode	現行のマテリアライズド・ビュー・サイトの変更内容を、関連するマスター・サイトあるいはマスター・マテリアライズド・ビュー・サイトに伝播する方法。適切な値は synchronous および asynchronous です。
comment	DBA_REPPROP ビューに追加されるコメント。
gowner	マテリアライズド・ビュー・グループの所有者。

例外

表 53-22 ALTER_MVIEW_PROPAGATION プロシージャの例外

例外	説明
missingrepgroup	指定したレプリケーション・グループが存在しません。
typefailure	伝播モードの指定に誤りがあります。
nonmview	現行のサイトが、指定したレプリケーション・グループのマテリアライズド・ビュー・サイトではありません。
commfailure	マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトと接続できません。
notcompat	互換モードは 7.3.0.0 以上であることが必要です。
failaltermviewrop	マテリアライズド・ビュー・グループの伝播が変更できるのは、マテリアライズド・ビュー・サイトを共有する同一のマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトを持つ、マテリアライズド・ビュー・グループが他にない場合のみです。

ALTER_PRIORITY プロシージャ

このプロシージャは、指定した優先グループ・メンバーに関連付けられている優先順位レベルを変更します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.ALTER_PRIORITY (
   gname            IN   VARCHAR2,
   pgroup           IN   VARCHAR2,
   old_priority     IN   NUMBER,
   new_priority     IN   NUMBER);
```

パラメータ

表 53-23 ALTER_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているマスター・グループ。
pgroup	変更する優先順位を含んだ優先グループの名前。
old_priority	優先グループ・メンバーの現行の優先順位レベル。
new_priority	優先グループ・メンバーに割り当てる新しい優先順位レベル。

例外

表 53-24 ALTER_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
duplicatepriority	新しい優先順位レベルが、優先グループ内にすでに存在しています。
missingrepgroup	指定したマスター・グループが存在しません。
missingvalue	値が、DBMS_REPCAT.ADD_PRIORITY_*datatype* のコールで登録されていません。
missingprioritygroup	指定した優先グループが存在しません。
notquiesced	指定したマスター・グループが停止中ではありません。

ALTER_PRIORITY_*datatype* プロシージャ

このプロシージャは、優先グループ内のメンバーの値を変更します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、priority 列のデータ・タイプによって決まります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.ALTER_PRIORITY_datatype (
   gname        IN   VARCHAR2,
   pgroup       IN   VARCHAR2,
   old_value    IN   datatype,
   new_value    IN   datatype);
```

datatype には次のいずれかを指定します。

```
{ NUMBER
| VARCHAR2
| CHAR
| DATE
| RAW
| NCHAR
| NVARCHAR2 }
```

パラメータ

表 53-25 ALTER_PRIORITY_*datatype* プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているマスター・グループ。
pgroup	変更する値を含んだ優先グループの名前。
old_value	優先グループ・メンバーの現行の値。
new_value	優先グループ・メンバーに割り当てる新規の値。

例外

表 53-26 ALTER_PRIORITY_*datatype* プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
duplicatevalue	新しい値が、優先グループ内にすでに存在しています。
missingrepgroup	指定したマスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。
missingvalue	元の値が存在しません。
paramtype	優先グループに対する新しい値のデータ・タイプが正しくありません。
typefailure	優先グループに対して指定した値のデータ・タイプが正しくありません。
notquiesced	指定したマスター・グループが停止中ではありません。

ALTER_SITE_PRIORITY プロシージャ

このプロシージャは、指定したサイトに関連付けられている優先順位レベルを変更します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.ALTER_SITE_PRIORITY (
   gname         IN   VARCHAR2,
   name          IN   VARCHAR2,
   old_priority  IN   NUMBER,
   new_priority  IN   NUMBER);
```

パラメータ

表 53-27 ALTER_SITE_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループが関連付けられているマスター・グループ。
name	メンバーを変更するサイト優先グループの名前。
old_priority	優先順位レベルを変更するサイトの現行の優先順位レベル。
new_priority	サイトの新しい優先順位レベル。数値が大きいほど優先順位レベルは高くなります。

例外

表 53-28 ALTER_SITE_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
missingpriority	元の優先順位レベルが、どのグループ・メンバーにも関連付けられていません。
duplicatepriority	新しい優先順位レベルが、グループ内の別のサイト用にすでに存在しています。
missingvalue	元の値が存在しません。
paramtype	優先グループに対する新しい値のデータ・タイプが正しくありません。
notquiesced	マスター・グループが停止中ではありません。

ALTER_SITE_PRIORITY_SITE プロシージャ

このプロシージャは、指定した優先順位レベルに関連付けられているサイトを変更します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.ALTER_SITE_PRIORITY_SITE (
   gname     IN   VARCHAR2,
   name      IN   VARCHAR2,
   old_site  IN   VARCHAR2,
   new_site  IN   VARCHAR2);
```

パラメータ

表 53-29 ALTER_SITE_PRIORITY_SITE プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループが関連付けられているマスター・グループ。
name	メンバーを変更するサイト優先グループの名前。
old_site	優先順位レベルから分離するサイトの現行のグローバル・データベース名。
new_site	現行の優先順位レベルと関連付ける新しいグローバル・データベース名。

例外

表 53-30 ALTER_SITE_PRIORITY_SITE プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
missingpriority	指定したサイト優先グループが存在しません。
missingvalue	旧サイトがグループ・メンバーではありません。
notquiesced	マスター・グループが停止中ではありません。

CANCEL_STATISTICS プロシージャ

このプロシージャは、表の更新競合、一意性競合および削除競合の正常な解消に関する統計の収集を停止します。

構文

```
DBMS_REPCAT.CANCEL_STATISTICS (
   sname     IN   VARCHAR2,
   oname     IN   VARCHAR2);
```

パラメータ

表 53-31 CANCEL_STATISTICS プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマの名前。
oname	競合解消統計の収集を停止する表の名前。

例外

表 53-32 CANCEL_STATISTICS プロシージャの例外

例外	説明
missingschema	指定したスキーマが存在しません。
missingobject	指定した表が存在しません。
statnotreg	現在、指定した表は統計収集用に登録されていません。

COMMENT_ON_COLUMN_GROUP プロシージャ

このプロシージャは、列グループの DBA_REPCOLUMN_GROUP ビューのコメント・フィールドを更新します。このコメントは、次回の DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT コール後に、すべてのマスター・サイトで追加されます。

構文

```
DBMS_REPCAT.COMMENT_ON_COLUMN_GROUP (
   sname           IN   VARCHAR2,
   oname           IN   VARCHAR2,
   column_group    IN   VARCHAR2,
   comment         IN   VARCHAR2);
```

パラメータ

表 53-33 COMMENT_ON_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。
oname	列グループが関連付けられているレプリケート表の名前。
column_group	列グループの名前。
comment	DBA_REPCOLUMN_GROUP ビューの GROUP_COMMENT フィールドに含める更新したコメントのテキスト。

例外

表 53-34 COMMENT_ON_COLUMN_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missinggroup	指定した列グループが存在しません。
missingobj	オブジェクトが見つかりません。

COMMENT_ON_MVIEW_REPSITES プロシージャ

このプロシージャは、指定したマテリアライズド・ビュー・グループの DBA_REPGROUP データ・ディクショナリ・ビューの SCHEMA_COMMENT フィールドを更新します。グループ名は、レプリケート・マテリアライズド・ビュー・グループとしてローカルで登録してください。このプロシージャは、マテリアライズド・ビュー・サイトで実行する必要があります。

構文

```
DBMS_REPCAT.COMMENT_ON_MVIEW_REPSITES (
   gowner     IN   VARCHAR2,
   gname      IN   VARCHAR2,
   comment    IN   VARCHAR2);
```

パラメータ

表 53-35 COMMENT_ON_MVIEW_REPSITES プロシージャのパラメータ

パラメータ	説明
gowner	マテリアライズド・ビュー・グループの所有者。
gname	マテリアライズド・ビュー・グループの名前。
comment	DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドに含める更新したコメント。

表 53-36 COMMENT_ON_MVIEW_REPSITES プロシージャの例外

パラメータ	説明
missingrepgroup	マテリアライズド・ビュー・グループが存在しません。
nonmview	接続サイトがマテリアライズド・ビュー・サイトではありません。

COMMENT_ON_PRIORITY_GROUP/COMMENT_ON_SITE_PRIORITY プロシージャ

COMMENT_ON_PRIORITY_GROUP は、優先グループの DBA_REPPRIORITY_GROUP ビューのコメント・フィールドを更新します。このコメントは、次回の GENERATE_REPLICATION_SUPPORT コール後に、すべてのマスター・サイトに追加されます。

COMMENT_ON_SITE_PRIORITY は、サイト優先グループの DBA_REPPRIORITY_GROUP ビューのコメント・フィールドを更新します。このプロシージャは、COMMENT_ON_COLUMN_GROUP プロシージャに対するラッパーで、便宜上の目的で提供されています。このプロシージャは、マスター定義サイトで発行する必要があります。

構文

```
DBMS_REPCAT.COMMENT_ON_PRIORITY_GROUP (
   gname        IN   VARCHAR2,
   pgroup       IN   VARCHAR2,
   comment      IN   VARCHAR2);

DBMS_REPCAT.COMMENT_ON_SITE_PRIORITY (
   gname        IN   VARCHAR2,
   name         IN   VARCHAR2,
   comment      IN   VARCHAR2);
```

パラメータ

表 53-37 COMMENT_ON_PRIORITY_GROUP および COMMENT_ON_SITE_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	マスター・グループの名前。
pgroup/name	優先グループまたはサイト優先グループの名前。
comment	DBA_REPPRIORITY_GROUP ビューの PRIORITY_COMMENT フィールドに含める更新したコメントのテキスト。

例外

表 53-38 COMMENT_ON_PRIORITY_GROUP および COMMENT_ON_SITE_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。

COMMENT_ON_REPGROUP プロシージャ

このプロシージャは、マスター・グループの DBA_REPGROUP ビューのコメント・フィールドを更新します。このプロシージャは、マスター定義サイトで発行する必要があります。

構文

```
DBMS_REPCAT.COMMENT_ON_REPGROUP (
   gname      IN    VARCHAR2,
   comment    IN    VARCHAR2);
```

パラメータ

表 53-39 COMMENT_ON_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	コメントを記述するレプリケーション・グループの名前。
comment	DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドに含める更新したコメント。

例外

表 53-40 COMMENT_ON_REPGROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

COMMENT_ON_REPOBJECT プロシージャ

このプロシージャは、マスター・グループにあるレプリケート・オブジェクトの DBA_REPOBJECT ビューのコメント・フィールドを更新します。このプロシージャは、マスター定義サイトで発行する必要があります。

構文

```
DBMS_REPCAT.COMMENT_ON_REPOBJECT (
   sname    IN   VARCHAR2,
   oname    IN   VARCHAR2,
   type     IN   VARCHAR2,
   comment  IN   VARCHAR2);
```

パラメータ

表 53-41 COMMENT_ON_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。
oname	コメントを記述するオブジェクトの名前。ネストした表の記憶表はこのオブジェクトに該当しません。
type	オブジェクトのタイプ。次のタイプがサポートされています。 FUNCTION　SYNONYM INDEX　TABLE INDEXTYPE　TRIGGER OPERATOR　TYPE PACKAGE　TYPE BODY PACKAGE BODY　VIEW PROCEDURE
comment	DBA_REPOBJECT ビューの OBJECT_COMMENT フィールドに含める更新したコメントのテキスト。

例外

表 53-42 COMMENT_ON_REPOBJECT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが存在しません。
typefailure	指定したタイプ・パラメータはサポートされていません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

COMMENT_ON_REPSITES プロシージャ

レプリケーション・グループがマスター・グループの場合、このプロシージャはマスター・サイトの DBA_REPSITES ビューの MASTER_COMMENT フィールドを更新します。レプリケーション・グループがマテリアライズド・ビュー・グループの場合、このプロシージャはマテリアライズド・ビュー・サイトの DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドを更新します。

このプロシージャは、マスター・サイトまたはマテリアライズド・ビュー・サイトで実行できます。マテリアライズド・ビュー・サイトでこのプロシージャを実行する場合、マテリアライズド・ビュー・グループの所有者は PUBLIC である必要があります。

> **関連項目：** マテリアライズド・ビュー・グループの所有者が PUBLIC でない場合に、マテリアライズド・ビュー・サイトの DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドにコメントを記入する方法は、53-46 ページの「COMMENT_ON_conflicttype_RESOLUTION プロシージャ」を参照してください。

構文

```
DBMS_REPCAT.COMMENT_ON_REPSITES (
   gname       IN   VARCHAR2,
   [ master    IN   VARCHAR,]
   comment     IN   VARCHAR2);
```

パラメータ

表 53-43 COMMENT_ON_REPSITES プロシージャのパラメータ

パラメータ	説明
gname	レプリケーション・グループの名前。データベースが複数のレプリケーション環境のマスター・サイトである場合は、このパラメータによって混乱を避けることができます。
master	コメントを記述するマスター・サイトの完全修飾データベース名。マスター・サイトでこのプロシージャを実行するときは、このパラメータが必要です。マテリアライズド・ビュー・サイトのコメントを更新するときは、このパラメータを省略します。このパラメータはオプションです。
comment	適切なディクショナリ・ビューのコメント・フィールドに含める更新したコメントのテキスト。サイトがマスター・サイトの場合、このプロシージャは DBA_REPSITES ビューの MASTER_COMMENT フィールドを更新します。サイトがマテリアライズド・ビュー・サイトの場合、このプロシージャは DBA_REPGROUP ビューの SCHEMA_COMMENT フィールドを更新します。

例外

表 53-44 COMMENT_ON_REPSITES プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
nonmaster	起動サイトがマスター・サイトではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。
missingrepgroup	レプリケーション・グループが存在しません。
commfailure	1 つ以上のマスター・サイトがアクセス不可です。
corrupt	レプリケーション・カタログ・ビューに一貫性がありません。

COMMENT_ON_*conflicttype*_RESOLUTION プロシージャ

このプロシージャは、競合解消ルーチンの DBA_REPRESOLUTION ビューの RESOLUTION_COMMENT フィールドを更新します。コールする必要があるプロシージャは、ルーチンが解消する競合のタイプによって決まります。このプロシージャは、マスター定義サイトで発行する必要があります。

表 53-45 COMMENT_ON_conflicttype_RESOLUTION プロシージャ

競合のタイプ	プロシージャ名
update	COMMENT_ON_UPDATE_RESOLUTION
uniqueness	COMMENT_ON_UNIQUE_RESOLUTION
delete	COMMENT_ON_DELETE_RESOLUTION

コメントは、次回の GENERATE_REPLICATION_SUPPORT コール後に、すべてのマスター・サイトに追加されます。

構文

```
DBMS_REPCAT.COMMENT_ON_UPDATE_RESOLUTION (
   sname                 IN   VARCHAR2,
   oname                 IN   VARCHAR2,
   column_group          IN   VARCHAR2,
   sequence_no           IN   NUMBER,
   comment               IN   VARCHAR2);

DBMS_REPCAT.COMMENT_ON_UNIQUE_RESOLUTION (
   sname                 IN   VARCHAR2,
   oname                 IN   VARCHAR2,
   constraint_name       IN   VARCHAR2,
   sequence_no           IN   NUMBER,
   comment               IN   VARCHAR2);

DBMS_REPCAT.COMMENT_ON_DELETE_RESOLUTION (
   sname                 IN   VARCHAR2,
   oname                 IN   VARCHAR2,
   sequence_no           IN   NUMBER,
   comment               IN   VARCHAR2);
```

パラメータ

表 53-46 COMMENT_ON_*conflicttype*_RESOLUTION プロシージャのパラメータ

パラメータ	説明
sname	スキーマの名前。
oname	競合解消ルーチンが関連付けられているレプリケート表の名前。
column_group	更新競合解消ルーチンが関連付けられている列グループの名前。
constraint_name	一意性競合解消ルーチンが関連付けられている一意制約の名前。
sequence_no	競合解消プロシージャの順序番号。
comment	DBA_REPRESOLUTION ビューの RESOLUTION_COMMENT フィールドに含める更新したコメントのテキスト。

例外

表 53-47 COMMENT_ON_*conflicttype*_RESOLUTION プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが存在しません。
missingresolution	指定した競合解消ルーチンが登録されていません。

COMPARE_OLD_VALUES プロシージャ

このプロシージャは、更新および削除のために、レプリケート表の各非キー列に対して、遅延トランザクションの伝播時に、各マスター・サイトの元の値を比較するかどうかを指定します。デフォルトでは、すべての列の元の値が比較されます。DBMS_REPCAT.COMPARE_OLD_VALUES をマスター定義サイトで起動すると、すべてのマスター・サイトおよびマテリアライズド・ビュー・サイトでこの動作を変更できます。

ユーザー定義型を使用する場合は、列オブジェクトのリーフ属性を指定することも、列オブジェクト全体を指定することもできます。たとえば、cust_address という名前の列オブジェクトが属性として street_address を持つ場合、column_list パラメータまたは column_table パラメータの一部に cust_address.street_address を指定することも、cust_address のみを指定することもできます。

等価比較を実行して競合を検出する場合、次の条件が真であるときに限りオブジェクトが等しいものとして扱われます。

- 両方のオブジェクトがアトミック NULL である（オブジェクト全体が NULL である）。
- オブジェクト内の対応する属性がすべて等しい。

これらの条件から、一方のオブジェクトがアトミック NULL であり、もう一方のオブジェクトがそうでない場合、両オブジェクトは等しいとはみなされません。等価比較の実行時、MAP メソッドおよび ORDER メソッドは考慮されません。

構文

```
DBMS_REPCAT.COMPARE_OLD_VALUES(
   sname           IN  VARCHAR2,
   oname           IN  VARCHAR2,
   { column_list   IN  VARCHAR2,
   | column_table  IN  DBMS_UTILITY.VARCHAR2s | DBMS_UTILITY.LNAME_ARRAY,}
   operation       IN  VARCHAR2 := 'UPDATE',
   compare         IN  BOOLEAN := true );
```

注意： このプロシージャはオーバーロードされています。column_list パラメータと column_table パラメータは、両方同時には指定できません。

パラメータ

表 53-48 COMPARE_OLD_VALUES プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマ。
oname	レプリケート表の名前。ネストした表の記憶表がこの表に該当します。
column_list	表内の列のカンマで区切られたリスト。エントリ間に空白を挿入しないでください。
column_table	リストのかわりに、列名を含む DBMS_REPCAT.VARCHAR2S タイプまたは DBMS_UTILITY.LNAME_ARRAY の PL/SQL 索引付き表を使用できます。最初の列名は位置 1、2 番目は位置 2、以下同様に設定されている必要があります。 列オブジェクトの属性を指定すると、30 バイト以上の列名が発生することがありますが、その場合は DBMS_UTILITY.LNAME_ARRAY を使用します。
operation	有効な値は、update、delete またはアスタリスクのワイルド・カード '*'（更新および削除を意味します）です。

表 53-48 COMPARE_OLD_VALUES プロシージャのパラメータ（続き）

パラメータ	説明
compare	TRUE の場合は、指定した列の元の値が送信時に比較されます。FALSE の場合、指定した列の元の値は送信時に比較されません。指定外の列および指定外の操作には影響しません。マスター定義サイトでは、表の min_communication が TRUE になると、指定した変更がすぐに有効となります。変更内容がマスター・サイトまたはマテリアライズド・ビュー・サイトで有効になるのは、min_communication に TRUE を設定して、次回そのサイトでレプリケーション・サポートを生成したときです。

注意： operation パラメータを使用すると、行の削除時または更新時に、非キー列の元の値を比較するかどうかを決定できます。元の値を比較しない場合は、更新または削除の適用時に、送信先の現行の列値と元の値が等しいとみなされます。

Oracle のデフォルト動作を変更する前に、COMPARE_OLD_VALUES プロシージャを使用したデータ伝播の最小化の詳細について、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

例外

表 53-49 COMPARE_OLD_VALUES プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーション情報を待機している表として指定のスキーマ内に存在していません。
missingcolumn	表に存在しない列が少なくとも 1 つあります。
notquiesced	マスター・グループが停止中ではありません。
typefailure	無効な操作が指定されています。
keysendcomp	指定した列は表のキー列です。
dbnotcompatible	機能がデータベースのバージョンと互換性がありません。通常は、列オブジェクトの属性を比較しようとしたときにこの例外が発生します。この場合、データベースはいずれも 9.0.1 以上の互換性レベルを持つものにする必要があります。

CREATE_MASTER_REPGROUP プロシージャ

このプロシージャは、新規で、空で、かつ停止中のマスター・グループを作成します。

構文

```
DBMS_REPCAT.CREATE_MASTER_REPGROUP (
   gname            IN   VARCHAR2,
   group_comment    IN   VARCHAR2     := '',
   master_comment   IN   VARCHAR2     := ''),
   qualifier        IN   VARCHAR2     := '');
```

パラメータ

表 53-50 CREATE_MASTER_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	作成するマスター・グループの名前。
group_comment	DBA_REPGROUP ビューに追加されるコメント。
master_comment	DBA_REPSITES ビューに追加されるコメント。
qualifier	マスター・グループの接続修飾子。必ず @ 符号を使用してください。接続修飾子の詳細は、『Oracle9*i* アドバンスト・レプリケーション』および『Oracle9*i* データベース管理者ガイド』を参照してください。

例外

表 53-51 CREATE_MASTER_REPGROUP プロシージャの例外

例外	説明
duplicaterepgroup	マスター・グループはすでに存在しています。
norepopt	アドバンスト・レプリケーション・オプションがインストールされていません。
missingrepgroup	マスター・グループ名が指定されていません。
qualifiertoolong	接続修飾子が長すぎます。

CREATE_MASTER_REPOBJECT プロシージャ

このプロシージャでは、オブジェクトをマスター・グループに追加して、レプリケート・オブジェクトにします。ユーザー定義型およびオブジェクト表の識別子はすべてのレプリケーション・サイトで保存されます。

クラスタ化表のレプリケーションはサポートされていますが、use_existing_object パラメータをクラスタ化表に FALSE で設定することはできません。つまり、マスター・グループに関与するすべてのマスター・サイトにクラスタ化表を作成してから CREATE_MASTER_REPOBJECT プロシージャを実行する必要があります。ただし、これらの表に表データは必要ありません。このようにして、クラスタ化表に対して copy_rows パラメータを TRUE に設定できます。

構文

```
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
   sname                IN   VARCHAR2,
   oname                IN   VARCHAR2,
   type                 IN   VARCHAR2,
   use_existing_object  IN   BOOLEAN     := true,
   ddl_text             IN   VARCHAR2    := NULL,
   comment              IN   VARCHAR2    := '',
   retry                IN   BOOLEAN     := false
   copy_rows            IN   BOOLEAN     := true,
   gname                IN   VARCHAR2    := '');
```

パラメータ

このプロシージャのパラメータは、次のとおりです。

表 53-52 CREATE_MASTER_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	レプリケートするオブジェクトが置かれているスキーマの名前。
oname	レプリケートするオブジェクトの名前。ddl_text が NULL の場合は、指定したスキーマ内にこのオブジェクトがすでに存在している必要があります。一意性を確保するために、表名は 27 バイト以内、パッケージ名は 24 バイト以内で指定してください。ネストした表の記憶表はこのオブジェクトに該当しません。
type	レプリケートするオブジェクトのタイプ。次のタイプがサポートされています。 FUNCTION　　SYNONYM INDEX　　TABLE INDEXTYPE　　TRIGGER OPERATOR　　TYPE PACKAGE　　TYPE BODY PACKAGE BODY　　VIEW PROCEDURE

表 53-52 CREATE_MASTER_REPOBJECT プロシージャのパラメータ（続き）

パラメータ	説明
use_existing_object	現行のマスター・サイトの同じタイプ、同じ形式のオブジェクトを再利用する場合は、TRUE を指定します。詳細は、表 53-54 を参照してください。 **注意：**クラスタ化表に対してこのパラメータを TRUE に設定する必要があります。
ddl_text	オブジェクトがマスター定義サイトにまだ存在していない場合は、このオブジェクトの作成に必要な DDL テキストを指定します。PL/SQL パッケージ、パッケージ本体、プロシージャおよびファンクションの後には、セミコロンを付ける必要があります。SQL 文には、セミコロンを付ける必要はありません。この DDL は、適用前には解析されません。したがって、作成するオブジェクトに対応する適切なスキーマ名とオブジェクト名を DDL テキストで使用していることを確認してください。 スキーマ（sname パラメータ）を指定しないで DDL を発行すると、レプリケーション管理者のスキーマがデフォルトのスキーマとして使用されます。レプリケーション管理者のスキーマを使用しない場合は、スキーマを必ず指定してください。 **注意：**ddl_text パラメータを使用してユーザー定義型またはオブジェクト表を追加するのではなく、まずオブジェクトを作成してからそのオブジェクトを追加してください。
comment	DBA_REPOBJECT ビューの OBJECT_COMMENT フィールドに追加されるコメント。
retry	以前にオブジェクトを作成できなかった場合に、そのオブジェクトの作成を再試行する場合は、TRUE を指定します。このパラメータは、エラーが一時的な場合やすでに修正されている場合、あるいはリソースが不十分などのエラーの場合に使用します。TRUE の場合、オブジェクトは、オブジェクトの状態が VALID 以外のマスター・サイトでのみ作成されます。
copy_rows	新規レプリケート・オブジェクトの初期の内容を、マスター定義サイトのオブジェクトの内容と一致させる場合は TRUE を指定します。詳細は、表 53-54 を参照してください。
gname	レプリケート・オブジェクトを作成するレプリケーション・グループの名前。指定しない場合はスキーマ名がデフォルトのレプリケーション・グループ名に使用されます。この場合、このプロシージャを正常に完了するには、スキーマと同じ名前のレプリケーション・グループが存在している必要があります。

表 53-53 CREATE_MASTER_REPOBJECT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	マスター・グループが停止中ではありません。
duplicateobject	指定したオブジェクトがマスター・グループにすでに存在しており、retry が FALSE であるか、または名前の競合が発生しています。
missingobject	sname および oname に該当するオブジェクトが存在せず、適切な DDL も指定されていません。
typefailure	指定したタイプのオブジェクトはレプリケートできません。
ddlfailure	マスター定義サイトで DDL が成功しませんでした。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。
notcompat	最低でも 7.3 互換モードではないリモート・マスターがあります。

オブジェクト作成

表 53-54 マスター・サイトにおけるオブジェクト作成

オブジェクトはすでに存在？	COPY_ROWS	USE_EXISTING_OBJECTS	結果
はい	TRUE	TRUE	オブジェクトが一致しない場合は duplicatedobject メッセージが戻されます。表の場合は、マスター定義サイトのデータが使用されます。
はい	FALSE	TRUE	オブジェクトが一致しない場合は duplicatedobject メッセージが戻されます。表の場合、DBA は内容が同じであることを確認する必要があります。
はい	TRUE/FALSE	FALSE	duplicatedobject メッセージが戻されます。
いいえ	TRUE	TRUE/FALSE	オブジェクトが作成されます。表には、マスター定義サイトのデータが移入されます。
いいえ	FALSE	TRUE/FALSE	オブジェクトが作成されます。DBA は表にデータを移入し、すべてのサイトで表の内容が一致していることを確認する必要があります。

CREATE_MVIEW_REPGROUP プロシージャ

このプロシージャは、空のマテリアライズド・ビュー・グループをローカル・データベース内に新規に作成します。CREATE_MVIEW_REPGROUP は、REGISTER_MIEW_REPGROUP を自動的にコールしますが、登録中に発生したエラーは無視します。

構文

```
DBMS_REPCAT.CREATE_MVIEW_REPGROUP (
   gname              IN   VARCHAR2,
   master             IN   VARCHAR2,
   comment            IN   VARCHAR2     := '',
   propagation_mode   IN   VARCHAR2     := 'ASYNCHRONOUS',
   fname              IN   VARCHAR2     := NULL
   gowner             IN   VARCHAR2     := 'PUBLIC');
```

パラメータ

表 53-55 CREATE_MVIEW_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	レプリケーション・グループの名前。そのレプリケーション・グループは、指定したマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに存在している必要があります。
master	レプリケーション環境でマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトとして使用するデータベースの完全修飾データベース名。必要に応じて接続修飾子を指定できます。接続修飾子の使用方法は、『Oracle9*i* アドバンスト・レプリケーション』および『Oracle9*i* データベース管理者ガイド』を参照してください。
comment	DBA_REPGROUP ビューに追加されるコメント。
propagation_mode	レプリケーション・グループ内のすべての更新可能なマテリアライズド・ビューに使用する伝播方法。受入れ可能値は synchronous および asynchronous です。
fname	内部使用のためのパラメータ。 **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。
gowner	マテリアライズド・ビュー・グループの所有者。

例外

表 53-56 CREATE_MVIEW_REPGROUP プロシージャの例外

例外	説明
duplicaterepgroup	レプリケーション・グループが、起動サイトにすでに存在しています。
nonmaster	指定したデータベースはマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトではありません。
commfailure	指定したデータベースにアクセスできません。
norepopt	アドバンスト・レプリケーション・オプションがインストールされていません。
typefailure	伝播モードの指定に誤りがあります。
missingrepgroup	レプリケート・グループがマスター・サイトに存在していません。
invalidqualifier	マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトの接続識別子は、レプリケーション・グループに対して有効ではありません。
alreadymastered	ローカル・サイトに、グループ名が同じで、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトが異なる別のマテリアライズド・ビュー・グループがあります。

CREATE_MVIEW_REPOBJECT プロシージャ

このプロシージャは、マテリアライズド・ビュー・グループにレプリケート・オブジェクトを追加します。

構文

```
DBMS_REPCAT.CREATE_MVIEW_REPOBJECT (
   sname                   IN   VARCHAR2,
   oname                   IN   VARCHAR2,
   type                    IN   VARCHAR2,
   ddl_text                IN   VARCHAR2  := '',
   comment                 IN   VARCHAR2  := '',
   gname                   IN   VARCHAR2  := '',
   gen_objs_owner          IN   VARCHAR2  := '',
   min_communication       IN   BOOLEAN   := true,
   generate_80_compatible  IN   BOOLEAN   := true,
  gowner                  IN   VARCHAR2  := 'PUBLIC');
```

パラメータ

表 53-57 CREATE_MVIEW_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。このスキーマは、このマテリアライズド・ビューがベースとしているマスター表またはマスター・マテリアライズド・ビューを所有するスキーマと同じである必要があります。
oname	レプリケート・マテリアライズド・ビュー・グループに追加するオブジェクトの名前。
type	レプリケートするオブジェクトのタイプ。次のタイプがサポートされています。 FUNCTION　　SNAPSHOT INDEX　　SYNONYM INDEXTYPE　　TRIGGER OPERATOR　　TYPE PACKAGE　　TYPE BODY PACKAGE BODY　　VIEW PROCEDURE

表 53-57 CREATE_MVIEW_REPOBJECT プロシージャのパラメータ（続き）

パラメータ	説明
ddl_text	SNAPSHOT タイプのオブジェクトの場合は、DDL がオブジェクトの作成に必要です。その他のタイプの場合は、次に示すデフォルトを使用します。 ''（空の文字列） 同じ名前のマテリアライズド・ビューがすでに存在している場合、DDL は無視され、既存のマテリアライズド・ビューがレプリケート・オブジェクトとして登録されます。マテリアライズド・ビューのマスター表またはマスター・マテリアライズド・ビューがこのスキーマに指定したマスターのレプリケーション・グループに存在しない場合、missingobject エラーが発生します。 スキーマを指定せずに DDL を発行すると、レプリケーション管理者のスキーマがデフォルトのスキーマとして使用されます。レプリケーション管理者のスキーマを使用しない場合は、スキーマを必ず指定してください。 オブジェクトが SNAPSHOT タイプでない場合、マテリアライズド・ビュー・サイトはマスター・サイトまたはマテリアライズド・ビュー・サイトに接続し、DDL テキストをプルダウンしてオブジェクトを作成します。オブジェクト・タイプが TYPE または TYPE BODY の場合、マテリアライズド・ビュー・サイトのオブジェクトのオブジェクト識別子（OID）は、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトの OID と同じになります。
comment	DBA_REPOBJECT ビューの OBJECT_COMMENT フィールドに追加されるコメント。
gname	オブジェクトを追加するレプリケート・マテリアライズド・ビュー・グループの名前。指定しない場合はスキーマ名がデフォルトのレプリケーション・グループ名に使用されます。このプロシージャが正常に実行されるようにするには、スキーマと同じ名前のマテリアライズド・ビュー・グループが存在している必要があります。
gen_objs_owner	トランザクションの所有者として割り当てるユーザーの名前。
min_communication	マテリアライズド・ビューのマスター・サイトが Oracle7 リリース 7.3 を実行している場合は、FALSE を設定します。新旧の値の伝播を最小化する場合は、TRUE を設定します。デフォルトは TRUE です。競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

表 53-57 CREATE_MVIEW_REPOBJECT プロシージャのパラメータ（続き）

パラメータ	説明
generate_80_compatible	マテリアライズド・ビューのマスター・サイトが Oracle8*i* リリース 8.1.5 より前のバージョンの Oracle サーバーを実行している場合は、TRUE を設定します。マテリアライズド・ビューのマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトが Oracle8*i* リリース 8.1.5 以上を実行している場合は、FALSE を設定します。
gowner	マテリアライズド・ビュー・グループの所有者。

例外

表 53-58 CREATE_MVIEW_REPOBJECT プロシージャの例外

例外	説明
nonmview	起動サイトがマテリアライズド・ビュー・サイトではありません。
nonmaster	マスターはマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトではなくなりました。
missingobject	指定したオブジェクトがマスターのレプリケーション・グループに存在していません。
duplicateobject	指定したオブジェクトは、すでに別の形式で存在しています。
typefailure	このタイプは許可されていません。
ddlfailure	DDL は成功しませんでした。
commfailure	マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトにはアクセスできません。
missingschema	スキーマが、データベース・スキーマとして存在していません。
badmviewddl	DDL は実行されましたが、マテリアライズド・ビューが存在しません。
onlyonemview	マスター表またはマスター・マテリアライズド・ビューに作成できるマテリアライズド・ビューは 1 つです。
badmviewname	マテリアライズド・ビュー・ベース表がマスター表またはマスター・マテリアライズド・ビューと異なります。
missingrepgroup	マスターにレプリケーション・グループが存在しません。

DEFINE_COLUMN_GROUP プロシージャ

このプロシージャは、空の列グループを作成します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DEFINE_COLUMN_GROUP (
   sname            IN   VARCHAR2,
   oname            IN   VARCHAR2,
   column_group     IN   VARCHAR2,
   comment          IN   VARCHAR2 := NULL);
```

パラメータ

表 53-59 DEFINE_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	列グループを作成するレプリケート表の名前。
column_group	作成する列グループの名前。
comment	DBA_REPCOLUMN_GROUP ビューに表示されるユーザー・テキスト。

例外

表 53-60 DEFINE_COLUMN_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定した表が存在しません。
duplicategroup	指定した列グループが、表にすでに存在しています。
notquiesced	指定した表が属しているレプリケーション・グループが停止中ではありません。

DEFINE_PRIORITY_GROUP プロシージャ

このプロシージャは、マスター・グループに新規の優先グループを作成します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DEFINE_PRIORITY_GROUP (
   gname          IN   VARCHAR2,
   pgroup         IN   VARCHAR2,
   datatype       IN   VARCHAR2,
   fixed_length   IN   INTEGER := NULL,
   comment        IN   VARCHAR2 := NULL);
```

パラメータ

表 53-61 DEFINE_PRIORITY_GROUP プロシージャのパラメータ

パラメータ	説明
gname	優先グループを作成するマスター・グループ。
pgroup	作成する優先グループの名前。
datatype	優先グループ・メンバーのデータ・タイプ。サポートされているデータ・タイプは、CHAR、VARCHAR2、NUMBER、DATE、RAW、NCHAR および NVARCHAR2 です。
fixed_length	CHAR データ・タイプの場合は、列の長さを指定してください。その他のすべてのタイプでは、デフォルトの NULL を使用できます。
comment	DBA_REPPRIORITY ビューに追加されるユーザー・コメント。

例外

表 53-62 DEFINE_PRIORITY_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
duplicatepriority group	指定した優先グループは、マスター・グループ内にすでに存在しています。
typefailure	指定したデータ・タイプはサポートされていません。
notquiesced	マスター・グループが停止中ではありません。

DEFINE_SITE_PRIORITY プロシージャ

このプロシージャは、マスター・グループに新規のサイト優先グループを作成します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DEFINE_SITE_PRIORITY (
   gname        IN   VARCHAR2,
   name         IN   VARCHAR2,
   comment      IN   VARCHAR2 := NULL);
```

パラメータ

表 53-63 DEFINE_SITE_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループを作成するマスター・グループ。
name	作成するサイト優先グループの名前。
comment	DBA_REPPRIORITY ビューに追加されるユーザー・コメント。

例外

表 53-64 DEFINE_SITE_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
duplicate prioritygroup	指定したサイト優先グループは、マスター・グループ内にすでに存在しています。
notquiesced	マスター・グループが停止中ではありません。

DO_DEFERRED_REPCAT_ADMIN プロシージャ

このプロシージャは、現行のマスター・サイトで指定したマスター・グループまたはすべてのマスター・サイト（ジョブ・キューを利用）に対して未処理のローカル遅延管理プロシージャを実行します。

DO_DEFERRED_REPCAT_ADMIN は、DO_DEFERRED_REPCAT_ADMIN をコールした接続ユーザーが送信した管理要求のみ実行します。その他のユーザーが送信した要求は無視されます。

構文

```
DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN (
   gname          IN   VARCHAR2,
   all_sites      IN   BOOLEAN := false);
```

パラメータ

表 53-65 DO_DEFERRED_REPCAT_ADMIN プロシージャのパラメータ

パラメータ	説明
gname	マスター・グループの名前。
all_sites	TRUE の場合は、ジョブを使用して各マスター・サイトごとにローカル管理プロシージャを実行します。

例外

表 53-66 DO_DEFERRED_REPCAT_ADMIN プロシージャの例外

例外	説明
nonmaster	起動サイトがマスター・サイトではありません。
commfailure	all_sites が TRUE ですが、アクセスできないマスター・サイトが少なくとも 1 つあります。

DROP_COLUMN_GROUP プロシージャ

このプロシージャは列グループを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DROP_COLUMN_GROUP (
   sname        IN   VARCHAR2,
   oname        IN   VARCHAR2,
   column_group IN   VARCHAR2);
```

パラメータ

表 53-67 DROP_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	列グループを削除するレプリケート表の名前。
column_group	削除する列グループの名前。

例外

表 53-68 DROP_COLUMN_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
referenced	指定した列グループは、競合の検出および解消で使用されています。
missingobject	指定した表が存在しません。
missinggroup	指定した列グループが存在しません。
notquiesced	表が属しているマスター・グループが停止中ではありません。

DROP_GROUPED_COLUMN プロシージャ

このプロシージャは、列グループからメンバーを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DROP_GROUPED_COLUMN (
   sname                 IN   VARCHAR2,
   oname                 IN   VARCHAR2,
   column_group          IN   VARCHAR2,
   list_of_column_names  IN   VARCHAR2 | DBMS_REPCAT.VARCHAR2s);
```

パラメータ

表 53-69 DROP_GROUPED_COLUMN プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	列グループが置かれているレプリケート表の名前。ネストした表の記憶表がこの表に該当します。
column_group	メンバーを削除する列グループの名前。
list_of_column_names	指定した列グループから削除する列の名前。列名は、カンマで区切られたリストまたは PL/SQL 索引付き表のいずれかで示します。PL/SQL 索引付き表は、DBMS_REPCAT.VARCHAR2S タイプにしてください。 列オブジェクトは指定できますが、列オブジェクトの属性は指定できません。 表がオブジェクトの場合、オブジェクト識別子列を列グループに追加するには SYS_NC_OID$ を指定します。オブジェクト識別子列では、各行オブジェクトのオブジェクト識別子が追跡されます。 表がネストした表の記憶表の場合、ネストした表の各行の識別子を追跡する列を追加するには NESTED_TABLE_ID を指定します。

例外

表 53-70 DROP_GROUPED_COLUMN プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定した表が存在しません。
notquiesced	表が属しているマスター・グループが停止中ではありません。

DROP_MASTER_REPGROUP プロシージャ

このプロシージャは、現行のサイトからマスター・グループを削除します。マスター定義サイトも含め、すべてのマスター・サイトからマスター・グループを削除するには、all_sites を TRUE に設定して、このプロシージャをマスター定義サイトでコールします。

構文

```
DBMS_REPCAT.DROP_MASTER_REPGROUP (
   gname             IN VARCHAR2,
   drop_contents     IN BOOLEAN    := false,
   all_sites         IN BOOLEAN    := false);
```

パラメータ

表 53-71 DROP_MASTER_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	現行のマスター・サイトから削除するマスター・グループの名前。
drop_contents	デフォルトでは、マスター・サイトのレプリケーション・グループを削除したとき、すべてのオブジェクトがデータベースに残ります。このオブジェクトは、この後レプリケートされなくなります。つまり、レプリケーション・グループ内のレプリケート・オブジェクトと他のマスター・サイトとの間で、変更内容の送受信が行われなくなります。このパラメータに TRUE を設定すると、マスター・グループ内のすべてのレプリケート・オブジェクトが、関連するスキーマから削除されます。
all_sites	このパラメータが TRUE で起動サイトがマスター定義サイトの場合、このプロシージャは、要求をすべてのマスターに同期式でマルチキャストします。この場合、マスター定義サイトでは要求がすぐに実行され、その他のマスター・サイトでは遅れて実行される可能性があります。

例外

表 53-72 DROP_MASTER_REPGROUP プロシージャの例外

例外	説明
nonmaster	起動サイトがマスター・サイトではありません。
nonmasterdef	all_sites が TRUE ですが、起動サイトがマスター定義サイトではありません。
commfailure	all_sites が TRUE ですが、アクセスできないマスター・サイトが少なくとも 1 つあります。
fullqueue	遅延リモート・プロシージャ・コール（RPC）・キューにマスター・グループのエントリがあります。
masternotremoved	all_sites が TRUE ですが、マスターでマスター定義サイトが認識されていません。

DROP_MASTER_REPOBJECT プロシージャ

このプロシージャは、マスター・グループからレプリケート・オブジェクトを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.DROP_MASTER_REPOBJECT (
   sname          IN   VARCHAR2,
   oname          IN   VARCHAR2,
   type           IN   VARCHAR2,
   drop_objects   IN   BOOLEAN     := false);
```

パラメータ

表 53-73 DROP_MASTER_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。
oname	マスター・グループから削除するオブジェクトの名前。ネストした表の記憶表はこのオブジェクトに該当しません。
type	削除するオブジェクトのタイプ。次のタイプがサポートされています。 FUNCTION　SYNONYM INDEX　TABLE INDEXTYPE　TRIGGER OPERATOR　TYPE PACKAGE　TYPE BODY PACKAGE BODY　VIEW PROCEDURE
drop_objects	デフォルトでは、オブジェクトはスキーマ内に残りますが、マスター・グループからは削除されます。つまり、このオブジェクトに対する変更は、他のマスター・サイトおよびマテリアライズド・ビュー・サイトにレプリケートされなくなります。レプリケーション環境からオブジェクトを完全に削除するには、このパラメータに TRUE を設定します。パラメータを TRUE に設定すると、各マスター・サイトでデータベースからオブジェクトが削除されます。

例外

表 53-74 DROP_MASTER_REPOBJECT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが存在しません。
typefailure	指定したタイプ・パラメータはサポートされていません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

DROP_MVIEW_REPGROUP プロシージャ

このプロシージャは、レプリケーション環境からマテリアライズド・ビュー・サイトを削除します。DROP_MVIEW_REPGROUP は、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで自動的に UNREGISTER_MVIEW_REPGROUP をコールして、マテリアライズド・ビューの登録を解除しますが、登録解除時に発生したエラーは無視されます。DROP_MVIEW_REPGROUP が失敗した場合は、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに接続し、UNREGISTER_MVIEW_REPGROUP を実行します。

構文

```
DBMS_REPCAT.DROP_MVIEW_REPGROUP (
   gname                IN   VARCHAR2,
   drop_contents        IN   BOOLEAN   := false
   gowner               IN   VARCHAR2  := 'PUBLIC');
```

パラメータ

表 53-75 DROP_MVIEW_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	現行のマテリアライズド・ビュー・サイトから削除するレプリケーション・グループの名前。トリガーやパッケージなど、レプリケーションをサポートするために生成されたすべてのオブジェクトが削除されます。
drop_contents	デフォルトでは、マテリアライズド・ビュー・サイトのレプリケーション・グループを削除すると、すべてのオブジェクトは関連するスキーマに残ります。このオブジェクトは、この後レプリケートされなくなります。このパラメータに TRUE を設定すると、レプリケーション・グループ内のすべてのレプリケート・オブジェクトが、関連するスキーマから削除されます。
gowner	マテリアライズド・ビュー・グループの所有者。

例外

表 53-76 DROP_MVIEW_REPGROUP プロシージャの例外

例外	説明
nonmview	起動サイトがマテリアライズド・ビュー・サイトではありません。
missingrepgroup	指定したレプリケーション・グループが存在しません。

DROP_MVIEW_REPOBJECT プロシージャ

このプロシージャは、マテリアライズド・ビュー・サイトからレプリケート・オブジェクトを削除します。

構文

```
DBMS_REPCAT.DROP_MVIEW_REPOBJECT (
   sname          IN   VARCHAR2,
   oname          IN   VARCHAR2,
   type           IN   VARCHAR2,
   drop_objects   IN   BOOLEAN  := false);
```

パラメータ

表 53-77 DROP_MVIEW_REPOBJECT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマの名前。
oname	レプリケーション・グループから削除するオブジェクトの名前。
type	削除するオブジェクトのタイプ。次のタイプがサポートされています。 FUNCTION　SNAPSHOT INDEX　SYNONYM INDEXTYPE　TRIGGER OPERATOR　TYPE PACKAGE　TYPE BODY PACKAGE BODY　VIEW PROCEDURE
drop_objects	デフォルトでは、オブジェクトは関連スキーマ内に残りますが、その関連レプリケーション・グループからは削除されます。現行のマテリアライズド・ビュー・サイトのスキーマからオブジェクトを完全に削除するには、このパラメータに TRUE を設定します。パラメータを TRUE に設定すると、マテリアライズド・ビュー・サイトでデータベースからオブジェクトが削除されます。

例外

表 53-78 DROP_MVIEW_REPOBJECT プロシージャの例外

例外	説明
nonmview	起動サイトがマテリアライズド・ビュー・サイトではありません。
missingobject	指定したオブジェクトが存在しません。
typefailure	指定したタイプ・パラメータはサポートされていません。

DROP_PRIORITY プロシージャ

このプロシージャは、優先順位レベルに従って優先グループのメンバーを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DROP_PRIORITY(
      gname          IN   VARCHAR2,
      pgroup         IN   VARCHAR2,
      priority_num   IN   NUMBER);
```

パラメータ

表 53-79 DROP_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているマスター・グループ。
pgroup	削除するメンバーを含んだ優先グループの名前。
priority_num	グループから削除する優先グループ・メンバーの優先順位レベル。

例外

表 53-80 DROP_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。
notquiesced	マスター・グループが停止中ではありません。

DROP_PRIORITY_GROUP プロシージャ

このプロシージャは、指定したマスター・グループの優先グループを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DROP_PRIORITY_GROUP (
   gname     IN   VARCHAR2,
   pgroup    IN   VARCHAR2);
```

パラメータ

表 53-81 DROP_PRIORITY_GROUP プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているマスター・グループ。
pgroup	削除する優先グループの名前。

例外

表 53-82 DROP_PRIORITY_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
referenced	指定した優先グループは、競合の解消で使用されています。
notquiesced	指定したマスター・グループが停止中ではありません。

DROP_PRIORITY_*datatype* プロシージャ

このプロシージャは、値による優先グループのメンバーを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、priority 列のデータ・タイプによって決まります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DROP_PRIORITY_datatype (
   gname    IN   VARCHAR2,
   pgroup   IN   VARCHAR2,
   value    IN   datatype);
```

datatype には次のいずれかを指定します。

```
{ NUMBER
| VARCHAR2
| CHAR
| DATE
| RAW
| NCHAR
| NVARCHAR2 }
```

パラメータ

表 53-83 DROP_PRIORITY_*datatype* プロシージャのパラメータ

パラメータ	説明
gname	優先グループが関連付けられているマスター・グループ。
pgroup	削除するメンバーを含んだ優先グループの名前。
value	グループから削除する優先グループ・メンバーの値。

例外

表 53-84 DROP_PRIORITY_*datatype* プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
missingprioritygroup	指定した優先グループが存在しません。
paramtype、typefailure	優先グループに対する値のデータ・タイプが正しくありません。
notquiesced	指定したマスター・グループが停止中ではありません。

DROP_SITE_PRIORITY プロシージャ

このプロシージャは、指定したマスター・グループのサイト優先グループを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

関連項目： 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DROP_SITE_PRIORITY (
   gname      IN   VARCHAR2,
   name       IN   VARCHAR2);
```

パラメータ

表 53-85 DROP_SITE_PRIORITY プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループが関連付けられているマスター・グループ。
name	削除するサイト優先グループの名前。

例外

表 53-86 DROP_SITE_PRIORITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
referenced	指定したサイト優先グループは、競合の解消で使用されています。
notquiesced	指定したマスター・グループが停止中ではありません。

DROP_SITE_PRIORITY_SITE プロシージャ

このプロシージャは、サイト優先グループから、名前で指定したサイトを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.DROP_SITE_PRIORITY_SITE (
   gname      IN   VARCHAR2,
   name       IN   VARCHAR2,
   site       IN   VARCHAR2);
```

パラメータ

表 53-87 DROP_SITE_PRIORITY_SITE プロシージャのパラメータ

パラメータ	説明
gname	サイト優先グループが関連付けられているマスター・グループ。
name	メンバーを削除するサイト優先グループの名前。
site	グループから削除するサイトのグローバル・データベース名。

例外

表 53-88 DROP_SITE_PRIORITY_SITE プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingrepgroup	指定したマスター・グループが存在しません。
missingpriority	指定したサイト優先グループが存在しません。
notquiesced	指定したマスター・グループが停止中ではありません。

DROP_*conflicttype*_RESOLUTION プロシージャ

このプロシージャは、更新、削除または一意性競合の解消ルーチンを削除します。このプロシージャは、マスター定義サイトからコールする必要があります。コールする必要があるプロシージャは、ルーチンが解消する競合のタイプによって決まります。

競合解消ルーチン

各競合解消ルーチンのプロシージャ名を次の表に示します。

表 53-89 競合解消ルーチン

ルーチン	プロシージャ名
update	DROP_UPDATE_RESOLUTION
uniqueness	DROP_UNIQUE_RESOLUTION
delete	DROP_DELETE_RESOLUTION

構文

```
DBMS_REPCAT.DROP_UPDATE_RESOLUTION (
   sname              IN   VARCHAR2,
   oname              IN   VARCHAR2,
   column_group       IN   VARCHAR2,
   sequence_no        IN   NUMBER);

DBMS_REPCAT.DROP_DELETE_RESOLUTION (
   sname              IN   VARCHAR2,
   oname              IN   VARCHAR2,
   sequence_no        IN   NUMBER);

DBMS_REPCAT.DROP_UNIQUE_RESOLUTION (
   sname              IN   VARCHAR2,
   oname              IN   VARCHAR2,
   constraint_name    IN   VARCHAR2,
   sequence_no        IN   NUMBER);
```

パラメータ

表 53-90 DROP_*conflicttype*_RESOLUTION プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマ。
oname	競合解消ルーチンを削除する表の名前。
column_group	更新競合解消ルーチンを削除する列グループの名前。
constraint_name	一意性競合解消ルーチンを削除する一意性制約の名前。
sequence_no	削除する競合解消方法に割り当てられている順序番号。この番号でルーチンが一意に識別されます。

例外

表 53-91 DROP_*conflicttype*_RESOLUTION プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、指定のスキーマ内の表として存在していないか、または指定した順序番号の競合解消ルーチンが登録されていません。
notquiesced	マスター・グループが停止中ではありません。

EXECUTE_DDL プロシージャ

このプロシージャは、一部またはすべてのマスター・サイトで実行される DDL を提供します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.EXECUTE_DDL (
   gname           IN   VARCHAR2,
   { master_list   IN   VARCHAR2     := NULL,
   | master_table  IN   DBMS_UTILITY.DBLINK_ARRAY,}
   DDL_TEXT        IN   VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。master_list パラメータと master_table パラメータは、両方同時には指定できません。

パラメータ

表 53-92 EXECUTE_DDL プロシージャのパラメータ

パラメータ	説明
gname	マスター・グループの名前。
master_list	提供された DDL を実行するマスター・サイトの名前のカンマで区切られたリスト。各サイト名の間には空白を入れないでください。デフォルト値 NULL は、マスター定義サイトも含め、すべてのサイトで DDL が実行されることを示します。
master_table	提供された DDL を実行するマスター・サイトをリストした表。最初のマスターは位置 1、2 番目は位置 2、以下同様に設定されている必要があります。
ddl_text	指定した各マスター・サイトで実行する DDL。スキーマを指定せずに DDL を発行すると、レプリケーション管理者のスキーマがデフォルトのスキーマとして使用されます。レプリケーション管理者のスキーマを使用しない場合は、スキーマを必ず指定してください。

例外

表 53-93 EXECUTE_DDL プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
nonmaster	マスター・サイト以外のサイトが少なくとも 1 つあります。
ddlfailure	マスター定義サイトで DDL が成功しませんでした。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

GENERATE_MVIEW_SUPPORT プロシージャ

このプロシージャは、トリガーを起動し、更新可能なマテリアライズド・ビューのレプリケーションまたはプロシージャ・レプリケーションのサポートに必要なパッケージを生成します。このプロシージャは、マテリアライズド・ビュー・サイトからコールする必要があります。

> **注意：** CREATE_MVIEW_REPOBJECT は、更新可能なマテリアライズド・ビューに自動的にマテリアライズド・ビュー・サポートを生成します。

構文

```
DBMS_REPCAT.GENERATE_MVIEW_SUPPORT (
   sname                    IN VARCHAR2,
   oname                    IN VARCHAR2,
   type                     IN VARCHAR2,
   gen_objs_owner           IN VARCHAR2 := '',
   min_communication        IN BOOLEAN  := true,
   generate_80_compatible   IN BOOLEAN  := true);
```

パラメータ

表 53-94 GENERATE_MVIEW_SUPPORT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマ。
oname	サポートを生成するオブジェクトの名前。
type	オブジェクトのタイプ。サポートされているタイプは、SNAPSHOT、PACKAGE および PACKAGE BODY です。
gen_objs_owner	タイプが PACKAGE または PACKAGE BODY のオブジェクトの場合は、生成されたオブジェクトが作成されるスキーマを指定します。NULL の場合、オブジェクトは sname に作成されます。
min_communication	TRUE の場合は、更新文で列を変更する場合にかぎり、更新トリガーによって列の新しい値が送信されます。その列がキー列または変更された列グループ内の列の場合、更新トリガーはその列の元の値のみ送信します。
generate_80_compatible	マテリアライズド・ビューのマスター・サイトが Oracle8*i* リリース 8.1.5 より前のバージョンの Oracle サーバーを実行している場合は、TRUE を設定します。マテリアライズド・ビューのマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトが Oracle8*i* リリース 8.1.5 以上を実行している場合は、FALSE を設定します。

例外

表 53-95 GENERATE_MVIEW_SUPPORT プロシージャの例外

例外	説明
nonmview	起動サイトがマテリアライズド・ビュー・サイトではありません。
missingobject	指定したオブジェクトが、行または列レベルのレプリケーション情報を待機しているマテリアライズド・ビューとしてレプリケート・スキーマ内に存在していないか、またはラッパー生成を待機しているパッケージ（本体）として存在していません。
typefailure	指定したタイプ・パラメータはサポートされていません。
missingschema	生成オブジェクトの指定の所有者が存在しません。
missingremoteobject	マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのオブジェクトは、まだレプリケーション・サポートを生成していません。
commfailure	マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトにはアクセスできません。

GENERATE_REPLICATION_SUPPORT プロシージャ

このプロシージャは、指定したオブジェクトのレプリケーションのサポートに必要なトリガーおよびパッケージを生成します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  sname                     IN    VARCHAR2,
  oname                     IN    VARCHAR2,
  type                      IN    VARCHAR2,
  package_prefix            IN    VARCHAR2   := NULL,
  procedure_prefix          IN    VARCHAR2   := NULL,
  distributed               IN    BOOLEAN    := true,
  gen_objs_owner            IN    VARCHAR2   := NULL,
  min_communication         IN    BOOLEAN    := true,
  generate_80_compatible    IN    BOOLEAN    := true);
```

パラメータ

表 53-96 GENERATE_REPLICATION_SUPPORT プロシージャのパラメータ

パラメータ	説明
sname	オブジェクトが置かれているスキーマ。
oname	レプリケーション・サポートを生成するオブジェクトの名前。
type	オブジェクトのタイプ。サポートされているタイプは、TABLE、PACKAGE および PACKAGE BODY です。
package_prefix	タイプが PACKAGE または PACKAGE BODY のオブジェクトの場合は、生成されたラッパー・パッケージ名の前にこのパラメータの値が付加されます。デフォルトは DEFER_ です。
procedure_prefix	タイプが PACKAGE または PACKAGE BODY のオブジェクトの場合は、生成されたラッパー・プロシージャ名の前にこのパラメータの値が付加されます。デフォルトでは、接頭辞は割り当てられません。
distributed	この値は TRUE に設定してください。
gen_objs_owner	タイプが PACKAGE または PACKAGE BODY のオブジェクトの場合は、生成されたオブジェクトが作成されるスキーマを指定します。NULL の場合、オブジェクトは sname に作成されます。

表 53-96 GENERATE_REPLICATION_SUPPORT プロシージャのパラメータ（続き）

パラメータ	説明
min_communication	マスター・サイトのいずれかが Oracle7 リリース 7.3 を実行している場合は、FALSE を設定します。新旧の値の伝播を最小化する場合は、TRUE を設定します。デフォルトは TRUE です。詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。
generate_80_compatible	Oracle8*i* リリース 8.1.5 より前のバージョンの Oracle サーバーを実行しているマスター・サイトがある場合は、TRUE を設定します。すべてのマスター・サイトが Oracle8*i* リリース 8.1.5 以上を実行している場合は、FALSE を設定します。

例外

表 53-97 GENERATE_REPLICATION_SUPPORT プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーション情報を待機している表として指定のスキーマ内に存在していないか、またはラッパー生成を待機しているパッケージ（本体）として存在していません。
typefailure	指定したタイプ・パラメータはサポートされていません。
notquiesced	レプリケーション・グループが停止中ではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。
missingschema	スキーマが存在しません。
dbnotcompatible	マスター・サイトの 1 つがリリース 7.3.0.0 と互換性がありません。
notcompat	マスター・サイトの 1 つがリリース 7.3.0.0 と互換性がありません（dbnotcompatible と同じです）。
duplicateobject	オブジェクトがすでに存在しています。

MAKE_COLUMN_GROUP プロシージャ

このプロシージャは、1 つ以上のメンバーを含んだ新しい列グループを作成します。このプロシージャは、マスター定義サイトからコールする必要があります。

> **関連項目：** 競合の解消方法の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT.MAKE_COLUMN_GROUP (
   sname                 IN   VARCHAR2,
   oname                 IN   VARCHAR2,
   column_group          IN   VARCHAR2,
   list_of_column_names  IN   VARCHAR2 | DBMS_REPCAT.VARCHAR2s);
```

パラメータ

表 53-98 MAKE_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	新しい列グループを作成するレプリケート表の名前。ネストした表の記憶表がこの表に該当します。
column_group	作成する列グループに割り当てる名前。
list_of_column_names	グループ化する列の名前。列名は、カンマで区切られたリストまたは PL/SQL 索引付き表のいずれかで示します。PL/SQL 索引付き表は、DBMS_REPCAT.VARCHAR2S タイプにしてください。表内のすべての列を含んだ列グループを作成するには、単一の値 '*' を使用します。 列オブジェクトは指定できますが、列オブジェクトの属性は指定できません。 表がオブジェクト表の場合に、オブジェクト識別子列を列グループに追加するには SYS_NC_OID$ を指定します。オブジェクト識別子列では、各行オブジェクトのオブジェクト識別子が追跡されます。 表がネストした表の記憶表の場合に、ネストした表の各行の識別子を追跡する列を追加するには NESTED_TABLE_ID を指定します。

例外

表 53-99 MAKE_COLUMN_GROUP プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
duplicategroup	指定した列グループが、表にすでに存在しています。
missingobject	指定した表が存在しません。
missingcolumn	指定した列が、指定した表内に存在していません。
duplicatecolumn	指定した列は、すでに別の列グループのメンバーです。
notquiesced	マスター・グループが停止中ではありません。

PREPARE_INSTANTIATED_MASTER プロシージャ

このプロシージャは、準備済みのその他の新規マスター・サイトおよび既存のマスター・サイトの遅延トランザクションを、起動マスター・サイトに伝播することを可能にします。また、起動マスター・サイトの遅延トランザクションを、準備済みのその他の新規マスター・サイトおよび既存のマスター・サイトに伝播することも可能にします。

データベース全体のエクスポート / インポートまたは変更ベースのリカバリを実行した場合は、マスター定義サイトの遅延トランザクション・キューに存在したすべての遅延トランザクションが新規マスター・サイトに含まれます。これらの遅延トランザクションは新規マスター・サイトに存在すべきではないため、データベース全体のエクスポート / インポートまたは変更ベースのリカバリが使用されると、このプロシージャで遅延トランザクション・キューおよびエラー・キューのすべてのトランザクションが削除されます。

オブジェクトレベルのエクスポート / インポートの場合は、このプロシージャの実行前に、拡張グループの DBA_REPCATLOG データ・ディクショナリ・ビューの全要求がエラーなしで処理されたことを確認してください。

注意：

- このプロシージャは、新規マスター・サイトのインスタンス化（エクスポート / インポートまたは変更ベースのリカバリ）が完了するまで起動しないでください。
- データ操作言語（DML）文は、このプロシージャの実行が正常に戻らないかぎり、新規マスター・サイトの拡張マスター・グループのオブジェクトに直接使用しないでください。これらの DML 文はレプリケートできません。
- このプロシージャの実行が正常に戻らないかぎり、DBMS_DEFER パッケージを使用して遅延トランザクションを作成しないでください。これらの遅延トランザクションはレプリケートできません。

注意： 変更ベースのリカバリを使用するには、同じオペレーティング・システムで既存のマスター・サイトおよび新規マスター・サイトを実行する必要があります。ただし、オペレーティング・システムのリリースは異なっていてもかまいません。

構文

```
DBMS_REPCAT.PREPARE_INSTANTIATED_MASTER (
   extension_id             IN       RAW);
```

パラメータ

表 53-100 PREPARE_INSTANTIATED_MASTER プロシージャのパラメータ

パラメータ	説明
extension_id	マスター・データベースを停止せずにマスター・グループに追加するための現行の保留中の要求の識別子。extension_id 値を検索するには、DBA_REPSITES_NEW データ・ディクショナリ・ビューおよび DBA_REPEXTENSIONS データ・ディクショナリ・ビューに問い合せます。

例外

表 53-101 PREPARE_INSTANTIATED_MASTER プロシージャの例外

例外	説明
typefailure	パラメータに指定したパラメータ値が適切ではありません。
dbnotcompatible	機能がデータベースのバージョンと互換性がありません。データベースはいずれも 9.0.1 以上の互換性レベルを持つものにする必要があります。

PURGE_MASTER_LOG プロシージャ

このプロシージャは、指定した識別番号、ソースまたはマスター・グループに関連する DBA_REPCATLOG ビュー内のローカル・メッセージを削除します。

特定のソースからすべての管理要求をパージするには、id パラメータに NULL を指定します。全ソースからすべての管理要求をパージするには、id パラメータおよび source パラメータの両方に NULL を指定します。

構文

```
DBMS_REPCAT.PURGE_MASTER_LOG (
   id      IN   BINARY_INTEGER,
   source  IN   VARCHAR2,
   gname   IN   VARCHAR2);
```

パラメータ

表 53-102 PURGE_MASTER_LOG プロシージャのパラメータ

パラメータ	説明
id	要求の識別番号。DBA_REPCATLOG ビューに表示される番号です。
source	要求発信元のマスター・サイト。
gname	要求の作成対象となったマスター・グループの名前。

例外

表 53-103 PURGE_MASTER_LOG プロシージャの例外

例外	説明
nonmaster	gname が NULL でなく、起動サイトがマスター・サイトではありません。

PURGE_STATISTICS プロシージャ

このプロシージャは、DBA_REPRESOLUTION_STATISTICS ビューから情報を削除します。

構文

```
DBMS_REPCAT.PURGE_STATISTICS (
   sname      IN   VARCHAR2,
   oname      IN   VARCHAR2,
   start_date IN   DATE,
   end_date   IN   DATE);
```

パラメータ

表 53-104 PURGE_STATISTICS プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマの名前。
oname	競合解消統計を削除する表の名前。
start_date/ end_date	統計を削除する日付範囲。start_date が NULL の場合は、end_date までの統計がすべて削除されます。end_date が NULL の場合は、start_date 以降の統計がすべて削除されます。

例外

表 53-105 PURGE_STATISTICS プロシージャの例外

例外	説明
missingschema	指定したスキーマが存在しません。
missingobject	指定した表が存在しません。
statnotreg	この表は統計収集をするために登録されていません。

REFRESH_MVIEW_REPGROUP プロシージャ

このプロシージャは、マテリアライズド・ビュー・グループを、関連するマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトからの最新データでリフレッシュします。

構文

```
DBMS_REPCAT.REFRESH_MVIEW_REPGROUP (
   gname                  IN   VARCHAR2,
   drop_missing_contents  IN   BOOLEAN    := false,
   refresh_mviews         IN   BOOLEAN    := false,
   refresh_other_objects  IN   BOOLEAN    := false,
   gowner                 IN   VARCHAR2   := 'PUBLIC');
```

パラメータ

表 53-106 REFRESH_MVIEW_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	レプリケーション・グループの名前。
drop_missing_contents	マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのレプリケーション・グループからオブジェクトを削除しても、マテリアライズド・ビュー・サイトのスキーマからそのオブジェクトは自動的に削除されません。ただし、そのオブジェクトはレプリケートされなくなります。つまり、このオブジェクトに対する変更は、関連するマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトに送信されなくなります。マテリアライズド・ビューは関連付けられたマスター表またはマスター・マテリアライズド・ビューから引続きリフレッシュできます。ただし、更新可能なマテリアライズド・ビューに対する変更は失われます。レプリケーション・グループからオブジェクトを削除する場合は、このパラメータを TRUE に設定してスキーマから完全に削除することもできます。
refresh_mviews	TRUE に設定すると、レプリケーション・グループ内のマテリアライズド・ビューの内容がリフレッシュされます。
refresh_other_objects	これを TRUE に設定すると、レプリケーション・グループ内の非マテリアライズド・ビュー・オブジェクトの内容がリフレッシュされます。非マテリアライズド・ビュー・オブジェクトには、次の項目を含めることができます。 ■ 表 ■ ビュー ■ 索引 ■ PL/SQL パッケージおよびパッケージ本体 ■ PL/SQL プロシージャおよびファンクション ■ トリガー ■ シノニム
gowner	マテリアライズド・ビュー・グループの所有者。

例外

表 53-107 REFRESH_MVIEW_REPGROUP プロシージャの例外

例外	説明
nonmview	起動サイトがマテリアライズド・ビュー・サイトではありません。
nonmaster	マスターはマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトではなくなりました。
commfailure	マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトにはアクセスできません。
missingrepgroup	レプリケーション・グループ名が指定されていません。

REGISTER_MVIEW_REPGROUP プロシージャ

このプロシージャは、DBA_REGISTERED_MVIEW_GROUPS にマテリアライズド・ビュー・グループを挿入または変更して、各マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのマテリアライズド・ビューの管理を容易にします。

構文

```
DBMS_REPCAT.REGISTER_MVIEW_REPGROUP (
   gname            IN   VARCHAR2,
   mviewsite        IN   VARCHAR2,
   comment          IN   VARCHAR2  := NULL,
   rep_type         IN   NUMBER    := reg_unknown,
   fname            IN   VARCHAR2  := NULL
   gowner           IN   VARCHAR2  := 'PUBLIC');
```

パラメータ

表 53-108 REGISTER_MVIEW_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	登録するマテリアライズド・ビュー・グループの名前。
mviewsite	マテリアライズド・ビュー・サイトのグローバル名。
comment	マテリアライズド・ビュー・サイトに対するコメント、または既存のコメントに対する更新。
rep_type	マテリアライズド・ビュー・グループのバージョン。割当て可能な定数は次のとおりです。 ■ dbms_repcat.reg_unknown（デフォルト） ■ dbms_repcat.reg_v7_group ■ dbms_repcat.reg_v8_group
fname	内部使用のためのパラメータ。 **注意：**オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。
gowner	マテリアライズド・ビュー・グループの所有者。

例外

表 53-109 REGISTER_MVIEW_REPGROUP プロシージャの例外

例外	説明
failregmviewrepgroup	マテリアライズド・ビュー・グループの登録に失敗しました。
missingrepgroup	レプリケーション・グループ名が指定されていません。
nullsitename	マテリアライズド・ビュー・サイトが指定されていません。
nonmaster	プロシージャは、マテリアライズド・ビューのマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで実行する必要があります。
duplicaterepgroup	レプリケーション・グループはすでに存在しています。

REGISTER_STATISTICS プロシージャ

このプロシージャは、表の更新競合、削除競合および一意性競合の正常な解消に関する情報を収集します。

構文

```
DBMS_REPCAT.REGISTER_STATISTICS (
   sname IN   VARCHAR2,
   oname IN   VARCHAR2);
```

パラメータ

表 53-110 REGISTER_STATISTICS プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマの名前。
oname	競合解消統計を収集する表の名前。

例外

表 53-111 REGISTER_STATISTICS プロシージャの例外

例外	説明
missingschema	指定したスキーマが存在しません。
missingobject	指定した表が存在しません。

RELOCATE_MASTERDEF プロシージャ

このプロシージャは、マスター定義サイトをレプリケーション環境内の別のマスター・サイトに変更します。

RELOCATE_MASTERDEF のコール時に、元のマスター定義サイトおよび新しいマスター定義サイトのいずれも使用可能である必要はありません。計画的に再構成する場合は、notify_masters に TRUE および include_old_masterdef に TRUE を設定して RELOCATE_MASTERDEF を起動してください。

構文

```
DBMS_REPCAT.RELOCATE_MASTERDEF (
   gname                  IN   VARCHAR2,
   old_masterdef          IN   VARCHAR2,
   new_masterdef          IN   VARCHAR2,
   notify_masters         IN   BOOLEAN    := true,
   include_old_masterdef  IN   BOOLEAN    := true,
   require_flavor_change  IN   BOOLEAN    := false);
```

パラメータ

表 53-112 RELOCATE_MASTERDEF プロシージャのパラメータ

パラメータ	説明
gname	マスター定義を再配布するレプリケーション・グループの名前。
old_masterdef	現行のマスター定義サイトの完全修飾データベース名。
new_masterdef	新しいマスター定義サイトにする既存のマスター・サイトの完全修飾データベース名。
notify_masters	TRUE の場合、このプロシージャは、変更内容をすべてのマスターに同期式でマルチキャストします (include_old_masterdef が TRUE の場合のみ old_masterdef も含まれます)。変更を適用できないマスターがある場合は、すべてのマスターで変更がロールバックされます。 マスター定義サイトでのみ障害が発生した場合は、notify_masters に TRUE および include_old_masterdef に FALSE を設定して RELOCATE_MASTERDEF を起動してください。複数のマスター・サイトおよびマスター定義サイトで障害が発生した場合、管理者は、notify_masters に FALSE を設定し、各マスターで RELOCATE_MASTERDEF を起動する必要があります。
include_old_masterdef	notify_masters が TRUE で、include_old_masterdef も TRUE の場合は、元のマスター定義サイトにも変更内容が通知されます。
require_flavor_change	内部使用のためのパラメータ。 **注意 :** オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

例外

表 53-113 RELOCATE_MASTERDEF プロシージャの例外

例外	説明
nonmaster	new_masterdef がマスター・サイトではないか、または起動サイトがマスター・サイトではありません。
nonmasterdef	old_masterdef がマスター定義サイトではありません。
commfailure	notify_masters が TRUE ですが、アクセスできないマスター・サイトが少なくとも 1 つあります。

REMOVE_MASTER_DATABASES プロシージャ

このプロシージャは、レプリケーション環境から 1 つ以上のマスター・データベースを削除します。また、トリガーと関連するパッケージを、残りのマスター・サイトで再生成します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.REMOVE_MASTER_DATABASES (
   gname           IN   VARCHAR2,
   master_list     IN   VARCHAR2 |
   master_table    IN   DBMS_UTILITY.DBLINK_ARRAY);
```

注意： このプロシージャはオーバーロードされています。master_list パラメータと master_table パラメータは、両方同時には指定できません。

パラメータ

表 53-114 REMOVE_MASTER_DATABASES プロシージャのパラメータ

パラメータ	説明
gname	レプリケーション環境に関連付けられているレプリケーション・グループの名前。マスター・データベースが複数のレプリケーション環境で使用されている場合に、このパラメータによって混乱を避けることができます。
master_list	レプリケーション環境から削除する完全修飾マスター・データベース名のカンマで区切られたリスト。リストの名前の間に空白を挿入しないでください。
master_table	リストのかわりに、DBMS_UTILITY.DBLINK_ARRAY タイプの PL/SQL 索引付き表でデータベース名を指定できます。

例外

表 53-115 REMOVE_MASTER_DATABASES プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
nonmaster	指定したデータベースの中に、マスター・サイト以外のデータベースが少なくとも 1 つあります。
reconfigerror	指定したデータベースの 1 つが、マスター定義サイトです。
commfailure	残りのマスター・サイトの中に、アクセスできないサイトが少なくとも 1 つあります。

RENAME_SHADOW_COLUMN_GROUP プロシージャ

このプロシージャは、レプリケート表のシャドウ列グループを改名して、同グループを名前付き列グループにします。このプロシージャを実行する場合、レプリケート表のマスター・グループを停止する必要はありません。

構文

```
DBMS_REPCAT.RENAME_SHADOW_COLUMN_GROUP (
   sname                IN VARCHAR2,
   oname                IN VARCHAR2,
   new_col_group_name   IN VARCHAR2)
```

パラメータ

表 53-116 RENAME_SHADOW_COLUMN_GROUP プロシージャのパラメータ

パラメータ	説明
sname	レプリケート表が置かれているスキーマ。
oname	レプリケート表の名前。
new_col_group_name	新しい列グループの名前。現在、シャドウ列グループにある列は、指定した名前の列グループに配置されます。

例外

表 53-117 RENAME_SHADOW_COLUMN_GROUP プロシージャの例外

例外	説明
missmview	指定したスキーマが存在しません。
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが存在しません。
duplicategroup	作成時に指定した列グループはすでに存在します。

REPCAT_IMPORT_CHECK プロシージャ

このプロシージャは、レプリケート・オブジェクトまたは Oracle Replication が使用しているオブジェクトのエクスポートまたはインポートを実行した後で、マスター・グループ内のオブジェクトの識別子とステータス値が適切であることを確認します。

構文

```
DBMS_REPCAT.REPCAT_IMPORT_CHECK (
   gname      IN   VARCHAR2,
   master     IN   BOOLEAN,
   gowner     IN   VARCHAR2  := 'PUBLIC');
```

パラメータ

表 53-118 REPCAT_IMPORT_CHECK プロシージャのパラメータ

パラメータ	説明
gname	マスター・グループの名前。パラメータを両方とも省略すると、現行のサイトのすべてのマスター・グループがチェックされます。
master	マスター・サイトをチェックする場合はこのパラメータを TRUE に、マテリアライズド・ビュー・サイトをチェックする場合は FALSE に設定します。
gowner	マスター・グループの所有者。

例外

表 53-119 REPCAT_IMPORT_CHECK プロシージャの例外

例外	説明
nonmaster	master は TRUE ですが、データベースはレプリケーション・グループのマスター・サイトではないか、予測したものではありません。
nonmview	master は FALSE ですが、データベースはレプリケーション・グループのマテリアライズド・ビュー・サイトではありません。
missingobject	レプリケーション・グループ内に有効なレプリケート・オブジェクトが存在しません。
missingrepgroup	指定したレプリケート・レプリケーション・グループが存在しません。
missingschema	指定したスキーマが存在しません。

RESUME_MASTER_ACTIVITY プロシージャ

このプロシージャは、レプリケーション環境の休止後、通常のレプリケーション・アクティビティを再開します。

構文

```
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
   gname       IN  VARCHAR2,
   override    IN  BOOLEAN := false);
```

パラメータ

表 53-120 RESUME_MASTER_ACTIVITY プロシージャのパラメータ

パラメータ	説明
gname	マスター・グループの名前。
override	TRUE の場合、保留中の RepCat 管理要求は無視され、通常のレプリケーション・アクティビティが各マスターで可能なかぎり迅速に再開されます。これは、緊急の状況でのみ指定してください。 FALSE の場合は、各マスターの gname に対する保留中の RepCat 管理要求がない場合にかぎり、そのマスターで通常のレプリケーション・アクティビティが再開されます。

例外

表 53-121 RESUME_MASTER_ACTIVITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notquiesced	マスター・グループが休止中または停止中ではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。
notallgenerated	レプリケーション・アクティビティを再開する前に、レプリケーション・サポートを生成してください。

RESUME_PROPAGATION_TO_MDEF プロシージャ

新規マスター・サイトをマスター・グループに、停止せずに追加する際、このプロシージャは、エクスポートが効果的に終了し、マスター・サイトに存在する拡張レプリケーション・グループおよび影響を受けないレプリケーション・グループのマスター定義サイトへの伝播を有効にできることを示します。このプロシージャは、新規マスター・サイトをマスター・グループに追加するために必要なエクスポートが完了した後に実行します。

構文

```
DBMS_REPCAT.RESUME_PROPAGATION_TO_MDEF (
   extension_id                 IN    RAW);
```

パラメータ

表 53-122 RESUME_PROPAGATION_TO_MDEF プロシージャのパラメータ

パラメータ	説明
extension_id	マスター・データベースを停止せずにマスター・グループに追加するための現行の保留中の要求の識別子。extension_id 値を検索するには、DBA_REPSITES_NEW データ・ディクショナリ・ビューおよび DBA_REPEXTENSIONS データ・ディクショナリ・ビューに問い合せます。

例外

表 53-123 RESUME_PROPAGATION_TO_MDEF プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
extstinapp	拡張ステータスが不適切です。このプロシージャを実行する場合、拡張ステータスは EXPORTING である必要があります。拡張ステータスをチェックするには、DBA_REPEXTENSIONS データ・ディクショナリ・ビューに問い合せます。
dbnotcompatible	機能がデータベースのバージョンと互換性がありません。データベースはいずれも 9.0.1 以上の互換性レベルを持つものにする必要があります。

SEND_OLD_VALUES プロシージャ

表内の行を更新または削除する場合、レプリケート表の各非キー列の遅延トランザクションを伝播する際に、元の列値を送信することもできます。min_communication を TRUE に設定した場合のデフォルトは次のとおりです。

- 削除行の場合は、すべての列の元の値が送信されます。
- 更新行の場合は、キー列および列グループ内の修正済み列の元の値が送信されます。

DBMS_REPCAT.SEND_OLD_VALUES をマスター定義サイトで起動すると、すべてのマスター・サイトおよびマテリアライズド・ビュー・サイトでこの動作を変更できます。その場合は、すべてのマスター・サイトおよび各マテリアライズド・ビュー・サイトでレプリケーション・サポートを生成します。

ユーザー定義型を使用する場合は、列オブジェクトのリーフ属性を指定することも、列オブジェクト全体を指定することもできます。たとえば、cust_address という名前の列オブジェクトが属性として street_address を持つ場合、column_list パラメータまたは column_table パラメータの一部に cust_address.street_address を指定することも、cust_address のみを指定することもできます。

構文

```
DBMS_REPCAT.SEND_OLD_VALUES(
   sname           IN  VARCHAR2,
   oname           IN  VARCHAR2,
   { column_list   IN  VARCHAR2,
   | column_table  IN  DBMS_UTILITY.VARCHAR2s | DBMS_UTILITY.LNAME_ARRAY,}
   operation       IN  VARCHAR2 := 'UPDATE',
   send            IN  BOOLEAN  := true );
```

注意： このプロシージャはオーバーロードされています。column_list パラメータと column_table パラメータは、両方同時には指定できません。

パラメータ

表 53-124 SEND_OLD_VALUES プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマ。
oname	レプリケート表の名前。ネストした表の記憶表がこの表に該当します。
column_list	表内の列のカンマで区切られたリスト。エントリ間に空白を挿入しないでください。
column_table	リストのかわりに、列名を含む DBMS_REPCAT.VARCHAR2S タイプまたは DBMS_UTILITY.LNAME_ARRAY の PL/SQL 索引付き表を使用できます。最初の列名は位置 1、2 番目は位置 2、以下同様に設定されている必要があります。 列オブジェクトの属性を指定すると、30 バイト以上の列名が発生することがありますが、その場合は DBMS_UTILITY.LNAME_ARRAY を使用します。
operation	有効な値は、update、delete またはアスタリスクのワイルド・カード '*'（更新および削除を意味します）です。
send	TRUE の場合は、指定した列の元の値が送信されます。FALSE の場合は、指定した列の元の値が送信されません。指定外の列および指定外の操作には影響しません。 マスター定義サイトでは、表の min_communication が TRUE になると、指定した変更がすぐに有効となります。変更内容がマスター・サイトまたはマテリアライズド・ビュー・サイトで有効になるのは、min_communication に TRUE を設定して、次回そのサイトでレプリケーション・サポートを生成したときです。

注意： operation パラメータを使用すると、行の削除時または更新時に、非キー列の元の値を送信するかどうかを決定できます。元の値を送信しないと、元の値のかわりに NULL が送信され、更新または削除の適用時に送信先の現行の列値と元の値が等しいとみなされます。

Oracle のデフォルト動作を変更する前に、SEND_OLD_VALUES プロシージャを使用したデータ伝播の最小化について『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

例外

表 53-125 SEND_OLD_VALUES プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーション情報を待機している表として指定のスキーマ内に存在していません。
missingcolumn	表に存在しない列が少なくとも 1 つあります。
notquiesced	マスター・グループが停止中ではありません。
typefailure	無効な操作が指定されています。
keysendcomp	指定した列は表のキー列です。
dbnotcompatible	機能がデータベースのバージョンと互換性がありません。通常は、列オブジェクトの属性を送信しようとしたときにこの例外が発生します。この場合、データベースはいずれも 9.0.1 以上の互換性レベルを持つものにする必要があります。

SET_COLUMNS プロシージャ

このプロシージャは、行レベル・レプリケーションの使用時に、主キーのかわりに代替列または列グループを使用するために、表のどの列を比較するかを判断します。このプロシージャは、マスター定義サイトからコールする必要があります。

列オブジェクトを使用するときに、列オブジェクトの属性が主キーまたは主キーの一部として使用できる場合、その属性は代替キー列に組み込むことができます。たとえば、cust_address という名前の列オブジェクトが VARCHAR2 属性として street_address を持つ場合、column_list パラメータまたは column_table パラメータの一部に cust_address.street_address を指定できます。ただし、列オブジェクト全体（cust_address）を指定することはできません。

ネストした表の行の記憶表の場合、このプロシージャでは NESTED_TABLE_ID が代替キー列として受け入れられます。

オブジェクト表を使用する場合、代替キー列は指定できません。オブジェクト識別子（OID）がオブジェクト表にシステム生成された場合、そのオブジェクト表の OID 列が同表のキーとして使用されます。OID がオブジェクト表にユーザー定義された場合、そのオブジェクト表の主キーがキーとして使用されます。

代替キー列に使用できない列のタイプは次のとおりです。

- 列オブジェクトの LOB または LOB 属性
- 列オブジェクトのコレクションまたはコレクション属性
- REF
- 列オブジェクト全体

> **関連項目：** 主キー列の制約の詳細は、『Oracle9*i* SQL リファレンス』の *constraint_clause* を参照してください。

構文

```
DBMS_REPCAT.SET_COLUMNS (
   sname           IN    VARCHAR2,
   oname           IN    VARCHAR2,
   { column_list   IN    VARCHAR2
   | column_table  IN    DBMS_UTILITY.NAME_ARRAY | DBMS_UTILITY.LNAME_ARRAY } );
```

> **注意：** このプロシージャはオーバーロードされています。column_list パラメータと column_table パラメータは、両方同時には指定できません。

パラメータ

表 53-126 SET_COLUMNS プロシージャのパラメータ

パラメータ	説明
sname	表が置かれているスキーマ。
oname	表の名前。
column_list	表の中で主キーとして使用する列のカンマで区切られたリスト。エントリ間に空白を挿入しないでください。
column_table	リストのかわりに、列名を含んだ DBMS_UTILITY.NAME_ARRAY タイプまたは DBMS_UTILITY.LNAME_ARRAY の PL/SQL 索引付き表を使用できます。最初の列名は位置 1、2 番目は位置 2、以下同様に設定されている必要があります。 列オブジェクトの属性を指定すると、30 バイト以上の列名が発生することがありますが、その場合は DBMS_UTILITY.LNAME_ARRAY を使用します。

例外

表 53-127 SET_COLUMNS プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
missingobject	指定したオブジェクトが、行レベル・レプリケーション情報を待機している表として指定のスキーマ内に存在していません。
missingcolumn	表に存在しない列が少なくとも 1 つあります。
notquiesced	レプリケーション・グループが休止中または停止中ではありません。

SPECIFY_NEW_MASTERS プロシージャ

このプロシージャは、既存のレプリケーション・グループに追加するマスター・サイトを、そのグループを休止させずに指定します。このプロシージャは、指定したマスター・グループのマスター定義サイトで実行する必要があります。

必要に応じて、新規マスター・サイトをマスター・グループに追加するプロセスを追跡する extension_id がこのプロシージャで作成されます。この extension_id は、同プロセスの様々な段階で実行される他のプロシージャで使用します。extension_id の情報は、DBA_REPSITES_NEW データ・ディクショナリ・ビューおよび DBA_REPEXTENSIONS データ・ディクショナリ・ビューに示されます。

指定されたレプリケーション・グループの DBA_REPSITES_NEW データ・ディクショナリ・ビューに新規マスター・サイトが追加されます。このプロシージャは、指定したレプリケーション・グループに任意の回数実行できます。複数回実行すると、指定したレプリケーション・グループのローカル DBA_REPSITES_NEW データ・ディクショナリ・ビューのマスターが、master_list/master_table パラメータで指定したマスターに置き換えられます。

このプロシージャは、ADD_NEW_MASTERS プロシージャの前に実行する必要があります。ADD_NEW_MASTERS プロシージャを実行しないかぎり、マスター・グループに新規マスター・サイトは追加されません。

関連項目： 53-10 ページ「ADD_NEW_MASTERS プロシージャ」

構文

```
DBMS_REPCAT.SPECIFY_NEW_MASTERS (
   gname           IN    VARCHAR2,
   { master_list   IN    VARCHAR2
   | master_table  IN   DBMS_UTILITY.DBLINK_ARRAY});
```

注意： このプロシージャはオーバーロードされています。master_list パラメータと master_table パラメータは、両方同時には指定できません。

パラメータ

表 53-128 SPECIFY_NEW_MASTERS プロシージャのパラメータ

パラメータ	説明
gname	新規マスター・サイトを追加するマスター・グループ。
master_list	マスター・グループに追加する新規マスター・サイトのカンマで区切られたリスト。新規マスター・サイトのみをリストし、既存のマスター・サイトはリストしません。各サイト名の間には空白を入れないでください。 master_list が NULL の場合は、指定したレプリケーション・グループのすべてのマスター・サイトが DBA_REPSITES_NEW データ・ディクショナリ・ビューから削除されます。マスター・グループが拡張中でないことを示すには NULL を指定します。
master_table	マスター・グループに追加する新規マスター・サイトがリストされた表。新規マスター・サイトのみをリストし、既存のマスター・サイトはリストしません。最初のマスター・サイトは位置 1、2 番目は位置 2、以下同様に設定されている必要があります。 この表が空の場合は、指定したレプリケーション・グループのすべてのマスター・サイトが DBA_REPSITES_NEW データ・ディクショナリ・ビューから削除されます。マスター・グループが拡張中でないことを示すには空の表を使用します。

例外

表 53-129 SPECIFY_NEW_MASTERS プロシージャの例外

例外	説明
duplicaterepgroup	追加しようとしているマスター・サイトは、すでにマスター・グループに組み込まれています。
nonmasterdef	起動サイトがマスター定義サイトではありません。
propmodenotallowed	同期伝播モードはこの操作では使用できません。使用できる伝播モードは非同期です。
extstinapp	ステータス付きの拡張要求は使用できません。マスター・グループの extension_id が存在しないようにするか、extension_id ステータスを READY にする必要があります。マスター・サイトの各 extension_id のステータスは、DBA_REPEXTENSIONS データ・ディクショナリ・ビューに示されます。
dbnotcompatible	機能がデータベースのバージョンと互換性がありません。データベースはいずれも 9.0.1 以上の互換性レベルを持つものにする必要があります。
notsamecq	各マスター・グループの接続識別子は同じではありません。

SUSPEND_MASTER_ACTIVITY プロシージャ

このプロシージャは、マスター・グループのレプリケーション・アクティビティを中断します。マスター・グループを停止するには、このプロシージャを使用します。このプロシージャは、マスター定義サイトからコールする必要があります。

構文

```
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
   gname    IN    VARCHAR2);
```

パラメータ

表 53-130 SUSPEND_MASTER_ACTIVITY プロシージャのパラメータ

パラメータ	説明
gname	アクティビティを中断するマスター・グループの名前。

例外

表 53-131 SUSPEND_MASTER_ACTIVITY プロシージャの例外

例外	説明
nonmasterdef	起動サイトがマスター定義サイトではありません。
notnormal	マスター・グループが通常の操作モードではありません。
commfailure	アクセスできないマスター・サイトが少なくとも 1 つあります。

SWITCH_MVIEW_MASTER プロシージャ

このプロシージャは、マテリアライズド・ビュー・グループのマスター・サイトを別のマスター・サイトに変更します。影響するマテリアライズド・ビューは完全にリフレッシュされ、必要に応じて、トリガーおよびトリガーと関連するパッケージが再生成されます。このプロシージャは、マスター・サイトの変更前に、元のマスター・サイトにキューを送信しません。

マテリアライズド・ビューの min_communication が TRUE で、かつ新規マスター・サイトが Oracle7 のマスター・サイトである場合は、min_communication を FALSE に設定して、マテリアライズド・ビューのレプリケーション・サポートを再生成します。

マテリアライズド・ビューの generate_80_compatible が FALSE で、かつ新規マスター・サイトのリリースが Oracle8*i* より前のリリース（Oracle7 または Oracle8）である場合は、generate_80_compatible を TRUE に設定して、マテリアライズド・ビューのレプリケーション・サポートを再生成します。

マテリアライズド・ビューの両方のパラメータは、DBMS_REPCAT.GENERATE_MVIEW_SUPPORT の 1 回のコールで設定できます。

注意： 他のマテリアライズド・ビュー（レベル 2 以上のマテリアライズド・ビュー）を基にしているマテリアライズド・ビューのマスターは切替えできません。別のマスターを基にする場合、そのマテリアライズド・ビューを削除し再作成する必要があります。

関連項目： 53-81 ページ「GENERATE_MVIEW_SUPPORT プロシージャ」

構文

```
DBMS_REPCAT.SWITCH_MVIEW_MASTER (
   gname          IN   VARCHAR2,
   master         IN   VARCHAR2,
   gowner         IN   VARCHAR2  := 'PUBLIC');
```

パラメータ

表 53-132 SWITCH_MVIEW_MASTER プロシージャのパラメータ

パラメータ	説明
gname	マスター・サイトを変更するマテリアライズド・ビュー・グループの名前。
master	マテリアライズド・ビュー・グループに使用する新規マスター・サイトの完全修飾されたデータベース名。
gowner	マテリアライズド・ビュー・グループの所有者。

例外

表 53-133 SWITCH_MVIEW_MASTER プロシージャの例外

例外	説明
nonmview	起動サイトがマテリアライズド・ビュー・サイトではありません。
nonmaster	指定したデータベースはマスター・サイトではありません。
commfailure	指定したデータベースにアクセスできません。
missingrepgroup	マテリアライズド・ビュー・グループが存在しません。
qrytoolong	マテリアライズド・ビュー定義の問合せが 32KB を超えています。
alreadymastered	ローカル・サイトに、同じグループ名を持ち、元のマスター・サイトでマスターとなっている別のマテリアライズド・ビュー・グループがあります。

UNDO_ADD_NEW_MASTERS_REQUEST プロシージャ

このプロシージャは、指定した extension_id の SPECIFY_NEW_MASTERS プロシージャおよび ADD_NEW_MASTERS プロシージャで行った変更をすべて取り消します。

このプロシージャは、1 つのマスター・サイト（マスター定義サイトの場合があります）で実行され、そのマスター・サイトにのみ影響を与えます。要求で影響を受ける 1 つのマスター・サイトでこのプロシージャを実行する場合は、その要求で影響を受けるすべての新規マスター・サイトおよび既存のマスターサイトでも実行する必要があります。extension_id で影響を受ける新規マスター・サイトを確認するには、DBA_REPSITES_NEW データ・ディクショナリ・ビューに問い合せます。このデータ・ディクショナリ・ビューにはレプリケーション・グループ名もリストされるため、このプロシージャはそのレプリケーション・グループの既存の全マスター・サイトで実行する必要があります。

注意： このプロシージャは通常はコールしません。このプロシージャを使用するのは、停止操作をせずに新規マスターを追加することが、1 つ以上のマスター・サイトで続行できない場合のみです。このプロシージャは、SPECIFY_NEW_MASTERS プロシージャおよび ADD_NEW_MASTERS プロシージャの後で、しかも RESUME_PROPAGATION_TO_MDEF プロシージャおよび PREPARE_INSTANTIATED_MASTER プロシージャの前に実行します。

このプロシージャは、特定の extension_id の RESUME_PROPAGATION_TO_MDEF または PREPARE_INSTANTIATED_MASTER を実行した後には実行しないでください。

関連項目：

- 53-107 ページ「SPECIFY_NEW_MASTERS プロシージャ」
- 53-10 ページ「ADD_NEW_MASTERS プロシージャ」
- 53-101 ページ「RESUME_PROPAGATION_TO_MDEF プロシージャ」
- 53-86 ページ「PREPARE_INSTANTIATED_MASTER プロシージャ」

構文

```
DBMS_REPCAT.UNDO_ADD_NEW_MASTERS_REQUEST (
   extension_id          IN RAW,
   drop_contents         IN BOOLEAN := TRUE);
```

パラメータ

表 53-134 UNDO_ADD_NEW_MASTERS_REQUEST プロシージャのパラメータ

パラメータ	説明
extension_id	マスター・データベースを停止せずにマスター・グループに追加するための現行の保留中の要求の識別子。extension_id 値を検索するには、DBA_REPSITES_NEW データ・ディクショナリ・ビューおよび DBA_REPEXTENSIONS データ・ディクショナリ・ビューに問い合せます。
drop_contents	ローカル・サイトで拡張中の新規レプリケーション・グループのオブジェクトの内容を削除するには、デフォルトの TRUE を指定します。この内容を保持するには、FALSE を指定します。

例外

表 53-135 UNDO_ADD_NEW_MASTERS_REQUEST プロシージャの例外

例外	説明
dbnotcompatible	機能がデータベースのバージョンと互換性がありません。データベースはいずれも 9.0.1 以上の互換性レベルを持つものにする必要があります。
typefail	指定したパラメータ値が不適切です。

UNREGISTER_MVIEW_REPGROUP プロシージャ

このプロシージャは、マテリアライズド・ビュー・グループを DBA_REGISTERED_MVIEW_GROUPS から削除して、各マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトのマテリアライズド・ビュー管理を容易にします。このプロシージャは、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで実行します。

構文

```
DBMS_REPCAT.UNREGISTER_MVIEW_REPGROUP (
   gname      IN   VARCHAR2,
   mviewsite  IN   VARCHAR2
   gowner     IN   VARCHAR2  := 'PUBLIC');
```

パラメータ

表 53-136 UNREGISTER_MVIEW_REPGROUP プロシージャのパラメータ

パラメータ	説明
gname	登録解除するマテリアライズド・ビュー・グループの名前。
mviewsite	マテリアライズド・ビュー・サイトのグローバル名。
gowner	マテリアライズド・ビュー・グループの所有者。

VALIDATE ファンクション

このファンクションは、マルチマスター・レプリケーション環境のキー状態が正しいかどうかを検証します。

構文

```
DBMS_REPCAT.VALIDATE (
   gname                IN  VARCHAR2,
   check_genflags       IN  BOOLEAN := false,
   check_valid_objs     IN  BOOLEAN := false,
   check_links_sched    IN  BOOLEAN := false,
   check_links          IN  BOOLEAN := false,
   error_table          OUT DBMS_REPCAT.VALIDATE_ERR_TABLE)
  RETURN BINARY_INTEGER;

DBMS_REPCAT.VALIDATE (
   gname                IN  VARCHAR2,
   check_genflags       IN  BOOLEAN := false,
   check_valid_objs     IN  BOOLEAN := false,
   check_links_sched    IN  BOOLEAN := false,
   check_links          IN  BOOLEAN := false,
   error_msg_table      OUT DBMS_UTILITY.UNCL_ARRAY,
   error_num_table      OUT DBMS_UTILITY.NUMBER_ARRAY )
  RETURN BINARY_INTEGER;
```

注意： このファンクションはオーバーロードされています。VALIDATE の戻り値は、検出されたエラーの数です。このファンクションの OUT パラメータには、検出されたエラーが戻されます。53-114 ページの「構文」に示されている最初のインタフェース・ファンクションでは、error_table はレコードの配列で構成されています。各レコードには、VARCHAR2 および NUMBER があります。文字列フィールドにはエラー・メッセージが格納され、番号フィールドには Oracle エラー番号が格納されます。

53-114 ページの「構文」に示されている 2 番目のインタフェース・ファンクションは、2 つの OUT 配列があること以外は同じです。VARCHAR2 配列にエラー・メッセージが格納され、NUMBER 配列にエラー番号が格納されます。

パラメータ

表 53-137 VALIDATE ファンクションのパラメータ

パラメータ	説明
gname	検証するマスター・グループの名前。
check_genflags	グループ内のオブジェクトがすべて生成済みであるかどうかをチェックします。このチェックは、マスター定義サイトでのみ実行してください。
check_valid_objs	グループ内のオブジェクトに対応する基礎オブジェクトが有効かどうかをチェックします。このチェックは、マスター定義サイトでのみ実行してください。マスター定義サイトは他のすべてのサイトを調べて、基礎となっているオブジェクトが有効であることを確認します。オブジェクトの有効性は、接続ユーザーのスキーマ内でチェックされます。
check_links_sched	リンクの実行がスケジュールされているかどうかをチェックします。このチェックは、各マスター・サイトで起動してください。
check_links	接続ユーザー（レプリケーション管理者）およびプロパゲータに、レプリケーションが適切に動作するための正しいリンクがあるかどうかをチェックします。リンクがデータベースに存在していて、アクセス可能であることを確認します。このチェックは、各マスター・サイトで起動してください。
error_table	検出されたすべてのエラーのメッセージと番号を戻します。
error_msg_table	検出されたすべてのエラーのメッセージを戻します。
error_num_table	検出されたすべてのエラーの番号を戻します。

例外

表 53-138 VALIDATE ファンクションの例外

例外	説明
missingdblink	データベース・リンクがレプリケーション・プロパゲータのスキーマ内に存在していないか、またはスケジュールされていません。データベース・リンクがデータベースに存在していてアクセス可能であること、および実行がスケジュールされていることを確認してください。
dblinkmismatch	ローカル・ノードのデータベース・リンク名が、そのリンクがアクセスするデータベースのグローバル名と一致しません。 GLOBAL_NAMES 初期化パラメータに TRUE が設定され、そのリンク名がグローバル名と一致していることを確認してください。

表 53-138 VALIDATE ファンクションの例外（続き）

例外	説明
dblinkuidmismatch	ローカル・ノードのレプリケーション管理ユーザーのユーザー名、およびそのデータベース・リンクに対応するノードのユーザー名が同じではありません。Oracle Replication では、両方のユーザーが同じであることが必要です。ローカル・ノードのレプリケーション管理ユーザーのユーザー ID、およびそのデータベース・リンクに対応するノードのユーザー ID を同じにしてください。
objectnotgenerated	オブジェクトが、他のマスター・サイトで生成されていないか、または生成中です。マスター定義サイトのオブジェクトに対して GENERATE_REPLICATION_SUPPORT および DO_DEFERRED_REPCAT_ADMIN をコールして、オブジェクトを確実に生成してください。
opnotsupported	レプリケーション・グループが Oracle8 より前のノードでレプリケートされている場合は、操作がサポートされていません。マスター・グループのすべてのノードで、Oracle8 以上が実行されていることを確認してください。

使用上の注意

VALIDATE の戻り値は、検出されたエラーの数です。このファンクションの OUT パラメータには、検出されたエラーが戻されます。最初のインタフェース・ファンクションでは、error_table はレコードの配列で構成されています。各レコードには、VARCHAR2 および NUMBER があります。文字列フィールドにはエラー・メッセージが格納され、番号フィールドには Oracle エラー番号が格納されます。

2 番目のインタフェースは、OUT 配列が 2 つある以外は最初のインタフェースと同じです。VARCHAR2 配列にエラー・メッセージが格納され、NUMBER 配列にエラー番号が格納されます。

WAIT_MASTER_LOG プロシージャ

このプロシージャは、マスター・サイトに非同期で伝播された変更内容が適用されたかどうかを判断します。

構文

```
DBMS_REPCAT.WAIT_MASTER_LOG (
   gname          IN    VARCHAR2,
   record_count   IN    NATURAL,
   timeout        IN    NATURAL,
   true_count     OUT   NATURAL);
```

パラメータ

表 53-139 WAIT_MASTER_LOG プロシージャのパラメータ

パラメータ	説明
gname	マスター・グループの名前。
record_count	未完了のアクティビティ件数がこのしきい値以下になるたびにプロシージャに戻ります。
timeout	プロシージャが戻るまで待機する最大秒数。
true_count (out パラメータ)	未完了のアクティビティの数を戻します。

例外

表 53-140 WAIT_MASTER_LOG プロシージャの例外

例外	説明
nonmaster	起動サイトがマスター・サイトではありません。

54

DBMS_REPCAT_ADMIN

DBMS_REPCAT_ADMIN によって、対称型レプリケーション機能に必要な権限を付与したユーザーを作成できます。

この章では、次の項目について説明します。

DBMS_REPCAT_ADMIN サブプログラムの要約

表 54-1 DBMS_REPCAT_ADMIN パッケージのサブプログラム

サブプログラム	説明
「GRANT_ADMIN_ANY_SCHEMA プロシージャ」54-2 ページ	現行のサイトでレプリケーション・グループを管理するために必要な権限を、レプリケーション管理者に付与します。
「GRANT_ADMIN_SCHEMA プロシージャ」54-3 ページ	現行のサイトでスキーマを管理するために必要な権限を、レプリケーション管理者に付与します。
「REGISTER_USER_REPGROUP プロシージャ」54-4 ページ	リモート・サイトで使用するための代理マテリアライズド・ビュー管理者権限または受信者権限を、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで割り当てます。
「REVOKE_ADMIN_ANY_SCHEMA プロシージャ」54-6 ページ	GRANT_ADMIN_ANY_SCHEMA によって付与された権限およびロールを、レプリケーション管理者から取り消します。
「REVOKE_ADMIN_SCHEMA プロシージャ」54-7 ページ	GRANT_ADMIN_SCHEMA によって付与された権限およびロールを、レプリケーション管理者から取り消します。
「UNREGISTER_USER_REPGROUP プロシージャ」54-7 ページ	REGISTER_USER_REPGROUP プロシージャによって付与された権限およびロールを、代理マテリアライズド・ビュー管理者または受信者から取り消します。

GRANT_ADMIN_ANY_SCHEMA プロシージャ

このプロシージャは、現行のサイトでレプリケーション・グループを管理するために必要な権限を、レプリケーション管理者に付与します。

構文

```
DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_SCHEMA (
   username  IN   VARCHAR2);
```

パラメータ

表 54-2 GRANT_ADMIN_ANY_SCHEMA プロシージャのパラメータ

パラメータ	説明
username	現行のサイトでレプリケーション・グループを管理するために必要な権限およびロールを付与するレプリケーション管理者の名前。

例外

表 54-3 GRANT_ADMIN_ANY_SCHEMA プロシージャの例外

例外	説明
ORA-01917	ユーザーが存在しません。

GRANT_ADMIN_SCHEMA プロシージャ

このプロシージャは、現行のサイトでスキーマを管理するために必要な権限をレプリケーション管理者に付与します。このプロシージャは、レプリケーション・グループが単一のスキーマ内にある場合に最も役立ちます。

構文

```
DBMS_REPCAT_ADMIN.GRANT_ADMIN_SCHEMA (
   username IN VARCHAR2);
```

パラメータ

表 54-4 GRANT_ADMIN_SCHEMA プロシージャのパラメータ

パラメータ	説明
username	レプリケーション管理者の名前。このユーザーには、現行のサイトでレプリケーション・グループ内の同名のスキーマを管理するために必要な権限およびロールが付与されます。

例外

表 54-5 GRANT_ADMIN_SCHEMA プロシージャの例外

例外	説明
ORA-01917	ユーザーが存在しません。

REGISTER_USER_REPGROUP プロシージャ

このプロシージャは、リモート・サイトで使用するための代理マテリアライズド・ビュー管理者権限または受信者権限を、マスター・サイトまたはマスター・マテリアライズド・ビュー・サイトで割り当てます。このプロシージャで付与されるのは、代理マテリアライズド・ビュー管理者または受信者に必要な権限のみです。GRANT_ADMIN_SCHEMA または GRANT_ADMIN_ANY_SCHEMA プロシージャで付与される強力な権限は付与されません。

構文

```
DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP (
   username         IN   VARCHAR2,
   privilege_type   IN   VARCHAR2,
   {list_of_gnames  IN   VARCHAR2 |
   table_of_gnames  IN   DBMS_UTILITY.NAME_ARRAY)};
```

注意： このプロシージャはオーバーロードされています。list_of_gnames パラメータと table_of_gnames パラメータは、両方同時には指定できません。

パラメータ

表 54-6 REGISTER_USER_REPGROUP プロシージャのパラメータ

パラメータ	説明
username	代理マテリアライズド・ビュー管理者権限または受信者権限のいずれかを付与するユーザーの名前。
privilege_type	割り当てる権限タイプ。privilege_type の定義には次の値を使用します。 ■ receiver: 受信者権限 ■ proxy_snapadmin: 代理マテリアライズド・ビュー管理者権限
list_of_gnames	ユーザーの受信者権限を登録するレプリケーション・グループのカンマで区切られたリスト。リストのエントリの間に空白を挿入しないでください。list_of_gnames を NULL に設定すると、このプロシージャのコール時点で認識されていないレプリケーション・グループも含め、すべてのレプリケーション・グループに対してそのユーザーが登録されます。list_of_gnames を NULL に設定するには、名前表記法を使用する必要があります。リスト内に無効なレプリケーション・グループがあると、リスト全体の登録に失敗します。
table_of_gnames	ユーザーの受信者権限を登録するレプリケーション・グループの PL/SQL 索引付き表。PL/SQL 索引付き表は、DBMS_UTILITY.NAME_ARRAY タイプにしてください。この表は 1 が基準（1 の位置から開始し、1 ずつ増加します）です。値に NULL を使用すると、すべてのレプリケーション・グループに対してそのユーザーが登録されます。表内に無効なレプリケーション・グループがあると、表全体の登録に失敗します。

例外

表 54-7 REGISTER_USER_REPGROUP プロシージャの例外

例外	説明
nonmaster	指定したレプリケーション・グループが存在しないか、起動データベースがマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトではありません。
ORA-01917	ユーザーが存在しません。
typefailure	権限タイプの指定に誤りがあります。

REVOKE_ADMIN_ANY_SCHEMA プロシージャ

このプロシージャは、GRANT_ADMIN_ANY_SCHEMA によって付与された権限およびロールを、レプリケーション管理者から取り消します。

注意： GRANT_ADMIN_ANY_SCHEMA とは別に付与された同一の権限およびロールも取り消されます。

構文

```
DBMS_REPCAT_ADMIN.REVOKE_ADMIN_ANY_SCHEMA (
   username IN VARCHAR2);
```

パラメータ

表 54-8 REVOKE_ADMIN_ANY_SCHEMA プロシージャのパラメータ

パラメータ	説明
username	権限を取り消すレプリケーション管理者の名前。

例外

表 54-9 REVOKE_ADMIN_ANY_SCHEMA プロシージャの例外

例外	説明
ORA-01917	ユーザーが存在しません。

REVOKE_ADMIN_SCHEMA プロシージャ

このプロシージャは、GRANT_ADMIN_SCHEMA によって付与された権限およびロールを、レプリケーション管理者から取り消します。

> **注意：** GRANT_ADMIN_SCHEMA とは別に付与された同一の権限およびロールも取り消されます。

構文

```
DBMS_REPCAT_ADMIN.REVOKE_ADMIN_SCHEMA (
   username IN VARCHAR2);
```

パラメータ

表 54-10 REVOKE_ADMIN_SCHEMA プロシージャのパラメータ

パラメータ	説明
username	権限を取り消すレプリケーション管理者の名前。

例外

表 54-11 REVOKE_ADMIN_SCHEMA プロシージャの例外

例外	説明
ORA-01917	ユーザーが存在しません。

UNREGISTER_USER_REPGROUP プロシージャ

このプロシージャは、REGISTER_USER_REPGROUP プロシージャによって付与された権限およびロールを、代理マテリアライズド・ビュー管理者または受信者から取り消します。

構文

```
DBMS_REPCAT_ADMIN.UNREGISTER_USER_REPGROUP (
   username         IN   VARCHAR2,
   privilege_type   IN   VARCHAR2,
   {list_of_gnames  IN   VARCHAR2 |
   table_of_gnames  IN   DBMS_UTILITY.NAME_ARRAY)};
```

注意： このプロシージャはオーバーロードされています。list_of_gnames パラメータと table_of_gnames パラメータは、両方同時には指定できません。

パラメータ

表 54-12 UNREGISTER_USER_REPGROUP プロシージャのパラメータ

パラメータ	説明
username	登録解除するユーザーの名前。
privilege_type	取り消す権限タイプ。privilege_type の定義には次の値を使用します。 ■ receiver: 受信者権限 ■ proxy_snapadmin: 代理マテリアライズド・ビュー管理者権限
list_of_gnames	ユーザーの受信者権限を登録解除するレプリケーション・グループのカンマで区切られたリスト。リストのエントリの間に空白を挿入しないでください。list_of_gnames を NULL に設定すると、登録されているすべてのレプリケーション・グループに対する登録が解除されます。list_of_gnames を NULL に設定するには、名前表記法を使用する必要があります。リスト内に無効なレプリケーション・グループがあると、リスト全体の登録解除に失敗します。
table_of_gnames	ユーザーの受信者権限を登録解除するレプリケーション・グループの PL/SQL 索引付き表。PL/SQL 索引付き表は、DBMS_UTILITY.NAME_ARRAY タイプにしてください。この表は 1 が基準（1 の位置から開始し、1 ずつ増加します）です。値に NULL を使用すると、登録されているすべてのレプリケーション・グループに対してそのユーザーの登録が解除されます。表内に無効なレプリケーション・グループがあると、表全体の登録解除に失敗します。

例外

表 54-13 UNREGISTER_USER_REPGROUP プロシージャの例外

例外	説明
nonmaster	指定したレプリケーション・グループが存在しないか、起動データベースがマスター・サイトまたはマスター・マテリアライズド・ビュー・サイトではありません。
ORA-01917	ユーザーが存在しません。
typefailure	権限タイプの指定に誤りがあります。

55

DBMS_REPCAT_INSTANTIATE

DBMS_REPCAT_INSTANTIATE パッケージは、配置テンプレートをインスタンス化します。

この章では、次の項目について説明します。

DBMS_REPCAT_INSTANTIATE サブプログラムの要約

表 55-1 DBMS_REPCAT_INSTANTIATE パッケージのサブプログラム

サブプログラム	説明
「DROP_SITE_INSTANTIATION プロシージャ」55-2 ページ	DBA_REPCAT_TEMPLATE_SITES ビューからターゲット・サイトを削除します。
「INSTANTIATE_OFFLINE ファンクション」55-3 ページ	マスター・サイトでスクリプトを生成します。このスクリプトは、オフライン時にリモート・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー環境を作成するために使用します。
「INSTANTIATE_ONLINE ファンクション」55-5 ページ	マスター・サイトでスクリプトを生成します。このスクリプトは、オンライン時にリモート・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー環境を作成するために使用します。

DROP_SITE_INSTANTIATION プロシージャ

このプロシージャは、ターゲット・サイトのテンプレート・インスタンス化を削除します。また、マスター・サイトの関連メタデータをすべて削除し、指定したサイトにおけるマテリアライズド・ビューのリフレッシュを使用禁止にします。このプロシージャは、テンプレートを最初にインスタンス化したユーザーとして実行する必要があります。テンプレートをインスタンス化したユーザーを調べるには、ALL_REPCAT_TEMPLATE_SITES ビューを問い合せてください。

構文

```
DBMS_REPCAT_INSTANTIATE.DROP_SITE_INSTANTIATION(
     refresh_template_name  IN   VARCHAR2,
     site_name              IN   VARCHAR2);
```

表 55-2 DROP_SITE_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除する配置テンプレートの名前。
site_name	指定したテンプレート・インスタンス化を削除するマスター・サイトを識別します。

INSTANTIATE_OFFLINE ファンクション

このファンクションは、マスター・サイトでファイルを生成します。このファイルは、オフライン時にリモート・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー環境を作成するために使用します。生成されたファイルはオフライン・インスタンス化ファイルです。このファイルは、長時間マスター・サイトに接続したままにできないリモート・マテリアライズド・ビュー・サイトで使用してください。

これは、リモート・マテリアライズド・ビュー・サイトがラップトップの場合に便利な方法です。レプリケーション管理ツールのパッケージ作成インタフェースを使用して、生成したファイルおよびデータを 1 つのファイルにパッケージ化すると、このファイルを FTP サイトに転送したり、CD-ROM やフロッピィ・ディスクなどにロードできます。

このファンクションで生成されたスクリプトは、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納され、配置テンプレートの配布時に、レプリケーション管理ツールなどの Oracle のツール製品で使用されます。このファンクションで戻される数値は、USER_REPCAT_TEMP_OUTPUT ビューから該当する情報を取り出すために使用されます。

このパブリック・ファンクションを実行するユーザーは、指定したサイトでインスタンス化されたテンプレートの登録ユーザーになります。

注意： このファンクションは、配置テンプレートのオフライン・インスタンス化の実行時に使用されます。

このファンクションを、DBMS_OFFLINE_OG パッケージ内のプロシージャ（マスター表のオフライン・インスタンス化の実行に使用）や DBMS_OFFLINE_SNAPSHOT パッケージ内のプロシージャ（マテリアライズド・ビューのオフライン・インスタンス化の実行に使用）と混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

関連項目：

- 『Oracle9*i* アドバンスト・レプリケーション』
- レプリケーション管理ツールのオンライン・ヘルプ

構文

```
DBMS_REPCAT_INSTANTIATE.INSTANTIATE_OFFLINE(
     refresh_template_name   IN   VARCHAR2,
     site_name               IN   VARCHAR2,
     runtime_parm_id         IN   NUMBER    := -1e-130,
     next_date               IN   DATE      := SYSDATE,
     interval                IN   VARCHAR2  := 'SYSDATE + 1',
     use_default_gowner      IN   BOOLEAN   := true)
     return NUMBER;
```

表 55-3 INSTANTIATE_OFFLINE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_name	配置テンプレートをインスタンス化するリモート・サイトの名前。
runtime_parm_id	INSERT_RUNTIME_PARMS プロシージャを使用してランタイム・パラメータ値を定義している場合は、ランタイム・パラメータの作成時に使用した ID（GET_RUNTIME_PARM_ID ファンクションを使用して取り出された ID）を指定します。
next_date	リフレッシュ・グループの作成時に使用される次回リフレッシュ日付の値。
interval	リフレッシュ・グループの作成時に使用されるリフレッシュ間隔。
use_default_gowner	TRUE の場合、作成されたすべてのマテリアライズド・ビュー・グループはデフォルト・ユーザー PUBLIC が所有します。FALSE の場合、作成されたすべてのマテリアライズド・ビュー・グループはインスタンス化を実行したユーザーが所有します。

例外

表 55-4 INSTANTIATE_OFFLINE ファンクションの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
dupl_template_site	マテリアライズド・ビュー・サイトで配置テンプレートがすでにインスタンス化されています。特定のマテリアライズド・ビュー・サイトで配置テンプレートをインスタンス化できるのは 1 度のみです。
not_authorized	ユーザーがインスタンス化しようとした配置テンプレートは、インスタンス化を認可されていません。

戻り値

表 55-5 INSTANTIATE_OFFLINE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	生成したインスタンス化スクリプトを USER_REPCAT_TEMP_OUTPUT ビューで取り出すように選択したとき、output_id に対して生成されるシステム番号を指定します。

INSTANTIATE_ONLINE ファンクション

このファンクションは、マスター・サイトでスクリプトを生成します。このファイルは、オンライン時にリモート・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー環境を作成するために使用します。リモート・マテリアライズド・ビュー・サイトでのインスタンス化プロセスは長くかかる場合があるため（必要な時間は新しいマテリアライズド・ビューに移入されるデータの量によって異なります）、生成されたスクリプトは、マスター・サイトに長時間接続したままにできるリモート・マテリアライズド・ビュー・サイトで使用してください。

このファンクションで生成されたスクリプトは、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納され、配置テンプレートの配布時に、レプリケーション管理ツールなどの Oracle のツール製品で使用されます。このファンクションで戻される数値は、USER_REPCAT_TEMP_OUTPUT ビューから該当する情報を取り出すために使用されます。

このパブリック・ファンクションを実行するユーザーは、指定したサイトでインスタンス化されたテンプレートの登録ユーザーになります。

関連項目：

- 『Oracle9*i* アドバンスト・レプリケーション』
- レプリケーション管理ツールのオンライン・ヘルプ

構文

```
DBMS_REPCAT_INSTANTIATE.INSTANTIATE_ONLINE(
     refresh_template_name   IN   VARCHAR2,
     site_name               IN   VARCHAR2,
     runtime_parm_id         IN   NUMBER    := -1e-130,
     next_date               IN   DATE      := SYSDATE,
     interval                IN   VARCHAR2  := 'SYSDATE + 1',
     use_default_gowner      IN   BOOLEAN   := true)
     return NUMBER;
```

表 55-6 INSTANTIATE_ONLINE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_name	配置テンプレートをインスタンス化するリモート・サイトの名前。
runtime_parm_id	INSERT_RUNTIME_PARMS プロシージャを使用してランタイム・パラメータ値を定義している場合は、ランタイム・パラメータの作成時に使用した ID（GET_RUNTIME_PARM_ID ファンクションを使用して取り出された ID）を指定します。
next_date	リフレッシュ・グループの作成時に使用される次回リフレッシュ日付の値を指定します。
interval	リフレッシュ・グループの作成時に使用されるリフレッシュ間隔を指定します。
use_default_gowner	TRUE の場合、作成されたすべてのマテリアライズド・ビュー・グループはデフォルト・ユーザー PUBLIC が所有します。FALSE の場合、作成されたすべてのマテリアライズド・ビュー・グループはインスタンス化を実行したユーザーが所有します。

表 55-7 INSTANTIATE_ONLINE ファンクションの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
dupl_template_site	マテリアライズド・ビュー・サイトで配置テンプレートがすでにインスタンス化されています。特定のマテリアライズド・ビュー・サイトで配置テンプレートをインスタンス化できるのは 1 度のみです。
not_authorized	ユーザーがインスタンス化しようとした配置テンプレートは、インスタンス化を認可されていません。

戻り値

表 55-8 INSTANTIATE_ONLINE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	生成したインスタンス化スクリプトを USER_REPCAT_TEMP_OUTPUT ビューで取り出すように選択したとき、output_id に対して生成されるシステム番号を指定します。

56

DBMS_REPCAT_RGT

DBMS_REPCAT_RGT は、リフレッシュ・グループ・テンプレートのメンテナンスおよび定義を制御します。

この章では、次の項目について説明します。

DBMS_REPCAT_RGT サブプログラムの要約

表 56-1 DBMS_REPCAT_RGT パッケージのサブプログラム

サブプログラム	説明
「ALTER_REFRESH_TEMPLATE プロシージャ」56-4 ページ	DBA が既存の配置テンプレートを変更できます。
「ALTER_TEMPLATE_OBJECT プロシージャ」56-6 ページ	指定した配置テンプレートに追加されているオブジェクトを変更します。
「ALTER_TEMPLATE_PARM プロシージャ」56-9 ページ	DBA が特定の配置テンプレートのパラメータを変更できます。
「ALTER_USER_AUTHORIZATION プロシージャ」56-11 ページ	DBA_REPCAT_USER_AUTHORIZATIONS ビューの内容を変更します。
「ALTER_USER_PARM_VALUE プロシージャ」56-12 ページ	特定のユーザーに対して定義されている既存のパラメータ値を変更します。
「COMPARE_TEMPLATES ファンクション」56-14 ページ	DBA が 2 つの配置テンプレートの内容を比較できます。
「COPY_TEMPLATE ファンクション」56-16 ページ	DBA が配置テンプレートをコピーできます。
「CREATE_OBJECT_FROM_EXISTING ファンクション」56-18 ページ	既存のデータベース・オブジェクトからテンプレート・オブジェクト定義を作成し、それをターゲット配置テンプレートに追加します。
「CREATE_REFRESH_TEMPLATE ファンクション」56-20 ページ	配置テンプレートを作成します。DBA がテンプレート名、プライベートまたはパブリック・ステータス、およびターゲット・リフレッシュ・グループを定義できます。
「CREATE_TEMPLATE_OBJECT ファンクション」56-22 ページ	ターゲット配置テンプレートのコンテナにオブジェクト定義を追加します。
「CREATE_TEMPLATE_PARM ファンクション」56-25 ページ	特定の配置テンプレートのパラメータを作成します。この結果、リモート・マテリアライズド・ビュー・サイトでカスタム・データ・セットを作成できます。
「CREATE_USER_AUTHORIZATION ファンクション」56-27 ページ	特定のユーザーに、プライベート配置テンプレートのインスタンス化を認可します。
「CREATE_USER_PARM_VALUE ファンクション」56-29 ページ	特定のユーザーに対する配置テンプレートのパラメータ値を事前定義します。
「DELETE_RUNTIME_PARMS プロシージャ」56-32 ページ	INSERT_RUNTIME_PARMS プロシージャを使用して定義したランタイム・パラメータ値を削除します。

表 56-1 DBMS_REPCAT_RGT パッケージのサブプログラム（続き）

サブプログラム	説明
「DROP_ALL_OBJECTS プロシージャ」56-33 ページ	DBA が、配置テンプレートからすべてのまたは特定のオブジェクト・タイプを削除できます。
「DROP_ALL_TEMPLATE_PARMS プロシージャ」56-34 ページ	DBA が、指定した配置テンプレートのテンプレート・パラメータを削除できます。
「DROP_ALL_TEMPLATE_SITES プロシージャ」56-35 ページ	DBA_REPCAT_TEMPLATE_SITES ビューからすべてのエントリを削除します。
「DROP_ALL_TEMPLATES プロシージャ」56-36 ページ	プロシージャがコールされるサイトの配置テンプレートをすべて削除します。
「DROP_ALL_USER_AUTHORIZATIONS プロシージャ」56-36 ページ	DBA が、指定した配置テンプレートに対するユーザー認証をすべて削除できます。
「DROP_ALL_USER_PARM_VALUES プロシージャ」56-37 ページ	特定の配置テンプレートに対するユーザー・パラメータ値を削除します。
「DROP_REFRESH_TEMPLATE プロシージャ」56-38 ページ	配置テンプレートを削除します。
「DROP_SITE_INSTANTIATION プロシージャ」56-39 ページ	DBA_REPCAT_TEMPLATE_SITES ビューからターゲット・サイトを削除します。
「DROP_TEMPLATE_OBJECT プロシージャ」56-40 ページ	特定の配置テンプレートからテンプレート・オブジェクトを削除します。
「DROP_TEMPLATE_PARM プロシージャ」56-42 ページ	DBA_REPCAT_TEMPLATE_PARMS ビューから既存のテンプレート・パラメータを削除します。
「DROP_USER_AUTHORIZATION プロシージャ」56-43 ページ	DBA_REPCAT_USER_AUTHORIZATIONS ビューからユーザー認証エントリを削除します。
「DROP_USER_PARM_VALUE プロシージャ」56-44 ページ	特定の配置テンプレートに対する事前定義ユーザー・パラメータ値を削除します。
「GET_RUNTIME_PARM_ID ファンクション」56-45 ページ	ランタイム・パラメータ値の定義時に使用する ID を取り出します。
「INSERT_RUNTIME_PARMS プロシージャ」56-45 ページ	テンプレートのインスタンス化前にランタイム・パラメータ値を定義します。
「INSTANTIATE_OFFLINE ファンクション」56-47 ページ	マスター・サイトでスクリプトを生成します。このスクリプトは、オフライン時にリモート・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー環境を作成するために使用します。

表 56-1 DBMS_REPCAT_RGT パッケージのサブプログラム（続き）

サブプログラム	説明
「INSTANTIATE_ONLINE ファンクション」56-50 ページ	マスター・サイトでスクリプトを生成します。このスクリプトは、オンライン時にリモート・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー環境を作成するために使用します。
「LOCK_TEMPLATE_EXCLUSIVE プロシージャ」56-52 ページ	配置テンプレートの更新中または変更中に、ユーザーがテンプレートの読込みまたはインスタンス化を実行できないようにします。
「LOCK_TEMPLATE_SHARED プロシージャ」56-53 ページ	指定した配置テンプレートを読取り専用にします。

ALTER_REFRESH_TEMPLATE プロシージャ

このプロシージャでは、DBA が既存の配置テンプレートを変更できます。新規配置テンプレート名、新規リフレッシュ・グループまたは新規所有者の定義、およびパブリック / プライベート・ステータスの変更などを実行できます。

構文

```
DBMS_REPCAT_RGT.ALTER_REFRESH_TEMPLATE (
   refresh_template_name      IN   VARCHAR2,
   new_owner                  IN   VARCHAR2 := '-',
   new_refresh_group_name     IN   VARCHAR2 := '-',
   new_refresh_template_name  IN   VARCHAR2 := '-',
   new_template_comment       IN   VARCHAR2 := '-',
   new_public_template        IN   VARCHAR2 := '-',
   new_last_modified          IN   DATE := to_date('1', 'J'),
   new_modified_by            IN   NUMBER := -1e-130);
```

パラメータ

表 56-2 ALTER_REFRESH_TEMPLATE プロシージャのパラメータ

パラメータ	説明
refresh_template_name	変更する配置テンプレートの名前。
new_owner	配置テンプレートの新しい所有者名。現行の所有者を変更しないときは値を指定しないでください。
new_refresh_group_name	必要な場合は、このパラメータを使用してテンプレート・オブジェクトが追加される新規リフレッシュ・グループの名前を指定します。現行のリフレッシュ・グループを変更しないときは値を指定しないでください。
new_refresh_template_name	新しい配置テンプレート名を指定します。現行の配置テンプレート名を変更しないときは値を指定しないでください。
new_template_comment	配置テンプレートの新しいコメント。現行のテンプレート・コメントを変更しないときは値を指定しないでください。
new_public_template	配置テンプレートがパブリックかプライベートかを決定します。指定できる値は 'Y' および 'N'（'Y' = パブリック、'N' = プライベート）のみです。現行の値を変更しないときは値を指定しないでください。
new_last_modified	この配置テンプレートの最終更新日付。値を指定しないと、現行の日付が自動的に使用されます。
new_modified_by	この配置テンプレートの最終更新ユーザーの名前。値を指定しないと、カレント・ユーザーが自動的に使用されます。

例外

表 56-3 ALTER_REFRESH_TEMPLATE プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
bad_public_template	public_template パラメータの指定に誤りがあります。public_template パラメータには、'Y'（パブリック・テンプレート）または 'N'（プライベート・テンプレート）のいずれかを指定してください。
dupl_refresh_template	指定した名前のテンプレートは、すでに存在しています。

ALTER_TEMPLATE_OBJECT プロシージャ

このプロシージャは、指定した配置テンプレートに追加されているオブジェクトを変更します。最も一般的な変更は、オブジェクト DDL の変更および異なる配置テンプレートへのオブジェクトの割当てです。

テンプレートに対する変更内容は、配置テンプレートをインスタンス化する新規サイトにのみ反映されます。テンプレートをすでにインスタンス化しているリモート・サイトは、配置テンプレートを再インスタンス化して変更内容を適用する必要があります。

構文

```
DBMS_REPCAT_RGT.ALTER_TEMPLATE_OBJECT (
   refresh_template_name       IN   VARCHAR2,
   object_name                 IN   VARCHAR2,
   object_type                 IN   VARCHAR2,
   new_refresh_template_name   IN   VARCHAR2 := '-',
   new_object_name             IN   VARCHAR2 := '-',
   new_object_type             IN   VARCHAR2 := '-',
   new_ddl_text                IN   CLOB  := '-',
   new_master_rollback_seg     IN   VARCHAR2 := '-',
   new_flavor_id               IN   NUMBER := -1e-130);
```

パラメータ

表 56-4 ALTER_TEMPLATE_OBJECT プロシージャのパラメータ

パラメータ	説明
refresh_template_name	変更するオブジェクトを含んだ配置テンプレート名。
object_name	変更するテンプレート・オブジェクトの名前。
object_type	変更するオブジェクトのタイプ。
new_refresh_template_name	このオブジェクトを再割当てする新しい配置テンプレートの名前。オブジェクトを現行の配置テンプレートに割り当てたままにするときは値を指定しないでください。
new_object_name	テンプレート・オブジェクトの新しい名前。現行のオブジェクト名を変更しないときは値を指定しないでください。
new_object_type	指定済みの場合は、新しいオブジェクト・タイプを指定します。次のタイプのオブジェクトを指定できます。 SNAPSHOT　PROCEDURE INDEX　FUNCTION TABLE　PACKAGE VIEW　PACKAGE BODY SYNONYM　TRIGGER SEQUENCE　DATABASE LINK
new_ddl_text	指定したオブジェクトに対する新しいオブジェクト DDL。現行のオブジェクト DDL を変更しないときは値を指定しないでください。
new_master_rollback_seg	指定したオブジェクトの新しいマスター・ロールバック・セグメント。現行のロールバック・セグメントを変更しないときは値を指定しないでください。
new_flavor_id	内部使用のためのパラメータ。 **注意：**オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

例外

表 56-5 ALTER_TEMPLATE_OBJECT プロシージャの例外

例外	説明
miss_refresh_ template	指定した配置テンプレート名が無効か、または存在しません。
miss_flavor_id	この例外が発生した場合は、オラクル社カスタマ・サポート・センターに連絡してください。
bad_object_type	オブジェクト・タイプの指定に誤りがあります。有効なオブジェクト・タイプのリストは、表 56-4 を参照してください。
miss_template_object	指定したテンプレート・オブジェクト名が無効か、または存在しません。
dupl_template_object	new_refresh_template_name パラメータに指定した新規テンプレート名は、すでに存在しています。

使用上の注意

ALTER_TEMPLATE_OBJECT プロシージャは CLOB を使用しているため、このプロシージャを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、ALTER_TEMPLATE_OBJECT プロシージャで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
   tempstring VARCHAR2(100);
   templob CLOB;
BEGIN
   DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
   tempstring := 'CREATE MATERIALIZED VIEW mview_sales AS SELECT *
      FROM sales WHERE salesperson = :salesid and region_id = :region';
   DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
   DBMS_REPCAT_RGT.ALTER_TEMPLATE_OBJECT(
      refresh_template_name => 'rgt_personnel',
      object_name => 'MVIEW_SALES',
      object_type => 'SNAPSHOT',
      new_ddl_text => templob);
   DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

ALTER_TEMPLATE_PARM プロシージャ

このプロシージャでは、DBA が特定の配置テンプレートのパラメータを変更できます。パラメータ名の変更およびデフォルト値やプロンプト文字列の再定義などを実行できます。

構文

```
DBMS_REPCAT_RGT.ALTER_TEMPLATE_PARM (
   refresh_template_name       IN   VARCHAR2,
   parameter_name              IN   VARCHAR2,
   new_refresh_template_name   IN   VARCHAR2 := '-',
   new_parameter_name          IN   VARCHAR2 := '-',
   new_default_parm_value      IN   CLOB := NULL,
   new_prompt_string           IN   VARCHAR2 := '-',
   new_user_override           IN   VARCHAR2 := '-');
```

パラメータ

表 56-6 ALTER_TEMPLATE_PARM プロシージャのパラメータ

パラメータ	説明
refresh_template_name	変更するパラメータを含んだ配置テンプレートの名前。
parameter_name	変更するパラメータの名前。
new_refresh_template_name	指定したパラメータを再割当てする配置テンプレートの名前(あるテンプレートのパラメータを別のテンプレートに移動するときに便利です)。パラメータを現行のテンプレートに割り当てたままにするときは値を指定しないでください。
new_parameter_name	テンプレート・パラメータの新しい名前。現行のパラメータ名を変更しないときは値を指定しないでください。
new_default_parm_value	指定したパラメータの新しいデフォルト値。現行のデフォルト値を変更しないときは値を指定しないでください。
new_prompt_string	指定したパラメータの新しいプロンプト・テキスト。現行のプロンプト文字列を変更しないときは値を指定しないでください。
new_user_override	インスタンス化プロセス時のプロンプトで、ユーザーがデフォルト値を上書きできるかどうかを決定します。このパラメータにユーザー・パラメータ値が定義されていないと、プロンプトが表示されます。デフォルト値の上書きを許可する場合は 'Y' を、許可しない場合は 'N' を設定します。

例外

表 56-7 ALTER_TEMPLATE_PARM プロシージャの例外

例外	説明
miss_refresh_ template	指定した配置テンプレート名が無効か、または存在しません。
miss_template_parm	指定したテンプレート・パラメータが無効か、または存在しません。
dupl_template_parm	new_refresh_template_name と new_parameter_name の組合せは、すでに存在しています。

使用上の注意

ALTER_TEMPLATE_PARM プロシージャは CLOB を使用しているため、このプロシージャを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、ALTER_TEMPLATE_PARM プロシージャで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
   tempstring VARCHAR2(100);
   templob CLOB;
BEGIN
   DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
   tempstring := 'REGION 20';
   DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
   DBMS_REPCAT_RGT.ALTER_TEMPLATE_PARM(
      refresh_template_name => 'rgt_personnel',
      parameter_name => 'region',
      new_default_parm_value => templob);
   DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

ALTER_USER_AUTHORIZATION プロシージャ

このプロシージャは、DBA_REPCAT_USER_AUTHORIZATIONS ビューの内容を変更します。具体的には、ユーザー・テンプレートまたは配置テンプレートの認可割当てを変更できます。たとえば、このプロシージャは、従業員が異動し、別の配置テンプレートのマテリアライズド・ビュー環境が必要な場合に役立ちます。DBA が新規配置テンプレートを従業員に割り当てるだけで、そのユーザーはターゲット・テンプレートのインスタンス化を認可されます。

構文

```
DBMS_REPCAT_RGT.ALTER_USER_AUTHORIZATION (
   user_name                    IN   VARCHAR2,
   refresh_template_name        IN   VARCHAR2,
   new_user_name                IN   VARCHAR2 := '-',
   new_refresh_template_name    IN   VARCHAR2 := '-');
```

パラメータ

表 56-8 ALTER_USER_AUTHORIZATION プロシージャのパラメータ

パラメータ	説明
user_name	認可を変更対象とするユーザーの名前。
refresh_template_name	指定したユーザーに現在割り当てられている、変更対象の配置テンプレートの名前。
new_user_name	このパラメータは、このテンプレート認可に対して、新しいユーザーを定義するときに使用します。カレント・ユーザーを変更しないときは値を指定しないでください。
new_refresh_template_name	指定したユーザー（既存のユーザーまたは指定した新規ユーザー）がインスタンス化を認可される配置テンプレート。現行の配置テンプレートを変更しないときは値を指定しないでください。

例外

表 56-9 ALTER_USER_AUTHORIZATION プロシージャの例外

例外	説明
miss_user_ authorization	指定した user_name と refresh_template_name の組合せが DBA_REPCAT_USER_AUTHORIZATIONS ビューに存在しません。
miss_user	new_user_name または user_name パラメータに指定したユーザー名が無効か、または存在しません。
miss_refresh_template	new_refresh_template パラメータに指定した配置テンプレートが無効か、または存在しません。
dupl_user_ authorization	指定したユーザー名および配置テンプレート名に対する行は、すでに存在しています。

ALTER_USER_PARM_VALUE プロシージャ

このプロシージャは、特定のユーザーに対して定義されている既存のパラメータ値を変更します。マテリアライズド・ビュー環境で割当て表を使用している場合に特に便利なプロシージャです。ユーザー・パラメータ値を変更して、リモート・マテリアライズド・ビュー・サイトのデータ・セットを安全にすばやく変更できます。

> **関連項目：** 割当て表の使用の詳細は、『Oracle9*i* アドバンスト・レプリケーション』を参照してください。

構文

```
DBMS_REPCAT_RGT.ALTER_USER_PARM_VALUE(
   refresh_template_name         IN   VARCHAR2,
   parameter_name                IN   VARCHAR2,
   user_name                     IN   VARCHAR2,
   new_refresh_template_name     IN   VARCHAR2 := '-',
   new_parameter_name            IN   VARCHAR2 := '-',
   new_user_name                 IN   VARCHAR2 := '-',
   new_parm_value                IN   CLOB := NULL);
```

パラメータ

表 56-10 ALTER_USER_PARM_VALUE プロシージャのパラメータ

パラメータ	説明
refresh_template_name	変更するユーザー・パラメータ値を含んだ配置テンプレートの名前。
parameter_name	変更するパラメータの名前。
user_name	パラメータ値を変更対象とするユーザーの名前。
new_refresh_template_name	指定したユーザー・パラメータ値を再割当てする配置テンプレートの名前（別のテンプレートに対してユーザーを認可するときに便利です）。パラメータを現行のテンプレートに割り当てたままにするときは値を指定しないでください。
new_parameter_name	新規テンプレート・パラメータ名。既存のパラメータに対して定義されたユーザー値を変更しないときは値を指定しないでください。
new_user_name	このパラメータ値を適用する新規ユーザー名。パラメータをカレント・ユーザーに割り当てたままにするときは値を指定しないでください。
new_parm_value	指定したユーザー・パラメータの新しいパラメータ値。現行のパラメータ値を変更しないときは値を指定しないでください。

例外

表 56-11 ALTER_USER_PARM_VALUE プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_template_parm	指定したテンプレート・パラメータが無効か、または存在しません。
miss_user	user_name または new_user_name パラメータに指定したユーザー名が無効か、または存在しません。
miss_user_parm_values	指定したユーザー・パラメータ値が存在しません。
dupl_user_parm_values	指定した新規ユーザー・パラメータはすでに存在しています。

使用上の注意

ALTER_USER_PARM_VALUE プロシージャは CLOB を使用しているため、このプロシージャを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、ALTER_USER_PARM_VALUE プロシージャで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
   tempstring VARCHAR2(100);
   templob CLOB;
BEGIN
   DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
   tempstring := 'REGION 20';
   DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
   DBMS_REPCAT_RGT.ALTER_USER_PARM_VALUE(
      refresh_template_name => 'rgt_personnel',
      parameter_name => 'region',
      user_name => 'BOB',
      new_parm_value => templob);
   DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

COMPARE_TEMPLATES ファンクション

このファンクションでは、DBA が 2 つの配置テンプレートの内容を比較できます。2 つの配置テンプレートの相違点は、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納されます。

COMPARE_TEMPLATES ファンクションは数値を戻します。この数値は、USER_REPCAT_TEMP_OUTPUT 一時ビューの問合せ時に WHERE 句に指定する値です。たとえば、COMPARE_TEMPLATES プロシージャが 10 を戻した場合に、指定した 2 つのテンプレート間の相違点をすべて表示するには、次の SELECT 文を実行します（テンプレートが同じ場合、SELECT 文は行を戻しません）。

```
SELECT TEXT FROM USER_REPCAT_TEMP_OUTPUT
   WHERE OUTPUT_ID =  10 ORDER BY LINE;
```

USER_REPCAT_TEMP_OUTPUT 一時ビューの内容は、ユーザーが切断するか、またはロールバックが実行されると失われます。

構文

```
DBMS_REPCAT_RGT.COMPARE_TEMPLATES (
   source_template_name    IN   VARCHAR2,
   compare_template_name   IN   VARCHAR2)
```

```
  return NUMBER;
```

パラメータ

表 56-12 COMPARE_TEMPLATES ファンクションのパラメータ

パラメータ	説明
source_template_name	比較対象の最初の配置テンプレートの名前。
compare_template_name	比較対象の 2 番目の配置テンプレートの名前。

例外

表 56-13 COMPARE_TEMPLATES ファンクションの例外

例外	説明
miss_refresh_template	比較対象の配置テンプレート名が無効か、または存在しません。

戻り値

表 56-14 COMPARE_TEMPLATES ファンクションの戻り値

戻り値	説明
< システム生成番号 >	比較したテンプレート間の相違点を USER_REPCAT_TEMP_OUTPUT 一時ビューで表示するように選択したときに、output_id の値に戻される番号を指定します。

COPY_TEMPLATE ファンクション

このファンクションを使用すると、配置テンプレートをコピーできるため、既存の配置テンプレートに含まれるオブジェクトを新しい配置テンプレートで多数使用するときに便利です。さらに、このファンクションは、配置テンプレート、テンプレート・オブジェクト、テンプレート・パラメータおよびユーザー・パラメータ値をコピーします。DBA は、オプションでこのテンプレートに対するユーザー認証をコピーできます。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

注意： DBA_REPCAT_TEMPLATE_SITES ビュー内の値はコピーされません。

このファンクションでは、配置テンプレートを別のマスター・サイトにコピーすることもできます。この機能は、配置テンプレートの配布、および複数サイト間でのネットワーク負荷の分散に役立ちます。

構文

```
DBMS_REPCAT_RGT.COPY_TEMPLATE (
   old_refresh_template_name     IN   VARCHAR2,
   new_refresh_template_name     IN   VARCHAR2,
   copy_user_authorizations      IN   VARCHAR2,
   dblink                        IN   VARCHAR2 := NULL)
  return NUMBER;
```

パラメータ

表 56-15 COPY_TEMPLATE ファンクションのパラメータ

パラメータ	説明
old_refresh_template_name	コピー元の配置テンプレートの名前。
new_refresh_template_name	新規配置テンプレートの名前。
copy_user_authorizations	コピー元のテンプレートのテンプレート認可を新しい配置テンプレート用にコピーするかどうかを指定します。有効な値は、Y、N および NULL です。 **注意**：すべてのユーザーがターゲット・データベースに存在している必要があります。
dblink	オプションで、配置テンプレートのコピー元を定義します（この機能は、他のマスター・サイトに配置テンプレートを配布するときに便利です）。指定しないと、配置テンプレートはローカル・マスター・サイトからコピーされます。

例外

表 56-16 COPY_TEMPLATE ファンクションの例外

例外	説明
miss_refresh_template	コピー元の配置テンプレート名が無効か、または存在しません。
dupl_refresh_template	指定した新規リフレッシュ・テンプレートの名前は、すでに存在しています。
bad_copy_auth	copy_user_authorization パラメータに指定した値が無効です。有効な値は、Y、N および NULL です。

戻り値

表 56-17 COPY_TEMPLATE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号。

CREATE_OBJECT_FROM_EXISTING ファンクション

このファンクションは、既存のデータベース・オブジェクトからテンプレート・オブジェクト定義を作成し、それをターゲット配置テンプレートに追加します。ターゲット配置テンプレートがリモート・マテリアライズド・ビュー・サイトでインスタンス化されるときに、元のデータベース・オブジェクトを作成したオブジェクト DDL が実行されます。これは、既存のトリガーおよびプロシージャをテンプレートに追加するときに便利な方法です。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_OBJECT_FROM_EXISTING(
     refresh_template_name  IN   VARCHAR2,
     object_name            IN   VARCHAR2,
     sname                  IN   VARCHAR2,
     oname                  IN   VARCHAR2,
     otype                  IN   VARCHAR2)
     return NUMBER;
```

パラメータ

表 56-18 CREATE_OBJECT_FROM_EXISTING ファンクションのパラメータ

パラメータ	説明
refresh_template_name	このオブジェクトを追加する配置テンプレートの名前。
object_name	オプションで、配置テンプレートに追加する既存オブジェクトの新規名を指定します（既存のオブジェクトに新しい名前を定義できます）。
sname	テンプレート・オブジェクトの作成元のオブジェクトを含んだスキーマ。
oname	テンプレート・オブジェクトの作成元のオブジェクト名。
otype	テンプレートに追加するデータベース・オブジェクトのタイプ（PROCEDURE、TRIGGER など）。オブジェクト・タイプは、次の数値タイプ識別子を使用して指定する必要があります（DATABASE LINK、MATERIALIZED VIEW および SNAPSHOT は、このファンクションでは有効なオブジェクト・タイプではありません）。 SEQUENCE　PROCEDURE INDEX　FUNCTION TABLE　PACKAGE VIEW　PACKAGE BODY SYNONYM　TRIGGER

例外

表 56-19 CREATE_OBJECT_FROM_EXISTING ファンクションの例外

例外	説明
miss_refresh_template	指定したリフレッシュ・テンプレート名が無効か、または存在しません。既存の配置テンプレートのリストは、DBA_REPCAT_REFRESH_TEMPLATES ビューを問い合せてください。
bad_object_type	オブジェクト・タイプの指定に誤りがあります。
dupl_template_object	同じ名前とタイプのオブジェクトが、指定した配置テンプレートにすでに追加されています。
objectmissing	指定したオブジェクトが存在しません。

戻り値

表 56-20 CREATE_OBJECT_FROM_EXISTING ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号。

CREATE_REFRESH_TEMPLATE ファンクション

このファンクションは、配置テンプレートを作成します。テンプレート名、プライベートまたはパブリック・ステータス、およびターゲット・リフレッシュ・グループを定義できます。テンプレート・オブジェクト、ユーザー認証またはテンプレート・パラメータを作成するたびに、このファンクションで作成された配置テンプレートを参照します。このファンクションは、DBA_REPCAT_REFRESH_TEMPLATES ビューに行を追加します。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_REFRESH_TEMPLATE (
   owner                  IN   VARCHAR2,
   refresh_group_name     IN   VARCHAR2,
   refresh_template_name  IN   VARCHAR2,
   template_comment       IN   VARCHAR2 := NULL,
   public_template        IN   VARCHAR2 := NULL,
   last_modified          IN   DATE := SYSDATE,
   modified_by            IN   VARCHAR2 := USER,
   creation_date          IN   DATE := SYSDATE,
   created_by             IN   VARCHAR2 := USER)
  return NUMBER;
```

パラメータ

表 56-21 CREATE_REFRESH_TEMPLATE ファンクションのパラメータ

パラメータ	説明
owner	配置テンプレート所有者のユーザー名。指定しない場合、テンプレートの作成ユーザーが自動的に使用されます。
refresh_group_name	このテンプレートのインスタンス化時に作成されるリフレッシュ・グループの名前。このテンプレートで作成されたオブジェクトはすべて、指定したリフレッシュ・グループに割り当てられます。
refresh_template_name	作成する配置テンプレートの名前。この名前は、この配置テンプレートを含むすべてのアクティビティで参照されます。
template_comment	このパラメータで定義したユーザー・コメントは、DBA_REPCAT_REFRESH_TEMPLATES ビューに表示されます。
public_template	配置テンプレートがパブリックかプライベートかを指定します。指定できる値は 'Y' および 'N'（'Y' = パブリックおよび 'N' = プライベート）のみです。
last_modified	この配置テンプレートの最終更新日付。値を指定しないと、現行の日付が自動的に使用されます。
modified_by	この配置テンプレートの最終更新ユーザーの名前。値を指定しないと、カレント・ユーザーが自動的に使用されます。
creation_date	この配置テンプレートの作成日付。値を指定しないと、現行の日付が自動的に使用されます。
created_by	この配置テンプレートの作成ユーザーの名前。値を指定しないと、カレント・ユーザーが自動的に使用されます。

例外

表 56-22 CREATE_REFRESH_TEMPLATE ファンクションの例外

例外	説明
dupl_refresh_template	指定した名前のテンプレートは、すでに存在しています。
bad_public_template	public_template パラメータの指定に誤りがあります。public_template パラメータには、'Y'（パブリック・テンプレート）または 'N'（プライベート・テンプレート）のいずれかを指定してください。

戻り値

表 56-23 CREATE_REFRESH_TEMPLATE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号。

CREATE_TEMPLATE_OBJECT ファンクション

このファンクションは、ターゲット配置テンプレートのコンテナにオブジェクト定義を追加します。指定したオブジェクト DDL は、ターゲット配置テンプレートがリモート・マテリアライズド・ビュー・サイトでインスタンス化されるときに実行されます。マテリアライズド・ビュー以外に、表、プロシージャおよびその他のオブジェクトをテンプレートに追加できます。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_TEMPLATE_OBJECT (
   refresh_template_name  IN   VARCHAR2,
   object_name            IN   VARCHAR2,
   object_type            IN   VARCHAR2,
   ddl_text               IN   CLOB,
   master_rollback_seg    IN   VARCHAR2 := NULL,
   flavor_id              IN   NUMBER := -1e-130)
  return NUMBER;
```

パラメータ

表 56-24 CREATE_TEMPLATE_OBJECT ファンクションのパラメータ

パラメータ	説明
refresh_template_name	このオブジェクトを追加する配置テンプレートの名前。
object_name	作成するテンプレート・オブジェクトの名前。
object_type	テンプレートに追加するデータベース・オブジェクトのタイプ（SNAPSHOT、TRIGGER、PROCEDURE など）。次のタイプのオブジェクトを指定できます。 SNAPSHOT　PROCEDURE INDEX　FUNCTION TABLE　PACKAGE VIEW　PACKAGE BODY SYNONYM　TRIGGER SEQUENCE　DATABASE LINK
ddl_text	テンプレートに追加するオブジェクトを作成する DDL。DDL は必ずセミコロンで終了してください。テンプレート・オブジェクトのテンプレート・パラメータの作成にはコロン（:）を使用できます。 CREATE MATERIALIZED VIEW 文でマテリアライズド・ビュー（スナップショット）を追加するときは、必ずマテリアライズド・ビューの問合せでマスター表所有者のスキーマ名を指定してください。
master_rollback_seg	定義済みのオブジェクト DDL をリモート・マテリアライズド・ビュー・サイトで実行するときに使用するロールバック・セグメントの名前。
flavor_id	内部使用のためのパラメータ。 **注意：**オラクル社カスタマ・サポート・センターから指示がないかぎり、このパラメータは設定しないでください。

例外

表 56-25 CREATE_TEMPLATE_OBJECT ファンクションの例外

例外	説明
miss_refresh_template	指定したリフレッシュ・テンプレート名が無効か、または存在しません。既存の配置テンプレートのリストは、DBA_REPCAT_REFRESH_TEMPLATES ビューを問い合せてください。
bad_object_type	オブジェクト・タイプの指定に誤りがあります。有効なオブジェクト・タイプのリストは、表 56-24 を参照してください。
dupl_template_object	同じ名前とタイプのオブジェクトが、指定した配置テンプレートにすでに追加されています。

戻り値

表 56-26 CREATE_TEMPLATE_OBJECT ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号。

使用上の注意

CREATE_TEMPLATE_OBJECT は CLOB を使用しているため、このファンクションを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、CREATE_TEMPLATE_OBJECT ファンクションで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
   tempstring VARCHAR2(100);
   templob CLOB;
   a NUMBER;
BEGIN
   DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
   tempstring := 'CREATE MATERIALIZED VIEW mview_sales AS SELECT *
        FROM sales WHERE salesperson = :salesid';
   DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
   a := DBMS_REPCAT_RGT.CREATE_TEMPLATE_OBJECT(
        refresh_template_name => 'rgt_personnel',
        object_name => 'mview_sales',
        object_type => 'SNAPSHOT',
        ddl_text => templob,
        master_rollback_seg => 'RBS');
   DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

CREATE_TEMPLATE_PARM ファンクション

このファンクションは、特定の配置テンプレートのパラメータを作成します。この結果、リモート・マテリアライズド・ビュー・サイトでカスタム・データ・セットを作成できます。このファンクションは、DBA がテンプレート・オブジェクトの追加前に一連のテンプレート変数を定義するときにのみ必要です。CREATE_TEMPLATE_OBJECT ファンクションを使用して、オブジェクトがテンプレートに追加されると、オブジェクト DDL 内の変数は DBA_REPCAT_TEMPLATE_PARMS ビューに自動的に追加されます。

DBA は通常、ALTER_TEMPLATE_PARM プロシージャを使用して、デフォルトのパラメータ値やプロンプト文字列を変更します（詳細は、56-9 ページの「ALTER_TEMPLATE_PARM プロシージャ」を参照してください）。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_TEMPLATE_PARM (
   refresh_template_name  IN   VARCHAR2,
   parameter_name         IN   VARCHAR2,
   default_parm_value     IN   CLOB := NULL,
   prompt_string          IN   VARCHAR2 := NULL,
   user_override          IN   VARCHAR2 := NULL)
   return NUMBER;
```

パラメータ

表 56-27 CREATE_TEMPLATE_PARM ファンクションのパラメータ

パラメータ	説明
refresh_template_name	パラメータを作成する配置テンプレートの名前。
parameter_name	作成するパラメータの名前。
default_parm_value	作成するパラメータのデフォルト値。ユーザー・パラメータ値またはランタイム・パラメータ値が存在しない場合は、インスタンス化プロセス時にこのデフォルト値が使用されます。
prompt_string	インスタンス化プロセス時にこのテンプレート・パラメータに対して表示される説明的なプロンプト・テキスト。
user_override	インスタンス化プロセス時のプロンプトで、ユーザーがデフォルト値を上書きできるかどうかを決定します。このパラメータにユーザー・パラメータ値が定義されていないと、プロンプトが表示されます。デフォルト値の上書きを許可する場合は 'Y' を、許可しない場合は 'N' を設定します。

例外

表 56-28 CREATE_TEMPLATE_PARM ファンクションの例外

例外	説明
miss_refresh_template	指定したリフレッシュ・テンプレート名が無効か、または存在しません。
dupl_template_parm	同じ名前のパラメータが、指定した配置テンプレートにすでに定義されています。

戻り値

表 56-29 CREATE_TEMPLATE_PARM ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号。

使用上の注意

CREATE_TEMPLATE_PARM ファンクションは CLOB を使用しているため、このファンクションを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、CREATE_TEMPLATE_PARM ファンクションで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
   tempstring VARCHAR2(100);
   templob CLOB;
   a NUMBER;
BEGIN
   DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
   tempstring := 'REGION 20';
   DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
   a := DBMS_REPCAT_RGT.CREATE_TEMPLATE_PARM(
        refresh_template_name => 'rgt_personnel',
        parameter_name => 'region',
        default_parm_value => templob,
        prompt_string => 'Enter your region ID:',
        user_override => 'Y');
   DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

CREATE_USER_AUTHORIZATION ファンクション

このファンクションは、特定のユーザーに、プライベート配置テンプレートのインスタンス化を認可します。プライベート配置テンプレートに対する認可を付与されていないユーザーは、プライベート・テンプレートをインスタンス化できません。このファンクションは、DBA_REPCAT_USER_AUTHORIZATIONS ビューに行を追加します。

ユーザーを許可する前に、配置テンプレートをインスタンス化するマスター・サイトに、そのユーザーが存在していることを検証してください。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_USER_AUTHORIZATION (
   user_name               IN   VARCHAR2,
   refresh_template_name   IN   VARCHAR2)
   return NUMBER;
```

パラメータ

表 56-30 CREATE_USER_AUTHORIZATION ファンクションのパラメータ

パラメータ	説明
user_name	指定したテンプレートのインスタンス化を許可するユーザーの名前。複数のユーザーを指定するときは、ユーザー名をカンマで区切ります（例：'john, mike, bob'）。
refresh_template_name	指定したユーザーにインスタンス化を許可するテンプレートの名前。

例外

表 56-31 CREATE_USER_AUTHORIZATION ファンクションの例外

例外	説明
miss_user	指定したユーザー名が無効か、または存在しません。
miss_refresh_template	指定したリフレッシュ・テンプレート名が無効か、または存在しません。
dupl_user_authorization	指定したユーザーと配置テンプレートに対する認可はすでに作成されています。

戻り値

表 56-32 CREATE_USER_AUTHORIZATION ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号。

CREATE_USER_PARM_VALUE ファンクション

このファンクションは、特定のユーザーに対する配置テンプレートのパラメータ値を事前定義します。たとえば、ユーザー 33456 のリージョン・パラメータを west に事前定義する場合などに、このファンクションを使用します。

このファンクションで指定した値は、テンプレート・パラメータに対して指定したデフォルト値よりも優先されます。このファンクションで戻される番号は、配置テンプレートを管理するために Oracle で内部的に使用されます。

構文

```
DBMS_REPCAT_RGT.CREATE_USER_PARM_VALUE (
   refresh_template_name    IN   VARCHAR2,
   parameter_name           IN   VARCHAR2,
   user_name                IN   VARCHAR2,
   parm_value               IN   CLOB := NULL)
  return NUMBER;
```

パラメータ

表 56-33 CREATE_USER_PARM_VALUE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	ユーザー・パラメータ値を作成するパラメータを含んだ配置テンプレートの名前。
parameter_name	ユーザー・パラメータ値を定義するテンプレート・パラメータの名前。
user_name	ユーザー・パラメータ値を事前定義するユーザーの名前。
parm_value	事前定義パラメータ値。指定したユーザーが開始したインスタンス化プロセス時に使用されます。

例外

表 56-34 CREATE_USER_PARM_VALUE ファンクションの例外

例外	説明
miss_refresh_ template	指定した配置テンプレート名が無効か、または存在しません。
dupl_user_parm_ values	指定したユーザー、パラメータおよび配置テンプレートに対するパラメータ値はすでに定義されています。既存のユーザー・パラメータ値のリストは、DBA_REPCAT_USER_PARM_VALUES ビューを問い合せてください。
miss_template_parm	指定した配置テンプレート・パラメータ名が無効か、または存在しません。
miss_user	指定したユーザー名が無効か、または存在しません。

戻り値

表 56-35 CREATE_USER_PARM_VALUE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	Oracle で内部的に使用されるシステム生成番号。

使用上の注意

CREATE_USER_PARM_VALUE ファンクションは CLOB を使用しているため、このファンクションを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、CREATE_USER_PARM_VALUE ファンクションで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
   tempstring VARCHAR2(100);
   templob CLOB;
   a NUMBER;
BEGIN
   DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
   tempstring := 'REGION 20';
   DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
   a := DBMS_REPCAT_RGT.CREATE_USER_PARM_VALUE(
        refresh_template_name => 'rgt_personnel',
        parameter_name => 'region',
        user_name => 'BOB',
        user_parm_value => templob);
   DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

DELETE_RUNTIME_PARMS プロシージャ

このプロシージャは、配置テンプレートをインスタンス化する前に、INSERT_RUNTIME_PARMS プロシージャを使用して定義したランタイム・パラメータ値を削除するために使用します。

構文

```
DBMS_REPCAT_RGT.DELETE_RUNTIME_PARMS(
   runtime_parm_id     IN   NUMBER,
   parameter_name      IN   VARCHAR2);
```

パラメータ

表 56-36 DELETE_RUNTIME_PARMS プロシージャのパラメータ

パラメータ	説明
runtime_parm_id	以前にランタイム・パラメータ値に割り当てた ID（GET_RUNTIME_PARM_ID ファンクションを使用して取り出された値です）を指定します。
parameter_name	削除するパラメータ値の名前（配置テンプレート・パラメータのリストは、DBA_REPCAT_TEMPLATE_PARMS ビューを問い合せてください）を指定します。

例外

表 56-37 DELETE_RUNTIME_PARMS プロシージャの例外

例外	説明
miss_template_parm	指定した配置テンプレート・パラメータ名が無効か、または存在しません。

DROP_ALL_OBJECTS プロシージャ

このプロシージャでは、DBA が配置テンプレートからすべてのオブジェクトまたは特定のオブジェクト・タイプを削除できます。

注意： このプロシージャは、元に戻すことのできない危険度の高いプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_OBJECTS (
   refresh_template_name   IN   VARCHAR2,
   object_type             IN   VARCHAR2 := NULL);
```

パラメータ

表 56-38 DROP_ALL_OBJECTS プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するオブジェクトを含んだ配置テンプレートの名前。
object_type	NULL を指定すると、テンプレート内のすべてのオブジェクトが削除されます。オブジェクト・タイプを指定すると、そのタイプのオブジェクトのみ削除されます。次のタイプのオブジェクトを指定できます。 SNAPSHOT　PROCEDURE INDEX　FUNCTION TABLE　PACKAGE VIEW　PACKAGE BODY SYNONYM　TRIGGER SEQUENCE　DATABASE LINK

例外

表 56-39 DROP_ALL_OBJECTS プロシージャの例外

例外	説明
miss_refresh_ template	指定した配置テンプレート名が無効か、または存在しません。
bad_object_type	オブジェクト・タイプの指定に誤りがあります。有効なオブジェクト・タイプのリストは、表 56-38 を参照してください。

DROP_ALL_TEMPLATE_PARMS プロシージャ

このプロシージャによって、指定した配置テンプレートのテンプレート・パラメータを削除できます。テンプレート・オブジェクトが参照していないすべてのパラメータ、またはパラメータを参照しているすべてのオブジェクトをそのパラメータ自体とともにすべてテンプレートから削除できます。

> **注意：** このプロシージャは、元に戻すことのできない危険度の高いプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_TEMPLATE_PARMS (
   refresh_template_name   IN   VARCHAR2,
   drop_objects            IN   VARCHAR2 := 'n');
```

パラメータ

表 56-40 DROP_ALL_TEMPLATE_PARMS プロシージャのパラメータ

パラメータ	説明
refresh_template_ name	削除するパラメータおよびオブジェクトを含んだ配置テンプレートの名前。
drop_objects	値を指定しないと、このパラメータはデフォルトで 'N' に設定され、テンプレート・オブジェクトが参照していないパラメータはすべて削除されます。 'Y' を指定すると、テンプレート・パラメータを参照しているすべてのオブジェクトおよびそのテンプレート・パラメータ自体が削除されます。オブジェクトは、データベースからではなく、テンプレートから削除されます。

例外

表 56-41 DROP_ALL_TEMPLATE_PARMS プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。

DROP_ALL_TEMPLATE_SITES プロシージャ

このプロシージャは、DBA_REPCAT_TEMPLATE_SITES ビューからエントリをすべて削除します。このビューには、特定の配置テンプレートをインスタンス化したサイトの記録が格納されています。

> **注意：** このプロシージャは、元に戻すことのできない危険度の高いプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_TEMPLATE_SITES (
   refresh_template_name    IN    VARCHAR2);
```

パラメータ

表 56-42 DROP_ALL_TEMPLATE_SITES プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するサイトを含んだ配置テンプレートの名前。

例外

表 56-43 DROP_ALL_TEMPLATE_SITES プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。

DROP_ALL_TEMPLATES プロシージャ

このプロシージャは、プロシージャがコールされるサイトの配置テンプレートをすべて削除します。

注意： このプロシージャは、元に戻すことのできない危険度の高いプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_TEMPLATES;
```

DROP_ALL_USER_AUTHORIZATIONS プロシージャ

このプロシージャでは、DBA が、指定した配置テンプレートに対するユーザー認証をすべて削除できます。このプロシージャを実行すると、DBA_REPCAT_USER_AUTHORIZATIONS ビューから行が削除されます。

このプロシージャは、プライベート・テンプレートがパブリック・テンプレートに変換され、ユーザー認証が不要になってから実装されることがあります。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_USER_AUTHORIZATIONS (
   refresh_template_name   IN   VARCHAR2);
```

パラメータ

表 56-44 DROP_ALL_USER_AUTHORIZATIONS プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するユーザー認証を含んだ配置テンプレートの名前。

例外

表 56-45 DROP_ALL_USER_AUTHORIZATIONS プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。

DROP_ALL_USER_PARM_VALUES プロシージャ

このプロシージャは、特定の配置テンプレートに対するユーザー・パラメータ値を削除します。このプロシージャは柔軟に設計されており、削除するユーザー・パラメータ値のセットを定義できます。たとえば、パラメータの指定方法によって、次の表 56-46 のように処理されます。

表 56-46 DROP_ALL_USER_PARM_VALUES プロシージャ

パラメータ	影響
refresh_template_name	指定した配置テンプレートに対するユーザー・パラメータをすべて削除します。
refresh_template_name および user_name	指定した配置テンプレートに対する指定したユーザー・パラメータをすべて削除します。
refresh_template_name および parameter_name	指定した配置テンプレート・パラメータに対するユーザー・パラメータ値をすべて削除します。
refresh_template_name、parameter_name および user_name	指定した配置テンプレート・パラメータに対する指定したユーザー値を削除します（drop_user_parm と同じです）。

構文

```
DBMS_REPCAT_RGT.DROP_ALL_USER_PARMS_VALUES (
   refresh_template_name    IN   VARCHAR2,
   user_name                IN   VARCHAR2,
   parameter_name           IN   VARCHAR2);
```

パラメータ

表 56-47 DROP_ALL_USER_PARMS プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するパラメータ値を含んだ配置テンプレートの名前。
user_name	パラメータ値を削除対象とするユーザーの名前。
parameter_name	削除する値を含んだテンプレート・パラメータ。

例外

表 56-48 DROP_ALL_USER_PARMS プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定したユーザー名が無効か、または存在しません。
miss_user_parm_values	配置テンプレート、ユーザーおよびパラメータの組合せが DBA_REPCAT_USER_PARM_VALUES ビューに存在しません。

DROP_REFRESH_TEMPLATE プロシージャ

このプロシージャは配置テンプレートを削除します。配置テンプレートを削除すると、関連するすべてのテンプレート・パラメータ、ユーザー認証、テンプレート・オブジェクトおよびユーザー・パラメータも段階的に削除されます（このプロシージャでは、テンプレート・サイトは削除されません）。

構文

```
DBMS_REPCAT_RGT.DROP_REFRESH_TEMPLATE (
   refresh_template_name  IN   VARCHAR2);
```

パラメータ

表 56-49 DROP_REFRESH_TEMPLATE プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除する配置テンプレートの名前。

例外

表 56-50 DROP_REFRESH_TEMPLATE プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。配置テンプレートのリストは、DBA_REPCAT_REFRESH_TEMPLATES ビューを問い合せてください。

DROP_SITE_INSTANTIATION プロシージャ

このプロシージャは、ターゲット・サイトのテンプレート・インスタンス化を削除します。また、マスター・サイトの関連メタデータをすべて削除し、指定したサイトにおけるマテリアライズド・ビューのリフレッシュを使用禁止にします。

構文

```
DBMS_REPCAT_RGT.DROP_SITE_INSTANTIATION (
     refresh_template_name  IN   VARCHAR2,
     user_name              IN   VARCHAR2,
     site_name              IN   VARCHAR2);
```

表 56-51 DROP_SITE_INSTANTIATION プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除する配置テンプレートの名前。
user_name	リモート・マテリアライズド・ビュー・サイトでテンプレートを最初にインスタンス化したユーザーの名前。テンプレートをインスタンス化したユーザーを調べるには、ALL_REPCAT_TEMPLATE_SITES ビューを問い合せてください。
site_name	指定したテンプレート・インスタンス化を削除するマスター・サイトを識別します。

例外

表 56-52 DROP_SITE_INSTANTIATION プロシージャの例外

例外	説明
miss_refresh_ template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定したユーザー名が存在しません。
miss_template_site	ユーザーおよびサイトに対する配置テンプレートがインスタンス化されていません。

DROP_TEMPLATE_OBJECT プロシージャ

このプロシージャは、特定の配置テンプレートからテンプレート・オブジェクトを削除します。たとえば、DBA は、古いマテリアライズド・ビューを配置テンプレートから削除するためにこのプロシージャを使用できます。テンプレートの変更内容は、変更後に配置テンプレートをインスタンス化する新規サイトに反映されます。テンプレートをすでにインスタンス化しているリモート・サイトは、配置テンプレートを再インスタンス化して変更内容を適用する必要があります。

構文

```
DBMS_REPCAT_RGT.DROP_TEMPLATE_OBJECT (
   refresh_template_name  IN   VARCHAR2,
   object_name            IN   VARCHAR2,
   object_type            IN   VARCHAR2);
```

パラメータ

表 56-53 DROP_TEMPLATE_OBJECT プロシージャのパラメータ

パラメータ	説明
refresh_template_name	オブジェクトを削除する配置テンプレートの名前。
object_name	削除するテンプレート・オブジェクトの名前。
object_type	削除するオブジェクトのタイプ。次のタイプのオブジェクトを指定できます。 SNAPSHOT　　PROCEDURE INDEX　　FUNCTION TABLE　　PACKAGE VIEW　　PACKAGE BODY SYNONYM　　TRIGGER SEQUENCE　　DATABASE LINK

例外

表 56-54 DROP_TEMPLATE_OBJECT プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_template_object	指定したテンプレート・オブジェクトが無効か、または存在しません。配置テンプレート・オブジェクトのリストは、DBA_REPCAT_TEMPLATE_OBJECTS ビューを問い合せてください。

DROP_TEMPLATE_PARM プロシージャ

このプロシージャは、DBA_REPCAT_TEMPLATE_PARMS ビューから既存のテンプレート・パラメータを削除します。あるテンプレート・オブジェクトを削除して、特定のパラメータが不要になった場合に有効なプロシージャです。

構文

```
DBMS_REPCAT_RGT.DROP_TEMPLATE_PARM (
   refresh_template_name  IN   VARCHAR2,
   parameter_name         IN   VARCHAR2);
```

パラメータ

表 56-55 DROP_TEMPLATE_PARM プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するパラメータを含んだ配置テンプレート名。
parameter_name	削除するパラメータの名前。

例外

表 56-56 DROP_TEMPLATE_PARM プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_template_parm	指定したパラメータ名が無効か、または存在しません。テンプレート・パラメータのリストは、DBA_REPCAT_TEMPLATE_PARMS ビューを問い合せてください。

DROP_USER_AUTHORIZATION プロシージャ

このプロシージャは、DBA_REPCAT_USER_AUTHORIZATIONS ビューからユーザー認証エントリを削除します。ユーザーのテンプレート認可を削除するときに使用します。ユーザーの認証が削除されると、そのユーザーはターゲット配置テンプレートをインスタンス化できません。

関連項目： 56-36 ページ「DROP_ALL_USER_AUTHORIZATIONS プロシージャ」

構文

```
DBMS_REPCAT_RGT.DROP_USER_AUTHORIZATION (
   refresh_template_name   IN   VARCHAR2,
   user_name               IN   VARCHAR2);
```

パラメータ

表 56-57 DROP_USER_AUTHORIZATION プロシージャのパラメータ

パラメータ	説明
refresh_template_name	ユーザーの認証を削除する配置テンプレートの名前。
user_name	認証を削除対象とするユーザーの名前。

例外

表 56-58 DROP_USER_AUTHORIZATION プロシージャの例外

例外	説明
miss_user	指定したユーザー名が無効か、または存在しません。
miss_user_authorization	指定したユーザーおよび配置テンプレートの組合せが存在しません。ユーザーまたは配置テンプレートのリストは、DBA_REPCAT_USER_AUTHORIZATIONS を問い合せてください。
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。

DROP_USER_PARM_VALUE プロシージャ

このプロシージャは、特定の配置テンプレートに対する事前定義のユーザー・パラメータ値を削除します。通常、ユーザーのテンプレート認可を削除した後に実行します。

構文

```
DBMS_REPCAT_RGT.DROP_USER_PARM_VALUE (
   refresh_template_name    IN   VARCHAR2,
   parameter_name           IN   VARCHAR2,
   user_name                IN   VARCHAR2);
```

パラメータ

表 56-59 DROP_USER_PARM_VALUE プロシージャのパラメータ

パラメータ	説明
refresh_template_name	削除するパラメータ値を含んだ配置テンプレート名。
parameter_name	削除する事前定義値を含んだテンプレートの名前。
user_name	パラメータ値を削除対象とするユーザーの名前。

例外

表 56-60 DROP_USER_PARM_VALUE プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定したユーザー名が無効か、または存在しません。
miss_user_parm_values	配置テンプレート、ユーザーおよびパラメータの組合せが DBA_REPCAT_USER_PARM_VALUES ビューに存在しません。

GET_RUNTIME_PARM_ID ファンクション

このファンクションは、ランタイム・パラメータ値の定義時に使用する ID を取り出します。ランタイム・パラメータ値はすべてこの ID に割り当てられ、インスタンス化プロセス時にも使用されます。

構文

```
DBMS_REPCAT_RGT.GET_RUNTIME_PARM_ID
  RETURN NUMBER;
```

戻り値

表 56-61 GET_RUNTIME_PARM_ID ファンクションの戻り値

戻り値	対応するデータ・タイプ
< システム生成番号 >	ランタイム・パラメータ値は、このシステム生成番号に割り当てられ、インスタンス化プロセス時にも使用されます。

INSERT_RUNTIME_PARMS プロシージャ

このプロシージャは、テンプレートのインスタンス化前にランタイム・パラメータ値を定義します。ユーザー・パラメータ値が未定義で、デフォルトのパラメータ値を使用しない場合は、このプロシージャを使用してパラメータ値を定義する必要があります。

このプロシージャを使用する前に、必ず GET_RUNTIME_PARM_ID ファンクションを実行して、ランタイム・パラメータの挿入時に使用するパラメータ ID を取り出してください。この ID は、ランタイム・パラメータ値の定義および配置テンプレートのインスタンス化に使用されます。

構文

```
DBMS_REPCAT_RGT.INSERT_RUNTIME_PARMS (
   runtime_parm_id    IN   NUMBER,
   parameter_name     IN   VARCHAR2,
   parameter_value    IN   CLOB);
```

パラメータ

表 56-62 INSERT_RUNTIME_PARMS プロシージャのパラメータ

パラメータ	説明
runtime_parm_id	GET_RUNTIME_PARM_ID ファンクションで取り出された ID。この ID は、配置テンプレートのインスタンス化時にも使用されます。配置テンプレートのすべてのパラメータ値には、必ず同じ ID を使用してください。
parameter_name	ランタイム・パラメータ値を定義するテンプレート・パラメータの名前。テンプレート・パラメータのリストは、DBA_REPCAT_TEMPLATE_PARMS ビューを問い合せてください。
parameter_value	配置テンプレートのインスタンス化プロセス時に使用するランタイム・パラメータ値。

例外

表 56-63 INSERT_RUNTIME_PARMS プロシージャの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定したユーザー名が無効か、または存在しません。
miss_user_parm_values	配置テンプレート、ユーザーおよびパラメータの組合せが DBA_REPCAT_USER_PARM_VALUES ビューに存在しません。

使用上の注意

INSERT_RUNTIME_PARMS は、CLOB を使用しているため、このプロシージャを使用する場合は、DBMS_LOB パッケージを使用する必要があります。次の例は、INSERT_RUNTIME_PARMS プロシージャで DBMS_LOB パッケージを使用する方法を示しています。

```
DECLARE
   tempstring VARCHAR2(100);
   templob CLOB;
BEGIN
   DBMS_LOB.CREATETEMPORARY(templob, TRUE, DBMS_LOB.SESSION);
   tempstring := 'REGION 20';
   DBMS_LOB.WRITE(templob, length(tempstring), 1, tempstring);
   DBMS_REPCAT_RGT.INSERT_RUNTIME_PARMS(
      runtime_parm_id => 20,
      parameter_name => 'region',
      parameter_value => templob);
   DBMS_LOB.FREETEMPORARY(templob);
END;
/
```

INSTANTIATE_OFFLINE ファンクション

このファンクションは、マテリアライズド・ビュー・サイトがマスターから切断されているときに、リモート・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー環境を作成するために使用するスクリプトをマスター・サイトで生成します（つまり、マテリアライズド・ビュー・サイトがオフラインのとき）。リモート・マテリアライズド・ビュー・サイトでのインスタンス化プロセスには時間を要する場合があるため（必要な時間は新しいマテリアライズド・ビューに移入されるデータの量によって異なります）、生成されたスクリプトは、マスター・サイトに長時間接続したままにできないリモート・マテリアライズド・ビュー・サイトで使用してください。このファンクションは、ユーザー・インスタンス化ごとに個別に実行する必要があります。

このファンクションで生成されたスクリプトは、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納され、配置テンプレートの配布時に、レプリケーション管理ツールなどの Oracle のツール製品で使用されます。このファンクションで戻される数値は、USER_REPCAT_TEMP_OUTPUT 一時ビューから該当する情報を取り出すために使用されます。

注意： このファンクションは、配置テンプレートのオフライン・インスタンス化の実行時に使用されます。また、別のユーザーのためにインスタンス化を実行しているレプリケーション管理者用でもあります。独自のインスタンス化を実行するユーザーは、パブリック・バージョンの INSTANTIATE_OFFLINE ファンクションを使用してください。詳細は、56-47 ページの「INSTANTIATE_OFFLINE ファンクション」を参照してください。

このファンクションを、DBMS_OFFLINE_OG パッケージ内のプロシージャ（マスター表のオフライン・インスタンス化の実行に使用）や DBMS_OFFLINE_SNAPSHOT パッケージ内のプロシージャ（マテリアライズド・ビューのオフライン・インスタンス化の実行に使用）と混同しないでください。それぞれのパッケージの使用方法は、該当の章を参照してください。

構文

```
DBMS_REPCAT_RGT.INSTANTIATE_OFFLINE(
     refresh_template_name   IN   VARCHAR2,
     site_name               IN   VARCHAR2,
     user_name               IN   VARCHAR2  := NULL,
     runtime_parm_id         IN   NUMBER    := -1e-130,
     next_date               IN   DATE      := SYSDATE,
     interval                IN   VARCHAR2  := 'SYSDATE + 1',
     use_default_gowner      IN   BOOLEAN   := true)
     return NUMBER;
```

パラメータ

表 56-64 INSTANTIATE_OFFLINE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_name	配置テンプレートをインスタンス化するリモート・サイトの名前。
user_name	配置テンプレートをインスタンス化する認可ユーザーの名前。
runtime_parm_id	INSERT_RUNTIME_PARMS プロシージャを使用してランタイム・パラメータ値を定義している場合は、ランタイム・パラメータの作成時に使用した ID（GET_RUNTIME_PARM_ID ファンクションを使用して取り出された ID）を指定します。
next_date	リフレッシュ・グループの作成時に使用される次回リフレッシュ日付の値を指定します。
interval	リフレッシュ・グループの作成時に使用されるリフレッシュ間隔を指定します。
use_default_gowner	TRUE の場合、作成されたすべてのマテリアライズド・ビュー・グループはデフォルト・ユーザー PUBLIC が所有します。FALSE の場合、作成されたすべてのマテリアライズド・ビュー・グループはインスタンス化を実行したユーザーが所有します。

例外

表 56-65 INSTANTIATE_OFFLINE ファンクションの例外

例外	説明
miss_refresh_template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定した認可ユーザーの名前が無効か、または存在しません。指定したユーザーが DBA_REPCAT_USER_AUTHORIZATIONS ビューにリストされていることを検証してください。リストされていない場合、指定したユーザーは、ターゲット配置テンプレートのインスタンス化を認可されていません。

戻り値

表 56-66 INSTANTIATE_OFFLINE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	生成したインスタンス化スクリプトを USER_REPCAT_TEMP_OUTPUT 一時ビューで取り出すように選択したとき、output_id に対して生成されるシステム番号を指定します。

INSTANTIATE_ONLINE ファンクション

このファンクションは、マテリアライズド・ビュー・サイトがマスターに接続されているときに、リモート・マテリアライズド・ビュー・サイトでマテリアライズド・ビュー環境を作成するために使用する、スクリプトをマスター・サイトで生成します（つまり、マテリアライズド・ビュー・サイトがオンラインのとき）。リモート・マテリアライズド・ビュー・サイトでのインスタンス化プロセスは長くかかる場合があるため（必要な時間は新しいマテリアライズド・ビューに移入されるデータの量によって異なります）、生成されたスクリプトは、マスター・サイトに長時間接続したままにできるリモート・マテリアライズド・ビュー・サイトで使用してください。このファンクションは、ユーザー・インスタンス化ごとに個別に実行する必要があります。

このファンクションで生成されたスクリプトは、USER_REPCAT_TEMP_OUTPUT 一時ビューに格納され、配置テンプレートの配布時に、レプリケーション管理ツールなどの Oracle のツール製品で使用されます。このファンクションで戻される数値は、USER_REPCAT_TEMP_OUTPUT 一時ビューから該当する情報を取り出すために使用されます。

注意： このファンクションは、別のユーザーのためにインスタンス化を実行しているレプリケーション管理者用でもあります。独自のインスタンス化を実行するユーザーは、56-47 ページの「INSTANTIATE_OFFLINE ファンクション」の項に記述されている、パブリック・バージョンの INSTANTIATE_OFFLINE ファンクションを使用してください。

構文

```
DBMS_REPCAT_RGT.INSTANTIATE_ONLINE(
   refresh_template_name   IN   VARCHAR2,
   site_name               IN   VARCHAR2  := NULL,
   user_name               IN   VARCHAR2  := NULL,
   runtime_parm_id         IN   NUMBER    := -1e-130,
   next_date               IN   DATE      := SYSDATE,
   interval                IN   VARCHAR2  := 'SYSDATE + 1',
   use_default_gowner      IN   BOOLEAN   := true)
   return NUMBER;
```

パラメータ

表 56-67 INSTANTIATE_ONLINE ファンクションのパラメータ

パラメータ	説明
refresh_template_name	インスタンス化する配置テンプレートの名前。
site_name	配置テンプレートをインスタンス化するリモート・サイトの名前。
user_name	配置テンプレートをインスタンス化する認可ユーザーの名前。
runtime_parm_id	INSERT_RUNTIME_PARMS プロシージャを使用してランタイム・パラメータ値を定義している場合は、ランタイム・パラメータの作成時に使用した ID（GET_RUNTIME_PARM_ID ファンクションを使用して取り出された ID）を指定します。
next_date	リフレッシュ・グループの作成時に使用される次回リフレッシュ日付の値を指定します。
interval	リフレッシュ・グループの作成時に使用されるリフレッシュ間隔を指定します。
use_default_gowner	TRUE の場合、作成されたすべてのマテリアライズド・ビュー・グループはデフォルト・ユーザー PUBLIC が所有します。FALSE の場合、作成されたすべてのマテリアライズド・ビュー・グループはインスタンス化を実行したユーザーが所有します。

例外

表 56-68 INSTANTIATE_ONLINE ファンクションの例外

例外	説明
miss_refresh_ template	指定した配置テンプレート名が無効か、または存在しません。
miss_user	指定した認可ユーザーの名前が無効か、または存在しません。指定したユーザーが DBA_REPCAT_USER_AUTHORIZATIONS ビューにリストされていることを検証してください。リストされていない場合、指定したユーザーは、ターゲット配置テンプレートのインスタンス化を認可されていません。
bad_parms	定義したユーザー・パラメータ値またはテンプレート・デフォルト値によって移入されていないテンプレート・パラメータがあります。事前定義値の数とテンプレート・パラメータ数が一致していないか、または事前定義値がターゲット・パラメータに対して無効です（タイプの不一致など）。

戻り値

表 56-69 INSTANTIATE_ONLINE ファンクションの戻り値

戻り値	説明
< システム生成番号 >	生成したインスタンス化スクリプトを USER_REPCAT_TEMP_OUTPUT 一時ビューで取り出すように選択したとき、output_id に対して生成されるシステム番号を指定します。

LOCK_TEMPLATE_EXCLUSIVE プロシージャ

このプロシージャは、配置テンプレートの更新中または変更中に、ユーザーがテンプレートの読込みまたはインスタンス化を実行できないようにします。

ロックは、ROLLBACK または COMMIT が実行されるとリリースされます。

> **注意：** 配置テンプレートを変更する前に、このプロシージャを必ず実行してください。

構文

```
DBMS_REPCAT_RGT.LOCK_TEMPLATE_EXCLUSIVE();
```

LOCK_TEMPLATE_SHARED プロシージャ

この LOCK_TEMPLATE_SHARED プロシージャは、指定した配置テンプレートを読取り専用にします。テンプレートをインスタンス化する前に、このプロシージャを必ずコールしてください。この結果、インスタンス化中に他のユーザーによって配置テンプレートが変更されないことが保証されます。

ロックは、ROLLBACK または COMMIT が実行されるとリリースされます。

構文

```
DBMS_REPCAT_RGT.LOCK_TEMPLATE_SHARED();
```

57

DBMS_REPUTIL

DBMS_REPUTIL には、表のレプリケーションで使用するシャドウ表、トリガーおよびパッケージを生成するサブプログラムの他に、スタンドアロン・プロシージャ起動およびパッケージ・プロシージャ起動のレプリケーションに使用するラッパーを生成するサブプログラムが含まれています。このパッケージは、生成コードでのみ参照されます。

この章では、次の項目について説明します。

DBMS_REPUTIL サブプログラムの要約

表 57-1 DBMS_REPUTIL パッケージのサブプログラム

サブプログラム	説明
「REPLICATION_OFF プロシージャ」57-3 ページ	変更内容をレプリケーション環境にある他のサイトにレプリケートせずに表を変更したり、またはプロシージャ・レプリケーションの使用時に行レベル・レプリケーションを使用禁止にします。
「REPLICATION_ON プロシージャ」57-3 ページ	レプリケーションが一時的に中断された後に、変更内容のレプリケーションを再び使用可能にします。
「REPLICATION_IS_ON ファンクション」57-3 ページ	レプリケーションが実行中かどうかを判別します。
「FROM_REMOTE ファンクション」57-4 ページ	内部レプリケーション・パッケージにあるプロシージャの開始時に TRUE を戻し、終了時に FALSE を戻します。
「GLOBAL_NAME ファンクション」57-4 ページ	ローカル・データベースのグローバル・データベース名を判別します（戻り値はグローバル名）。
「MAKE_INTERNAL_PKG プロシージャ」57-4 ページ	レプリケーション・カタログの内部パッケージおよび表を同期化します。 **注意：**オラクル社カスタマ・サポート・センターから指示がないかぎり、このプロシージャは実行しないでください。
「SYNC_UP_REP プロシージャ」57-5 ページ	レプリケーション・カタログの内部トリガーおよび表またはマテリアライズド・ビューを同期化します。 **注意：**オラクル社カスタマ・サポート・センターから指示がないかぎり、このプロシージャは実行しないでください。

REPLICATION_OFF プロシージャ

このプロシージャでは、変更内容をレプリケーション環境にある他のサイトにレプリケートせずに表を変更できます。プロシージャ・レプリケーションを使用する場合に、行レベル・レプリケーションが使用禁止になります。一般的に、このフラグを設定する前には、レプリケーション環境にあるすべてのマスター・グループに対するレプリケーション・アクティビティを中断する必要があります。

構文

```
DBMS_REPUTIL.REPLICATION_OFF();
```

REPLICATION_ON プロシージャ

このプロシージャは、レプリケーションが一時的に中断された後に、変更内容のレプリケーションを再び使用可能にします。

構文

```
DBMS_REPUTIL.REPLICATION_ON();
```

REPLICATION_IS_ON ファンクション

このファンクションは、レプリケーションが実行中かどうかを判別します。戻り値が TRUE の場合は、生成されたレプリケーション・トリガーが使用可能であることを示します。戻り値が FALSE の場合は、レプリケーション・グループに対する現行のサイトでのレプリケーションは使用禁止であることを示します。

このファンクションの戻り値は、DBMS_REPUTIL パッケージの REPLICATION_ON または REPLICATION_OFF プロシージャをコールして設定されます。

構文

```
DBMS_REPUTIL.REPLICATION_IS_ON()
  return BOOLEAN;
```

FROM_REMOTE ファンクション

このファンクションは、内部レプリケーション・パッケージにあるプロシージャの開始時に TRUE を戻し、終了時に FALSE を戻します。内部パッケージによる更新の結果、起動するトリガーがある場合は、このファンクションをチェックする必要があります。

構文

```
DBMS_REPUTIL.FROM_REMOTE()
   return BOOLEAN;
```

GLOBAL_NAME ファンクション

このファンクションは、ローカル・データベースのグローバル・データベース名を判別します（戻り値はグローバル名）。

構文

```
DBMS_REPUTIL.GLOBAL_NAME()
   return VARCHAR2;
```

MAKE_INTERNAL_PKG プロシージャ

このプロシージャは、実在する内部パッケージとレプリケーション・カタログの表またはマテリアライズド・ビューを同期化します。表にレプリケーション・サポートがある場合は、このプロシージャを実行して内部パッケージを作成できます。レプリケーション・サポートが存在しない場合、関連する内部パッケージはこのプロシージャによって破棄されます。このプロシージャでは、ネストした表の記憶表は受け入れられません。

注意： オラクル社カスタマ・サポート・センターから指示がないかぎり、このプロシージャは実行しないでください。

構文

```
DBMS_REPUTIL.MAKE_INTERNAL_PKG (
   canon_sname    IN   VARCHAR2,
   canon_oname    IN   VARCHAR2);
```

パラメータ

表 57-2 MAKE_INTERNAL_PKG プロシージャのパラメータ

パラメータ	説明
canon_sname	同期化する表を含んだスキーマ。 このパラメータ値は規範的に定義する必要があります（オブジェクトと一致する大文字を使用し、二重引用符で囲まないでください）。
canon_oname	同期化する表の名前。 このパラメータ値は規範的に定義する必要があります（オブジェクトと一致する大文字を使用し、二重引用符で囲まないでください）。

SYNC_UP_REP プロシージャ

このプロシージャは、実在する内部トリガーとレプリケーション・カタログの表またはマテリアライズド・ビューを同期化します。表またはマテリアライズド・ビューにレプリケーション・サポートがある場合は、このプロシージャを実行して内部レプリケーション・トリガーを作成できます。レプリケーション・サポートが存在しない場合、関連する内部トリガーはこのプロシージャによって破棄されます。このプロシージャでは、ネストした表の記憶表は受け入れられません。

注意： オラクル社カスタマ・サポート・センターから指示がないかぎり、このプロシージャは実行しないでください。

構文

```
DBMS_REPUTIL.SYNC_UP_REP (
   canon_sname    IN   VARCHAR2,
   canon_oname    IN   VARCHAR2);
```

パラメータ

表 57-3 SYNC_UP_REP プロシージャのパラメータ

パラメータ	説明
canon_sname	同期化する表またはマテリアライズド・ビューを含んだスキーマ。 このパラメータ値は規範的に定義する必要があります（オブジェクトと一致する大文字を使用し、二重引用符で囲まないでください）。

表 57-3 SYNC_UP_REP プロシージャのパラメータ

パラメータ	説明
canon_oname	同期化する表またはマテリアライズド・ビューの名前。 このパラメータ値は規範的に定義する必要があります（オブジェクトと一致する大文字を使用し、二重引用符で囲まないでください）。

58

DBMS_RESOURCE_MANAGER

DBMS_RESOURCE_MANAGER パッケージは、プラン、コンシューマ・グループおよびプラン・ディレクティブをメンテナンスします。また、プラン・スキーマへの変更内容をグループ化する方法も提供します。

> **関連項目：** データベース・リソース・マネージャの使用方法の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

この章では、次の項目について説明します。

要件

実行者には、このプロシージャを実行するための ADMINISTER_RESOURCE_MANAGER システム権限が必要です。この権限の付与および取消しを行うプロシージャは、DBMS_RESOURCE_MANAGER_PRIVS パッケージにあります。

DBMS_RESOURCE_MANAGER サブプログラムの要約

表 58-1 DBMS_RESOURCE_MANAGER パッケージのサブプログラム

サブプログラム	説明
「CREATE_PLAN プロシージャ」 58-3 ページ	リソース・プランを定義するエントリを作成します。
「CREATE_SIMPLE_PLAN プロシージャ」 58-4 ページ	最大 8 つのコンシューマ・グループが含まれた単一レベルのリソース・プランを 1 ステップで作成します。
「UPDATE_PLAN プロシージャ」 58-5 ページ	リソース・プランを定義するエントリを更新します。
「DELETE_PLAN プロシージャ」 58-6 ページ	指定のプランおよびそれが参照するすべてのプラン・ディレクティブを削除します。
「DELETE_PLAN_CASCADE プロシージャ」 58-6 ページ	指定のプランおよびそのすべての子（プラン・ディレクティブ、サブプラン、コンシューマ・グループ）を削除します。
「CREATE_CONSUMER_GROUP プロシージャ」 58-7 ページ	リソース・コンシューマ・グループを定義するエントリを作成します。
「UPDATE_CONSUMER_GROUP プロシージャ」 58-8 ページ	リソース・コンシューマ・グループを定義するエントリを更新します。
「DELETE_CONSUMER_GROUP プロシージャ」 58-9 ページ	リソース・コンシューマ・グループを定義するエントリを削除します。
「CREATE_PLAN_DIRECTIVE プロシージャ」 58-9 ページ	リソース・プラン・ディレクティブを作成します。
「UPDATE_PLAN_DIRECTIVE プロシージャ」 58-11 ページ	リソース・プラン・ディレクティブを更新します。
「DELETE_PLAN_DIRECTIVE プロシージャ」 58-13 ページ	リソース・プラン・ディレクティブを削除します。
「CREATE_PENDING_AREA プロシージャ」 58-13 ページ	リソース・マネージャ・オブジェクトへの変更を行うための作業領域を作成します。

表 58-1 DBMS_RESOURCE_MANAGER パッケージのサブプログラム（続き）

サブプログラム	説明
「VALIDATE_PENDING_AREA プロシージャ」58-15 ページ	リソース・マネージャに対する保留中の変更を検証します。
「CLEAR_PENDING_AREA プロシージャ」58-15 ページ	リソース・マネージャに対する作業領域を消去します。
「SUBMIT_PENDING_AREA プロシージャ」58-15 ページ	リソース・マネージャに対する保留中の変更を実行します。
「SET_INITIAL_CONSUMER_GROUP プロシージャ」58-19 ページ	ユーザーに対して、初期リソース・コンシューマ・グループを割り当てます。
「SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャ」58-20 ページ	指定のセッションのリソース・コンシューマ・グループを変更します。
「SWITCH_CONSUMER_GROUP_FOR_USER プロシージャ」58-20 ページ	指定のユーザー名で、すべてのセッションのリソース・コンシューマ・グループを変更します。

CREATE_PLAN プロシージャ

このプロシージャは、リソース・プランを定義するエントリを作成します。Oracle9*i* では、`max_active_sess_target_mth` が改名され、`active_sess_pool_mth` および `new_queueing_mth` が追加されています。

構文

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN (
   plan                      IN VARCHAR2,
   comment                   IN VARCHAR2,
   cpu_mth                   IN VARCHAR2 DEFAULT 'EMPHASIS',
   active_sess_pool_mth       IN VARCHAR2 DEFAULT 'ACTIVE_SESS_POOL_ABSOLUTE',
   parallel_degree_limit_mth  IN VARCHAR2 DEFAULT
                             'PARALLEL_DEGREE_LIMIT_ABSOLUTE',
   new_queueing_mth              IN VARCHAR2 DEFAULT 'FIFO_TIMEOUT',);
```

パラメータ

表 58-2 CREATE_PLAN プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前。
comment	ユーザーのコメント。
cpu_mth	CPU リソースに対する割当て方法。
active_sess_pool_mth	アクティブな最大セッションに対する割当て方法。
parallel_degree_limit_mth	並列度に対する割当て方法。
new_queueing_mth	アクティブなセッションのプール機能で使用するキューイング・ポリシーのタイプを指定します。

CREATE_SIMPLE_PLAN プロシージャ

このプロシージャでは、最大 8 つのコンシューマ・グループが含まれた単一レベルのリソース・プランを 1 ステップで作成します。リソース・プランの作成前にペンディング・エリアを手動で作成したり、CREATE_CONSUMER_GROUP プロシージャおよび CREATE_RESOURCE_PLAN_DIRECTIVES プロシージャを個別に使用する必要がありません。

構文

```
DBMS_RESOURCE_MANAGER.CREATE_SIMPLE_PLAN (
   SIMPLE_PLAN      IN  VARCHAR2  DEFAULT,
   CONSUMER_GROUP1  IN  VARCHAR2  DEFAULT,
   GROUP1_CPU       IN  NUMBER    DEFAULT,
   CONSUMER_GROUP2  IN  VARCHAR2  DEFAULT,
   GROUP2_CPU       IN  NUMBER    DEFAULT,
   CONSUMER_GROUP3  IN  VARCHAR2  DEFAULT,
   GROUP3_CPU       IN  NUMBER    DEFAULT,
   CONSUMER_GROUP4  IN  VARCHAR2  DEFAULT,
   GROUP4_CPU       IN  NUMBER    DEFAULT,
   CONSUMER_GROUP5  IN  VARCHAR2  DEFAULT,
   GROUP5_CPU       IN  NUMBER    DEFAULT,
   CONSUMER_GROUP6  IN  VARCHAR2  DEFAULT,
   GROUP6_CPU       IN  NUMBER    DEFAULT,
   CONSUMER_GROUP7  IN  VARCHAR2  DEFAULT,
   GROUP7_CPU       IN  NUMBER    DEFAULT,
   CONSUMER_GROUP8  IN  VARCHAR2  DEFAULT,
   GROUP8_CPU       IN  NUMBER    DEFAULT);
```

UPDATE_PLAN プロシージャ

このプロシージャでは、リソース・プランを定義するエントリが更新されます。Oracle9*i* では、new_max_active_sess_target_mth が改名され、new_active_sess_pool_mth および new_queueing_mth が追加されています。

構文

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN (
   plan                              IN VARCHAR2,
   new_comment                       IN VARCHAR2 DEFAULT NULL,
   new_cpu_mth                       IN VARCHAR2 DEFAULT NULL,
   new_active_sess_pool_mth          IN VARCHAR2 DEFAULT NULL,
   new_parallel_degree_limit_mth     IN VARCHAR2 DEFAULT NULL,
   new_queueing_mth                  IN VARCHAR2 DEFAULT NULL,
   new_group_switch_mth              IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 58-3 UPDATE_PLAN プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前。
new_comment	新しいユーザーのコメント。
new_cpu_mth	CPU リソースに対する新規の割当て方法名。
new_active_sess_pool_mth	アクティブな最大セッションに対する新規の方法名。
new_parallel_degree_limit_mth	並列度に対する新規の方法名。
new_queueing_mth	アクティブなセッションのプール機能で使用するキューイング・ポリシーのタイプを指定します。

使用上の注意

UPDATE_PLAN に対するパラメータを指定しない場合、これらのパラメータは、データ・ディクショナリ内で変更されないまま残ります。

DELETE_PLAN プロシージャ

このプロシージャは、指定のプランおよびそれが参照するすべてのプラン・ディレクティブを削除します。

構文

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN (
   plan IN VARCHAR2);
```

パラメータ

表 58-4 DELETE_PLAN プロシージャのパラメータ

パラメータ	説明
plan	削除するリソース・プランの名前。

DELETE_PLAN_CASCADE プロシージャ

このプロシージャは、指定のプランおよびそのすべての子（プラン・ディレクティブ、サブプラン、コンシューマ・グループ）を削除します。必須オブジェクトおよび必須ディレクティブは削除されません。

構文

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN_CASCADE (
   plan IN VARCHAR2);
```

パラメータ

表 58-5 DELETE_PLAN_CASCADE プロシージャのパラメータ

パラメータ	説明
plan	プランの名前。

エラー

DELETE_PLAN_CASCADE プロシージャでエラーが発生した場合は、ロールバックされるため、何も削除されません。

> **注意：** デフォルトのリソース割当て方法を使用する場合は、プランの作成または更新時に方法を指定する必要はありません。

使用上の注意

デフォルトは次のとおりです。

- cpu_method = EMPHASIS
- parallel_degree_limit_mth = PARALLEL_DEGREE_LIMIT_ABSOLUTE
- active_sess_pool_mth = MAX_ACTIVE_SESS_ABSOLUTE

> **注意：** パラメータ max_active_sess_target_mth は、このリリースでは記載されていません。このパラメータは、将来使用するために予約されています。

CREATE_CONSUMER_GROUP プロシージャ

このプロシージャによって、リソース・コンシューマ・グループを定義するエントリを作成します。

構文

```
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
   consumer_group  IN VARCHAR2,
   comment         IN VARCHAR2,
   cpu_mth         IN VARCHAR2 DEFAULT 'ROUND-ROBIN');
```

パラメータ

表 58-6 CREATE_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
consumer_group	コンシューマ・グループの名前。
comment	ユーザーのコメント。
cpu_mth	CPU リソース割当て方法名。

UPDATE_CONSUMER_GROUP プロシージャ

このプロシージャによって、リソース・コンシューマ・グループを定義するエントリを更新します。

構文

```
DBMS_RESOURCE_MANAGER.UPDATE_CONSUMER_GROUP (
   consumer_group  IN VARCHAR2,
   new_comment     IN VARCHAR2 DEFAULT NULL,
   new_cpu_mth     IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 58-7 UPDATE_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
consumer_group	コンシューマ・グループの名前。
new_comment	新しいユーザーのコメント。
new_cpu_mth	CPU リソース割当てに対する新規の方法名。

UPDATE_CONSUMER_GROUP に対するパラメータを指定しない場合、これらのパラメータは、データ・ディクショナリ内で変更されないまま残ります。

DELETE_CONSUMER_GROUP プロシージャ

このプロシージャによって、リソース・コンシューマ・グループを定義するエントリを削除します。

構文

```
DBMS_RESOURCE_MANAGER.DELETE_CONSUMER_GROUP (
   consumer_group IN VARCHAR2);
```

パラメータ

表 58-8 DELETE_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
consumer_group	削除するコンシューマ・グループの名前。

CREATE_PLAN_DIRECTIVE プロシージャ

このプロシージャでは、リソース・プラン・ディレクティブを作成できます。Oracle9*i* では、new_max_active_sess_target_mth が改名され、複数の新しいパラメータが追加されています。

構文

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   plan                     IN VARCHAR2,
   group_or_subplan         IN VARCHAR2,
   comment                  IN VARCHAR2,
   cpu_p1                   IN NUMBER   DEFAULT NULL,
   cpu_p2                   IN NUMBER   DEFAULT NULL,
   cpu_p3                   IN NUMBER   DEFAULT NULL,
   cpu_p4                   IN NUMBER   DEFAULT NULL,
   cpu_p5                   IN NUMBER   DEFAULT NULL,
   cpu_p6                   IN NUMBER   DEFAULT NULL,
   cpu_p7                   IN NUMBER   DEFAULT NULL,
   cpu_p8                   IN NUMBER   DEFAULT NULL,
   active_sess_pool_p1      IN NUMBER   DEFAULT UNLIMITED,
   queueing_p1              IN NUMBER   DEFAULT UNLIMITED,
   switch_group             IN VARCHAR2 DEFAULT NULL,
   switch_time              IN NUMBER   DEFAULT UNLIMITED,
   switch_estimate          IN BOOLEAN  DEFAULT FALSE,
   max_est_exec_time        IN NUMBER   DEFAULT UNLIMITED,
```

```
  undo_pool                 IN NUMBER   DEFAULT UNLIMITED,
  parallel_degree_limit_p1  IN NUMBER   DEFAULT UNLIMITED);
```

パラメータ

表 58-9 CREATE_PLAN_DIRECTIVE プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前。
group_or_subplan	コンシューマ・グループの名前またはサブプランの名前。
comment	プラン・ディレクティブについてのコメント。
cpu_p1	CPU リソース割当て方法に対する第 1 パラメータ。
cpu_p2	CPU リソース割当て方法に対する第 2 パラメータ。
cpu_p3	CPU リソース割当て方法に対する第 3 パラメータ。
cpu_p4	CPU リソース割当て方法に対する第 4 パラメータ。
cpu_p5	CPU リソース割当て方法に対する第 5 パラメータ。
cpu_p6	CPU リソース割当て方法に対する第 6 パラメータ。
cpu_p7	CPU リソース割当て方法に対する第 7 パラメータ。
cpu_p8	CPU リソース割当て方法に対する第 8 パラメータ。
active_sess_pool_p1	最大アクティブ・セッション割当て方法に対する第 1 パラメータ（将来使用のために予約）。
queueing_p1	キューのタイムアウト（秒数）。
switch_group	切替え時間に達したときに、切替え先となるグループ。
switch_time	切替え時間。
switch_estimate	TRUE の場合は、実行前に、見積り実行時間が操作のコンシューマ・グループに自動的に切り換えられるように Oracle に指示します。デフォルトは FALSE です。
max_est_exec_time	最大見積り実行時間（秒）。
undo_pool	コンシューマ・グループの取消しプール・サイズ（キロバイト）。
parallel_degree_limit_p1	並列度の割当て方法に対する第 1 パラメータ。

すべてのパラメータは、NULL にデフォルト設定されます。ただし、EMPHASIS CPU リソース割当て方法の場合は、ユーザーがすべてのパラメータを入力する必要があります。

UPDATE_PLAN_DIRECTIVE プロシージャ

このプロシージャによって、リソース・プラン・ディレクティブを更新します。Oracle9*i* では、new_max_active_sess_target_mth が改名され、new_active_sess_pool_p1 の他、複数の新しいパラメータが追加されています。

構文

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE (
   plan                          IN VARCHAR2,
   group_or_subplan              IN VARCHAR2,
   new_comment                   IN VARCHAR2 DEFAULT NULL,
   new_cpu_p1                    IN NUMBER   DEFAULT NULL,
   new_cpu_p2                    IN NUMBER   DEFAULT NULL,
   new_cpu_p3                    IN NUMBER   DEFAULT NULL,
   new_cpu_p4                    IN NUMBER   DEFAULT NULL,
   new_cpu_p5                    IN NUMBER   DEFAULT NULL,
   new_cpu_p6                    IN NUMBER   DEFAULT NULL,
   new_cpu_p7                    IN NUMBER   DEFAULT NULL,
   new_cpu_p8                    IN NUMBER   DEFAULT NULL,
   new_active_sess_pool_p1       IN NUMBER   DEFAULT NULL,
   new_queueing_p1               IN NUMBER   DEFAULT NULL,
   new_parallel_degree_limit_p1  IN NUMBER   DEFAULT NULL
   new_switch_group              IN VARCHAR2 DEFAULT NULL,
   new_switch_time               IN NUMBER   DEFAULT NULL,
   new_switch_estimate           IN BOOLEAN  DEFAULT FALSE,
   new_max_est_exec_time         IN NUMBER   DEFAULT NULL,
   new_undo_pool                 IN NUMBER   DEFAULT UNLIMITED);
```

パラメータ

表 58-10 UPDATE_PLAN_DIRECTIVE プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前。
group_or_subplan	コンシューマ・グループの名前またはサブプランの名前。
new_comment	プラン・ディレクティブについてのコメント。
new_cpu_p1	CPU リソース割当て方法に対する第 1 パラメータ。
new_cpu_p2	CPU リソース割当て方法に対する第 2 パラメータ。
new_cpu_p3	CPU リソース割当て方法に対する第 3 パラメータ。

表 58-10 UPDATE_PLAN_DIRECTIVE プロシージャのパラメータ（続き）

パラメータ	説明
new_cpu_p4	CPU リソース割当て方法に対する第 4 パラメータ。
new_cpu_p5	CPU リソース割当て方法に対する第 5 パラメータ。
new_cpu_p6	CPU リソース割当て方法に対する第 6 パラメータ。
new_cpu_p7	CPU リソース割当て方法に対する第 7 パラメータ。
new_cpu_p8	CPU リソース割当て方法に対する第 8 パラメータ。
new_active_sess_pool_p1	最大アクティブ・セッション割当て方法に対する第 1 パラメータ（将来使用のために予約）。
new_queueing_p1	キューのタイムアウト（秒数）。
new_switch_group	切替え時間に達したときに、切替え先となるグループ。
new_switch_time	切替え時間。
new_switch_estimate	TRUE の場合は、実行前に、見積り実行時間が操作のコンシューマ・グループに自動的に切り換えられるように Oracle に指示します。デフォルトは FALSE です。
new_max_est_exec_time	最大見積り実行時間（秒）。
new_undo_pool	コンシューマ・グループの取消しプール・サイズ（キロバイト）。
new_parallel_degree_limit_p1	並列度の割当て方法に対する第 1 パラメータ。

UPDATE_PLAN_DIRECTIVE に対してパラメータを指定しない場合、これらのパラメータは、データ・ディクショナリ内で変更されないまま残ります。

DELETE_PLAN_DIRECTIVE プロシージャ

このプロシージャによって、リソース・プラン・ディレクティブを削除します。

構文

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN_DIRECTIVE (
   plan              IN VARCHAR2,
   group_or_subplan  IN VARCHAR2);
```

パラメータ

表 58-11 DELETE_PLAN_DIRECTIVE プロシージャのパラメータ

パラメータ	説明
plan	リソース・プランの名前。
group_or_subplan	グループの名前またはサブプランの名前。

CREATE_PENDING_AREA プロシージャ

このプロシージャによって、リソース・マネージャ・オブジェクトに変更を加えます。

プラン・スキーマへのすべての変更は、ペンディング・エリア内で行う必要があります。ペンディング・エリアは、プラン・スキーマを変更するためのスクラッチ領域とみなすことができます。管理者は、このペンディング・エリアを作成し、必要に応じて変更を加え、場合によってその変更を検証します。その実行が完了したときのみ、その変更内容がアクティブになります。

ペンディング・エリアがアクティブな間は、変更された現行のプラン・スキーマを適切なユーザー・ビューから選択して、いつでも表示できます。

現行の変更を中止する場合は、いつでもペンディング・エリアを消去できます。また、VALIDATE プロシージャをコールして、変更が有効になっているかどうかを確認できます。変更は、エントリ・グループの一貫性を維持するための指定の順序で行う必要はありません。これらのチェックは、ペンディング・エリアが実行されるときにも暗黙的に行われます。

注意： Oracle では、孤立したコンシューマ・グループ（つまり、そのコンシューマ・グループを参照するプラン・ディレクティブがないコンシューマ・グループ）が可能です。これは、現在は使用しないが将来使用するコンシューマ・グループを管理者があらかじめ作成できるようにするためです。

構文

```
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;
```

使用上の注意

次のルールを厳守してください。これらのルールは、VALIDATE または SUBMIT プロシージャが実行されるたびにチェックされます。

1. プラン・スキーマにループがないこと。
2. プラン・ディレクティブが参照するすべてのプランおよびコンシューマ・グループがあること。
3. すべてのプランに、プランまたはコンシューマ・グループのいずれかを参照するプラン・ディレクティブがあること。
4. リソース割当て方法が EMPHASIS の場合は、指定レベルでのパーセントの合計が 100 を超えないこと。
5. アクティブなインスタンスでトップレベルのプランとして現在使用されているプランを削除しないこと。
6. Oracle8*i* の場合、プラン・ディレクティブのパラメータ parallel_degree_limit_p1 は、コンシューマ・グループ（つまり、サブプランではなく）を参照するプラン・ディレクティブでのみ表示されます。
7. 指定のプランでのプラン・ディレクティブは 32 を超えないこと（つまり、プランは 33 以上の子を持つことはできません）。
8. アクティブなプラン・スキーマ内のコンシューマ・グループは 32 を超えないこと。
9. プランとコンシューマ・グループは同じネームスペースを使用するため、コンシューマ・グループと同じ名前のプランがないこと。
10. アクティブなプラン・スキーマ内に OTHER_GROUPS に対するプラン・ディレクティブがあること。これにより、現在アクティブなプランがカバーしていないセッションに OTHER_GROUPS ディレクティブが指定したリソースが割り当てられます。

VALIDATE または SUBMIT プロシージャによるチェック時に、前述のルールのいずれかに違反していると、それを通知するエラー・メッセージが戻されます。変更して問題を修正し、VALIDATE または SUBMIT プロシージャを再発行できます。

VALIDATE_PENDING_AREA プロシージャ

このプロシージャによって、リソース・マネージャに対する保留中の変更内容を検証します。

構文

```
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA;
```

CLEAR_PENDING_AREA プロシージャ

このプロシージャによって、リソース・マネージャに対する保留中の変更内容を消去します。

構文

```
DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA;
```

SUBMIT_PENDING_AREA プロシージャ

このプロシージャによって、リソース・マネージャに対する保留中の変更を発行します。変更内容を検証してコミットした後（その変更内容が有効な場合）、ペンディング・エリアを消去します。

注意： SUBMIT_PENDING_AREA へのコールは、VALIDATE_PENDING_AREA が成功していても失敗する場合があります。これは、削除するプランがインスタンスによって VALIDATE_PENDING_AREA へのコール後、SUBMIT_PENDING_AREA へのコールまでにロードされると発生する可能性があります。

構文

```
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA;
```

例

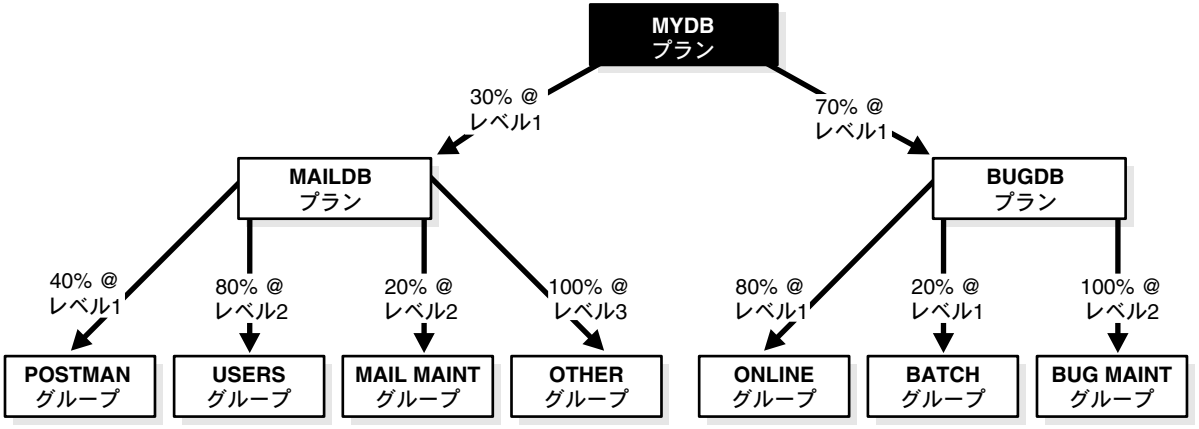
プランの利点の 1 つは、相互に参照できることです。プランのエントリは、コンシューマ・グループまたはサブプランのいずれかになります。次に、有効な CPU プラン・ディレクティブのセット例を示します。

表 58-12 MYDB PLAN CPU プラン・ディレクティブ

サブプラン / グループ	CPU_Level 1
MAILDB プラン	30%
BUGDB プラン	70%

これらのプラン・ディレクティブが有効で、すべてのコンシューマ・グループに実行可能なセッションが無限にある場合、MAILDB プランには使用可能な CPU リソースの 30% が割り当てられ、一方、BUGDB プランにはその 70% が割り当てられます。これをさらに分割すると、POSTMAN コンシューマ・グループにあるセッションは 12%（30% の内の 40%）の時間を実行し、ONLINE コンシューマ・グループにあるセッションは 56%（70% の内の 80%）の時間を実行します。図 58-1 は、この使用例を表しています。

図 58-1 リソース・マネージャの使用例



次の説明では、コンシューマ・グループはアクティブなセッションです。つまり、セッションはリソース・コンシューマ・グループに所属し、このコンシューマ・グループは、処理するリソースの割当てを決定するためにプランが使用します。

CPU プラン・ディレクティブのマルチプラン（1 つ以上のサブプランを持つプラン）定義は、各プランがそれ自体をエンティティとして所有するため、1 セットのプラン・ディレクティブを持つ単一プランに縮小できません。プランまたはサブプランに割り当てられた

CPU 量は、アクティブ・セッションを持つコンシューマ・グループがそのプランに含まれていないかぎり、そのプラン内でのみ使用されます。したがって、この例では、BUG MAINT グループが CPU 量をまったく使用しなかった場合はそのプラン内でリサイクルされるので、BUGDB プラン内のレベル 1 に戻ります。前述の例で、マルチプラン定義が複数のコンシューマ・グループを持つ単一プランに縮小された場合は、BUG MAINT グループで使用されていない CPU 量を明示的にリサイクルする方法はありません。CPU 量はグローバルにリサイクルされるので、MAIL セッションにもそれを使用する機会が与えられます。

データベースのリソースは、複数のアプリケーション間の高いレベルでパーティション化でき、アプリケーション内で再パーティション化できます。アプリケーション内の指定のグループで、割り当てられたすべてのリソースが必要ない場合、そのリソースは、同じアプリケーション内でのみ再パーティション化されます。

次の例では、デフォルトのプランとコンシューマ・グループの割当て方法が使用されます。

```
BEGIN
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'bugdb_plan',
   COMMENT => 'Resource plan/method for bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'maildb_plan',
   COMMENT => 'Resource plan/method for mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_PLAN(PLAN => 'mydb_plan',
   COMMENT => 'Resource plan/method for bug and mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Online_group',
   COMMENT => 'Resource consumer group/method for online bug users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Batch_group',
COMMENT => 'Resource consumer group/method for bug users sessions who run batch jobs');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Bug_Maintenance_group',
   COMMENT => 'Resource consumer group/method for users sessions who maintain
   the bug  db');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_users_group',
   COMMENT => 'Resource consumer group/method for mail users sessions');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_Postman_group',
   COMMENT => 'Resource consumer group/method for mail postman');
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(CONSUMER_GROUP => 'Mail_Maintenance_group',
   COMMENT => 'Resource consumer group/method for users sessions who maintain the mail
   db');
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan', GROUP_OR_SUBPLAN =>
'Bug_Online_group',
   COMMENT => 'online bug users sessions at level 1', CPU_P1 => 80, CPU_P2=> 0,
   PARALLEL_DEGREE_LIMIT_P1 => 8);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan', GROUP_OR_SUBPLAN =>
'Bug_Batch_group',
   COMMENT => 'batch bug users sessions at level 1', CPU_P1 => 20, CPU_P2 => 0,
   PARALLEL_DEGREE_LIMIT_P1 => 2);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan', GROUP_OR_SUBPLAN =>
'Bug_Maintenance_group',
   COMMENT => 'bug maintenance users sessions at level 2', CPU_P1 => 0, CPU_P2 => 100,
PARALLEL_DEGREE_LIMIT_P1 => 3);
```

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'bugdb_plan', GROUP_OR_SUBPLAN =>
'OTHER_GROUPS',
   COMMENT => 'all other users sessions at level 3', CPU_P1 => 0, CPU_P2 => 0, CPU_P3 =>
100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan', GROUP_OR_SUBPLAN =>
'Mail_Postman_group',
   COMMENT => 'mail postman at level 1', CPU_P1 => 40, CPU_P2 => 0,
   PARALLEL_DEGREE_LIMIT_P1 => 4);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan', GROUP_OR_SUBPLAN =>
'Mail_users_group',
   COMMENT => 'mail users sessions at level 2', CPU_P1 => 0, CPU_P2 => 80,
   PARALLEL_DEGREE_LIMIT_P1 => 4);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan', GROUP_OR_SUBPLAN =>
'Mail_Maintenance_group',
   COMMENT => 'mail maintenance users sessions at level 2', CPU_P1 => 0, CPU_P2 => 20,
   PARALLEL_DEGREE_LIMIT_P1 => 2);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'maildb_plan', GROUP_OR_SUBPLAN =>
'OTHER_GROUPS',
   COMMENT => 'all other users sessions at level 3', CPU_P1 => 0, CPU_P2 => 0, CPU_P3 =>
100);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'mydb_plan', GROUP_OR_SUBPLAN =>
'maildb_plan',
   COMMENT=> 'all mail users sessions at level 1', CPU_P1 => 30);
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(PLAN => 'mydb_plan', GROUP_OR_SUBPLAN =>
'bugdb_plan',
   COMMENT => 'all bug users sessions at level 1', CPU_P1 => 70);
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA();
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA();
end;
```

検証は SUBMIT_PENDING_AREA で暗黙的に行われるので、前述の VALIDATE_PENDING_AREA へのコールはオプションです。

SET_INITIAL_CONSUMER_GROUP プロシージャ

ユーザーの初期コンシューマ・グループは、そのユーザーが作成したセッションが最初に所属しているコンシューマ・グループです。このプロシージャは、ユーザーに対して、初期のリソース・コンシューマ・グループを設定します。

構文

```
DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP (
   user           IN VARCHAR2,
   consumer_group IN VARCHAR2);
```

パラメータ

表 58-13 SET_INITIAL_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
user	ユーザー名。
consumer_group	ユーザーの初期コンシューマ・グループ。

使用上の注意

このプロシージャを実行するためには、ADMINISTER_RESOURCE_MANAGER または ALTER USER システム権限が必要です。ユーザーの初期コンシューマ・グループが設定される前に、ユーザーまたは PUBLIC に対して、コンシューマ・グループへの切替え権限が直接付与されている必要があります。初期コンシューマ・グループに対する切替え権限は、そのユーザーに付与されているロールから与えることはできません。

注意： この方法は、ALTER USER DEFAULT ROLE に対する方法に類似しています。

ユーザーの初期コンシューマ・グループが設定されていない場合は、自動的に DEFAULT_CONSUMER_GROUP がコンシューマ・グループになります。

DEFAULT_CONSUMER_GROUP は PUBLIC に付与された切替え権限を持つため、すべてのユーザーはこのコンシューマ・グループに対する切替え権限を自動的に付与されます。コンシューマ・グループを削除するとき、削除するグループを初期コンシューマ・グループとしていたすべてのユーザーは、DEFAULT_CONSUMER_GROUP を初期コンシューマ・グループとします。削除するコンシューマ・グループに所属している現行のアクティブなセッションは、すべて DEFAULT_CONSUMER_GROUP に切り替えられます。

SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャ

このプロシージャによって、指定のセッションのリソース・コンシューマ・グループを変更します。また、トップレベルのユーザー・セッションに関連する（PQ）スレーブ・セッションのコンシューマ・グループも変更します。

構文

```
DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESS (
   session_id      IN NUMBER,
   session_serial  IN NUMBER,
   consumer_group  IN VARCHAR2);
```

パラメータ

表 58-14 SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャのパラメータ

パラメータ	説明
session_id	ビュー V$SESSION での SID 列。
session_serial	ビュー V$SESSION での SERIAL# 列。
consumer_group	切り替えるコンシューマ・グループの名前。

SWITCH_CONSUMER_GROUP_FOR_USER プロシージャ

このプロシージャによって、指定のユーザー ID を持つすべてのセッションに対するリソース・コンシューマ・グループを変更します。また、トップレベルのユーザー・セッションに関連する（PQ）スレーブ・セッションのコンシューマ・グループも変更します。

構文

```
DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_USER (
   user            IN VARCHAR2,
   consumer_group  IN VARCHAR2);
```

パラメータ

表 58-15 SWITCH_CONSUMER_GROUP_FOR_USER プロシージャのパラメータ

パラメータ	説明
user	ユーザー名。
consumer_group	切り替えるコンシューマ・グループの名前。

使用上の注意

SWITCH_CONSUMER_GROUP_FOR_SESS プロシージャおよび SWITCH_CONSUMER_GROUP_FOR_USER プロシージャによって、特定のセッションまたはユーザーの CPU リソース割当てを増減します。これにより、UNIX の nice コマンドと類似した機能性が提供されます。

これらのプロシージャは、新規に指定されたコンシューマ・グループにセッションを即時に移動します。

59

DBMS_RESOURCE_MANAGER_PRIVS

DBMS_RESOURCE_MANAGER_PRIVS パッケージは、リソース・マネージャに関連付けられている権限をメンテナンスします。

> **関連項目：** データベース・リソース・マネージャの使用方法の詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

この章では、次の項目について説明します。

DBMS_RESOURCE_MANAGER_PRIVS サブプログラムの要約

表 59-1 DBMS_RESOURCE_MANAGER_PRIVS サブプログラム

サブプログラム	説明
「GRANT_SYSTEM_PRIVILEGE プロシージャ」59-2 ページ	システム権限を付与します。
「REVOKE_SYSTEM_PRIVILEGE プロシージャ」59-3 ページ	システム権限を取り消します。
「GRANT_SWITCH_CONSUMER_GROUP プロシージャ」59-4 ページ	リソース・コンシューマ・グループに切替え権限を付与します。
「REVOKE_SWITCH_CONSUMER_GROUP プロシージャ」59-5 ページ	リソース・コンシューマ・グループに切替え権限を取り消します。

GRANT_SYSTEM_PRIVILEGE プロシージャ

このプロシージャは、ユーザーまたはロールにシステム権限を付与します。

構文

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE (
   grantee_name   IN VARCHAR2,
   privilege_name IN VARCHAR2 DEFAULT 'ADMINISTER_RESOURCE_MANAGER',
   admin_option   IN BOOLEAN);
```

パラメータ

表 59-2 GRANT_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
grantee_name	権限を付与されるユーザーまたはロールの名前
privilege_name	付与される権限の名前
admin_option	ADMIN OPTION の付いた権限付与の場合は TRUE、そうでない場合は FALSE

Oracle では現在、リソース・マネージャに対するシステム権限は ADMINISTER_RESOURCE_MANAGER のみ提供しています。データベース管理者には、ADMIN OPTION を伴うこのシステム権限があります。その権限の付与と取消しは、ユーザーまたはロールに対して行うことができます。ADMIN OPTION を伴うシステム権限を付与されたユーザーは、この権限を他のユーザーにも付与できます。

例

次のコールは、scott というユーザーに対して ADMIN OPTION を付けずにこの権限を付与します。

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE (
   grantee_name => 'scott',
   admin_option => FALSE);
```

REVOKE_SYSTEM_PRIVILEGE プロシージャ

このプロシージャは、ユーザーまたはロールからシステム権限を取り消します。

構文

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SYSTEM_PRIVILEGE (
   revokee_name   IN VARCHAR2,
   privilege_name IN VARCHAR2 DEFAULT 'ADMINISTER_RESOURCE_MANAGER');
```

パラメータ

表 59-3 REVOKE_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
revokee_name	権限を取り消すユーザーまたはロールの名前
privilege_name	取り消す権限の名前

例

次のコールは、ユーザー名 scott から ADMINISTER_RESOURCE_MANAGER を取り消します。

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SYSTEM_PRIVILEGE ('scott');
```

GRANT_SWITCH_CONSUMER_GROUP プロシージャ

このプロシージャは、リソース・コンシューマ・グループに切替え権限を付与します。

構文

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP (
   grantee_name   IN VARCHAR2,
   consumer_group IN VARCHAR2,
   grant_option   IN BOOLEAN);
```

パラメータ

表 59-4 GRANT_SWITCH_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
grantee_name	権限を付与されるユーザーまたはロールの名前
consumer_group	コンシューマ・グループの名前
grant_option	権限を付与されたユーザーが他のユーザーにアクセス権を付与できる場合は TRUE、そうでない場合は FALSE

使用上の注意

ユーザーに対して特定のコンシューマ・グループに切替え許可を付与すると、そのユーザーは、即時に現行のコンシューマ・グループを新規のコンシューマ・グループに切り替えることができます。

ロールに対して特定のコンシューマ・グループに切替え許可を付与すると、そのロールを付与され、そのロールを使用可能にしたユーザーは、即時に現行のコンシューマ・グループを新規のコンシューマ・グループに切り替えることができます。

PUBLIC に対して特定のコンシューマ・グループに切替え許可を付与すると、すべてのユーザーがそのコンシューマ・グループに切り替えることができます。

grant_option パラメータが TRUE の場合、コンシューマ・グループに対する切替え権限が付与されたユーザーは、そのコンシューマ・グループに対する切替え権限を他のユーザーにも付与できます。

ユーザーの初期のコンシューマ・グループを設定するには、そのグループに対する切替え権限をユーザーに付与する必要があります。

関連項目： 第 58 章「DBMS_RESOURCE_MANAGER」

例

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_COMSUMER_GROUP (
   'scott', 'mail_maintenance_group', true);

DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP (
   'scott', 'mail_maintenance_group');
```

REVOKE_SWITCH_CONSUMER_GROUP プロシージャ

このプロシージャは、リソース・コンシューマ・グループから切替え権限を取り消します。

構文

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SWITCH_CONSUMER_GROUP (
   revokee_name   IN VARCHAR2,
   consumer_group IN VARCHAR2);
```

パラメータ

表 59-5 REVOKE_SWITCH_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
revokee_name	アクセス権を取り消すユーザーまたはロールの名前
consumer_group	コンシューマ・グループの名前

使用上の注意

特定のコンシューマ・グループに対する切替え権限をユーザーから取り消した後で、そのユーザーがそのコンシューマ・グループに切り替えようとしても失敗します。

ユーザーから初期のコンシューマ・グループを取り消した場合、そのユーザーは、ログイン時に自動的に DEFAULT_CONSUMER_GROUP コンシューマ・グループに所属します。

コンシューマ・グループに対する切替え権限をロールから取り消すと、そのロールを介してのみコンシューマ・グループに対する切替え権限を持っていたユーザーは、そのコンシューマ・グループに切り替えることができなくなります。

コンシューマ・グループに対する切替え権限を PUBLIC から取り消すと、それまで PUBLIC を介してのみコンシューマ・グループを使用できたユーザーは、そのコンシューマ・グループに切り替えることができなくなります。

例

次の例は、mail_maintenance_group に切替え権限を scott から取り消します。

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SWITCH_CONSUMER_GROUP (
   'scott', 'mail_maintenance_group');
```

60

DBMS_RESUMABLE

DBMS_RESUMABLE を使用すると、長時間にわたる実行後に領域を使い果たしたり、領域制限に達するような大規模な操作を一時停止し、問題を修正した後で文を再開できます。したがって、領域関連のエラーの発生を心配せずに、アプリケーションを作成できます。

文を中断した場合は、その中断をアラート・ログに記録する必要があります。文が中断されたときに実行するプロシージャも登録する必要があります。ビューを使用すると、文の進行状況を監視したり、その文が現在実行中なのか、または中断されている状態なのか表示できます。

文が中断されると、自動的にトランザクションも中断されます。したがって、文の中断時および再開時はすべてのトランザクション・リソースが保持されます。エラー条件が解消されると、中断されていた文は自動的に実行を再開します。再開領域割当ては、実行時に複数回中断および再開できます。

再開領域割当てには、中断タイムアウト時間が関連付けられます。タイムアウト時間（デフォルトでは 2 時間）中断している再開領域割当ては、起動して例外をユーザーに戻します。DMBS_RESUMABLE.ABORT() プロシージャを使用して、中断した文に強制的に例外を発生させることもできます。

この章では、次の項目について説明します。

- DBMS_RESUMABLE サブプログラムの要約

DBMS_RESUMABLE サブプログラムの要約

表 60-1 DBMS_RESUMABLE サブプログラム

サブプログラム	説明
「ABORT プロシージャ」 60-2 ページ	中断した再開領域割当てを終了します。
「GET_SESSION_TIMEOUT ファンクション」 60-3 ページ	session_id のセッションにおける再開領域割当ての現行のタイムアウト値を戻します。
「SET_SESSION_TIMEOUT プロシージャ」 60-3 ページ	session_id のセッションにおける再開領域割当てのタイムアウトを設定します。
「GET_TIMEOUT ファンクション」 60-4 ページ	現行のセッションにおける再開領域割当ての現行のタイムアウト値を戻します。
「SET_TIMEOUT プロシージャ」 60-4 ページ	現行のセッションにおける再開領域割当てのタイムアウトを設定します。
「SPACE_ERROR_INFO ファンクション」 60-5 ページ	エラー・スタックで領域関連のエラーを検索します。領域関連のエラーが見つからなかった場合は、FALSE が戻されます。

ABORT プロシージャ

このプロシージャでは、中断した再開領域割当てを終了します。パラメータ session_id は、文が実行されるセッション ID を示します。パラレル DML/DDL の場合は、session_id は、パラレル DML/DDL に参加するセッション ID を示します。この操作は成功が保証されています。このプロシージャのコールは、AFTER SUSPEND トリガーの内部からでも外部からでもコールできます。

ABORT プロシージャをコールするには、session_id のセッションの所有者、ALTER SYSTEM 権限の所持者または DBA である必要があります。

構文

```
DBMS_RESUMABLE.ABORT (
   session_id  IN NUMBER);
```

パラメータ

表 60-2 ABORT プロシージャのパラメータ

パラメータ	説明
session_id	再開領域割当てのセッション識別子。

GET_SESSION_TIMEOUT ファンクション

このファンクションは、session_id のセッションにおける再開領域割当ての現行のタイムアウト値を戻します。戻されるタイムアウトは秒で示されます。session_id が存在しない場合、GET_SESSION_TIMEOUT ファンクションは -1 を戻します。

構文

```
DBMS_RESUMABLE.GET_SESSION_TIMEOUT (
   session_id  IN NUMBER);
```

パラメータ

表 60-3 GET_SESSION_TIMEOUT ファンクションのパラメータ

パラメータ	説明
session_id	再開領域割当てのセッション識別子。

SET_SESSION_TIMEOUT プロシージャ

このプロシージャは、session_id のセッションにおける再開領域割当てのタイムアウトを設定します。戻されるタイムアウトは秒で示されます。新しいタイムアウト設定はセッションに即時適用されます。session_id が存在しない場合、操作は実行されません。

構文

```
DBMS_RESUMABLE.SET_SESSION_TIMEOUT (
   session_id  IN NUMBER,
   timeout     IN NUMBER);
```

パラメータ

表 60-4 SET_SESSION_TIMEOUT プロシージャのパラメータ

パラメータ	説明
session_id	再開領域割当てのセッション識別子。
timeout	再開領域割当てのタイムアウト。

GET_TIMEOUT ファンクション

このファンクションは、現行のセッションにおける再開領域割当ての現行のタイムアウト値を戻します。戻される値は秒で示されます。セッションが確立されていない場合、GET_TIMEOUT ファンクションは -1 を戻します。

構文

```
DBMS_RESUMABLE.GET_TIMEOUT;
```

SET_TIMEOUT プロシージャ

このプロシージャは、現行のセッションにおける再開領域割当てのタイムアウトを設定します。戻されるタイムアウトは秒で示されます。新しいタイムアウト設定はセッションに即時適用されます。セッションが確立されていない場合、操作は実行されません。

構文

```
DBMS_RESUMABLE.SET_TIMEOUT (
   timeout  IN NUMBER);
```

パラメータ

表 60-5 SET_TIMEOUT プロシージャのパラメータ

パラメータ	説明
timeout	再開領域割当てのタイムアウト。

SPACE_ERROR_INFO ファンクション

このファンクションは、エラー・スタックで領域関連のエラーを検索します。領域関連のエラーが見つからなかった場合は FALSE が戻されます。見つかった場合は TRUE が戻され、領域エラーの原因となった特定のオブジェクトの情報が戻されます。

構文

```
DBMS_RESUMABLE.SPACE_ERROR_INFO
   error_type        OUT VARCHAR2,
   object_type       OUT VARCHAR2,
   object_owner      OUT VARCHAR2,
   table_space_name  OUT VARCHAR2,
   object_name       OUT VARCHAR2,
   sub_object_name   OUT VARCHAR2)
return boolean;
```

パラメータ

表 60-6 SPACE_ERROR_INFO ファンクションのパラメータ

パラメータ	説明
error_type	領域エラーのタイプ。次のいずれかになります。 ■ NO MORE SPACE ■ MAX EXTENTS REACHED ■ SPACE QUOTA EXCEEDED

表 60-6 SPACE_ERROR_INFO ファンクションのパラメータ（続き）

パラメータ	説明
object_type	オブジェクト・タイプ。次のいずれかになります。 ■ TABLE SPACE ■ ROLLBACK SEGMENT ■ UNDO SEGMENT ■ TABLE ■ INDEX ■ CLUSTER ■ TEMP SEGMENT ■ INDEX PARTITION ■ TABLE PARTITION ■ LOB SEGMENT ■ TABLE SUBPARTITION ■ INDEX SUBPARTITION ■ LOB SUBPARTITION
object_owner	オブジェクトの所有者。特定できない場合は NULL になります。
table_space_name	オブジェクトが常駐する表領域。特定できない場合は NULL になります。
object_name	ロールバック・セグメント、一時セグメント、表、索引またはクラスタの名前。
sub_object_name	LOB、TABLE または INDEX のパーティション名またはサブパーティション名。特定できない場合は NULL になります。

61

DBMS_RLS

DBMS_RLS パッケージには、ファイングレイン・アクセス・コントロールの管理インタフェースが含まれています。DBMS_RLS は、Enterprise Edition でのみ使用できます。

> **関連項目：** DBMS_RLS の詳細な例および使用情報は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』を参照してください。

この章では、次の項目について説明します。

動的な述語

ファイングレイン・アクセス・コントロールをサポートする機能は、動的な述語に基づいています。セキュリティ・ルールはビューに埋め込まれていませんが、ベース表またはビューが DML 文で参照される、文の解析時に取得されます。

表、ビューまたはシノニムに対する動的な述語は PL/SQL ファンクションが生成し、PL/SQL インタフェースを介してセキュリティ・ポリシーに関連付けられます。たとえば、次のようにします。

```
DBMS_RLS.ADD_POLICY (
   'hr', 'employees', 'emp_policy', 'hr', 'emp_sec', 'select');
```

HR スキーマの下にある EMPLOYEES 表が問合せまたは副問合せ（SELECT）で参照されるたびに、サーバーは、EMP_SEC ファンクション（HR スキーマの下にある）をコールします。このファンクションは、EMP_POLICY ポリシーについて、カレント・ユーザーに固有の述語を戻します。ポリシー・ファンクションは、ファンクション・コール時に使用可能なセッション環境変数に基づいて述語を生成できます。これらの変数は通常、アプリケーション・コンテキストのフォームで表示されます。

次に、サーバーは、テキストがある一時ビューを作成します。

```
SELECT * FROM hr.employees WHERE P1
```

ここで、P1（SAL > 10000 や副問合せなど）は、EMP_SEC ファンクションから戻された述語です。サーバーは、EMPLOYEES 表をビューとして処理し、ビューのテキストをデータ・ディクショナリからではなく一時ビューから取得すること以外は、通常のビューと同様にビューの展開を行います。

述語に副問合せがある場合は、ポリシー・ファンクションの所有者（定義者）を使用して副問合せ内のオブジェクトを解決し、そのオブジェクトのセキュリティをチェックします。つまり、ポリシー保護されたオブジェクトへのアクセス権限を持つユーザーには、ポリシーについての知識は不要です。このユーザーには、基礎となるセキュリティ・ポリシーに対するオブジェクト権限の付与は不要です。さらに、サーバーはファンクション定義者の権限でコールを行うため、このユーザーにはポリシー・ファンクションでの EXECUTE 権限は不要です。

注意： 一時ビューは、単一表または述語のみを持つビュー（つまり、JOIN、ORDER BY、GROUP BY などがない）から導出されるため、親オブジェクトの更新可能性を保持できます。

DBMS_RLS は、セキュリティ・ポリシーを削除、使用可能および使用禁止にするためのインタフェースも提供します。たとえば、次の PL/SQL 文を使用して、EMP_POLICY を削除または使用禁止にできます。

```
DBMS_RLS.DROP_POLICY('hr', 'employees', 'emp_policy');
DBMS_RLS.ENABLE_POLICY('hr', 'employees', 'emp_policy', FALSE)
```

セキュリティ

一時ビューが副問合せを使用して作成されると、セキュリティ・チェックが実行されます。ポリシー・ファンクションを持ち、動的な述語を生成するスキーマは、セキュリティ・チェックおよびオブジェクト検索を行う一時ビューの定義者です。

使用上の注意

DBMS_RLS プロシージャは、現行の DML トランザクションがある場合は、操作前にコミットします。ただし、プロシージャが DDL イベント・トリガーの内部にある場合、プロシージャは最初にコミットを実行しません。DDL トランザクションに関して、DBMS_RLS は、DDL トランザクションの一部です。

たとえば、ユーザーは CREATE TABLE のトリガーを作成できます。トリガー内部で、ALTER TABLE を介して列を追加でき、DBMS_RLS を介してポリシーを追加できます。これらすべての操作は、それぞれが DDL 文であっても、CREATE TABLE と同じトランザクション内にあります。CREATE TABLE は、トリガーが正常終了した場合のみ成功します。

現行のカーソルおよび対応する述語のビューは v$vpd_policies から利用できます。

シノニムが参照できるのは、ビューまたは表のみです。

DBMS_RLS サブプログラムの要約

表 61-1 DBMS_RLS のサブプログラム

サブプログラム	説明
「ADD_POLICY プロシージャ」 61-4 ページ	ファイングレイン・アクセス・コントロールのポリシーを表、ビューまたはシノニムに追加します。
「DROP_POLICY プロシージャ」 61-7 ページ	ファイングレイン・アクセス・コントロールのポリシーを表、ビューまたはシノニムから削除します。
「REFRESH_POLICY プロシージャ」 61-8 ページ	ポリシーに関連付けられているすべてのキャッシュ済みの文を再解析します。
「ENABLE_POLICY プロシージャ」 61-9 ページ	ファイングレイン・アクセス・コントロールのポリシーを使用可能または使用禁止にします。
「CREATE_POLICY_GROUP プロシージャ」 61-10 ページ	ポリシー・グループを作成します。
「ADD_GROUPED_POLICY プロシージャ」 61-10 ページ	ポリシー・グループに関連付けられたポリシーを追加します。
「ADD_POLICY_CONTEXT プロシージャ」 61-12 ページ	アクティブなアプリケーションのコンテキストを追加します。

表 61-1 DBMS_RLS のサブプログラム（続き）

サブプログラム	説明
「DELETE_POLICY_GROUP プロシージャ」61-13 ページ	ポリシー・グループを削除します。
「DROP_GROUPED_POLICY プロシージャ」61-14 ページ	ポリシー・グループに関連付けられたポリシーを削除します。
「DROP_POLICY_CONTEXT プロシージャ」61-14 ページ	駆動コンテキストが 1 つ少なくなるように、オブジェクトから駆動コンテキストを削除します。
「ENABLE_GROUPED_POLICY プロシージャ」61-15 ページ	行レベルのグループ・セキュリティ・ポリシーを使用可能または使用禁止にします。
「REFRESH_GROUPED_POLICY プロシージャ」61-16 ページ	リフレッシュ・ポリシーに関連付けられた SQL 文を再解析します。

ADD_POLICY プロシージャ

このプロシージャは、ファイングレイン・アクセス・コントロールのポリシーを表、ビューまたはシノニムに追加します。

トランザクションがある場合、その現行のトランザクションはプロシージャによって操作の実行前にコミットを実行します。ただし、トランザクションが DDL イベント・トリガー内にある場合は、最初にコミットを実行しません。

関連項目： 61-3 ページ「使用上の注意」

コミットは、操作の最後にも実行されます。

構文

```
DBMS_RLS.ADD_POLICY (
   object_schema   IN VARCHAR2 NULL,
   object_name     IN VARCHAR2,
   policy_name     IN VARCHAR2,
   function_schema IN VARCHAR2 NULL,
   policy_function IN VARCHAR2,
   statement_types IN VARCHAR2 NULL,
   update_check    IN BOOLEAN  FALSE,
   enable          IN BOOLEAN  TRUE,
   static_policy   IN BOOLEAN  FALSE);
```

パラメータ

表 61-2 ADD_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ（NULL の場合は現行のデフォルト・スキーマ）。
object_name	ポリシーを追加する表、ビューまたはシノニムの名前。
policy_name	追加するポリシーの名前。この名前は、表またはビュー内に対して一意である必要があります。
function_schema	ポリシー・ファンクションのスキーマ（NULL の場合は現行のデフォルト・スキーマ）。
policy_function	ポリシーの述語を生成するファンクションの名前。ファンクションがパッケージ内で定義されている場合、パッケージ名は必ず存在する必要があります。
statement_types	ポリシーを適用する文の種類。SELECT、INSERT、UPDATE および DELETE を任意に組み合せることができます。デフォルトでは、すべてのタイプが適用されます。
update_check	文タイプ INSERT または UPDATE に対するオプションの引数。デフォルトは FALSE です。update_check を TRUE に設定すると、サーバーは、挿入または更新後の値に対してもポリシーをチェックします。
enable	ポリシーの追加時に、そのポリシーを使用可能にするかどうかを示します。デフォルトは TRUE です。
static_policy	デフォルトは FALSE です。TRUE に設定されている場合、サーバーは、静的ポリシーのポリシー・ファンクションによって、同一の述語文字列が、オブジェクトにアクセスするすべてユーザー（ただし、SYS または EXEMPT ACCESS POLICY 権限を持つ特権ユーザーを除く）に対して生成されるとみなします。

使用上の注意

- サーバーは各カーソルごとにポリシー・ファンクションを 1 回起動します。このため、文の解析と実行に関するパフォーマンスが向上します。ポリシー・ファンクション DETERMINISTIC の宣言は、パフォーマンスに影響を与えません。
- SYS は、セキュリティ・ポリシーの制約を受けません。
- 動的な述語を生成するポリシー・ファンクションは、サーバーがコールします。次の例は、ファンクションのインタフェースを示します。

```
FUNCTION policy_function (object_schema IN VARCHAR2, object_name VARCHAR2)
    RETURN VARCHAR2
--- object_schema is the schema owning the table of view.
--- object_name is the name of table, view, or synonym to which the policy
applies.
```

  ポリシー・ファンクションが戻す述語の最大長は、32K です。

- ポリシー・ファンクションには、WNDS（データベースへの書込み禁止状態）の純正レベルが必要です。

  関連項目： RESTRICT_REFERENCES プラグマの詳細は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』を参照してください。

- 同じオブジェクトにある複数のポリシーから生成された動的な述語は、全述語の論理積（AND 条件）の結合効果があります。
- セキュリティ・チェックおよびオブジェクト検索は、動的な述語の副問合せで、オブジェクトのポリシー・ファンクションの所有者に対して実行されます。
- ファンクションが長さ 0（ゼロ）の述語を戻す場合は、ポリシーについてカレント・ユーザーに適用する制限はないと解釈されます。
- 述語で表の別名が必要な場合（たとえば、親オブジェクトが表タイプの場合）は、表またはビューの名前自体を別名として使用する必要があります。サーバーは、一時ビューを "select c1, c2, ... from tab where <predicate>" のように構成します。
- ファンクションの妥当性チェックは、インストレーションを容易にしたり、インポートまたはエクスポート時に発生するその他の依存する問題を軽減するため、実行時に行われます。

DROP_POLICY プロシージャ

このプロシージャは、ファイングレイン・アクセス・コントロールのポリシーを表、ビューまたはシノニムから削除します。

トランザクションがある場合、その現行のトランザクションはプロシージャによって操作の実行前にコミットを実行します。ただし、トランザクションが DDL イベント・トリガー内にある場合は、最初にコミットを実行しません。

関連項目： 61-3 ページ「使用上の注意」

コミットは、操作の最後にも実行されます。

構文

```
DBMS_RLS.DROP_POLICY (
   object_schema IN VARCHAR2 NULL,
   object_name   IN VARCHAR2,
   policy_name   IN VARCHAR2);
```

パラメータ

表 61-3 DROP_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ（NULL の場合は現行のデフォルト・スキーマ）。
object_name	表、ビューまたはシノニムの名前。
policy_name	表、ビューまたはシノニムから削除するポリシーの名前。

REFRESH_POLICY プロシージャ

このプロシージャは、ポリシーに関連付けられたすべてのキャッシュ済みの文を再解析します。これにより、ポリシーへの最新の変更内容がプロシージャの実行直後に有効になります。

トランザクションがある場合、その現行のトランザクションはプロシージャによって操作の実行前にコミットを実行します。ただし、トランザクションが DDL イベント・トリガー内にある場合は、最初にコミットを実行しません。

関連項目： 61-3 ページ「使用上の注意」

コミットは、操作の最後にも実行されます。

構文

```
DBMS_RLS.REFRESH_POLICY (
   object_schema IN VARCHAR2 NULL,
   object_name   IN VARCHAR2 NULL,
   policy_name   IN VARCHAR2 NULL);
```

パラメータ

表 61-4 REFRESH_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ。
object_name	ポリシーが関連付けられている表、ビューまたはシノニムの名前。
policy_name	リフレッシュするポリシーの名前。

エラー

使用禁止になっているポリシーをリフレッシュしようとすると、プロシージャはエラーを戻します。

ENABLE_POLICY プロシージャ

このプロシージャは、ファイングレイン・アクセス・コントロールのポリシーを使用可能または使用禁止にします。ポリシーは、その作成時に使用可能になっています。

トランザクションがある場合、その現行のトランザクションはプロシージャによって操作の実行前にコミットを実行します。ただし、トランザクションが DDL イベント・トリガー内にある場合は、最初にコミットを実行しません。

関連項目： 61-3 ページ「使用上の注意」

コミットは、操作の最後にも実行されます。

構文

```
DBMS_RLS.ENABLE_POLICY (
   object_schema IN VARCHAR2 NULL,
   object_name   IN VARCHAR2,
   policy_name   IN VARCHAR2,
   enable        IN BOOLEAN);
```

パラメータ

表 61-5 ENABLE_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ（NULL の場合は現行のデフォルト・スキーマ）。
object_name	ポリシーが対応付けられている表、ビューまたはシノニムの名前。
policy_name	使用可能または使用禁止にするポリシーの名前。
enable	ポリシーを使用可能にする場合は TRUE、使用禁止にする場合は FALSE。

CREATE_POLICY_GROUP プロシージャ

このプロシージャはポリシー・グループを作成します。

構文

```
DBMS_RLS.CREATE_POLICY_GROUP (
   object_schema   VARCHAR2,
   object_name     VARCHAR2,
   policy_group    VARCHAR2 );
```

パラメータ

表 61-6 CREATE_POLICY_GROUP プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ。
object_name	ポリシーを追加する表、ビューまたはシノニムの名前。
policy_group	ポリシーが属するポリシー・グループの名前。

使用上の注意

グループは、各表または各ビューで一意である必要があります。

ADD_GROUPED_POLICY プロシージャ

このプロシージャは、ポリシー・グループに関連付けられたポリシーを追加します。

構文

```
DBMS_RLS.ADD_GROUPED_POLICY(
   object_schema   VARCHAR2,
   object_name     VARCHAR2,
   policy_group    VARCHAR2,
   policy_name     VARCHAR2,
   function_schema VARCHAR2,
   policy_function VARCHAR2,
   statement_types VARCHAR2,
   update_check    BOOLEAN,
   enable          BOOLEAN,
   static_policy   BOOLEAN  FALSE );
```

パラメータ

表 61-7 ADD_GROUPED_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ。
object_name	ポリシーを追加する表、ビューまたはシノニムの名前。
policy_group	ポリシーが属するポリシー・グループの名前。
policy_name	ポリシーの名前。同一の表またはビューで一意である必要があります。
function_schema	ポリシー・ファンクションを所有するスキーマ。
policy_function	ポリシーの述語を生成するファンクションの名前。ファンクションがパッケージ内で定義されている場合、パッケージ名は必ず存在する必要があります。
statement_types	ポリシーの適用が可能な文の種類のリスト。SELECT、INSERT、UPDATE または DELETE を任意に組み合せることができます。オプション。
update_check	INSERT 文および UPDATE 文の場合に限り、update_check を TRUE に設定すると、サーバーでポリシーが INSERT 後または UPDATE 後の値に対してチェックされます。
enable	ポリシーの追加時に、そのポリシーを使用可能にするかどうかを示します。デフォルトは TRUE です。
static_policy	デフォルトは FALSE です。TRUE に設定されている場合、サーバーは、静的ポリシーのポリシー・ファンクションによって、同一の述語文字列が、オブジェクトにアクセスするすべてユーザー（ただし、SYS または EXEMPT ACCESS POLICY 権限を持つ特権ユーザーを除く）に対して生成されるとみなします。

使用上の注意

- サーバーは各カーソルごとにポリシー・ファンクションを 1 回起動します。したがって、文の解析および実行に要するパフォーマンスは向上します。ポリシー・ファンクション DETERMINISTIC の宣言は、パフォーマンスには影響しません。
- このプロシージャでは、指定した表、ビューまたはシノニムにポリシーが追加され、指定したポリシー・グループにそのポリシーが関連付けられます。
- ポリシー・グループは、CREATE_POLICY_GROUP インタフェースを使用して作成したものである必要があります。
- ポリシー名は、指定したオブジェクトのポリシー・グループ内で一意である必要があります。

- デフォルトのポリシー・グループ（SYS_DEFAULT）のポリシーは、アクティブなポリシー・グループに関係なく常に実行されます。ただし、ファイングレイン・アクセス・コントロールのポリシーは、EXEMPT ACCESS POLICY システム権限を持つユーザーには適用されません。

ADD_POLICY_CONTEXT プロシージャ

このプロシージャは、アクティブなアプリケーションのコンテキストを追加します。

構文

```
DBMS_RLS.ADD_POLICY_CONTEXT (
   object_schema   VARCHAR2,
   object_name     VARCHAR2,
   namespace       VARCHAR2,
   attribute       VARCHAR2 );
```

パラメータ

表 61-8 ADD_POLICY_CONTEXT プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ
object_name	ポリシーを追加する表、ビューまたはシノニムの名前
namespace	駆動コンテキストのネームスペース
attribute	駆動コンテキストの属性

使用上の注意

次の点に注意してください。

- このプロシージャでは、ポリシーを実施するアプリケーション・コンテキストが示されます。このコンテキストは、実行するアプリケーションを決定します。
- 駆動コンテキストはセッションにもグローバルにもできます。
- 実行時、サーバーはこのコンテキストの値からアクティブなポリシー・グループの名前を取り出します。
- ファイングレイン・アクセス・コントロールのポリシーを持つ各オブジェクトには、それぞれ少なくとも 1 つの駆動コンテキストを定義する必要があります。定義しない場合は、そのオブジェクトのすべてのポリシーが実行されます。

- 複数のコンテキストを同一のオブジェクトに追加すると、複数のポリシー・グループのポリシーが施行されます。
- 駆動コンテキストが NULL の場合は、すべてのポリシー・グループのポリシーが使用されます。
- ポリシーを所有するポリシー・グループが駆動コンテキストである場合、そのポリシー・グループ内の使用可能にされた全ポリシーとともに、SYS_DEFAULT ポリシー・グループの全ポリシーが適用されます。
- ポリシーを access_control_group グループの hr.employees 表に追加するには、次のコマンドを発行します。

```
DBMS_RLS.ADD_GROUPED_POLICY('hr','employees','access_control_
group','policy1','SYS', 'HR.ACCESS');
```

DELETE_POLICY_GROUP プロシージャ

このプロシージャはポリシー・グループを削除します。

構文

```
DBMS_RLS.DELETE_POLICY_GROUP (
  object_schema   VARCHAR2,
  object_name     VARCHAR2,
  policy_group    VARCHAR2 );
```

パラメータ

表 61-9 DELETE_POLICY_GROUP プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ
object_name	ポリシーを追加する表、ビューまたはシノニムの名前
policy_group	ポリシーが属するポリシー・グループの名前

使用上の注意

次の点に注意してください。

- このプロシージャでは、指定した表、ビューまたはシノニムのポリシー・グループが削除されます。
- ポリシー・グループ内にポリシーを残さないでください。

DROP_GROUPED_POLICY プロシージャ

ポリシー・グループに対応付けられたポリシーを削除します。

構文

```
DBMS_RLS.DROP_GROUPED_POLICY (
   object_schema   VARCHAR2,
   object_name     VARCHAR2,
   policy_group    VARCHAR2,
   policy_name     VARCHAR2 );
```

パラメータ

表 61-10 DROP_GROUPED_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ
object_name	ポリシーを削除する表、ビューまたはシノニムの名前
policy_group	ポリシーが属するポリシー・グループの名前
policy_name	ポリシー名

DROP_POLICY_CONTEXT プロシージャ

このプロシージャは、駆動コンテキストが 1 つ少なくなるように、オブジェクトから駆動コンテキストを削除します。

構文

```
DBMS_RLS.DROP_POLICY_CONTEXT (
   object_schema   VARCHAR2,
   object_name     VARCHAR2,
   namespace       VARCHAR2,
   attribute       VARCHAR2 );
```

パラメータ

表 61-11 DROP_POLICY_CONTEXT プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ
object_name	ポリシーを削除する表、ビューまたはシノニムの名前
namespace	駆動コンテキストのネームスペース
attribute	駆動コンテキストの属性

ENABLE_GROUPED_POLICY プロシージャ

このプロシージャは、行レベルのグループ・セキュリティ・ポリシーを使用可能または使用禁止にします。

構文

```
DBMS_RLS.ENABLE_GROUPED_POLICY (
   object_schema   VARCHAR2,
   object_name     VARCHAR2,
   group_name      VARCHAR2,
   policy_name     VARCHAR2,
   enable          BOOLEAN  );
```

パラメータ

表 61-12 ENABLE_GROUPED_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ。
object_name	ポリシーを関連付ける表、ビューまたはシノニムの名前。
group_name	ポリシーのグループの名前。
policy_name	使用可能または使用禁止にするポリシーの名前。
enable	TRUE に設定するとポリシーが使用可能になり、FALSE に設定すると使用禁止になります。

使用上の注意

- トランザクションがある場合、その現行のトランザクションはプロシージャによって操作の実行前にコミットを実行します。
- コミットは、操作の最後に実行されます。
- ポリシーは、その作成時に使用可能になっています。

REFRESH_GROUPED_POLICY プロシージャ

このプロシージャは、リフレッシュ・ポリシーに関連付けられた SQL 文を再解析します。

構文

```
DBMS_RLS.REFRESH_GROUPED_POLICY (
   object_schema   VARCHAR2,
   object_name     VARCHAR2,
   group_name      VARCHAR2,
   policy_name     VARCHAR2 );
```

パラメータ

表 61-13 REFRESH_GROUPED_POLICY プロシージャのパラメータ

パラメータ	説明
object_schema	表、ビューまたはシノニムを含んでいるスキーマ
object_name	ポリシーを関連付ける表、ビューまたはシノニムの名前
group_name	ポリシーのグループの名前
policy_name	ポリシー名

使用上の注意

- このプロシージャは、ポリシーに関連付けられたすべてのキャッシュ済みの文を再解析します。これにより、ポリシーへの最新の変更内容がプロシージャの実行直後に有効になります。
- トランザクションがある場合、その現行のトランザクションはプロシージャによって操作の実行前にコミットを実行します。
- コミットは、操作の最後に実行されます。
- 使用禁止になっているポリシーをリフレッシュしようとすると、プロシージャはエラーを戻します。

62

DBMS_ROWID

DBMS_ROWID パッケージによって、ユーザーは ROWID を作成し、PL/SQL プログラムと SQL 文から ROWID に関する情報を取得できます。64 文字ベースの外部 ROWID を解析するためのコードを記述しないで、データ・ブロック番号、オブジェクト番号および他の ROWID コンポーネントを検索できます。

注意： DBMS_ROWID は、ユニバーサル ROWID（UROWID）とともには使用しません。

この章では、次の項目について説明します。

使用上の注意

このパッケージにあるファンクションの一部は、ROWID などの単一のパラメータを必要とします。このパラメータは、1 つの文字または PL/SQL ROWID で、その内容は必要に応じて制限または拡張されます。

DBMS_ROWID のファンクションとプロシージャは PL/SQL コードからコールでき、そのファンクションは SQL 文で使用することもできます。

注意： ROWID_INFO は 1 つのプロシージャです。このプロシージャは、PL/SQL コードでのみ使用できます。

DBMS_ROWID パッケージのファンクションは、他の組込み式の SQL ファンクションと同じように使用できます。つまり、式が使用できればどこでも使用できます。次の例では、EMP 表にある単一行のブロック番号のみを戻すために、ROWID_BLOCK_NUMBER が使用されています。

```
SELECT dbms_rowid.rowid_block_number(rowid)
   FROM emp
   WHERE ename = 'KING';
```

RESTRICT_REFERENCES プラグマ使用上のトラブルシューティング

「ORA: 452,0, サブプログラム String が対応付けられたプラグマに違反しています。」（String には文字列が入ります）のエラーが発生した場合は、次の理由が考えられます。

- 現行のプロシージャまたはファンクションに問題があるため
- プラグマなしでプロシージャまたはファンクションをコールしたか、または制限が不十分なプラグマでプロシージャまたはファンクションをコールしたため
- パッケージの初期化コードに関連したパッケージ・プロシージャまたはファンクションをコールしたか、またはデフォルト値を設定するパッケージ・プロシージャまたはファンクションをコールしたため

PL/SQL の例

次の例では、EMP 表にある行の ROWID を戻し、DBMS_ROWID パッケージの ROWID_OBJECT ファンクションを使用して、その ROWID からデータ・オブジェクト番号を抽出して表示します。

```
DECLARE
  object_no   INTEGER;
  row_id      ROWID;
  ...
BEGIN
  SELECT ROWID INTO row_id FROM emp
    WHERE empno = 7499;
  object_no := dbms_rowid.rowid_object(row_id);
  dbms_output.put_line('The obj. # is '|| object_no);
  ...
```

要件

このパッケージは、パッケージ所有者（sys）ではなく、コール・ユーザーの権限で実行されます。

ROWID のタイプ

次のタイプがあります。

- RESTRICTED －制限付き ROWID
- EXTENDED －拡張 ROWID

たとえば、次のようにします。

```
rowid_type_restricted constant integer := 0;
rowid_type_extended   constant integer := 1;
```

> **注意：** 拡張 ROWID は、Oracle8*i* 以上でのみ使用されます。

ROWID の検証結果

結果	説明
VALID	有効 ROWID
INVALID	無効 ROWID

たとえば、次のようにします。

```
rowid_is_valid   constant integer := 0;
rowid_is_invalid constant integer := 1;
```

オブジェクト・タイプ

結果	説明
UNDEFINED	（制限付き ROWID に対する）オブジェクト番号が定義されていません。

たとえば、次のようにします。

```
rowid_object_undefined constant integer := 0;
```

ROWID の変換タイプ

結果	説明
INTERNAL	ROWID タイプの列から、またはその列への変換
EXTERNAL	文字列フォーマットから、または文字列フォーマットへの変換

たとえば、次のようにします。

```
rowid_convert_internal constant integer := 0;
rowid_convert_external constant integer := 1;
```

例外

例外	説明
ROWID_INVALID	無効な ROWID 形式。
ROWID_BAD_BLOCK	ブロックがファイルの最後を超えています。

たとえば、次のようにします。

```
ROWID_INVALID exception;
   pragma exception_init(ROWID_INVALID, -1410);

ROWID_BAD_BLOCK exception;
   pragma exception_init(ROWID_BAD_BLOCK, -28516);
```

DBMS_ROWID サブプログラムの要約

表 62-1 DBMS_ROWID のサブプログラム

サブプログラム	説明
「ROWID_CREATE ファンクション」62-6 ページ	ROWID をテスト用に限って作成します。
「ROWID_INFO プロシージャ」62-7 ページ	ROWID のタイプとコンポーネントを戻します。
「ROWID_TYPE ファンクション」62-8 ページ	ROWID のタイプを戻します。0（ゼロ）は制限付き、1 は拡張です。
「ROWID_OBJECT ファンクション」62-9 ページ	拡張 ROWID のオブジェクト番号を戻します。
「ROWID_RELATIVE_FNO ファンクション」62-10 ページ	ROWID のファイル番号を戻します。
「ROWID_BLOCK_NUMBER ファンクション」62-11 ページ	ROWID のブロック番号を戻します。
「ROWID_ROW_NUMBER ファンクション」62-11 ページ	行番号を戻します。
「ROWID_TO_ABSOLUTE_FNO ファンクション」62-12 ページ	特定の表にある行について、ROWID に関連する絶対ファイル番号を戻します。
「ROWID_TO_EXTENDED ファンクション」62-13 ページ	ROWID を制限付き形式から拡張形式に変換します。
「ROWID_TO_RESTRICTED ファンクション」62-15 ページ	拡張 ROWID を制限付き形式に変換します。
「ROWID_VERIFY ファンクション」62-16 ページ	ROWID_TO_EXTENDED ファンクションが ROWID を適切に拡張できるかどうかをチェックします。

ROWID_CREATE ファンクション

このファンクションによって、コンポーネント部分をパラメータとして ROWID を作成します。

Oracle サーバーでは、データベース内のデータを指す有効な ROWID を作成することしかできないため、このファンクションは、ROWID 操作のテストに役立ちます。

構文

```
DBMS_ROWID.ROWID_CREATE (
   rowid_type    IN NUMBER,
   object_number IN NUMBER,
   relative_fno  IN NUMBER,
   block_number  IN NUMBER,
   file_number   IN NUMBER)
  RETURN ROWID;
```

プラグマ

```
pragma RESTRICT_REFERENCES(rowid_create,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 62-2 ROWID_CREATE ファンクションのパラメータ

パラメータ	説明
rowid_type	タイプ（制限付きまたは拡張）。 制限付き ROWID については、rowid_type パラメータを 0（ゼロ）に設定します。拡張 ROWID を作成するには、1 を設定します。 rowid_type を 0（ゼロ）に指定すると、指定された object_number パラメータは無視され、ROWID_CREATE ファンクションは制限付き ROWID を戻します。
object_number	データ・オブジェクト番号（制限付きの場合は ROWID_OBJECT_UNDEFINED）。
relative_fno	相対ファイル番号。
block_number	このファイル内のブロック番号。
file_number	このブロック内のファイル番号。

例

ダミーの拡張 ROWID を作成します。

```
my_rowid := DBMS_ROWID.ROWID_CREATE(1, 9999, 12, 1000, 13);
```

rowid_object ファンクションが戻す値を検索します。

```
obj_number := DBMS_ROWID.ROWID_OBJECT(my_rowid);
```

変数 obj_number には、9999 が入っています。

ROWID_INFO プロシージャ

このプロシージャは、ROWID のタイプ（制限付きまたは拡張）を含めた情報と、ROWID のコンポーネントを戻します。これはプロシージャで、SQL 文では使用できません。

構文

```
DBMS_ROWID.ROWID_INFO (
   rowid_in         IN   ROWID,
   rowid_type       OUT  NUMBER,
   object_number    OUT  NUMBER,
   relative_fno     OUT  NUMBER,
   file_number      OUT  NUMBER,
   row_number       OUT  NUMBER);
```

プラグマ

```
pragma RESTRICT_REFERENCES(rowid_info,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 62-3 ROWID_INFO プロシージャのパラメータ

パラメータ	説明
rowid_in	解釈する ROWID。ROWID が制限付き（0）か、または拡張（1）ROWID かを判別します。
rowid_type	タイプ（制限付きまたは拡張）を戻します。
object_number	データ・オブジェクト番号（制限付きの場合は ROWID_OBJECT_UNDEFINED）を戻します。
relative_fno	相対ファイル番号を戻します。

表 62-3 ROWID_INFO プロシージャのパラメータ（続き）

パラメータ	説明
block_number	このファイル内のブロック番号を戻します。
file_number	このブロック内のファイル番号を戻します。

関連項目： 62-8 ページ「ROWID_TYPE ファンクション」

例

この例では、ROWID_CREATE で作成した ROWID の値を読み込んで戻します。

```
DBMS_ROWID.ROWID_INFO(my_rowid, rid_type, obj_num,
  file_num, block_num, row_num);

DBMS_OUTPUT.PUT_LINE('The type is ' || rid_type);
DBMS_OUTPUT.PUT_LINE('Data object number is ' || obj_num);
-- and so on...
```

ROWID_TYPE ファンクション

このファンクションは、ROWID が制限付き ROWID の場合は 0（ゼロ）、拡張の場合は 1 を戻します。

構文

```
DBMS_ROWID.ROWID_TYPE (
   row_id IN ROWID)
  RETURN NUMBER;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_type,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 62-4 ROWID_TYPE ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

例

```
IF DBMS_ROWID.ROWID_TYPE(my_rowid) = 1 THEN
   my_obj_num := DBMS_ROWID.ROWID_OBJECT(my_rowid);
```

ROWID_OBJECT ファンクション

このファンクションは、拡張 ROWID のデータ・オブジェクト番号を戻します。入力された ROWID が制限付き ROWID の場合は、0（ゼロ）を戻します。

構文

```
DBMS_ROWID.ROWID_OBJECT (
   row_id IN ROWID)
  RETURN NUMBER;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_object,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 62-5 ROWID_OBJECT ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

> **注意：** 制限付き ROWID については、ROWID_OBJECT_UNDEFINED の定数が戻されます。

例

```
SELECT dbms_rowid.rowid_object(ROWID)
   FROM emp
   WHERE empno = 7499;
```

ROWID_RELATIVE_FNO ファンクション

このファンクションは、IN パラメータで指定された ROWID の相対ファイル番号を戻します(ファイル番号は表領域に関連しています)。

構文

```
DBMS_ROWID.ROWID_RELATIVE_FNO (
   row_id IN ROWID)
  RETURN NUMBER;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_relative_fno,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 62-6 ROWID_RELATIVE_FNO ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

例

次の例の PL/SQL コードは、相対ファイル番号を戻します。

```
DECLARE
   file_number    INTEGER;
   rowid_val      ROWID;
BEGIN
   SELECT ROWID INTO rowid_val
     FROM dept
     WHERE loc = 'Boston';
   file_number :=
     dbms_rowid.rowid_relative_fno(rowid_val);
    ...
```

ROWID_BLOCK_NUMBER ファンクション

このファンクションは、入力された ROWID のデータベース・ブロック番号を戻します。

構文

```
DBMS_ROWID.ROWID_BLOCK_NUMBER (
   row_id IN ROWID)
  RETURN NUMBER;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_block_number,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 62-7 ROWID_BLOCK_NUMBER ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

例

次の SQL 文の例では、ROWID からブロック番号を選択し、別の表に挿入します。

```
INSERT INTO T2 (SELECT dbms_rowid.rowid_block_number(ROWID)
   FROM some_table
   WHERE key_value = 42);
```

ROWID_ROW_NUMBER ファンクション

このファンクションは、ROWID IN パラメータから行番号を抽出します。

構文

```
DBMS_ROWID.ROWID_ROW_NUMBER (
   row_id IN ROWID)
  RETURN NUMBER;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_row_number,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 62-8 ROWID_ROW_NUMBER ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID

例

行番号を選択します。

```
SELECT dbms_rowid.rowid_row_number(ROWID)
   FROM emp
   WHERE ename = 'ALLEN';
```

ROWID_TO_ABSOLUTE_FNO ファンクション

このファンクションは、指定のスキーマと表にある行について、ファイル番号が絶対番号である ROWID から、絶対ファイル番号を抽出します。スキーマ名とスキーマ・オブジェクト名（表名など）がこのファンクションの IN パラメータとして提供されます。

構文

```
DBMS_ROWID.ROWID_TO_ABSOLUTE_FNO (
   row_id      IN ROWID,
   schema_name IN VARCHAR2,
   object_name IN VARCHAR2)
  RETURN NUMBER;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_to_absolute_fno,WNDS,WNPS,RNPS);
```

パラメータ

表 62-9 ROWID_TO_ABSOLUTE_FNO ファンクションのパラメータ

パラメータ	説明
row_id	解釈する ROWID
schema_name	表が含まれるスキーマの名前
object_name	表名

例

```
DECLARE
   abs_fno        INTEGER;
   rowid_val      CHAR(18);
   object_name    VARCHAR2(20) := 'EMP';
BEGIN
    SELECT ROWID INTO rowid_val
     FROM emp
     WHERE empno = 9999;
     abs_fno := dbms_rowid.rowid_to_absolute_fno(
     rowid_val, 'SCOTT', object_name);
```

> **注意：** パーティション・オブジェクトの名前には、パーティション名やサブパーティション名ではなく、表名を指定する必要があります。

ROWID_TO_EXTENDED ファンクション

このファンクションは、ユーザーが指定するスキーマや表にある行をアドレス指定している制限付き形式の ROWID を、拡張 ROWID 形式に変換します。このファンクションは、今後、このパッケージから削除されて別の場所に移動する可能性があります。

構文

```
DBMS_ROWID.ROWID_TO_EXTENDED (
   old_rowid       IN ROWID,
   schema_name     IN VARCHAR2,
   object_name     IN VARCHAR2,
   conversion_type IN INTEGER)
  RETURN ROWID;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_to_extended,WNDS,WNPS,RNPS);
```

パラメータ

表 62-10 ROWID_TO_EXTENDED ファンクションのパラメータ

パラメータ	説明
old_rowid	変換する ROWID。
schema_name	表が含まれるスキーマの名前（オプション）。
object_name	表名（オプション）。
conversion_type	rowid_convert_internal/external_convert_external（old_rowid が ROWID タイプの列に格納されていたか、または文字列に格納されていたかによります）。

戻り値

ROWID_TO_EXTENDED は、拡張文字列形式で ROWID を戻します。入力された ROWID が NULL の場合、ファンクションは NULL を戻します。ゼロ値（00000000.0000.0000）の ROWID が提供されると、ゼロ値の制限付き ROWID が戻されます。

例

SCOTT スキーマに RIDS と呼ばれる表があり、その表には、ROWID（制限付き）を保持する列 ROWID_COL と、SCOTT スキーマのその他の表を指す列 TABLE_COL が含まれていると仮定します。次の文を使用して、ROWID を拡張形式に変換できます。

```
UPDATE SCOTT.RIDS
   SET rowid_col =
   dbms_rowid.rowid_to_extended (
      rowid_col, 'SCOTT", TABLE_COL, 0);
```

使用上の注意

スキーマ名とオブジェクト名が IN パラメータとして提供されると、このファンクションは、指定の表における SELECT 認可レベルを検証し、表のデータ・オブジェクト番号を使用して、提供された制限付き ROWID を拡張 ROWID に変換します。ROWID_TO_EXTENDED は値を戻しますが、このファンクションがコールされたとき、または拡張 ROWID が実際に使用されたときに、変換された ROWID が表内の有効な行を実際に参照していることは保証しません。

スキーマ名とオブジェクト名が提供されない場合（NULL として渡された場合）、このファンクションは、提供された制限付き ROWID が指定しているページをフェッチしようとします。この ROWID に格納されているファイル番号は、絶対ファイル番号として処理されます。このため、そのファイルが削除されていたり、その番号が移行前に使用されている場合は問題が生じます。フェッチされたページが有効な表に属している場合、この表のデータ・オブジェクト番号は拡張 ROWID 値への変換時に使用されます。これは非常に効率が悪いため、

ターゲット表が不明な場合に行う最終手段としてのみお薦めします。ユーザーは、変換された値の使用時まで、正しい表名を覚えておく必要があります。

拡張 ROWID 値が提供された場合、入力した拡張 ROWID のデータ・オブジェクト番号は、表名パラメータから計算されたデータ・オブジェクト番号と照合して検証されます。2 つの番号が一致しない場合は、INVALID_ROWID 例外が発生します。一致する場合は、入力した ROWID が戻されます。

> **関連項目：** ROWID_VERIFY ファンクションには、指定の ROWID を拡張形式に変換できるかどうかの判別方法が記述されています

ROWID_TO_RESTRICTED ファンクション

このファンクションは、拡張 ROWID を制限付き ROWID 形式に変換します。

構文

```
DBMS_ROWID.ROWID_TO_RESTRICTED (
   old_rowid       IN ROWID,
   conversion_type IN INTEGER)
  RETURN ROWID;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_to_restricted,WNDS,RNDS,WNPS,RNPS);
```

パラメータ

表 62-11 ROWID_TO_RESTRICTED ファンクションのパラメータ

パラメータ	説明
old_rowid	変換する ROWID
conversion_type	内部または外部ー戻される ROWID の形式 rowid_convert_internal/external_convert_external (戻される ROWID が ROWID タイプの列に格納されるか、または文字列に格納されるかによります)。

ROWID_VERIFY ファンクション

このファンクションは、ROWID を検証します。入力した制限付き ROWID が、入力スキーマ名と表名を指定して、拡張形式に変換できる場合は、0（ゼロ）を戻し、変換できない場合は 1 を戻します。

注意： 次の例に示すように、このファンクションを SQL 文の WHERE 句で使用できます。

構文

```
DBMS_ROWID.ROWID_VERIFY (
   rowid_in        IN ROWID,
   schema_name     IN VARCHAR2,
   object_name     IN VARCHAR2,
   conversion_type IN INTEGER
  RETURN NUMBER;
```

プラグマ

```
   pragma RESTRICT_REFERENCES(rowid_verify,WNDS,WNPS,RNPS);
```

パラメータ

表 62-12 ROWID_VERIFY ファンクションのパラメータ

パラメータ	説明
rowid_in	検証する ROWID
schema_name	表が含まれるスキーマの名前
object_name	表名
conversion_type	rowid_convert_internal/external_convert_external (old_rowid が ROWID タイプの列に格納されていたか、または文字列に格納されていたかによります)。

例

ROWID_TO_EXTENDED ファンクションの例にあるスキーマを考慮しながら、次の文を使用すると、無効な ROWID を変換前に検索できます。これにより、無効を事前に修正できます。

```
SELECT ROWID, rowid_col
   FROM SCOTT.RIDS
   WHERE dbms_rowid.rowid_verify(rowid_col, NULL, NULL, 0) =1;
```

関連項目： 第 98 章「UTL_RAW」、第 99 章「UTL_REF」

63

DBMS_RULE

DBMS_RULE パッケージには、EVALUATE プロシージャが含まれています。

この章では、次の項目について説明します。

注意： PUBLIC には、このパッケージの実行権限が付与されます。

関連項目：

- DBMS_RULE パッケージで使用される各種ルールの詳細は、第 109 章「ルール・タイプ」を参照してください。
- ルールの詳細は、第 64 章「DBMS_RULE_ADM」および『Oracle9*i* Streams』を参照してください。

DBMS_RULE サブプログラムの要約

表 63-1 DBMS_RULE サブプログラム

サブプログラム	説明
「EVALUATE プロシージャ」 63-2 ページ	指定した評価コンテキストを使用している指定ルール・セット内のルールを評価します。

EVALUATE プロシージャ

指定した評価コンテキストを使用している指定ルール・セット内のルールを評価します。

注意： 指定したルールとは異なる評価コンテキストを使用しているルール・セット内のルールは、評価の対象になりません。

ルール・セット内のルールは、table_values、column_values、variable_values および attribute_values に指定されているデータを使用して評価されます。これらの値は、指定した評価コンテキストの表および変数を参照する必要があります。そうでない場合、これらの値は無視されます。

コール元は、stop_on_first_hit を使用して、最初の TRUE ルールまたは最初の MAYBE ルール（TRUE ルールがない場合）の検出直後に評価を停止するかどうかを指定できます。

また、コール元は、simple_rules_only を使用して、迅速に評価できる単純な（SQL を使用しない）ルールのみを評価対象とするかどうかも指定できます。この指定で、評価は迅速化されますが、SQL なしでは評価できないルールは、MAYBE ルールとして戻されます。

部分評価はサポートされています。EVALUATE プロシージャは、一部の表、列、変数または属性に関するデータによってのみコールできます。この場合、ルール内の 1 つ以上の単純な式の値に基づいても TRUE または FALSE の判断ができないかぎり、データ不足が原因で評価できないルールは MAYBE ルールとして戻されます。たとえば、変数「x」の属性「a.b」に 1 の値を指定すると、表「tab」の値を指定しなくても、次のルール条件のルールが TRUE で戻ります。

```
(:x.a.b = 1) or (tab.c > 10)
```

評価の結果には、次の 2 種類があります。

- TRUE ルール。指定したデータに基づいて TRUE に評価されたルールのリストです。これらのルールは、OUT パラメータ true_rules で戻されます。

- MAYBE ルール。次の理由の 1 つのために評価できなかったルールのリストです。
  - ルールが使用不可のデータを参照しているため。たとえば、変数の属性 "x.a.b" は指定されているが、変数「x」、属性「a」または属性「a.b」には値が指定されていない場合がこれに該当します。
  - ルールが、(SQL なしで) 迅速に評価できるほど単純ではなく、simple_rules_only が TRUE に指定されているため。

  MAYBE ルールは、OUT パラメータ maybe_rules で戻されます。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- ルール・セットに対する EXECUTE_ON_RULE_SET 権限があること。
- EXECUTE_ANY_RULE_SET システム権限があること。
- ルール・セットの所有者であること。

注意： ルール・エンジンはアクションを開始しません。戻されたルールとともにアクション・コンテキストが戻る場合がありますが、ルール・エンジンのクライアントは、必要なアクションを開始する必要があります。

関連項目： DBMS_RULE パッケージで使用される各種ルールの詳細は、第 109 章「ルール・タイプ」を参照してください。

構文

```
DBMS_RULE.EVALUATE(
  rule_set_name         IN   VARCHAR2,
  evaluation_context    IN   VARCHAR2,
  event_context         IN   SYS.RE$NV_LIST               DEFAULT NULL,
  table_values          IN   SYS.RE$TABLE_VALUE_LIST      DEFAULT NULL,
  column_values         IN   SYS.RE$COLUMN_VALUE_LIST,
  variable_values       IN   SYS.RE$VARIABLE_VALUE_LIST   DEFAULT NULL,
  attribute_values      IN   SYS.RE$ATTRIBUTE_VALUE_LIST,
  stop_on_first_hit     IN   BOOLEAN                      DEFAULT FALSE,
  simple_rules_only     IN   BOOLEAN                      DEFAULT FALSE,
  true_rules            OUT  SYS.RE$RULE_HIT_LIST,
  maybe_rules           OUT  SYS.RE$RULE_HIT_LIST);
```

注意： このプロシージャはオーバーロードされています。このプロシージャの一方のバージョンには、column_values パラメータと attribute_values パラメータがあり、他方のバージョンにはこれらのパラメータがありません。

パラメータ

表 63-2 EVALUATE プロシージャのパラメータ

パラメータ	説明
rule_set_name	ルール・セット名。[*schema_name.*]*rule_set_name* の形式で指定します。たとえば、hr スキーマの hr_rules という名前のルール・セットにあるルールすべてを評価するには、このパラメータに hr.hr_rules を入力します。スキーマが未指定の場合は、現行のユーザーのスキーマが使用されます。
evaluation_context	評価コンテキスト名。[*schema_name.*]*evaluation_context_name* の形式で指定します。スキーマが未指定の場合は、現行のユーザーの名前が使用されます。 評価されるのは、指定した評価コンテキストを使用しているルールのみです。
event_context	評価の原因となるイベントを識別する名前と値の組合せリスト。
table_values	評価コンテキストの作成時に指定された表の別名を使用した、表の行のデータ。
column_values	表の行の部分データ。値が table_values ですでに指定されている表には、列値を指定しないでください。
variable_values	変数のデータを含むリスト。 明示的な変数値を認識する唯一の方法は、その値をこのリストに指定することです。 暗黙的な変数値がリストに指定されていない場合は、暗黙的な変数の値を取得するためのファンクションが起動されます。暗黙的な変数値がリストに指定されている場合は、この値が使用され、ファンクションは起動されません。
attribute_values	変数の部分データ。値が variable_values ですでに指定されている変数の属性値は指定しないでください。
stop_on_first_hit	TRUE の場合、ルール・エンジンは、TRUE ルールを検出次第、評価を停止します。 TRUE を指定して、TRUE ルールがない場合、ルール・エンジンは、指定データの多くを TRUE に評価できるルールが見つかり次第、評価を停止します。 FALSE の場合、ルール・エンジンは、TRUE ルールの検出後も、ルールの評価を続行します。

表 63-2 EVALUATE プロシージャのパラメータ（続き）

パラメータ	説明
simple_rules_only	TRUE の場合は、迅速に評価できる単純な（SQL を使用しない）ルールのみが、評価の対象となります。 FALSE の場合は、すべてのルールが評価されます。
true_rules	EVALUATE プロシージャの出力を RE$RULE_HIT_LIST タイプの配列で受け取ります。 TRUE に評価されるルールがない場合、true_rules は空です。 少なくとも 1 つのルールが TRUE に評価され、stop_on_first_hit が TRUE の場合、true_rules には、TRUE に評価された 1 つのルールが含まれます。 stop_on_first_hit が FALSE の場合、true_rules には、TRUE に評価されたすべてのルールが含まれます。
maybe_rules	すべてのルールが、追加データの必要もなく、完全に評価できる場合、maybe_rules は空です。 stop_on_first_hit が TRUE の状態で、指定データの多くを TRUE に評価できるルールが少なくとも 1 つあり、TRUE に評価されるルールがない場合、maybe_rules には、TRUE に評価できるルールが 1 つ含まれます。 stop_on_first_hit が FALSE の場合、maybe_rules には、指定データの多くを TRUE に評価できるすべてのルールが含まれます。

64

DBMS_RULE_ADM

DBMS_RULE_ADM パッケージは、ルール、ルール・セットおよびルール評価コンテキストを作成および管理するための管理インタフェースを提供します。

この章では、次の項目について説明します。

注意： PUBLIC には、このパッケージの実行権限が付与されます。

関連項目：

- DBMS_RULE_ADM パッケージで使用される各種ルールの詳細は、第 109 章「ルール・タイプ」を参照してください。
- ルールの詳細は、第 63 章「DBMS_RULE」および『Oracle9*i* Streams』を参照してください。

DBMS_RULE_ADM サブプログラムの要約

表 64-1 DBMS_RULE_ADM サブプログラム

サブプログラム	説明
「ADD_RULE プロシージャ」64-3 ページ	指定のルールを指定のルール・セットに追加します。
「ALTER_RULE プロシージャ」64-5 ページ	指定したルールの 1 つ以上の要素を変更します。
「CREATE_EVALUATION_CONTEXT プロシージャ」64-8 ページ	ルール評価コンテキストを作成します。
「CREATE_RULE プロシージャ」64-10 ページ	ルールを指定の名前で作成します。
「CREATE_RULE_SET プロシージャ」64-12 ページ	ルール・セットを指定の名前で作成します。
「DROP_EVALUATION_CONTEXT プロシージャ」64-13 ページ	指定した名前のルール評価コンテキストを削除します。
「DROP_RULE プロシージャ」64-14 ページ	指定した名前のルールを削除します。
「DROP_RULE_SET プロシージャ」64-15 ページ	指定した名前のルール・セットを削除します。
「GRANT_OBJECT_PRIVILEGE プロシージャ」64-16 ページ	指定オブジェクトに関する指定オブジェクト権限を指定のユーザーまたはロールに付与します。
「GRANT_SYSTEM_PRIVILEGE プロシージャ」64-18 ページ	指定のシステム権限を指定のユーザーまたはロールに付与します。
「REMOVE_RULE プロシージャ」64-20 ページ	指定のルールを指定のルール・セットから削除します。
「REVOKE_OBJECT_PRIVILEGE プロシージャ」64-23 ページ	指定オブジェクトに関する指定オブジェクト権限を指定のユーザーまたはロールから取り消します。
「REVOKE_SYSTEM_PRIVILEGE プロシージャ」64-24 ページ	指定のシステム権限を指定のユーザーまたはロールから取り消します。

注意： 特に指定がないかぎり、すべてのプロシージャがコミットされます。

ADD_RULE プロシージャ

指定のルールを指定のルール・セットに追加します。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- ルール・セットに対する ALTER_ON_RULE_SET 権限があること。
- ALTER_ANY_RULE_SET システム権限があること。
- ルール・セットの所有者であること。

また、ルール・セットの所有者は、次の要件を最低 1 つ満たす必要があります。

- ルール・セットに対する EXECUTE_ON_RULE 権限があること。
- EXECUTE_ANY_RULE システム権限があること。
- ルールの所有者であること。

このプロシージャの実行時に、ルールに評価コンテキストがなく、評価コンテキストが未指定の場合、ルールで使用されるのは、ルール・セットに関連付けられている評価コンテキストです。このような場合は、評価コンテキストを使用するルールがアクセスするベース・オブジェクトすべてについて、ルール所有者に必要な権限が付与されている必要があります。

評価コンテキストが指定されている場合、ルール・セットの所有者は、次の要件を最低 1 つ満たす必要があります。

- 評価コンテキストに関する EXECUTE_ON_EVALUATION_CONTEXT 権限があること。
- EXECUTE_ANY_EVALUATION_CONTEXT システム権限があり、評価コンテキストの所有者が SYS ではないこと。
- 評価コンテキストの所有者であること。

評価コンテキストの所有者がルール所有者と異なる場合、ルール所有者には、評価コンテキストを使用するルールがアクセスするベース・オブジェクトすべてについて、必要な権限が付与されている必要があります。

構文

```
DBMS_RULE_ADM.ADD_RULE(
   rule_name           IN  VARCHAR2,
   rule_set_name       IN  VARCHAR2,
   evaluation_context  IN  VARCHAR2   DEFAULT NULL,
   rule_comment        IN  VARCHAR2   DEFAULT NULL);
```

パラメータ

表 64-2 ADD_RULE プロシージャのパラメータ

パラメータ	説明
rule_name	ルール・セットに追加するルールの名前。[*schema_name*.]*rule_name* の形式で指定します。たとえば、hr スキーマに all_a という名前のルールを追加するには、このパラメータに hr.all_a を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
rule_set_name	ルールを追加するルール・セットの名前。[*schema_name*.]*rule_set_name* の形式で指定します。たとえば、hr スキーマの apply_rules という名前のルール・セットにルールを追加するには、このパラメータに hr.apply_rules を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
evaluation_context	評価コンテキスト名。[*schema_name*.]*evaluation_context_name* の形式で指定します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 評価コンテキストを指定するのは、ルール自体に評価コンテキストがなく、ルール・セットの評価コンテキストをルールに使用しない場合のみです。
rule_comment	ルールの説明（オプション）。ルールをルール・セットに追加する理由などを指定できます。

ALTER_RULE プロシージャ

指定したルールの 1 つ以上の要素を変更します。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- ルール・セットに対する ALTER_ON_RULE 権限があること。
- ALTER_ANY_RULE システム権限があること。
- 変更するルールの所有者であること。

評価コンテキストが指定されている場合、ルール所有者は、次の要件を最低 1 つ満たす必要があります。

- 評価コンテキストに関する EXECUTE_ON_EVALUATION_CONTEXT 権限があること。
- EXECUTE_ANY_EVALUATION_CONTEXT システム権限があり、評価コンテキストの所有者が SYS ではないこと。
- 評価コンテキストの所有者であること。

評価コンテキストの所有者がルール所有者と異なる場合、ルール所有者には、評価コンテキストを使用するルールがアクセスするベース・オブジェクトすべてについて、必要な権限が付与されている必要があります。

> **関連項目：** DBMS_RULE_ADM パッケージで使用される各種ルールの詳細は、第 109 章「ルール・タイプ」を参照してください。

構文

```
DBMS_RULE_ADM.ALTER_RULE(
   rule_name                  IN  VARCHAR2,
   condition                  IN  VARCHAR2        DEFAULT NULL,
   evaluation_context         IN  VARCHAR2        DEFAULT NULL,
   remove_evaluation_context  IN  BOOLEAN         DEFAULT FALSE,
   action_context             IN  SYS.RE$NV_LIST  DEFAULT NULL,
   remove_action_context      IN  BOOLEAN         DEFAULT FALSE,
   rule_comment               IN  VARCHAR2        DEFAULT NULL,
   remove_rule_comment        IN  BOOLEAN         DEFAULT FALSE);
```

パラメータ

表 64-3 ALTER_RULE プロシージャのパラメータ

パラメータ	説明
rule_name	変更するルールの名前。[*schema_name*.]*rule_name* の形式で指定します。たとえば、hr スキーマの all_a という名前のルールを変更するには、このパラメータに hr.all_a を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
condition	ルールに関連付けるブール条件。 NULL 以外の場合は、ルールの条件が変更されます。
evaluation_context	評価コンテキスト名。[*schema_name*.]*evaluation_context_name* の形式で指定します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 NULL 以外の場合は、ルールの評価コンテキストが変更されます。
remove_evaluation_context	TRUE の場合は、ルールの評価コンテキストが NULL に設定され、ルールから評価コンテキストが事実上削除されます。 FALSE の場合は、指定したルールの評価コンテキストが保持されます。 evaluation_context パラメータが NULL でない場合、このパラメータは FALSE に設定してください。
action_context	NULL 以外の場合は、ルールに関連付けられているアクション・コンテキストが変更されます。ルールのアクション・コンテキストは、ルールの評価時に、ルール・エンジンのクライアントが解析するルールに関連付けられている情報です。
remove_action_context	TRUE の場合は、ルールのアクション・コンテキストが NULL に設定され、ルールからアクション・コンテキストが事実上削除されます。 FALSE の場合は、指定したルールのアクション・コンテキストが保持されます。 action_context パラメータが NULL でない場合、このパラメータは FALSE に設定してください。

表 64-3 ALTER_RULE プロシージャのパラメータ（続き）

パラメータ	説明
rule_comment	NULL 以外の場合は、ルールの説明が変更されます。
remove_rule_comment	TRUE の場合は、ルールのコメントが NULL に設定され、ルールからコメントが事実上削除されます。 FALSE の場合は、指定したルールのコメントが保持されます。 rule_comment パラメータが NULL でない場合、このパラメータは FALSE に設定してください。

CREATE_EVALUATION_CONTEXT プロシージャ

ルール評価コンテキストを作成します。ルール評価コンテキストは、ルール条件の中で参照できる外部データを定義します。外部データは、変数または表データの場合があります。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- 作成する評価コンテキストの所有者であり、CREATE_EVALUATION_CONTEXT_OBJ システム権限があること。
- CREATE_ANY_EVALUATION_CONTEXT システム権限があること。

関連項目： DBMS_RULE_ADM パッケージで使用される各種ルールの詳細は、第 109 章「ルール・タイプ」を参照してください。

構文

```
DBMS_RULE_ADM.CREATE_EVALUATION_CONTEXT(
  evaluation_context_name      IN  VARCHAR2,
  table_aliases                IN  SYS.RE$TABLE_ALIAS_LIST    DEFAULT NULL,
  variable_types               IN  SYS.RE$VARIABLE_TYPE_LIST  DEFAULT NULL,
  evaluation_function          IN  VARCHAR2                   DEFAULT NULL,
  evaluation_context_comment   IN  VARCHAR2                   DEFAULT NULL);
```

パラメータ

表 64-4 CREATE_EVALUATION_CONTEXT プロシージャのパラメータ

パラメータ	説明
evaluation_context_name	作成する評価コンテキストの名前。[*schema_name*.]*evaluation_context_name* の形式で指定します。 たとえば、hr スキーマに dept_eval_context という名前の評価コンテキストを作成するには、このパラメータに hr.dept_eval_context を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
table_aliases	評価コンテキストに表を指定する表の別名。表の別名は、ルール条件の表を参照するために使用されます。
variable_types	変数のリスト。評価コンテキストに対する明示的および暗黙的な変数が含まれています。

表 64-4 CREATE_EVALUATION_CONTEXT プロシージャのパラメータ（続き）

パラメータ	説明
evaluation_function	評価コンテキストを使用しているルールの評価でコールされるファンクション（オプション）。 DBMS_RULE.EVALUATE プロシージャと同じ形式が必要です。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 評価ファンクションの詳細は、「使用上の注意」を参照してください。
evaluation_context_comment	ルール評価コンテキストに関する説明（オプション）。

使用上の注意

評価ファンクションには、次の署名が必要です。

```
FUNCTION evaluation_function_name(
  rule_set_name       IN   VARCHAR2,
  evaluation_context  IN   VARCHAR2,
  event_context       IN   SYS.RE$NV_LIST               DEFAULT NULL,
  table_values        IN   SYS.RE$TABLE_VALUE_LIST      DEFAULT NULL,
  column_values       IN   SYS.RE$COLUMN_VALUE_LIST     DEFAULT NULL,
  variable_values     IN   SYS.RE$VARIABLE_VALUE_LIST   DEFAULT NULL,
  attribute_values    IN   SYS.RE$ATTRIBUTE_VALUE_LIST  DEFAULT NULL,
  stop_on_first_hit   IN   BOOLEAN                      DEFAULT FALSE,
  simple_rules_only   IN   BOOLEAN                      DEFAULT FALSE,
  true_rules          OUT  SYS.RE$RULE_HIT_LIST,
  maybe_rules         OUT  SYS.RE$RULE_HIT_LIST);
RETURN BINARY_INTEGER;
```

注意： 各パラメータは必須で、指定のデータ・タイプであることが必要です。ただし、パラメータの名前は変更できます。

ファンクションの戻り値は、次のいずれか 1 つです。

- DBMS_RULE_ADM.EVALUATION_SUCCESS
- DBMS_RULE_ADM.EVALUATION_FAILURE
- DBMS_RULE_ADM.EVALUATION_CONTINUE

CREATE_RULE プロシージャ

ルールを作成します。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- ユーザーは、作成するルールの所有者であり、CREATE_RULE_OBJ システム権限が付与されていること。
- CREATE_ANY_RULE システム権限があること。

評価コンテキストが指定されている場合、ルール所有者は、次の要件を最低 1 つ満たす必要があります。

- 評価コンテキストに関する EXECUTE_ON_EVALUATION_CONTEXT 権限があること。
- EXECUTE_ANY_EVALUATION_CONTEXT システム権限があり、評価コンテキストの所有者が SYS ではないこと。
- 評価コンテキストの所有者であること。

評価コンテキストの所有者がルール所有者と異なる場合、ルール所有者には、評価コンテキストを使用するルールがアクセスするベース・オブジェクトすべてについて、必要な権限が付与されている必要があります。

> **関連項目：** DBMS_RULE_ADM パッケージで使用される各種ルールの詳細は、第 109 章「ルール・タイプ」を参照してください。

構文

```
DBMS_RULE_ADM.CREATE_RULE(
   rule_name           IN  VARCHAR2,
   condition           IN  VARCHAR2,
   evaluation_context  IN  VARCHAR2        DEFAULT NULL,
   action_context      IN  SYS.RE$NV_LIST  DEFAULT NULL,
   rule_comment        IN  VARCHAR2        DEFAULT NULL);
```

パラメータ

表 64-5 CREATE_RULE プロシージャのパラメータ

パラメータ	説明
rule_name	作成するルールの名前。[*schema_name*.]*rule_name* の形式で指定します。たとえば、hr スキーマに all_a という名前のルールを作成するには、このパラメータに hr.all_a を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
condition	ルールに関連付けるブール条件。TRUE または FALSE に評価するブール条件。SELECT 文の WHERE 句で使用可能な条件を指定できます。次の例は、有効なルール条件です。 `department_id = 30` **注意：**条件に「WHERE」という語句は指定しないでください。
evaluation_context	ルールに関連付ける評価コンテキスト名（オプション）。[*schema_name*.]*evaluation_context_name* の形式で指定します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 evaluation_context が未指定の場合は、ルール・セットの評価コンテキストがルールに継承されます。
action_context	ルールに関連付けるアクション・コンテキスト。ルールのアクション・コンテキストは、ルールの評価時に、ルール・エンジンのクライアントが解析するルールに関連付けられている情報です。
rule_comment	ルールに関する説明（オプション）。

CREATE_RULE_SET プロシージャ

ルール・セットを作成します。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- 作成するルール・セットの所有者であり、CREATE_RULE_SET_OBJ システム権限があること。
- CREATE_ANY_RULE_SET システム権限があること。

評価コンテキストが指定されている場合、ルール・セットの所有者は、次の要件を最低 1 つ満たす必要があります。

- 評価コンテキストに関する EXECUTE_ON_EVALUATION_CONTEXT 権限があること。
- EXECUTE_ANY_EVALUATION_CONTEXT システム権限があり、評価コンテキストの所有者が SYS ではないこと。
- 評価コンテキストの所有者であること。

構文

```
DBMS_RULE_ADM.CREATE_RULE_SET(
   rule_set_name       IN  VARCHAR2,
   evaluation_context  IN  VARCHAR2  DEFAULT NULL,
   rule_set_comment    IN  VARCHAR2  DEFAULT NULL);
```

パラメータ

表 64-6 CREATE_RULE_SET プロシージャのパラメータ

パラメータ	説明
rule_set_name	作成するルール・セットの名前。[*schema_name*.]*rule_set_name* の形式で指定します。たとえば、hr スキーマに apply_rules という名前のルール・セットを作成するには、このパラメータに hr.apply_rules を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
evaluation_context	評価コンテキスト名（オプション）。[*schema_name*.]*evaluation_context_name* の形式で指定します。この評価コンテキスト名は、評価コンテキストが明示的に関連付けられていないルール・セットのすべてのルールに適用されます。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
rule_set_comment	ルール・セットに関する説明（オプション）。

DROP_EVALUATION_CONTEXT プロシージャ

ルール評価コンテキストを削除します。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- 評価コンテキストの所有者であること。
- DROP_ANY_EVALUATION_CONTEXT システム権限があること。

構文

```
DBMS_RULE_ADM.DROP_EVALUATION_CONTEXT(
   evaluation_context_name  IN  VARCHAR2,
   force                    IN  BOOLEAN   DEFAULT false);
```

パラメータ

表 64-7 DROP_EVALUATION_CONTEXT プロシージャのパラメータ

パラメータ	説明
evaluation_context_name	削除する評価コンテキストの名前。[*schema_name*.]*evaluation_context_name* の形式で指定します。 たとえば、hr スキーマの dept_eval_context という名前の評価コンテキストを削除するには、このパラメータに hr.dept_eval_context を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
force	TRUE の場合は、ルール評価コンテキストを使用するすべてのルールおよびルール・セットからルール評価コンテキストが削除されます。 FALSE を指定し、ルール評価コンテキストを使用するルールまたはルール・セットがない場合は、ルール評価コンテキストが削除されます。 FALSE を指定し、ルール評価コンテキストを使用するルールまたはルール・セットが 1 つ以上ある場合は、例外が発生します。 **注意：** force を TRUE に設定すると、評価コンテキストが設定されていないルールまたはルール・セットが存在することになります。ルールまたはルール・セットに評価コンテキストが未指定で、ADD_RULE プロシージャによる評価コンテキストの指定がない場合、ルールの評価はできません。

DROP_RULE プロシージャ

ルールを削除します。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- ルールの所有者であること。
- DROP_ANY_RULE システム権限があること。

注意：

- データベースからルールを削除せずに、ルール・セットからルールを削除するには、REMOVE_RULE プロシージャを使用します。
- ルールに関連付けられているルール評価コンテキストは、このプロシージャを実行しても削除されません。

構文

```
DBMS_RULE_ADM.DROP_RULE(
   rule_name  IN  VARCHAR2,
   force      IN  BOOLEAN   DEFAULT false);
```

パラメータ

表 64-8 DROP_RULE プロシージャのパラメータ

パラメータ	説明
rule_name	削除するルールの名前。[*schema_name*.]*rule_name* の形式で指定します。たとえば、hr スキーマの all_a という名前のルールを削除するには、このパラメータに hr.all_a を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
force	TRUE の場合は、このルールが含まれているすべてのルール・セットからルールが削除されます。 FALSE に指定し、このルールが含まれているルール・セットがない場合は、ルールが削除されます。 FALSE に指定し、このルールを含むルール・セットが 1 つ以上ある場合は、例外が発生します。

DROP_RULE_SET プロシージャ

ルール・セットを削除します。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- DROP_ANY_RULE_SET システム権限があること。
- ルール・セットの所有者であること。

> **注意：** ルール・セットに関連付けられているルール評価コンテキストは、このプロシージャを実行しても削除されません。

構文

```
DBMS_RULE_ADM.DROP_RULE_SET(
   rule_set_name    IN  VARCHAR2,
   delete_rules     IN  BOOLEAN   DEFAULT false);
```

パラメータ

表 64-9 DROP_RULE_SET プロシージャのパラメータ

パラメータ	説明
rule_set_name	削除するルール・セットの名前。[*schema_name*.]*rule_set_name* の形式で指定します。たとえば、hr スキーマの apply_rules という名前のルール・セットを削除するには、このパラメータに hr.apply_rules を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
delete_rules	TRUE の場合は、ルール・セット内にあるルールも削除されます。ルール・セット内のルールが別のルール・セットのルールでもある場合、そのルールは削除されません。 FALSE の場合、ルール・セット内のルールは削除されません。

GRANT_OBJECT_PRIVILEGE プロシージャ

指定オブジェクトに関する指定オブジェクト権限を指定のユーザーまたはロールに付与します。オブジェクトを所有しているユーザーには、そのオブジェクトに関するすべての権限および付与オプションが自動的に付与されます。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- 権限を付与するオブジェクトの所有者であること。
- 付与する権限と同じ権限および付与オプションがあること。

さらに、オブジェクトがルール・セットの場合は、ルール・セット内のすべてのルールに対する EXECUTE 権限および付与オプションがあるか、またはルール・セット内のルールを所有していることが必要です。

構文

```
DBMS_RULE_ADM.GRANT_OBJECT_PRIVILEGE(
   privilege     IN  BINARY_INTEGER,
   object_name   IN  VARCHAR2,
   grantee       IN  VARCHAR2,
   grant_option  IN  BOOLEAN   DEFAULT false);
```

パラメータ

表 64-10 GRANT_OBJECT_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	オブジェクトの権限受領者に付与するオブジェクト権限の名前。使用可能なオブジェクト権限は、64-17 ページの「使用上の注意」を参照してください。
object_name	権限受領者に付与するオブジェクトの名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr スキーマの apply_rules という名前のルール・セットに関する権限を付与するには、このパラメータに hr.apply_rules を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。オブジェクトは、既存のルール、ルール・セットまたは評価コンテキストであることが必要です。

表 64-10 GRANT_OBJECT_PRIVILEGE プロシージャのパラメータ（続き）

パラメータ	説明
grantee	権限を付与するユーザーまたはロールの名前。オブジェクトの所有者を指定することはできません。
grant_option	TRUE の場合、指定ユーザーまたは指定の権限を付与されたユーザーは、この権限を他のユーザーに付与できます。 FALSE の場合、指定ユーザーまたは指定の権限を付与されたユーザーは、この権限を他のユーザーに付与できません。

使用上の注意

表 64-11 に、オブジェクト権限を示します。

表 64-11 評価コンテキスト、ルールおよびルール・セットに関するオブジェクト権限

権限	説明
SYS.DBMS_RULE_ADM.ALL_ON_EVALUATION_CONTEXT	他のユーザーのスキーマにある特定の評価コンテキストを変更および実行する権限
SYS.DBMS_RULE_ADM.ALL_ON_RULE	他のユーザーのスキーマにある特定のルールを変更および実行する権限
SYS.DBMS_RULE_ADM.ALL_ON_RULE_SET	他のユーザーのスキーマにある特定のルール・セットを変更および実行する権限
SYS.DBMS_RULE_ADM.ALTER_ON_EVALUATION_CONTEXT	他のユーザーのスキーマにある特定の評価コンテキストを変更する権限
SYS.DBMS_RULE_ADM.ALTER_ON_RULE	他のユーザーのスキーマにある特定のルールを変更する権限
SYS.DBMS_RULE_ADM.ALTER_ON_RULE_SET	他のユーザーのスキーマにある特定のルール・セットを変更する権限
SYS.DBMS_RULE_ADM.EXECUTE_ON_EVALUATION_CONTEXT	他のユーザーのスキーマにある特定の評価コンテキストを実行する権限
SYS.DBMS_RULE_ADM.EXECUTE_ON_RULE	他のユーザーのスキーマにある特定のルールを実行する権限
SYS.DBMS_RULE_ADM.EXECUTE_ON_RULE_SET	他のユーザーのスキーマにある特定のルール・セットを実行する権限

たとえば、strmadmin スキーマの hr_dml という名前のルールを変更する権限を hr ユーザーに付与するには、次のように入力します。

```
BEGIN
  DBMS_RULE_ADM.GRANT_OBJECT_PRIVILEGE(
    privilege    => SYS.DBMS_RULE_ADM.ALTER_ON_RULE,
    object_name  => 'strmadmin.hr_dml',
    grantee      => 'hr',
    grant_option => false);
END;
/
```

GRANT_SYSTEM_PRIVILEGE プロシージャ

指定のシステム権限を指定のユーザーまたはロールに付与します。

構文

```
DBMS_RULE_ADM.GRANT_SYSTEM_PRIVILEGE(
   privilege     IN  BINARY_INTEGER,
   grantee       IN  VARCHAR2,
   grant_option  IN  BOOLEAN   DEFAULT false);
```

パラメータ

表 64-12 GRANT_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	権限受領者に付与するシステム権限の名前。使用可能なシステム権限は、64-19 ページの「使用上の注意」を参照してください。
grantee	権限を付与するユーザーまたはロールの名前。
grant_option	TRUE の場合、指定ユーザーまたは指定の権限を付与されたユーザーは、システム権限を他のユーザーに付与できます。 FALSE の場合、指定ユーザーまたは指定の権限を付与されたユーザーは、システム権限を他のユーザーに付与できません。

使用上の注意

表 64-13 に、システム権限を示します。

表 64-13 評価コンテキスト、ルールおよびルール・セットに関するシステム権限

権限	説明
SYS.DBMS_RULE_ADM.ALTER_ANY_EVALUATION_CONTEXT	任意のユーザーが所有する評価コンテキストを変更する権限
SYS.DBMS_RULE_ADM.ALTER_ANY_RULE	任意のユーザーが所有するルールを変更する権限
SYS.DBMS_RULE_ADM.ALTER_ANY_RULE_SET	任意のユーザーが所有するルール・セットを変更する権限
SYS.DBMS_RULE_ADM.CREATE_ANY_EVALUATION_CONTEXT	新規評価コンテキストを任意のスキーマに作成する権限
SYS.DBMS_RULE_ADM.CREATE_EVALUATION_CONTEXT_OBJ	新規評価コンテキストを権限受領者のスキーマに作成する権限
SYS.DBMS_RULE_ADM.CREATE_ANY_RULE	新規ルールを任意のスキーマに作成する権限
SYS.DBMS_RULE_ADM.CREATE_RULE_OBJ	新規ルールを権限受領者のスキーマに作成する権限
SYS.DBMS_RULE_ADM.CREATE_ANY_RULE_SET	新規ルール・セットを任意のスキーマに作成する権限
SYS.DBMS_RULE_ADM.CREATE_RULE_SET_OBJ	新規ルール・セットを権限受領者のスキーマに作成する権限
SYS.DBMS_RULE_ADM.DROP_ANY_EVALUATION_CONTEXT	任意のスキーマの評価コンテキストを削除する権限
SYS.DBMS_RULE_ADM.DROP_ANY_RULE	任意のスキーマのルールを削除する権限
SYS.DBMS_RULE_ADM.DROP_ANY_RULE_SET	任意のスキーマのルール・セットを削除する権限
SYS.DBMS_RULE_ADM.EXECUTE_ANY_EVALUATION_CONTEXT	任意のユーザーが所有する評価コンテキストを実行する権限
SYS.DBMS_RULE_ADM.EXECUTE_ANY_RULE	任意のユーザーが所有するルールを実行する権限
SYS.DBMS_RULE_ADM.EXECUTE_ANY_RULE_SET	任意のユーザーが所有するルール・セットを実行する権限

たとえば、ルール・セットを任意のスキーマに作成する権限を strmadmin ユーザーに付与するには、次のように入力します。

```
BEGIN
  DBMS_RULE_ADM.GRANT_SYSTEM_PRIVILEGE(
    privilege    => SYS.DBMS_RULE_ADM.CREATE_ANY_RULE_SET,
    grantee      => 'strmadmin',
    grant_option => false);
END;
/
```

注意：「ANY」オブジェクトに関する権限（ALTER_ANY_RULE など）を付与する際に、O7_DICTIONARY_ACCESSIBILITY 初期化パラメータが FALSE に設定されている場合は、SYS スキーマを除くすべてのスキーマで、該当するタイプのオブジェクトへのアクセス権をユーザーに付与してください。デフォルトでは、O7_DICTIONARY_ACCESSIBILITY 初期化パラメータは、FALSE に設定されています。

SYS スキーマのオブジェクトに対してアクセス権を付与する場合は、そのオブジェクトに対するオブジェクト権限を明示的に付与できます。また、O7_DICTIONARY_ACCESSIBILITY 初期化パラメータを TRUE に設定することもできます。この設定によって、「ANY」オブジェクトに対して付与した権限で、SYS も含めた任意のスキーマにアクセスできます。

REMOVE_RULE プロシージャ

指定のルールを指定のルール・セットから削除します。

このプロシージャを実行するユーザーは、次の要件を最低 1 つ満たす必要があります。

- ルール・セットに対する ALTER_ON_RULE_SET 権限があること。
- ALTER_ANY_RULE_SET システム権限があること。
- ルール・セットの所有者であること。

注意： これは、データベースからルールを削除するプロシージャではありません。データベースからルールを削除するには、DROP_RULE プロシージャを使用します。

構文

```
DBMS_RULE_ADM.REMOVE_RULE(
   rule_name                IN VARCHAR2,
   rule_set_name            IN VARCHAR2,
   evaluation_context_name IN VARCHAR2  DEFAULT NULL,
   all_evaluation_contexts IN BOOLEAN   DEFAULT false);
```

パラメータ

表 64-14 REMOVE_RULE プロシージャのパラメータ

パラメータ	説明
rule_name	ルール・セットから削除するルールの名前。[*schema_name*.]*rule_name* の形式で指定します。たとえば、hr スキーマの all_a という名前のルールを削除するには、このパラメータに hr.all_a を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
rule_set_name	ルールを削除するルール・セットの名前。[*schema_name*.]*rule_set_name* の形式で指定します。たとえば、hr スキーマの apply_rules という名前のルール・セットからルールを削除するには、このパラメータに hr.apply_rules を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
evaluation_context_name	削除するルールに関連付けられている評価コンテキストの名前。[*schema_name*.]*evaluation_context_name* の形式で指定します。たとえば、hr スキーマの dept_eval_context という名前の評価コンテキストを指定するには、このパラメータに hr.dept_eval_context を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 削除するルールに、ADD_RULE プロシージャを使用してルールをルール・セットに追加するときに指定した評価コンテキストがある場合は、それと同じ評価コンテキストを指定します。異なる評価コンテキストで同じルールを複数回追加した場合は、削除する評価コンテキストでルールを指定します。ルールに関連付けられていない評価コンテキストを指定すると、エラーが発生します。 ルールをルール・セットに追加するときに評価コンテキストを指定しなかった場合は、NULL を指定します。1 つ以上の評価コンテキストがルールに関連付けられている場合に NULL を指定すると、エラーが発生します。

表 64-14 REMOVE_RULE プロシージャのパラメータ（続き）

パラメータ	説明
all_evaluation_contexts	TRUE の場合は、ルールおよびそのルール関連付けられているすべての評価コンテキストがルール・セットから削除されます。 FALSE の場合は、指定した評価コンテキストを持つルールのみが削除されます。 このパラメータが関連するのは、同じルールを異なる評価コンテキストを持つルール・セットに複数回追加した場合のみです。

REVOKE_OBJECT_PRIVILEGE プロシージャ

指定オブジェクトに関する指定のオブジェクト権限を指定のユーザーまたはロールから取り消します。

構文

```
DBMS_RULE_ADM.REVOKE_OBJECT_PRIVILEGE(
   privilege     IN  BINARY_INTEGER,
   object_name   IN  VARCHAR2,
   revokee       IN  VARCHAR2);
```

パラメータ

表 64-15 REVOKE_OBJECT_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	権限受領者から取り消すオブジェクトに関するオブジェクト権限の名前。オブジェクト権限のリストは、64-16 ページの「GRANT_OBJECT_PRIVILEGE プロシージャ」を参照してください。
object_name	権限受領者から権限を取り消すオブジェクトの名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr スキーマの apply_rules という名前のルール・セットに関するオブジェクト権限を取り消すには、このパラメータに hr.apply_rules を入力します。スキーマが未指定の場合は、現行のユーザーがデフォルトです。オブジェクトは、既存のルール、ルール・セットまたは評価コンテキストであることが必要です。
revokee	権限が取り消されるユーザーまたはロールの名前。オブジェクトの所有者を指定することはできません。

REVOKE_SYSTEM_PRIVILEGE プロシージャ

指定のシステム権限を指定のユーザーまたはロールから取り消します。

構文

```
DBMS_RULE_ADM.REVOKE_SYSTEM_PRIVILEGE(
   privilege   IN  BINARY_INTEGER,
   revokee     IN  VARCHAR2);
```

パラメータ

表 64-16 REVOKEE_SYSTEM_PRIVILEGE プロシージャのパラメータ

パラメータ	説明
privilege	権限受領者から取り消すシステム権限の名前。システム権限のリストは、64-18 ページの「GRANT_SYSTEM_PRIVILEGE プロシージャ」を参照してください。
revokee	権限が取り消されるユーザーまたはロールの名前。

65

DBMS_SESSION

このパッケージは、PL/SQL から SQL ALTER SESSION 文と SET ROLE 文へのアクセスおよび他のセッション情報へのアクセスを提供します。このパッケージを使用すると、作業環境とセキュリティ・レベルを設定できます。

この章では、次の項目について説明します。

要件

このパッケージは、パッケージ所有者 SYS ではなく、コール・ユーザーの権限で実行されます。

DBMS_SESSION サブプログラムの要約

表 65-1 DBMS_SESSION のサブプログラム

サブプログラム	説明
「SET_IDENTIFIER」65-3 ページ	識別子を設定します。
「SET_CONTEXT」65-4 ページ および 65-5 ページ	コンテキストを設定します。
「CLEAR_CONTEXT」65-6 ページ	コンテキストを消去します。
「CLEAR_IDENTIFIER」65-7 ページ	識別子を消去します。
「SET_ROLE プロシージャ」65-7 ページ	ロールを設定します。
「SET_SQL_TRACE プロシージャ」65-8 ページ	トレースをオンまたはオフにします。
「SET_NLS プロシージャ」65-8 ページ	National Language Support（NLS）を設定します。
「CLOSE_DATABASE_LINK プロシージャ」65-9 ページ	データベース・リンクをクローズします。
「RESET_PACKAGE プロシージャ」65-10 ページ	セッション内のすべてのパッケージのインスタンス化の解除をします。
「MODIFY_PACKAGE_STATE プロシージャ」65-11 ページ	セッション内でアクティブな PL/SQL プログラム・ユニットのセッション状態に対してアクションを実行します。
「UNIQUE_SESSION_ID ファンクション」65-15 ページ	データベースに現在接続しているすべてのセッションに対して一意の識別子を戻します。
「IS_ROLE_ENABLED ファンクション」65-15 ページ	指定のロールがセッションで使用可能かどうかを判別します。
「IS_SESSION_ALIVE ファンクション」65-16 ページ	指定されたセッションがアクティブかどうかを判別します。

表 65-1 DBMS_SESSION のサブプログラム（続き）

サブプログラム	説明
「SET_CLOSE_CACHED_OPEN_CURSORS プロシージャ」 65-17 ページ	close_cached_open_cursors をオンまたはオフにします。
「FREE_UNUSED_USER_MEMORY プロシージャ」 65-17 ページ	大量のメモリーが必要な操作を実行した後で未使用のメモリーを再要求できます。
「SET_CONTEXT プロシージャ」 65-20 ページ	コンテキスト属性の値を設定または再設定します。
「LIST_CONTEXT プロシージャ」 65-21 ページ	現行のセッションについて、アクティブなネームスペースとコンテキストのリストを戻します。
「SWITCH_CURRENT_CONSUMER_GROUP プロシージャ」 65-22 ページ	ユーザーの現行のセッションについて、現行のリソース・コンシューマ・グループの変更を容易にします。

SET_IDENTIFIER

このプロシージャでは、セッションのクライアント ID を設定します。

構文

```
DBMS_SESSION.SET_IDENTIFIER (
   client_id VARCHAR2);
```

パラメータ

表 65-2 SET_IDENTIFIER プロシージャのパラメータ

パラメータ	説明
client_id	現行のデータベース・セッションのアプリケーション固有の識別子。

使用上の注意

次の点に注意してください。

- SET_IDENTIFIER は、クライアント識別子付きの現行のセッションを初期化して、関連付けられたグローバル・アプリケーション・コンテキストを識別します。
- client_id は大 / 小文字を区別するため、set_context の client_id パラメータと一致させる必要があります。
- このプロシージャはパブリックで実行可能です。

SET_CONTEXT

このプロシージャはコンテキストを設定します。

構文

```
DBMS_SESSION.SET_CONTEXT (
   namespace VARCHAR2,
   attribute VARCHAR2,
   value     VARCHAR2);
```

パラメータ

表 65-3 SET_CONTEXT プロシージャのパラメータ

パラメータ	説明
namespace	設定するアプリケーション・コンテキストのネームスペース
attribute	設定するアプリケーション・コンテキストの属性
value	設定するアプリケーション・コンテキストの値

使用上の注意

次の点に注意してください。

- このインタフェースは *8i* との互換性を確保するために維持されています。
- ネームスペースがグローバル・コンテキストのものである場合は、username には現行のユーザーの名前が、client_id には現行のセッションの client_id が割り当てられます。設定しない場合は NULL になります。
- このプロシージャは、信頼されたパッケージで直接的または間接的に起動する必要があります。

SET_CONTEXT プロシージャ

このプロシージャはコンテキストを設定します。

構文

```
DBMS_SESSION.SET_CONTEXT (
   namespace VARCHAR2,
   attribute VARCHAR2,
   value     VARCHAR2,
   username  VARCHAR2,
   client_id VARCHAR2 );
```

パラメータ

表 65-4 SET_CONTEXT プロシージャのパラメータ

パラメータ	説明
namespace	設定するアプリケーション・コンテキストのネームスペース
attribute	設定するアプリケーション・コンテキストの属性
value	設定するアプリケーション・コンテキストの値
username	アプリケーション・コンテキストのユーザー名属性
client_id	アプリケーション・コンテキストの client_id 属性（最大 64 バイト）

使用上の注意

次の点に注意してください。

- アプリケーション・コンテキストが設定され、client_id に関連付けられます。
- ユーザー名は有効な SQL 識別子にする必要があります。
- client_id は最大 64 バイトの文字列です。
- client_id は大 / 小文字を区別するため、set_identifier に対する引数と一致させる必要があります。
- 直接か、信頼されたパッケージで間接的に起動する必要があります。
- グローバル・ネームスペースでのみ使用できます。

CLEAR_CONTEXT

構文

```
DBMS_SESSION.CLEAR_CONTEXT
   namespace         VARCHAR2,
   client_identifier VARCHAR2,
   attribute         VARCHAR2);
```

パラメータ

表 65-5 CLEAR_CONTEXT プロシージャのパラメータ

パラメータ	説明
namespace	アプリケーション・コンテキストを消去するネームスペース（必須）。 セッション・ローカル・コンテキストの場合、namespace の指定は必須です。Session Local Context として定義した namespace は、グローバルにアクセスされるコンテキストのみに関連付けられるため、client_identifier はオプションとなります。 グローバルにアクセスされるコンテキストの場合、namespace の指定は必須です。NULL は client_identifier に対する有効な値です。これは、識別子の設定がないセッションでは、SET_CONTEXT を使用して設定された「namespace、attribute、value、username、null」のようなコンテキストを参照できるためです。
client_identifier	グローバル・コンテキストに適用され、その他のタイプのコンテキストではオプションです（最大 64 バイト）。
attribute	消去するネームスペース内の特定の属性（オプション）。デフォルトは NULL です。attribute に NULL を指定すると、該当するネームスペースの「namespace、attribute、value」がセッションから消去されます。attribute を指定しないと、namespace 引数と client_identifier 引数に関するすべてのコンテキスト情報が消去されます。

使用上の注意

このプロシージャは、直接か、信頼されたパッケージで間接的に起動する必要があります。

CLEAR_IDENTIFIER

このプロシージャは、セッションの set_client_id を削除します。

構文

```
DBMS_SESSION.CLEAR_IDENTIFIER();
```

使用上の注意

このプロシージャはパブリックで実行可能です。

SET_ROLE プロシージャ

このプロシージャは、ロールを使用可能または使用禁止にします。このプロシージャは、SET ROLE SQL 文と同じです。

構文

```
DBMS_SESSION.SET_ROLE (
   role_cmd VARCHAR2);
```

パラメータ

表 65-6 SET_ROLE プロシージャのパラメータ

パラメータ	説明
role_cmd	このテキストは "set role" に追加され、SQL として実行されます。

SET_SQL_TRACE プロシージャ

このプロシージャは、トレースをオンまたはオフにします。このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION SET SQL_TRACE ...
```

構文

```
DBMS_SESSION.SET_SQL_TRACE (
   sql_trace boolean);
```

パラメータ

表 65-7 SET_SQL_TRACE プロシージャのパラメータ

パラメータ	説明
sql_trace	TRUE はトレースをオンにし、FALSE はトレースをオフにします。

SET_NLS プロシージャ

このプロシージャは、National Language Support（NLS）を設定します。このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION SET <nls_parameter> = <value>
```

構文

```
DBMS_SESSION.SET_NLS (
   param VARCHAR2,
   value VARCHAR2);
```

パラメータ

表 65-8 SET_NLS プロシージャのパラメータ

パラメータ	説明
param	NLS パラメータ。このパラメータ名は、'NLS' で開始する必要があります。
value	パラメータ値。 パラメータがテキスト・リテラルの場合は、埋込みの一重引用符が必要です。たとえば、次のように指定します。 "set_nls('nls_date_format','"DD-MON-YY"')"

CLOSE_DATABASE_LINK プロシージャ

このプロシージャは、オープンしているデータベース・リンクをクローズします。このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION CLOSE DATABASE LINK <name>
```

構文

```
DBMS_SESSION.CLOSE_DATABASE_LINK (
   dblink VARCHAR2);
```

パラメータ

表 65-9 CLOSE_DATABASE_LINK プロシージャのパラメータ

パラメータ	説明
dblink	クローズするデータベース・リンク名

RESET_PACKAGE プロシージャ

このプロシージャは、このセッションのすべてのパッケージのインスタンス化を解除します。このプロシージャは、すべてのパッケージ状態を解放します。65-11 ページの「MODIFY_PACKAGE_STATE プロシージャ」を参照してください。

実行状態をキャッシュするために使用するメモリーは、セッションで実行された PL/SQL ファンクション、プロシージャおよびパッケージに関連付けられています。

パッケージに関して、このメモリーのコレクションはパッケージ変数の現行の値を保持し、各 PL/SQL プログラムがオープンしたカーソルのキャッシュを制御します。RESET_PACKAGE へのコールは、以前実行した各 PL/SQL プログラムに関連付けられていたメモリーをセッションから解放します。したがって、グローバルなパッケージの現行の値は消去され、キャッシュされたカーソルがクローズします。

RESET_PACKAGE は、セッションで失敗したプログラムを確実に再起動するためにも使用できます。パッケージ変数を含んだプログラムが失敗すると、どの変数を初期化しなおす必要があるかを判別することは困難です。RESET_PACKAGE は、すべてのパッケージ変数が初期値に再設定されることを保証します。

構文

```
DBMS_SESSION.RESET_PACKAGE;
```

使用上の注意

すべての実行 PL/SQL が消費するメモリーは大量になるため、RESET_PACKAGE を使用して、データベース・アプリケーション内のある時点でセッション・メモリー・フットプリントを削減できます。ただし、パッケージ変数値の再設定がアプリケーションに影響を与えないことを確認してください。また、キャッシュしたメモリーとカーソルのないプログラムを後で実行すると、解放されたメモリーとカーソルを再作成する必要があるため、実行速度が遅くなることに留意してください。

RESET_PACKAGE は、メモリー、カーソルおよびパッケージ変数をコール直後に解放しません。

注意： RESET_PACKAGE は、起動した PL/SQL コールの実行が完了した後でのみ、メモリー、カーソルおよびパッケージ変数を解放します。

たとえば、PL/SQL プロシージャ P1 が PL/SQL プロシージャ P2 をコールし、P2 が RESET_PACKAGE をコールしたとします。プロシージャ P1 の実行が完了するまで（PL/SQL コールが終了するまで）、RESET_PACKAGE の処理は行われません。

例

SQL*Plus スクリプトは、グローバル変数を使用する場合もしない場合もある多数の PL/SQL プログラム・ユニットを伴った大きいプログラムを実行します。ただし、実行後はグローバル変数は必要ありません。

```
EXCECUTE large_plsql_program1;
```

キャッシュされた PL/SQL セッション・メモリーを解放します。

```
EXECUTE DBMS_SESSION.RESET_PACKAGE;
```

別の大きいプログラムを実行します。

```
EXECUTE large_plsql_program2;
```

MODIFY_PACKAGE_STATE プロシージャ

このプロシージャは、セッション内でアクティブな PL/SQL プログラム・ユニットのセッション状態に対してアクションを実行します。このプロシージャでは、表 65-10 に示す DBMS_SESSION 定数を使用します。

クライアント側の PL/SQL コードは、リモート・パッケージの変数または定数を参照できないため、定数の値を明示的に使用する必要があります。たとえば、次のコードは、DBMS_SESSION.REINITIALIZE 定数を使用しているため、クライアント側ではコンパイルしません。

```
    DBMS_SESSION.MODIFY_PACKAGE_STATE(DBMS_SESSION.REINITIALIZE);
```

かわりに、クライアント側では次のコードを使用します。これは、引数が明示的に指定されているためです。

```
DBMS_SESSION.MODIFY_PACKAGE_STATE(2) -- compiles on the client
```

DBMS_SESSION.MODIFY_PACKAGE_STATE(DBMS_SESSION.FREE_ALL_RESOURCES) は、DBMS_SESSION.RESET_PACKAGE と同様に動作します。DBMS_SESSION.RESET_PACKAGE ではなく、DBMS_SESSION.MODIFY_PACKAGE_STATE(DBMS_SESSION.FREE_ALL_RESOURCES) を使用してください。

構文

```
DBMS_SESSION.MODIFY_PACKAGE_STATE(
   action_flags IN PLS_INTEGER);
```

定数

各フラグの相違点および DBMS_SESSION.REINITIALIZE が DBMS_SESSION.FREE_ALL_RESOURCES よりも高いパフォーマンスを示す理由は、65-13 ページの「使用上の注意」を参照してください。

表 65-10 MODIFY_PACKAGE_STATE の action_flags 定数

定数	説明
FREE_ALL_RESOURCES	PLS_INTEGER := 1
REINITIALIZE	PLS_INTEGER := 2

パラメータ

表 65-11 MODIFY_PACKAGE_STATE プロシージャのパラメータ

パラメータ	説明
action_flags	PL/SQL プログラム・ユニットで実行するアクションを判別するビット・フラグ。 ■ FREE_ALL_RESOURCES（または 1）－以前に実行した各 PL/SQL プログラムに関連付けられているメモリーすべてをセッションから解放します。グローバルなパッケージの現行の値を消去し、キャッシュされたカーソルをクローズします。その後の使用では、PL/SQL プログラム・ユニットが再インスタンス化され、グローバルなパッケージが再初期化されます。 ■ REINITIALIZE（または 2）－実際に解放して最初から再作成せずに、パッケージを再初期化します。パッケージ・メモリーは再利用されます。

使用上の注意

- FREE_ALL_RESOURCES および REINITIALIZE の場合、再初期化が有効になるのは、現行の起動が行われた PL/SQL コールの実行が終了した後です。
- 再初期化が発生するのは、パッケージが実際に参照された場合のみです。パッケージは、参照された順に再初期化されます。
- REINITIALIZE は、すべてのオープン・カーソルが意味的にはクローズしている点で、FREE_ALL_RESOURCES とは異なります。ただし、カーソル・リソースは実際には解放されていません。このリソースは、PL/SQL のカーソル・キャッシュに戻ります。このカーソル・キャッシュはフラッシュされません。したがって、PL/SQL で頻繁にアクセスされる静的 SQL に対応するカーソルは、PL/SQL カーソル・キャッシュにキャッシュされたままになり、その後使用する PL/SQL 文では、新規カーソルのオープン、解析およびクローズに関わるアプリケーションへのオーバーヘッドがありません。
- グローバルな状態のない PL/SQL モジュール（タイプやストアド・プロシージャなど）のセッション・メモリーは、解放されず、再作成されません。
- FREE_ALL_RESOURCES または REINITIALIZE を使用する場合は、パッケージ変数値のリセットによって、アプリケーションに影響を与えないようにしてください。
- DBMS_SESSION.REINITIALIZE は、実際にすべてのパッケージ状態を解放するとはかぎりません。場合によっては、FREE_ALL_RESOURCES フラグや RESET_PACKAGE プロシージャを使用した場合よりも、はるかに大量のセッション・メモリーがアプリケーションで使用されます。たとえば、DBMS_SESSION.MODIFY_PACKAGE_STATE(DBMS_SESSION.REINITIALIZE) の実行後、アプリケーションが以前に参照したパッケージのほとんどを参照しない場合、それらのパッケージのセッション・メモリーは、セッションの終了まで（または DBMS_SESSION.RESET_PACKAGE がコールされるまで）保持されます。

DBMS_SESSION.MODIFY_PACKAGE_STATE の使用 : 例

ここでは、DBMS_SESSION.MODIFY_PACKAGE_STATE の使用例を示します。グローバルな状態（カーソルは c、変数値は cnt）のパッケージ P を想定します。パッケージの最初の初期化では、パッケージ変数 cnt は 0、カーソル c は CLOSED です。次に、セッションで、cnt の値が 111 に変更され、OPEN 操作もカーソルで実行されます。print_status をコールしてパッケージの状態を表示すると、cnt が 111 で、カーソルが OPEN であることを確認できます。次に、DBMS_SESSION.MODIFY_PACKAGE_STATE をコールします。print_status を使用してパッケージ P の状態を再度出力すると、cnt が再度 0 になり、カーソルが CLOSED されていることを確認できます。DBMS_SESSION.MODIFY_PACKAGE_STATE のコールが行われなかった場合は、2 番目の print_status によって、111 および OPEN が出力されます。

```
create or replace package P is
  cnt    number := 0;
  cursor c is select * from emp;
  procedure print_status;
```

```
end P;
/
show errors;

create or replace package body P is
  procedure print_status is
  begin
    dbms_output.put_line('P.cnt = ' || cnt);
    if c%ISOPEN then
      dbms_output.put_line('P.c is OPEN');
    else
      dbms_output.put_line('P.c is CLOSED');
    end if;
  end;
end P;
/
show errors;

SQL> set serveroutput on;
SQL> begin
  2   P.cnt := 111;
  3   open p.c;
  4   P.print_status;
  5  end;
  6  /
P.cnt = 111
P.c is OPEN

PL/SQL procedure successfully completed.

SQL> begin
  2   dbms_session.modify_package_state(dbms_session.reinitialize);
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL> set serveroutput on;
SQL>
SQL> begin
  2   P.print_status;
  3  end;
  4  /
P.cnt = 0
P.c is CLOSED

PL/SQL procedure successfully completed.
```

UNIQUE_SESSION_ID ファンクション

このファンクションは、データベースに現在接続しているすべてのセッションに対して一意の識別子を戻します。同じセッション中にこのファンクションを複数回コールしても、常に同じ結果が戻されます。

構文

```
DBMS_SESSION.UNIQUE_SESSION_ID
  RETURN VARCHAR2;
```

プラグマ

```
pragma restrict_references(unique_session_id,WNDS,RNDS,WNPS);
```

戻り値

表 65-12 UNIQUE_SESSION_ID ファンクションの戻り値

戻り値	説明
unique_session_id	最大 24 バイトまで戻します。

IS_ROLE_ENABLED ファンクション

このファンクションは、指定のロールがこのセッションで使用可能かどうかを判別します。

構文

```
DBMS_SESSION.IS_ROLE_ENABLED (
   rolename VARCHAR2)
  RETURN BOOLEAN;
```

パラメータ

表 65-13 IS_ROLE_ENABLED ファンクションのパラメータ

パラメータ	説明
rolename	ロール名

戻り値

表 65-14 IS_ROLE_ENABLED ファンクションの戻り値

戻り値	説明
is_role_enabled	ロールが使用可能かどうかによって TRUE または FALSE

IS_SESSION_ALIVE ファンクション

このファンクションは、指定されたセッションがアクティブかどうかを判別します。

構文

```
DBMS_SESSION.IS_SESSION_ALIVE (
   uniqueid VARCHAR2)
  RETURN BOOLEAN;
```

パラメータ

表 65-15 IS_SESSION_ALIVE ファンクションのパラメータ

パラメータ	説明
uniqueid	セッションの一意の ID。これは、UNIQUE_SESSION_ID で戻される ID と同じです。

戻り値

表 65-16 IS_SESSION_ALIVE ファンクションの戻り値

戻り値	説明
is_session_alive	セッションがアクティブかどうかによって TRUE または FALSE

SET_CLOSE_CACHED_OPEN_CURSORS プロシージャ

このプロシージャは、close_cached_open_cursors をオンまたはオフにします。このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION SET CLOSE_CACHED_OPEN_CURSORS ...
```

構文

```
DBMS_SESSION.SET_CLOSE_CACHED_OPEN_CURSORS (
   close_cursors BOOLEAN);
```

パラメータ

表 65-17 SET_CLOSE_CACHED_OPEN_CURSORS プロシージャのパラメータ

パラメータ	説明
close_cursors	TRUE または FALSE

FREE_UNUSED_USER_MEMORY プロシージャ

このプロシージャは、大量のメモリー（100K を超えるメモリー）が必要な操作の実行後、未使用のメモリーを再要求します。

大量のメモリーを使用する操作の例を次に示します。

- sort_area_size 全部を使用し、sort_area_size が数百 KB になる大規模なソート処理
- 大規模な PL/SQL パッケージ、プロシージャまたはファンクションのコンパイル処理
- PL/SQL 索引表内にある数百 KB のデータを格納する処理

V$SESSTAT または V$STATNAME の固定ビューにある統計情報 "session uga memory" と "session pga memory" を追跡調査して、ユーザー・メモリーを監視できます。これらの統計情報の監視では、このプロシージャが解放したメモリー量も表示されます。

> **注意：** このプロシージャは、メモリーが非常に不足している場合のみ使用してください。頻繁に使用せず、慎重に使用してください。

構文

```
DBMS_SESSION.FREE_UNUSED_USER_MEMORY;
```

戻り値

このプロシージャの動作は、クライアントのかわりに稼働しているサーバーの構成によって決まります。

- **専用サーバー**: 使用されていない PGA メモリーとセッション・メモリーをオペレーティング・システムに戻します。セッション・メモリーは、この構成内の PGA から割り当てられます。
- **共有サーバー**: 使用されていないセッション・メモリーを shared_pool に戻します。セッション・メモリーは、この構成内の shared_pool から割り当てられます。

使用上の注意

このプロシージャを使用してメモリーを解放するためには、そのメモリーが使用中でないことが必要です。

ある操作でメモリーを割り当てた後は、同じタイプの操作でのみ、割り当てられたメモリーを再利用できます。たとえば、ソート用にメモリーが割り当てられ、ソート完了後にそのメモリーが使用されなくなると、そのソート用に割り当てられたメモリーは別のソートでのみ再利用できます。ソートとコンパイルについては、操作の完了後にメモリーが使用されなくなると、ユーザーはこのプロシージャをコールして、この未使用メモリーを解放できます。

索引表にはメモリーが暗黙的に割り当てられ、その索引表の要素に割り当てられた値が格納されます。したがって、索引表内の要素が多いほど、RDBMS は多くのメモリーを索引表に割り当てます。索引表に要素がある間は、索引表に関連するメモリーは使用中になります。

索引表の有効範囲によって、メモリーの使用期間が決まります。グローバルに宣言された索引表は、パッケージまたはパッケージ本体で宣言された索引表です。これらの索引表には、セッション・メモリーからメモリーが割り当てられます。グローバルに宣言された索引表について、メモリーは、ユーザーのログイン中（ユーザーのセッションの間）は使用中のままとなり、Oracle から切断した後に解放されます。

ローカルで宣言された索引表は、ファンクション、プロシージャまたは無名ブロック内で宣言された索引表です。このような索引表には、PGA メモリーからメモリーが割り当てられます。ローカルに宣言された索引表については、索引表が宣言されたプロシージャ、ファンクションまたは無名ブロックをユーザーが実行しているかぎり、メモリーは使用中のままになります。プロシージャ、ファンクションまたは無名ブロックの実行が完了すると、そのメモリーはローカルに宣言された他の索引表で使用可能になります（つまり、メモリーは使用中ではなくなります）。

初期化されていない空の索引表を既存の索引表に割り当てることは、索引表とその索引表に関連付けられたメモリーを明示的に再初期化するための 1 つの方法です。この操作を行うと、索引表に関連付けられているメモリーは使用中ではなくなり、このプロシージャをコールして解放できます。この方法は、グローバルに宣言した索引表がユーザー・セッションの期間中に大きくなる可能性があり、ユーザーが索引表の内容を必要としない場合は、特に役立ちます。

索引表の有効範囲に関連するメモリー・ルールは適用されたままです。しかし、この方法とプロシージャによって、ユーザーが介入して、索引表に関連付けられているメモリーを明示的に解放できます。

例

次の PL/SQL は、この方法と FREE_UNUSED_USER_MEMORY プロシージャの使用方法を示します。

```
CREATE PACKAGE foobar
   type number_idx_tbl is table of number indexed by binary_integer;

   store1_table  number_idx_tbl;     --  PL/SQL indexed table
   store2_table  number_idx_tbl;     --  PL/SQL indexed table
   store3_table  number_idx_tbl;     --  PL/SQL indexed table
   ...
END;            --  end of foobar

DECLARE
   ...
   empty_table   number_idx_tbl;     --  uninitialized ("empty") version
BEGIN
   FOR i in 1..1000000 loop
     store1_table(i) := i;           --  load data
   END LOOP;
   ...
   store1_table := empty_table;      --  "truncate" the indexed table
   ...
   -
   dbms_session.free_unused_user_memory;  -- give memory back to system

   store1_table(1) := 100;           --  index tables still declared;
   store2_table(2) := 200;           --  but truncated.
   ...
END;
```

SET_CONTEXT プロシージャ

このプロシージャは、コンテキスト属性の値を設定または再設定します。

構文

```
DBMS_SESSION.SET_CONTEXT (
   namespace VARCHAR2,
   attribute VARCHAR2,
   value     VARCHAR2,
   username  VARCHAR2,
   client_id VARCHAR2);
```

パラメータ

表 65-18 SET_CONTEXT プロシージャのパラメータ

パラメータ	説明
namespace	アプリケーションのコンテキストで使用するネームスペースの名前（最大 30 バイト）
attribute	設定する属性の名前（最大 30 バイト）
value	設定する値（最大 4KB）
username	アプリケーション・コンテキストのユーザー名属性
client_id	現行のデータベース・セッションのアプリケーション固有の識別子

使用上の注意

このファンクションのコール元は、CREATE CONTEXT 文を介してコンテキスト・ネームスペースに関連付けられたプロシージャのコール・スタックに存在している必要があります。コール・スタックのチェックは、データベース管理システムの境界を越えません。

ネームスペースに設定できる属性の数に制限はありません。属性値は、ユーザーが再設定するまでユーザー・セッションの間そのまま残ります。

LIST_CONTEXT プロシージャ

このプロシージャは、現行のセッションに関するアクティブなネームスペースとコンテキストのリストを戻します。

構文

```
DBMS_SESSION.LIST_CONTEXT (
   list OUT AppCtxTabTyp,
   size OUT NUMBER);
```

パラメータ

表 65-19 LIST_CONTEXT プロシージャのパラメータ

パラメータ	説明
list	現行のセッション内のアプリケーション・コンテキスト設定リストを格納するバッファ。

戻り値

表 65-20 LIST_CONTEXT プロシージャの戻り値

戻り値	説明
list	現行のセッションにある設定（ネームスペース、属性、値）リスト。
size	戻されたバッファ内のエントリ数を戻します。

使用上の注意

リスト内のコンテキスト情報は、<namespace> <attribute> <value> の順に表示されます。list は表タイプの変数なので、そのサイズは戻されるリストのサイズに合せて動的に調整されます。

SWITCH_CURRENT_CONSUMER_GROUP プロシージャ

このプロシージャは、ユーザーの現行のセッションでの現行のリソース・コンシューマ・グループを変更します。

ある特定のグループに対して切替え権限がある場合は、コンシューマ・グループを切り替えることができます。コール元が別のプロシージャの場合、ユーザーは、そのプロシージャの所有者が切替え権限を持っているコンシューマ・グループに切り替えることができます。

構文

```
DBMS_SESSION.switch_current_consumer_group (
   new_consumer_group     IN  VARCHAR2,
   old_consumer_group     OUT VARCHAR2,
   initial_group_on_error IN  BOOLEAN);
```

パラメータ

表 65-21 SWITCH_CURRENT_CONSUMER_GROUP プロシージャのパラメータ

パラメータ	説明
new_consumer_group	切り替える先のコンシューマ・グループの名前。
old_consumer_group	切り替える元のコンシューマ・グループの名前。
initial_group_on_error	TRUE の場合は、エラー発生時にコール元の現行のコンシューマ・グループが初期コンシューマ・グループとして設定されます。

戻り値

このプロシージャは、old_consumer_group パラメータにある、ユーザーの古いコンシューマ・グループを出力します。

注意： old_consumer_group で戻された値を使用して、古いコンシューマ・グループに後で切り替えることができます。

例外

表 65-22 SWITCH_CURRENT_CONSUMER_GROUP プロシージャの例外

例外	説明
29368	コンシューマ・グループが存在しません。
1031	権限が不十分です。
29396	OTHER_GROUPS コンシューマ・グループに切り替えることができません。

使用上の注意

プロシージャの所有者には、ユーザーを古いコンシューマ・グループに再切替えするために、ユーザーの古いグループ（old_consumer_group）に対する権限が必要です。例外が 1 つあります。このプロシージャでは、ユーザーを初期のコンシューマ・グループにいつでも切り替えることができます（権限チェックはスキップ）。

initial_group_on_error を TRUE に設定すると、SWITCH_CURRENT_CONSUMER_GROUP プロシージャは、new_consumer_group が指定したグループに現行のセッションを設定できない場合、そのセッションをデフォルト・グループに設定します。現行のコンシューマ・グループが初期のコンシューマ・グループに変更されていても、new_consumer_group へのセッションの移動に関連するエラーは発生します。

例

```
CREATE OR REPLACE PROCEDURE high_priority_task is
   old_group varchar2(30);
   prev_group varchar2(30);
   curr_user varchar2(30);
BEGIN
  -- switch invoker to privileged consumer group. If we fail to do so, an
  -- error will be thrown, but the consumer group will not change
  -- because 'initial_group_on_error' is set to FALSE

  dbms_session.switch_current_consumer_group('tkrogrp1', old_group, FALSE);
  -- set up exception handler (in the event of an error, we do not want to
  -- return to caller while leaving the session still in the privileged
  -- group)

  BEGIN
    -- perform some operations while under privileged group

  EXCEPTION
    WHEN OTHERS THEN
```

```
      -- It is possible that the procedure owner does not have privileges
      -- on old_group. 'initial_group_on_error' is set to TRUE to make sure
      -- that the user is moved out of the privileged group in such a
      -- situation

      dbms_session.switch_current_consumer_group(old_group,prev_group,TRUE);
      RAISE;
    END;

  -- we've succeeded. Now switch to old_group, or if cannot do so, switch
  -- to caller's initial consumer group

  dbms_session.switch_current_consumer_group(old_group,prev_group,TRUE);
END high_priority_task;
/
```

66

DBMS_SHARED_POOL

DBMS_SHARED_POOL は、共有プールへのアクセスを提供します。この共有プールは、カーソルと PL/SQL オブジェクトが格納されている共有メモリー領域です。
DBMS_SHARED_POOL によって、共有プール内のオブジェクト・サイズを表示したり、メモリーの断片化を減らすためにオブジェクトを保存または非保存としてマークすることができます。

この章では、次の項目について説明します。

インストール時の注意

DBMS_SHARED_POOL を作成するには、DBMSPOOL.SQL スクリプトを実行します。DBMSPOOL.SQL の実行後には、自動的に PRVTPOOL.PLB スクリプトが実行されます。これらのスクリプトは、CATPROC.SQL スクリプトでは実行されません。

使用上の注意

ここで提供されるプロシージャは、大規模な PL/SQL オブジェクトのロード時に役立ちます。大規模な PL/SQL オブジェクトのロード時には、大量の小規模オブジェクトを共有プールから期限切れとして削除して空き領域を用意する必要があるため（メモリー断片化のため）、ユーザーの応答時間に影響を与えます。場合によっては、大規模なオブジェクトをロードするためのメモリーが不足している場合があります。

DBMS_SHARED_POOL は、頻繁に実行するトリガーに対しても有効です。コンパイルしたトリガーを共有プールで頻繁に使用する表に保管できます。さらに、DBMS_SHARED_POOL は順序もサポートします。順序番号は、順序が期限切れで共有プールから削除されると失われます。DBMS_SHARED_POOL は、共有プールに順序を保存し、順序番号の損失を防止するのに役立ちます。

DBMS_SHARED_POOL サブプログラムの要約

表 66-1 DBMS_SHARED_POOL サブプログラム

サブプログラム	説明
「SIZES プロシージャ」 66-3 ページ	共有プールにあるオブジェクトで、指定サイズより大きいオブジェクトを表示します。
「KEEP プロシージャ」 66-3 ページ	共有プールにオブジェクトを保存します。
「UNKEEP プロシージャ」 66-5 ページ	指定オブジェクトを解放します。
「ABORTED_REQUEST_THRESHOLD プロシージャ」 66-6 ページ	共有プールについて、異常終了を要求するしきい値を設定します。

SIZES プロシージャ

このプロシージャは、shared_pool にあるオブジェクトが指定したサイズより大きいオブジェクトを表示します。オブジェクト名も表示され、KEEP または UNKEEP いずれかのコールへの引数として使用できます。

構文

```
DBMS_SHARED_POOL.SIZES (
   minsize NUMBER);
```

パラメータ

表 66-2 SIZES プロシージャのパラメータ

パラメータ	説明
minsize	オブジェクトを表示するために、共有プール内で占有する必要があるサイズ（KB 単位）。

使用上の注意

このプロシージャの使用前に、SQL*DBA または SQL*Plus 'SET SERVEROUTPUT ON SIZE XXXXX' コマンドを発行すると、結果が表示されます。

KEEP プロシージャ

このプロシージャは、共有プールにオブジェクトを保存します。オブジェクトが一度共有プールに保存されると、期間切れ削除の対象となりません。このことは、頻繁に使用されるラージ・オブジェクトに役立つ場合があります。ラージ・オブジェクトを共有プールに配置する場合、十分な連続領域を作成するために複数のオブジェクトを削除する必要がある場合があります。

構文

```
DBMS_SHARED_POOL.KEEP (
   name VARCHAR2,
   flag CHAR      DEFAULT 'P');
```

注意： このプロシージャは、自動メカニズムが将来実装されて不要になった場合は、サポートされなくなる可能性があります。

パラメータ

表 66-3 KEEP プロシージャのパラメータ

パラメータ	説明
name	保存するオブジェクトの名前。 この識別子は、アドレスと、v$sqlarea ビューの hash_value 列を連結した値です。これは、SIZES プロシージャで表示されます。 現在、TABLE と VIEW オブジェクトは保存できません。
flag	(オプション) このパラメータを指定しないと、パッケージは、最初のパラメータをパッケージ、プロシージャまたはファンクションの名前とみなして名前を解決します。 入力内容がパッケージ、プロシージャまたはファンクションの名前であることを完全に指定するには、'P' または 'p' を設定します。 入力内容がタイプ名であることを指定するには、'T' または 't' を設定します。 入力内容がトリガー名であることを指定するには、'R' または 'r' を設定します。 入力内容が順序名であることを指定するには、'Q' または 'q' を設定します。 最初の引数がカーソル・アドレスとハッシュ値の場合、パラメータには、'P' か 'p'、'Q' か 'q'、'R' か 'r'、'T' か 't' を除く任意の文字を指定する必要があります。

例外

指定されたオブジェクトが見つからない場合は、例外が発生します。

使用上の注意

オブジェクトは 2 種類あります。

- 名前で指定する PL/SQL オブジェクト、トリガー、順序およびタイプ
- 2 つの番号（共有プール内の位置を示す）で指定する SQL カーソル・オブジェクト

たとえば、次のようにします。

```
DBMS_SHARED_POOL.KEEP('scott.hispackage')
```

この例では、SCOTT が所有するパッケージ HISPACKAGE を保存します。PL/SQL オブジェクトの名前は、SQL のオブジェクト命名ルールに従っています（たとえば、デリミタ付き識別子やマルチバイトの名前などが可能です）。カーソルは、DBMS_SHARED_POOL.KEEP('0034CDFF,20348871') によって保存できます。最初の 8 文字は 16 進数の完全なアドレスである必要があります。

UNKEEP プロシージャ

このプロシージャは、指定のオブジェクトを解放します。

構文

```
DBMS_SHARED_POOL.UNKEEP (
   name VARCHAR2,
   flag CHAR     DEFAULT 'P');
```

> **注意：** このプロシージャは、自動メカニズムが将来実装されて不要になった場合は、サポートされなくなる可能性があります。

パラメータ

表 66-4 UNKEEP プロシージャのパラメータ

パラメータ	説明
name	保存しないオブジェクトの名前。KEEP プロシージャの name パラメータの説明を参照してください。
flag	KEEP プロシージャの flag パラメータの説明を参照してください。

例外

指定されたオブジェクトが見つからない場合は、例外が発生します。

ABORTED_REQUEST_THRESHOLD プロシージャ

このプロシージャは、共有プールについて、異常終了を要求するしきい値を設定します。

構文

```
DBMS_SHARED_POOL.ABORTED_REQUEST_THRESHOLD (
   threshold_size NUMBER);
```

パラメータ

表 66-5 ABORTED_REQUEST_THRESHOLD プロシージャのパラメータ

パラメータ	説明
threshold_size	共有プール内で確保解除されたメモリー（解放されたメモリーではない）を解放しないための要求サイズ（バイト単位）。threshold_size の範囲は 5000 ～ 2GB までです。

例外

しきい値が有効な範囲内にない場合は、例外が発生します。

使用上の注意

通常、要求が空きリストで満たされない場合、RDBMS は、LRU リストからオブジェクトを解放して要求を満たすことができるかを定期的にチェックし、メモリーを再要求しようとします。このステップの完了後、RDBMS は、'ALTER SYSTEM FLUSH SHARED_POOL' とほぼ同等の内容を実行します。

これは、システム上のすべてのユーザーに影響を与えるため、このプロシージャは、その影響を thresh_hold サイズを超える共有メモリーの断片検索に失敗した処理にローカライズします。このユーザーは、LRU リストを検索しなくても、' メモリー不足 ' エラーとなります。

67

DBMS_SPACE

DBMS_SPACE パッケージによって、セグメントの成長と領域要件を分析できます。

この章では、次の項目について説明します。

セキュリティ

このパッケージの実行には、SYS 権限が必要です。

要件

実行権限は、PUBLIC に付与されます。このパッケージのサブプログラムは、コール元のセキュリティ下で実行されます。ユーザーには、オブジェクトに関する ANALYZE 権限が必要です。

DBMS_SPACE サブプログラムの要約

表 67-1 DBMS_SPACE のサブプログラム

サブプログラム	説明
「UNUSED_SPACE プロシージャ」67-2 ページ	オブジェクト（表、索引またはクラスタ）にある未使用領域に関する情報を戻します。
「FREE_BLOCKS プロシージャ」67-4 ページ	オブジェクト（表、索引またはクラスタ）にある空きブロックに関する情報を戻します。
「SPACE_USAGE プロシージャ」67-6 ページ	ビットマップ化セグメントの空きブロックに関する情報を戻します。

UNUSED_SPACE プロシージャ

このプロシージャは、オブジェクト（表、索引またはクラスタ）にある未使用領域に関する情報を戻します。

構文

```
DBMS_SPACE.UNUSED_SPACE (
   segment_owner              IN  VARCHAR2,
   segment_name               IN  VARCHAR2,
   segment_type               IN  VARCHAR2,
   total_blocks               OUT NUMBER,
   total_bytes                OUT NUMBER,
   unused_blocks              OUT NUMBER,
   unused_bytes               OUT NUMBER,
   last_used_extent_file_id   OUT NUMBER,
   last_used_extent_block_id  OUT NUMBER,
   last_used_block            OUT NUMBER,
   partition_name             IN  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 67-2 UNUSED_SPACE プロシージャのパラメータ

パラメータ	説明
segment_owner	分析するセグメントのスキーマ名。
segment_name	分析するセグメントのセグメント名。
segment_type	分析するセグメントのタイプは、次の中から選択します。 TABLE TABLE PARTITION TABLE SUBPARTITION INDEX INDEX PARTITION INDEX SUBPARTITION CLUSTER LOB
total_blocks	セグメント内のブロック合計数を戻します。
total_bytes	セグメント内のブロック合計数をバイト単位で戻します。
unused_blocks	未使用のブロック数を戻します。
unused_bytes	未使用のブロック数をバイト単位で戻します。
last_used_extent_file_id	データを含んだ最新エクステントのファイル ID を戻します。
last_used_extent_block_id	データを含んだ最新エクステントのブロック ID を戻します。
last_used_block	データを含んだエクステント内の最終ブロックを戻します。
partition_name	分析するセグメントのパーティション名。 これは、パーティション表についてのみ使用します。サブパーティションの名前は、パーティションの構成時に使用します。

FREE_BLOCKS プロシージャ

このプロシージャは、オブジェクト（表、索引またはクラスタ）にある空きブロックに関する情報を戻します。ビットマップ化セグメントの空きブロック情報が戻される際の詳細は、「SPACE_USAGE プロシージャ」を参照してください。

構文

```
DBMS_SPACE.FREE_BLOCKS (
   segment_owner     IN  VARCHAR2,
   segment_name      IN  VARCHAR2,
   segment_type      IN  VARCHAR2,
   freelist_group_id IN  NUMBER,
   free_blks         OUT NUMBER,
   scan_limit        IN  NUMBER DEFAULT NULL,
   partition_name    IN  VARCHAR2 DEFAULT NULL);
```

プラグマ

```
pragma restrict_references(free_blocks,WNDS);
```

パラメータ

表 67-3 FREE_BLOCKS プロシージャのパラメータ

パラメータ	説明
segment_owner	分析するセグメントのスキーマ名。
segment_name	分析するセグメントのセグメント名。
segment_type	分析するセグメントのタイプは、次の中から選択します。 TABLE TABLE PARTITION TABLE SUBPARTITION INDEX INDEX PARTITION INDEX SUBPARTITION CLUSTER LOB
freelist_group_id	空きリスト・サイズが計算される空きリスト・グループ（インスタンス）。

表 67-3 FREE_BLOCKS プロシージャのパラメータ（続き）

パラメータ	説明
free_blks	指定されたグループに関する空きブロック数を戻します。
scan_limit	読み込む空きリストのブロックの最大数（オプション）。 空きリストに X 個のブロックがある場合は、X 個の走査制限を使用します。
partition_name	分析するセグメントのパーティション名。 これは、パーティション表についてのみ使用します。サブパーティションの名前は、パーティションの構成時に使用します。

例 1

次の例では、必要なバインド変数を宣言してから実行します。

```
DBMS_SPACE.UNUSED_SPACE('SCOTT', 'EMP', 'TABLE', :total_blocks,
   :total_bytes,:unused_blocks, :unused_bytes, :lastextf,
   :last_extb, :lastusedblock);
```

これにより、SCOTT スキーマの EMP 表に、バインド変数に関する未使用領域の情報が入ります。

例 2

次の例では、4 つの空きリスト・グループを持つ SCOTT スキーマにある CLUS クラスタが使用されます。そして、CLUS の空きリスト・グループ 3 にあるブロック数が戻されます。

```
DBMS_SPACE.FREE_BLOCKS('SCOTT', 'CLUS', 'CLUSTER', 3, :free_blocks);
```

注意： scan_limit が正数でない場合は、エラーが発生します。

SPACE_USAGE プロシージャ

このプロシージャは、セグメント最高水位標のデータ・ブロックのスペース使用を示します。ビットマップ・ブロック、セグメント・ヘッダーおよびエクステント・マップ・ブロックはこのプロシージャの対象外です。このプロシージャは、自動セグメント領域管理で作成された表領域でのみ使用します。

構文

```
DBMS_SPACE.SPACE_USAGE(
   segment_owner IN varchar2,
   segment_name IN varchar2,
   segment_type IN varchar2,
   unformatted_blocks OUT number,
   unformatted_bytes OUT number,
   fs1_blocks OUT number,
   fs1_bytes  OUT number,
   fs2_blocks OUT number,
   fs2_bytes  OUT number,
   fs3_blocks OUT number,
   fs3_bytes  OUT number,
   fs4_blocks OUT number,
   fs4_bytes  OUT number,
   full_blocks OUT number,
   full_bytes OUT number,
   partition_name IN varchar2 DEFAULT NULL);
```

パラメータ

表 67-4 SPACE_USAGE プロシージャのパラメータ

パラメータ	説明
segment_owner	分析するセグメントのスキーマ名
segment_name	分析するセグメントの名前
partition_name	分析するセグメントのパーティション名
segment_type	分析するセグメントのタイプ（TABLE、INDEX または CLUSTER）
unformatted_blocks	未フォーマットのブロックの合計数
unformatted bytes	未フォーマットのバイトの合計数
fs1_blocks	空き領域が最低 0 ～ 25% のブロック数
fs1_bytes	空き領域が最低 0 ～ 25% のバイト数
fs2_blocks	空き領域が最低 25 ～ 50% のブロック数
fs2_bytes	空き領域が最低 25 ～ 50% のバイト数
fs3_blocks	空き領域が最低 50 ～ 75% のブロック数
fs3_bytes	空き領域が最低 50 ～ 75% のバイト数
fs4_blocks	空き領域が最低 75 ～ 100% のブロック数
fs4_bytes	空き領域が最低 75 ～ 100% のバイト数
full_blocks	セグメントがいっぱいのブロックの合計数
full_bytes	セグメントがいっぱいのバイトの合計数

例

```
variable unf number;
variable unfb number;
variable fs1 number;
variable fs1b number;
variable fs2 number;
variable fs2b number;
variable fs3 number;
variable fs3b number;
variable fs4 number;
variable fs4b number;
variable full number;
variable fullb number;

begin
dbms_space.space_usage('U1','T',
                        'TABLE',
                        :unf, :unfb,
                        :fs1, :fs1b,
                        :fs2, :fs2b,
                        :fs3, :fs3b,
                        :fs4, :fs4b,
                        :full, :fullb);
end;
/
print unf ;
print unfb ;
print fs4 ;
print fs4b;
print fs3 ;
print fs3b;
print fs2 ;
print fs2b;
print fs1 ;
print fs1b;
print full;
print fullb;
```

68

DBMS_SPACE_ADMIN

DBMS_SPACE_ADMIN パッケージは、ローカルに管理される表領域に対する機能を提供します。

> **関連項目：** DBMS_SPACE_ADMIN 使用の例および詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。

この章では、次の項目について説明します。

セキュリティ

このパッケージは、SYS 権限で実行されるため、このパッケージを実行する権限を持つユーザーは、ビットマップも操作できます。

SYSTEM 表領域の移行 : 条件

SYSTEM 表領域を移行する前に、読込み / 書込みモードで使用するディクショナリ管理の表領域をローカル管理に移行してください。SYSTEM 表領域を移行した後で、ディクショナリ管理表領域を読込み / 書込みモードに変更することはできません。

関連項目 :

- 『Oracle9*i* データベース管理者ガイド』
- 68-13 ページ「TABLESPACE_MIGRATE_TO_LOCAL プロシージャ」

SYSTEM 表領域を移行する前に、次の条件を満たす必要があります。これらの条件は、コールド・バックアップを除く TABLESPACE_MIGRATE_TO_LOCAL プロシージャによって施行されます。

- データベースに、SYSTEM でないデフォルトの一時表領域があること。
- ディクショナリ管理表領域には、ロールバック・セグメントを保持できないこと。
- ローカル管理表領域には、最低 1 つのオンライン・ロールバック・セグメントがあること。自動 UNDO 管理を使用している場合は、UNDO 表領域がオンラインであること。
- ロールバック・セグメントまたは UNDO 表領域を含む表領域を除くすべての表領域が、読取り専用であること。
- データベースのコールド・バックアップがあること。
- システムが制限モードであること。

DBMS_SPACE_ADMIN の定数

表 68-1 DBMS_SPACE_ADMIN 定数

定数	説明
SEGMENT_VERIFY_ EXTENTS	セグメントが所有する領域の使用状況がビットマップに適切に反映されていることを検証します。
SEGMENT_VERIFY_ EXTENTS_GLOBAL	セグメントが所有する領域の使用状況がビットマップに適切に反映されており、この領域が他のセグメントから要求されていないことを検証します。
SEGMENT_MARK_CORRUPT	一時セグメントを破損としてマークします。これにより、ディクショナリからの排除が容易になります（領域の再生なし）。
SEGMENT_MARK_VALID	破損した一時セグメントを有効としてマークします。これは、セグメントのエクステント・マップまたは他の場所での破損が解決され、セグメントが正常に削除できる場合に便利です。
SEGMENT_DUMP_EXTENT_ MAP	指定のセグメントのエクステント・マップをダンプします。
TABLESPACE_VERIFY_ BITMAP	表領域のビットマップを、その表領域内のセグメントのエクステント・マップを使用して検証し、すべてが一致していることを検証します。
TABLESPACE_EXTENT_ MAKE_FREE	この範囲（エクステント）の領域をビットマップ上で空き領域にします。
TABLESPACE_EXTENT_ MAKE_USED	この範囲（エクステント）の領域をビットマップ上で使用領域にします。

DBMS_SPACE_ADMIN サブプログラムの要約

表 68-2 DBMS_SPACE_ADMIN のサブプログラム

サブプログラム	説明
「SEGMENT_VERIFY プロシージャ」68-5 ページ	セグメントのエクステント・マップの一貫性を検証します。
「SEGMENT_CORRUPT プロシージャ」68-6 ページ	セグメントを破損または有効としてマークし、適切なエラーのリカバリを可能にします。
「SEGMENT_DROP_CORRUPT プロシージャ」68-7 ページ	現在破損としてマークされているセグメントを削除します（領域の再生なし）。
「SEGMENT_DUMP プロシージャ」68-8 ページ	指定セグメントのセグメント・ヘッダーとエクステント・マップをダンプします。
「TABLESPACE_VERIFY プロシージャ」68-9 ページ	表領域内のセグメントについて、ビットマップとエクステント・マップが同期していることを検証します。
「TABLESPACE_FIX_BITMAPS プロシージャ」68-10 ページ	適切な DBA 範囲（エクステント）を、ビットマップ上で空きまたは使用領域としてマークします。
「TABLESPACE_REBUILD_BITMAPS プロシージャ」68-11 ページ	適切なビットマップを再作成します。
「TABLESPACE_REBUILD_QUOTAS プロシージャ」68-12 ページ	指定の表領域について割当て制限を再作成します。
「TABLESPACE_MIGRATE_FROM_LOCAL プロシージャ」68-12 ページ	ローカルに管理される表領域を、ディクショナリ管理の表領域に移行します。
「TABLESPACE_MIGRATE_TO_LOCAL プロシージャ」68-13 ページ	ディクショナリ管理形式からローカル管理形式に、表領域を移行します。
「TABLESPACE_RELOCATE_BITMAPS プロシージャ」68-15 ページ	指定先にビットマップを再割当てします。
「TABLESPACE_FIX_SEGMENT_STATES プロシージャ」68-16 ページ	移行が異常終了した表領域内のセグメントの状態を修正します。

SEGMENT_VERIFY プロシージャ

このプロシージャは、セグメントのエクステント・マップがビットマップと一致していることを検証します。

構文

```
DBMS_SPACE_ADMIN.SEGMENT_VERIFY (
   tablespace_name         IN    VARCHAR2,
   header_relative_file    IN    POSITIVE,
   header_block            IN    POSITIVE,
   verify_option           IN    POSITIVE  DEFAULT SEGMENT_VERIFY_EXTENTS);
```

パラメータ

表 68-3 SEGMENT_VERIFY プロシージャのパラメータ

パラメータ	説明
tablespace_name	セグメントが常駐している表領域名
header_relative_file	セグメント・ヘッダーの相対ファイル番号
header_block	セグメント・ヘッダーのブロック番号
verify_option	チェックの種類：SEGMENT_VERIFY_EXTENTS または SEGMENT_VERIFY_EXTENTS_GLOBAL

使用上の注意

すべての DBA 範囲で誤った領域表現として検出された異常は、DBA 範囲、ビットマップ・ブロック、ビットマップ・ブロック範囲、異常情報としてトレース・ファイルに出力されます。レポートされる問題の種類は、空きとみなされない空き領域、空きとみなされた使用済み領域、および複数のセグメントで使用中とみなされた同一領域です。

例

次の例では、相対ファイル番号 4、ブロック番号 33 のセグメント・ヘッダーを持つセグメントで、エクステント・マップとビットマップが同期していることを検証します。

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_VERIFY('USERS', 4, 33, 1);
```

注意： DBMS_SPACE_ADMIN パッケージのすべての例では、SCOTT.EMP を含んだ表領域 USERS が使用されます。

SEGMENT_CORRUPT プロシージャ

このプロシージャは、セグメントを破損または有効としてマークし、適切なエラーのリカバリを可能にします。SYSTEM 表領域では使用できません。

構文

```
DBMS_SPACE_ADMIN.SEGMENT_CORRUPT (
   tablespace_name         IN    VARCHAR2,
   header_relative_file    IN    POSITIVE,
   header_block            IN    POSITIVE,
   corrupt_option          IN    POSITIVE  DEFAULT SEGMENT_MARK_CORRUPT);
```

パラメータ

表 68-4 SEGMENT_CORRUPT プロシージャのパラメータ

パラメータ	説明
tablespace_name	セグメントが常駐している表領域名
header_relative_file	セグメント・ヘッダーの相対ファイル番号
header_block	セグメント・ヘッダーのブロック番号
corrupt_option	SEGMENT_MARK_CORRUPT（デフォルト）または SEGMENT_MARK_VALID

例

次の例では、セグメントを破損としてマークします。

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_CORRUPT('USERS', 4, 33, 3);
```

次の例では、破損のセグメントを有効としてマークします。

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_CORRUPT('USERS', 4, 33, 4);
```

SEGMENT_DROP_CORRUPT プロシージャ

このプロシージャは、現在破損としてマークされているセグメントを削除します（領域の再生なし）。これを実行するには、セグメントは一時的としてマークされている必要があります。破損のセグメントを一時的としてマークするには、セグメントで DROP コマンドを発行します。

SYSTEM 表領域では使用できません。

セグメント用の領域は解放されないため、TABLESPACE_FIX_BITMAPS プロシージャまたは TABLESPACE_REBUILD_BITMAPS プロシージャを使用して調整する必要があります。

構文

```
DBMS_SPACE_ADMIN.SEGMENT_DROP_CORRUPT (
   tablespace_name         IN    VARCHAR2,
   header_relative_file    IN    POSITIVE,
   header_block            IN    POSITIVE);
```

パラメータ

表 68-5 SEGMENT_DROP_CORRUPT プロシージャのパラメータ

パラメータ	説明
tablespace_name	セグメントが常駐している表領域名
header_relative_file	セグメント・ヘッダーの相対ファイル番号
header_block	セグメント・ヘッダーのブロック番号

例

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_DROP_CORRUPT('USERS', 4, 33);
```

SEGMENT_DUMP プロシージャ

このプロシージャは、指定のセグメントのセグメント・ヘッダーとエクステント・マップのブロックをダンプします。

構文

```
DBMS_SPACE_ADMIN.SEGMENT_DUMP (
   tablespace_name         IN    VARCHAR2,
   header_relative_file    IN    POSITIVE,
   header_block            IN    POSITIVE,
   dump_option             IN    POSITIVE  DEFAULT SEGMENT_DUMP_EXTENT_MAP);
```

パラメータ

表 68-6 SEGMENT_DUMP プロシージャのパラメータ

パラメータ	説明
tablespace_name	セグメントが常駐している表領域名
header_relative_file	セグメント・ヘッダーの相対ファイル番号
header_block	セグメント・ヘッダーのブロック番号
dump_option	SEGMENT_DUMP_EXTENT_MAP

例

```
EXECUTE DBMS_SPACE_ADMIN.SEGMENT_DUMP('USERS', 4, 33);
```

TABLESPACE_VERIFY プロシージャ

このプロシージャは、表領域内のセグメントについて、ビットマップとエクステント・マップが同期していることを検証します。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_VERIFY (
   tablespace_name         IN    VARCHAR2,
   verify_option           IN    POSITIVE DEFAULT TABLESPACE_VERIFY_BITMAP);
```

パラメータ

表 68-7 TABLESPACE_VERIFY プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前
verify_option	TABLESPACE_VERIFY_BITMAP

例

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_VERIFY('USERS');
```

TABLESPACE_FIX_BITMAPS プロシージャ

このプロシージャは、適切な DBA 範囲（エクステント）を、ビットマップ上で空きまたは使用領域としてマークします。SYSTEM 表領域に対しては使用できません。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_FIX_BITMAPS (
   tablespace_name         IN    VARCHAR2,
   dbarange_relative_file  IN    POSITIVE,
   dbarange_begin_block    IN    POSITIVE,
   dbarange_end_block      IN    POSITIVE,
   fix_option              IN    POSITIVE);
```

パラメータ

表 68-8 TABLESPACE_FIX_BITMAPS プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前
dbarange_relative_file	DBA 範囲（エクステント）の相対ファイル番号
dbarange_begin_block	エクステントの開始ブロック番号
dbarange_end_block	エクステントの終了ブロック番号
fix_option	TABLESPACE_EXTENT_MAKE_FREE または TABLESPACE_EXTENT_MAKE_USED

例

次の例では、相対ファイル番号 4 の、ブロック番号 33 からブロック番号 83 までの 50 ブロックを、ビットマップ内のビットを USED としてマークします。

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_FIX_BITMAPS('USERS', 4, 33, 83, 7);
```

または、オプションに 8 を指定して、ビットマップ内 FREE としてビットをマークします。BEGIN と END ブロックは、エクステント境界内にあり、エクステントの倍数である必要があります。そうでない場合は、エラーが発生します。

TABLESPACE_REBUILD_BITMAPS プロシージャ

このプロシージャは、適切なビットマップを再作成します。ビットマップ・ブロックの DBA が指定されていない場合は、指定の表領域のすべてのビットマップが再作成されます。

SYSTEM 表領域に対しては使用できません。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_BITMAPS (
   tablespace_name          IN    VARCHAR2,
   bitmap_relative_file     IN    POSITIVE   DEFAULT NULL,
   bitmap_block             IN    POSITIVE   DEFAULT NULL);
```

パラメータ

表 68-9 TABLESPACE_REBUILD_BITMAPS プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前
bitmap_relative_file	再作成するビットマップ・ブロックの相対ファイル番号
bitmap_block	再作成するビットマップ・ブロックのブロック番号

例

次の例では、USERS 表領域にあるすべてのファイルについて、ビットマップを再作成します。

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_BITMAPS('USERS');
```

注意： すべてのファイルの再作成のみサポートされます。

TABLESPACE_REBUILD_QUOTAS プロシージャ

このプロシージャは、指定の表領域に関する割当て制限を再作成します。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_QUOTAS (
   tablespace_name          IN     VARCHAR2);
```

パラメータ

表 68-10 TABLESPACE_REBUILD_QUOTAS プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前

例

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_REBUILD_QUOTAS('USERS');
```

TABLESPACE_MIGRATE_FROM_LOCAL プロシージャ

このプロシージャは、ローカルに管理される表領域をディクショナリ管理の表領域に移行します。このプロシージャは、SYSTEM 表領域に対しては使用できません。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL (
   tablespace_name          IN     VARCHAR2);
```

パラメータ

表 68-11 TABLESPACE_MIGRATE_FROM_LOCAL プロシージャのパラメータ

パラメータ	説明
tablespace_name	表領域の名前

使用上の注意

表領域はオンラインで保持し、移行中に読込み / 書込みを行う必要があります。一時表領域の移行と SYSTEM 表領域の移行はサポートされていません。

例

```
EXECUTE DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL('USERS');
```

TABLESPACE_MIGRATE_TO_LOCAL プロシージャ

このプロシージャを使用して、ディクショナリ管理形式からローカル管理形式に表領域を移行します。ローカル管理形式に移行した表領域は、ユーザーが管理します。

注意： 満たす必要がある条件を完全に理解していない場合は、SYSTEM 表領域を移行しないでください。68-2 ページの「SYSTEM 表領域の移行 : 条件」を参照してください。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL (
   tablespace_name
   allocation_unit
   relative_fno);
```

パラメータ

表 68-12 TABLESPACE_MIGRATE_TO_LOCAL のパラメータ

パラメータ名	用途	データ・タイプ	パラメータ・タイプ
tablespace_name	移行する表領域の名前	VARCHAR	IN
allocation_unit	表領域の単位サイズ（割当て可能な領域の最小許容チャンク・サイズ）	INTEGER	IN
relative_fno	ビットマップ・ブロックが配置されるファイルの相対ファイル番号（オプション）	INTEGER	IN

使用上の注意

表領域はオンラインで保持し、移行中に読込み / 書込みを行う必要があります。一時表領域は移行できないことに注意してください。

割当て単位はオプションで指定できます。デフォルトは、表領域の全エクステント（使用中または空き）の最大公約数に基づいてシステムが計算します。この数値は、表領域の MINIMUM EXTENT を基にさらに削減されます（MINIMUM EXTENT の指定がない場合は 5）。したがって、計算された値が表領域の MINIMUM EXTENT を超えることはありません。各ファイルの最後にある使用可能エクステントは、GCD 計算には含まれません。指定する単位サイズは、システムが計算した単位サイズの因数にしてください。それ以外の場合はエラー・メッセージが戻ります。

相対ファイル番号のパラメータは、目的のファイルにビットマップを配置するために使用されます。ファイルに領域がなかった場合は、エラーが発行されます。指定するデータ・ファイルは、移行する表領域の一部であることが必要です。データ・ファイルを指定しない場合、初期ビットマップ・ブロックを配置するデータ・ファイルはシステムが選択します。初期ビットマップに必要な領域が見つからなかった場合は、エラーになります。

例

次の例は、最小エクステント・サイズ 1MB で表領域 'TS1' を移行します。

```
execute dbms_space_admin.tablespace_migrate_to_local('TS1', 512, 2);
```

ビットマップは、相対ファイル番号 2 のファイルに配置されます。

TABLESPACE_RELOCATE_BITMAPS プロシージャ

このプロシージャを使用して、ビットマップを指定先に再配置します。表領域のディクショナリ管理からローカル管理形式への移行の結果は、ビットマップ・ブロックを含んだ SPACE HEADER セグメントの作成につながる場合があります。SPACE HEADER セグメントは、ユーザー・データとして取り扱われます。ユーザーがファイルを領域ヘッダー・セグメント以下に明示的にサイズ変更しようとすると、エラーが発行されます。制御情報を別の場所に移動してからファイルのサイズを変更する場合は、tablespace_relocate_bitmaps コマンドを使用します。

このプロシージャは、SYSTEM 表領域では使用できません。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_RELOCATE_BITMAPS (
   tablespace_name
   relative_fno
   block_number )
```

パラメータ

表 68-13 TABLESPACE_RELOCATE_BITMAPS のパラメータ

パラメータ名	用途	データ・タイプ	パラメータ・タイプ
tablespace_name	表領域の名前	VARCHAR	IN
relative_fno	宛先ファイルの相対ファイル番号	NUMBER	IN
block_number	宛先データベースのブロック番号	NUMBER	IN

使用上の注意

ビットマップの再配置中、表領域はオンラインで読込み / 書込みを維持する必要があります。移行したローカルに管理される表領域でのみ実行できます。

例

```
execute  dbms_space_admin.tablespace_relocate_bitmaps('TS1', 3, 4);
```

ビットマップをファイル 3、ブロック 4 に移動します。

> **注意：** ソース・アドレスと宛先アドレスはオーバーラップできません。宛先ブロック番号は単位の境界まで切り下げられます。宛先位置にユーザー・データが存在する場合は、エラーになります。

TABLESPACE_FIX_SEGMENT_STATES プロシージャ

このプロシージャを使用して、移行が異常終了した表領域内のセグメントの状態を修正します。ローカルから、またはローカルへの表領域の移行中、セグメントは転送状態に置かれます。移行が異常終了した場合は、イベント 10906 の設定時点でセグメントの状態が SMON によって訂正されます。セグメントがこのような転送状態にあるデータベースは、ダウングレードできません。このプロシージャは、このようなセグメントの状態を修正するために使用できます。

構文

```
DBMS_SPACE_ADMIN.TABLESPACE_FIX_SEGMENT_STATES (
   tablespace_name);
```

パラメータ

表 68-14 TABLESPACE_FIX_SEGMENT_STATES のパラメータ

パラメータ名	用途	データ・タイプ	パラメータ・タイプ
tablespace_name	セグメントを修正する必要がある表領域の名前	VARCHAR	IN

使用上の注意

このプロシージャがコールされたとき、表領域はオンラインで読込み / 書込みを維持する必要があります。

例

```
execute dbms_space_admin.tablespace_fix_segment_states('TS1');
```

69

DBMS_SQL

Oracle によって、ユーザーは動的 SQL を使用するストアド・プロシージャと無名 PL/SQL ブロックを記述できます。動的 SQL 文は、ユーザーのソース・プログラムに埋め込まれていません。実行時にプログラムに入力されるか、またはプログラムによって作成されるように、文字列で格納されています。これによって、ユーザーは用途の広いプロシージャを作成できます。たとえば、この動的 SQL によって、実行時まで名前がわからない表で動作するプロシージャを作成できます。

また、データ操作言語（DML）文やデータ定義言語（DDL）文はいずれも、DBMS_SQL パッケージを使用して解析できます。したがって、PL/SQL を使用して DDL 文を直接解析できます。たとえば、DBMS_SQL パッケージが提供する PARSE プロシージャを使用することによって、ストアド・プロシージャ内から DROP TABLE 文の入力を選択できるようになりました。

注意： Oracle8*i* は、DBMS_SQL パッケージのかわりにシステム固有の動的 SQL を導入しています。システム固有の動的 SQL を使用して、動的 SQL 文を PL/SQL ブロックに直接設定できます。

ほとんどの場合、DBMS_SQL のかわりにシステム固有の動的 SQL を使用できます。システム固有の動的 SQL は、DBMS_SQL と比べて使用方法が簡単でパフォーマンスが向上します。

関連項目： システム固有の動的 SQL の詳細は、『PL/SQL ユーザーズ・ガイドおよびリファレンス』を参照してください。

DBMS_SQL とシステム固有の動的 SQL の比較は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』を参照してください。

この章では、次の項目について説明します。

- DBMS_SQL の使用方法
- DBMS_SQL の定数、タイプおよび例外
- セキュリティ
- 問合せの処理
- 例
- 更新、挿入および削除の処理
- エラーの位置
- DBMS_SQL サブプログラムの要約

DBMS_SQL の使用方法

ストアド・プロシージャ内から動的 SQL を使用する機能は一般的に、Oracle Call Interface（OCI）の手順に従っています。

関連項目：『Oracle Call Interface プログラマーズ・ガイド』

PL/SQL は、C などの他の一般的なプログラム言語とは、多少異なります。たとえば、ユーザーはアドレス（ポインタとも呼ばれます）を PL/SQL で参照できません。そのため、Oracle Call Interface と DBMS_SQL パッケージの間には、いくつか相違点があります。相違点は、次のとおりです。

- OCI はアドレスによるバインドを使用するのに対して、DBMS_SQL パッケージは値によるバインドを使用します。
- DBMS_SQL では、無名ブロックの OUT パラメータの値を検索するには、VARIABLE_VALUE をコールする必要があります。また、行をフェッチして、行内の列の値を実際にプログラムに取り出した後で、COLUMN_VALUE をコールする必要があります。
- 現行のリリースの DBMS_SQL パッケージは、CANCEL カーソル・プロシージャを提供していません。
- NULL は PL/SQL 変数の値として完全にサポートされているため、インジケータ変数は不要です。

DBMS_SQL パッケージの使用例は、次のとおりです。このコードは、Oracle Call Interface のユーザーにとってはかなり簡潔です。

例

この文のテキストはコンパイル時に判明しているため、この例では、動的 SQL を実際に使用する必要はありません。ここでは、このパッケージの概念をわかりやすく説明します。

DEMO プロシージャは、DEMO の実行時に指定した給与よりも高い給与のすべての従業員を EMP 表から削除します。

```
CREATE OR REPLACE PROCEDURE demo(salary IN NUMBER) AS
    cursor_name INTEGER;
    rows_processed INTEGER;
BEGIN
    cursor_name := dbms_sql.open_cursor;
    DBMS_SQL.PARSE(cursor_name, 'DELETE FROM emp WHERE sal > :x',
                   dbms_sql.native);
    DBMS_SQL.BIND_VARIABLE(cursor_name, ':x', salary);
    rows_processed := dbms_sql.execute(cursor_name);
    DBMS_SQL.close_cursor(cursor_name);
EXCEPTION
WHEN OTHERS THEN
    DBMS_SQL.CLOSE_CURSOR(cursor_name);
END;
```

DBMS_SQL の定数、タイプおよび例外

定数

```
v6 constant INTEGER     := 0;
native constant INTEGER := 1;
v7 constant INTEGER     := 2;
```

タイプ

```
TYPE varchar2s IS TABLE OF VARCHAR2(256) INDEX BY BINARY_INTEGER;
TYPE desc_rec  IS RECORD (
      col_type            BINARY_INTEGER := 0,
      col_max_len         BINARY_INTEGER := 0,
      col_name            VARCHAR2(32)   := '',
      col_name_len        BINARY_INTEGER := 0,
      col_schema_name     VARCHAR2(32)   := '',
      col_schema_name_len BINARY_INTEGER := 0,
      col_precision       BINARY_INTEGER := 0,
      col_scale           BINARY_INTEGER := 0,
      col_charsetid       BINARY_INTEGER := 0,
      col_charsetform     BINARY_INTEGER := 0,
      col_null_ok         BOOLEAN        := TRUE);
TYPE desc_tab IS TABLE OF desc_rec INDEX BY BINARY_INTEGER;
```

バルク SQL タイプ

```
type Number_Table   IS TABLE OF NUMBER         INDEX BY BINARY_INTEGER;
type Varchar2_Table IS TABLE OF VARCHAR2(2000) INDEX BY BINARY_INTEGER;
type Date_Table     IS TABLE OF DATE           INDEX BY BINARY_INTEGER;
type Blob_Table     IS TABLE OF BLOB           INDEX BY BINARY_INTEGER;
type Clob_Table     IS TABLE OF CLOB           INDEX BY BINARY_INTEGER;
type Bfile_Table    IS TABLE OF BFILE          INDEX BY BINARY_INTEGER;
type Urowid_Table   IS TABLE OF UROWID         INDEX BY BINARY_INTEGER;
```

例外

```
inconsistent_type exception;
  pragma exception_init(inconsistent_type, -6562);
```

この例外は、指定した OUT パラメータ（要求した値を設定するパラメータ）のタイプがその値のタイプと異なる場合、プロシージャ COLUMN_VALUE または VARIABLE_VALUE で発生します。

実行フロー

OPEN_CURSOR

SQL 文を処理するためには、オープン・カーソルが必要です。OPEN_CURSOR ファンクションをコールすると、ユーザーは Oracle が保持している有効なカーソルを表すデータ構造のカーソル ID 番号を受け取ります。これらのカーソルは、プリコンパイラ、OCI または PL/SQL レベルで定義されたカーソルとは異なり、DBMS_SQL パッケージでのみ使用されます。

PARSE

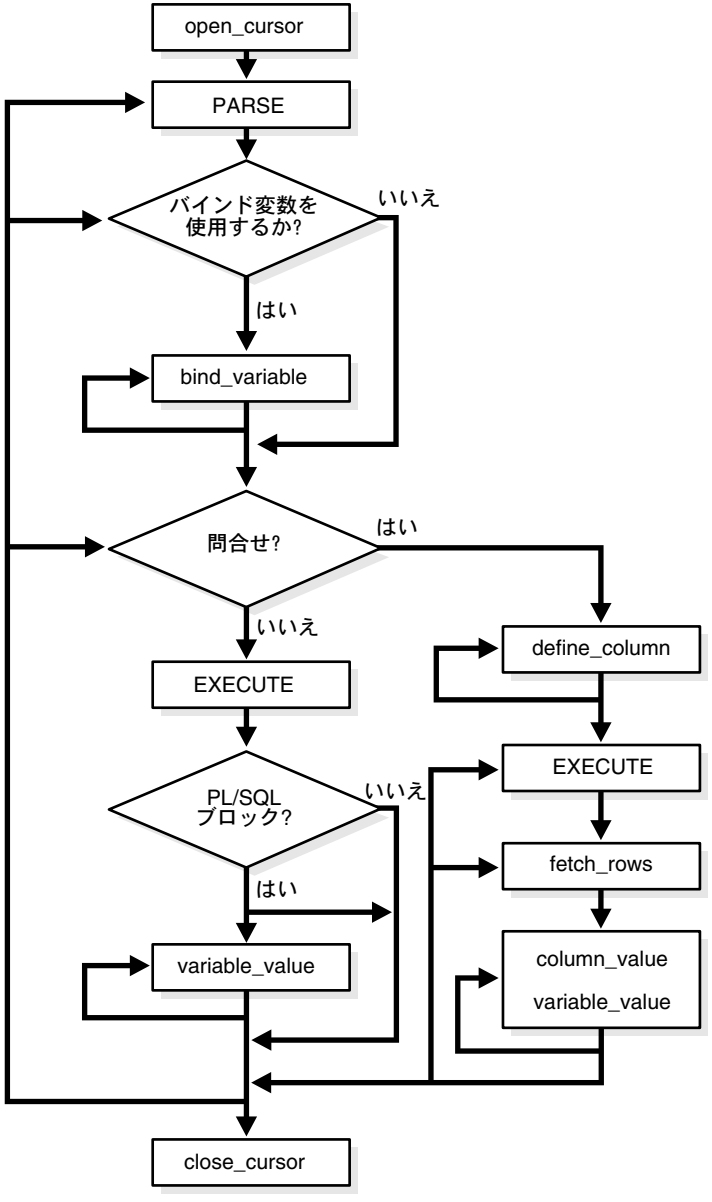
SQL 文はすべて、PARSE プロシージャをコールして解析する必要があります。文を解析することによって、その文の構文がチェックされ、プログラム内のカーソルに関連付けられます。

DML 文または DDL 文はすべて解析できます。DDL 文は解析時に実行され、暗黙のコミットを実行します。

> **注意：** パッケージやプロシージャを削除するために DDL 文を解析するときに、パッケージ内のプロシージャが使用中の場合はデッドロックが起こる可能性があります。プロシージャへのコール後は、実行がユーザー側に戻されるまで、そのプロシージャは使用中であるとみなされます。このようなデッドロックは、5 分後にタイムアウトします。

DBMS_SQL 実行フローを図 69-1 に示します。

図 69-1　DBMS_SQL 実行フロー



BIND_VARIABLE または BIND_ARRAY

多くの DML 文では、プログラム内のデータを Oracle に入力することが必要です。実行時に提供する入力データを含んでいる SQL 文を定義する場合は、SQL 文内のプレースホルダを使用して、データの提供場所にマークを付ける必要があります。

SQL 文内の各プレースホルダに対してバインド・プロシージャ（BIND_VARIABLE プロシージャまたは BIND_ARRAY プロシージャ）の 1 つをコールして、プログラム内の変数の値（または配列の値）をプレースホルダに提供する必要があります。SQL 文が引き続き実行されると、Oracle は、ユーザーのプログラムが入力変数と出力変数、またはバインド変数に設定したデータを使用します。

DBMS_SQL は、その都度異なるバインド変数を使用して DML 文を繰り返し実行できます。BIND_ARRAY プロシージャを使用すると、スカラーの集合をバインドでき、それぞれの値は各 EXECUTE につき 1 度のみ入力変数として使用されます。これは、OCI がサポートする配列インタフェースに類似しています。

DEFINE_COLUMN、DEFINE_COLUMN_LONG または DEFINE_ARRAY

SELECT 文内で選択されている行の列は、選択リスト内での相対位置（左から右）によって識別されます。問合せの場合は、定義プロシージャの 1 つ（DEFINE_COLUMN、DEFINE_COLUMN_LONG または DEFINE_ARRAY）をコールして SELECT 値を受け入れる変数を指定する必要があります。これは、INTO 句が静的問合せに対して行う方法とほとんど同じです。

DEFINE_COLUMN を使用して LONG 列以外の列を定義するのと同様に、
DEFINE_COLUMN_LONG プロシージャを使用して LONG 列を定義します。
COLUMN_VALUE_LONG を使用して LONG 列からフェッチする前に、
DEFINE_COLUMN_LONG をコールする必要があります。

DEFINE_ARRAY プロシージャを使用して、行を単一の SELECT 文でフェッチする PL/SQL コレクションを定義します。DEFINE_ARRAY は、1 回のフェッチで複数行をフェッチするインタフェースを提供します。COLUMN_VALUE プロシージャで行をフェッチする前に、DEFINE_ARRAY をコールする必要があります。

EXECUTE

EXECUTE ファンクションをコールして、SQL 文を実行します。

FETCH_ROWS または EXECUTE_AND_FETCH

FETCH_ROWS ファンクションは、問合せを満たす行を検索します。フェッチが行を検索できなくなるまで、連続する各フェッチは別の行を検索します。1 回のみの実行に対して EXECUTE をコールしている場合は、EXECUTE の次に FETCH_ROWS をコールするより、EXECUTE_AND_FETCH をコールする方が効率的です。

VARIABLE_VALUE、COLUMN_VALUE または COLUMN_VALUE_LONG

問合せの場合は、COLUMN_VALUE をコールして、FETCH_ROWS コールで検索する列の値を判別します。PL/SQL プロシージャへのコールまたは returning 句がある DML 文を含んだ無名ブロックの場合は、VARIABLE_VALUE をコールして、文の実行時に出力変数に割り当てられた値を検索します。

LONG データベース列（サイズは最大 2GB まで可能）の一部のみをフェッチするには、COLUMN_VALUE_LONG プロシージャを使用します。列値へのオフセット（バイト単位）とフェッチするバイト数を指定できます。

CLOSE_CURSOR

セッションでカーソルが不要な場合は、CLOSE_CURSOR をコールしてカーソルをクローズします。Oracle Open Gateway を使用している場合は、これ以外のときにもカーソルのクローズが必要になる場合があります。追加情報は、Oracle Open Gateway の関連文書を参照してください。

カーソルをクローズしないと、カーソルが不要になっても、そのカーソルが使用しているメモリーは割り当てられたままになります。

セキュリティ

定義者権限モジュール

定義者権限モジュールは、モジュールの所有者の権限下で実行されます。定義者権限モジュールからコールされた DBMS_SQL サブプログラムは、モジュールで定義されたスキーマに関して実行されます。

注意： Oracle8*i* より前のバージョンでは、すべての PL/SQL ストアド・プロシージャとパッケージが定義者権限モジュールでした。

実行者権限モジュール

実行者権限モジュールは、モジュールの実行者の権限下で実行されます。したがって、実行者権限モジュールからコールされた DBMS_SQL サブプログラムは、モジュールの実行者の権限下で実行されます。

モジュールに current_user に設定された AUTHID があると、未修飾の名前は、実行者のスキーマに関連付けて変換されます。

例

income は USER1 のスキーマにある実行者権限ストアド・プロシージャで、USER2 には、そのストアド・プロシージャに対する EXECUTE 権限が付与されています。

```
CREATE PROCEDURE income(amount number)
   AUTHID current_user IS
   c number;
   n number;
BEGIN
   c:= dbms_sql.open_cursor;
   dbms_sql.parse(c, 'insert into accts(''income'', :1)', dbms_sql.native);
   dbms_sql.bind_variable(c, '1', amount);
   n := dbms_sql.execute(c);
   dbms_sql.close_cursor(c);
END;
```

USER1 が USER1.income をコールする場合は、USER1 の権限が使用され、未修飾名の名前解決が USER1 のスキーマに関して行われます。

USER2 が USER1.income をコールする場合は、USER2 の権限が使用され、未修飾名の名前解決（たとえば、accts）が USER2 のスキーマに関して行われます。

関連項目：『PL/SQL ユーザーズ・ガイドおよびリファレンス』

無名ブロック

無名 PL/SQL ブロックからコールされたすべての DBMS_SQL サブプログラムは、カレント・ユーザーの権限を使用して実行されます。

問合せの処理

動的 SQL を使用して問合せを処理する場合は、次のステップを実行する必要があります。

1. DEFINE_COLUMN、DEFINE_COLUMN_LONG または DEFINE_ARRAY をコールして、SELECT 文が戻す値を受け入れる変数を指定します。
2. EXECUTE をコールして、SELECT 文を実行します。
3. FETCH_ROWS（または EXECUTE_AND_FETCH）をコールして、問合せに一致した行を検索します。
4. 問合せに関して FETCH_ROWS コールが検索した列の値を判別するために、COLUMN_VALUE または COLUMN_VALUE_LONG をコールします。PL/SQL プロシージャへのコールを含んだ無名ブロックを使用した場合は、VARIABLE_VALUE をコールして、これらのプロシージャの出力変数に割り当てられた値を検索します。

例

この項には、DBMS_SQL パッケージを使用するプロシージャの例が記述されています。

例 1

次のプロシージャの例は、プロシージャに SQL 文を渡し、その SQL 文を解析して実行します。

```
CREATE OR REPLACE PROCEDURE exec(STRING IN varchar2) AS
    cursor_name INTEGER;
    ret INTEGER;
BEGIN
   cursor_name := DBMS_SQL.OPEN_CURSOR;
```

DDL 文は PARSE をコールして実行され、暗黙のコミットが実行されます。

```
   DBMS_SQL.PARSE(cursor_name, string, DBMS_SQL.native);
   ret := DBMS_SQL.EXECUTE(cursor_name);
   DBMS_SQL.CLOSE_CURSOR(cursor_name);
END;
```

このようなプロシージャを作成すると、次の操作を実行できます。

- コール元のプログラムによって、実行時に SQL 文を動的に生成できます。
- SQL 文は、DDL 文またはバインドなしの DML で構いません。

たとえば、このプロシージャの作成後に、次のコールを行うことができます。

```
exec('create table acct(c1 integer)');
```

次の例のように、このプロシージャはリモートでコールすることもできます。これによって、リモート DDL を実行できます。

```
exec@hq.com('CREATE TABLE acct(c1 INTEGER)');
```

例 2

次のプロシージャの例は、コピー元表とコピー先表の名前が渡され、コピー元表からコピー先表に行をコピーします。このプロシージャの例は、コピー元表とコピー先表にはいずれも次の列があることを前提としています。

```
id        of type NUMBER
name      of type VARCHAR2(30)
birthdate of type DATE
```

このプロシージャでは、動的 SQL を使用する必要は特にありませんが、ここでは、このパッケージの概念をわかりやすく説明しています。

```
CREATE OR REPLACE PROCEDURE copy (
     source      IN VARCHAR2,
     destination IN VARCHAR2) IS
     id_var              NUMBER;
     name_var            VARCHAR2(30);
     birthdate_var       DATE;
     source_cursor       INTEGER;
     destination_cursor INTEGER;
     ignore              INTEGER;
  BEGIN

  -- Prepare a cursor to select from the source table:
     source_cursor := dbms_sql.open_cursor;
     DBMS_SQL.PARSE(source_cursor,
         'SELECT id, name, birthdate FROM ' || source,
          DBMS_SQL.native);
     DBMS_SQL.DEFINE_COLUMN(source_cursor, 1, id_var);
     DBMS_SQL.DEFINE_COLUMN(source_cursor, 2, name_var, 30);
     DBMS_SQL.DEFINE_COLUMN(source_cursor, 3, birthdate_var);
     ignore := DBMS_SQL.EXECUTE(source_cursor);

  -- Prepare a cursor to insert into the destination table:
     destination_cursor := DBMS_SQL.OPEN_CURSOR;
     DBMS_SQL.PARSE(destination_cursor,
                  'INSERT INTO ' || destination ||
                  ' VALUES (:id_bind, :name_bind, :birthdate_bind)',
                   DBMS_SQL.native);

  -- Fetch a row from the source table and insert it into the destination table:
     LOOP
       IF DBMS_SQL.FETCH_ROWS(source_cursor)>0 THEN
         -- get column values of the row
         DBMS_SQL.COLUMN_VALUE(source_cursor, 1, id_var);
         DBMS_SQL.COLUMN_VALUE(source_cursor, 2, name_var);
         DBMS_SQL.COLUMN_VALUE(source_cursor, 3, birthdate_var);
```

```
  -- Bind the row into the cursor that inserts into the destination table. You
  -- could alter this example to require the use of dynamic SQL by inserting an
  -- if condition before the bind.
        DBMS_SQL.BIND_VARIABLE(destination_cursor, ':id_bind', id_var);
        DBMS_SQL.BIND_VARIABLE(destination_cursor, ':name_bind', name_var);
        DBMS_SQL.BIND_VARIABLE(destination_cursor, ':birthdate_bind',
birthdate_var);
        ignore := DBMS_SQL.EXECUTE(destination_cursor);
      ELSE

  -- No more rows to copy:
        EXIT;
      END IF;
    END LOOP;

  -- Commit and close all cursors:
     COMMIT;
     DBMS_SQL.CLOSE_CURSOR(source_cursor);
     DBMS_SQL.CLOSE_CURSOR(destination_cursor);
   EXCEPTION
     WHEN OTHERS THEN
       IF DBMS_SQL.IS_OPEN(source_cursor) THEN
         DBMS_SQL.CLOSE_CURSOR(source_cursor);
       END IF;
       IF DBMS_SQL.IS_OPEN(destination_cursor) THEN
         DBMS_SQL.CLOSE_CURSOR(destination_cursor);
       END IF;
       RAISE;
  END;
/
```

例 3、4 および 5: バルク DML

次の一連の例では、DELETE、INSERT および UPDATE の各 SQL DML 文でのバルク配列バインド（表項目）の使用方法を示します。

たとえば、DELETE 文では、WHERE 句に配列をバインドし、配列内の要素ごとに文を実行できます。

```
declare
  stmt varchar2(200);
  dept_no_array dbms_sql.Number_Table;
  c number;
  dummy number;
begin
  dept_no_array(1) := 10; dept_no_array(2) := 20;
  dept_no_array(3) := 30; dept_no_array(4) := 40;
  dept_no_array(5) := 30; dept_no_array(6) := 40;
  stmt := 'delete from emp where deptno = :dept_array';
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, stmt, dbms_sql.native);
  dbms_sql.bind_array(c, ':dept_array', dept_no_array, 1, 4);
  dummy := dbms_sql.execute(c);
  dbms_sql.close_cursor(c);

  exception when others then
    if dbms_sql.is_open(c) then
      dbms_sql.close_cursor(c);
    end if;
    raise;
end;
/
```

前述の例では、1 から 4 までの要素のみが、bind_array コールで指定したとおりに使用されます。配列の各要素は、大量の従業員をデータベースから削除する可能性があります。

次に、バルク INSERT 文の例を示します。

```
declare
  stmt varchar2(200);
  empno_array dbms_sql.Number_Table;
  empname_array dbms_sql.Varchar2_Table;
  c number;
  dummy number;
begin
  for i in 0..9 loop
    empno_array(i) := 1000 + i;
    empname_array(I) := get_name(i);
  end loop;
  stmt := 'insert into emp values(:num_array, :name_array)';
```

```
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, stmt, dbms_sql.native);
  dbms_sql.bind_array(c, ':num_array', empno_array);
  dbms_sql.bind_array(c, ':name_array', empname_array);
  dummy := dbms_sql.execute(c);
  dbms_sql.close_cursor(c);

  exception when others then
    if dbms_sql.is_open(c) then
      dbms_sql.close_cursor(c);
    end if;
    raise;
end;
/
```

実行が開始されると、10 人の従業員はすべて表に挿入されます。

最後に、バルク UPDATE 文の例を示します。

```
declare
  stmt varchar2(200);
  emp_no_array dbms_sql.Number_Table;
  emp_addr_array dbms_sql.Varchar2_Table;
  c number;
  dummy number;
begin
  for i in 0..9 loop
    emp_no_array(i) := 1000 + i;
    emp_addr_array(I) := get_new_addr(i);
  end loop;
  stmt := 'update emp set ename = :name_array
    where empno = :num_array';
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, stmt, dbms_sql.native);
  dbms_sql.bind_array(c, ':num_array', empno_array);
  dbms_sql.bind_array(c, ':name_array', empname_array);
  dummy := dbms_sql.execute(c);
  dbms_sql.close_cursor(c);

  exception when others then
    if dbms_sql.is_open(c) then
      dbms_sql.close_cursor(c);
    end if;
    raise;
end;
/
```

EXECUTE がコールされると、全従業員のアドレスが一度に更新されます。2 つのコレクションの処理は、いつも同時に進行します。WHERE 句で複数の行が戻される場合、全従業員は、その時点で addr_array が参照するアドレスを取得します。

例 6 および 7: 配列の定義

次の例では、DEFINE_ARRAY プロシージャの使用方法を示します。

```
declare
  c       number;
  d       number;
  n_tab   dbms_sql.Number_Table;
  indx    number := -10;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'select n from t order by 1', dbms_sql);

  dbms_sql.define_array(c, 1, n_tab, 10, indx);

  d := dbms_sql.execute(c);
  loop
    d := dbms_sql.fetch_rows(c);

    dbms_sql.column_value(c, 1, n_tab);

    exit when d != 10;
  end loop;

  dbms_sql.close_cursor(c);

  exception when others then
    if dbms_sql.is_open(c) then
      dbms_sql.close_cursor(c);
    end if;
    raise;
end;
/
```

前述の例では FETCH_ROWS をコールするたびに、DBMS_SQL バッファに保持されている 10 行がフェッチされます。COLUMN_VALUE コールが実行されると、それらの行は指定した PL/SQL 表（この場合は n_tab）の DEFINE 文で指定した -10 から -1 の位置に移動します。次に、2 番目のバッチがループ内でフェッチされ、行が 0（ゼロ）から 9 の位置に移動し、あとは同様に続きます。

各配列への現行の索引は、自動的にメンテナンスされます。この索引は、EXECUTE 時に indx に初期化され、COLUMN_VALUE がコールされるたびに更新されます。任意の時点で再実行した場合、各 DEFINE の現行の索引は indx に再初期化されます。

このようにして、問合せのすべての結果が表内にフェッチされます。FETCH_ROWS で 10 行をフェッチできない場合は、実際にフェッチされた行数を戻して（1 行もフェッチできなかった場合は 0（ゼロ）を戻します)、ループを終了します。

DEFINE_ARRAY プロシージャの別の使用例を次に示します。

次のように定義された MULTI_TAB 表を想定します。

```
create table multi_tab (num number,
                        dat1 date,
                        var varchar2(24),
                        dat2 date)
```

この表からすべてを選択して 4 つの PL/SQL 表に移動するには、次の簡単なプログラムを使用できます。

```
declare
  c       number;
  d       number;
  n_tab  dbms_sql.Number_Table;
  d_tab1 dbms_sql.Date_Table;
  v_tab  dbms_sql.Varchar2_Table;
  d_tab2 dbms_sql.Date_Table;
  indx number := 10;
begin

  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'select * from multi_tab order by 1', dbms_sql);

  dbms_sql.define_array(c, 1, n_tab,  5, indx);
  dbms_sql.define_array(c, 2, d_tab1, 5, indx);
  dbms_sql.define_array(c, 3, v_tab,  5, indx);
  dbms_sql.define_array(c, 4, d_tab2, 5, indx);

  d := dbms_sql.execute(c);

  loop
    d := dbms_sql.fetch_rows(c);

    dbms_sql.column_value(c, 1, n_tab);
    dbms_sql.column_value(c, 2, d_tab1);
    dbms_sql.column_value(c, 3, v_tab);
    dbms_sql.column_value(c, 4, d_tab2);

    exit when d != 5;
```

```
  end loop;

  dbms_sql.close_cursor(c);

/*
```

これで、4 つの表はあらゆる用途に使用できます。使用方法の 1 つは、'INSERT into SOME_T values (:a, :b, :c, :d) ' などの問合せを使用して、行を他の表に移動するために BIND_ARRAY を使用できます。

```
*/

exception when others then
    if dbms_sql.is_open(c) then
      dbms_sql.close_cursor(c);
    end if;
    raise;
end;
/
```

例 8: 列の定義の表示

この例は、定義を表示する表に対して SELECT * による問合せを使用し、SQL*Plus の DESCRIBE コールのかわりに使用できます。

```
declare
  c number;
  d number;
  col_cnt integer;
  f boolean;
  rec_tab dbms_sql.desc_tab;
  col_num number;
  procedure print_rec(rec in dbms_sql.desc_rec) is
  begin
    dbms_output.new_line;
    dbms_output.put_line('col_type            =    '
                         || rec.col_type);
    dbms_output.put_line('col_maxlen          =    '
                         || rec.col_max_len);
    dbms_output.put_line('col_name            =    '
                         || rec.col_name);
    dbms_output.put_line('col_name_len        =    '
                         || rec.col_name_len);
    dbms_output.put_line('col_schema_name     =    '
                         || rec.col_schema_name);
    dbms_output.put_line('col_schema_name_len =    '
                         || rec.col_schema_name_len);
    dbms_output.put_line('col_precision       =    '
```

```
                        || rec.col_precision);
    dbms_output.put_line('col_scale           =    '
                        || rec.col_scale);
    dbms_output.put('col_null_ok         =    ');
    if (rec.col_null_ok) then
      dbms_output.put_line('true');
    else
      dbms_output.put_line('false');
    end if;
  end;
begin
  c := dbms_sql.open_cursor;

  dbms_sql.parse(c, 'select * from scott.bonus', dbms_sql);

  d := dbms_sql.execute(c);

  dbms_sql.describe_columns(c, col_cnt, rec_tab);

/*
 * Following loop could simply be for j in 1..col_cnt loop.
 * Here we are simply illustrating some of the PL/SQL table
 * features.
 */
  col_num := rec_tab.first;
  if (col_num is not null) then
    loop
      print_rec(rec_tab(col_num));
      col_num := rec_tab.next(col_num);
      exit when (col_num is null);
    end loop;
  end if;

  dbms_sql.close_cursor(c);
end;
/
```

例 9: RETURNING 句 RETURNING 句は、Oracle 8.0.3 で DML 文に追加されています。この句を使用して、INSERT 文、UPDATE 文および DELETE 文は、式の値を戻すことができます。この値は、バインド変数に戻されます。

単一行が挿入、更新または削除される場合は、DBMS_SQL.BIND_VARIABLE を使用して、これらのアウトバインドをバインドします。複数行が挿入、更新または削除された場合は、DBMS_SQL.BIND_ARRAY を使用します。これらのバインド変数の値を取得するには、DBMS_SQL.VARIABLE_VALUE をコールする必要があります。

注意： これは、DBMS_SQL 内でアウトバインドを使用した PL/SQL ブロックを実行した後で、DBMS_SQL.VARIABLE_VALUE をコールする必要があることと同様です。

i) 単一行の挿入

```
create or replace procedure single_Row_insert
     (c1 number, c2 number, r out number) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'insert into tab values (:bnd1, :bnd2) ' ||
                    'returning c1*c2 into :bnd3', 2);
dbms_sql.bind_variable(c, 'bnd1', c1);
  dbms_sql.bind_variable(c, 'bnd2', c2);
  dbms_sql.bind_variable(c, 'bnd3', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd3', r); -- get value of outbind variable
  dbms_Sql.close_Cursor(c);
end;
/
```

ii) 単一行の更新

```
create or replace procedure single_Row_update
     (c1 number, c2 number, r out number) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'update tab set c1 = :bnd1, c2 = :bnd2 ' ||
                    'where rownum < 2' ||
                    'returning c1*c2 into :bnd3', 2);
  dbms_sql.bind_variable(c, 'bnd1', c1);
  dbms_sql.bind_variable(c, 'bnd2', c2);
  dbms_sql.bind_variable(c, 'bnd3', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd3', r);-- get value of outbind variable
  dbms_Sql.close_Cursor(c);
end;
/
```

iii) 単一行の削除

```
create or replace procedure single_Row_Delete
     (c1 number, c2 number, r out number) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'delete from tab ' ||
                    'where rownum < 2 ' ||
                   'returning c1*c2 into :bnd3', 2);
  dbms_sql.bind_variable(c, 'bnd1', c1);
  dbms_sql.bind_variable(c, 'bnd2', c2);
  dbms_sql.bind_variable(c, 'bnd3', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd3', r);-- get value of outbind variable
  dbms_Sql.close_Cursor(c);
end;
/
```

iv) 複数行の挿入

```
create or replace procedure multi_Row_insert
     (c1 dbms_sql.number_table, c2 dbms_sql.number_table,
      r out dbms_sql.number_table) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'insert into tab values (:bnd1, :bnd2) ' ||
                    'returning c1*c2 into :bnd3', 2);
  dbms_sql.bind_array(c, 'bnd1', c1);
  dbms_sql.bind_array(c, 'bnd2', c2);
  dbms_sql.bind_array(c, 'bnd3', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd3', r);-- get value of outbind variable
  dbms_Sql.close_Cursor(c);
end;
/
```

v) 複数行の更新

```
create or replace procedure multi_Row_update
     (c1 number, c2 number, r out dbms_Sql.number_table) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'update tab set c1 = :bnd1 where c2 = :bnd2 ' ||
                    'returning c1*c2 into :bnd3', 2);
  dbms_sql.bind_variable(c, 'bnd1', c1);
  dbms_sql.bind_variable(c, 'bnd2', c2);
  dbms_sql.bind_array(c, 'bnd3', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd3', r);-- get value of outbind variable
  dbms_Sql.close_Cursor(c);
end;
/
```

注意： bnd1 と bnd2 は、同様に配列にできます。更新されたすべての行に対する式の値は、bnd3 に入れられます。bnd1 と bnd2 の各値について、どの行が更新されたかを区別する方法はありません。

vi) 複数行の削除

```
create or replace procedure multi_row_delete
     (c1 dbms_Sql.number_table,
      r out dbms_sql.number_table) is
c number;
n number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'delete from tab where c1 = :bnd1' ||
                    'returning c1*c2 into :bnd2', 2);
  dbms_sql.bind_array(c, 'bnd1', c1);
  dbms_sql.bind_array(c, 'bnd2', r);
  n := dbms_sql.execute(c);
  dbms_sql.variable_value(c, 'bnd2', r);-- get value of outbind variable
  dbms_Sql.close_Cursor(c);
end;
/
```

vii) バルク PL/SQL でのアウトバインド

```
create or replace foo (n number, square out number) is
begin square := n * n; end;/

create or replace procedure bulk_plsql
   (n dbms_sql.number_Table, square out dbms_sql.number_table) is
c number;
r number;
begin
  c := dbms_sql.open_cursor;
  dbms_sql.parse(c, 'begin foo(:bnd1, :bnd2); end;', 2);
  dbms_sql.bind_array(c, 'bnd1', n);
  dbms_Sql.bind_Array(c, 'bnd2', square);
  r := dbms_sql.execute(c);
  dbms_Sql.variable_Value(c, 'bnd2', square);
end;
/
```

注意： number_Table の DBMS_SQL.BIND_ARRAY は、数値を内部的にバインドします。文を実行する回数は、インバインド配列内の要素数によって決まります。

更新、挿入および削除の処理

動的 SQL を使用して INSERT、UPDATE または DELETE を処理する場合は、次のステップを実行する必要があります。

1. 最初に、EXECUTE をコールして、INSERT 文、UPDATE 文または DELETE 文を実行します。
2. 文に returning 句がある場合は、VARIABLE_VALUE をコールして出力変数に割り当てられた値を取り出します。

エラーの位置

DBMS_SQL パッケージには、セッションで最後に参照されたカーソルの情報を取得するための追加ファンクションがいくつかあります。これらのファンクションが戻す値は、SQL 文の実行直後にのみ意味を持ちます。また、エラーを検出するファンクションは、特定の DBMS_SQL コール後にのみ意味を持ちます。たとえば、PARSE 直後に LAST_ERROR_POSITION をコールします。

DBMS_SQL サブプログラムの要約

表 69-1 DBMS_SQL のサブプログラム

サブプログラム	説明
「OPEN_CURSOR ファンクション」 69-24 ページ	新規カーソルのカーソル番号を戻します。
「PARSE プロシージャ」 69-25 ページ	指定した文を解析します。
「BIND_VARIABLE および BIND_ARRAY プロシージャ」 69-27 ページ	指定の値を指定の変数にバインドします。
「BIND_VARIABLE および BIND_ARRAY プロシージャ」 69-27 ページ	指定の値を指定のコレクションにバインドします。
「DEFINE_COLUMN プロシージャ」 69-32 ページ	指定したカーソルから選択し、SELECT 文のみで使用する列を定義します。
「DEFINE_ARRAY プロシージャ」 69-34 ページ	指定したカーソルから選択し、SELECT 文のみで使用するコレクションを定義します。
「DEFINE_COLUMN_LONG プロシージャ」69-36 ページ	指定したカーソルから選択し、SELECT 文のみで使用する LONG 列を定義します。
「EXECUTE ファンクション」 69-36 ページ	指定のカーソルを実行します。
「EXECUTE_AND_FETCH ファンクション」69-37 ページ	指定のカーソルを実行して、行をフェッチします。
「FETCH_ROWS ファンクション」 69-38 ページ	指定のカーソルから行をフェッチします。
「COLUMN_VALUE プロシージャ」 69-39 ページ	カーソルの指定位置にあるカーソル要素の値を戻します。
「COLUMN_VALUE_LONG プロシージャ」69-42 ページ	DEFINE_COLUMN_LONG で定義した LONG 列の選択された部分を戻します。
「VARIABLE_VALUE プロシージャ」 69-43 ページ	指定のカーソルについて指定の変数の値を戻します。
「IS_OPEN ファンクション」 69-45 ページ	指定のカーソルがオープンの場合に TRUE を戻します。

表 69-1 DBMS_SQL のサブプログラム （続き）

サブプログラム	説明
「DESCRIBE_COLUMNS プロシージャ」69-46 ページ	DBMS_SQL を介してオープンして解析されたカーソルの列を記述します。
「CLOSE_CURSOR プロシージャ」69-48 ページ	指定したカーソルをクローズして、メモリーを解放します。
「LAST_ERROR_POSITION ファンクション」69-48 ページ	エラーが発生した SQL 文テキスト内のバイト・オフセットを戻します。
「LAST_ROW_COUNT ファンクション」69-49 ページ	フェッチされた累積行数を戻します。
「LAST_ROW_ID ファンクション」69-49 ページ	最後に処理された行の ROWID を戻します。
「LAST_SQL_FUNCTION_CODE ファンクション」69-50 ページ	文の SQL ファンクション・コードを戻します。

OPEN_CURSOR ファンクション

このプロシージャは、新規のカーソルをオープンします。このカーソルが不要になった場合は、CLOSE_CURSOR をコールして、明示的にクローズする必要があります。

カーソルを使用すると、同じ SQL 文を繰り返し実行したり、新規の SQL 文を実行することができます。カーソルを再使用した場合、新しい SQL 文が解析されたときに対応するカーソル・データ領域の内容がリセットされます。カーソルを再使用しないかぎり、クローズして再オープンする必要はありません。

構文

```
DBMS_SQL.OPEN_CURSOR
   RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(open_cursor,RNDS,WNDS);
```

戻り値

このファンクションは、新規カーソルのカーソル ID 番号を戻します。

PARSE プロシージャ

このプロシージャは、指定したカーソル内の指定した文を解析します。すべての文が即時に解析されます。さらに、DDL 文は、解析時にただちに実行されます。

PARSE プロシージャには、2 つのバージョンがあります。1 つは引数として VARCHAR2 文を使用するバージョン、もう 1 つは引数として VARCHAR2S（VARCHAR2 の表）を使用するバージョンです。

> **注意：** DBMS_SQL を使用して DDL 文を動的に実行すると、プログラムがハングする場合があります。たとえば、パッケージ内のプロシージャをコールすると、実行がユーザー側に戻るまでそのパッケージがロックされます。最初のロックを解放する前に、動的にパッケージを削除するなど、ロックの競合を引き起こす操作を行うとプログラムはハングします。

前述の構文を持つ SQL 文を解析するためのサイズは、32KB に制限されています。

構文

```
DBMS_SQL.PARSE (
   c                 IN INTEGER,
   statement         IN VARCHAR2,
   language_flag     IN INTEGER);
```

PARSE プロシージャは、大規模な SQL 文について次の構文もサポートしています。

> **注意：** このプロシージャは、PL/SQL 表の文の要素を連結し、その結果の文字列を解析します。このプロシージャを使用すると、文を分割することによって、単一の VARCHAR2 変数についての制限を超えた長い文を解析できます。

```
DBMS_SQL.PARSE (
   c                 IN INTEGER,
   statement         IN VARCHAR2S,
   lb                IN INTEGER,
   ub                IN INTEGER,
   lfflg             IN BOOLEAN,
   language_flag     IN INTEGER);
```

パラメータ

表 69-2 PARSE プロシージャのパラメータ

パラメータ	説明
c	文を解析するカーソルの ID 番号。

statement	解析する SQL 文。
	PL/SQL 文と異なり、SQL 文の終わりにはセミコロンを含めないでください。たとえば、次のようにします。
	`DBMS_SQL.PARSE(cursor1, 'BEGIN proc; END;', 2);`
	`DBMS_SQL.PARSE(cursor1, 'INSERT INTO tab values(1)', 2);`

lb	文内の要素の下限。
ub	文内の要素の上限。
lfflg	TRUE の場合、連結している各要素の後に改行を挿入します。
language_flag	Oracle で SQL 文を処理する方法を決定します。次のオプションが認識されます。 ■ V6（または 0）は、バージョン 6 の動作を指定します。 ■ NATIVE（または 1）は、プログラムの接続先のデータベースに関する通常の動作を指定します。 ■ V7（または 2）は、Oracle7 の動作を指定します。

注意： クライアント側コードは、リモート・パッケージの変数または定数を参照できないため、定数の値を明示的に使用する必要があります。

たとえば、次のコードは、クライアント側でコンパイルしません。

```
DBMS_SQL.PARSE(cur_hdl, stmt_str, dbms_sql.V7); -- uses
constant dbms_sql.V7
```

次のコードは、引数が明示的に指定されているので、クライアント側で有効です。

```
DBMS_SQL.PARSE(cur_hdl, stmt_str, 2); -- compiles on the
client
```

例 9: 大規模な SQL 文字列を解析するための VARCHAR2S データ・タイプ

32KB を超える SQL 文を解析するために、DBMS_SQL パッケージは、PL/SQL 表を使用して文字列の表を PARSE プロシージャに渡します。これらの文字列は連結された後、Oracle サーバーに渡されます。

ローカル変数を VARCHAR2S 表項目タイプとして宣言し、次に、PARSE プロシージャを使用すると、大規模な SQL 文を VARCHAR2S として解析できます。

VARCHAR2S データ・タイプの定義は、次のとおりです。

```
TYPE varchar2s IS TABLE OF VARCHAR2(256) INDEX BY BINARY_INTEGER;
```

例外

コンパイルに関する警告を伴った DBMS_SQL を使用して、タイプ、プロシージャ、ファンクションまたはパッケージを作成する場合、ORA-24344 例外が発生しますが、プロシージャはそのまま作成されます。

BIND_VARIABLE および BIND_ARRAY プロシージャ

この 2 つのプロシージャは、文内の変数の名前に基づいて、カーソル内の指定の変数に指定の値または値のセットをバインドします。この変数が IN 変数、IN/OUT 変数または IN コレクションである場合は、指定したバインド値が、変数タイプまたは配列タイプに対して有効である必要があります。OUT 変数のバインド値は無視されます。

SQL 文のバインド変数またはコレクションは、名前によって識別されます。バインド変数またはバインド配列に値をバインドする場合は、次の例に示すように、文中でバインド変数を識別する文字列の先頭にコロンを付ける必要があります。

```
SELECT emp_name FROM emp WHERE SAL > :X;
```

この例では、対応するバインド・コールは次のようになります。

```
BIND_VARIABLE(cursor_name, ':X', 3500);
```

または

```
BIND_VARIABLE (cursor_name, 'X', 3500);
```

構文

```
DBMS_SQL.BIND_VARIABLE (
   c              IN INTEGER,
   name           IN VARCHAR2,
   value          IN <datatype>);
```

<datatype> は、次のいずれかのタイプである必要があります。

```
NUMBER
DATE
VARCHAR2 CHARACTER SET ANY_CS
BLOB
CLOB CHARACTER SET ANY_CS
BFILE
UROWID
```

BIND_VARIABLE は、異なるデータ・タイプを受け入れるためにオーバーロードされていることに注意してください。

> **関連項目：**『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』

プラグマ

```
pragma restrict_references(bind_variable,WNDS);
```

使用上の注意

BIND_VARIABLE では、次の構文もサポートされています。大カッコ [] は、BIND_VARIABLE ファンクションのオプション・パラメータを示します。

```
DBMS_SQL.BIND_VARIABLE (
   c              IN INTEGER,
   name           IN VARCHAR2,
   value          IN VARCHAR2 CHARACTER SET ANY_CS [,out_value_size IN INTEGER]);
```

CHAR、RAW および ROWID データをバインドするために、次のバリエーションを構文で使用できます。

```
DBMS_SQL.BIND_VARIABLE_CHAR (
   c              IN INTEGER,
   name           IN VARCHAR2,
   value          IN CHAR CHARACTER SET ANY_CS [,out_value_size IN INTEGER]);

DBMS_SQL.BIND_VARIABLE_RAW (
   c              IN INTEGER,
   name           IN VARCHAR2,
   value          IN RAW [,out_value_size IN INTEGER]);
```

```
DBMS_SQL.BIND_VARIABLE_ROWID (
   c               IN INTEGER,
   name            IN VARCHAR2,
   value           IN ROWID);
```

パラメータ

表 69-3 BIND_VARIABLE プロシージャのパラメータ

パラメータ	説明
c	値をバインドするカーソルの ID 番号。
name	文内の変数の名前。
value	カーソル内の変数にバインドする値。 IN 変数と IN/OUT 変数の場合、この値は、このパラメータで渡される値のタイプと同じタイプです。
out_value_size	VARCHAR2、RAW、CHAR OUT または IN/OUT 変数の最大予測 OUT 値サイズ（バイト単位）。 サイズの指定がない場合は、現行値の長さが使用されます。このパラメータは、value パラメータが初期化されていない場合、指定する必要があります。

バルク配列バインド

バルク SELECT、INSERT、UPDATE および DELETE は、多くのコールを 1 つにまとめることによって、アプリケーションのパフォーマンスを改善できます。DBMS_SQL パッケージによって、ユーザーは PL/SQL 表タイプを使用しながらデータの収集に対する処理を実行できます。

表項目は、バインドされていない同種のコレクションです。表項目は、持続記憶域では他のリレーショナル表に似ており、組込みの配列を持ちません。ただし、表項目が、(問合せまたは持続データのナビゲーション・アクセスのいずれかによって）作業領域に移されたり、あるいは PL/SQL の変数またはパラメータの値として作成されると、要素の値を取得して設定するために配列形式の構文で使用できる添字が、その表項目の要素に与えられます。

これらの要素の添字は詳細である必要はなく、負数を含むあらゆる数値が使用できます。たとえば、表項目には、-10、2 および 7 の位置のみにある要素を含めることができます。

表項目が一時作業領域から持続記憶域に移されると、添字は格納されません。つまり、表項目は、持続記憶域内では順序が付いていません。

表は、バインド実行時に、PL/SQL バッファからローカルの DBMS_SQL バッファにコピーされ（すべてのスカラー・タイプについて同様）、ローカルの DBMS_SQL バッファから操作されます。したがって、バインド・コール後に表を変更した場合でも、その変更が実行方法に影響を与えることはありません。

スカラー・タイプと LOB コレクション・タイプ

ローカル変数を次のいずれかの表項目タイプとして宣言できます。これらの表項目タイプは、DBMS_SQL ではパブリック・タイプとして定義されています。

```
type Number_Table   IS TABLE OF NUMBER         INDEX BY BINARY_INTEGER;
type Varchar2_Table IS TABLE OF VARCHAR2(2000) INDEX BY BINARY_INTEGER;
type Date_Table     IS TABLE OF DATE           INDEX BY BINARY_INTEGER;
type Blob_Table     IS TABLE OF BLOB           INDEX BY BINARY_INTEGER;
type Clob_Table     IS TABLE OF CLOB           INDEX BY BINARY_INTEGER;
type Bfile_Table    IS TABLE OF BFILE          INDEX BY BINARY_INTEGER;
type Urowid_Table   IS TABLE OF UROWID         INDEX BY BINARY_INTEGER;
```

構文

```
DBMS_SQL.BIND_ARRAY (
   c                   IN INTEGER,
   name                IN VARCHAR2,
   <table_variable>    IN <datatype>
 [,index1              IN INTEGER,
   index2              IN INTEGER)] );
```

<table_variable> とそれに対応する <datatype> は、次のいずれかの組合せになります。

```
<num_tab>      Number_Table
<vchr2_tab>    Varchar2_Table
<date_tab>     Date_Table
<blob_tab>     Blob_Table
<clob_tab>     Clob_Table
<bfile_tab>    Bfile_Table
<urowid_tab>   Urowid_Table
```

BIND_ARRAY プロシージャは、異なるデータ・タイプを受け入れるためにオーバーロードされていることに注意してください。

パラメータ

表 69-4 BIND_ARRAY プロシージャのパラメータ

パラメータ	説明
c	値をバインドするカーソルの ID 番号。
name	文内のコレクションの名前。
table_variable	<datatype> として宣言されたローカル変数。
index1	範囲の下限を示す表要素の索引。
index2	範囲の上限を示す表要素の索引。

使用上の注意

範囲をバインドするためには、範囲を指定する要素（タブ（index1）とタブ（index2））が表に含まれている必要がありますが、その範囲は詳細でなくても構いません。index1 には、index2 以下の値を指定してください。タブ（index1）とタブ（index2）の間にあるすべての要素がバインドして使用されます。

バインド・コールで索引を指定しない場合で、かつ文内の 2 つの異なるバインドが異なる数の要素を含んだ表を指定している場合、実際に使用される要素の数は、すべての表の最小値となります。これは索引を指定する場合にも当てはまります。つまり、すべての表に関する 2 つの索引の間では最小範囲が選択されます。

問合せ内のすべてのバインド変数が、配列バインドである必要はありません。一部は通常のバインドの場合があり、式の評価などでは、同じ値がコレクションの各要素に使用されます。

> **関連項目：** コレクションのバインド方法の例は、69-13 ページの「例 3、4 および 5: バルク DML」を参照してください。

DEFINE_COLUMN プロシージャ

このプロシージャは、指定のカーソルから選択する列を定義します。このプロシージャが使用できるのは、SELECT カーソルのみです。

定義されている列は、指定のカーソル内にある文の SELECT リスト内での相対位置によって識別されます。COLUMN 値のタイプによって、定義されている列のタイプが決まります。

構文

```
DBMS_SQL.DEFINE_COLUMN (
   c              IN INTEGER,
   position       IN INTEGER,
   column         IN <datatype>);
```

<datatype> は、次のいずれかのタイプである必要があります。

```
NUMBER
DATE
BLOB
CLOB CHARACTER SET ANY_CS
BFILE
UROWID
```

DEFINE_COLUMN は、異なるデータ・タイプを受け入れるためにオーバーロードされていることに注意してください。

> **関連項目：**『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』

プラグマ

```
pragma restrict_references(define_column,RNDS,WNDS);
```

DEFINE_COLUMN プロシージャでは、次の構文もサポートされます。

```
DBMS_SQL.DEFINE_COLUMN (
   c              IN INTEGER,
   position       IN INTEGER,
   column         IN VARCHAR2 CHARACTER SET ANY_CS,
   column_size    IN INTEGER,
   urowid         IN INTEGER);
```

CHAR、RAW および ROWID データを持つ列の定義には、プロシージャ構文で次のバリエーションを使用できます。

```
DBMS_SQL.DEFINE_COLUMN_CHAR (
   c              IN INTEGER,
   position       IN INTEGER,
   column         IN CHAR CHARACTER SET ANY_CS,
   column_size    IN INTEGER);

DBMS_SQL.DEFINE_COLUMN_RAW (
   c              IN INTEGER,
   position       IN INTEGER,
   column         IN RAW,
   column_size    IN INTEGER);

DBMS_SQL.DEFINE_COLUMN_ROWID (
   c              IN INTEGER,
   position       IN INTEGER,
   column         IN ROWID);
```

パラメータ

表 69-5 DEFINE_COLUMN プロシージャのパラメータ

パラメータ	説明
c	選択対象に定義されている行のカーソルの ID 番号。
position	定義している行内にある列の相対位置。 文の最初の列は位置 1 です。
column	定義している列の値。 この値のタイプによって、定義している列のタイプが決まります。
column_size	VARCHAR2 タイプ、CHAR タイプおよび RAW タイプの列に対する列値の最大予測サイズ（バイト単位）。

DEFINE_ARRAY プロシージャ

このプロシージャは、(FETCH_ROWS コールで) 行をフェッチする列に対してコレクションを定義します。このプロシージャによって、ユーザーは単一の SELECT 文から、行を一括してフェッチできます。1 回のフェッチ・コールで、PL/SQL の集計オブジェクトに多数の行をフェッチできます。

行をフェッチすると、それらの行は COLUMN_VALUE コールを実行するまで DBMS_SQL バッファにコピーされ、COLUMN_VALUE コールの実行時点で、引数として渡された表にコピーされます。

コレクション用のスカラー・タイプと LOB タイプ

ローカル変数を、次のいずれかの表項目タイプとして宣言し、DBMS_SQL を使用して、任意の行数をその中にフェッチできます (これらの表項目タイプは、BIND_ARRAY プロシージャに指定できるタイプと同じです)。

```
type Number_Table   IS TABLE OF NUMBER         INDEX BY BINARY_INTEGER;
type Varchar2_Table IS TABLE OF VARCHAR2(2000) INDEX BY BINARY_INTEGER;
type Date_Table     IS TABLE OF DATE           INDEX BY BINARY_INTEGER;
type Blob_Table     IS TABLE OF BLOB           INDEX BY BINARY_INTEGER;
type Clob_Table     IS TABLE OF CLOB           INDEX BY BINARY_INTEGER;
type Bfile_Table    IS TABLE OF BFILE          INDEX BY BINARY_INTEGER;
type Urowid_Table   IS TABLE OF UROWID         INDEX BY BINARY_INTEGER;
```

構文

```
DBMS_SQL.DEFINE_ARRAY (
   c           IN INTEGER,
   position    IN INTEGER,
   <table_variable>    IN <datatype>
   cnt         IN INTEGER,
   lower_bnd   IN INTEGER);
```

<table_variable> とそれに対応する <datatype> は、次のいずれかの組合せになります。

```
<num_tab>      Number_Table
<vchr2_tab>    Varchar2_Table
<date_tab>     Date_Table
<blob_tab>     Blob_Table
<clob_tab>     Clob_Table
<bfile_tab>    Bfile_Table
<urowid_tab>   Urowid_Table
```

DEFINE_ARRAY は、異なるデータ・タイプを受け入れるためにオーバーロードされていることに注意してください。

プラグマ

```
pragma restrict_references(define_array,RNDS,WNDS);
```

後続の FETCH_ROWS コールが cnt 行をフェッチします。COLUMN_VALUE がコールされると、これらの行は lower_bound、lower_bound+1、lower_bound+2 のように配置されます。行が送られてくる間、ユーザーは FETCH_ROWS コールまたは COLUMN_VALUE コールを継続して発行します。行は、COLUMN_VALUE コールに引数として指定した表内に蓄積されます。

パラメータ

表 69-6 DEFINE_ARRAY プロシージャのパラメータ

パラメータ	説明
c	配列をバインドするカーソルの ID 番号。
position	定義している配列内にある列の相対位置。 文の最初の列は位置 1 です。
table_variable	<datatype> として宣言されたローカル変数。
cnt	フェッチする行数。
lower_bnd	結果のコレクションへのコピーは、この下限の索引から開始されます。

行数（cnt）には 0（ゼロ）より大きい整数を指定する必要があります。それ以外の値が指定されると、例外が発生します。lower_bound は、正の数、負の数または 0（ゼロ）でもかまいません。DEFINE_ARRAY コールが発行された問合せに、配列バインドを含めることはできません。

> **関連項目：** コレクションの定義方法の例は、69-15 ページの「例 6 および 7: 配列の定義」を参照してください。

DEFINE_COLUMN_LONG プロシージャ

このプロシージャは、SELECT カーソルに対して LONG 列を定義します。定義されている列は、指定のカーソルの文の SELECT リスト内での相対位置によって識別されます。COLUMN 値のタイプによって、定義されている列のタイプが決まります。

構文

```
DBMS_SQL.DEFINE_COLUMN_LONG (
   c              IN INTEGER,
   position       IN INTEGER);
```

パラメータ

表 69-7 DEFINE_COLUMN_LONG プロシージャのパラメータ

パラメータ	説明
c	選択対象に定義されている行のカーソルの ID 番号。
position	定義している行内にある列の相対位置。 文の最初の列は位置 1 です。

EXECUTE ファンクション

このファンクションは、指定のカーソルを実行します。このファンクションはカーソルの ID 番号を受け入れて、処理された行数を戻します。戻り値は、DDL 文を含めて、INSERT 文、UPDATE 文および DELETE 文に対してのみ有効で、戻り値が未定義の場合は無視されます。

構文

```
DBMS_SQL.EXECUTE (
   c   IN INTEGER)
  RETURN INTEGER;
```

パラメータ

表 69-8 EXECUTE ファンクションのパラメータ

パラメータ	説明
c	実行するカーソルのカーソル ID 番号。

EXECUTE_AND_FETCH ファンクション

このファンクションは、指定のカーソルを実行して行をフェッチします。このファンクションは、EXECUTE をコールしてから FETCH_ROWS をコールするのと同じ機能を提供します。ただし、リモート・データベースに対して使用する場合は、EXECUTE_AND_FETCH をコールした方がネットワークのラウンドトリップ数を低減できます。

EXECUTE_AND_FETCH ファンクションは、実際にフェッチされた行数を戻します。

構文

```
DBMS_SQL.EXECUTE_AND_FETCH (
   c               IN INTEGER,
   exact           IN BOOLEAN DEFAULT FALSE)
  RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(execute_and_fetch,WNDS);
```

パラメータ

表 69-9 EXECUTE_AND_FETCH ファンクションのパラメータ

パラメータ	説明
c	実行してフェッチするカーソルのカーソル ID 番号。
exact	問合せで実際に一致する行数が 1 以外の場合は、TRUE に設定すると、例外が発生します。 **注意 :** LONG 列に対して、exact パラメータを TRUE に設定するオプションはサポートされていません。 例外が発生しても、行はフェッチされ、使用可能です。

FETCH_ROWS ファンクション

このファンクションは、指定のカーソルから行をフェッチします。FETCH_ROWS は、フェッチする行が残っているかぎり、繰り返しコールできます。これらの行はバッファに取り出し、FETCH_ROWS への各コール後に、COLUMN_VALUE をコールして各列ごとに読み込む必要があります。

FETCH_ROWS ファンクションは、フェッチするカーソルの ID 番号を受け入れて、実際にフェッチされた行数を戻します。

構文

```
DBMS_SQL.FETCH_ROWS (
   c               IN INTEGER)
  RETURN INTEGER;
```

パラメータ

表 69-10 FETCH_ROWS ファンクションのパラメータ

パラメータ	説明
c	ID 番号。

プラグマ

```
pragma restrict_references(fetch_rows,WNDS);
```

COLUMN_VALUE プロシージャ

このプロシージャは、指定したカーソル内の指定の位置にあるカーソル要素の値を戻します。このプロシージャは、FETCH_ROWS をコールしてフェッチしたデータへのアクセスに使用されます。

構文

```
DBMS_SQL.COLUMN_VALUE (
   c                 IN  INTEGER,
   position          IN  INTEGER,
   value             OUT <datatype>
 [,column_error      OUT NUMBER]
 [,actual_length     OUT INTEGER]);
```

<datatype> は、次のいずれかのタイプである必要があります。

```
NUMBER
DATE
VARCHAR2 CHARACTER SET ANY_CS
BLOB
CLOB CHARACTER SET ANY_CS
BFILE
UROWID
```

注意： 大カッコ [ ] は、オプション・パラメータを示します。

関連項目： 『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』

プラグマ

```
pragma restrict_references(column_value,RNDS,WNDS);
```

COLUMN_VALUE プロシージャでは、次の構文もサポートされます。

```
DBMS_SQL.COLUMN_VALUE(
   c                 IN  INTEGER,
   position          IN  INTEGER,
   <table_variable>  IN  <datatype>);
```

<table_variable> とそれに対応する <datatype> は、次のいずれかの組合せが可能です。

```
<num_tab>      Number_Table
<vchr2_tab>    Varchar2_Table
<date_tab>     Date_Table
<blob_tab>     Blob_Table
<clob_tab>     Clob_Table
<bfile_tab>    Bfile_Table
<urowid_tab>   Urowid_Table
```

CHAR、RAW および ROWID データを含んだ列では、次のバリエーションを構文で使用できます。

```
DBMS_SQL.COLUMN_VALUE_CHAR (
   c               IN  INTEGER,
   position        IN  INTEGER,
   value           OUT CHAR CHARACTER SET ANY_CS
 [,column_error    OUT NUMBER]
 [,actual_length   OUT INTEGER]);

DBMS_SQL.COLUMN_VALUE_RAW (
   c               IN  INTEGER,
   position        IN  INTEGER,
   value           OUT RAW
 [,column_error    OUT NUMBER]
 [,actual_length   OUT INTEGER]);

DBMS_SQL.COLUMN_VALUE_ROWID (
   c               IN  INTEGER,
   position        IN  INTEGER,
   value           OUT ROWID
 [,column_error    OUT NUMBER]
 [,actual_length   OUT INTEGER]);
```

パラメータ

表 69-11 COLUMN_VALUE プロシージャのパラメータ

パラメータ	説明
c	値をフェッチするカーソルの ID 番号。
position	カーソル内の列の相対位置。 文の最初の列は位置 1 です。
value	指定した列と行の値を戻します。 指定した行番号がフェッチされた行数の合計より大きい場合は、エラー・メッセージが発生します。 この出力パラメータのタイプが、DEFINE_COLUMN へのコールで定義されている値の実際のタイプと異なる場合、例外エラー ORA-06562、inconsistent_type が発生します。
table_variable	<datatype> として宣言されたローカル変数。
column_error	指定した列値のエラー・コードを戻します。
actual_length	指定した列内の値の（切捨て前の）実際の長さ。

例外 :

指定した OUT パラメータの value が、実際の値のタイプと異なる場合は、inconsistent_type（ORA-06562）が発生します。このタイプは、DEFINE_COLUMN プロシージャをコールして列を定義したときに指定したタイプです。

COLUMN_VALUE_LONG プロシージャ

このプロシージャは、LONG 列の値の一部を取得します。

構文

```
DBMS_SQL.COLUMN_VALUE_LONG (
   c            IN  INTEGER,
   position     IN  INTEGER,
   length       IN  INTEGER,
   offset       IN  INTEGER,
   value        OUT VARCHAR2,
   value_length OUT INTEGER);
```

プラグマ

```
pragma restrict_references(column_value_long,RNDS,WNDS);
```

パラメータ

表 69-12 COLUMN_VALUE_LONG プロシージャのパラメータ

パラメータ	説明
c	値を取得するカーソルのカーソル ID 番号。
position	値を取得する列の位置。
length	フェッチする LONG 値のバイト数。
offset	フェッチを開始するための LONG フィールドへのオフセット。
value	VARCHAR2 の列の値。
value_length	値に実際に戻されるバイト数。

VARIABLE_VALUE プロシージャ

このプロシージャは、指定のカーソルについて指定の変数の値を戻します。このプロシージャは、returning 句を使用して PL/SQL ブロックまたは DML 文内のバインド変数の値を戻すために使用されます。

構文

```
DBMS_SQL.VARIABLE_VALUE (
   c               IN  INTEGER,
   name            IN  VARCHAR2,
   value           OUT <datatype>);
```

<datatype> は、次のいずれかのタイプである必要があります。

```
NUMBER
DATE
VARCHAR2 CHARACTER SET ANY_CS
BLOB
CLOB CHARACTER SET ANY_CS
BFILE
UROWID
```

プラグマ

```
pragma restrict_references(variable_value,RNDS,WNDS);
```

VARIABLE_VALUE プロシージャでは、次の構文もサポートされます。

```
DBMS_SQL.VARIABLE_VALUE (
   c                IN  INTEGER,
   name             IN  VARCHAR2,
   <table_variable>  IN  <datatype>);
```

<table_variable> とそれに対応する <datatype> は、次のいずれかの組合せが可能です。

```
<num_tab>      Number_Table
<vchr2_tab>    Varchar2_Table
<date_tab>     Date_Table
<blob_tab>     Blob_Table
<clob_tab>     Clob_Table
<bfile_tab>    Bfile_Table
<urowid_tab>   Urowid_Table
```

CHAR、RAW および ROWID データを含んだ変数では、次のバリエーションを構文で使用できます。

```
DBMS_SQL.VARIABLE_VALUE_CHAR (
   c               IN  INTEGER,
   name            IN  VARCHAR2,
   value           OUT CHAR CHARACTER SET ANY_CS);

DBMS_SQL.VARIABLE_VALUE_RAW (
   c               IN  INTEGER,
   name            IN  VARCHAR2,
   value           OUT RAW);

DBMS_SQL.VARIABLE_VALUE_ROWID (
   c               IN  INTEGER,
   name            IN  VARCHAR2,
   value           OUT ROWID);
```

パラメータ

表 69-13 VARIABLE_VALUE プロシージャのパラメータ

パラメータ	説明
c	値を取得するカーソルの ID 番号。
name	値を検索する変数名。
value	指定した位置の変数の値を戻します。 この出力パラメータのタイプが、BIND_VARIABLE へのコールで定義されている値の実際のタイプと異なる場合、例外エラー ORA-06562、inconsistent_type が発生します。

IS_OPEN ファンクション

このファンクションは、指定のカーソルが現在オープンしているかどうかをチェックします。

構文

```
DBMS_SQL.IS_OPEN (
   c               IN INTEGER)
  RETURN BOOLEAN;
```

プラグマ

```
pragma restrict_references(is_open,RNDS,WNDS);
```

パラメータ

表 69-14 IS_OPEN ファンクションのパラメータ

パラメータ	説明
c	チェックするカーソルのカーソル ID 番号。

戻り値

表 69-15 IS_OPEN ファンクションの戻り値

戻り値	説明
TRUE	指定のカーソルは、現在オープンしています。
FALSE	指定のカーソルは、現在オープンしていません。

DESCRIBE_COLUMNS プロシージャ

このプロシージャは、DBMS_SQL によってオープンして解析されたカーソルの列の定義を表示します。

DESC_REC タイプ

DBMS_SQL パッケージでは、DESC_REC レコード・タイプを次のように宣言します。

```
type desc_rec is record (
   col_type            BINARY_INTEGER := 0,
   col_max_len         BINARY_INTEGER := 0,
   col_name            VARCHAR2(32)   := '',
   col_name_len        BINARY_INTEGER := 0,
   col_schema_name     VARCHAR2(32)   := '',
   col_schema_name_len BINARY_INTEGER := 0,
   col_precision       BINARY_INTEGER := 0,
   col_scale           BINARY_INTEGER := 0,
   col_charsetid       BINARY_INTEGER := 0,
   col_charsetform     BINARY_INTEGER := 0,
   col_null_ok         BOOLEAN        := TRUE);
```

パラメータ

表 69-16 DESC_REC タイプのパラメータ

パラメータ	説明
col_type	表示される列のタイプ。
col_max_len	列の最大長。
col_name	列の名前。
col_name_len	列名の長さ。
col_schema_name	列のタイプが定義されたスキーマの名前（オブジェクト・タイプの場合）。
col_schema_name_len	スキーマ名の長さ。
col_precision	数値の場合は、列の精度。
col_scale	数値の場合は、列の位取り。
col_charsetid	列のキャラクタ・セットの識別子。
col_charsetform	列のキャラクタ・セット・フォーム。
col_null_ok	列で NULL が可能な場合は、TRUE。

DESC_TAB タイプ

DESC_TAB タイプは、DESC_REC レコードの PL/SQL 表です。

```
type desc_tab is table of desc_rec index by BINARY_INTEGER;
```

ローカル変数を PL/SQL 表タイプである DESC_TAB として宣言し、次に、DESCRIBE_COLUMNS プロシージャをコールして、各列の説明を表に含めることができます。その際、すべての列の定義が表示されます。単一の列のみを表示することはできません。

構文

```
DBMS_SQL.DESCRIBE_COLUMNS (
   c              IN  INTEGER,
   col_cnt        OUT INTEGER,
   desc_t         OUT DESC_TAB);
```

パラメータ

表 69-17 DBMS_SQL.DESCRIBE_COLUMNS プロシージャのパラメータ

パラメータ	説明
c	表示される列のカーソルの ID 番号。
col_cnt	問合せの選択リストにある列数。
desc_t	DESC_REC の表。各 DESC_REC が問合せ内の列を説明します。

関連項目： DESCRIBE_COLUMNS の使用方法は、69-17 ページの「例 8: 列の定義の表示」を参照してください。

CLOSE_CURSOR プロシージャ

このファンクションは、指定のカーソルをクローズします。

構文

```
DBMS_SQL.CLOSE_CURSOR (
   c    IN OUT INTEGER);
```

プラグマ

```
pragma restrict_references(close_cursor,RNDS,WNDS);
```

パラメータ

表 69-18 CLOSE_CURSOR プロシージャのパラメータ

パラメータ	モード	説明
c	IN	クローズするカーソルの ID 番号。
c	OUT	カーソルは NULL に設定されています。 CLOSE_CURSOR をコールした後、カーソルに割り当てられたメモリーは解放され、そのカーソルからはフェッチできなくなります。

LAST_ERROR_POSITION ファンクション

このファンクションは、エラーが発生した SQL 文テキスト内のバイト・オフセットを戻します。SQL 文内の最初の文字は、位置 0（ゼロ）にあります。

構文

```
DBMS_SQL.LAST_ERROR_POSITION
   RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(last_error_position,RNDS,WNDS);
```

使用上の注意

このファンクションは、別の DBMS_SQL プロシージャまたはファンクションのコール前、かつ PARSE のコール後にコールしてください。

LAST_ROW_COUNT ファンクション

このファンクションは、フェッチされた累積行数を戻します。

構文

```
DBMS_SQL.LAST_ROW_COUNT
   RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(last_row_count,RNDS,WNDS);
```

使用上の注意

このファンクションは、FETCH_ROWS コールまたは EXECUTE_AND_FETCH コール後にコールしてください。EXECUTE コール後にコールすると、戻される値は 0（ゼロ）です。

LAST_ROW_ID ファンクション

このファンクションは、処理された最後の行の ROWID を戻します。

構文

```
DBMS_SQL.LAST_ROW_ID
   RETURN ROWID;
```

プラグマ

```
pragma restrict_references(last_row_id,RNDS,WNDS);
```

使用上の注意

このファンクションは、FETCH_ROWS コールまたは EXECUTE_AND_FETCH コール後にコールしてください。

LAST_SQL_FUNCTION_CODE ファンクション

このファンクションは、文の SQL ファンクション・コードを戻します。これらのコードは、『Oracle Call Interface プログラマーズ・ガイド』に一覧があります。

構文

```
DBMS_SQL.LAST_SQL_FUNCTION_CODE
   RETURN INTEGER;
```

プラグマ

```
pragma restrict_references(last_sql_function_code,RNDS,WNDS);
```

使用上の注意

このファンクションは、SQL 文の実行直後にコールする必要があります。それ以外の場合は、戻り値が未定義になります。

70

DBMS_STATS

DBMS_STATS を使用すると、データベース・オブジェクト用に収集したオプティマイザの統計情報を表示および変更できます。統計情報は、この目的のためにユーザーのスキーマに作成したディクショナリまたは表に常駐させることができます。また、このパッケージを使用して、表およびドメイン索引に関するユーザー定義の統計情報を収集および管理できます。たとえば、関連性が定義されている列に対して DELETE_COLUMN_STATS プロシージャが起動されると、標準的な統計情報に加え、その列にあるユーザー定義の統計情報も削除されます。

コストベースのオプティマイザに影響を与える統計情報は、ディクショナリに格納されている統計情報のみです。また、DBMS_STATS を使用して、並列的に統計情報を収集できます。

この章では、次の項目について説明します。

DBMS_STATS の使用方法

DBMS_STATS サブプログラムには、次に示す一般的な機能があります。

- 統計情報の設定または取得
- 統計情報の転送
- オプティマイザ統計情報の収集

ほとんどの DBMS_STATS プロシージャには、statown、stattab および statid の 3 つのパラメータが含まれています。これらのパラメータによって、ユーザーはオプティマイザに影響を与えない自分自身の表（ディクショナリ外）に統計情報を格納できます。したがって、ユーザーは統計情報のセットをメンテナンスおよび試用することができます。

stattab パラメータで、統計情報を保持する表の名前を指定します。この表は、（statown パラメータが指定されていないかぎり）統計情報が収集されるオブジェクトと同じスキーマ内に常駐しているとみなされます。異なる stattab 識別子で複数の表を作成し、統計情報セットを別々に保持できます。

さらに、statid パラメータを使用して 1 つの stattab 内で複数の統計情報セットをメンテナンスできるため、ユーザーのスキーマが混乱するのを回避できます。

SET および GET プロシージャについて、stattab が準備されていない場合（つまり、NULL の場合）、操作はディクショナリにある統計情報に対して直接行われます。したがって、ディクショナリを直接変更する場合は、統計表の作成は不要です。ただし、stattab が NULL でない場合、SET または GET 操作は、指定したユーザー統計表に対して行われ、ディクショナリに対しては行われません。

DBMS_STATS サブプログラムによって、オブジェクトの統計情報が変更または削除されると、依存しているすべてのカーソルは無効化され（デフォルト）、新しい統計情報がただちに有効になるように、対応する文が次回のコンパイル対象となります。この動作は、no_invalidate 引数を使用して変更できます。

ユーザー定義の統計情報

DBMS_STATS は、ユーザー定義の統計情報に関する操作をサポートしています。ドメイン索引または列が（associate を使用して）統計タイプに関連付けられている場合は、索引または列に関する操作によって、ユーザー定義の統計情報が操作されます。たとえば、GATHER_INDEX_STATS インタフェースを使用して（統計タイプに関連付けられている）ドメイン索引の統計情報を収集すると、関連付けられている統計タイプに対応するユーザー定義の統計情報収集メソッドが起動されます。同様に、削除、転送、インポートおよびエクスポートの各操作によって、ユーザー定義の統計情報が操作されます。

ユーザー定義の統計情報に対する SET および GET 操作も、列および索引に対する特別バージョンの SET および GET インタフェースを使用してサポートされます。

このパッケージにおける次のプロシージャは、現行のトランザクションをコミットし、操作を実行してから再度コミットします。

- SET_*
- DELETE_*
- EXPORT_*
- IMPORT_*
- GATHER_*
- *_STAT_TABLE

型

最小値と最大値およびヒストグラム終点のタイプは、次のとおりです。

```
TYPE numarray  IS VARRAY(256) OF NUMBER;
TYPE datearray IS VARRAY(256) OF DATE;
TYPE chararray IS VARRAY(256) OF VARCHAR2(4000);
TYPE rawarray  IS VARRAY(256) OF RAW(2000);

type StatRec is record (
  epc    NUMBER,
  minval RAW(2000),
  maxval RAW(2000),
  bkvals NUMARRAY,
  novals NUMARRAY);
```

失効した表のリストのタイプは、次のとおりです。

```
type ObjectElem is record (
  ownname     VARCHAR2(30),     -- owner
  objtype     VARCHAR2(6),      -- 'TABLE' or 'INDEX'
  objname     VARCHAR2(30),     -- table/index
  partname    VARCHAR2(30),     -- partition
  subpartname VARCHAR2(30),     -- subpartition
  confidence  NUMBER);          -- not used
type ObjectTab is TABLE of ObjectElem;
```

自動サンプル・サイズのアルゴリズムを使用することを示すために、次の定数が使用されます。

```
AUTO_SAMPLE_SIZE CONSTANT NUMBER;
```

システムにおける初期化パラメータに基づいてデフォルトの並列度を判別するために、次の定数が使用されます。

```
DEFAULT_DEGREE CONSTANT NUMBER;
```

統計情報の設定または取得

次のプロシージャを使用して、個別の列、索引および表に関連する統計情報を格納および取り出します。

```
PREPARE_COLUMN_VALUES
SET_COLUMN_STATS
SET_INDEX_STATS
SET_SYSTEM_STATS
SET_TABLE_STATS
```

ユーザー定義の統計情報を設定する特別バージョンの SET_*_STATS プロシージャでは、次の情報（提供された場合）がディレクトリまたは外部統計表に格納されます。

- ユーザー定義の統計情報（extstats）
- 統計タイプのスキーマ名（statsschema）
- 統計タイプ名（statsname）

ユーザー定義の統計情報と対応する統計タイプは、USTATS$ ディクショナリ表に挿入されます。ユーザー定義の統計情報は、統計タイプ名の指定なしに指定できます。

```
CONVERT_RAW_VALUE
GET_COLUMN_STATS
GET_INDEX_STATS
GET_SYSTEM_STATS
GET_TABLE_STATS
```

特別バージョンの GET_*_STATS プロシージャは、ユーザー定義の統計情報および統計タイプの所有者と名前を、指定したスキーマ・オブジェクトに対応する OUT 引数として戻します。ユーザー定義の統計情報が収集されていない場合は、NULL 値が戻ります。

```
DELETE_COLUMN_STATS
DELETE_INDEX_STATS
DELETE_SYSTEM_STATS
DELETE_TABLE_STATS
DELETE_SCHEMA_STATS
DELETE_DATABASE_STATS
```

DELETE_* プロシージャは、ユーザー定義の統計情報および指定したスキーマ・オブジェクトに関する標準的な統計情報を削除します。

統計情報の転送

次のプロシージャを使用して、ディクショナリからユーザー統計表に（export_*）、およびユーザー統計表からディクショナリに（import_*）統計情報を転送します。

```
CREATE_STAT_TABLE
DROP_STAT_TABLE
```

CREATE_STAT_TABLE は、ユーザー定義の統計情報および統計タイプのオブジェクト番号を保持できます。

```
EXPORT_COLUMN_STATS
EXPORT_INDEX_STATS
EXPORT_SYSTEM_STATS
EXPORT_TABLE_STATS
EXPORT_SCHEMA_STATS
EXPORT_DATABASE_STATS

IMPORT_COLUMN_STATS
IMPORT_INDEX_STATS
IMPORT_SYSTEM_STATS
IMPORT_TABLE_STATS
IMPORT_SCHEMA_STATS
IMPORT_DATABASE_STATS
```

IMPORT_* プロシージャは、ユーザー定義の統計情報も含めた統計情報を stattab 表から取得して、ディクショナリに格納します。SET_*_STATS および GET_*_STATS インタフェースはユーザー定義の統計情報をサポートしているため、このインタフェースを使用すると、ユーザー定義の統計情報を別のデータベースにコピーできます。

オプティマイザ統計情報の収集

次のプロシージャを使用して、特定のクラスのオプティマイザ統計情報を収集し、ANALYZE コマンドの潜在的なパフォーマンスを向上させます。

```
GATHER_INDEX_STATS
GATHER_TABLE_STATS
GATHER_SCHEMA_STATS
GATHER_DATABASE_STATS
GATHER_SYSTEM_STATS
```

GATHER_* プロシージャは、列およびドメイン索引に関するユーザー定義の統計情報も収集します。

statown、stattab および statid の各パラメータは、パッケージに対して、新規の統計情報を収集する前に、指定した表内の現行の統計情報をバックアップするように指示します。

関連オブジェクトに十分な統計情報がある場合、導出オブジェクトの統計情報を生成するために、Oracle は次のプロシージャも提供します。

GENERATE_STATS

DBMS_STATS サブプログラムの要約

表 70-1 DBMS_STATS のサブプログラム

サブプログラム	説明
「PREPARE_COLUMN_VALUES プロシージャ」70-9 ページ	ユーザー指定の最小値、最大値およびヒストグラム終点のデータ・タイプ固有の値を、SET_COLUMN_STATS を使用して将来格納するために Oracle の内部表記に変換します。
「SET_COLUMN_STATS プロシージャ」70-12 ページ	列に関連する情報を設定します。
「SET_INDEX_STATS プロシージャ」70-14 ページ	索引に関連する情報を設定します。
「SET_SYSTEM_STATS プロシージャ」70-16 ページ	システムの統計情報を設定します。
「SET_TABLE_STATS プロシージャ」70-17 ページ	表に関連する情報を設定します。
「CONVERT_RAW_VALUE プロシージャ」70-19 ページ	最大値または最小値の内部表記を、データ・タイプ固有の値に変換します。
「GET_COLUMN_STATS プロシージャ」70-20 ページ	列に関連するすべての情報を取得します。
「GET_INDEX_STATS プロシージャ」70-22 ページ	索引に関連するすべての情報を取得します。
「GET_SYSTEM_STATS プロシージャ」70-24 ページ	stattab（stattab が NULL の場合はディクショナリ）からシステムの統計情報を取得します。
「GET_TABLE_STATS プロシージャ」70-26 ページ	表に関連するすべての情報を取得します。
「DELETE_COLUMN_STATS プロシージャ」70-27 ページ	列に関連する統計情報を削除します。
「DELETE_INDEX_STATS プロシージャ」70-29 ページ	索引に関連する統計情報を削除します。
「DELETE_SYSTEM_STATS プロシージャ」70-30 ページ	システムの統計情報を削除します。

表 70-1 DBMS_STATS のサブプログラム（続き）

サブプログラム	説明
「DELETE_TABLE_STATS プロシージャ」70-31 ページ	表に関連する統計情報を削除します。
「DELETE_SCHEMA_STATS プロシージャ」70-32 ページ	スキーマに関連する統計情報を削除します。
「DELETE_DATABASE_STATS プロシージャ」70-33 ページ	データベース全体に関する統計情報を削除します。
「CREATE_STAT_TABLE プロシージャ」70-34 ページ	統計情報を保持できる ownname のスキーマに stattab の名前で表を作成します。
「DROP_STAT_TABLE プロシージャ」70-35 ページ	CREATE_STAT_TABLE で作成したユーザー統計表を削除します。
「EXPORT_COLUMN_STATS プロシージャ」70-36 ページ	特定の列に関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。
「EXPORT_INDEX_STATS プロシージャ」70-37 ページ	特定の索引に関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。
「EXPORT_SYSTEM_STATS プロシージャ」70-38 ページ	システムの統計情報を取り出し、ユーザー統計表に格納します。
「EXPORT_TABLE_STATS プロシージャ」70-39 ページ	特定の表に関する統計情報を取り出し、ユーザー統計表に格納します。
「EXPORT_SCHEMA_STATS プロシージャ」70-40 ページ	ownname で識別されるスキーマ内のすべてのオブジェクトに関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。
「EXPORT_DATABASE_STATS プロシージャ」70-41 ページ	データベース内のすべてのオブジェクトに関する統計情報を取り出し、statown.stattab で識別されるユーザー統計表に格納します。
「IMPORT_COLUMN_STATS プロシージャ」70-42 ページ	stattab で識別されるユーザー統計表から特定の列に関する統計情報を取り出し、ディクショナリに格納します。
「IMPORT_INDEX_STATS プロシージャ」70-43 ページ	stattab で識別されるユーザー統計表から特定の索引に関する統計情報を取り出し、ディクショナリに格納します。
「IMPORT_SYSTEM_STATS プロシージャ」70-44 ページ	システムの統計情報をユーザー統計表から取り出し、ディクショナリに格納します。
「IMPORT_TABLE_STATS プロシージャ」70-45 ページ	stattab で識別されるユーザー統計表から特定の表に関する統計情報を取り出し、ディクショナリに格納します。

表 70-1 DBMS_STATS のサブプログラム（続き）

サブプログラム	説明
「IMPORT_SCHEMA_STATS プロシージャ」70-46 ページ	ownname で識別されるスキーマ内のすべてのオブジェクトに関する統計情報をユーザー統計表から取り出し、ディクショナリに格納します。
「IMPORT_DATABASE_STATS プロシージャ」70-47 ページ	データベース内のすべてのオブジェクトに関する統計情報をユーザー統計表から取り出し、ディクショナリに格納します。
「GATHER_INDEX_STATS プロシージャ」70-48 ページ	索引の統計情報を収集します。
「GATHER_TABLE_STATS プロシージャ」70-50 ページ	表と列（および索引）の統計情報を収集します。
「GATHER_SCHEMA_STATS プロシージャ」70-52 ページ	スキーマ内のすべてのオブジェクトに関する統計情報を収集します。
「GATHER_DATABASE_STATS プロシージャ」70-56 ページ	データベース内のすべてのオブジェクトに関する統計情報を収集します。
「GATHER_SYSTEM_STATS プロシージャ」70-60 ページ	システムの統計情報を収集します。
「GENERATE_STATS プロシージャ」70-61 ページ	関連するオブジェクトに関して以前に収集した統計情報から、オブジェクトの統計情報を生成します。
「FLUSH_DATABASE_MONITORING_INFO プロシージャ」70-62 ページ	メモリー内のすべての表の監視情報をディクショナリにフラッシュします。
「ALTER_SCHEMA_TAB_MONITORING プロシージャ」70-64 ページ	スキーマにおけるすべての表の機能を監視する DML を使用可能または使用禁止にします。ただし、スナップショット・ログおよび表では監視がサポートされていないため対象外です。
「ALTER_DATABASE_TAB_MONITORING プロシージャ」70-64 ページ	データベースにおけるすべての表の機能を監視する DML を使用可能または使用禁止にします。ただし、スナップショット・ログおよび表では監視がサポートされていないため対象外です。

PREPARE_COLUMN_VALUES プロシージャ

このプロシージャは、ユーザー指定の最小値、最大値およびヒストグラム終点のデータ・タイプ固有の値を、SET_COLUMN_STATS を使用して将来格納するために Oracle の内部表記に変換します。

構文

```
DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec     IN OUT StatRec,
   charvals        CHARARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec      IN OUT StatRec,
   datevals         DATEARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec     IN OUT StatRec,
   numvals         NUMARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES (
   srec     IN OUT StatRec,
   rawvals         RAWARRAY);

DBMS_STATS.PREPARE_COLUMN_VALUES_NVARCHAR (
   srec  IN OUT StatRec,
   nvmin        NVARCHAR2,
   nvmax        NVARCHAR2);

DBMS_STATS.PREPARE_COLUMN_VALUES_ROWID (
   srec  IN OUT StatRec,
   rwmin        ROWID,
   rwmax        ROWID);
```

プラグマ

```
pragma restrict_references(prepare_column_values, WNDS, RNDS, WNPS, RNPS);
pragma restrict_references(prepare_column_values_nvarchar, WNDS, RNDS, WNPS, RNPS);
pragma restrict_references(prepare_column_values_rowid, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 70-2 PREPARE_COLUMN_VALUES プロシージャのパラメータ

パラメータ	説明
srec.epc	charvals、datevals、numvals または rawvals で指定される値の数。この値は 2 ～ 256 の範囲で指定し、ヒストグラム情報を認めないプロシージャ（nvarchar と rowid）では 2 を設定する必要があります。 最初に対応する配列のエントリは列の最小値を含み、最後のエントリは最大値を含みます。2 つを超えるエントリがある場合、その他のエントリは、残りの高さ調整ヒストグラムまたは頻度ヒストグラムのエンドポイント値（小さい値から大きい値に順序立てられた中間値を持つ）を含みます。この値は圧縮のために調整できるため、戻り値は、SET_COLUMN_STATS へのコールに対して現状のままになります。
srec.bkvals	頻度分布が必要な場合、この配列には、charvals、datevals、numvals または rawvals で指定されている各個別値の発生回数が含まれています。そうでない場合、この配列は単なる出力パラメータであり、このプロシージャのコール時には NULL が設定されている必要があります。

表 70-3 に、データ・タイプ固有の入力パラメータ（1 つを使用）を示します。

表 70-3 データ・タイプ固有の入力パラメータ

型	説明
charvals	列の型が文字ベースの場合の値の配列。各文字列の最初の 32 バイトまでが使用されます。配列のエントリ数は、2 ～ 256 の範囲である必要があります。データ・タイプが固定の CHAR である場合、適切な正規化を行うには、文字列にスペースを追加して 15 文字にする必要があります。
datevals	列の型が日付ベースの場合の値の配列。
numvals	列の型が数値ベースの場合の値の配列。
rawvals	列の型が RAW の場合の値の配列。各文字列の最初の 32 バイトまでが使用されます。

表 70-3 データ・タイプ固有の入力パラメータ

型	説明
nvmin、nvmax	列の型が各国語キャラクタ・セット（NLS）の場合の最大値と最小値。この型の列について、ヒストグラム情報は提供できません。データ・タイプが固定の CHAR である場合、適切な正規化を行うには、文字列にスペースを追加して 15 文字にする必要があります。
rwmin、rwmax	列タイプが ROWID の場合の最大値と最小値。このタイプの列に、ヒストグラム情報は提供されません。

出力パラメータ

表 70-4 PREPARE_COLUMN_VALUES プロシージャの出力パラメータ

パラメータ	説明
srec.minval	SET_COLUMN_STATS へのコールでの使用に適した最小値の内部表記。
srec.maxval	SET_COLUMN_STATS へのコールでの使用に適した最大値の内部表記。
srec.bkvals	SET_COLUMN_STATS へのコールでの使用に適した配列。
srec.novals	SET_COLUMN_STATS へのコールでの使用に適した配列。

例外

ORA-20001: 入力値が無効か、または矛盾しています。

SET_COLUMN_STATS プロシージャ

このプロシージャは、列に関連する情報を設定します。ユーザー定義の統計情報を処理するこのバージョンでは、指定の統計タイプは、実際のユーザー定義の統計情報に加え、ディクショナリに格納されるタイプでもあります。この統計タイプが NULL の場合は、索引または列に関連付けられている統計タイプが格納されます。

構文

標準統計情報には、次の構文を使用します。

```
DBMS_STATS.SET_COLUMN_STATS (
   ownname       VARCHAR2,
   tabname       VARCHAR2,
   colname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   distcnt       NUMBER DEFAULT NULL,
   density       NUMBER DEFAULT NULL,
   nullcnt       NUMBER DEFAULT NULL,
   srec          StatRec DEFAULT NULL,
   avgclen       NUMBER DEFAULT NULL,
   flags         NUMBER DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

ユーザー定義統計情報には、次の構文を使用します。

```
DBMS_STATS.SET_COLUMN_STATS (
   ownname       VARCHAR2,
   tabname       VARCHAR2,
   colname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   ext_stats     RAW,
   stattypown    VARCHAR2 DEFAULT NULL,
   stattypname   VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-5 SET_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	統計情報を格納する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで格納されます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリ内に直接格納されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
ext_stats	ユーザー定義の統計情報。
stattypown	統計タイプのスキーマ。
stattypname	統計タイプの名前。
distcnt	個別値の数。
density	列密度。この値が NULL で、distcnt が NULL でない場合、密度は distcnt から導出されます。
nullcnt	NULL の数。
srec	PREPARE_COLUMN_VALUES または GET_COLUMN_STATS へのコールで入力された StatRec 構造。
avgclen	列の平均長（バイト単位）。
flags	Oracle 内部で使用（NULL のままにします）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効か、または矛盾しています。

SET_INDEX_STATS プロシージャ

このプロシージャは、索引に関連する情報を設定します。ユーザー定義の統計情報を処理するこのバージョンでは、指定の統計タイプは、実際のユーザー定義の統計情報に加え、ディクショナリに格納されるタイプでもあります。この統計タイプが NULL の場合は、索引または列に関連付けられている統計タイプが格納されます。

構文

標準統計情報には、次の構文を使用します。

```
DBMS_STATS.SET_INDEX_STATS (
   ownname       VARCHAR2,
   indname       VARCHAR2,
   partname      VARCHAR2  DEFAULT NULL,
   stattab       VARCHAR2  DEFAULT NULL,
   statid        VARCHAR2  DEFAULT NULL,
   numrows       NUMBER    DEFAULT NULL,
   numlblks      NUMBER    DEFAULT NULL,
   numdist       NUMBER    DEFAULT NULL,
   avglblk       NUMBER    DEFAULT NULL,
   avgdblk       NUMBER    DEFAULT NULL,
   clstfct       NUMBER    DEFAULT NULL,
   indlevel      NUMBER    DEFAULT NULL,
   flags         NUMBER    DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE,
   guessq NUMBER DEFAULT NULL);
```

ユーザー定義統計情報には、次の構文を使用します。

```
DBMS_STATS.SET_INDEX_STATS (
   ownname       VARCHAR2,
   indname       VARCHAR2,
   partname      VARCHAR2  DEFAULT NULL,
   stattab       VARCHAR2  DEFAULT NULL,
   statid        VARCHAR2  DEFAULT NULL,
   ext_stats     RAW,
   stattypown    VARCHAR2 DEFAULT NULL,
   stattypname   VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE,
```

パラメータ

表 70-6 SET_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	統計情報を格納する索引パーティションの名前。索引がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな索引レベルで格納されます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリ内に直接格納されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
ext_stats	ユーザー定義の統計情報。
stattypown	統計タイプのスキーマ。
stattypname	統計タイプの名前。
numrows	索引（パーティション）内の行数。
numlblks	索引（パーティション）内のリーフ・ブロックの数。
numdist	索引（パーティション）内の個別キーの数。
avglblk	この索引（パーティション）について各個別キーが出現するリーフ・ブロックの平均整数値。この値が提供されない場合、この値は numlblks と numdist から導出されます。
avgdblk	この索引（パーティション）について個別キーが指す表内のデータ・ブロックの平均整数値。この値が提供されない場合、この値は clstfct と numdist から導出されます。
clstfct	all_indexes ビューの clustering_factor 列の説明を参照してください。
indlevel	索引（パーティション）の高さ。
flags	Oracle 内部で使用（NULL のままにします）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。
guessq	推測品質。all_indexes ビューの pct_direct_access 列の説明を参照してください。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

SET_SYSTEM_STATS プロシージャ

このプロシージャは、システムの統計情報を設定します。

構文

```
DBMS_STATS.SET_SYSTEM_STATS (
   pname          VARCHAR2,
   pvalue         NUMBER,
   stattab   IN   VARCHAR2 DEFAULT NULL,
   statid    IN   VARCHAR2 DEFAULT NULL,
   statown   IN   VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-7 SET_SYSTEM_STATS プロシージャのパラメータ

パラメータ	説明
pname	取得するパラメータ名。次の値の 1 つが入ります。 ■ sreadtim －シングル・ブロックの読込み（ランダム読込み）平均時間（ミリ秒単位） ■ mreadtim － mbrc ブロックを 1 回で読み込む（順次読取り）平均時間（ミリ秒単位） ■ cpuspeed － 1 秒当たりの平均 CPU サイクル数（100 万単位） ■ mbrc －順次読取りの平均マルチブロック読込み件数（ブロック単位） ■ maxthr －最大 I/O システム・スループット（1 秒当たりのバイト数単位） ■ slavethr －平均スレーブ I/O スループット（1 秒当たりのバイト数単位）
pvalue	取得するパラメータ値。

表 70-7 SET_SYSTEM_STATS プロシージャのパラメータ（続き）

パラメータ	説明
stattab	統計情報を取得するユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから取得されます。
statid	stattab に保存された統計情報に関連付けられた、オプションの識別子。
statown	stattab を含んだスキーマ（ユーザーのスキーマと異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

ORA-20002: ユーザー統計表が不正です。アップグレードする必要があります。

ORA-20003: システムの統計情報を設定できません。

ORA-20004: パラメータが存在しません。

SET_TABLE_STATS プロシージャ

このプロシージャは、表に関連する情報を設定します。

構文

```
DBMS_STATS.SET_TABLE_STATS (
   ownname  VARCHAR2,
   tabname  VARCHAR2,
   partname VARCHAR2 DEFAULT NULL,
   stattab  VARCHAR2 DEFAULT NULL,
   statid   VARCHAR2 DEFAULT NULL,
   numrows  NUMBER   DEFAULT NULL,
   numblks  NUMBER   DEFAULT NULL,
   avgrlen  NUMBER   DEFAULT NULL,
   flags    NUMBER   DEFAULT NULL,
   statown  VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-8 SET_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	表の名前。
partname	統計情報を格納する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで格納されます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリ内に直接格納されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
numrows	表（パーティション）内の行数。
numblks	表（パーティション）が占有するブロックの数。
avgrlen	表（パーティション）の行の平均長。
flags	Oracle 内部で使用（NULL のままにします）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

CONVERT_RAW_VALUE プロシージャ

このプロシージャは、最大値または最小値の内部表記をデータ・タイプ固有の値に変換します。GET_COLUMN_STATS または PREPARE_COLUMN_VALUES で入力された StatRec 構造の minval フィールドと maxval フィールドの値が、有効な入力値です。

構文

```
DBMS_STATS.CONVERT_RAW_VALUE (
   rawval     RAW,
   resval OUT VARCHAR2);

DBMS_STATS.CONVERT_RAW_VALUE (
   rawval     RAW,
   resval OUT DATE);

DBMS_STATS.CONVERT_RAW_VALUE (
   rawval     RAW,
   resval OUT NUMBER);

DBMS_STATS.CONVERT_RAW_VALUE_NVARCHAR (
   rawval     RAW,
   resval OUT NVARCHAR2);

DBMS_STATS.CONVERT_RAW_VALUE_ROWID (
   rawval     RAW,
   resval OUT ROWID);
```

プラグマ

```
pragma restrict_references(convert_raw_value, WNDS, RNDS, WNPS, RNPS);
pragma restrict_references(convert_raw_value_nvarchar, WNDS, RNDS, WNPS, RNPS);
pragma restrict_references(convert_raw_value_rowid, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 70-9 CONVERT_RAW_VALUE プロシージャのパラメータ

パラメータ	説明
rawval	列最大または列最小のデータ・タイプ固有の出力パラメータの内部表記。
resval	変換済みの型固有の値。

GET_COLUMN_STATS プロシージャ

このプロシージャは、列に関連するすべての情報を取得します。ユーザー定義の統計情報を処理するこのバージョンでは、戻される統計タイプは、ユーザー定義の統計情報に加え、格納されるタイプでもあります。

構文

標準統計情報には、次の構文を使用します。

```
DBMS_STATS.GET_COLUMN_STATS (
   ownname     VARCHAR2,
   tabname     VARCHAR2,
   colname     VARCHAR2,
   partname    VARCHAR2 DEFAULT NULL,
   stattab     VARCHAR2 DEFAULT NULL,
   statid      VARCHAR2 DEFAULT NULL,
   distcnt OUT NUMBER,
   density OUT NUMBER,
   nullcnt OUT NUMBER,
   srec    OUT StatRec,
   avgclen OUT NUMBER,
   statown     VARCHAR2 DEFAULT NULL);
```

ユーザー定義統計情報には、次の構文を使用します。

```
DBMS_STATS.GET_COLUMN_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   colname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   ext_stats   OUT  RAW,
   stattypown  OUT  VARCHAR2 DEFAULT NULL,
   stattypname OUT  VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-10 GET_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	統計情報を取得する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで取り出されます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接取り出されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
ext_stats	ユーザー定義の統計情報。
stattypown	統計タイプのスキーマ。
stattypname	統計タイプの名前。
distcnt	個別値の数。
density	列密度。
nullcnt	NULL の数。
srec	列最大、列最小およびヒストグラム値の内部表記を保持する構造。
avgclen	列の平均長（バイト単位）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか権限が不十分、または要求したオブジェクトの統計情報が格納されていません。

GET_INDEX_STATS プロシージャ

このプロシージャは、索引に関連するすべての情報を取得します。ユーザー定義の統計情報を処理するこのバージョンでは、戻される統計タイプは、ユーザー定義の統計情報に加え、格納されるタイプでもあります。

構文

標準統計情報には、次の構文を使用します。

```
DBMS_STATS.GET_INDEX_STATS (
   ownname      VARCHAR2,
   indname      VARCHAR2,
   partname     VARCHAR2 DEFAULT NULL,
   stattab      VARCHAR2 DEFAULT NULL,
   statid       VARCHAR2 DEFAULT NULL,
   numrows  OUT NUMBER,
   numlblks OUT NUMBER,
   numdist  OUT NUMBER,
   avglblk  OUT NUMBER,
   avgdblk  OUT NUMBER,
   clstfct  OUT NUMBER,
   indlevel OUT NUMBER,
   statown      VARCHAR2 DEFAULT NULL);

DBMS_STATS.GET_INDEX_STATS (
   ownname      VARCHAR2,
   indname      VARCHAR2,
   partname     VARCHAR2 DEFAULT NULL,
   stattab      VARCHAR2 DEFAULT NULL,
   statid       VARCHAR2 DEFAULT NULL,
   numrows  OUT NUMBER,
   numlblks OUT NUMBER,
   numdist  OUT NUMBER,
   avglblk  OUT NUMBER,
   avgdblk  OUT NUMBER,
   clstfct  OUT NUMBER,
   indlevel OUT NUMBER,
   statown      VARCHAR2 DEFAULT NULL,
   guessq   OUT NUMBER);
```

ユーザー定義統計情報には、次の構文を使用します。

```
DBMS_STATS.GET_INDEX_STATS (
   ownname          VARCHAR2,
   indname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   ext_stats   OUT  RAW,
   stattypown  OUT  VARCHAR2 DEFAULT NULL,
   stattypname OUT  VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
```

パラメータ

表 70-11 GET_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	統計情報を取得する索引パーティションの名前。索引がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな索引レベルで取り出されます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接取り出されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
ext_stats	ユーザー定義の統計情報。
stattypown	統計タイプのスキーマ。
stattypname	統計タイプの名前。
numrows	索引（パーティション）内の行数。
numlblks	索引（パーティション）内のリーフ・ブロックの数。
numdist	索引（パーティション）内の個別キーの数。
avglblk	この索引（パーティション）について各個別キーが出現するリーフ・ブロックの平均整数値。
avgdblk	この索引（パーティション）について個別キーが指す表内のデータ・ブロックの平均整数値。

表 70-11 GET_INDEX_STATS プロシージャのパラメータ（続き）

パラメータ	説明
clstfct	索引（パーティション）のクラスタ化要因。
indlevel	索引（パーティション）の高さ。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
guessq	索引の推測品質（パーティション）。

例外

ORA-20000: オブジェクトが存在しないか権限が不十分、または要求したオブジェクトの統計情報が格納されていません。

GET_SYSTEM_STATS プロシージャ

このプロシージャは、stattab（stattab が NULL の場合はディクショナリ）からシステムの統計情報を取得します。

構文

```
DBMS_STATS.GET_SYSTEM_STATS (
   status    OUT  VARCHAR2,
   dstart    OUT  DATE,
   dstop     OUT  DATE,
   pname          VARCHAR2,
   pvalue    OUT  NUMBER,
   stattab   IN   VARCHAR2 DEFAULT NULL,
   statid    IN   VARCHAR2 DEFAULT NULL,
   statown   IN   VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-12 GET_SYSTEM_STATS プロシージャのパラメータ

パラメータ	説明
status（OUT）	出力は次のいずれかです。 COMPLETED: AUTOGATHERING: MANUALGATHERING: BADSTATS:
dstart（OUT）	統計情報の収集が開始された日付。 status = MANUALGATHERING の場合、開始日が戻されます。
dstop（OUT）	統計情報の収集が停止された日付。 status = COMPLETE の場合、終了日が戻されます。 status = AUTOGATHERING の場合、将来の終了日が戻されます。 status = BADSTATS の場合、終了期限の日付が戻されます。
pname	取得するパラメータ名。次の値の 1 つが入ります。 ■ sreadtim －シングル・ブロックの読込み（ランダム読込み）平均時間（ミリ秒単位） ■ mreadtim － mbrc ブロックを 1 回に読み込む（順次読取り）平均時間（ミリ秒単位） ■ cpuspeed － 1 秒当たりの平均 CPU サイクル数（100 万単位） ■ mbrc －順次読取りのマルチブロック読込み平均数（ブロック単位） ■ maxthr －最大 I/O システム・スループット（1 秒当たりのバイト数単位） ■ slavethr －平均スレーブ I/O スループット（1 秒当たりのバイト数単位）
pvalue	取得するパラメータ値。
stattab	統計情報を取得するユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから取得されます。
statid	stattab に保存された統計情報に関連付けられた、オプションの識別子。
statown	stattab を含んだスキーマ（ユーザーのスキーマと異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20002: ユーザー統計表が不正です。アップグレードする必要があります。

ORA-20003: システムの統計情報を収集できません。

ORA-20004: パラメータが存在しません。

GET_TABLE_STATS プロシージャ

このプロシージャは、表に関連するすべての情報を取得します。

構文

```
DBMS_STATS.GET_TABLE_STATS (
   ownname     VARCHAR2,
   tabname     VARCHAR2,
   partname    VARCHAR2 DEFAULT NULL,
   stattab     VARCHAR2 DEFAULT NULL,
   statid      VARCHAR2 DEFAULT NULL,
   numrows OUT NUMBER,
   numblks OUT NUMBER,
   avgrlen OUT NUMBER,
   statown     VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-13 GET_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
partname	統計情報を取得する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで取り出されます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接取り出されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。

表 70-13 GET_TABLE_STATS プロシージャのパラメータ（続き）

パラメータ	説明
numrows	表（パーティション）内の行数。
numblks	表（パーティション）が占有するブロックの数。
avgrlen	表（パーティション）の行の平均長。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか権限が不十分、または要求されたオブジェクトの統計情報が格納されていません。

DELETE_COLUMN_STATS プロシージャ

このプロシージャは、列に関連する統計情報を削除します。

構文

```
DBMS_STATS.DELETE_COLUMN_STATS (
   ownname       VARCHAR2,
   tabname       VARCHAR2,
   colname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   cascade_parts BOOLEAN  DEFAULT TRUE,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-14 DELETE_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	統計情報を削除する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、グローバルな列統計情報が削除されます。
stattab	統計情報を削除する場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接削除されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）(stattab が NULL でない場合のみ該当します)。
cascade_parts	表がパーティション化されていて、partname が NULL の場合、このパラメータを TRUE に設定すると、基礎となるすべてのパーティションについても列の統計情報が削除されます。
statown	stattab を含んだスキーマ（ownname と異なる場合)。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

DELETE_INDEX_STATS プロシージャ

このプロシージャは、索引に関連する統計情報を削除します。

構文

```
DBMS_STATS.DELETE_INDEX_STATS (
   ownname       VARCHAR2,
   indname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   cascade_parts BOOLEAN  DEFAULT TRUE,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-15 DELETE_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	統計情報を削除する索引パーティションの名前。索引がパーティション化されていて、partname が NULL の場合、索引統計情報はグローバル・レベルで削除されます。
stattab	統計情報を削除する場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接削除されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
cascade_parts	索引がパーティション化されていて、partname が NULL の場合、このパラメータを TRUE に設定すると、基礎となるすべてのパーティションについても索引の統計情報が削除されます。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

DELETE_SYSTEM_STATS プロシージャ

このプロシージャは、システムの統計情報を削除します。

構文

```
DBMS_STATS.DELETE_INDEX_STATS (
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-16 DELETE_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報が保存されるユーザー統計表の識別子。
statid	stattab に保存された統計情報に関連付けられた、オプションの識別子。
statown	stattab を含んだスキーマ（ユーザーのスキーマと異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20002: ユーザー統計表が不正です。アップグレードする必要があります。

DELETE_TABLE_STATS プロシージャ

このプロシージャは、表に関連する統計情報を削除します。

構文

```
DBMS_STATS.DELETE_TABLE_STATS (
   ownname         VARCHAR2,
   tabname         VARCHAR2,
   colname         VARCHAR2,
   partname        VARCHAR2  DEFAULT NULL,
   stattab         VARCHAR2  DEFAULT NULL,
   statid          VARCHAR2  DEFAULT NULL,
   cascade_parts   BOOLEAN   DEFAULT TRUE,
   cascade_columns BOOLEAN   DEFAULT TRUE,
   cascade_indexes BOOLEAN   DEFAULT TRUE,
   statown         VARCHAR2 DEFAULT NULL,
   no_invalidate   BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-17 DELETE_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	統計情報を取得する表パーティションの名前。表がパーティション化されていて、partname が NULL の場合、統計情報はグローバルな表レベルで取り出されます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接取り出されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。

表 70-17 DELETE_TABLE_STATS プロシージャのパラメータ（続き）

パラメータ	説明
cascade_parts	表がパーティション化されていて、partname が NULL の場合、このパラメータを TRUE に設定すると、基礎となるすべてのパーティションについても表の統計情報が削除されます。
cascade_columns	基礎となるすべての列について、DELETE_COLUMN_STATS をコールする必要があることを示します（cascade_parts パラメータを渡します）。
cascade_indexes	基礎となるすべての索引について、DELETE_INDEX_STATS をコールする必要があることを示します（cascade_parts パラメータを渡します）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

DELETE_SCHEMA_STATS プロシージャ

このプロシージャは、スキーマ全体の統計情報を削除します。

構文

```
DBMS_STATS.DELETE_SCHEMA_STATS (
   ownname       VARCHAR2,
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-18 DELETE_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
stattab	統計情報を削除する場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接削除されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

DELETE_DATABASE_STATS プロシージャ

このプロシージャは、データベース全体の統計情報を削除します。

構文

```
DBMS_STATS.DELETE_DATABASE_STATS (
   stattab       VARCHAR2 DEFAULT NULL,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-19 DELETE_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報を削除する場所を示すユーザー統計表の識別子。stattab が NULL の場合、統計情報はディクショナリから直接削除されます。
statid	stattab 内の統計情報を関連付ける識別子（オプション）（stattab が NULL でない場合のみ該当します）。
statown	stattab を含んだスキーマ。stattab が NULL ではなく、statown が NULL の場合は、データベース内のすべてのスキーマに、同じ名前の stattab を持つユーザー統計表が含まれているとみなされます。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

CREATE_STAT_TABLE プロシージャ

このプロシージャは、統計情報を保持できる ownname のスキーマにある stattab の名前で表を作成します。この表は、このパッケージのプロシージャを介して単独にアクセスされるため、この表を構成する列と型は互いに関係がありません。

構文

```
DBMS_STATS.CREATE_STAT_TABLE (
   ownname  VARCHAR2,
   stattab  VARCHAR2,
   tblspace VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-20 CREATE_STAT_TABLE プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
stattab	作成する表の名前。ユーザーがディクショナリ統計情報を直接変更しない場合、この値は、stattab パラメータとして他のプロシージャに渡されます。
tblspace	統計表を作成する表領域。このパラメータを指定しないと、統計表はユーザーのデフォルトの表領域に作成されます。

例外

ORA-20000: 表がすでに存在するか、権限が不十分です。

ORA-20001: 表領域が存在しません。

DROP_STAT_TABLE プロシージャ

このプロシージャは、ユーザー統計表を削除します。

構文

```
DBMS_STATS.DROP_STAT_TABLE (
   ownname VARCHAR2,
   stattab VARCHAR2);
```

パラメータ

表 70-21 DROP_STAT_TABLE プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
stattab	ユーザー統計表の識別子。

例外

ORA-20000: 表が存在しないか、または権限が不十分です。

EXPORT_COLUMN_STATS プロシージャ

このプロシージャは、特定の列に関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_COLUMN_STATS (
   ownname  VARCHAR2,
   tabname  VARCHAR2,
   colname  VARCHAR2,
   partname VARCHAR2 DEFAULT NULL,
   stattab  VARCHAR2,
   statid   VARCHAR2 DEFAULT NULL,
   statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-22 EXPORT_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	表パーティション名。表がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された列の統計情報がエクスポートされます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

EXPORT_INDEX_STATS プロシージャ

このプロシージャは、特定の索引に関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_INDEX_STATS (
   ownname  VARCHAR2,
   indname  VARCHAR2,
   partname VARCHAR2 DEFAULT NULL,
   stattab  VARCHAR2,
   statid   VARCHAR2 DEFAULT NULL,
   statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-23 EXPORT_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	索引パーティション名。索引がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された索引統計情報がエクスポートされます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

EXPORT_SYSTEM_STATS プロシージャ

このプロシージャは、システムの統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_SYSTEM_STATS (
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-24 EXPORT_SYSTEM_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報が格納される場所を示すユーザー統計表の識別子。
statid	stattab に格納された統計情報に関連付けられた、オプションの識別子。
statown	stattab を含んだスキーマ（ユーザーのスキーマと異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20002: ユーザー統計表が不正です。アップグレードする必要があります。

ORA-20003: システムの統計情報をエクスポートできません。

EXPORT_TABLE_STATS プロシージャ

このプロシージャは、特定の表に関する統計情報を取り出し、ユーザー統計表に格納します。cascade を使用すると、指定した表に関連付けられている索引および列統計情報もすべてエクスポートされます。

構文

```
DBMS_STATS.EXPORT_TABLE_STATS (
   ownname  VARCHAR2,
   tabname  VARCHAR2,
   partname VARCHAR2 DEFAULT NULL,
   stattab  VARCHAR2,
   statid   VARCHAR2 DEFAULT NULL,
   cascade  BOOLEAN  DEFAULT TRUE,
   statown  VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-25 EXPORT_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	表の名前。
partname	表パーティション名。表がパーティション化されていて、partname が NULL の場合、グローバルでパーティション表統計情報がエクスポートされます。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
cascade	TRUE の場合は、この表の列と索引の統計情報もまたエクスポートされます。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

EXPORT_SCHEMA_STATS プロシージャ

このプロシージャは、ownname で識別されるスキーマ内のすべてのオブジェクトに関する統計情報を取り出し、stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_SCHEMA_STATS (
   ownname VARCHAR2,
   stattab VARCHAR2,
   statid  VARCHAR2 DEFAULT NULL,
   statown VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-26 EXPORT_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

EXPORT_DATABASE_STATS プロシージャ

このプロシージャは、データベース内のすべてのオブジェクトに関する統計情報を取り出し、statown.stattab で識別されるユーザー統計表に格納します。

構文

```
DBMS_STATS.EXPORT_DATABASE_STATS (
   stattab VARCHAR2,
   statid  VARCHAR2 DEFAULT NULL,
   statown VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-27 EXPORT_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報の格納場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ。statown が NULL の場合、データベース内のすべてのスキーマには stattab の名前を持つユーザー統計表が含まれているとみなされます。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

IMPORT_COLUMN_STATS プロシージャ

このプロシージャは、stattab で識別されるユーザー統計表から特定の列に関する統計情報を取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_COLUMN_STATS (
   ownname       VARCHAR2,
   tabname       VARCHAR2,
   colname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-28 IMPORT_COLUMN_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	列が所属している表の名前。
colname	列の名前。
partname	表パーティション名。表がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された列統計情報がインポートされます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: ユーザー統計表の値が無効または矛盾しています。

IMPORT_INDEX_STATS プロシージャ

このプロシージャは、stattab で識別されるユーザー統計表から特定の索引に関する統計情報を取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_INDEX_STATS (
   ownname       VARCHAR2,
   indname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-29 IMPORT_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
indname	索引名。
partname	索引パーティション名。索引がパーティション化されていて、partname が NULL の場合、グローバルでパーティション化された索引統計情報がインポートされます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: ユーザー統計表の値が無効または矛盾しています。

IMPORT_SYSTEM_STATS プロシージャ

このプロシージャは、stattab で識別されるユーザー統計表からシステムの統計情報を取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_SYSTEM_STATS (
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-30 IMPORT_SYSTEM_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報を取り出すユーザー統計表の識別子。
statid	stattab から取り出された統計情報に関連付けられた、オプションの識別子。
statown	stattab を含んだスキーマ（ユーザーのスキーマと異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: ユーザー統計表の値が無効または矛盾しています。

ORA-20002: ユーザー統計表が不正です。アップグレードする必要があります。

ORA-20003: システムの統計情報をインポートできません。

IMPORT_TABLE_STATS プロシージャ

このプロシージャは、stattab で識別されるユーザー統計表から特定の表に関する統計情報を取り出し、ディクショナリに格納します。cascade を使用すると、指定した表に関連付けられている索引および列統計情報もすべてインポートされます。

構文

```
DBMS_STATS.IMPORT_TABLE_STATS (
   ownname       VARCHAR2,
   tabname       VARCHAR2,
   partname      VARCHAR2 DEFAULT NULL,
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   cascade       BOOLEAN  DEFAULT TRUE,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-31 IMPORT_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
tabname	表の名前。
partname	表パーティション名。表がパーティション化されていて、partname が NULL の場合、グローバルでパーティション表統計情報がインポートされます。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
cascade	TRUE の場合は、この表の列と索引の統計情報もまたインポートされます。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: ユーザー統計表の値が無効または矛盾しています。

IMPORT_SCHEMA_STATS プロシージャ

このプロシージャは、ownname で識別されるスキーマ内のすべてのオブジェクトに関する統計情報をユーザー統計表から取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_SCHEMA_STATS (
   ownname       VARCHAR2,
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-32 IMPORT_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
ownname	スキーマの名前。
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: ユーザー統計表の値が無効または矛盾しています。

IMPORT_DATABASE_STATS プロシージャ

このプロシージャは、データベース内のすべてのオブジェクトに関する統計情報をユーザー統計表から取り出し、ディクショナリに格納します。

構文

```
DBMS_STATS.IMPORT_DATABASE_STATS (
   stattab       VARCHAR2,
   statid        VARCHAR2 DEFAULT NULL,
   statown       VARCHAR2 DEFAULT NULL,
   no_invalidate BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-33 IMPORT_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
stattab	統計情報を取り出す場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ。statown が NULL の場合、データベース内のすべてのスキーマには stattab の名前を持つユーザー統計表が含まれているとみなされます。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: ユーザー統計表の値が無効または矛盾しています。

GATHER_INDEX_STATS プロシージャ

このプロシージャは、索引の統計情報を収集します。このプロシージャは、可能なかぎり多くの作業をパラレル化します。個々のパラメータで説明するように、いくつかの制限があります。この操作では、クラスタ索引、ドメイン索引、ビットマップ・ジョイン・インデックスなど、特定タイプの索引でのパラレル化は行いません。granularity 引数および no_invalidate 引数は、これらのタイプの索引とは関係がありません。

構文

```
DBMS_STATS.GATHER_INDEX_STATS (
   ownname          VARCHAR2,
   indname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   estimate_percent NUMBER   DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
   degree           NUMBER   DEFAULT NULL,
   granularity      VARCHAR2 DEFAULT 'DEFAULT',
   no_invalidate    BOOLEAN  DEFAULT FALSE);
```

パラメータ

表 70-34 GATHER_INDEX_STATS プロシージャのパラメータ

パラメータ	説明
ownname	分析する索引のスキーマ。
indname	索引名。
partname	パーティション名。
estimate_percent	推定する行のパーセント（NULL は計算を意味します）。有効な範囲は、0.000001 ～ 100 です。適切な統計を行うために最適なサンプル・サイズを Oracle で決定するには、定数 DBMS_STATS.AUTO_SAMPLE_SIZE を使用します。
stattab	現行の統計情報を保存する場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。

表 70-34 GATHER_INDEX_STATS プロシージャのパラメータ（続き）

パラメータ	説明
degree	並列度（NULL は、CREATE/ALTER INDEX 文の DEGREE 句で指定した表のデフォルト値を使用することを意味します)。初期化パラメータに基づいたデフォルト値には、定数 DBMS_STATS.DEFAULT_DEGREE を使用します。
granularity	収集する統計情報の細分化（索引がパーティション化されている場合のみ該当します)。 DEFAULT ーグローバル・レベルおよびパーティション・レベルの統計情報を収集します。 SUBPARTITION ーサブパーティション・レベルの統計情報を収集します。 PARTITION ーパーティション・レベルの統計情報を収集します。 GLOBAL ーグローバル・レベルの統計情報を収集します。 ALL ーすべての統計情報（サブパーティション、パーティションおよびグローバル）を収集します。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。

例外

ORA-20000: 索引が存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

GATHER_TABLE_STATS プロシージャ

このプロシージャは、表と列（および索引）の統計情報を収集します。このプロシージャは、可能なかぎり多くの作業をパラレル化しますが、個々のパラメータで説明するように、いくつかの制限があります。分析する表に対する SELECT 権限がユーザーにない場合、この操作はパラレル化しません。

構文

```
DBMS_STATS.GATHER_TABLE_STATS (
   ownname          VARCHAR2,
   tabname          VARCHAR2,
   partname         VARCHAR2 DEFAULT NULL,
   estimate_percent NUMBER   DEFAULT NULL,
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
   degree           NUMBER   DEFAULT NULL,
   granularity      VARCHAR2 DEFAULT 'DEFAULT',
   cascade          BOOLEAN  DEFAULT FALSE,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT FALSE);
```

パラメータ

表 70-35 GATHER_TABLE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	分析する表のスキーマ。
tabname	表の名前。
partname	パーティション名。
estimate_percent	推定する行のパーセント（NULL は計算を意味します）。有効な範囲は、0.000001 ～ 100 です。適切な統計を行うために最適なサンプル・サイズを Oracle で決定するには、定数 DBMS_STATS.AUTO_SAMPLE_SIZE を使用します。
block_sample	ランダム行サンプリングのかわりにランダム・ブロック・サンプリングを使用するかどうかを示します。ランダム・ブロック・サンプリングがより効率的ですが、データがディスク上にランダムに分散していない場合、サンプル値はある程度相関があります。統計情報の推定を行った場合のみ該当します。

表 70-35 GATHER_TABLE_STATS プロシージャのパラメータ（続き）

パラメータ	説明
method_opt	次を受け入れます。 FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] FOR COLUMNS [size clause] column|attribute [size_clause] [,column|attribute [size_clause]...] size_clause は次のように定義されます。 size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} integer －ヒストグラム・バケット数。1 ～ 254 の範囲です。 REPEAT －すでにヒストグラムがある列に関してのみ、ヒストグラムを収集します。 AUTO －列のデータ配分とワークロードに基づいて、ヒストグラムを収集する列が判断されます。 SKEWONLY －列のデータ配分に基づいて、ヒストグラムを収集する列が判断されます。
degree	並列度。NULL の場合、CREATE TABLE 文または ALTER TABLE 文の DEGREE 句で指定された表のデフォルト値が使用されます。初期化パラメータに基づいてデフォルト値を指定するには、定数 DBMS_STATS.DEFAULT_DEGREE を使用します。
granularity	収集する統計情報の細分化（表がパーティション化されている場合のみ該当します）。 DEFAULT: グローバル・レベルでパーティション・レベルの統計情報を収集します。 SUBPARTITION: サブパーティション・レベルの統計情報を収集します。 PARTITION: パーティション・レベルの統計情報を収集します。 GLOBAL: グローバルな統計情報を収集します。 ALL: すべての統計情報（サブパーティション、パーティションおよびグローバル）を収集します。
cascade	表の索引について統計情報を収集します。索引統計情報の収集はパラレル化されません。このオプションを使用することは、表の各索引で GATHER_INDEX_STATS プロシージャを実行するのと同じです。

表 70-35 GATHER_TABLE_STATS プロシージャのパラメータ（続き）

パラメータ	説明
stattab	現行の統計情報を保存する場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。cascade 引数が指定されている場合、このパラメータは 70-48 ページの「GATHER_INDEX_STATS プロシージャ」に説明されているように、特定の索引タイプと関連がありません。

例外

ORA-20000: 表が存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

GATHER_SCHEMA_STATS プロシージャ

このプロシージャは、スキーマ内のすべてのオブジェクトに関する統計情報を収集します。

構文

```
DBMS_STATS.GATHER_SCHEMA_STATS (
   ownname          VARCHAR2,
   estimate_percent NUMBER   DEFAULT NULL,
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
   degree           NUMBER   DEFAULT NULL,
   granularity      VARCHAR2 DEFAULT 'DEFAULT',
   cascade          BOOLEAN  DEFAULT FALSE,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER',
   objlist      OUT ObjectTab,
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN DEFAULT FALSE,
   gather_temp      BOOLEAN DEFAULT FALSE);

DBMS_STATS.GATHER_SCHEMA_STATS (
   ownname          VARCHAR2,
   estimate_percent NUMBER   DEFAULT NULL,
   block_sample     BOOLEAN  DEFAULT FALSE,
```

```
   method_opt       VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
   degree           NUMBER   DEFAULT NULL,
   granularity      VARCHAR2 DEFAULT 'DEFAULT',
   cascade          BOOLEAN  DEFAULT FALSE,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER',
   statown          VARCHAR2 DEFAULT NULL,
   no_invalidate    BOOLEAN  DEFAULT FALSE,
   gather_temp      BOOLEAN  DEFAULT FALSE);
```

パラメータ

表 70-36 GATHER_SCHEMA_STATS プロシージャのパラメータ

パラメータ	説明
ownname	分析するスキーマ（NULL は現行のスキーマを意味します）。
estimate_percent	推定する行のパーセント（NULL は計算を意味します）。有効な範囲は、0.000001 ～ 100 です。適切な統計を行うために最適なサンプル・サイズを Oracle で決定するには、定数 DBMS_STATS.AUTO_SAMPLE_SIZE を使用します。
block_sample	ランダム行サンプリングのかわりにランダム・ブロック・サンプリングを使用するかどうかを示します。ランダム・ブロック・サンプリングがより効率的ですが、データがディスク上にランダムに分散していない場合、サンプル値はある程度相関があります。統計情報の推定を行った場合のみ該当します。
method_opt	次を受け入れます。 FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] FOR COLUMNS [size clause] column|attribute [size_clause] [,column|attribute [size_clause]...] size_clause は次のように定義されます。 size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} integer －ヒストグラム・バケット数。1 ～ 254 の範囲です。 REPEAT －すでにヒストグラムがある列に対してのみ、ヒストグラムを収集します。 AUTO －列のデータ配分とワークロードに基づいて、ヒストグラムを収集する列が判断されます。 SKEWONLY －列のデータ配分に基づいて、ヒストグラムを収集する列が判断されます。

表 70-36 GATHER_SCHEMA_STATS プロシージャのパラメータ（続き）

パラメータ	説明
degree	並列度。NULL の場合、CREATE TABLE 文または ALTER TABLE 文の DEGREE 句で指定された表のデフォルト値が使用されます。初期化パラメータに基づいてデフォルト値を指定するには、定数 DBMS_STATS.DEFAULT_DEGREE を使用します。
granularity	収集する統計情報の細分化（表がパーティション化されている場合のみ該当します）。 DEFAULT: グローバル・レベルでパーティション・レベルの統計情報を収集します。 SUBPARTITION: サブパーティション・レベルの統計情報を収集します。 PARTITION: パーティション・レベルの統計情報を収集します。 GLOBAL: グローバルな統計情報を収集します。 ALL: すべての統計情報（サブパーティション、パーティションおよびグローバル）を収集します。
cascade	索引についても統計情報を収集します。 索引統計情報の収集はパラレル化されません。このオプションを使用することは、表と列の統計情報の収集に加えて、スキーマ内の各索引で gather_index_stats プロシージャを実行するのと同じです。
stattab	現行の統計情報を保存する場所を示すユーザー統計表の識別子。
statid	stattab 内の統計情報を関連付ける識別子（オプション）。

表 70-36 GATHER_SCHEMA_STATS プロシージャのパラメータ（続き）

パラメータ	説明
options	統計情報を収集するオブジェクトの詳細は、次のように指定します。 GATHER: スキーマ内のすべてのオブジェクトに関する統計情報を収集します。 GATHER AUTO: 必要な統計情報をすべて自動的に収集します。Oracle は、新しい統計情報を必要とするオブジェクトを暗黙的に判別し、その統計情報を収集する方法を判別します。GATHER AUTO が指定された場合、有効な追加パラメータは ownname、stattab、statid、objlist および statown のみです。その他のパラメータ設定はすべて無視されます。処理されたオブジェクトのリストを戻します。 GATHER STALE: *_tab_modifications ビューを調べて判別した失効オブジェクトについて、統計情報を収集します。また、失効と判別されたオブジェクトのリストも戻します。 GATHER EMPTY: 現在統計情報がないオブジェクトについて統計情報を収集し、統計情報なしと判別されたオブジェクトのリストも戻します。 LIST AUTO: GATHER AUTO を使用して処理されるオブジェクトのリストを戻します。 LIST STALE: *_tab_modifications ビューを調べて判別した失効オブジェクトのリストを戻します。 LIST EMPTY: 現在統計情報がないオブジェクトのリストを戻します。
objlist	失効または空と判別されたオブジェクトのリスト。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。cascade 引数が指定されている場合、このパラメータは 70-48 ページの「GATHER_INDEX_STATS プロシージャ」に説明されているように、特定の索引タイプと関連がありません。
gather_temp	グローバル一時表の統計情報を収集します。この一時表は、on commit preserve rows 句を使用して作成されている必要があります。統計情報は、このプロシージャを実行しているセッションのデータに基づいて収集されますが、すべてのセッションで共有されます。

例外

ORA-20000: スキーマが存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

GATHER_DATABASE_STATS プロシージャ

このプロシージャは、データベース内のすべてのオブジェクトに関する統計情報を収集します。

構文

```
DBMS_STATS.GATHER_DATABASE_STATS (
   estimate_percent NUMBER   DEFAULT NULL,
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
   degree           NUMBER   DEFAULT NULL,
   granularity      VARCHAR2 DEFAULT 'DEFAULT',
   cascade          BOOLEAN  DEFAULT FALSE,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER',
   objlist   OUT    ObjectTab,
   statown          VARCHAR2 DEFAULT NULL,
   gather_sys       BOOLEAN DEFAULT FALSE,
   no_invalidate    BOOLEAN DEFAULT FALSE,
   gather_temp      BOOLEAN DEFAULT FALSE);

DBMS_STATS.GATHER_DATABASE_STATS (
   estimate_percent NUMBER   DEFAULT NULL,
   block_sample     BOOLEAN  DEFAULT FALSE,
   method_opt       VARCHAR2 DEFAULT 'FOR ALL COLUMNS SIZE 1',
   degree           NUMBER   DEFAULT NULL,
   granularity      VARCHAR2 DEFAULT 'DEFAULT',
   cascade          BOOLEAN  DEFAULT FALSE,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   options          VARCHAR2 DEFAULT 'GATHER',
   statown          VARCHAR2 DEFAULT NULL,
   gather_sys       BOOLEAN DEFAULT FALSE,
   no_invalidate    BOOLEAN DEFAULT FALSE,
   gather_temp      BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-37 GATHER_DATABASE_STATS プロシージャのパラメータ

パラメータ	説明
estimate_percent	推定する行のパーセント（NULL は計算を意味します）。有効な範囲は、0.000001 ～ 100 です。適切な統計を行うために最適なサンプル・サイズを Oracle で決定するには、定数 DBMS_STATS.AUTO_SAMPLE_SIZE を使用します。
block_sample	ランダム行サンプリングのかわりにランダム・ブロック・サンプリングを使用するかどうかを示します。ランダム・ブロック・サンプリングがより効率的ですが、データがディスク上にランダムに分散していない場合、サンプル値はある程度相関があります。統計情報の推定を行った場合のみ該当します。
method_opt	次を受け入れます。 FOR ALL [INDEXED | HIDDEN] COLUMNS [size_clause] FOR COLUMNS [size clause] column|attribute [size_clause] [,column|attribute [size_clause]...] size_clause は次のように定義されます。 size_clause := SIZE {integer | REPEAT | AUTO | SKEWONLY} integer －ヒストグラム・バケット数。1 ～ 254 の範囲です。 REPEAT －すでにヒストグラムがある列に対してのみ、ヒストグラムを収集します。 AUTO －列のデータ配分とワークロードに基づいて、ヒストグラムを収集する列が判断されます。 SKEWONLY －列のデータ配分に基づいて、ヒストグラムを収集する列が判断されます。
degree	並列度。NULL の場合、CREATE TABLE 文または ALTER TABLE 文の DEGREE 句で指定された表のデフォルト値が使用されます。初期化パラメータに基づいてデフォルト値を指定するには、定数 DBMS_STATS.DEFAULT_DEGREE を使用します。

表 70-37 GATHER_DATABASE_STATS プロシージャのパラメータ（続き）

パラメータ	説明
granularity	収集する統計情報の細分化（表がパーティション化されている場合のみ該当します)。 DEFAULT: グローバル・レベルでパーティション・レベルの統計情報を収集します。 SUBPARTITION: サブパーティション・レベルの統計情報を収集します。 PARTITION: パーティション・レベルの統計情報を収集します。 GLOBAL: グローバルな統計情報を収集します。 ALL: すべての統計情報（サブパーティション、パーティションおよびグローバル）を収集します。
cascade	索引についても統計情報を収集します。索引統計情報の収集はパラレル化されません。このオプションを使用することは、表と列の統計情報の収集に加えて、データベース内の各索引で gather_index_stats プロシージャを実行するのと同じです。
stattab	現行の統計情報を保存する場所を示すユーザー統計表の識別子。 統計表は、分析するオブジェクトと同じスキーマに常駐するとみなされるので、各スキーマにこのオプションを使用するための表が 1 つ必要です。
statid	stattab 内の統計情報を関連付ける識別子（オプション)。

表 70-37 GATHER_DATABASE_STATS プロシージャのパラメータ（続き）

パラメータ	説明
options	統計情報を収集するオブジェクトの詳細は、次のように指定します。 GATHER: スキーマ内のすべてのオブジェクトに関する統計情報を収集します。 GATHER AUTO: 必要な統計情報をすべて自動的に収集します。Oracle は、新しい統計情報を必要とするオブジェクトを暗黙的に判別し、その統計情報を収集する方法を判別します。GATHER AUTO が指定された場合、有効な追加パラメータは stattab、statid、objlist および statown のみです。その他のパラメータ設定はすべて無視されます。処理されたオブジェクトのリストを戻します。 GATHER STALE: *_tab_modifications ビューを調べて判別した失効オブジェクトについて、統計情報を収集します。また、失効と判別されたオブジェクトのリストも戻します。 GATHER EMPTY: 現在統計情報がないオブジェクトの統計情報を収集します。統計情報がないオブジェクトのリストを戻します。 LIST AUTO: GATHER AUTO を使用して処理されるオブジェクトのリストを戻します。 LIST STALE: *_tab_modifications ビューを調べて判別した失効オブジェクトのリストを戻します。 LIST EMPTY: 現在統計情報がないオブジェクトのリストを戻します。
objlist	失効または空と判別されたオブジェクトのリスト。
statown	stattab を含んだスキーマ（ownname と異なる場合）。
gather_sys	SYS ユーザーが所有するオブジェクトの統計情報を収集します。
no_invalidate	このパラメータを TRUE に設定すると、依存カーソルは無効化されません。cascade オプションが指定されている場合、このパラメータは 70-48 ページの「GATHER_INDEX_STATS プロシージャ」に説明されているように、特定の索引タイプと関連がありません。
gather_temp	グローバル一時表の統計情報を収集します。一時表は、on commit preserve rows 句を使用して作成されている必要があります。統計情報は、このプロシージャが実行するセッションのデータに基づいて収集されますが、すべてのセッションで共有されるデータは除外されます。

例外

ORA-20000: 権限が不十分です。

ORA-20001: 入力値が無効です。

GATHER_SYSTEM_STATS プロシージャ

このプロシージャは、システムの統計情報を収集します。

構文

```
DBMS_STATS.GATHER_SYSTEM_STATS (
   gathering_mode   VARCHAR2 DEFAULT 'NOWORKLOAD',
   interval         INTEGER  DEFAULT NULL,
   stattab          VARCHAR2 DEFAULT NULL,
   statid           VARCHAR2 DEFAULT NULL,
   statown          VARCHAR2 DEFAULT NULL);
```

パラメータ

表 70-38 GATHER_SYSTEM_STATS プロシージャのパラメータ

パラメータ	説明
gathering_mode	モードの値は、次のとおりです。 NOWORKLOAD: システム・アクティビティの取得にワークロードは不要です。システム統計情報は、内部のデフォルトを使用して生成されます。このモードは、適切なワークロードを発行できない場合（開発過程など）に使用できます。実際のシステム・アクティビティに基づいたシステム統計情報の値のためには、INTERVAL モードまたは START|STOP モードを使用します。 INTERVAL: 指定された間隔の間にシステムのアクティビティを獲得します。このモードは、interval パラメータとの組合せで動作します。間隔は分単位で指定してください。指定した後、ディクショナリまたは stattab でシステムの統計情報が作成または更新されます。予定よりも早く収集を停止する場合は、gather_system_stats(gathering_mode=>'STOP') を使用できます。 START | STOP: 指定された開始時間から停止時間の間にシステムのアクティビティを獲得し、経過期間の統計情報を使用してディクショナリまたは stattab をリフレッシュします。Interval 値は無視されます。

表 70-38 GATHER_SYSTEM_STATS プロシージャのパラメータ（続き）

パラメータ	説明
interval	統計情報を収集する時間（分単位）。このパラメータが適用されるのは、gathering_mode='INTERVAL' の場合のみです。
stattab	統計情報が保存されるユーザー統計表の識別子。
statid	stattab に保存された統計情報に関連付けられた、オプションの識別子。
statown	stattab を含んだスキーマ（ユーザーのスキーマと異なる場合）。

例外

ORA-20000: オブジェクトが存在しないか、または権限が不十分です。

ORA-20001: 入力値が無効です。

ORA-20002: ユーザー統計表が不正です。アップグレードする必要があります。

ORA-20003: システムの統計情報を収集できません。

ORA-20004: INTERVAL モードでのエラー。システム・パラメータ job_queue_processes は >0 に設定する必要があります。

GENERATE_STATS プロシージャ

このプロシージャは、関連するオブジェクトで以前に収集した統計情報から、オブジェクトの統計情報を生成します。完全に移入されたスキーマについては、より正確な統計情報が必要な場合は、GATHER プロシージャをかわりに使用する必要があります。現在サポートされているオブジェクトは、B ツリー索引とビットマップ索引です。

構文

```
DBMS_STATS.GENERATE_STATS (
   ownname    VARCHAR2,
   objname    VARCHAR2,
   organized NUMBER DEFAULT 7);
```

パラメータ

表 70-39 GENERATE_STATS プロシージャのパラメータ

パラメータ	説明
ownname	オブジェクトのスキーマ。
objname	オブジェクト名。
organized	索引とその基礎となる表の間で関連付けられた順位付けの量。複雑に編成されている索引には、ディスク上の連続行を参照する連続索引キーがその表（同じブロック）に対してあります。複雑に編成されていない索引には、ディスク上の異なる表ブロックを参照する連続キーがあります。 このパラメータは、B ツリー索引に対してのみ使用します。数値は、0 ～ 10 の範囲で使用でき、0（ゼロ）は完全に編成された索引、10 は完全に編成解除された索引を示します。

例外

ORA-20000: サポートされていないオブジェクト・タイプで、オブジェクトは存在しません。

ORA-20001: 無効なオプションまたは無効な統計情報です。

FLUSH_DATABASE_MONITORING_INFO プロシージャ

このプロシージャは、メモリー内のすべての表の監視情報をディクショナリにフラッシュします。

構文

```
DBMS_STATS.FLUSH_DATABASE_MONITORING_INFO;
```

例外

ORA-20000: 権限が不十分です。

ALTER_SCHEMA_TAB_MONITORING プロシージャ

このプロシージャは、スキーマにおけるすべての表の機能を監視する DML を使用可能または使用禁止にします。ただし、スナップショット・ログおよび表では監視がサポートされていないので対象外です。このプロシージャは、ALTER TABLE...MONITORING（または NOMONITORING）を個別に発行するのと同等です。GATHER_DATABASE_STATS または GATHER_SCHEMA_STATS を GATHER AUTO または GATHER STALE オプションとともに使用する場合は、監視機能を使用可能にしてください。

構文

```
DBMS_STATS.ALTER_SCHEMA_TAB_MONITORING (
   ownname    VARCHAR2 DEFAULT NULL,
   monitoring BOOLEAN DEFAULT TRUE);
```

パラメータ

表 70-40 ALTER_SCHEMA_TAB_MONITORING プロシージャのパラメータ

パラメータ	説明
ownname	スキーマ名 （NULL は現行のスキーマを意味します）。
monitoring	TRUE の場合は監視を使用可能にし、FALSE の場合は監視を使用禁止にします。

例外

ORA-20000: 権限が不十分です。

ALTER_DATABASE_TAB_MONITORING プロシージャ

このプロシージャは、スキーマにおけるすべての表の機能を監視する DML を使用可能または使用禁止にします。ただし、スナップショット・ログおよび表では監視がサポートされていないので対象外です。このプロシージャは、ALTER TABLE...MONITORING（または NOMONITORING）を個別に発行するのと同等です。GATHER_DATABASE_STATS または GATHER_SCHEMA_STATS を GATHER AUTO または GATHER STALE オプションとともに使用する場合は、監視機能を使用可能にしてください。

構文

```
DBMS_STATS.ALTER_DATABASE_TAB_MONITORING (
   monitoring BOOLEAN DEFAULT TRUE,
   sysobjs    BOOLEAN DEFAULT FALSE);
```

パラメータ

表 70-41 ALTER_DATABASE_TAB_MONITORING プロシージャのパラメータ

パラメータ	説明
monitoring	TRUE の場合は監視を使用可能にし、FALSE の場合は監視を使用禁止にします。
sysobjs	TRUE の場合、ディクショナリ・オブジェクトの監視を変更します。

例外

ORA-20000: 権限が不十分です。

元の統計情報の保存と新規統計情報の収集 : 例

employees 表から最後に統計情報が収集された後、数多くの変更が加えられたと想定します。コストベースのオプティマイザが最適プランを選択するために、統計情報を再度収集する必要があります。しかし、ユーザーは、現行のプランが許容されると、新規の統計情報によってオプティマイザが誤ったプランを選択することを懸念しています。ユーザーは、次のように指定できます。

```
BEGIN
   DBMS_STATS.CREATE_STAT_TABLE ('hr', 'savestats');
   DBMS_STATS.GATHER_TABLE_STATS ('hr', 'employees', stattab => 'savestats');
END;
```

この操作は、新規の統計情報を employees 表に収集しますが、最初に、元の統計情報をユーザー統計表 hr.savestats に保存します。

ユーザーが、新規統計情報によってオプティマイザが誤ったプランを生成すると考えている場合は、元の統計情報を次のようにリストアできます。

```
BEGIN
   DBMS_STATS.DELETE_TABLE_STATS ('hr', 'employees');
   DBMS_STATS.IMPORT_TABLE_STATS ('hr', 'employees', stattab => 'savestats');
END;
```

日中のシステム統計情報の収集 : 例

昼間 OLTP トランザクションを処理するデータベース・アプリケーションを実行し、夜間にレポートを実行すると想定します。

昼間のシステムの統計情報を収集するには、720 分間情報を収集します。収集した統計情報を MYSTATS 表に格納します。

```
BEGIN
   DBMS_STATS.GATHER_SYSTEM_STATS (
      interval => 720,
      stattab  => 'mystats',
      statid   => 'OLTP');
END;
```

夜間のシステムの統計情報を収集するには、720 分間情報を収集します。収集した統計情報を MYSTATS 表に格納します。

```
BEGIN
   DBMS_STATS.GATHER_SYSTEM_STATS (
      interval => 720,
      stattab  => 'mystats',
      statid   => 'OLAP');
END;
```

収集した統計情報を使用して、ディクショナリを更新します。

```
VARIABLE   jobno   number;
BEGIN
   DBMS_JOB.SUBMIT (:jobno, 'DBMS_STATS.IMPORT_SYSTEM_STATS
   (''mystats'',''OLTP'');'
   sysdate, 'sysdate + 1');
   COMMIT;
END;

BEGIN
   DBMS_JOB.SUBMIT (:jobno, 'DBMS_STATS.IMPORT_SYSTEM_STATS
   (''mystats'',''OLAP'');'
   sysdate + 0.5, 'sysdate + 1');
   COMMIT;
END;
```

71

DBMS_STORAGE_MAP

DBMS_STORAGE_MAP を使用すると、Oracle バックグラウンド・プロセスの FMON と通信して、マッピング・ビューを移入するマッピング操作を起動できます。FMON は、オペレーティング・システムやストレージ・システムのベンダーが供給するマッピング・ライブラリと通信します。

この章では、次の項目について説明します。

マッピングの用語

次に、DBMS_STORAGE_MAP API を理解する上で役に立つ用語と解説を示します。

- マッピング・ライブラリ

  マッピング・ライブラリは、I/O 処理スタック要素のコンポーネントのマッピングに役立ちます。I/O 処理コンポーネントの例には、ファイル、論理ボリュームおよびストレージ・アレイ I/O ターゲットなどがあります。マッピング・ライブラリは、filemap.ora で識別されます。

- マッピング・ファイル

  マッピング・ファイルは、ファイルを記述するマッピング構造です。ファイル・サイズ、ファイルを構成するエクステント数、ファイル・タイプなどの一連の属性を提供します。

- マッピング要素とマッピング副要素

  マッピング要素は、I/O スタック内の記憶域コンポーネントを示す抽象的なマッピング構造です。要素の例には、その構造がマッピングの基本要素である、ミラー化、ストライプ化、パーティション、RAID5、連結要素およびディスクなどがあります。マッピング副要素は、I/O マッピング・スタック内の 1 つの要素と次の要素との間のリンクを示します。

- マッピング・ファイル・エクステント

  マッピング・ファイル・エクステントは、1 つの要素に常駐する連続的なブロックの集合を示します。これには、デバイス・オフセット、エクステント・サイズ、ファイル・オフセット、タイプ（データまたはパリティ）およびエクステントが常駐する要素の名前が含まれます。RAW デバイスまたは RAW ボリュームの場合、ファイルは 1 つのファイル・エクステント・コンポーネントのみで構成されます。マッピング・ファイル・エクステントは、Oracle エクステントとは異なります。

  関連項目：

  - 詳細は、『Oracle9*i* データベース管理者ガイド』を参照してください。
  - V$MAP_FILE、V$MAP_ELEMENT、V$MAP_SUBELEMENT、V$MAP_FILE_EXTENT などの V$MAP ビューについては、『Oracle9*i* データベース・リファレンス』を参照してください。

DBMS_STORAGE_MAP サブプログラムの要約

表 71-1　DBMS_STORAGE_MAP パッケージのサブプログラム

サブプログラム	説明
「MAP_ELEMENT ファンクション」71-5 ページ	elemname で識別される要素のマッピング情報を作成します。
「MAP_FILE ファンクション」71-6 ページ	filename で識別されるファイルのマッピング情報を作成します。
「MAP_OBJECT ファンクション」71-6 ページ	オブジェクト名、所有者およびタイプで識別される Oracle オブジェクトのマッピング情報を作成します。
「MAP_ALL ファンクション」71-6 ページ	すべてのタイプの Oracle ファイルに関する完全なマッピング情報を作成します（アーカイブ・ログを除く）。これには、すべての有向非巡回グラフ（DAG）要素が含まれます。
「DROP_ELEMENT ファンクション」71-7 ページ	elemname によって定義された要素のマッピング情報を削除します。
「DROP_FILE ファンクション」71-7 ページ	filename によって定義されたマッピング情報を削除します。
「DROP_ALL ファンクション」71-8 ページ	インスタンスの共有メモリー内にあるマッピング情報をすべて削除します。
「SAVE ファンクション」71-8 ページ	データ・ディクショナリへの完全なマッピングの再生成に必要な情報を保存します。
「RESTORE ファンクション」71-8 ページ	データ・ディクショナリからインスタンスの共有メモリーに、完全なマッピング情報をロードします。
「LOCK_MAP プロシージャ」71-8 ページ	インスタンスの共有メモリー内にあるマッピング情報をロックします。
「UNLOCK_MAP プロシージャ」71-9 ページ	インスタンスの共有メモリー内にあるマッピング情報のロックを解除します。

MAP_ELEMENT ファンクション

このファンクションは、elemname で識別される要素のマッピング情報を作成します。明示的な同期化を必要とするライブラリが、マップする要素またはその I/O スタック内のいずれか 1 つの要素を所有している場合（cascade が TRUE の場合）は、最新のマッピング情報を取得できない可能性があります。

構文

```
DBMS_STORAGE_MAP.MAP_ELEMENT(
   elemname          IN VARCHAR2,
   cascade           IN BOOLEAN,
   dictionary_update IN BOOLEAN DEFAULT TRUE);
```

表 71-2 MAP_ELEMENT ファンクションのパラメータ

パラメータ	説明
elemname	マッピング情報を作成する要素。
cascade	TRUE の場合は、elemname の I/O スタック DAG 内の全要素がマップされます。
dictionary_update	TRUE の場合は、データ・ディクショナリのマッピング情報が更新され、変更が反映されます。デフォルト値は TRUE です。dictionary_update はオーバーロードされた引数です。

MAP_FILE ファンクション

このファンクションは、filename で識別されるファイルのマッピング情報を作成します。このファンクションは、ある特定のファイルのマッピングが変更された場合に使用します。Oracle データベース・サーバーでは完全なマッピングを再作成する必要はありません。

このファンクションは、明示的な同期化を必要とするライブラリが、マップするファイルまたはその I/O スタック内のいずれか 1 つの要素を所有している場合（cascade が TRUE の場合）は、最新のマッピング情報を取得できない可能性があります。

構文

```
DBMS_STORAGE_MAP.MAP_FILE(
   filename           IN VARCHAR2,
   filetype           IN VARCHAR2,
   cascade            IN BOOLEAN,
   max_num_fileextent IN NUMBER DEFAULT 100,
   dictionary_update  IN BOOLEAN DEFAULT TRUE);
```

表 71-3 MAP_FILE ファンクションのパラメータ

パラメータ	説明
filename	マッピング情報を作成するファイル。
filetype	マップするファイルのタイプ。"DATAFILE"、"SPFILE"、"TEMPFILE"、"CONTROLFILE"、"LOGFILE" または "ARCHIVEFILE" を指定できます。
cascade	記憶域の再構成が発生する場合にのみ、TRUE を指定してください。ファイル・サイズの変更（ALTER SYSTEM コマンドまたは拡張ファイルの DML 操作のいずれかを使用した）など、他のすべてのインスタンスの場合は、マッピング変更がファイル・エクステントのみに制限されるため、cascade を FALSE に設定できます。 TRUE の場合は、ファイルが常駐している要素に対してもマッピング DAG が作成されます。
max_num_fileextent	マップするファイル・エクステントの最大数を定義します。この指定によって、ファイル・エクステントのマッピン時に使用するメモリー量が制限されます。デフォルト値は 100 です。max_num_fileextent はオーバーロードされた引数です。
dictionary_update	TRUE の場合は、データ・ディクショナリのマッピング情報が更新され、変更が反映されます。デフォルト値は TRUE です。dictionary_update はオーバーロードされた引数です。

MAP_OBJECT ファンクション

このファンクションは、オブジェクト名、所有者およびタイプで識別される Oracle オブジェクトのマッピング情報を作成します。

構文

```
DBMS_STORAGE_MAP.MAP_OBJECT(
   objname IN VARCHAR2,
   owner   IN VARCHAR2,
   objtype IN VARCHAR2);
```

MAP_ALL ファンクション

このファンクションは、すべてのタイプの Oracle ファイルに関する完全なマッピング情報を作成します（アーカイブ・ログを除く）。これには、すべての有向非巡回グラフ（DAG）要素が含まれます。このファンクションは、すべてのマッピング・ライブラリを明示的に同期化するため、最新のマッピング情報を取得します。
MAP_ALL は、コールド・スタートの際に明示的にコールする必要があります。

構文

```
DBMS_STORAGE_MAP.MAP_ALL(
   max_num_fileext   IN NUMBER DEFAULT 100,
   dictionary_update IN BOOLEAN DEFAULT TRUE);
```

表 71-4 MAP_ALL ファンクションのパラメータ

パラメータ	説明
max_num_fileext	マップするファイル・エクステントの最大数を定義します。この指定によって、ファイル・エクステントのマッピン時に使用するメモリー量が制限されます。デフォルト値は 100 です。max_num_fileextent はオーバーロードされた引数です。
dictionary_update	TRUE の場合は、データ・ディクショナリのマッピング情報が更新され、変更が反映されます。デフォルト値は TRUE です。dictionary_update はオーバーロードされた引数です。

DROP_ELEMENT ファンクション

このファンクションは、elemname によって定義された要素のマッピング情報を削除します。

構文

```
DBMS_STORAGE_MAP.DROP_ELEMENT(
   elemname          IN VARCHAR2,
   cascade           IN BOOLEAN,
   dictionary_update IN BOOLEAN DEFAULT TRUE);
```

表 71-5 DROP_ELEMENT ファンクションのパラメータ

パラメータ	説明
elemname	マッピング情報を削除する要素。
cascade	TRUE の場合は、elemname によって定義された DAG の全要素に対して、DROP_ELEMENT が必要に応じて再帰的に起動されます。
dictionary_update	TRUE の場合は、データ・ディクショナリのマッピング情報が更新され、変更が反映されます。デフォルト値は TRUE です。dictionary_update はオーバーロードされた引数です。

DROP_FILE ファンクション

このファンクションは、filename によって定義されたマッピング情報を削除します。

構文

```
DBMS_STORAGE_MAP.DROP_FILE(
   filename          IN VARCHAR2,
   cascade           IN BOOLEAN,
   dictionary_update IN BOOLEAN DEFAULT TRUE);
```

表 71-6 DROP_FILE ファンクションのパラメータ

パラメータ	説明
filename	マッピング情報を削除するファイル。
cascade	TRUE の場合は、ファイルが常駐している要素のマッピング DAG も、必要に応じて削除されます。
dictionary_update	TRUE の場合は、データ・ディクショナリのマッピング情報が更新され、変更が反映されます。デフォルト値は TRUE です。dictionary_update はオーバーロードされた引数です。

DROP_ALL ファンクション

このファンクションは、インスタンスの共有メモリー内にあるマッピング情報をすべて削除します。

構文

```
DBMS_STORAGE_MAP.DROP_ALL(
      dictionary_update IN BOOLEAN DEFAULT TRUE);
```

表 71-7 DROP_ALL ファンクションのパラメータ

パラメータ	説明
dictionary_update	TRUE の場合は、データ・ディクショナリのマッピング情報が更新され、変更が反映されます。デフォルト値は TRUE です。dictionary_update はオーバーロードされた引数です。

SAVE ファンクション

このファンクションは、データ・ディクショナリへの完全なマッピングの再生成に必要な情報を保存します。

構文

```
DBMS_STORAGE_MAP.SAVE();
```

RESTORE ファンクション

このファンクションは、データ・ディクショナリからインスタンスの共有メモリーに、完全なマッピング情報をロードします。RESTORE を起動できるのは、SAVE 操作を実行した後のみです。RESTORE は、ウォーム・スタートの際に明示的にコールする必要があります。

構文

```
DBMS_STORAGE_MAP.RESTORE();
```

LOCK_MAP プロシージャ

このプロシージャは、インスタンスの共有メモリー内にあるマッピング情報をロックします。V$MAP 表の一貫したスナップショットが必要な場合に便利です。マッピング情報をロックしないと、V$MAP_ELEMENT や V$MAP_SUBELEMENT などの一貫性が失われる可能性があります。

構文

```
DBMS_STORAGE_MAP.LOCK_MAP();
```

UNLOCK_MAP プロシージャ

このプロシージャは、インスタンスの共有メモリー内にあるマッピング情報のロックを解除します。

構文

```
DBMS_STORAGE_MAP.UNLOCK_MAP();
```

DBMS_STORAGE_MAP サブプログラムの使用上の注意

MAP_ELEMENT、MAP_FILE および MAP_ALL の場合：構成 ID がサポートされている場合は、情報のマッピングがすでに存在するときにこれらのファンクションを起動すると、マッピングがリフレッシュされます。構成情報 ID がサポートされていない場合は、これらのファンクションの再起動によって、マッピングが再作成されます。

> **関連項目：** 構成 ID、変更される要素またはファイルの属性は、『Oracle9*i* データベース管理者ガイド』を参照してください。

72

DBMS_STREAMS

DBMS_STREAMS パッケージは、SYS.AnyData オブジェクトを論理変更レコード（LCR）に変換し、Streams 属性に関する情報を戻し、セッションで生成された REDO エントリにバイナリ・タグで注釈を付けるためのインタフェースを提供します。このタグは、REDO エントリまたは LCR のバイナリ・タグに対する仕様を含んだルールを持つ、取得プロセス、伝播ジョブまたは適用プロセスの動作に影響を与えます。

この章では、次の項目について説明します。

注意： PUBLIC には、このパッケージの実行権限が付与されます。

関連項目： Streams の詳細は、『Oracle9*i* Streams』を参照してください。

DBMS_STREAMS サブプログラムの要約

表 72-1 DBMS_STREAMS サブプログラム

サブプログラム	説明
「CONVERT_ANYDATA_TO_LCR_DDL ファンクション」72-3 ページ	SYS.AnyData オブジェクトを SYS.LCR$_DDL_RECORD オブジェクトに変換します。
「CONVERT_ANYDATA_TO_LCR_ROW ファンクション」72-4 ページ	SYS.AnyData オブジェクトを SYS.LCR$_ROW_RECORD オブジェクトに変換します。
「GET_INFORMATION ファンクション」72-5 ページ	様々な Streams 属性に関する情報を戻します。
「GET_TAG ファンクション」72-6 ページ	現行のセッションで生成される REDO エントリすべてに対してバイナリ・タグを取得します。
「SET_TAG プロシージャ」72-7 ページ	現行のセッションで継続して生成される REDO エントリすべてに対してバイナリ・タグを設定します。

CONVERT_ANYDATA_TO_LCR_DDL ファンクション

SYS.AnyData オブジェクトを SYS.LCR$_DDL_RECORD オブジェクトに変換します。データ定義言語（DDL）の LCR を SYS.AnyData キューから SYS.LCR$_DDL_RECORD タイプのキューに伝播するときに、ルール・ベースの変換でこのファンクションを指定できます。

他の方法として、DBMS_TRANSFORM パッケージの CREATE_TRANSFORMATION プロシージャで作成される変換で、このファンクションを使用することもできます。この場合は、SYS.AnyData キューから SYS.LCR$_DDL_RECORD タイプのキューに DDL の LCR を伝播するサブスクライバを追加するときに作成する変換を使用してください。

> **関連項目：** このファンクションの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
DBMS_STREAMS.CONVERT_ANYDATA_TO_LCR_DDL(
   source      IN  SYS.AnyData)
RETURN SYS.LCR$_DDL_RECORD;
```

パラメータ

表 72-2 CONVERT_ANYDATA_TO_LCR_DDL ファンクションのパラメータ

パラメータ	説明
source	変換する SYS.AnyData オブジェクト。このオブジェクトが DDL の LCR でない場合は、例外が発生します。

CONVERT_ANYDATA_TO_LCR_ROW ファンクション

SYS.AnyData オブジェクトを SYS.LCR$_ROW_RECORD オブジェクトに変換します。行の LCR を SYS.AnyData キューから SYS.LCR$_ROW_RECORD タイプのキューに伝播するときに、ルール・ベースの変換でこのファンクションを使用できます。

他の方法として、DBMS_TRANSFORM パッケージの CREATE_TRANSFORMATION プロシージャで作成される変換で、このファンクションを使用することもできます。この場合は、SYS.AnyData キューから SYS.LCR$_ROW_RECORD タイプのキューに行の LCR を伝播するサブスクライバを追加するときに作成する変換を使用してください。

> **関連項目：** このファンクションの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
DBMS_STREAMS.CONVERT_ANYDATA_TO_LCR_ROW(
   source      IN  SYS.AnyData)
RETURN SYS.LCR$_ROW_RECORD;
```

パラメータ

表 72-3 CONVERT_ANYDATA_TO_LCR_ROW ファンクションのパラメータ

パラメータ	説明
source	変換する SYS.AnyData オブジェクト。このオブジェクトが行の LCR でない場合は、例外が発生します。

GET_INFORMATION ファンクション

様々な Streams 属性に関する情報を戻します。

構文

```
DBMS_STREAMS.GET_INFORMATION(
   name      IN  VARCHAR2)
RETURN SYS.AnyData;
```

パラメータ

表 72-4 GET_INFORMATION ファンクションのパラメータ

パラメータ	説明
name	取得する情報のタイプ。現在、次の指定が可能です。 ■ SENDER: 現行の LCR の送信者名を戻します（その AQ メッセージのプロパティから）。このファンクションは、適用ハンドラ内部でコールされます。適用ハンドラは、DML ハンドラ、DDL ハンドラ、エラー・ハンドラまたはメッセージ・ハンドラです。適用ハンドラの外部からコールされた場合は、NULL が戻ります。戻り値は、VARCHAR2 として解析されます。 ■ CONSTRAINT_NAME: エラーが発生した LCR について、違反した制約名を戻します。このファンクションは、適用プロセスの DML ハンドラまたはエラー・ハンドラ内部でコールされます。DML ハンドラまたはエラー・ハンドラの外部からコールされた場合は、NULL が戻ります。戻り値は、VARCHAR2 として解析されます。

GET_TAG ファンクション

現行のセッションで生成される REDO エントリすべてに対してバイナリ・タグを取得します。

関連項目： タグの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
DBMS_STREAMS.GET_TAG()
RETURN RAW;
```

使用上の注意

次の例は、現行の LCR タグを出力として表示する方法を示しています。

```
SET SERVEROUTPUT ON
DECLARE
   raw_tag RAW(2000);
BEGIN
   raw_tag := DBMS_STREAMS.GET_TAG();
   DBMS_OUTPUT.PUT_LINE('Tag Value = ' || RAWTOHEX(raw_tag));
END;
/
```

DUAL ビューを問い合せて値を表示することもできます。

```
SELECT DBMS_STREAMS.GET_TAG FROM DUAL;
```

SET_TAG プロシージャ

現行のセッションで継続して生成される REDO エントリすべてに対してバイナリ・タグを設定します。現行セッションの DML 文または DDL 文によって生成される各 REDO エントリには、このタグが含まれます。このプロシージャの影響があるのは、現行のセッションのみです。

注意： このプロシージャはトランザクション型ではありません。つまり、SET_TAG の結果はロールバックできません。

関連項目： タグの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
DBMS_STREAMS.SET_TAG(
   tag   IN RAW   DEFAULT NULL);
```

パラメータ

表 72-5 SET_TAG プロシージャのパラメータ

パラメータ	説明
tag	現行のセッションで継続して生成される REDO エントリすべてに対するバイナリ・タグ。RAW 値は一連のバイトで、バイトは一連のビットです。 デフォルトでは、セッションのタグは NULL です。 タグ値のサイズ制限は、2000 バイトです。

使用上の注意

現行のセッションでタグを 16 進数値の '17' に設定するには、次のプロシージャを実行します。

```
EXEC DBMS_STREAMS.SET_TAG(tag => HEXTORAW('17'));
```

73

DBMS_STREAMS_ADM

DBMS_STREAMS_ADM パッケージは、表、スキーマおよびデータベースのレベルでの取得、伝播および適用に関する単純なルールを、変換せずに追加および削除するための管理プロシージャを提供します。これらのルールは、論理変更レコード（LCR）をサポートしています。この LCR には、行の LCR とデータ定義言語（DDL）の LCR が含まれます。また、このパッケージには、キューを作成するためのプロシージャや、Streams のメタデータ（データ・ディクショナリ情報など）を管理するためのプロシージャも含まれています。

この章では、次の項目について説明します。

さらに複雑なルールや、LCR ではないイベントを含むルールが必要な場合は、DBMS_RULE_ADM パッケージを使用できます。

単純なルールの変換が必要な場合は、DBMS_RULE_ADM パッケージを使用して、DBMS_STREAMS_ADM パッケージで作成したルールに関する変換を追加、更新または削除することができます。

関連項目：

- Streams の詳細は、『Oracle9*i* Streams』を参照してください。
- 第 64 章「DBMS_RULE_ADM」

DBMS_STREAMS_ADM サブプログラムの要約

表 73-1 DBMS_STREAMS_ADM サブプログラム

サブプログラム	説明
「ADD_GLOBAL_PROPAGATION_RULES プロシージャ」73-3 ページ	ソース・キューの LCR すべてを宛先キューに伝播する伝播ルールを追加します。
「ADD_GLOBAL_RULES プロシージャ」73-6 ページ	データベース全体に対する取得ルールまたはキューの LCR すべてに対する適用ルールを追加します。
「ADD_SCHEMA_PROPAGATION_RULES プロシージャ」73-10 ページ	ソース・キューの指定スキーマに関係する LCR を宛先キューに伝播する伝播ルールを追加します。
「ADD_SCHEMA_RULES プロシージャ」73-14 ページ	スキーマに対する取得ルールまたは適用ルールを追加します。
「ADD_SUBSET_RULES プロシージャ」73-18 ページ	表内の行のサブセットに対して適用ルールを追加します。
「ADD_TABLE_PROPAGATION_RULES プロシージャ」73-22 ページ	ソース・キューの指定表に関係する LCR を宛先キューに伝播する伝播ルールを追加します。
「ADD_TABLE_RULES プロシージャ」73-25 ページ	表に対する取得ルールまたは適用ルールを追加します。
「PURGE_SOURCE_CATALOG プロシージャ」73-29 ページ	ローカル・データベースにある、指定オブジェクトに関する全 Streams のデータ・ディクショナリ情報を削除します。
「REMOVE_RULE プロシージャ」73-31 ページ	指定のルールまたはすべてのルールを、指定の取得プロセス、適用プロセスまたは伝播ジョブに関連付けられているルール・セットから削除します。
「SET_UP_QUEUE プロシージャ」73-32 ページ	Streams の取得、伝播および適用機能で使用するキュー表とキューを作成します。

注意： 特に指定がないかぎり、すべてのプロシージャがコミットされます。

ADD_GLOBAL_PROPAGATION_RULES プロシージャ

ソース・キューの LCR すべてを宛先キューに伝播する伝播ルールを追加します。また、このプロシージャは、必要に応じて現行のユーザーを使用して伝播を構成し、デフォルトの伝播スケジュールを確立します。このプロシージャは、フィルタ条件に従って、ソース・キューの全 LCR の宛先キューへの伝播を可能にします。ソース・キューと宛先キューの間で許可される伝播ジョブは 1 つのみです。

伝播ルールを追加すると、伝播ジョブによって、DML または DDL の変更（あるいはその両方）が指定のソース・キューから指定の宛先キューに伝播されます。このプロシージャは、include_dml パラメータ値と include_ddl パラメータ値に基づいて、DML と DDL のルールをそれぞれ自動的に作成します。システム生成のルールには、データベース名とそれに付加された順序番号で構成された名前が付きます。この順序番号によって、名前の競合が回避されます。データベース名と順序番号を組み合せた名前が長すぎる場合は、データベース名が切り捨てられます。オーバーロードされた ADD_GLOBAL_PROPAGATION_RULES プロシージャの場合は、DML と DDL の変更に対するシステム生成のルール名が戻ります。

伝播ジョブは、作成されたルールをフィルタ処理に使用します。伝播ジョブにルール・セットがない場合は、ルール・セットが自動的に作成され、データベースに変更を伝播するルールがルール・セットに追加されます。伝播ジョブの既存ルール・セットにある他のルールは、この影響を受けません。DBMS_RULE_ADM パッケージを使用すると、別のルールを追加できます。

次に、グローバル・ルール条件の例を示します。この条件は、伝播ジョブによる DML 変更の伝播用に作成できます。

```
:dml.get_source_database_name() = 'DBS1.NET' AND :dml.is_null_tag() = 'Y'
```

注意： 前述の例の引用符は、すべて一重引用符です。

伝播が適切に動作するために、ソース・キューの所有者には伝播イベントに必要な権限を付与してください。

注意：

- 現在は、データベース・リンクによってイベントが複数の宛先キューに伝播される場合でも、単一の伝播ジョブによって、特定のデータベース・リンクを使用するイベントすべてが伝播されます。
- ソース・キューの所有者は伝播を実行しますが、伝播ジョブの所有者はジョブを作成したユーザーです。これらのユーザーは、同一のユーザーまたは別のユーザーの場合があります。

関連項目： 必要な権限の詳細は、47-4 ページの「CREATE_PROPAGATION プロシージャ」を参照してください。

構文

```
DBMS_STREAMS_ADM.ADD_GLOBAL_PROPAGATION_RULES(
   streams_name             IN  VARCHAR2,
   source_queue_name        IN  VARCHAR2,
   destination_queue_name   IN  VARCHAR2,
   include_dml              IN  BOOLEAN   DEFAULT true,
   include_ddl              IN  BOOLEAN   DEFAULT false,
   include_tagged_lcr       IN  BOOLEAN   DEFAULT false,
   source_database          IN  VARCHAR2  DEFAULT NULL,
   dml_rule_name            OUT VARCHAR2,
   ddl_rule_name            OUT VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。このプロシージャの一方のバージョンには、2 つの OUT パラメータがあり、他方のバージョンにはこれらのパラメータはありません。

パラメータ

表 73-2 ADD_GLOBAL_PROPAGATION_RULES プロシージャのパラメータ

パラメータ	説明
streams_name	伝播ジョブの名前。 指定の伝播ジョブが存在しない場合は、自動的に作成されます。 この指定が NULL で、同じソース・キューと宛先キュー（データベース・リンクを含む）を指定した伝播ジョブがすでにある場合は、その伝播ジョブが使用されます。 この指定が NULL で、同じソース・キューと宛先キュー（データベース・リンクを含む）を指定した伝播ジョブがない場合は、システム生成の名前で伝播ジョブが自動的に作成されます。
source_queue_name	ソース・キューの名前。現行のデータベースにはソース・キューが含まれている必要があります。

表 73-2 ADD_GLOBAL_PROPAGATION_RULES プロシージャのパラメータ（続き）

パラメータ	説明
destination_queue_name	宛先キューの名前（STREAMS_QUEUE@DBS2 などのデータベース・リンクを含む）。 データベース・リンクが省略されている場合は、現行のデータベースのグローバル名が使用されます。この場合、ソース・キューと宛先キューは同じデータベース内にあることが必要です。 **注意：**接続修飾子は使用できません。
include_dml	TRUE の場合は、DML 変更のためのルールが作成されます。FALSE の場合、DML ルールは作成されません。NULL は許可されません。
include_ddl	TRUE の場合は、DDL 変更のためのルールが作成されます。FALSE の場合、DDL ルールは作成されません。NULL は許可されません。
include_tagged_lcr	TRUE の場合は、LCR に NULL 以外のタグがあるかどうかに関係なく、LCR が常に伝播の対象として考慮されます。この設定は、データベースの完全な（スタンバイなど）コピーに適しています。 FALSE の場合は、LCR に NULL のタグが含まれている場合にのみ、伝播に LCR が考慮されます。FALSE の設定は、通常、どこでも更新可能な構成で、ソース・データベースに変更が戻ることを回避するために、指定されます。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。
source_database	ソース・データベースのグローバル名。ソース・データベースは、変更が発生したデータベースです。NULL の場合、ソース・データベースに関係する条件は、生成されるルールに追加されません。 ドメイン名が未指定の場合は、データベース名に自動的に追加されます。たとえば、ドメイン名が .NET の場合に DBS1 を指定すると、自動的に DBS1.NET が指定されます。 オラクル社では、伝播ルールにソース・データベースを指定することをお薦めします。

表 73-2 ADD_GLOBAL_PROPAGATION_RULES プロシージャのパラメータ（続き）

パラメータ	説明
dml_rule_name	include_dml が TRUE の場合は、DML ルール名が含まれます。 include_dml が FALSE の場合は、NULL が含まれます。
ddl_rule_name	include_ddl が TRUE の場合は、DDL ルール名が含まれます。 include_ddl が FALSE の場合は、NULL が含まれます。

ADD_GLOBAL_RULES プロシージャ

データベース全体に対する取得ルールまたはキューの LCR すべてに対する適用ルールを追加します。

取得ルールを追加すると、DML または DDL の変更（あるいはその両方）が現行のデータベースに取得され、指定のキューにエンキューされます。取得ルールの場合は、このプロシージャをソース・データベースで実行してください。このプロシージャは、DBMS_CAPTURE_ADM パッケージの PREPARE_GLOBAL_INSTANTIATION プロシージャを自動的に起動します。

適用ルールを追加すると、適用プロセスは取得イベントを受け取って適用します。このイベントには、source_database パラメータと一致するソース・データベースで発生した DML または DDL の変更（あるいはその両方）が含まれています。適用ルールの場合は、このプロシージャを接続先データベースで実行してください。

このプロシージャによって作成された適用プロセスの適用先は、ローカル・データベースでのイベントと取得イベントのみです。リモート・データベースでのイベントまたはユーザーがエンキューしたイベントに対する適用プロセスを作成するには、DBMS_APPLY_ADM パッケージの CREATE_APPLY プロシージャを使用します。

このプロシージャで作成した適用プロセスが適用した変更によって、'00'（2 桁のゼロ）の値のタグが接続先データベースの REDO ログに生成されます。DBMS_APPLY_ADM パッケージの ALTER_APPLY プロシージャを使用すると、適用プロセスの作成後に必要に応じてタグの値を変更できます。

適用プロセスは、DBMS_APPLY_ADM.CREATE_APPLY プロシージャを使用し、その実行時に、デフォルト以外の値を apply_captured、apply_database_link および apply_tag の各パラメータに指定して作成することもできます。その後、この ADD_GLOBAL_RULES プロシージャを使用して、その適用プロセスで使用するルール・セットにルールを追加できます。

このプロシージャは、include_dml パラメータ値と include_ddl パラメータ値に基づいて、DML と DDL のルールをそれぞれ自動的に作成します。システム生成のルールには、

データベース名とそれに付加された順序番号で構成された名前が付きます。この順序番号によって、名前の競合が回避されます。データベース名と順序番号を組み合せた名前が長すぎる場合は、データベース名が切り捨てられます。

オーバーロードされた ADD_GLOBAL_RULES プロシージャの場合は、DML と DDL の変更に対するシステム生成のルール名が戻ります。

取得プロセスまたは適用プロセスは、作成されたルールをフィルタ処理に使用します。生成されたプロセスにルール・セットがない場合は、ルール・セットが自動的に作成され、ルールがそのルール・セットに追加されます。そのプロセスの既存ルール・セットにある他のルールは、この影響を受けません。DBMS_RULE_ADM パッケージを使用すると、別のルールを追加できます。

次に、グローバル・ルール条件の例を示します。この条件は、取得プロセスによる DML 変更の取得用に作成できます。

```
:dml.is_null_tag() = 'Y'
```

次に、グローバル・ルール条件の例を示します。この条件は、適用プロセスによる DML 変更の適用のために作成できます。

```
:dml.get_source_database_name() = 'DBS1.NET' AND :dml.is_null_tag() = 'Y'
```

注意： 前述の例の引用符は、すべて一重引用符です。

このプロシージャによって取得プロセスまたは適用プロセスを作成する場合、このプロシージャを実行するユーザーは、変更を取得または適用するユーザーとなります。指定されたユーザーには、これらのアクションの実行に必要な権限を付与してください。

関連項目：

- 第 64 章「DBMS_RULE_ADM」
- 変更の取得に必要な権限は、8-6 ページの「CREATE_CAPTURE プロシージャ」を参照してください。
- 変更の適用に必要な権限は、4-9 ページの「CREATE_APPLY プロシージャ」を参照してください（apply_user パラメータを参照）。

構文

```
DBMS_STREAMS_ADM.ADD_GLOBAL_RULES(
   streams_type             IN  VARCHAR2,
   streams_name             IN  VARCHAR2 DEFAULT NULL,
   queue_name               IN  VARCHAR2 DEFAULT 'streams_queue',
   include_dml              IN  BOOLEAN  DEFAULT true,
   include_ddl              IN  BOOLEAN  DEFAULT false,
```

```
include_tagged_lcr       IN  BOOLEAN  DEFAULT false,
source_database          IN  VARCHAR2 DEFAULT NULL,
dml_rule_name            OUT VARCHAR2,
ddl_rule_name            OUT VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。このプロシージャの一方のバージョンには、2 つの OUT パラメータがあり、他方のバージョンにはこれらのパラメータはありません。

パラメータ

表 73-3 ADD_GLOBAL_RULES プロシージャのパラメータ

パラメータ	説明
streams_type	プロセスのタイプ。capture または apply です。
streams_name	取得プロセスまたは適用プロセスの名前。 指定のプロセスが存在しない場合は、自動的に作成されます。 この指定が NULL で、キューに関連する取得プロセスまたは適用プロセスが存在している場合は、その関連プロセスが使用されます。キューに関連するプロセスがない場合は、システム生成の名前を使用して取得プロセスまたは適用プロセスが自動的に作成されます。この指定が NULL で、キューに対して指定した streams_type のプロセスが複数存在する場合は、エラーが発生します。
queue_name	ローカル・キューの名前。取得ルールの場合は、変更がエンキューされるキューを指定します。適用ルールの場合は、変更がデキューされるキューを指定します。
include_dml	TRUE の場合は、DML 変更のためのルールが作成されます。FALSE の場合、DML ルールは作成されません。NULL は許可されません。
include_ddl	TRUE の場合は、DDL 変更のためのルールが作成されます。FALSE の場合、DDL ルールは作成されません。NULL は許可されません。

表 73-3 ADD_GLOBAL_RULES プロシージャのパラメータ（続き）

パラメータ	説明
include_tagged_lcr	TRUE の場合は、REDO エントリまたは LCR に NULL でないタグがあるかどうかに関係なく、取得では REDO エントリが、適用では LCR が常に処理の対象として考慮されます。この設定は、データベースの完全な（スタンバイなど）コピーに適しています。 FALSE の場合は、REDO エントリまたは LCR に NULL のタグが含まれている場合にのみ、取得については REDO エントリ、適用については LCR が考慮されます。FALSE の設定は、通常、どこでも更新可能な構成で、ソース・データベースに変更が戻ることを回避するために、指定されます。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。
source_database	ソース・データベースのグローバル名。NULL の場合、ソース・データベースに関係する条件は、生成されるルールに追加されません。 取得ルールの場合、現在、取得データベースとソース・データベースは必ず同一であるため、NULL を指定できます。 適用ルールの場合は、適用プロセスによって適用される変更のソース・データベースを指定します。ソース・データベースは、変更が発生したデータベースです。取得イベントを適用する場合、適用プロセスによって適用できるのは、1 つのソース・データベースでの 1 つの取得プロセスによるイベントのみです。 ドメイン名が未指定の場合は、データベース名に自動的に追加されます。たとえば、ドメイン名が .NET の場合に DBS1 を指定すると、自動的に DBS1.NET が指定されます。
dml_rule_name	include_dml が TRUE の場合は、DML ルール名が含まれます。 include_dml が FALSE の場合は、NULL が含まれます。
ddl_rule_name	include_ddl が TRUE の場合は、DDL ルール名が含まれます。 include_ddl が FALSE の場合は、NULL が含まれます。

ADD_SCHEMA_PROPAGATION_RULES プロシージャ

ソース・キューの指定スキーマに関係する LCR を宛先キューに伝播する伝播ルールを追加します。また、このプロシージャは、必要に応じて現行のユーザーを使用して伝播を構成し、デフォルトの伝播スケジュールを確立します。このプロシージャを使用すると、フィルタ条件に従って指定スキーマの LCR を伝播できます。ソース・キューと宛先キューの間で許可される伝播ジョブは 1 つのみです。

伝播ルールを追加すると、伝播ジョブによって、指定のスキーマに関連する DML または DDL の変更（あるいはその両方）が指定のソース・キューから指定の宛先キューに伝播されます。このプロシージャは、include_dml パラメータ値と include_ddl パラメータ値に基づいて、DML と DDL のルールをそれぞれ自動的に作成します。システム生成のルールには、スキーマ名とそれに付加された順序番号で構成された名前が付きます。この順序番号によって、名前の競合が回避されます。スキーマ名と順序番号を組み合せた名前が長すぎる場合は、スキーマ名が切り捨てられます。オーバーロードされた ADD_SCHEMA_PROPAGATION_RULES プロシージャの場合は、DML と DDL の変更に対するシステム生成のルール名が戻ります。

伝播ジョブは、作成されたルールをフィルタ処理に使用します。伝播ジョブにルール・セットがない場合は、ルール・セットが自動的に作成され、スキーマに変更を伝播するルールがルール・セットに追加されます。伝播ジョブの既存ルール・セットにある他のルールは、この影響を受けません。DBMS_RULE_ADM パッケージを使用すると、別のルールを追加できます。

次に、スキーマ・ルール条件の例を示します。この条件は、伝播ジョブによる DML 変更の伝播用に作成できます。

```
:dml.get_object_owner() = 'HR' AND :dml.is_null_tag() = 'Y'
AND :dml.get_source_database_name() = 'DBS1.NET'
```

注意： 前述の例の引用符は、すべて一重引用符です。

伝播が適切に動作するために、ソース・キューの所有者には伝播イベントに必要な権限を付与してください。

注意：

- 現在は、データベース・リンクによってイベントが複数の宛先キューに伝播される場合でも、単一の伝播ジョブによって、特定のデータベース・リンクを使用するイベントすべてが伝播されます。
- ソース・キューの所有者は伝播を実行しますが、伝播ジョブの所有者はジョブを作成したユーザーです。これらのユーザーは、同一のユーザーまたは別のユーザーの場合があります。

関連項目： 必要な権限の詳細は、47-4 ページの「CREATE_PROPAGATION プロシージャ」を参照してください。

構文

```
DBMS_STREAMS_ADM.ADD_SCHEMA_PROPAGATION_RULES(
   schema_name              IN  VARCHAR2,
   streams_name             IN  VARCHAR2,
   source_queue_name        IN  VARCHAR2,
   destination_queue_name   IN  VARCHAR2,
   include_dml              IN  BOOLEAN  DEFAULT true,
   include_ddl              IN  BOOLEAN  DEFAULT false,
   include_tagged_lcr       IN  BOOLEAN  DEFAULT false,
   source_database          IN  VARCHAR2 DEFAULT NULL,
   dml_rule_name            OUT VARCHAR2,
   ddl_rule_name            OUT VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。このプロシージャの一方のバージョンには、2 つの OUT パラメータがあり、他方のバージョンにはこれらのパラメータはありません。

パラメータ

表 73-4 ADD_SCHEMA_PROPAGATION_RULES プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマ名。たとえば、hr のように指定します。
streams_name	伝播ジョブの名前。 指定の伝播ジョブが存在しない場合は、自動的に作成されます。 この指定が NULL で、同じソース・キューと宛先キュー（データベース・リンクを含む）を指定した伝播ジョブがすでにある場合は、その伝播ジョブが使用されます。 この指定が NULL で、同じソース・キューと宛先キュー（データベース・リンクを含む）を指定した伝播ジョブがない場合は、システム生成の名前で伝播ジョブが自動的に作成されます。
source_queue_name	ソース・キューの名前。現行のデータベースにはソース・キューが含まれている必要があります。
destination_queue_name	宛先キューの名前（STREAMS_QUEUE@DBS2 などのデータベース・リンクを含む）。 データベース・リンクが省略されている場合は、現行のデータベースのグローバル名が使用されます。この場合、ソース・キューと宛先キューは同じデータベース内にあることが必要です。 **注意：**接続修飾子は使用できません。
include_dml	TRUE の場合は、DML 変更のためのルールが作成されます。FALSE の場合、DML ルールは作成されません。NULL は許可されません。
include_ddl	TRUE の場合は、DDL 変更のためのルールが作成されます。FALSE の場合、DDL ルールは作成されません。NULL は許可されません。

表 73-4 ADD_SCHEMA_PROPAGATION_RULES プロシージャのパラメータ（続き）

パラメータ	説明
include_tagged_lcr	TRUE の場合は、LCR に NULL 以外のタグがあるかどうかに関係なく、LCR が常に伝播の対象に考慮されます。この設定は、データベースの完全な（スタンバイなど）コピーに適しています。 FALSE の場合は、LCR に NULL のタグが含まれている場合にのみ、伝播に LCR が考慮されます。FALSE の設定は、通常、どこでも更新可能な構成で、ソース・データベースに変更が戻ることを回避するために、指定されます。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。
source_database	ソース・データベースのグローバル名。ソース・データベースは、変更が発生したデータベースです。NULL の場合、ソース・データベースに関係する条件は、生成されるルールに追加されません。 ドメイン名が未指定の場合は、データベース名に自動的に追加されます。たとえば、ドメイン名が .NET の場合に DBS1 を指定すると、自動的に DBS1.NET が指定されます。 オラクル社では、伝播ルールにソース・データベースを指定することをお薦めします。
dml_rule_name	include_dml が TRUE の場合は、DML ルール名が含まれます。 include_dml が FALSE の場合は、NULL が含まれます。
ddl_rule_name	include_ddl が TRUE の場合は、DDL ルール名が含まれます。 include_ddl が FALSE の場合は、NULL が含まれます。

ADD_SCHEMA_RULES プロシージャ

スキーマの取得ルールまたは適用ルールを追加します。

取得ルールを追加すると、取得プロセスによって、DML または DDL の変更（あるいはその両方）が指定したスキーマについて取得され、指定キューにエンキューされます。取得ルールの場合は、このプロシージャをソース・データベースで実行してください。このプロシージャは、DBMS_CAPTURE_ADM パッケージの PREPARE_SCHEMA_INSTANTIATION プロシージャを指定スキーマに対して自動的に起動します。

適用ルールを追加すると、適用プロセスは取得イベントを受け取って適用します。このイベントには、指定スキーマの DML または DDL の変更（あるいはその両方）が含まれています。適用ルールの場合は、このプロシージャを接続先データベースで実行してください。

このプロシージャによって作成された適用プロセスの適用先は、ローカル・データベースでのイベントと取得イベントのみです。リモート・データベースでのイベントまたはユーザーがエンキューしたイベントに対する適用プロセスを作成するには、DBMS_APPLY_ADM パッケージの CREATE_APPLY プロシージャを使用します。

このプロシージャで作成した適用プロセスが適用した変更によって、'00'（2 桁のゼロ）の値のタグが接続先データベースの REDO ログに生成されます。DBMS_APPLY_ADM パッケージの ALTER_APPLY プロシージャを使用すると、適用プロセスの作成後に必要に応じてタグの値を変更できます。

適用プロセスは、DBMS_APPLY_ADM.CREATE_APPLY プロシージャを使用し、その実行時に、デフォルト以外の値を apply_captured、apply_database_link および apply_tag の各パラメータに指定して作成することもできます。その後、この ADD_SCHEMA_RULES プロシージャを使用して、その適用プロセスで使用するルール・セットにルールを追加できます。

このプロシージャは、include_dml パラメータ値と include_ddl パラメータ値に基づいて、DML と DDL のルールをそれぞれ自動的に作成します。システム生成のルールには、スキーマ名とそれに付加された順序番号で構成された名前が付きます。この順序番号によって、名前の競合が回避されます。スキーマ名と順序番号を組み合せた名前が長すぎる場合は、スキーマ名が切り捨てられます。

次に、スキーマ・ルール条件の例を示します。この条件は、DML 文のフィルタ処理用に作成できます。

```
:dml.get_object_owner() = 'HR' AND :dml.is_null_tag() = 'Y'
```

注意： 前述の例の引用符は、すべて一重引用符です。

オーバーロードされた ADD_SCHEMA_RULES プロシージャの場合は、DML と DDL の変更に対するシステム生成のルール名が戻ります。

取得プロセスまたは適用プロセスは、作成されたルールをフィルタ処理に使用します。プロセスにルール・セットがない場合は、ルール・セットが自動的に作成され、スキーマのルールがそのルール・セットに追加されます。そのプロセスの既存ルール・セットにある他のルールは、この影響を受けません。DBMS_RULE_ADM パッケージを使用すると、別のルールを追加できます。

このプロシージャによって取得プロセスまたは適用プロセスを作成する場合、このプロシージャを実行するユーザーは、変更を取得または適用するユーザーとなります。指定されたユーザーには、これらのアクションの実行に必要な権限を付与してください。

関連項目：

- 第 64 章「DBMS_RULE_ADM」
- 変更の取得に必要な権限は、8-6 ページの「CREATE_CAPTURE プロシージャ」を参照してください。
- 変更の適用に必要な権限は、4-9 ページの「CREATE_APPLY プロシージャ」を参照してください（apply_user パラメータを参照）。

構文

```
DBMS_STREAMS_ADM.ADD_SCHEMA_RULES(
   schema_name              IN  VARCHAR2,
   streams_type             IN  VARCHAR2,
   streams_name             IN  VARCHAR2 DEFAULT NULL,
   queue_name               IN  VARCHAR2 DEFAULT 'streams_queue',
   include_dml              IN  BOOLEAN  DEFAULT true,
   include_ddl              IN  BOOLEAN  DEFAULT false,
   include_tagged_lcr       IN  BOOLEAN  DEFAULT false,
   source_database          IN  VARCHAR2 DEFAULT NULL,
   dml_rule_name            OUT VARCHAR2,
   ddl_rule_name            OUT VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。このプロシージャの一方のバージョンには、2 つの OUT パラメータがあり、他方のバージョンにはこれらのパラメータはありません。

パラメータ

表 73-5 ADD_SCHEMA_RULES プロシージャのパラメータ

パラメータ	説明
schema_name	スキーマ名。たとえば、hr のように指定します。 Streams ではスキーマの存在を検証しないため、まだ存在していないスキーマを指定できます。
streams_type	プロセスのタイプ。capture または apply です。
streams_name	プロセス名。 指定のプロセスが存在しない場合は、自動的に作成されます。 この指定が NULL で、キューに関連する取得プロセスまたは適用プロセスが存在している場合は、その関連プロセスが使用されます。キューに関連する取得プロセスまたは適用プロセスがない場合は、システム生成の名前を使用して取得プロセスまたは適用プロセスが自動的に作成されます。この指定が NULL で、キューに対して指定した streams_type のプロセスが複数存在する場合は、エラーが発生します。
queue_name	ローカル・キューの名前。取得ルールの場合は、変更がエンキューされるキューを指定します。適用ルールの場合は、変更がデキューされるキューを指定します。
include_dml	TRUE の場合は、DML 変更のためのルールが作成されます。FALSE の場合、DML ルールは作成されません。NULL は許可されません。
include_ddl	TRUE の場合は、DDL 変更のためのルールが作成されます。FALSE の場合、DDL ルールは作成されません。NULL は許可されません。
include_tagged_lcr	TRUE の場合は、REDO エントリまたは LCR に NULL でないタグがあるかどうかに関係なく、取得では REDO エントリが、適用では LCR が常に処理の対象として考慮されます。この設定は、データベースの完全な（スタンバイなど）コピーに適しています。 FALSE の場合は、REDO エントリまたは LCR に NULL のタグが含まれている場合にのみ、取得については REDO エントリ、適用については LCR が考慮されます。FALSE の設定は、通常、どこでも更新可能な構成でソース・データベースに変更が戻ることを回避するために、指定されます。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。

表 73-5 ADD_SCHEMA_RULES プロシージャのパラメータ（続き）

パラメータ	説明
source_database	ソース・データベースのグローバル名。NULL の場合、ソース・データベースに関係する条件は、生成されるルールに追加されません。 取得ルールの場合、現在、取得データベースとソース・データベースは必ず同一であるため、NULL を指定できます。 適用ルールの場合は、適用プロセスによって適用される変更のソース・データベースを指定します。ソース・データベースは、変更が発生したデータベースです。取得イベントを適用する場合、適用プロセスによって適用できるのは、1 つのソース・データベースでの 1 つの取得プロセスによるイベントのみです。 ドメイン名が未指定の場合は、データベース名に自動的に追加されます。たとえば、ドメイン名が .NET の場合に DBS1 を指定すると、自動的に DBS1.NET が指定されます。
dml_rule_name	include_dml が TRUE の場合は、DML ルール名が含まれます。 include_dml が FALSE の場合は、NULL が含まれます。
ddl_rule_name	include_ddl が TRUE の場合は、DDL ルール名が含まれます。 include_ddl が FALSE の場合は、NULL が含まれます。

ADD_SUBSET_RULES プロシージャ

表内の行のサブセットに対して適用ルールを追加します。

適用プロセスは、取得イベントを受け取って適用します。このイベントには、表内の行の指定サブセットに対する DML 変更が含まれています。このプロシージャは、接続先データベースで実行してください。

このプロシージャを実行すると、指定の適用プロセスに対して 3 つのルール（INSERT 文、UPDATE 文および DELETE 文）が生成されます。INSERT 文と DELETE 文については、dml_condition パラメータに指定された条件を満たす行 LCR にのみ適用されます。UPDATE 文については、次のバリエーションがあります。

- 行 LCR の新旧両方の値が指定の dml_condition を満たす場合、その LCR は変更なしで適用されます。
- 行 LCR の新旧両方の値が指定の dml_condition を満たさない場合、その行 LCR は適用されません。
- 行 LCR の古い値が指定の dml_condition を満たし、新しい値が満たさない場、その行 LCR は削除されるものとして変換されます。
- 行 LCR の新しい値が指定の dml_condition を満たし、古い値が満たさない場合、その行 LCR は挿入されるものとして変換されます。

次に、ルール条件の例を示します。この条件は、dml_condition が region_id = 2、table_name が hr.regions の場合に、更新操作を含む LCR のフィルタ処理用に作成できます。

```
:dml.get_object_owner() = 'HR' AND :dml.get_object_name() = 'REGIONS' AND
:dml.is_null_tag() = 'Y' AND :dml.get_command_type() = 'UPDATE' AND
(:dml.get_value('NEW','"REGION_ID"') IS NOT NULL) AND
(:dml.get_value('OLD','"REGION_ID"') IS NOT NULL) AND
(:dml.get_value('OLD','"REGION_ID"').AccessNumber()=2) AND
(:dml.get_value('NEW','"REGION_ID"').AccessNumber()=2)
```

注意： 前述の例の引用符は、すべて一重引用符です。

適用プロセスは、生成されたルールを LCR のフィルタ処理に使用します。適用プロセスにルール・セットがない場合は、ルール・セットが自動的に作成され、表に対するルールがそのルール・セットに追加されます。適用プロセスの既存ルール・セットにある他のルールは、この影響を受けません。DBMS_RULE_ADM パッケージを使用すると、別のルールを追加できます。

このプロシージャを実行すると、INSERT 文、UPDATE 文および DELETE 文のルールが自動的に作成されます。これらのルールには、システム生成のルール名が指定されます。システ

ム生成のルールには、表名とそれに付加された順序番号で構成される名前が付きます。この順序番号によって、名前の競合が回避されます。表名と順序番号を組み合せた名前が長すぎる場合は、表名が切り捨てられます。ADD_SUBSET_RULES プロシージャはオーバーロードされているため、INSERT 文、UPDATE 文および DELETE 文に対するシステム生成のルール名が戻されます。

このプロシージャによって作成された適用プロセスの適用先は、ローカル・データベースでのイベントと取得イベントのみです。リモート・データベースでのイベントまたはユーザーがエンキューしたイベントに対する適用プロセスを作成するには、DBMS_APPLY_ADM パッケージの CREATE_APPLY プロシージャを使用します。

このプロシージャで作成した適用プロセスが適用した変更によって、'00'（2 桁のゼロ）の値のタグが接続先データベースの REDO ログに生成されます。DBMS_APPLY_ADM パッケージの ALTER_APPLY プロシージャを使用すると、適用プロセスの作成後に必要に応じてタグの値を変更できます。

適用プロセスは、DBMS_APPLY_ADM.CREATE_APPLY プロシージャを使用し、その実行時に、デフォルト以外の値を apply_captured、apply_database_link および apply_tag の各パラメータに指定して作成することもできます。その後、この ADD_SUBSET_RULES プロシージャを使用して、その適用プロセスで使用するルール・セットにルールを追加できます。

表のサブセット・ルールを作成する場合は、表のすべての列を保持するソース・データベースで、無条件のサプリメンタル・ログ・グループを作成してください。サプリメンタル・ロギングは、更新を挿入に変換する必要がある場合に必要です。挿入を正しく実行するために、適用プロセスではすべての列値が必要です。

このプロシージャによって適用プロセスを作成する場合、このプロシージャを実行するユーザーは、変更を適用するユーザーとなります。この指定ユーザーには、イベントを適用するための権限が必要です。

関連項目：

- 第 64 章「DBMS_RULE_ADM」
- 変更の取得に必要な権限は、8-6 ページの「CREATE_CAPTURE プロシージャ」を参照してください。
- 変更の適用に必要な権限は、4-9 ページの「CREATE_APPLY プロシージャ」を参照してください（apply_user パラメータを参照）。

構文

```
DBMS_STREAMS_ADM.ADD_SUBSET_RULES(
   table_name               IN   VARCHAR2,
   dml_condition            IN   VARCHAR2,
   streams_type             IN   VARCHAR2 DEFAULT 'apply',
   streams_name             IN   VARCHAR2 DEFAULT NULL,
   queue_name               IN   VARCHAR2 DEFAULT 'streams_queue',
   include_tagged_lcr       IN   BOOLEAN  DEFAULT false,
   source_database          IN   VARCHAR2 DEFAULT NULL,
   insert_rule_name         OUT  VARCHAR2,
   update_rule_name         OUT  VARCHAR2,
   delete_rule_name         OUT  VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。このプロシージャの一方のバージョンには、3 つの OUT パラメータがあり、他方のバージョンにはこれらのパラメータはありません。

パラメータ

表 73-6 ADD_SUBSET_RULES プロシージャのパラメータ

パラメータ	説明
table_name	表の名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr.employees のようにします。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 Streams では表の存在を検証しないため、まだ存在していない表を指定できます。
dml_condition	サブセット条件。SQL の WHERE 句に条件を指定する場合と同様に、この条件を指定します。 たとえば、salary が 4000 を超え、かつ job_id が SA_MAN である hr.employees 表の行を指定するには、次のように条件を入力します。 `' salary > 4000 and job_id = ''SA_MAN'' '` **注意：** 前述の例の引用符は、すべて一重引用符です。
streams_type	プロセスのタイプ。現在は、apply のみ有効です。

表 73-6 ADD_SUBSET_RULES プロシージャのパラメータ（続き）

パラメータ	説明
streams_name	適用プロセスの名前。指定の適用プロセスが存在しない場合は、自動的に作成されます。 NULL の場合は、キューの適用プロセスが使用されます。キューに適用プロセスがない場合は、システム生成の名前で適用プロセスが自動的に作成されます。複数の適用プロセスがある場合は、エラーが発生します。
queue_name	変更がデキューされるローカル・キューの名前。
include_tagged_lcr	TRUE の場合は、LCR に NULL 以外のタグがあるかどうかに関係なく、適用に LCR が常に処理の対象として考慮されます。 FALSE の場合は、LCR に NULL のタグが含まれている場合にのみ、適用に LCR が考慮されます。FALSE の設定は、通常、どこでも更新可能な構成で、ソース・データベースに変更が戻ることを回避するために、指定されます。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。
source_database	ソース・データベースのグローバル名。NULL の場合、ソース・データベースに関係する条件は、生成されるルールに追加されません。 適用プロセスによって適用される変更のソース・データベースを指定します。ソース・データベースは、変更が発生したデータベースです。取得イベントを適用する場合、適用プロセスによって適用できるのは、1 つのソース・データベースでの 1 つの取得プロセスによるイベントのみです。 ドメイン名が未指定の場合は、データベース名に自動的に追加されます。たとえば、ドメイン名が .NET の場合に DBS1 を指定すると、自動的に DBS1.NET が指定されます。
insert_rule_name	システム生成の INSERT ルール名が含まれます。このルールは、挿入 LCR および（挿入 LCR に変換する必要がある）更新 LCR を処理します。
update_rule_name	システム生成の UPDATE ルール名が含まれます。このルールは、（更新 LCR のままで変換しない）更新 LCR を処理します。
delete_rule_name	システム生成の DELETE ルール名が含まれます。このルールは、削除 LCR および（削除 LCR に変換する必要がある）更新 LCR を処理します。

ADD_TABLE_PROPAGATION_RULES プロシージャ

ソース・キューの指定表に関係する LCR を宛先キューに伝播する伝播ルールを追加します。また、このプロシージャは、必要に応じて現行のユーザーを使用して伝播を構成し、デフォルトの伝播スケジュールを確立します。このプロシージャを使用すると、フィルタ条件に従って指定表の LCR を伝播できます。ソース・キューと宛先キューの間で許可される伝播ジョブは 1 つのみです。

伝播ルールを追加すると、伝播ジョブによって、指定の表に関連する DML または DDL の変更（あるいはその両方）が指定のソース・キューから指定の宛先キューに伝播されます。このプロシージャは、include_dml パラメータ値と include_ddl パラメータ値に基づいて、DML と DDL のルールをそれぞれ自動的に作成します。システム生成のルールには、表名とそれに付加された順序番号で構成された名前が付きます。この順序番号によって、名前の競合が回避されます。表名と順序番号を組み合せた名前が長すぎる場合は、表名が切り捨てられます。オーバーロードされた ADD_TABLE_PROPAGATION_RULES プロシージャの場合は、DML と DDL の変更に対するシステム生成のルール名が戻ります。

伝播ジョブは、作成されたルールをフィルタ処理に使用します。伝播ジョブにルール・セットがない場合は、ルール・セットが自動的に作成され、表に変更を伝播するルールがルール・セットに追加されます。伝播ジョブの既存ルール・セットにある他のルールは、この影響を受けません。DBMS_RULE_ADM パッケージを使用すると、別のルールを追加できます。

次に、表のルール条件の例を示します。この条件は、伝播ジョブによる DML 変更の伝播用に作成できます。

```
:dml.get_object_owner() = 'HR' AND :dml.get_object_name() = 'LOCATIONS'
AND :dml.is_null_tag() = 'Y' AND :dml.get_source_database_name() = 'DBS1.NET'
```

注意： 前述の例の引用符は、すべて一重引用符です。

伝播が適切に動作するために、ソース・キューの所有者には伝播イベントに必要な権限を付与してください。

注意：

- 現在は、データベース・リンクによってイベントが複数の宛先キューに伝播される場合でも、単一の伝播ジョブによって、特定のデータベース・リンクを使用するイベントすべてが伝播されます。
- ソース・キューの所有者は伝播を実行しますが、伝播ジョブの所有者はジョブを作成したユーザーです。これらのユーザーは、同一のユーザーまたは別のユーザーの場合があります。

関連項目： 必要な権限は、47-4 ページの「CREATE_PROPAGATION プロシージャ」を参照してください。

構文

```
DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(
   table_name                IN  VARCHAR2,
   streams_name              IN  VARCHAR2,
   source_queue_name         IN  VARCHAR2,
   destination_queue_name    IN  VARCHAR2,
   include_dml               IN  BOOLEAN  DEFAULT true,
   include_ddl               IN  BOOLEAN  DEFAULT false,
   include_tagged_lcr        IN  BOOLEAN  DEFAULT false,
   source_database           IN  VARCHAR2 DEFAULT NULL,
   dml_rule_name             OUT VARCHAR2,
   ddl_rule_name             OUT VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。このプロシージャの一方のバージョンには、2 つの OUT パラメータがあり、他方のバージョンにはこれらのパラメータはありません。

パラメータ

表 73-7 ADD_TABLE_PROPAGATION_RULES プロシージャのパラメータ

パラメータ	説明
table_name	表の名前。[*schema_name.*]*object_name* の形式で指定します。たとえば、hr.employees のようにします。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
streams_name	伝播ジョブの名前。 指定の存在しない場合は、自動的に作成されます。 この指定が NULL で、同じソース・キューと宛先キュー（データベース・リンクを含む）を指定した伝播ジョブがすでにある場合は、その伝播ジョブが使用されます。 この指定が NULL で、同じソース・キューと宛先キュー（データベース・リンクを含む）を指定した伝播ジョブがない場合は、システム生成の名前で伝播ジョブが自動的に作成されます。
source_queue_name	ソース・キューの名前。現行のデータベースにはソース・キューが含まれている必要があります。

表 73-7 ADD_TABLE_PROPAGATION_RULES プロシージャのパラメータ（続き）

パラメータ	説明
destination_queue_name	宛先キューの名前（STREAMS_QUEUE@DBS2 などのデータベース・リンクを含む）。 データベース・リンクが省略されている場合は、現行のデータベースのグローバル名が使用されます。この場合、ソース・キューと宛先キューは同じデータベース内にあることが必要です。 **注意：**接続修飾子は使用できません。
include_dml	TRUE の場合は、DML 変更のためのルールが作成されます。FALSE の場合、DML ルールは作成されません。NULL は許可されません。
include_ddl	TRUE の場合は、DDL 変更のためのルールが作成されます。FALSE の場合、DDL ルールは作成されません。NULL は許可されません。
include_tagged_lcr	TRUE の場合は、LCR に NULL 以外のタグがあるかどうかに関係なく、LCR が常に伝播の対象として考慮されます。この設定は、データベースの完全な（スタンバイなど）コピーに適しています。 FALSE の場合は、LCR に NULL のタグが含まれている場合にのみ、伝播に LCR が考慮されます。FALSE の設定は、通常、どこでも更新可能な構成で、ソース・データベースに変更が戻ることを回避するために、指定されます。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。
source_database	ソース・データベースのグローバル名。ソース・データベースは、変更が発生したデータベースです。NULL の場合、ソース・データベースに関係する条件は、生成されるルールに追加されません。 ドメイン名が未指定の場合は、データベース名に自動的に追加されます。たとえば、ドメイン名が .NET の場合に DBS1 を指定すると、自動的に DBS1.NET が指定されます。 オラクル社では、伝播ルールにソース・データベースを指定することをお薦めします。
dml_rule_name	include_dml が TRUE の場合は、DML ルール名が含まれます。 include_dml が FALSE の場合は、NULL が含まれます。

表 73-7 ADD_TABLE_PROPAGATION_RULES プロシージャのパラメータ（続き）

パラメータ	説明
ddl_rule_name	include_ddl が TRUE の場合は、DDL ルール名が含まれます。 include_ddl が FALSE の場合は、NULL が含まれます。

ADD_TABLE_RULES プロシージャ

表の取得ルールまたは適用ルールを追加します。

取得ルールを追加すると、取得プロセスによって、DML または DDL の変更（あるいはその両方）が指定した表について取得され、指定キューにエンキューされます。取得ルールの場合は、このプロシージャをソース・データベースで実行してください。このプロシージャは、DBMS_CAPTURE_ADM パッケージの PREPARE_TABLE_INSTANTIATION プロシージャを指定の表に対して自動的に起動します。

適用ルールを追加すると、適用プロセスは取得イベントを受け取って適用します。このイベントには、指定した表の DML または DDL の変更（あるいはその両方）が含まれています。適用ルールの場合は、このプロシージャを接続先データベースで実行してください。

このプロシージャによって作成された適用プロセスの適用先は、ローカル・データベースでのイベントと取得イベントのみです。リモート・データベースでのイベントまたはユーザーがエンキューしたイベントに対する適用プロセスを作成するには、DBMS_APPLY_ADM パッケージの CREATE_APPLY プロシージャを使用します。

このプロシージャで作成した適用プロセスが適用した変更によって、'00'（2 桁のゼロ）の値のタグが接続先データベースの REDO ログに生成されます。DBMS_APPLY_ADM パッケージの ALTER_APPLY プロシージャを使用すると、適用プロセスの作成後に必要に応じてタグの値を変更できます。

適用プロセスは、DBMS_APPLY_ADM.CREATE_APPLY プロシージャを使用し、その実行時に、デフォルト以外の値を apply_captured、apply_database_link および apply_tag の各パラメータに指定して作成することもできます。その後、この ADD_TABLE_RULES プロシージャを使用して、その適用プロセスで使用するルール・セットにルールを追加できます。

このプロシージャは、include_dml パラメータ値と include_ddl パラメータ値に基づいて、DML と DDL のルールをそれぞれ自動的に作成します。システム生成のルールには、表名とそれに付加された順序番号で構成された名前が付きます。この順序番号によって、名前の競合が回避されます。表名と順序番号を組み合せた名前が長すぎる場合は、表名が切り捨てられます。

次に、ルール条件の例を示します。この条件は、DML 文のフィルタ処理用に作成できます。

```
:dml.get_object_owner() = 'HR' and :dml.get_object_name() = 'EMPLOYEES'
AND :dml.is_null_tag() = 'Y' AND :dml.get_source_database_name() = 'DBS1.NET'
```

注意： 前述の例の引用符は、すべて一重引用符です。

オーバーロードされた ADD_TABLE_RULES プロシージャの場合は、DML と DDL の変更に対するシステム生成のルール名が戻ります。

取得プロセスまたは適用プロセスは、作成されたルールをフィルタ処理に使用します。プロセスにルール・セットがない場合は、ルール・セットが自動的に作成され、表に対するルールがそのルール・セットに追加されます。そのプロセスの既存ルール・セットにある他のルールは、この影響を受けません。DBMS_RULE_ADM パッケージを使用すると、別のルールを追加できます。

このプロシージャによって取得プロセスまたは適用プロセスを作成する場合、このプロシージャを実行するユーザーは、変更を取得または適用するユーザーとなります。指定されたユーザーには、これらのアクションの実行に必要な権限を付与してください。

関連項目：

- 第 64 章「DBMS_RULE_ADM」
- 変更の取得に必要な権限は、8-6 ページの「CREATE_CAPTURE プロシージャ」を参照してください。
- 変更の適用に必要な権限は、4-9 ページの「CREATE_APPLY プロシージャ」を参照してください（apply_user パラメータを参照）。

構文

```
DBMS_STREAMS_ADM.ADD_TABLE_RULES(
   table_name                IN  VARCHAR2,
   streams_type              IN  VARCHAR2,
   streams_name              IN  VARCHAR2 DEFAULT NULL,
   queue_name                IN  VARCHAR2 DEFAULT 'streams_queue',
   include_dml               IN  BOOLEAN  DEFAULT true,
   include_ddl               IN  BOOLEAN  DEFAULT false,
   include_tagged_lcr        IN  BOOLEAN  DEFAULT false,
   source_database           IN  VARCHAR2 DEFAULT NULL,
   dml_rule_name             OUT VARCHAR2,
   ddl_rule_name             OUT VARCHAR2);
```

注意： このプロシージャはオーバーロードされています。このプロシージャの一方のバージョンには、2 つの OUT パラメータがあり、他方のバージョンにはこれらのパラメータはありません。

パラメータ

表 73-8 ADD_TABLE_RULES プロシージャのパラメータ

パラメータ	説明
table_name	表の名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr.employees のようにします。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 Streams では表の存在を検証しないため、まだ存在していない表を指定できます。
streams_type	プロセスのタイプ。capture または apply です。
streams_name	プロセス名。 指定のプロセスが存在しない場合は、自動的に作成されます。 この指定が NULL で、キューに関連する取得プロセスまたは適用プロセスが存在している場合は、関連する取得プロセスまたは適用プロセスが使用されます。キューに関連するプロセスがない場合は、システム生成の名前を使用して取得プロセスまたは適用プロセスが自動的に作成されます。この指定が NULL で、キューに対して指定した streams_type のプロセスが複数存在する場合は、エラーが発生します。
queue_name	ローカル・キューの名前。取得ルールの場合は、変更がエンキューされるキューを指定します。適用ルールの場合は、変更がデキューされるキューを指定します。
include_dml	TRUE の場合は、DML 変更のための DML ルールが作成されます。FALSE の場合、DML ルールは作成されません。NULL は許可されません。
include_ddl	TRUE の場合は、DDL 変更のための DDL ルールが作成されます。FALSE の場合、DDL ルールは作成されません。NULL は許可されません。

表 73-8 ADD_TABLE_RULES プロシージャのパラメータ（続き）

パラメータ	説明
include_tagged_lcr	TRUE の場合、REDO エントリまたは LCR に NULL でないタグがあるかどうかに関係なく、取得では REDO エントリが、適用では LCR が常に処理の対象として考慮されます。この設定は、データベースの完全な（スタンバイなど）コピーに適しています。 FALSE の場合は、REDO エントリまたは LCR に NULL のタグが含まれている場合にのみ、取得については REDO エントリ、適用については LCR が考慮されます。FALSE の設定は、通常、どこでも更新可能な構成でソース・データベースに変更が戻ることを回避するために、指定されます。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。
source_database	ソース・データベースのグローバル名。NULL の場合、ソース・データベースに関係する条件は、生成されるルールに追加されません。 取得ルールの場合、現在、取得データベースとソース・データベースは必ず同一であるため、NULL を指定できます。 適用ルールの場合は、適用プロセスによって適用される変更のソース・データベースを指定します。ソース・データベースは、変更が発生したデータベースです。取得イベントを適用する場合、適用プロセスによって適用できるのは、1 つのソース・データベースでの 1 つの取得プロセスによるイベントのみです。 ドメイン名が未指定の場合は、データベース名に自動的に追加されます。たとえば、ドメイン名が .NET の場合に DBS1 を指定すると、自動的に DBS1.NET が指定されます。
dml_rule_name	include_dml が TRUE の場合は、DML ルール名が含まれます。 include_dml が FALSE の場合は、NULL が含まれます。
ddl_rule_name	include_ddl が TRUE の場合は、DDL ルール名が含まれます。 include_ddl が FALSE の場合は、NULL が含まれます。

PURGE_SOURCE_CATALOG プロシージャ

ローカル・データベースにある、指定オブジェクトに関する全 Streams のデータ・ディクショナリ情報を削除します。このプロシージャを使用すると、現在および今後も不要な Streams のメタデータを削除できます。

オブジェクトが含まれているソース・データベースのグローバル名を source_database パラメータに指定する必要があります。現行のデータベースがオブジェクトのソース・データベースでない場合、そのオブジェクトに関するデータ・ディクショナリ情報は、ソース・データベースではなく現行のデータベースで削除されます。

たとえば、ソース・データベース dbs1.net の hr.employees 表への変更が、接続先データベース dbs2.net の hr.employees 表に適用されると仮定します。また、dbs2.net の hr.employees は、ソース・データベースではないとします。この場合、この表に dbs2.net を source_database として指定すると、エラーが発生します。ただし、dbs2.net データベースでの PURGE_SOURCE_CATALOG プロシージャの実行時に、この表に dbs1.net を source_database として指定すると、dbs2.net にある表に関するデータ・ディクショナリ情報が削除されます。

次の条件のいずれかに当てはまる場合は、データベースでこのプロシージャを実行しないでください。

- オブジェクトの取得プロセスで取得した LCR が、オブジェクトの再インスタンス化を行わずにローカルで適用される場合（またはその可能性がある場合）。
- オブジェクトの取得プロセスで取得した LCR が、オブジェクトのインスタンス化を行わずにデータベースによって転送される場合（またはその可能性がある場合）。

> **注意：** これらの条件は、取得プロセスで作成されていない LCR には適用されません。つまり、これらの条件は、ユーザーが作成した LCR には適用されません。

構文

```
DBMS_STREAMS_ADM.PURGE_SOURCE_CATALOG(
   source_database        IN VARCHAR2,
   source_object_name     IN VARCHAR2,
   source_object_type     IN VARCHAR2);
```

パラメータ

表 73-9 PURGE_SOURCE_CATALOG プロシージャのパラメータ

パラメータ	説明
source_database	オブジェクトが含まれるソース・データベースのグローバル名。 ドメイン名が未指定の場合は、データベース名に自動的に追加されます。たとえば、ドメイン名が .NET の場合に DBS1 を指定すると、自動的に DBS1.NET が指定されます。
source_object_name	オブジェクトの名前。[*schema_name*.]*object_name* の形式で指定します。たとえば、hr.employees のようにします。スキーマが未指定の場合は、現行のユーザーがデフォルトです。
source_object_type	オブジェクトのタイプ。現在、使用できるオブジェクトのタイプは、TABLE のみです。

REMOVE_RULE プロシージャ

指定のルールまたはすべてのルールを、指定の取得プロセス、適用プロセスまたは伝播ジョブに関連付けられているルール・セットから削除します。

注意： ルールがシステムで自動的に作成されている場合は、DBMS_RULE_ADM.REMOVE_RULE プロシージャではなく、このプロシージャを使用してルールを削除してください。DBMS_RULE_ADM.REMOVE_RULE プロシージャを使用すると、ルールに関するメタデータの一部が残存する可能性があります。

構文

```
DBMS_STREAMS_ADM.REMOVE_RULE(
   rule_name             IN VARCHAR2,
   streams_type          IN VARCHAR2,
   streams_name          IN VARCHAR2,
   drop_unused_rule      IN BOOLEAN DEFAULT true);
```

パラメータ

表 73-10 REMOVE_RULE プロシージャのパラメータ

パラメータ	説明
rule_name	削除するルールの名前。NULL の場合は、指定の取得プロセス、適用プロセスまたは伝播ジョブのルール・セットに関するすべてのルールが削除されます。
streams_type	Streams のルールのタイプ。capture、apply または propagation です。
streams_name	取得プロセス、適用プロセスまたは伝播ジョブの名前。
drop_unused_rule	FALSE の場合、ルールはデータベースから削除されません。 TRUE を指定し、ルールがいずれのルール・セットにも属していない場合は、そのルールがデータベースから削除されます。 TRUE を指定し、ルールがいずれかのルール・セットに属している場合、そのルールはデータベースから削除されません。

SET_UP_QUEUE プロシージャ

Streams の取得、伝播および適用機能で使用するキュー表と Streams のキューを作成します。

設定には、次のアクションが含まれます。

- 指定のキュー表が存在していない場合、このプロシージャは、DBMS_AQADM パッケージの CREATE_QUEUE_TABLE プロシージャを実行し、指定の STORAGE 句でキュー表を作成します。
- 指定したキューの名前が存在していない場合、このプロシージャは、DBMS_AQADM パッケージの CREATE_QUEUE プロシージャを実行して、キューを作成します。
- このプロシージャは、キューを開始します。
- キュー・ユーザーが指定されている場合、このプロシージャは、このユーザーをキューの保護キュー・ユーザーとして構成し、キューに対する ENQUEUE 権限と DEQUEUE 権限を指定のキュー・ユーザーに付与します。

  キュー・ユーザーを保護キュー・ユーザーとして構成するために、このプロシージャは、ユーザー名と同名のアドバンスト・キューイング・エージェントを作成します(エージェントが存在していない場合)。この名前のエージェントがすでに存在し、キュー・ユーザーのみに関連付けられている場合は、そのエージェントが使用されます。SET_UP_QUEUE プロシージャは、次に、DBMS_AQADM パッケージの ENABLE_DB_ACCESS プロシージャを実行して、エージェントとユーザーを指定します。

このプロシージャによって、保護キューでありトランザクション・キューでもある SYS.AnyData キューが作成されます。

注意：

- イベントのキューへのエンキューまたはキューからのデキューでは、キュー・ユーザーに DBMS_AQ パッケージに関する EXECUTE 権限が必要です。この権限は、SET_UP_QUEUE プロシージャでは付与されません。
- SET_UP_QUEUE で作成を試みたエージェントがすでに存在し、そのエージェントが queue_user で指定されているユーザー以外のユーザーに関連付けられている場合は、エラーが発生します。この場合は、既存のエージェントを変更または削除してから、SET_UP_QUEUE を再試行してください。

関連項目： これらの権限の詳細は、DBMS_AQADM パッケージに関する章の GRANT_QUEUE_PRIVILEGE プロシージャを参照してください。

構文

```
DBMS_STREAMS_ADM.SET_UP_QUEUE(
   queue_table          IN VARCHAR2 DEFAULT 'streams_queue_table',
   storage_clause       IN VARCHAR2 DEFAULT NULL,
   queue_name           IN VARCHAR2 DEFAULT 'streams_queue',
   queue_user           IN VARCHAR2 DEFAULT NULL,
   comment              IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 73-11 SET_UP_QUEUE プロシージャのパラメータ

パラメータ	説明
queue_table	キュー表の名前。[*schema_name*.]*queue_table_name* の形式で指定します。たとえば、strmadmin.streams_queue_table のようにします。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 キュー表の所有者が未指定の場合は、このプロシージャを実行するユーザーがキュー表の所有者として自動的に指定されます。
storage_clause	キュー表に対す STORAGE 句。 記憶領域パラメータは、キュー表の作成時に、CREATE TABLE 文に組み込まれます。有効な表の STORAGE 句を任意に指定できます。 ここで表領域が指定されない場合は、キュー表とそのすべての関連オブジェクトが、このプロシージャを実行しているユーザーのデフォルトのユーザー表領域に作成されます。ここで表領域が指定されると、キュー表とそのすべての関連オブジェクトが、STORAGE 句で指定された表領域に作成されます。 NULL の場合は、キュー表が作成された表領域の記憶特性が使用されます。 **関連項目：**STORAGE 句の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
queue_name	Streams queue として機能するキューの名前。[*schema_name*.]*queue_name* の形式で指定します。たとえば、strmadmin.streams_queue のようにします。スキーマが未指定の場合は、現行のユーザーがデフォルトです。 キューの所有者が未指定の場合は、キュー表の所有者がデフォルトで設定されます。キュー表の所有者は、キューの所有者であることも必要です。キューの所有者には、キューに関するすべてのキュー操作を実行するための権限が自動的に付与されます。

表 73-11 SET_UP_QUEUE プロシージャのパラメータ（続き）

パラメータ	説明
queue_user	キューに対する ENQUEUE 権限と DEQUEUE 権限が必要なユーザー名。このユーザーは、キューの保護キュー・ユーザーとしても構成されます。キュー・ユーザーには GRANT オプションが付与されていないため、これらの権限を他のユーザーに付与することはできません。 NULL の場合、権限は付与されません。DBMS_AQADM パッケージを使用して、キューの権限を適切なユーザーに付与することもできます。
comment	キューに対するコメント。

74

DBMS_TRACE

Oracle9*i* PL/SQL は、サーバー上での PL/SQL プログラムの実行をトレースするために、API を提供します。DBMS_TRACE パッケージとしてサーバー上に実装されたトレース API を使用し、PL/SQL ファンクション、プロシージャおよび例外をトレースできます。

DBMS_TRACE は、セッションで PL/SQL トレースを開始および停止するサブプログラムを提供します。トレース・データはプログラムの実行時に収集され、データベース表に書き出されます。

一般的なセッションには、次の処理が含まれます。

- セッションで PL/SQL トレースを開始します（DBMS_TRACE.SET_PLSQL_TRACE）。
- トレースするアプリケーションを実行します。
- セッションでの PL/SQL トレースを停止します（DBMS_TRACE.CLEAR_PLSQL_TRACE）。

この章では、次の項目について説明します。

DBMS_TRACE の要件、制限事項および定数

要件

このパッケージは、SYS の下で作成する必要があります。

制限事項

PL/SQL トレースは、共有サーバー環境では使用できません。

定数

DBMS_TRACE では、次の定数を使用します。

```
trace_all_calls          constant INTEGER := 1;
trace_enabled_calls      constant INTEGER := 2;
trace_all_exceptions     constant INTEGER := 4;
trace_enabled_exceptions constant INTEGER := 8;
trace_all_sql            constant INTEGER := 32;
trace_enabled_sql        constant INTEGER := 64;
trace_all_lines          constant INTEGER := 128;
trace_enabled_lines      constant INTEGER := 256;
trace_stop               constant INTEGER := 16384;
trace_pause              constant INTEGER := 4096;
trace_resume             constant INTEGER := 8192;
trace_limit              constant INTEGER := 16;
trace_major_version      constant BINARY_INTEGER := 1;
trace_minor_version      constant BINARY_INTEGER := 0;
```

これらの定数についてはすべて記号形式（定数名）を使用することをお薦めします。

DBMS_TRACE の使用方法

データ量の制御

大規模なアプリケーションをプロファイルすると、データ量が膨大になる可能性があります。トレース・データの収集に関する特定のプログラム・ユニットを使用可能にして、収集するデータ量を制御できます。

プログラム・ユニットは、コンパイルとデバッグを行って使用可能にできます。次のいずれかの方法で行います。

```
alter session set plsql_debug=true;
create or replace ... /* create the library units - debug information will be
generated */
```

または

```
/* recompile specific library unit with debug option */
alter [PROCEDURE | FUNCTION | PACKAGE BODY]  <libunit-name> compile debug;
```

注意： 2 番目の方法は、無名ブロックに対しては使用できません。

SET_PLSQL_TRACE プロシージャの TRACE_LEVEL パラメータに TRACE_LIMIT を指定することにより、最新の 8,192 レコード（概算）のみを保持して、データベースで使用する記憶域量を制限できます。

DBMS_TRACE 出力を収集するデータベース表の作成

DBMS_TRACE パッケージで出力を書き込むためのデータベース表を作成する必要があります。作成しないと、データが収集されません。このような表を作成するには、TRACETAB.SQL スクリプトを実行してください。このスクリプトで作成した表の所有者は、SYS です。

トレース・データの収集

トレース可能な PL/SQL 機能は、DBMSPBT.SQL スクリプトに記述されています。主なトレース機能は次のとおりです。

- コールのトレース
- 例外のトレース
- SQL のトレース
- 行のトレース

DBMS_TRACE の追加機能として、トレースの解析と再開および出力の制限もできます。

コールのトレース

使用可能なコールのトレースには、次の 2 つのレベルがあります。

- レベル 1: すべてのコールをトレースします。これは、定数 trace_all_calls に対応します。
- レベル 2: 使用可能なプログラム・ユニットのみ、コールをトレースします。これは、定数 trace_enabled_calls に対応します。

リモート・プロシージャ・コール（RPC）については、使用可能かどうかを検出できないため、RPC ではレベル 1 でのみトレースできます。

例外のトレース

使用可能な例外のトレースには、次の 2 つのレベルがあります。

- レベル 1: すべての例外をトレースします。これは、定数 trace_all_exceptions に対応します。
- レベル 2: 使用可能なプログラム・ユニットのみ、発生した例外をトレースします。これは、定数 trace_enabled_exceptions に対応します。

SQL のトレース

使用可能な SQL のトレースには、次の 2 つのレベルがあります。

- レベル 1: すべての SQL をトレースします。これは、定数 trace_all_sql に対応します。
- レベル 2: 使用可能なプログラム・ユニットでのみ、SQL をトレースします。これは、定数 trace_enabled_sql に対応します。

行のトレース

使用可能な行のトレースには、次の 2 つのレベルがあります。

- レベル 1: すべての行をトレースします。これは、定数 trace_all_lines に対応します。
- レベル 2: 使用可能なプログラム・ユニットでのみ、行をトレースします。これは、定数 trace_enabled_lines に対応します。

行のトレース時には、行番号が変わるたびにレコードがデータベースに追加されます。プロシージャのコールおよびその戻り値によって行番号が変わる場合も、トレースの対象に含まれます。

注意： すべてのタイプのトレースでは、レベル 1 がレベル 2 を上書きします。たとえば、レベル 1 とレベル 2 の両方が使用可能な場合は、レベル 1 が優先されます。

収集されたデータ

使用可能なプログラム・ユニットについてのみトレースが要求されていて、現行のプログラム・ユニットが使用可能ではない場合、トレース・データは書き込まれません。

コールをトレースする場合は、コールと戻り値の両方がトレースされます。トレースが使用可能であるかどうかのチェックは、コールされるルーチンまたはコールするルーチンのいずれか一方が使用可能な場合に、トレース実施と判断されます。

コールのトレースでは、プログラム・ユニットのタイプ、名前および行番号が常に出力されます。さらに、コール元のスタックの深さが書き込まれます。コール元のユニットが使用可能な場合は、コール元プロシージャ名も出力されます。コールされる側のユニットが使用可能な場合は、コールされるプロシージャ名が出力されます。

例外のトレースでは、行番号が書き込まれます。例外が発生すると、その例外がユーザー定義か事前定義のいずれであるかが示されます。事前定義の例外の場合は例外番号も出力されます。例外が発生した場所とそのハンドラの両方がトレースされます。トレースが使用可能であるかどうかのチェックは、例外が発生した場所と例外がハンドルされた場所で個別に行われます。

DBMS_TRACE.SET_PLSQL_TRACE および DBMS_TRACE.CLEAR_PLSQL_TRACE へのコールはすべて、データベースの特別なトレース・レコードに配置されます。したがって、トレース設定が変更された時点を常に確認できます。

トレース管理

収集した項目を確認するのみではなく、トレース処理を一時停止したり、再開することもできます。トレースが一時停止されてから再開されるまでの間、情報は収集されません。一時停止と再開には、定数 TRACE_PAUSE と TRACE_RESUME を使用します。トレースが一時停止または再開したことを示すためにトレース・レコードが生成されます。

定数 TRACE_LIMIT を使用すると、最新の 8,192 トレース・イベントのみを保持できます。これにより、データベースをいっぱいにすることなくトレースを繰り返すことができます。トレースの停止時には、最新の 8,192 レコードが保存されています。各トレース・レコードごとにはチェックされないため、この制限は概算です。少なくとも要求されたトレース・レコードの数は生成され、1,000 レコードまでは追加生成できます。

DBMS_TRACE サブプログラムの要約

表 74-1 DBMS_TRACE のサブプログラム

サブプログラム	説明
「SET_PLSQL_TRACE プロシージャ」74-6 ページ	現行のセッションでトレースを開始します。
「CLEAR_PLSQL_TRACE プロシージャ」74-7 ページ	セッションでのトレース・データのダンプを停止します。
「PLSQL_TRACE_VERSION プロシージャ」74-7 ページ	トレース・パッケージのバージョン番号を取得します。

SET_PLSQL_TRACE プロシージャ

このプロシージャは、PL/SQL のトレース・データ収集を可能にします。

構文

```
DBMS_TRACE.SET_PLSQL_TRACE (
   trace_level INTEGER);
```

パラメータ

表 74-2 SET_PLSQL_TRACE プロシージャのパラメータ

パラメータ	説明
trace_level	74-2 ページにリストされている 1 つ以上の定数を指定する必要があります。定数を合計することにより、複数の PL/SQL 言語機能のトレースを同時に使用可能にできます。制御定数 trace_pause、trace_resume および trace_stop は、他の定数と組み合せて使用することはできません。 詳細は、74-3 ページの「トレース・データの収集」を参照してください。

CLEAR_PLSQL_TRACE プロシージャ

このプロシージャは、トレース・データ収集を使用禁止にします。

構文

```
DBMS_TRACE.CLEAR_PLSQL_TRACE;
```

PLSQL_TRACE_VERSION プロシージャ

このプロシージャは、トレース・パッケージのバージョン番号を取得します。DBMS_TRACE パッケージのバージョン番号とリリース番号を戻します。

構文

```
DBMS_TRACE.PLSQL_TRACE_VERSION (
   major OUT BINARY_INTEGER,
   minor OUT BINARY_INTEGER);
```

パラメータ

表 74-3 PLSQL_TRACE_VERSION プロシージャのパラメータ

パラメータ	説明
major	DBMS_TRACE のバージョン番号
minor	DBMS_TRACE のリリース番号

75

DBMS_TRANSACTION

このパッケージは、ストアド・プロシージャから SQL トランザクション文へのアクセスを提供します。

関連項目：『Oracle9*i* SQL リファレンス』

この章では、次の項目について説明します。

要件

このパッケージは、パッケージ所有者 SYS ではなく、コール・ユーザーの権限で実行されます。

DBMS_TRANSACTION サブプログラムの要約

表 75-1 DBMS_TRANSACTION サブプログラム

サブプログラム
「READ_ONLY プロシージャ」75-3 ページ
「READ_WRITE プロシージャ」75-3 ページ
「ADVISE_ROLLBACK プロシージャ」75-3 ページ
「ADVISE_NOTHING プロシージャ」75-3 ページ
「ADVISE_COMMIT プロシージャ」75-4 ページ
「USE_ROLLBACK_SEGMENT プロシージャ」75-4 ページ
「COMMIT_COMMENT プロシージャ」75-4 ページ
「COMMIT_FORCE プロシージャ」75-5 ページ
「COMMIT プロシージャ」75-5 ページ
「SAVEPOINT プロシージャ」75-6 ページ
「ROLLBACK プロシージャ」75-6 ページ
「ROLLBACK_SAVEPOINT プロシージャ」75-7 ページ
「ROLLBACK_FORCE プロシージャ」75-7 ページ
「BEGIN_DISCRETE_TRANSACTION プロシージャ」75-8 ページ
「PURGE_MIXED プロシージャ」75-8 ページ
「PURGE_LOST_DB_ENTRY プロシージャ」75-9 ページ
「LOCAL_TRANSACTION_ID ファンクション」75-11 ページ
「STEP_ID ファンクション」75-11 ページ

READ_ONLY プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
SET TRANSACTION READ ONLY
```

構文

```
DBMS_TRANSACTION.READ_ONLY;
```

READ_WRITE プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
SET TRANSACTION READ WRITE
```

構文

```
DBMS_TRANSACTION.READ_WRITE;
```

ADVISE_ROLLBACK プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION ADVISE ROLLBACK
```

構文

```
DBMS_TRANSACTION.ADVISE_ROLLBACK;
```

ADVISE_NOTHING プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION ADVISE NOTHING
```

構文

```
DBMS_TRANSACTION.ADVISE_NOTHING;
```

ADVISE_COMMIT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ALTER SESSION ADVISE COMMIT
```

構文

```
DBMS_TRANSACTION.ADVISE_COMMIT;
```

USE_ROLLBACK_SEGMENT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
SET TRANSACTION USE ROLLBACK SEGMENT <rb_seg_name>
```

構文

```
DBMS_TRANSACTION.USE_ROLLBACK_SEGMENT (
   rb_name VARCHAR2);
```

パラメータ

表 75-2 USE_ROLLBACK_SEGMENT プロシージャのパラメータ

パラメータ	説明
rb_name	使用するロールバック・セグメントの名前。

COMMIT_COMMENT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
COMMIT COMMENT <text>
```

構文

```
DBMS_TRANSACTION.COMMIT_COMMENT (
   cmnt VARCHAR2);
```

パラメータ

表 75-3 COMMIT_COMMENT プロシージャのパラメータ

パラメータ	説明
cmnt	このコミットに関連するコメント。

COMMIT_FORCE プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
COMMIT FORCE <text>, <number>"
```

構文

```
DBMS_TRANSACTION.COMMIT_FORCE (
   xid VARCHAR2,
   scn VARCHAR2 DEFAULT NULL);
```

パラメータ

表 75-4 COMMIT_FORCE プロシージャのパラメータ

パラメータ	説明
xid	ローカルまたはグローバルなトランザクション ID。
scn	システム変更番号。

COMMIT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
COMMIT
```

確認のためにここで記述します。このプロシージャは、PL/SQL の一部としてすでに実装されています。

構文

```
DBMS_TRANSACTION.COMMIT;
```

SAVEPOINT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
SAVEPOINT <savepoint_name>
```

確認のためにここで記述します。このプロシージャは、PL/SQL の一部としてすでに実装されています。

構文

```
DBMS_TRANSACTION.SAVEPOINT (
   savept VARCHAR2);
```

パラメータ

表 75-5 SAVEPOINT プロシージャのパラメータ

パラメータ	説明
savept	セーブポイントの識別子。

ROLLBACK プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ROLLBACK
```

確認のためにここで記述します。このプロシージャは、PL/SQL の一部としてすでに実装されています。

構文

```
DBMS_TRANSACTION.ROLLBACK;
```

ROLLBACK_SAVEPOINT プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ROLLBACK TO SAVEPOINT <savepoint_name>
```

確認のためにここで記述します。このプロシージャは、PL/SQL の一部としてすでに実装されています。

構文

```
DBMS_TRANSACTION.ROLLBACK_SAVEPOINT (
   savept VARCHAR2);
```

パラメータ

表 75-6 ROLLBACK_SAVEPOINT プロシージャのパラメータ

パラメータ	説明
savept	セーブポイントの識別子。

ROLLBACK_FORCE プロシージャ

このプロシージャは、次の SQL 文と同じです。

```
ROLLBACK FORCE <text>
```

構文

```
DBMS_TRANSACTION.ROLLBACK_FORCE (
   xid VARCHAR2);
```

パラメータ

表 75-7 ROLLBACK_FORCE プロシージャのパラメータ

パラメータ	説明
xid	ローカルまたはグローバルなトランザクション ID。

BEGIN_DISCRETE_TRANSACTION プロシージャ

このプロシージャは、このトランザクションについてディスクリート・トランザクション・モードを設定します。

構文

```
DBMS_TRANSACTION.BEGIN_DISCRETE_TRANSACTION;
```

例外

表 75-8 BEGIN_DISCRETE_TRANSACTION プロシージャの例外

例外	説明
ORA-08175	トランザクションが、ディスクリート・トランザクションとして実行できない操作を行おうとしました。 この例外が発生した場合は、ロールバックしてトランザクションを再試行します。
ORA-08176	トランザクションにおいて、索引の作成、ダイレクト・ロードまたはディスクリート・トランザクションの実行など、ロールバック・データを生成しない操作によりデータが変更されました。 この例外が発生した場合は、例外を受け取った操作を再試行します。

例

```
DISCRETE_TRANSACTION_FAILED exception;
    pragma exception_init(DISCRETE_TRANSACTION_FAILED, -8175);
CONSISTENT_READ_FAILURE exception;
    pragma exception_init(CONSISTENT_READ_FAILURE, -8176);
```

PURGE_MIXED プロシージャ

インダウト・トランザクションにコミットまたはロールバックを強制実行すると（自動リカバリで結果を解決するのではなく）、トランザクションで混合した結果が得られる可能性があります。一部のサイトはコミットして、それ以外はロールバックします。このような矛盾は Oracle で自動的に解決できませんが、Oracle は、MIXED 列を値 'yes' に設定して、DBA_2PC_PENDING にあるエントリにフラグ付けを行います。

Oracle では、混合出力トランザクションの情報が自動的に削除されることはありません。混合トランザクションの結果として発生したすべての不整合が解決されたことを、アプリケーションまたは DBA で確認した後、このプロシージャを使用して、指定した混合出力トランザクションの情報を削除できます。

構文

```
DBMS_TRANSACTION.PURGE_MIXED (
   xid VARCHAR2);
```

パラメータ

表 75-9 PURGE_MIXED プロシージャのパラメータ

パラメータ	説明
xid	DBA_2PC_PENDING 表にある LOCAL_TRAN_ID 列の値に設定する必要があります。

PURGE_LOST_DB_ENTRY プロシージャ

コミット処理中に障害が発生すると、自動リカバリ機能によって、そのトランザクションに関連しているすべてのサイトでの結果が一貫して解決されます。ただし、リカバリの完了前にリモート・データベースが破損または再作成されると、DBA_2PC_PENDING 内のリカバリを制御するために使用するエントリと、それに関連する表が削除されずに、リカバリ処理が定期的に再試行されます。プロシージャ PURGE_LOST_DB_ENTRY では、このようなトランザクションをローカル・サイトから削除できます。

構文

```
DBMS_TRANSACTION.PURGE_LOST_DB_ENTRY (
   xid VARCHAR2);
```

警告： PURGE_LOST_DB_ENTRY は、他のデータベースが失われたり再作成された場合にのみ使用してください。他の目的で使用すると、他のデータベースをリカバリ不能にしたり一貫性のない状態にする可能性があります。

自動リカバリの実行前に、トランザクションは、DBA_2PC_PENDING を Collecting、Committed または Prepared の状態で表示できます。DBA が COMMIT_FORCE または ROLLBACK_FORCE を使用して、インダウト・トランザクションに特定の結果を強制した場合は、Forced commit または Forced rollback の状態も表示できます。自動リカバリは通常、このような状態のエントリを削除します。唯一の例外は、トランザクション内の他のサイトとの間で整合性がとられていない状態の強制トランザクションがリカバリ処理で見つかった場合です。この場合、エントリは表内に残り、MIXED 列の値が 'yes' になります。

ただし、状態によっては、自動リカバリの実行が不可能な場合があります。たとえば、リモート・データベースが完全に破損した場合です。たとえ再作成しても、新規のデータベース ID を取得するため、リカバリ処理では元のデータベースと同じとは認識できません（起こり得るエラーは、ORA-02062 です）。このような場合、DBA はプロシージャ PURGE_LOST_DB_ENTRY を使用して、Prepared 以外の状態にあるエントリをクリーンアップできます。これらのエントリはデータベース・リソースを保持していないため、DBA は特に急いでこれらのエントリを解決する必要はありません。

次の表は、トランザクションについての様々な状態の内容および DBA に必要とされるアクションを示します。

表 75-10 PURGE_LOST_DB_ENTRY プロシージャの状態

列の状態	グローバル・トランザクションの状態	ローカル・トランザクションの状態	通常の DBA アクション	代替の DBA アクション
Collecting	ロールバック	ロールバック	なし	PURGE_LOST_DB_ENTRY（注意 1 参照）
Committed	コミット済み	コミット済み	なし	PURGE_LOST_DB_ENTRY（注意 1 参照）
Prepared	不明	準備済み	なし	FORCE COMMIT または ROLLBACK
Forced commit	不明	コミット済み	なし	PURGE_LOST_DB_ENTRY（注意 1 参照）
Forced rollback	不明	ロールバック	なし	PURGE_LOST_DB_ENTRY（注意 1 参照）
Forced commit (mixed)	混合	コミット済み	（注意 2 参照）	
Forced rollback (mixed)	混合	ロールバック	（注意 2 参照）	

注意 1: 重要な再構成が発生して自動リカバリではトランザクションを解決できない場合のみ使用します。例としては、リモート・データベース全体が破損し、ソフトウェアでの再構成によって 2 フェーズ・コミット機能を失う結果になった場合や TP モニターのような外部トランザクション・コーディネータからの情報を失った場合などがあります。

注意 2: 不整合を取り除くために調査したり手動で作業を行い、プロシージャ PURGE_MIXED を使用します。

パラメータ

表 75-11 PURGE_LOST_DB_ENTRY プロシージャのパラメータ

パラメータ	説明
xid	DBA_2PC_PENDING 表にある LOCAL_TRAN_ID 列の値に設定する必要があります。

LOCAL_TRANSACTION_ID ファンクション

このファンクションは、現行のトランザクションについてローカルな（インスタンスに対して）一意の識別子を戻します。現行のトランザクションがない場合は、NULL を戻します。

構文

```
DBMS_TRANSACTION.LOCAL_TRANSACTION_ID (
   create_transaction BOOLEAN := FALSE)
  RETURN VARCHAR2;
```

パラメータ

表 75-12 LOCAL_TRANSACTION_ID ファンクションのパラメータ

パラメータ	説明
create_transaction	TRUE に設定すると、トランザクションが現在アクティブでない場合は、トランザクションを開始します。

STEP_ID ファンクション

このファンクションは、トランザクションの DML 操作を順序付ける、ローカルで（ローカル・トランザクションに対して）一意の正の整数を戻します。

構文

```
DBMS_TRANSACTION.STEP_ID
   RETURN NUMBER;
```

76

DBMS_TRANSFORM

DBMS_TRANSFORM パッケージは、Oracle Advanced Queuing のメッセージ形式の変換機能へのインタフェースを提供します。

> **関連項目：** メッセージ形式の変換の詳細は、『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。

この章では、次の項目について説明します。

DBMS_TRANSFORM サブプログラムの要約

表 76-1 DBMS_TRANSFORM サブプログラム

サブプログラム	説明
「CREATE_TRANSFORMATION プロシージャ」76-2 ページ	ソース・タイプのオブジェクトをターゲット・タイプのオブジェクトにマップする変換を作成します。
「MODIFY_TRANSFORMATION プロシージャ」76-3 ページ	既存の変換を変更します。
「DROP_TRANSFORMATION プロシージャ」76-4 ページ	指定した変換を削除します。

CREATE_TRANSFORMATION プロシージャ

このプロシージャは、ソース・タイプのオブジェクトをターゲット・タイプのオブジェクトにマップする変換を作成します。変換式は、SQL 式または PL/SQL ファンクションのどちらでも作成できます。ターゲット・タイプのオブジェクトを戻す必要があります。

構文

```
DBMS_TRANSFORM.CREATE_TRANSFORMATION (
    schema               VARCHAR2(30),
    name                 VARCHAR2(30),
    from_schema          VARCHAR2(30),
    from_type            VARCHAR2(30),
    to_schema            VARCHAR2(30),
    to_type              VARCHAR2(30),
    transformation       VARCHAR2(4000));
```

パラメータ

表 76-2 CREATE_TRANSFORMATION プロシージャのパラメータ

パラメータ	説明
schema	変換のスキーマを指定します。
name	変換名を指定します。
from_schema	ソース・タイプのスキーマを指定します。
from_type	ソース・タイプを指定します。
to_schema	ターゲット・タイプのスキーマを指定します。
to_type	ターゲット・タイプを指定します。
transformation	ターゲット・タイプのオブジェクトを戻す変換式を指定します。ターゲット・タイプがユーザー定義型である場合、式はターゲット・タイプのオブジェクトを戻すファンクションか、ターゲット・タイプのコンストラクタ式にする必要があります。変換式を指定せずに、MODIFY_TRANSFORMATION を使用してターゲット・タイプの属性の変換を指定することも可能です。

MODIFY_TRANSFORMATION プロシージャ

このプロシージャは、ターゲット・タイプの指定した属性に対するマッピングを変更（または作成）します。変換式は、ターゲット・タイプの指定した属性のタイプを戻す SQL 式または PL/SQL ファンクションにする必要があります。ターゲット・タイプがスカラー・タイプである場合、属性番号 0（ゼロ）を指定する必要があります。ターゲット・タイプがユーザー定義型で、attribute_number が 0（ゼロ）の場合、式はターゲット・タイプのオブジェクトを戻す PL/SQL ファンクションか、ターゲット・タイプのコンストラクタ式にする必要があります。

構文

```
DBMS_TRANSFORM.MODIFY_TRANSFORMATION (
    schema            VARCHAR2(30)
    name              VARCHAR2(30),
    attribute_number  INTEGER,
    transformation    VARCHAR2(4000));
```

パラメータ

表 76-3 MODIFY_TRANSFORMATION プロシージャのパラメータ

パラメータ	説明
schema	変換のスキーマを指定します。
name	変換名を指定します。
attribute_number	ターゲット・タイプがスカラー・タイプの場合は、0（ゼロ）を指定する必要があります。
transformation	変換式は、ターゲット・タイプの指定した属性のタイプを戻す SQL 式または PL/SQL ファンクションにする必要があります。

DROP_TRANSFORMATION プロシージャ

このプロシージャは、指定した変換を削除します。

構文

```
DBMS_TRANSFORM.DROP_TRANSFORMATION (
    schema      VARCHAR2(30),
    name        VARCHAR2(30));
```

パラメータ

表 76-4 DROP_TRANSFORMATION プロシージャのパラメータ

パラメータ	説明
schema	変換のスキーマを指定します。
name	変換名を指定します。

77

DBMS_TTS

このパッケージは、トランスポータブル・セットが自己完結型かどうかをチェックします。すべての違反が、ビュー TRANSPORT_SET_VIOLATIONS から選択できる一時表に挿入されます。

このプロシージャは、execute_catalog_role を付与されているユーザーのみが実行できます。このロールは、初期段階ではユーザー SYS にのみ割り当てられています。

> **関連項目：**『Oracle9*i* データベース管理者ガイド』および『Oracle9*i* データベース移行ガイド』

この章では、次の項目について説明します。

例外

```
ts_not_found  EXCEPTION;
PRAGMA exception_init(ts_not_found, -29304);
ts_not_found_num NUMBER := -29304;

invalid_ts_list  EXCEPTION;
PRAGMA exception_init(invalid_ts_list, -29346);
invalid_ts_list_num NUMBER := -29346;

sys_or_tmp_ts     EXCEPTION;
PRAGMA exception_init(sys_or_tmp_ts, -29351);
sys_or_tmp_ts_num NUMBER := -29351;
```

DBMS_TTS サブプログラムの要約

この 2 つのプロシージャは、データベース管理者がコールするように設計されています。

表 77-1 DBMS_TTS のサブプログラム

サブプログラム	説明
「TRANSPORT_SET_CHECK プロシージャ」77-2 ページ	表領域（トランスポート可能）のセットが自己完結型かどうかをチェックします。
「DOWNGRADE プロシージャ」77-3 ページ	データに関連するトランスポータブル表領域をダウングレードします。

TRANSPORT_SET_CHECK プロシージャ

このプロシージャは、表領域（トランスポート可能）のセットが自己完結型かどうかをチェックします。このプロシージャのコール後、ユーザーはビューから選択して、違反があればそのリストを調べることができます。ビューで行が戻らない場合、表領域のセットは自己完結型です。次に例を示します。

```
SQLPLUS> EXECUTE TRANSPORT_SET_CHECK('foo,bar', TRUE);
SQLPLUS> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
```

構文

```
DBMS_TTS.TRANSPORT_SET_CHECK (
   ts_list          IN VARCHAR2,
   incl_constraints IN BOOLEAN DEFAULT,
   full_closure     IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 77-2 TRANSPORT_SET_CHECK プロシージャのパラメータ

パラメータ	説明
ts_list	表領域のカンマで区切られたリスト。
incl_constraints	表領域が自己完結型かどうかを調べるときに、参照整合性制約を考慮する場合は TRUE を設定します（ts_list 引数でコールされた場合にのみ TRANSPORT_SET_CHECK が動作するように、デフォルトには incl_constraints パラメータが設定されています）。
full_closure	完全または部分的な依存性チェックが必要かどうかを示します。TRUE の場合、IN および OUT ポインタ（依存性）のすべてを処理し、トランスポータブル・セット内で自己完結していないものを違反として獲得します。TSPITR の場合またはトランスポータブルの厳密なバージョンが必要な場合にも、パラメータを TRUE に設定してください。デフォルトは FALSE です。この場合、OUT ポインタのみが違反とみなされます。

DOWNGRADE プロシージャ

このプロシージャは、データに関連するトランスポータブル表領域をダウングレードします。

構文

```
DBMS_TTS.DOWNGRADE;
```

78

DBMS_TYPES

DBMS_TYPES パッケージは、組込みおよびユーザー定義のタイプを表す定数で構成されます。タイプの詳細は、『Oracle *inter*Media ユーザーズ・ガイドおよびリファレンス』を参照してください。

この章では、次の項目について説明します。

DBMS_TYPES の定数

次の表に、DBMS_TYPES パッケージの定数を示します。

表 78-1 DBMS_TYPES の定数

定数	説明
TYPECODE_DATE	DATE タイプ
TYPECODE_NUMBER	NUMBER タイプ
TYPECODE_RAW	RAW タイプ
TYPECODE_CHAR	CHAR タイプ
TYPECODE_VARCHAR2	VARCHAR2 タイプ
TYPECODE_VARCHAR	VARCHAR タイプ
TYPECODE_MLSLABEL	MLSLABEL タイプ
TYPECODE_BLOB	BLOB タイプ
TYPECODE_BFILE	BFILE タイプ
TYPECODE_CLOB	CLOB タイプ
TYPECODE_CFILE	CFILE タイプ
TYPECODE_TIMESTAMP	TIMESTAMP タイプ
TYPECODE_TIMESTAMP_TZ	TIMESTAMP_TZ タイプ
TYPECODE_TIMESTAMP_LTZ	TIMESTAMP_LTZ タイプ
TYPECODE_INTERVAL_YM	INTERVAL_YM タイプ
TYPECODE_INTERVAL_DS	INTERVAL_DS タイプ
TYPECODE_REF	REF タイプ
TYPECODE_OBJECT	OBJECT タイプ
TYPECODE_VARRAY	VARRAY コレクション・タイプ
TYPECODE_TABLE	ネストした表のコレクション・タイプ
TYPECODE_NAMEDCOLLECTION	
TYPECODE_OPAQUE	OPAQUE タイプ
SUCCESS	
NO_DATA	

例外

- INVALID_PARAMETERS
- INCORRECT_USAGE
- TYPE_MISMATCH

79

DBMS_UTILITY

このパッケージは、各種のユーティリティ・サブプログラムを提供します。

DBMS_UTILITY は、各パーティションに対してジョブを実行します。INIT.ORA パラメータの JOB_QUEUE_PROCESSES を正しく設定して、同時実行ジョブの数を制御するのは、ユーザーの責任です。正しい構文についての最小限度のエラー・チェックがあります。すべてのエラーは、SNP トレース・ファイルにレポートされます。

この章では、次の項目について説明します。

DBMS_UTILITY の要件およびタイプ

要件

DBMS_UTILITY は、NAME_RESOLVE、COMPILE_SCHEMA および ANALYZE_SCHEMA プロシージャに対するコール・ユーザーの権限で実行されます。これは、SQL の正しい動作のために必要です。

このパッケージは、SYS として実行されません。権限は、DBMS_DDL を使用してチェックされます。

タイプ

```
type uncl_array IS TABLE OF VARCHAR2(227) INDEX BY BINARY_INTEGER;
```
"USER"."NAME"."COLUMN"@LINK のリストは、ここに格納されます。

```
type name_array IS TABLE OF VARCHAR2(30) INDEX BY BINARY_INTEGER;
```
NAME のリストは、ここに格納されます。

```
type dblink_array IS TABLE OF VARCHAR2(128) INDEX BY BINARY_INTEGER;
```
データベース・リンクのリストは、ここに格納されます。

```
TYPE index_table_type IS TABLE OF BINARY_INTEGER INDEX BY BINARY_INTEGER;
```
オブジェクトの生成順序は、ここに戻されます。

```
TYPE number_array IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
```
ユーザーのためのオブジェクトの生成順序は、ここに戻されます。

```
  TYPE instance_record IS RECORD (
       inst_number   NUMBER,
       inst_name     VARCHAR2(60));
  TYPE instance_table IS TABLE OF instance_record INDEX BY BINARY_INTEGER;
```
アクティブなインスタンス番号とインスタンス名のリスト。

instance_table の索引の開始は 1 で、instance_table は稠密です。

DBMS_UTILITY サブプログラムの要約

表 79-1 DBMS_UTILITY のサブプログラム

サブプログラム	説明
「COMPILE_SCHEMA プロシージャ」79-4 ページ	指定したスキーマ内にあるすべてのプロシージャ、ファンクション、パッケージおよびトリガーをコンパイルします。
「ANALYZE_SCHEMA プロシージャ」79-5 ページ	スキーマ内にあるすべての表、クラスタおよび索引を分析します。
「ANALYZE_DATABASE プロシージャ」79-6 ページ	データベース内にあるすべての表、クラスタおよび索引を分析します。
「FORMAT_ERROR_STACK ファンクション」79-7 ページ	現行のエラー・スタックをフォーマットします。
「FORMAT_CALL_STACK ファンクション」79-8 ページ	現行のコール・スタックをフォーマットします。
「IS_CLUSTER_DATABASE ファンクション」79-8 ページ	このデータベースがクラスタ・データベース・モードで実行しているかどうかを検出します。
「GET_TIME ファンクション」79-8 ページ	現在の時間を 100 分の 1 秒単位で検出します。
「GET_PARAMETER_VALUE ファンクション」79-9 ページ	指定した init.ora パラメータの値を取得します。
「NAME_RESOLVE プロシージャ」79-10 ページ	指定した名前を解決します。
「NAME_TOKENIZE プロシージャ」79-12 ページ	指定した名前を解析するためにパーサーをコールします。
「COMMA_TO_TABLE プロシージャ」79-12 ページ	カンマで区切られた名前のリストを、名前の PL/SQL 表に変換します。
「TABLE_TO_COMMA プロシージャ」79-13 ページ	名前の PL/SQL 表を、カンマで区切られた名前のリストに変換します。
「PORT_STRING ファンクション」79-14 ページ	Oracle とオペレーティング・システムのバージョンを一意に識別する文字列を戻します。
「DB_VERSION プロシージャ」79-14 ページ	データベースに関するバージョン情報を戻します。
「MAKE_DATA_BLOCK_ADDRESS ファンクション」79-15 ページ	ファイル番号とブロック番号を指定して、データ・ブロック・アドレスを作成します。

表 79-1 DBMS_UTILITY のサブプログラム（続き）

サブプログラム	説明
「DATA_BLOCK_ADDRESS_FILE ファンクション」79-16 ページ	データ・ブロック・アドレスのファイル番号部分を取得します。
「DATA_BLOCK_ADDRESS_BLOCK ファンクション」79-17 ページ	データ・ブロック・アドレスのブロック番号部分を取得します。
「GET_HASH_VALUE ファンクション」79-17 ページ	指定した文字列についてハッシュ値を計算します。
「ANALYZE_PART_OBJECT プロシージャ」79-18 ページ	
「EXEC_DDL_STATEMENT プロシージャ」79-19 ページ	parse_string で DDL 文を実行します。
「CURRENT_INSTANCE ファンクション」79-20 ページ	現在接続しているインスタンス番号を戻します。
「ACTIVE_INSTANCES プロシージャ」79-20 ページ	

COMPILE_SCHEMA プロシージャ

このプロシージャは、指定したスキーマ内にあるすべてのプロシージャ、ファンクション、パッケージおよびトリガーをコンパイルします。このプロシージャのコール後、ステータスが INVALID の項目をビュー ALL_OBJECTS から選択して、すべてのオブジェクトが正常にコンパイルされたかどうかを調べます。

Oracle Enterprise Manager のコマンドを使用すると、INVALID のオブジェクトに関連するエラーを表示できます。

```
SHOW ERRORS <type> <schema>.<name>
```

構文

```
DBMS_UTILITY.COMPILE_SCHEMA (
   schema VARCHAR2);
```

パラメータ

表 79-2 COMPILE_SCHEMA プロシージャのパラメータ

パラメータ	説明
schema	スキーマの名前。

例外

表 79-3 COMPILE_SCHEMA プロシージャの例外

例外	説明
ORA-20000	このスキーマ内のいずれかのオブジェクトに対して権限が不十分です。

ANALYZE_SCHEMA プロシージャ

このプロシージャは、スキーマ内にあるすべての表、クラスタおよび索引で ANALYZE コマンドを実行します。このプロシージャを使用して、非オプティマイザ統計情報を収集します。オプティマイザ統計情報の場合は、DBMS_STATS.GATHER_SCHEMA_STATS プロシージャを使用します。

構文

```
DBMS_UTILITY.ANALYZE_SCHEMA (
   schema           VARCHAR2,
   method           VARCHAR2,
   estimate_rows    NUMBER   DEFAULT NULL,
   estimate_percent NUMBER   DEFAULT NULL,
   method_opt       VARCHAR2 DEFAULT NULL);
```

パラメータ

表 79-4 ANALYZE_SCHEMA プロシージャのパラメータ

パラメータ	説明
schema	スキーマの名前。
method	ESTIMATE、COMPUTE または DELETE のいずれかを設定します。 ESTIMATE を設定した場合は、estimate_rows または estimate_percent のいずれかを 0（ゼロ）以外に設定する必要があります。
estimate_rows	推定する行数。
estimate_percent	推定する行のパーセント。 estimate_rows が指定されている場合、このパラメータは無視されます。

表 79-4 ANALYZE_SCHEMA プロシージャのパラメータ

パラメータ	説明
method_opt	次の書式の分析方法オプション。 [FOR TABLE] [FOR ALL [INDEXED] COLUMNS] [SIZE n] [FOR ALL INDEXES]

例外

表 79-5 ANALYZE_SCHEMA プロシージャの例外

例外	説明
ORA-20000	このスキーマ内のいずれかのオブジェクトに対して権限が不十分です。

ANALYZE_DATABASE プロシージャ

このプロシージャは、データベース内にあるすべての表、クラスタおよび索引で ANALYZE コマンドを実行します。このプロシージャを使用して、非オプティマイザ統計情報を収集します。オプティマイザ統計情報の場合は、DBMS_STATS.GATHER_DATABASE_STATS プロシージャを使用します。

構文

```
DBMS_UTILITY.ANALYZE_DATABASE (
   method           VARCHAR2,
   estimate_rows    NUMBER   DEFAULT NULL,
   estimate_percent NUMBER   DEFAULT NULL,
   method_opt       VARCHAR2 DEFAULT NULL);
```

パラメータ

表 79-6 ANALYZE_DATABASE プロシージャのパラメータ

パラメータ	説明
method	ESTIMATE、COMPUTE または DELETE のいずれかを設定します。 ESTIMATE を設定した場合は、estimate_rows または estimate_percent のいずれかを 0（ゼロ）以外に設定する必要があります。
estimate_rows	推定する行数。
estimate_percent	推定する行のパーセント。 estimate_rows が指定されている場合、このパラメータは無視されます。
method_opt	次の書式の分析方法オプション。 [FOR TABLE] [FOR ALL [INDEXED] COLUMNS] [SIZE n] [FOR ALL INDEXES]

例外

表 79-7 ANALYZE_DATABASE プロシージャの例外

例外	説明
ORA-20000	このデータベース内のいずれかのオブジェクトに対して権限が不十分です。

FORMAT_ERROR_STACK ファンクション

このファンクションは、現行のエラー・スタックをフォーマットします。このファンクションは、全エラー・スタックを表示するための例外ハンドラで使用できます。

構文

```
DBMS_UTILITY.FORMAT_ERROR_STACK
  RETURN VARCHAR2;
```

戻り値

最大 2000 バイトまでのエラー・スタックを戻します。

FORMAT_CALL_STACK ファンクション

このファンクションは、現行のコール・スタックをフォーマットします。このファンクションは、コール・スタックにアクセスするための任意ストアド・プロシージャまたはトリガーで使用できます。このファンクションは、デバッグ時に役立ちます。

構文

```
DBMS_UTILITY.FORMAT_CALL_STACK
  RETURN VARCHAR2;
```

プラグマ

```
pragma restrict_references(format_call_stack,WNDS);
```

戻り値

最大 2000 バイトまでのコール・スタックを戻します。

IS_CLUSTER_DATABASE ファンクション

このファンクションは、このデータベースがクラスタ・データベース・モードで実行されているかどうかを検出します。

構文

```
DBMS_UTILITY.IS_CLUSTER_DATABASE
  RETURN BOOLEAN;
```

戻り値

インスタンスがクラスタ・データベース・モードで起動されている場合は TRUE を戻し、それ以外の場合は FALSE を戻します。

GET_TIME ファンクション

このファンクションは、現在の時間を 100 分の 1 秒単位で検出します。これは、主に経過時間を判断するのに役立ちます。

構文

```
DBMS_UTILITY.GET_TIME
  RETURN NUMBER;
```

戻り値

任意の時点からの時間を 100 分の 1 秒の数で戻します。

GET_PARAMETER_VALUE ファンクション

このファンクションは、指定した init.ora パラメータの値を取得します。

構文

```
DBMS_UTILITY.GET_PARAMETER_VALUE (
   parnam IN     VARCHAR2,
   intval IN OUT BINARY_INTEGER,
   strval IN OUT VARCHAR2)
  RETURN BINARY_INTEGER;
```

パラメータ

表 79-8 GET_PARAMETER_VALUE ファンクションのパラメータ

パラメータ	説明
parnam	パラメータ名。
intval	整数パラメータの値、または文字列パラメータの値の長さ。
strval	文字列パラメータの値。

戻り値

表 79-9 GET_PARAMETER_VALUE ファンクションの戻り値

戻り値	説明
partyp	パラメータ・タイプ パラメータが整数またはブール・パラメータの場合は 0 (ゼロ) パラメータが文字列またはファイル・パラメータの場合は 1

例

```
DECLARE
  parnam VARCHAR2(256);
  intval BINARY_INTEGER;
  strval VARCHAR2(256);
  partyp BINARY_INTEGER;
BEGIN
  partyp := dbms_utility.get_parameter_value('max_dump_file_size',
                                             intval, strval);
  dbms_output.put('parameter value is: ');
  IF partyp = 1 THEN
    dbms_output.put_line(strval);
  ELSE
    dbms_output.put_line(intval);
  END IF;
  IF partyp = 1 THEN
    dbms_output.put('parameter value length is: ');
    dbms_output.put_line(intval);
  END IF;
  dbms_output.put('parameter type is: ');
  IF partyp = 1 THEN
    dbms_output.put_line('string');
  ELSE
    dbms_output.put_line('integer');
  END IF;
END;
```

NAME_RESOLVE プロシージャ

このプロシージャは、指定した名前を解決します。必要に応じてシノニム変換と認可チェックが含まれます。

構文

```
DBMS_UTILITY.NAME_RESOLVE (
   name          IN  VARCHAR2,
   context       IN  NUMBER,
   schema        OUT VARCHAR2,
   part1         OUT VARCHAR2,
   part2         OUT VARCHAR2,
   dblink        OUT VARCHAR2,
   part1_type    OUT NUMBER,
   object_number OUT NUMBER);
```

パラメータ

表 79-10 NAME_RESOLVE プロシージャのパラメータ

パラメータ	説明
name	オブジェクト名。 この名前はフォーム [[a.]b.]c[@d] で指定します。a、b、c は SQL 識別子、d は DB リンクです。DB リンクでは、構文チェックは実行されません。DB リンクが指定されている場合や名前が DB リンクの一部に変換される場合、オブジェクトは解決されませんが、schema、part1、part2 および dblink OUT の各パラメータは入力されます。 a、b および c は、デリミタ付き識別子の場合があり、NLS 文字（シングルおよびマルチバイト）を含んでいる場合があります。
context	0 ～ 8 の範囲の整数を指定します。
schema	オブジェクトのスキーマ。name パラメータにスキーマが指定されていない場合、schema は、名前を解決することによって決定されます。
part1	名前の最初の部分。この名前のタイプは、part1_type に指定されます（シノニム、プロシージャまたはパッケージ）。
part2	このパラメータが NULL 以外の場合は、part1 で示されるパッケージ内のプロシージャ名です。
dblink	このパラメータが NULL 以外の場合、データベース・リンクは、name の一部として指定されたか、または name がデータベース・リンクの一部に変換されるシノニムとして指定されたかのいずれかです。後者の場合、part1_type はシノニムを示します。
part1_type	part1 のタイプは、次のとおりです。 5 シノニム 7 プロシージャ（最上位レベル） 8 ファンクション（最上位レベル） 9 パッケージ シノニムの場合、name は、データベース・リンクの一部に変換するシノニムです。名前変換がさらに必要な場合は、このリモート・ノードで DBMS_UTILITY.NAME_RESOLVE プロシージャをコールする必要があります。
object_number	オブジェクト識別子。

例外

すべてのエラーは、例外を呼び出すことによって処理されます。オブジェクト名の指定時に起こり得る各種の構文エラーに基づいて、広範囲にわたる例外が用意されています。

NAME_TOKENIZE プロシージャ

このプロシージャは、パーサーをコールして、"a [. b [. c]][@ dblink]" と指定した名前を解析します。二重引用符を削除するか、または引用符がない場合は大文字に変換します。ソートに関するすべてのコメントは無視し、意味的な分析は行いません。不明な値は NULL のまま残ります。

構文

```
DBMS_UTILITY.NAME_TOKENIZE (
   name    IN  VARCHAR2,
   a       OUT VARCHAR2,
   b       OUT VARCHAR2,
   c       OUT VARCHAR2,
   dblink  OUT VARCHAR2,
   nextpos OUT BINARY_INTEGER);
```

パラメータ

各 a、b、c および dblink は、それぞれの anext、bnext、cnext、dnext の次のトークンが開始される場所を示します。

COMMA_TO_TABLE プロシージャ

このプロシージャは、カンマで区切られた名前のリストを、名前の PL/SQL 表に変換します。このプロシージャは、NAME_TOKENIZE を使用して、名前とカンマを区別します。

構文

```
DBMS_UTILITY.COMMA_TO_TABLE (
   list   IN  VARCHAR2,
   tablen OUT BINARY_INTEGER,
   tab    OUT UNCL_ARRAY);
```

パラメータ

表 79-11 COMMA_TO_TABLE プロシージャのパラメータ

パラメータ	説明
list	表名のカンマで区切られたリスト。
tablen	PL/SQL 表にある表名の数。
tab	表名のリストを含んだ PL/SQL 表。

戻り値

PL/SQL 表が、値 1..n と n+1 is null とともに戻されます。

使用上の注意

list は、空ではないカンマで区切られたリストであることが必要です。カンマ以外で区切られたリストの場合は拒否されます。二重引用符内のカンマは無視されます。

カンマで区切られたリストに、ハイフン（-）などのマルチバイト・キャラクタを含めることはできません。

tab にある値は、変換されずに元のリストからカットされます。

TABLE_TO_COMMA プロシージャ

このプロシージャは、名前の PL/SQL 表を、カンマで区切られた名前のリストに変換します。PL/SQL 表を 1..n に変換して n+1 null で終了します。

構文

```
DBMS_UTILITY.TABLE_TO_COMMA (
   tab    IN  UNCL_ARRAY,
   tablen OUT BINARY_INTEGER,
   list   OUT VARCHAR2);
```

パラメータ

表 79-12 TABLE_TO_COMMA プロシージャのパラメータ

パラメータ	説明
tab	表名のリストを含んだ PL/SQL 表。
tablen	PL/SQL 表にある表名の数。
list	表名のカンマで区切られたリスト。

戻り値

カンマで区切られたリストと、表内で検出された要素の数が戻されます。

PORT_STRING ファンクション

このファンクションは、オペレーティング・システムと、TWO TASK PROTOCOL バージョンのデータベースを識別する文字列を戻します。たとえば、VAX/VMX-7.1.0.0 が戻ります。

最大長はポート固有の長さです。

構文

```
DBMS_UTILITY.PORT_STRING
   RETURN VARCHAR2;
```

プラグマ

```
pragma restrict_references(port_string, WNDS, RNDS, WNPS, RNPS);
```

DB_VERSION プロシージャ

このプロシージャは、データベースに関するバージョン情報を戻します。

構文

```
DBMS_UTILITY.DB_VERSION (
   version       OUT VARCHAR2,
   compatibility OUT VARCHAR2);
```

パラメータ

表 79-13 DB_VERSION プロシージャのパラメータ

パラメータ	説明
version	データベースの内部ソフトウェア・バージョンを表す文字列（例：7.1.0.0.0）。 この文字列の長さは可変なので、データベース・バージョンによって決定されます。
compatibility	COMPATIBLE がある init.ora パラメータによって決定する、データベースの互換性設定。 このパラメータが init.ora ファイルで指定されていない場合は、NULL が戻されます。

MAKE_DATA_BLOCK_ADDRESS ファンクション

このファンクションは、ファイル番号とブロック番号を指定したデータ・ブロック・アドレスを作成します。データ・ブロック・アドレスは、データベース内のブロックを識別するために使用する内部構造です。このファンクションは、データ・ブロック・アドレスを含んだ特定の固定表にアクセスするときに役立ちます。

構文

```
DBMS_UTILITY.MAKE_DATA_BLOCK_ADDRESS (
   file  NUMBER,
   block NUMBER)
  RETURN NUMBER;
```

パラメータ

表 79-14 MAKE_DATA_BLOCK_ADDRESS ファンクションのパラメータ

パラメータ	説明
file	ブロックを含むファイル。
block	ブロック増分値に基づくファイル内でのブロックのオフセット。

プラグマ

```
pragma restrict_references(make_data_block_address, WNDS, RNDS, WNPS,
RNPS);
```

戻り値

表 79-15 MAKE_DATA_BLOCK_ADDRESS ファンクションの戻り値

戻り値	説明
dba	データ・ブロック・アドレス。

DATA_BLOCK_ADDRESS_FILE ファンクション

このファンクションは、データ・ブロック・アドレスのファイル番号部分を取得します。

構文

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_FILE (
   dba NUMBER)
  RETURN NUMBER;
```

パラメータ

表 79-16 DATA_BLOCK_ADDRESS_FILE ファンクションのパラメータ

パラメータ	説明
dba	データ・ブロック・アドレス。

プラグマ

```
pragma restrict_references(data_block_address_file, WNDS, RNDS, WNPS,
RNPS);
```

戻り値

表 79-17 DATA_BLOCK_ADDRESS_FILE ファンクションの戻り値

戻り値	説明
file	ブロックを含むファイル。

DATA_BLOCK_ADDRESS_BLOCK ファンクション

このファンクションは、データ・ブロック・アドレスのブロック番号部分を取得します。

構文

```
DBMS_UTILITY.DATA_BLOCK_ADDRESS_BLOCK (
   dba NUMBER)
  RETURN NUMBER;
```

パラメータ

表 79-18 DATA_BLOCK_ADDRESS_BLOCK ファンクションのパラメータ

パラメータ	説明
dba	データ・ブロック・アドレス。

プラグマ

```
pragma restrict_references(data_block_address_block, WNDS, RNDS, WNPS,
RNPS);
```

戻り値

表 79-19 DATA_BLOCK_ADDRESS_BLOCK ファンクションの戻り値

戻り値	説明
block	ブロックのブロック・オフセット。

GET_HASH_VALUE ファンクション

このファンクションは、指定した文字列についてハッシュ値を計算します。

構文

```
DBMS_UTILITY.GET_HASH_VALUE (
   name      VARCHAR2,
   base      NUMBER,
   hash_size NUMBER)
  RETURN NUMBER;
```

パラメータ

表 79-20 GET_HASH_VALUE ファンクションのパラメータ

パラメータ	説明
name	ハッシュする文字列。
base	戻されるハッシュ値が始まるベース値。
hash_size	必要とするハッシュ表のサイズ。

プラグマ

```
pragma restrict_references(get_hash_value, WNDS, RNDS, WNPS, RNPS);
```

戻り値

ハッシュ値は、入力文字列に基づいています。たとえば、ハッシュ値が 1000 ～ 3047 の範囲にある文字列についてハッシュ値を取得するには、ベース値として 1000、hash_size 値として 2048 を使用します。hash_size パラメータは、2 の累乗を使用すると動作が最適になります。

ANALYZE_PART_OBJECT プロシージャ

このプロシージャは、次の SQL と同じです。

```
"ANALYZE TABLE|INDEX [<schema>.]<object_name> PARTITION <pname> [<command_type>]
[<command_opt>] [<sample_clause>]
```

オブジェクトの各パーティションについて、ジョブ・キューを使用して並列に実行します。

構文

```
DBMS_UTILITY.ANALYZE_PART_OBJECT (
   schema        IN VARCHAR2 DEFAULT NULL,
   object_name   IN VARCHAR2 DEFAULT NULL,
   object_type   IN CHAR     DEFAULT 'T',
   command_type  IN CHAR     DEFAULT 'E',
   command_opt   IN VARCHAR2 DEFAULT NULL,
   sample_clause IN VARCHAR2 DEFAULT 'SAMPLE 5 PERCENT');
```

パラメータ

表 79-21 ANALYZE_PART_OBJECT プロシージャのパラメータ

パラメータ	説明
schema	object_name のスキーマ。
object_name	分析するオブジェクトの名前。パーティション化されている必要があります。
object_type	オブジェクトのタイプ。T（表）または I（索引）である必要があります。
command_type	次のいずれかになります。 C（統計情報の計算） E（統計情報の推定） D（統計情報の削除） V（構造の検証）
command_opt	command_type のその他のオプション。 C と E については、FOR 表、FOR のすべての LOCAL 索引、FOR のすべての列、または分析統計（表）の 'FOR' オプションをいくつか組み合せることが可能です。V については、object_type が T のときに CASCADE が可能です。
sample_clause	command_type が 'E' の場合に使用するサンプル句。

EXEC_DDL_STATEMENT プロシージャ

このプロシージャは、parse_string で DDL 文を実行します。

構文

```
DBMS_UTILITY.EXEC_DDL_STATEMENT (
   parse_string IN VARCHAR2);
```

パラメータ

表 79-22 EXEC_DDL_STATEMENT プロシージャのパラメータ

パラメータ	説明
parse_string	実行する DDL 文。

CURRENT_INSTANCE ファンクション

このファンクションは、現在接続しているインスタンス番号を戻します。接続しているインスタンスが切断されると、NULL を戻します。

構文

```
DBMS_UTILITY.CURRENT_INSTANCE
   RETURN NUMBER;
```

ACTIVE_INSTANCES プロシージャ

構文

```
DBMS_UTILITY.ACTIVE_INSTANCE (
   instance_table   OUT INSTANCE_TABLE,
   instance_count   OUT NUMBER);
```

パラメータ

表 79-23 ACTIVE_INSTANCES プロシージャのパラメータ

プロシージャ	説明
instance_table	アクティブなインスタンス番号と名前のリストが含まれます。進行中のインスタンスがない場合、リストは空になります。
instance_count	アクティブなインスタンスの数。

80

DBMS_WM

この章では、Oracle Database Workspace Manager（または単に Workspace Manager と呼ばれる）へのプログラミング・インタフェースである DBMS_WM パッケージを使用して、長いトランザクションを処理する方法を説明します。

作業領域の管理は、同一レコード（つまり行）の異なるバージョンを 1 つ以上の作業領域に保持するためのデータベースの機能を参照します。データベースのユーザーは、これらのバージョンを個別に変更できます。

> **関連項目：** Workspace Manager の概念と使用方法の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。このマニュアルには、この章で説明する参照情報も記述されています。

この章では、次の項目について説明します。

DBMS_WM サブプログラムの要約

表 80-1 DBMS_WM のサブプログラム

サブプログラム	説明
「AlterSavepoint プロシージャ」80-6 ページ	セーブポイントの説明を変更します。
「AlterWorkspace プロシージャ」80-7 ページ	作業領域の説明を変更します。
「BeginDDL プロシージャ」80-8 ページ	指定した表の DDL（データ定義言語）セッションを開始します。
「BeginResolve プロシージャ」80-9 ページ	競合解消セッションを開始します。
「CommitDDL プロシージャ」80-10 ページ	指定した表の DDL セッション中に加えられた DDL（データ定義言語）変更をコミットし、その DDL セッションを終了します。
「CommitResolve プロシージャ」80-12 ページ	競合解消セッションを終了し、BeginResolve が実行された後に作業領域に加えられた変更を永続的に保存します。
「CompressWorkspace プロシージャ」80-13 ページ	作業領域内の削除可能なセーブポイントを削除し、その作業領域に対する Workspace Manager のメタデータ構造を最小化します。
「CompressWorkspaceTree プロシージャ」80-16 ページ	作業領域内の削除可能なセーブポイントとその子作業領域をすべて削除します。また、影響を受ける作業領域の Workspace Manager のメタデータ構造を最小化し、セーブポイントの削除により発生する冗長データを除去します。
「CopyForUpdate プロシージャ」80-18 ページ	バージョン対応表の LOB 列（BLOB、CLOB または NCLOB）を変更可能にします。
「CreateSavepoint プロシージャ」80-19 ページ	現行のバージョンにセーブポイントを作成します。
「CreateWorkspace プロシージャ」80-20 ページ	データベース内に新規作業領域を作成します。
「DeleteSavepoint プロシージャ」80-23 ページ	セーブポイントとバージョン対応表の関連行を削除します。
「DisableVersioning プロシージャ」80-25 ページ	バージョン化された行を表でサポートできるように作成されたサポート構造をすべて削除します。
「DropReplicationSupport プロシージャ」80-27 ページ	GenerateReplicationSupport プロシージャで作成したレプリケーション・サポート・オブジェクトを削除します。

表 80-1 DBMS_WM のサブプログラム（続き）

サブプログラム	説明
「EnableVersioning プロシージャ」80-28 ページ	複数バージョンの行を表でサポートするための構造を作成して、表をバージョン対応にします。
「FreezeWorkspace プロシージャ」80-30 ページ	作業領域へのアクセス、およびユーザーによる作業領域での変更を制限します。
「GenerateReplicationSupport プロシージャ」80-33 ページ	Workspace Manager オブジェクトのマルチマスター・レプリケーションに必要な構造を作成し、新しく作成したマスター・グループのマスター・アクティビティを開始します。
「GetConflictWorkspace ファンクション」80-35 ページ	セッションで SetConflictWorkspace プロシージャを実行した作業領域の名前を戻します。
「GetDiffVersions ファンクション」80-35 ページ	セッションで SetDiffVersions 操作を実行した組合せ（作業領域、セーブポイント）の名前を戻します。
「GetLockMode ファンクション」80-36 ページ	現行のセッションのロック・モードを戻します。このロック・モードは、バージョン化された行および前のバージョンの対応する行にアクセスできるかどうかを判別します。
「GetMultiWorkspaces ファンクション」80-37 ページ	バージョン対応表の複数作業領域ビューで表示可能な作業領域の名前を戻します。
「GetOpContext ファンクション」80-37 ページ	現行のセッションについて、現行の操作のコンテキストを戻します。
「GetPrivs ファンクション」80-38 ページ	指定の作業領域に対するカレント・ユーザーのすべての権限が記載された、カンマで区切られたリストを戻します。
「GetSessionInfo プロシージャ」80-39 ページ	現行の作業領域とセッション・コンテキストに関する情報を取得します。
「GetWorkspace ファンクション」80-41 ページ	セッションの現行の作業領域を戻します。
「GotoDate プロシージャ」80-41 ページ	現行の作業領域において指定された日付および時刻の時点またはその近くに移動します。
「GotoSavepoint プロシージャ」80-43 ページ	現行の作業領域における指定したセーブポイントに移動します。
「GotoWorkspace プロシージャ」80-44 ページ	現行のセッションを指定された作業領域に移動します。
「GrantSystemPriv プロシージャ」80-45 ページ	システムレベルの権限（特定の作業領域に制限されない）をユーザーおよびロールに付与します。`grant_option` パラメータを使用すると、権限を付与されたユーザーおよびロールが、他のユーザーおよびロールに対して指定した権限を付与できるようになります。

表 80-1 DBMS_WM のサブプログラム（続き）

サブプログラム	説明
「GrantWorkspacePriv プロシージャ」80-47 ページ	ユーザーおよびロールに作業領域レベルの権限を付与します。grant_option パラメータを使用すると、権限を付与されたユーザーおよびロールが、他のユーザーおよびロールに対して指定した権限を付与できるようになります。
「IsWorkspaceOccupied ファンクション」80-49 ページ	作業領域にアクティブなセッションがあるかどうかをチェックします。
「LockRows プロシージャ」80-50 ページ	指定された表におけるバージョン化された行およびその親作業領域の対応行に対するアクセスを制御します。
「MergeTable プロシージャ」80-51 ページ	作業領域の表（すべての行または WHERE 句で指定した行）への変更をその親作業領域に適用します。
「MergeWorkspace プロシージャ」80-53 ページ	作業領域で行われた変更をすべてその親作業領域に適用します。またオプションで、作業領域を削除します。
「RecoverAllMigratingTables プロシージャ」80-55 ページ	Workspace Manager の移行手順に失敗した後、一貫性のない状態になっているすべての表で、移行プロセスの完了を試行します。
「RecoverMigratingTable プロシージャ」80-56 ページ	Workspace Manager の移行手順が失敗して一貫性のない状態になっている単一の表で、移行プロセスの完了を試行します。
「RefreshTable プロシージャ」80-58 ページ	親作業領域の表（すべての行または WHERE 句で指定された行）に行われた変更をすべて作業領域に適用します。
「RefreshWorkspace プロシージャ」80-59 ページ	親作業領域で行われた変更をすべて作業領域に適用します。
「RelocateWriterSite プロシージャ」80-61 ページ	Workspace Manager のレプリケーション環境で、nonwriter サイトの 1 つを新規 writer サイトにします（古い writer サイトは nonwriter サイトになります）。
「RemoveWorkspace プロシージャ」80-62 ページ	作業領域に関連付けられているすべての行バージョンを廃棄し、その作業領域を削除します。
「RemoveWorkspaceTree プロシージャ」80-63 ページ	作業領域とその子作業領域に関連付けられているすべての行バージョンを廃棄し、影響を受ける作業領域を削除します。
「ResolveConflicts プロシージャ」80-65 ページ	作業領域間の競合を解消します。
「RevokeSystemPriv プロシージャ」80-67 ページ	ユーザーおよびロールからシステムレベルの権限を取り消し（削除）ます。
「RevokeWorkspacePriv プロシージャ」80-69 ページ	指定された作業領域のユーザーおよびロールから作業領域レベルの権限を取り消し（削除）ます。

表 80-1 DBMS_WM のサブプログラム（続き）

サブプログラム	説明
「RollbackDDL プロシージャ」 80-70 ページ	指定した表の DDL セッション中に加えられた DDL 変更をロールバック（取消し）して、その DDL セッションを終了します。
「RollbackResolve プロシージャ」 80-71 ページ	競合解消セッションを終了し、BeginResolve が実行された後に作業領域に行われた変更をすべて廃棄します。
「RollbackTable プロシージャ」 80-72 ページ	作業領域において指定した表（すべての行または WHERE 句で指定された行）に対して行われた変更をすべて廃棄します。
「RollbackToSP プロシージャ」 80-74 ページ	指定したセーブポイント以降、作業領域でバージョン対応表に対して行われた変更をすべて廃棄します。
「RollbackWorkspace プロシージャ」 80-76 ページ	作業領域でバージョン対応表に対して行われた変更をすべて廃棄します。
「SetConflictWorkspace プロシージャ」 80-77 ページ	作業領域とその親との間に競合が発生しているかどうかを判別します。
「SetDiffVersions プロシージャ」 80-78 ページ	バージョン対応表における 2 つのセーブポイントおよび共通祖先（ベース）について、値の差異を検出します。このような差異を示す差異ビューの内容を変更します。
「SetLockingOFF プロシージャ」 80-80 ページ	現行のセッションにおける Workspace Manager のロックを使用禁止にします。
「SetLockingON プロシージャ」 80-81 ページ	現行のセッションにおける Workspace Manager のロックを使用可能にします。
「SetMultiWorkspaces プロシージャ」 80-82 ページ	指定された単数または複数の作業領域を、バージョン対応表の複数作業領域ビューで表示できるようにします。
「SetWoOverwriteOFF プロシージャ」 80-83 ページ	EnableVersioning または SetWoOverwriteON プロシージャにより使用可能にされた VIEW_WO_OVERWRITE 履歴オプションを使用禁止にし、VIEW_W_OVERWRITE に変更します。
「SetWoOverwriteON プロシージャ」 80-84 ページ	SetWoOverwriteOFF プロシージャにより使用禁止にされた VIEW_WO_OVERWRITE 履歴オプションを使用可能にします。
「SetWorkspaceLockModeOFF プロシージャ」 80-85 ページ	指定した作業領域における Workspace Manager のロックを使用禁止にします。
「SetWorkspaceLockModeON プロシージャ」 80-86 ページ	指定した作業領域における Workspace Manager のロックを使用可能にします。
「SynchronizeSite プロシージャ」 80-88 ページ	RelocateWriterSite プロシージャを使用した writer サイトの移動後に、Workspace Manager レプリケーション環境のローカル・サイト（古い writer サイト）を最新にします。

表 80-1 DBMS_WM のサブプログラム（続き）

サブプログラム	説明
「UnfreezeWorkspace プロシージャ」80-89 ページ	FreezeWorkspace の結果を元に戻し、作業領域へのアクセスと変更を可能にします。
「UnlockRows プロシージャ」80-90 ページ	指定された表におけるバージョン化された行およびその親作業領域の対応行にアクセスできるようにします。

注意： Workspace Manager のサブプログラムは大部分がプロシージャですが、一部にはファンクションもあります。ファンクションであれば、大部分の場合、名前が Get で始まります（GetConflictWorkspace ファンクションおよび GetWorkspace ファンクションなど）。

この章では、プロシージャという用語を使用しますが、これは通常、プロシージャおよびファンクションの両方を指します。

AlterSavepoint プロシージャ

セーブポイントの説明を変更します。

構文

```
DBMS_WM.AlterSavepoint(
   workspace      IN VARCHAR2,
   sp_name        IN VARCHAR2,
   sp_description IN VARCHAR2);
```

パラメータ

表 80-2 AlterSavepoint プロシージャのパラメータ

パラメータ	説明
workspace	セーブポイントが作成された作業領域の名前。大文字と小文字が区別されます。
sp_name	セーブポイント名。大文字と小文字が区別されます。
sp_description	セーブポイントの説明。

使用上の注意

セーブポイントの現行の説明を確認するには、ALL_WORKSPACE_SAVEPOINTS メタデータ・ビューの DESCRIPTION 列でそのセーブポイントの値を検証します。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

ユーザーが作業領域の所有者またはセーブポイントの所有者のいずれでもない場合、または WM_ADMIN_ROLE ロールを付与されていない場合には、例外が発生します。

例

次の例では、NEWWORKSPACE 作業領域の SP1 セーブポイントの説明を変更します。

```
EXECUTE DBMS_WM.AlterSavepoint ('NEWWORKSPACE', 'SP1', 'First set of changes for
scenario');
```

AlterWorkspace プロシージャ

作業領域の説明を変更します。

構文

```
DBMS_WM.AlterWorkspace(
   workspace              IN VARCHAR2,
   workspace_description  IN VARCHAR2);
```

パラメータ

表 80-3 AlterWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
workspace_description	作業領域の説明。

使用上の注意

作業領域の現行の説明を確認するには、ALL_WORKSPACES メタデータ・ビューの DESCRIPTION 列で該当する作業領域の値を検証します。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

ユーザーが作業領域の所有者ではない場合、または WM_ADMIN_ROLE ロールを付与されていない場合には、例外が発生します。

例

次の例では、NEWWORKSPACE 作業領域の説明を変更します。

```
EXECUTE DBMS_WM.AlterWorkspace ('NEWWORKSPACE', 'Testing proposed scenario B');
```

BeginDDL プロシージャ

指定された表で DDL（データ定義言語）セッションを開始します。

構文

```
DBMS_WM.BeginDDL(
   table_name  IN VARCHAR2);
```

パラメータ

表 80-4 BeginDDL プロシージャのパラメータ

パラメータ	説明
table_name	バージョン対応表の名前。大文字と小文字は区別されません。

使用上の注意

このプロシージャは、DDL セッションを開始し、特殊な表（table_name に *_LTS* を追加した名前の表）を作成します。このプロシージャのコール後に、1 つ以上の DDL 操作を表で実行するか、表に基づく索引またはトリガーを実行すると、CommitDDL プロシージャまたは RollbackDDL プロシージャのいずれかをコールできます。

このプロシージャは、特殊な *<table-name>_LTS* 表以外に、他のオブジェクトも作成します。

- *<table-name>_LTS* 表には、*<table-name>* 表と同じトリガー、列および索引があります。
- *<table-name>* 表との参照整合性制約がある親表ごとに、*<table-name>_LTS* 表について同じ制約が定義されます。
- *<table-name>_LTS* 表と *<table-name>* 表では、対応するトリガー、列および参照整合性制約の名前は同じです。
- *<table-name>* 表の各索引の場合、*<table-name>_LTS* 表の対応する索引の名前は、*<index-name>_LTS* の形式になります。
- *<table-name>_LTS* 表の主キー制約の名前は、*<primary-key>_LTS* の形式になります。

バージョン対応表に関連した DDL 操作の実行方法、および Oracle Replication 環境でのバージョン対応表に対する DDL 操作の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

次の条件が 1 つ以上該当する場合、例外が発生します。

- table_name が存在しない、またはバージョン対応でない。
- ユーザーに CREATE TABLE 権限が付与されていない。
- table_name について、オープンした DDL セッションが存在する （つまり、この表を指定して BeginDDL プロシージャがすでにコールされ、CommitDDL プロシージャまたは RollbackDDL プロシージャがこの表を指定してコールされていない)。

例

次の例では、DDL セッションを開始し、COLA_MARKETING_BUDGET_LTS という名前の特殊な表（スケルトン表）を使用して COMMENTS 列を COLA_MARKETING_BUDGET 表に追加し、変更をコミットして DDL セッションを終了します。

```
EXECUTE DBMS_WM.BeginDDL('COLA_MARKETING_BUDGET');
ALTER TABLE cola_marketing_budget_lts ADD (comments VARCHAR2(100));
EXECUTE DBMS_WM.CommitDDL('COLA_MARKETING_BUDGET');
```

BeginResolve プロシージャ

競合解消セッションを開始します。

構文

```
DBMS_WM.BeginResolve(
   workspace  IN VARCHAR2);
```

パラメータ

表 80-5 BeginResolve プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。

使用上の注意

このプロシージャは、競合解消セッションを開始します。このプロシージャの実行中、作業領域は 1WRITER モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

このプロシージャのコール後、競合が発生している任意の表に対し、必要に応じて ResolveConflicts プロシージャを実行すると、CommitResolve プロシージャまたは RollbackResolve プロシージャのいずれかをコールできます。競合解消の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

次の条件が 1 つ以上該当する場合、例外が発生します。

- workspace で 1 つ以上のデータベース・トランザクションがオープンしている。
- BeginResolve プロシージャを実行しているユーザーに、workspace およびその親作業領域へのアクセス権限が付与されていない。

例

次の例では、Workspace1 で競合解消セッションを開始します。

```
EXECUTE  DBMS_WM.BeginResolve ('Workspace1');
```

CommitDDL プロシージャ

指定された表での DDL セッション中に加えられた DDL（データ定義言語）変更をコミットし、DDL セッションを終了します。

構文

```
DBMS_WM.CommitDDL(
   table_name           IN VARCHAR2
   [, ignore_last_error IN BOOLEAN DEFAULT FALSE]);
```

パラメータ

表 80-6 CommitDDL プロシージャのパラメータ

パラメータ	説明
table_name	バージョン対応表の名前。大文字と小文字は区別されません。
ignore_last_error	ブール値（TRUE または FALSE）。 TRUE に設定すると、CommitDDL プロシージャへの前回のコール中に発生した最新のエラー（ある場合）を無視します。最新のエラーに関する情報は、USER_WM_VT_ERRORS および ALL_WM_VT_ERRORS メタデータ・ビューに格納されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています （詳細は、「使用上の注意」を参照）。 FALSE（デフォルト）に設定すると、CommitDDL プロシージャへの前回のコール中に発生した最新のエラー（ある場合）を無視しません。

使用上の注意

このプロシージャは、DDL セッション中に、バージョン対応表に加えられた変更、およびバージョン対応表に基づく任意の索引、トリガーおよび参照整合性制約に加えられた変更をコミットします。また、このプロシージャは、BeginDDL プロシージャで作成された特殊な *<table-name>_LTS* 表（スケルトン表）を削除します。

バージョン対応表に関連した DDL 操作の実行方法、および Oracle Replication 環境でのバージョン対応表に対する DDL 操作の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

CommitDDL プロシージャへのコールに失敗すると、表は一貫性のない状態のままになります。このような状態が発生した場合は、エラーの原因を修正する必要があります。USER_WM_VT_ERRORS および ALL_WM_VT_ERRORS メタデータ・ビューを検証して、SQL 文とエラー・メッセージを確認してください。たとえば、列を追加するための表領域が不十分であるために、CommitDDL プロシージャが失敗する場合があります。エラーの原因を修正し、ignore_last_error パラメータをデフォルト値 FALSE に設定して、CommitDDL プロシージャを再度コールします。ただし、コールに再度失敗し、エラーの原因を修正できない場合、このエラーを無視することが安全かつ適切であることが確実であれば、ignore_last_error パラメータ値を TRUE に設定して CommitDDL プロシージャをコールし、エラーを無視できます。エラーを無視することが安全かつ適切であるかどうかは必ず確認してください。

次の条件が 1 つ以上該当する場合、例外が発生します。

- table_name が存在しない、またはバージョン対応でない。
- ユーザーに CREATE TABLE 権限が付与されていない。
- table_name について、オープンした DDL セッションが存在しない（つまり、この表を指定して BeginDDL プロシージャがコールされていない、または CommitDDL プロシージャか RollbackDDL プロシージャがこの表を指定してコールされている）。

CommitDDL プロシージャをコールしたとき、無効な DDL 操作によって例外が発生する場合があります（サポートされている DDL 操作については、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。

例

次の例では、DDL セッションを開始し、COLA_MARKETING_BUDGET_LTS という名前の特殊な表を使用して COMMENTS 列を COLA_MARKETING_BUDGET 表に追加し、変更をコミットして DDL セッションを終了します。

```
EXECUTE DBMS_WM.BeginDDL('COLA_MARKETING_BUDGET');
ALTER TABLE cola_marketing_budget_lts ADD (comments VARCHAR2(100));
EXECUTE DBMS_WM.CommitDDL('COLA_MARKETING_BUDGET');
```

CommitResolve プロシージャ

競合解消セッションを終了し、BeginResolve プロシージャが実行された後に作業領域に加えられた変更を永続的に保存します。

構文

```
DBMS_WM.CommitResolve(
   workspace  IN VARCHAR2);
```

パラメータ

表 80-7 CommitResolve プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。

使用上の注意

このプロシージャは、BeginResolve プロシージャで開始された現行の競合解消セッションを終了し、競合解消セッションが開始された後に作業領域に加えられた変更をすべて保存します。このプロシージャと、変更をすべて廃棄する RollbackResolve プロシージャを対比してください。

競合解消の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

次の条件が 1 つ以上該当する場合、例外が発生します。

- workspace で 1 つ以上のデータベース・トランザクションがオープンしている。
- WM_ADMIN_ROLE ロールを付与されていないユーザー、または workspace で BeginResolve プロシージャを実行しなかったユーザーによりプロシージャがコールされた。

例

次の例では、Workspace1 で競合解消セッションを終了し、変更をすべて保存します。

```
EXECUTE  DBMS_WM.CommitResolve ('Workspace1');
```

CompressWorkspace プロシージャ

作業領域における削除可能なセーブポイントを削除し、その作業領域に対する Workspace Manager のメタデータ構造を最小化します（削除可能なセーブポイントについては、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。

構文

```
DBMS_WM.CompressWorkspace(
   workspace                    IN VARCHAR2,
   compress_view_wo_overwrite   IN BOOLEAN
   [, firstSP                   IN VARCHAR2 DEFAULT NULL
   [, secondSP                  IN VARCHAR2 DEFAULT NULL] ]
   [, auto_commit               IN BOOLEAN DEFAULT TRUE]);
```

または

```
DBMS_WM.CompressWorkspace(
   workspace                    IN VARCHAR2
   [, firstSP                   IN VARCHAR2 DEFAULT NULL
   [, secondSP                  IN VARCHAR2 DEFAULT NULL] ]
   [, auto_commit               IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-8 CompressWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
compress_view_wo_overwrite	ブール値（TRUE または FALSE）。 TRUE の場合、バージョニングが使用可能にされたときに VIEW_WO_OVERWRITE が指定されていても、影響を受けるセーブポイント間の履歴情報が削除されます。 FALSE の場合、バージョニングが使用可能にされたときに VIEW_WO_OVERWRITE が指定されていると、表に対する（影響を受けるセーブポイント間の）履歴情報は削除されません（VIEW_WO_OVERWRITE が表に対して指定されていない場合、その表の履歴情報は、設定されているパラメータ値にかかわらず削除されます）。このパラメータを指定しないプロシージャ形式を使用した場合は、FALSE が設定されたとみなされます。

表 80-8 CompressWorkspace プロシージャのパラメータ（続き）

パラメータ	説明
firstSP	最初のセーブポイント。セーブポイント名は大文字と小文字が区別されます。 workspace および firstSP のみが指定されている場合、作業領域作成と firstSP（ただし firstSP は含みません）の間の削除可能なセーブポイントはすべて削除されます。 workspace、firstSP および secondSP が指定されている場合、firstSP（削除可能なセーブポイントの場合は firstSP を含みます）から secondSP（ただし secondSP は含みません）までの削除可能なセーブポイントはすべて削除されます。 workspace のみが指定されており、セーブポイントが指定されていない場合、その作業領域における削除可能なセーブポイントがすべて削除されます。
secondSP	2 番目のセーブポイント。firstSP（削除可能なセーブポイントの場合は firstSP を含みます）から secondSP（ただし secondSP は含みません）までの削除可能なセーブポイントはすべて削除されます。 ただし、secondSP が LATEST の場合、firstSP（削除可能なセーブポイントの場合は firstSP を含みます）から作業領域の最後までの削除可能なセーブポイントはすべて削除されます。 セーブポイント名は大文字と小文字が区別されます。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

作業領域内の明示的セーブポイントのすべてまたは一部が不要になった場合、作業領域を圧縮できます。圧縮処理には、次のような利点があります。

- 削除したセーブポイント名を再利用できます（ただし、既存のセーブポイントと同じ名前で新しくセーブポイントを作成することはできません）。
- Workspace Manager 操作の実行時のパフォーマンスが改善されます。
- Workspace Manager 構造に使用するディスク記憶域を節減できます。

このプロシージャの実行中、現行の作業領域は NO_ACCESS モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

作業領域にセッションがある場合（LIVE 作業領域を除く）、またはユーザーが作業領域のセーブポイントを指定した GotoDate プロシージャ操作または GotoSavepoint プロシージャ操作を実行した場合、作業領域は圧縮できません。

compress_view_wo_overwrite パラメータを指定しないプロシージャ形式を使用した場合は、パラメータに FALSE 値が設定されたとみなされます。

VIEW_WO_OVERWRITE およびその他の履歴オプションの詳細は、「EnableVersioning プロシージャ」を参照してください。

ユーザーが、workspace で変更にアクセスまたはマージする権限を付与されていない場合には、例外が発生します。

作業領域およびその子作業領域をすべて圧縮するには、CompressWorkspaceTree プロシージャを使用します。

例

次に、NEWWORKSPACE を圧縮した例を示します。

```
EXECUTE DBMS_WM.CompressWorkspace ('NEWWORKSPACE');
```

次の例では、作成した作業領域と SP1 セーブポイントとの間の明示的セーブポイントをすべて削除することにより、NEWWORKSPACE を圧縮します。

```
EXECUTE DBMS_WM.CompressWorkspace ('NEWWORKSPACE', 'SP1');
```

次の例は、明示的セーブポイント SP1 および SP2 より前（SP2 を含まない）の明示的セーブポイントすべてを削除することにより、NEWWORKSPACE を圧縮します。

```
EXECUTE DBMS_WM.CompressWorkspace ('NEWWORKSPACE', 'SP1', 'SP2');
```

次の例は、B_focus_1 を圧縮します。firstSP および secondSP パラメータのデフォルト値を受け入れ（つまり明示的セーブポイントをすべて削除する）、auto_commit パラメータの値に FALSE を指定しています。

```
EXECUTE DBMS_WM.CompressWorkspace ('B_focus_1', auto_commit => FALSE);
```

CompressWorkspaceTree プロシージャ

作業領域内の削除可能なセーブポイントとその子作業領域をすべて削除します（削除可能なセーブポイントについては、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。また、影響を受ける作業領域の Workspace Manager のメタデータ構造を最小化し、セーブポイントの削除により発生する冗長データを除去します。

構文

```
DBMS_WM.CompressWorkspaceTree(
   workspace                       IN VARCHAR2
   [, compress_view_wo_overwrite   IN BOOLEAN DEFAULT FALSE]
   [, auto_commit                  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-9 CompressWorkspaceTree プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
compress_view_wo_overwrite	ブール値（TRUE または FALSE）。 TRUE の場合、バージョニングが使用可能にされたときに VIEW_WO_OVERWRITE が指定されていても、履歴情報が削除されます。 FALSE（デフォルト）の場合、バージョニングが使用可能にされたときに VIEW_WO_OVERWRITE が指定されていると、履歴情報は削除されません（VIEW_WO_OVERWRITE が表に対して指定されていない場合、その表の履歴情報は、設定されているパラメータ値にかかわらず削除されます）。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

影響を受ける作業領域の明示的セーブポイントが不要となった場合（たとえば、いずれのセーブポイントにも移動またはロールバックする必要がない場合）、作業領域とその子作業領域をすべて圧縮できます。データベース作業領域の階層の説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

圧縮処理には、次のような利点があります。

- 削除したセーブポイント名を再利用できます （ただし、既存のセーブポイントと同じ名前で新しくセーブポイントを作成することはできません）。
- Workspace Manager 操作の実行時のパフォーマンスが改善されます。
- Workspace Manager 構造に使用するディスク記憶域を節減できます。

このプロシージャの実行中、現行の作業領域は NO_ACCESS モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

作業領域にセッションがある場合（LIVE を指定した作業領域を除く）、または作業領域でセーブポイントを指定することにより GotoDate プロシージャ操作または GotoSavepoint プロシージャ操作を実行したユーザーがいる場合は、作業領域を圧縮できません。

ユーザーが、workspace で変更にアクセスまたはマージする権限を付与されていない場合には、例外が発生します。

CompressWorkspaceTree の操作が、影響を受けるいずれかの作業領域で失敗した場合、全操作がロールバックされ、作業領域は一切圧縮されません。

明示的セーブポイントのすべてまたは一部を削除して単一の作業領域を圧縮するには、CompressWorkspace プロシージャを使用します。

例

次に、NEWWORKSPACE およびその子作業領域すべてを圧縮した例を示します。

```
EXECUTE DBMS_WM.CompressWorkspaceTree ('NEWWORKSPACE');
```

次の例は、NEWWORKSPACE およびその子作業領域すべてを圧縮します。compress_view_wo_overwrite パラメータのデフォルト値を受け入れ（つまり明示的セーブポイントをすべて削除する）、auto_commit パラメータの値に FALSE を指定しています。

```
EXECUTE DBMS_WM.CompressWorkspaceTree ('B_focus_1', auto_commit => FALSE);
```

CopyForUpdate プロシージャ

バージョン対応表の LOB 列（BLOB、CLOB または NCLOB）を変更可能にします。バージョン対応表に LOB 列がある場合にのみこのプロシージャを使用してください。

構文

```
DBMS_WM.CopyForUpdate(
   table_name       IN VARCHAR2,
   [, where_clause  IN VARCHAR2 DEFAULT '']);
```

パラメータ

表 80-10 CopyForUpdate プロシージャのパラメータ

パラメータ	説明
table_name	1 つ以上の LOB 列を含む表の名前。大文字と小文字は区別されません。
where_clause	影響を受ける行を識別する WHERE 句（WHERE キーワードを除く）。 例 :'department_id = 20' WHERE 句で指定できるのは、主キーの列のみです。WHERE 句に副問合せを含めることはできません。 where_clause が指定されていない場合、table_name のすべての行が影響を受けます。

使用上の注意

このプロシージャは、1 つ以上のラージ・オブジェクト（LOB）列を含むバージョン対応表でのみ使用できます。DBMS_LOB パッケージを使用して更新を実行した場合、バージョニングのビューでは INSTEAD OF トリガーが起動されないため、CopyForUpdate プロシージャを使用する必要があります。Workspace Manager は、バージョニングのビューに INSTEAD OF トリガーを作成し、copy-on-write セマンティクスを実装します（LOB ではない列については、更新操作およびトリガー作業を直接実行できます）。

例

次の例は、DOC_ID = 1 の設定が行われた文書について、TABLE1 の SOURCE_CLOB 列を更新しています。

```
Declare
  clob_var
Begin
   /* This procedure copies the LOB columns if necessary, that is,
      if the row with doc_id = 1 has not been versioned in the
      current version */
   dbms_wm.copyForUpdate('table1', 'doc_id = 1');
```

```
    select source_clob into clob_var
    from   table1
    where  doc_id = 1 for update;

    dbms_lob.write(clob_var,<amount>, <offset>, buff);

  End;
```

CreateSavepoint プロシージャ

現行のバージョンにセーブポイントを作成します。

構文

```
DBMS_WM.CreateSavepoint(
   workspace       IN VARCHAR2,
   savepoint_name  IN VARCHAR2
   [, description  IN VARCHAR2 DEFAULT NULL]
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-11 CreateSavepoint プロシージャのパラメータ

パラメータ	説明
workspace	セーブポイントを作成する作業領域の名前。大文字と小文字が区別されます。
savepoint_name	作成されるセーブポイントの名前。大文字と小文字が区別されます。
description	作成されるセーブポイントの説明。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

セーブポイントには明示的に関連付けられた権限がありません。つまり、作業領域にアクセスできるユーザーはすべて、その作業領域内にセーブポイントを作成できます。

このプロシージャは、作業領域内で処理を行っているユーザーがいる場合でも実行でき、データベース・トランザクションがオープンしていてもかまいません。

このプロシージャの実行中、現行の作業領域は READ_ONLY モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

次の条件が 1 つ以上該当する場合、例外が発生します。

- ユーザーがいる作業領域が最新バージョンではない（たとえば、ユーザーが GotoDate プロシージャをコールした場合など）。
- workspace が存在しない。
- savepoint_name がすでに存在する。
- ユーザーに指定した作業領域に移動する権限がない。

例

次の例では、NEWWORKSPACE 作業領域に Savepoint1 というセーブポイントを作成します。

```
EXECUTE DBMS_WM.CreateSavepoint ('NEWWORKSPACE', 'Savepoint1');
```

CreateWorkspace プロシージャ

データベース内に新規作業領域を作成します。

構文

```
DBMS_WM.CreateWorkspace(
   workspace       IN VARCHAR2
   [, description  IN VARCHAR2 DEFAULT NULL]
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

または

```
DBMS_WM.CreateWorkspace(
   workspace       IN VARCHAR2,
   isrefreshed     IN BOOLEAN
   [, description  IN VARCHAR2 DEFAULT NULL]
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-12 CreateWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。名前は大文字と小文字を区別します。一意の名前を指定してください（同じ名前の作業領域を複数作成することはできません）。
isrefreshed	ブール値（TRUE または FALSE）。 TRUE の場合、作業領域が継続的にリフレッシュされます。**継続的にリフレッシュされる作業領域**では、親作業領域に変更が行われると、親作業領域でマージまたはロールバック処理が行われた後、自動的に作業領域に適用されます。つまり、変更を適用するために RefreshWorkspace プロシージャをコールする必要がありません。継続的にリフレッシュされる作業領域の詳細は、「使用上の注意」を参照してください。 FALSE の場合、作業領域が継続的にリフレッシュされません。作業領域をリフレッシュするには、RefreshWorkspace プロシージャをコールする必要があります。 isrefreshed パラメータを構文で使用しない場合、その作業領域は自動的にリフレッシュされません。
description	作業領域の説明。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

新規作業領域は、現行の作業領域の子です。セッションで作業領域が明示的に入力されていない場合、作業領域は LIVE データベースの作業領域にあり、新規作業領域は LIVE 作業領域の子となります。データベース作業領域の階層の説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

現行の作業領域の現行のバージョンに、暗黙的セーブポイントが作成されます（現行のバージョンは、現行の作業領域の最新バージョンである必要はありません）。明示的および暗黙的セーブポイントについては、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

このプロシージャの実行中、現行の作業領域は READ_ONLY モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

このプロシージャは、作成された作業領域に暗黙的に移動しません。作業領域に移動するには、GotoWorkspace プロシージャを使用します。

継続的にリフレッシュされる作業領域（isrefreshed の値が TRUE）には、次のルールが適用されます。

- 継続的にリフレッシュされる作業領域は、LIVE 作業領域の子として作成する必要があります。
- 継続的にリフレッシュされる作業領域は、リーフ作業領域（つまり、子作業領域がない領域）であることが必要です。
- 継続的にリフレッシュされる作業領域を作成するため、セッションは最新バージョンにあることが必要です。
- 継続的にリフレッシュされる作業領域では、SetLockingOFF プロシージャまたは SetWorkspaceLockModeOFF プロシージャを使用してロックをオフにすることはできません。

次の条件が 1 つ以上該当する場合、例外が発生します。

- workspace がすでに存在している。
- ユーザーに作業領域を作成する権限がない。

例

次の例では、データベースに NEWWORKSPACE という作業領域を作成します。

```
EXECUTE DBMS_WM.CreateWorkspace ('NEWWORKSPACE');
```

DeleteSavepoint プロシージャ

セーブポイントおよびバージョン対応表の関連行を削除します。

構文

```
DBMS_WM.DeleteSavepoint(
   workspace                       IN VARCHAR2,
   savepoint_name                  IN VARCHAR2)
   [, compress_view_wo_overwrite  IN BOOLEAN DEFAULT FALSE]
   [, auto_commit                  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-13 DeleteSavepoint プロシージャのパラメータ

パラメータ	説明
workspace	セーブポイントが作成された作業領域の名前。大文字と小文字が区別されます。
savepoint_name	削除されるセーブポイントの名前。大文字と小文字が区別されます。
compress_view_wo_overwrite	ブール値（TRUE または FALSE）。 TRUE の場合、バージョニングが使用可能にされたときに VIEW_WO_OVERWRITE が指定されていても、履歴情報が削除されます。 FALSE（デフォルト）の場合、バージョニングが使用可能にされたときに VIEW_WO_OVERWRITE が指定されていると、履歴情報は削除されません（VIEW_WO_OVERWRITE が表に対して指定されていない場合、その表の履歴情報は、設定されているパラメータ値にかかわらず削除されます）。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

セーブポイントが不要になった場合（移動やロールバックの必要がなくなった場合など）に、削除します。

セーブポイントの削除には、次のような利点があります。

- 削除したセーブポイント名を再利用できます（ただし、既存のセーブポイントと同じ名前で新しくセーブポイントを作成することはできません）。
- Workspace Manager 操作の実行時のパフォーマンスが改善されます。
- Workspace Manager 構造に使用するディスク記憶域を節減できます。

このプロシージャの実行中、現行の作業領域は NO_ACCESS モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

セーブポイントを削除するには、WM_ADMIN_ROLE ロールが付与されているか、目的の作業領域またはセーブポイントの所有者である必要があります。

このプロシージャは、データベース・トランザクションがオープンされているセッションがある場合、またはユーザーが作業領域のセーブポイントを指定して、GotoDate プロシージャ操作または GotoSavepoint プロシージャ操作を実行した場合は、実行できません。

次の条件が 1 つ以上該当する場合、例外が発生します。

- すでに 1 つ以上のセッションが workspace にある（作業領域が LIVE の場合を除く）。
- workspace が存在しない。
- savepoint_name が存在しない。
- savepoint_name が削除可能なセーブポイントでない（削除可能なセーブポイントについては、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。
- ユーザーに指定した作業領域に移動する権限がない。

例

次の例では、NEWWORKSPACE 作業領域にある Savepoint1 というセーブポイントを削除します。

```
EXECUTE DBMS_WM.DeleteSavepoint ('NEWWORKSPACE', 'Savepoint1');
```

DisableVersioning プロシージャ

バージョン化された行を表でサポートできるように作成されたサポート構造をすべて削除します。

構文

```
DBMS_WM.DisableVersioning(
   table_name           IN VARCHAR2
   [, force             IN BOOLEAN DEFAULT FALSE]
   [, ignore_last_error IN BOOLEAN DEFAULT FALSE]);
```

パラメータ

表 80-14 DisableVersioning プロシージャのパラメータ

パラメータ	説明
table_name	表の名前、またはマルチレベルの参照整合性制約によって関連付けられた表名のカンマで区切られたリスト（マルチレベルの参照整合性制約の説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。表名は大 / 小文字が区別されません。
force	ブール値（TRUE または FALSE）。 TRUE の場合、バージョニングが使用禁止になる前に、LIVE 以外の作業領域のデータがすべて強制的に廃棄されます。 FALSE（デフォルト）の場合、table_name が LIVE 以外の作業領域で変更されており、table_name が変更された作業領域がまだ存在していると、バージョニングを使用禁止にできません。
ignore_last_error	ブール値（TRUE または FALSE）。 TRUE に設定すると、DisableVersioning プロシージャへの前回のコール中に発生した最新のエラー（ある場合）を無視します。最新のエラーに関する情報は、USER_WM_VT_ERRORS および ALL_WM_VT_ERRORS メタデータ・ビューに格納されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています（詳細は、「使用上の注意」を参照）。 FALSE（デフォルト）に設定すると、DisableVersioning プロシージャへの前回のコール中に発生した最新のエラー（ある場合）を無視しません。

使用上の注意

このプロシージャは、EnableVersioning プロシージャの効果を元に戻すために使用します。Workspace Manager のインフラストラクチャ（サポート構造）において行のバージョニングを削除しますが、LIVE 作業領域のユーザー・データには影響を与えません。作業領域の階層およびセーブポイントは残りますが、行はすべて LIVE 作業領域と同じになります（バージョニングが使用禁止にされた表の行において LIVE 作業領域に複数のバージョンが存在する場合は、行の最新バージョンのみが保持されます）。

DisableVersioning プロシージャへのコールに失敗すると、表は一貫性のない状態のままになります。このような状態が発生した場合は、エラーの原因を修正し（USER_WM_VT_ERRORS および ALL_WM_VT_ERRORS メタデータ・ビューを検証して、SQL 文とエラー・メッセージを確認します）、ignore_last_error パラメータをデフォルト値 FALSE に設定して、DisableVersioning プロシージャを再度コールする必要があります。ただし、コールに再度失敗し、エラーの原因を修正できない場合、このエラーを無視することが安全かつ適切であることが確実であれば、ignore_last_error パラメータ値を TRUE に設定して DisableVersioning プロシージャをコールし、エラーを無視できます。エラーを無視することが安全かつ適切であるかどうかは必ず確認してください。

DisableVersioning プロシージャが失敗する原因には、次の場合があります。

- 表に作業領域のデータが多数含まれ、DisableVersioning プロシージャに必要な UNDO 表領域のサイズが不十分な場合
- ユーザー定義トリガーのバージョン対応表からバージョン非対応表への転送中に、コンパイル・エラーが発生した場合

force の値が FALSE で、次のいずれかが該当する場合、DisableVersioning 操作は失敗します。

- LIVE 作業領域以外の作業領域のユーザーにより表が変更されている。
- LIVE 作業領域以外の作業領域の表でバージョン化された行がある。

表のバージョニングを使用禁止にできるのは、表の所有者または WM_ADMIN_ROLE ロールを付与されたユーザーのみです。

バージョン対応表およびバージョン対応表を所有するユーザーは削除できません。まず、関連するすべての表でバージョニングを使用禁止にする必要があります。

表がバージョン対応ではない場合は、例外が発生します。

Oracle Replication 環境の表でバージョニングを使用禁止にする場合のガイドラインと他の情報は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

例

次の例では、EMPLOYEE 表のバージョニングを使用禁止にします。

```
EXECUTE DBMS_WM.DisableVersioning ('employee');
```

次の例では、EMPLOYEE 表のバージョニングを使用禁止にして、DisableVersioning プロシージャへの前回のコール中に発生した最新のエラーを無視します。

```
EXECUTE DBMS_WM.DisableVersioning ('employee', ignore_last_error => true);
```

次の例では、マルチレベルの参照整合性制約を持つ EMPLOYEE、DEPARTMENT および LOCATION 表のバージョニングを使用禁止にします。

```
EXECUTE DBMS_WM.DisableVersioning('employee,department,location');
```

DropReplicationSupport プロシージャ

GenerateReplicationSupport プロシージャで作成したレプリケーション・サポート・オブジェクトを削除します。

構文

```
DBMS_WM.DropReplicationSupport();
```

パラメータ

なし。

使用上の注意

このプロシージャを使用するには、レプリケーションの Workspace Manager オブジェクトへの適用方法を理解する必要があります。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。また、Oracle Replication の主要な概念と技術についても理解する必要があります。詳細は、『Oracle9*i* アドバンスト・レプリケーション』および『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

このプロシージャは、writer サイトでレプリケーション管理者として実行する必要があります。

このプロシージャは、nonwriter サイトにおけるすべてのバージョン対応表のレプリケーション・サポートを削除します。ただし、バージョン対応表をバージョン非対応にはしません。

例

次の例では、以前に GenerateReplicationSupport プロシージャを使用して使用可能にしたレプリケーション・サポートを削除します。

```
DBMS_WM.DropReplicationSupport();
```

EnableVersioning プロシージャ

複数バージョンの行を表でサポートするために必要な構造を作成して、表をバージョン対応にします。

構文

```
DBMS_WM.EnableVersioning(
   table_name  IN VARCHAR2
   [, hist     IN VARCHAR2 DEFAULT 'NONE']);
```

パラメータ

表 80-15 EnableVersioning プロシージャのパラメータ

パラメータ	説明
table_name	表の名前、またはマルチレベルの参照整合性制約によって関連付けられた表の名前のカンマで区切られたリスト（マルチレベルの参照整合性制約の説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。表の名前は 25 文字以内で指定します。大文字と小文字は区別されません。
hist	table_name への変更を追跡する履歴オプション。次のいずれかの値になります。 NONE: 表の変更は追跡されません（これがデフォルトです）。 VIEW_W_OVERWRITE: 上書きを行う（*W*_OVERWRITE）オプション。履歴情報を含める <table_name>_HIST というビュー（詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）が作成されます。ただし、表示されるのは、同じバージョンの表に対する最新の変更のみです。バージョンに対する変更の履歴は保持されません。つまり、同じバージョンの行に対する後続の変更が前の変更に上書きされます（<table_name>_HIST ビューの CREATETIME 列には、最新の更新時刻のみが含まれています）。 VIEW_WO_OVERWRITE: 上書きを行わない（*WO*_OVERWRITE）オプション。履歴情報を含める <table_name>_HIST というビュー（詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）が作成され、同じバージョンの表に対するすべての変更が表示されます。バージョンに対する変更の履歴が保持されます。つまり、同じバージョンの行に対して後続の変更が行われても、前の変更に上書きされません。

使用上の注意

バージョン対応表には、主キーを定義する必要があります。

表のバージョニングを使用可能にできるのは、表の所有者または WM_ADMIN ロールを付与されたユーザーのみです。

バージョン対応表およびバージョン対応表を所有するユーザーは削除できません。まず、関連するすべての表でバージョニングを使用禁止にする必要があります。

SYS が所有する表は、バージョン対応にできません。

次の条件が 1 つ以上該当する場合、例外が発生します。

- table_name がすでにバージョン対応である。
- table_name に表のリストが含まれ、そのリストに含まれていない表との参照整合性制約を持つ表がある。

VIEW_WO_OVERWRITE *hist* オプションを指定して表がバージョン対応にされた場合、後で SetWoOverwriteOFF プロシージャおよび SetWoOverwriteON プロシージャをコールすることにより、このオプションが一度使用禁止にされてから、再度使用可能にされます。

履歴オプションによって、変更をログして監査できます。

履歴オプションは、GotoDate プロシージャの動作に影響を与えます。プロシージャの詳細は、「使用上の注意」を参照してください。

Oracle Replication 環境の表をバージョン対応にする場合のガイドラインと他の情報は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

現時点での注意事項および制約事項は次のとおりです。

- バージョン対応表に参照整合性の制約を定義している場合は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』に記載された考慮事項および制約事項に注意してください。
- バージョン対応表にトリガーを定義している場合は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』に記載された考慮事項および制約事項に注意してください。
- 表に定義された制約事項および権限は、バージョン対応表に継承されます。
- バージョン対応表での DDL 操作は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明する手順と制約事項に準じています。
- 索引構成表をバージョン対応にすることはできません。
- オブジェクト表をバージョン対応にすることはできません。
- LONG データ・タイプが 1 つ以上ある表をバージョン対応にすることはできません。
- ネストした表列が 1 つ以上ある表をバージョン対応にすることはできません。

例

次の例では、EMPLOYEE 表のバージョニングを使用可能にします。

```
EXECUTE DBMS_WM.EnableVersioning('employee');
```

次の例では、マルチレベルの参照整合性制約を持つ EMPLOYEE、DEPARTMENT および LOCATION 表のバージョニングを使用可能にします。

```
EXECUTE DBMS_WM.EnableVersioning('employee,department,location');
```

FreezeWorkspace プロシージャ

作業領域へのアクセス、およびユーザーによる作業領域の変更を制限します。

構文

```
DBMS_WM.FreezeWorkspace(
   workspace        IN VARCHAR2
   [, freezemode    IN VARCHAR2 DEFAULT 'NO_ACCESS']
   [, freezewriter  IN VARCHAR2 DEFAULT NULL]
   [, force         IN BOOLEAN DEFAULT FALSE]);
```

または

```
DBMS_WM.FreezeWorkspace(
   workspace        IN VARCHAR2,
   session_duration IN BOOLEAN
   [, freezemode    IN VARCHAR2 DEFAULT 'NO_ACCESS']
   [, freezewriter  IN VARCHAR2 DEFAULT NULL]
   [, force         IN BOOLEAN DEFAULT FALSE]);
```

パラメータ

表 80-16 FreezeWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
session_duration	ブール値（TRUE または FALSE）。 TRUE の場合は、FreezeWorkspace プロシージャをコールしたセッションがデータベースから切断されると、作業領域へのアクセス制限が解除されます。この値は、すべてのアクセス制限モードで有効です。 FALSE の場合は、FreezeWorkspace プロシージャをコールしたセッションがデータベースから切断されても、作業領域へのアクセス制限が解除されません。
freezemode	作業領域をアクセス制限するモード。次のいずれかの値になります。 NO_ACCESS: 作業領域でのセッションが許可されません （これがデフォルトです）。 READ_ONLY: 作業領域でのセッションは許可されますが、書込み操作（挿入、更新、削除）を行うことはできません。 1WRITER: 作業領域でのセッションは許可されますが、書込み操作（挿入、更新、削除）の実行が許可されるのは 1 人のユーザー（freezewriter パラメータを参照）に対してのみです。 1WRITER_SESSION: 作業領域でのセッションは許可されますが、書込み操作（挿入、更新、削除）が許可されるのは、FreezeWorkspace プロシージャをコールしたデータベース・セッション（データベース・ユーザーではなく）のみです。作業領域へのアクセス制限が解除されるのは、FreezeWorkspace プロシージャをコールしたセッションがデータベースから切断された後です。 WM_ONLY: Workspace Manager の操作のみが許可されます。セッションで直接データ値を変更したり、表データに関連する問合せを実行することはできません。ただし、子作業領域の作業領域へのマージおよび作業領域内へのセーブポイントの作成は実行できます。
freezewriter	作業領域への変更を許可されたユーザー。freezemode が 1WRITER である場合にのみ指定できます。デフォルトは USER（カレント・ユーザー）です。
force	ブール値（TRUE または FALSE）。 TRUE の場合、作業領域がすでにアクセス制限されている場合でも、強制的にアクセス制限されます。たとえば、TRUE を指定すると、freezemode パラメータ値が異なる作業領域を UnfreezeWorkspace プロシージャを最初にコールせずにアクセス制限させることができます。 FALSE（デフォルト）の場合、作業領域がすでにアクセス制限されている場合でも、アクセス制限を回避できます。

使用上の注意

プロシージャ構文に session_duration パラメータを含めずに指定すると、このパラメータを FALSE に指定した場合と同様になります。つまり、FreezeWorkspace プロシージャをコールしたセッションがデータベースから切断されたとき、作業領域へのアクセス制限は解除されません。

次のいずれかに該当する場合、操作は失敗します。

- workspace がすでにアクセス制限されている場合（force が TRUE の場合を除く）。
- workspace にセッションがあり、freezemode が NO_ACCESS の場合（指定またはデフォルト設定）。
- session_duration が FALSE、freezemode が 1WRITER_SESSION の場合。

freezemode が READ_ONLY または 1WRITER である場合、データベース・トランザクションがアクティブになっていると、作業領域をアクセス制限できません。

次のいずれかに該当する場合のみ、作業領域をアクセス制限できます。

- アクセス制限を行う人が、指定した作業領域の所有者である場合。
- アクセス制限を行う人に、指定した作業領域に対する WM_ADMIN_ROLE、FREEZE_ANY_WORKSPACE 権限または FREEZE_WORKSPACE 権限がある場合。

LIVE 作業領域をアクセス制限できるのは、freezemode が READ_ONLY または 1WRITER の場合のみです。

FreezeWorkspace の効果を元に戻すには、UnfreezeWorkspace プロシージャを使用します。

例

次に、NEWWORKSPACE をアクセス制限した例を示します。

```
EXECUTE DBMS_WM.FreezeWorkspace ('NEWWORKSPACE');
```

GenerateReplicationSupport プロシージャ

Workspace Manager オブジェクトのマルチマスター・レプリケーションで必要な構造を作成し、新しく作成したマスター・グループのマスター・アクティビティを開始します。

構文

```
DBMS_WM.GenerateReplicationSupport(
   mastersites          IN VARCHAR2,
   groupname            IN VARCHAR2
   [, groupdescription  IN VARCHAR2 DEFAULT 'Replication Group for OWM']);
```

パラメータ

表 80-17 GenerateReplicationSupport プロシージャのパラメータ

パラメータ	説明
mastersites	Workspace Manager レプリケーション環境に追加する nonwriter サイトの名前（データベース・リンク）のカンマで区切られたリスト。ローカル・サイト（writer サイト）をリストに含めないでください。
groupname	作成されるマスター・グループの名前。このグループは、標準レプリケーション・マスター・グループとして表示され、Oracle Enterprise Manager も含めたすべての Oracle Replication インタフェースから管理できます。
groupdescription	新規マスター・グループの説明。デフォルトは、Replication Group for OWM です。

使用上の注意

このプロシージャを使用するには、レプリケーションの Workspace Manager オブジェクトへの適用方法を理解する必要があります。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。また、Oracle Replication の主要な概念と技術についても理解する必要があります。詳細は、『Oracle9*i* アドバンスト・レプリケーション』および『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

このプロシージャは、writer サイトでレプリケーション管理者として実行する必要があります。

このプロシージャを実行する前に、次の事項を確認してください。

- mastersites リストに指定したすべてのリモート・サイトに、作業領域、セーブポイントまたはバージョン対応表がないこと。
- リモートとローカルのサイトすべてに、同じバージョンの Workspace Manager がインストールされていること。Workspace Manager のバージョン番号は、WM_INSTALLATION メタデータ・ビューでチェックできます。

- ローカル・サイトにバージョン対応表がある場合は、各リモート・サイトに存在する表がバージョン対応でないこと。

このプロシージャは、次の操作を実行します。

- ローカル・サイト、および mastersites リストに指定したすべてのサイトで、同じバージョンの Workspace Manager が実行されていることを確認します。
- mastersites リストに指定したすべてのリモート・サイトに、作業領域、セーブポイントまたはバージョン対応表がないことを確認します。
- マスター定義サイトとしてのローカル・サイトと writer サイトを持つマスター・グループを、groupname パラメータで指定した名前で作成します。
- Workspace Manager のメタデータ表をこのグループに追加します。
- すべての nonwriter サイト（mastersites リストに指定したリモート・サイト）での Workspace Manager 操作を使用禁止にします。
- ローカル・サイトにバージョン対応表がある場合は、mastersites リストに指定したリモート・サイトごとにその表をバージョン対応にし、レプリケーション用に設定します。
- 新しく作成したマスター・グループのマスター・アクティビティを開始します。

Workspace Manager 環境でのレプリケーション・サポートを削除するには、DropReplicationSupport プロシージャを使用します。

例

次の例では、ある企業における Workspace Manager 環境でのレプリケーション・サポートを生成します。

```
DBMS_WM.GenerateReplicationSupport(
    mastersites       =>  'BACKUP-SITE1.ACME.COM, BACKUP-SITE2.ACME.COM');
    groupname         =>  'OWM-GROUP',
    groupdescription  =>  'OWM Replication group for Acme Corp.');
```

GetConflictWorkspace ファンクション

セッションで SetConflictWorkspace プロシージャを実行した作業領域の名前を戻します。

書式

```
DBMS_WM.GetConflictWorkspace() RETURN VARCHAR2;
```

パラメータ

なし。

使用上の注意

SetConflictWorkspace プロシージャが実行されていない場合は、現行の作業領域の名前が戻されます。

例

次の例では、セッションで SetConflictWorkspace プロシージャを実行した作業領域の名前を表示します。

```
SELECT DBMS_WM.GetConflictWorkspace FROM DUAL;

GETCONFLICTWORKSPACE
--------------------------------------------------------------------------------
B_focus_2
```

GetDiffVersions ファンクション

セッションで SetDiffVersions プロシージャ操作を実行した組合せ（作業領域、セーブポイント）の名前を戻します。

書式

```
DBMS_WM.GetDiffVersions() RETURN VARCHAR2;
```

パラメータ

なし。

使用上の注意

文字列は '(WS1,SP1), (WS2,SP2)' の形式で戻されます。カッコを含めたこの形式は、戻された文字列の一部分を使用して SetDiffVersions プロシージャをコールする場合に便利です。

例

次の例は、セッションで SetDiffVersions プロシージャ操作を実行した組合せ（作業領域、セーブポイント）の名前を表示します。

```
SELECT DBMS_WM.GetDiffVersions FROM DUAL;

GETDIFFVERSIONS
--------------------------------------------------------------------------------
(B_focus_1, LATEST), (B_focus_2, LATEST)
```

GetLockMode ファンクション

現行のセッションのロック・モードを戻します。このロック・モードでは、バージョン化された行および前のバージョンの対応する行にアクセスできるかどうかを判別します。

書式

```
DBMS_WM.GetLockMode() RETURN VARCHAR2;
```

パラメータ

なし。

使用上の注意

このファンクションが戻すのは、E、S、C または NULL のいずれかです。

- E（排他的）、S（共有）および C（引継ぎ）の説明は、SetLockingON プロシージャの lockmode パラメータを参照してください。
- NULL は、ロックが有効ではないことを示します（SetLockingOFF プロシージャをコールするとこの設定が行われます）。

Workspace Manager のロックについては、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。また、SetLockingON プロシージャおよび SetLockingOFF プロシージャの説明も参照してください。

例

次の例は、セッションで有効なロック・モードを表示しています。

```
SELECT DBMS_WM.GetLockMode FROM DUAL;

GETLOCKMODE
--------------------------------------------------------------------------------
C
```

GetMultiWorkspaces ファンクション

バージョン対応表の複数作業領域ビューで表示可能な作業領域の名前を戻します。

書式

```
DBMS_WM.GetMultiWorkspaces() RETURN VARCHAR2;
```

パラメータ

なし。

使用上の注意

このプロシージャは、複数作業領域ビューで表示可能な作業領域の名前を戻します。複数作業領域ビューの説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

複数作業領域ビューで表示可能な作業領域がない場合、NULL が戻されます。複数の作業領域の名前が戻された場合は、名前がカンマで区切られます（例：workspace1,workspace2,workspace3）。

複数作業領域ビューで作業領域を表示可能にするには、SetMultiWorkspaces プロシージャを使用します。

例

次の例では、複数作業領域ビューで表示可能な作業領域の名前を表示します。

```
SELECT DBMS_WM.GetMultiWorkspaces FROM DUAL;
```

GetOpContext ファンクション

現行のセッションについて、現行の操作のコンテキストを戻します。

書式

```
DBMS_WM.GetOpContext() RETURN VARCHAR2;
```

パラメータ

なし。

使用上の注意

このファンクションは、次のいずれかの値を戻します。

- DML: ユーザーが起動したデータ操作言語（DML）により現在の操作が稼動しています。
- MERGE_REMOVE: 現在の操作は、remove_workspace パラメータを TRUE に設定した MergeWorkspace プロシージャ・コールまたは remove_data パラメータを TRUE に設定した MergeTable プロシージャ・コールによって開始されました。
- MERGE_NOREMOVE: 現在の操作は、remove_workspace パラメータを FALSE に設定した MergeWorkspace プロシージャ・コールまたは remove_data パラメータを FALSE に設定した MergeTable プロシージャ・コールによって開始されました。

この戻り値をユーザー定義トリガーで使用すると、現在の操作に基づいた適切なアクションを実行できます。

例

次の例は、現在の操作のコンテキストを表示します。

```
SELECT DBMS_WM.GetOpContext FROM DUAL;

GETOPCONTEXT
--------------------------------------------------------------------------------
DML
```

GetPrivs ファンクション

指定された作業領域に対してカレント・ユーザーに付与されているすべての権限を記載した、カンマで区切られたリストを戻します。

書式

```
DBMS_WM.GetPrivs(
   workspace  IN VARCHAR2) RETURN VARCHAR2;
```

パラメータ

表 80-18 GetPrivs ファンクションのパラメータ

パラメータ	説明
workspace	権限のリストを戻す作業領域の名前。大文字と小文字が区別されます。

使用上の注意

Workspace Manager の権限の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

例

次の例では、カレント・ユーザーが B_focus_2 作業領域に対して付与されている権限を表示しています。

```
SELECT DBMS_WM.GetPrivs ('B_focus_2') FROM DUAL;

DBMS_WM.GETPRIVS('B_FOCUS_2')
--------------------------------------------------------------------------------
ACCESS,MERGE,CREATE,REMOVE,ROLLBACK
```

GetSessionInfo プロシージャ

現行の作業領域およびセッション・コンテキストに関する情報を取得します。

書式

```
DBMS_WM.GetSessionInfo(
   workspace     OUT VARCHAR2,
   context       OUT VARCHAR2,
   context_type  OUT VARCHAR2);
```

パラメータ

表 80-19 GetSessionInfo プロシージャのパラメータ

パラメータ	説明
workspace	現行のセッションがある作業領域の名前。
context	作業領域での現行セッションのコンテキスト。LATEST、セーブポイント名、または 'DD-MON-YYYY HH24:MI:SS' 日付書式での特定時点のいずれかで表されます（詳細は、「使用上の注意」を参照）。
context_type	作業領域での現行セッションのコンテキスト・タイプ。具体的には、次のいずれかの値になります。LATEST（context が LATEST の場合）、SAVEPOINT（context がセーブポイント名の場合）または INSTANT（context が特定時点の場合）。

使用上の注意

このプロシージャは、セッションの場所（作業領域やコンテキスト）を認識する必要がある場合に役立ちます。たとえば、GotoWorkspace プロシージャ、GotoSavepoint プロシージャおよび GotoDate プロシージャ操作を組み合せて実行した後に使用します。

このプロシージャが正常に実行されると、context パラメータには次のいずれかの値が格納されます。

- LATEST: セッションは現在、LATEST の論理セーブポイントにあり（詳細は『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）、作業領域で行われた変更を認識できます。セッションが GotoWorkspace プロシージャを使用して作業領域に入ると、コンテキストは、LATEST に自動的に設定されます。
- セーブポイント名 : セッションは現在、作業領域のセーブポイントにあります。セッションでは、最新バージョンの作業領域で行われた変更を認識できませんが、セーブポイント作成時点でのデータの静的ビューを認識できます。セッション・コンテキストは、GotoSavepoint プロシージャのコール後、セーブポイント名に設定されます。
- 特定時点 : セッションは現在、特定の時点にあります。セッションでは、最新バージョンの作業領域で行われた変更を認識できませんが、特定の時点でのデータの静的ビューを認識できます。セッション・コンテキストは、GotoDate プロシージャのコール後、ある特定時点に設定されます。

セッション・コンテキストの詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

例

次の例では、現行の作業領域とセッションのコンテキストに関する情報を取得して表示します。

```
DECLARE
  current_workspace VARCHAR2(30);
  current_context VARCHAR2(30);
  current_context_type VARCHAR2(30);
BEGIN
  DBMS_WM.GetSessionInfo(current_workspace,
                         current_context,
                         current_context_type);
  DBMS_OUTPUT.PUT_LINE('Session currently in workspace: ' ||current_workspace);
  DBMS_OUTPUT.PUT_LINE('Session context is: ' ||current_context);
  DBMS_OUTPUT.PUT_LINE('Session context is on: ' ||current_context_type);
END;
/
Session currently in workspace: B_focus_2
Session context is: LATEST
Session context is on: LATEST

PL/SQL procedure successfully completed.
```

GetWorkspace ファンクション

セッションの現行の作業領域を戻します。

書式

```
DBMS_WM.GetWorkspace() RETURN VARCHAR2;
```

パラメータ

なし。

使用上の注意

なし。

例

次の例は、セッションの現行の作業領域を表示します。

```
SELECT DBMS_WM.GetWorkspace FROM DUAL;

GETWORKSPACE
--------------------------------------------------------------------------------
B_focus_2
```

GotoDate プロシージャ

現行の作業領域において指定された日付および時刻の時点またはその近くに移動します。

構文

```
DBMS_WM.GotoDate(
   in_date  IN DATE);
```

パラメータ

表 80-20 GotoDate プロシージャのパラメータ

パラメータ	説明
in_date	作業領域の読取り専用ビューの日付および時刻（詳細は、「使用上の注意」を参照）。

使用上の注意

現行の作業領域または指定した日付および時刻に近い読取り専用ビューが表示されます。正確な時刻ポイントは、バージョン対応表のデータへの変更を追跡する履歴オプションが EnableVersioning プロシージャによって設定されているか、SetWoOverwriteOFF プロシージャまたは SetWoOverwriteON プロシージャによって変更されているかにより異なります。

- NONE: 読取り専用ビューは、in_date の後の最初のセーブポイントを反映します。
- VIEW_W_OVERWRITE: 読取り専用ビューには、in_date で有効なデータ値が反映されます。ただし、in_date が 2 つのセーブポイントの間にあり、その 2 つのセーブポイントの間でデータが変更された場合を除きます。この場合、セーブポイント間で変更されたデータは、空または以前の値を保持したまま表示されます。データが完全で正確なビューを表示するには、表をバージョン対応にするとき、VIEW_WO_OVERWRITE 履歴オプションを指定します。
- VIEW_WO_OVERWRITE: 読取り専用ビューは、in_date で有効なデータ値を反映します。

履歴オプションの説明は、EnableVersioning プロシージャの hist パラメータの説明を参照してください。

次の例は、VIEW_WO_OVERWRITE の設定の効果を示しています。次の順序でイベントが発生するとします。

1. 行の MANAGER_NAME 値は Adams です。
2. SP1 セーブポイントが作成されます。
3. MANAGER_NAME 値が Baxter に変更されます。
4. ステップ 7 で in_date に指定された時刻ポイントが発生します。
5. MANAGER_NAME 値が Chang に変更されます（つまり、最初のセーブポイント以降および 2 番目のセーブポイントより前にある in_date の前および後の両方の値が変更されました）。
6. SP2 セーブポイントが作成されます。
7. ステップ 4 で時刻を in_date に指定することにより、GotoDate プロシージャ操作が実行されます。

前述の例で、有効な履歴オプションが VIEW_WO_OVERWRITE である場合、ステップ 7 の後の MANAGER_NAME 値は Baxter です。

GotoDate プロシージャは、ユーザーが作業領域内にいる間に実行してください。このプロシージャに関連付けられている明示的な権限はありません。

例

次の例では、2001 年 6 月 29 日の午前 0 時またはその前後に移動します。時刻は、現在有効になっている履歴オプションにより異なります。

```
EXECUTE DBMS_WM.GotoDate ('29-JUN-01');
```

GotoSavepoint プロシージャ

現行の作業領域における指定したセーブポイントに移動します。

構文

```
DBMS_WM.GotoSavePoint(
   [savepoint_name  IN VARCHAR2 DEFAULT 'LATEST']);
```

パラメータ

表 80-21 GotoSavepoint プロシージャのパラメータ

パラメータ	説明
savepoint_name	セーブポイント名。大文字と小文字が区別されます。savepoint_name が指定されていない場合は、LATEST がデフォルトです。

使用上の注意

セーブポイント作成時の作業領域の読取り専用ビューが表示されます。RollbackToSP プロシージャをコールして特定のセーブポイントへのロールバックを実行し、そのセーブポイントより前の行をすべて削除する前に、このプロシージャを使用して別のセーブポイントから作業領域を調べる際に便利です。

この操作は、ユーザーが作業領域内にいる間に実行できます。この操作に関連付けられている明示的な権限はありません。

セーブポイントにロールバックしない場合は、NULL パラメータを使用して GotoSavepoint プロシージャをコールし、作業領域で現在アクティブなバージョンに移動できます （この操作を実行すると、GotoWorkspace プロシージャをコールして作業領域を指定した場合と同じ結果が得られます)。

LATEST セーブポイントなどのセーブポイントの詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

例

次の例は、Savepoint1 というセーブポイントに移動しています。

```
EXECUTE DBMS_WM.GotoSavepoint ('Savepoint1');
```

GotoWorkspace プロシージャ

現行のセッションを指定された作業領域に移動します。

構文

```
DBMS_WM.GotoWorkspace(
   workspace  IN VARCHAR2);
```

パラメータ

表 80-22 GotoWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。

使用上の注意

ユーザーが作業領域に移動した後、移動先でデータを変更できます。

LIVE データベースに移動するには、作業領域に LIVE を指定します。ユーザーが作業領域にいる場合に禁止されている操作が多いため、LIVE 作業領域に移動してから、作成した作業領域への操作を実行する方法が便利です。

次の条件が 1 つ以上該当する場合、例外が発生します。

- workspace が存在しない。
- ユーザーが作業領域に対し、ACCESS_WORKSPACE の権限を付与されていない。
- workspace が NO_ACCESS モードでアクセス制限されている（FreezeWorkspace プロシージャを参照）。

例

次の例では、NEWWORKSPACE 作業領域にユーザーを含めます。ユーザーは作業領域の最新バージョンで作業を開始します。

```
EXECUTE DBMS_WM.GotoWorkspace ('NEWWORKSPACE');
```

次の例では、LIVE データベース作業領域にユーザーを含めます。デフォルトでは、ユーザーがデータベースに接続すると、この作業領域に配置されます。

```
EXECUTE DBMS_WM.GotoWorkspace ('LIVE');
```

GrantSystemPriv プロシージャ

システムレベルの権限（特定の作業領域に制限されない）をユーザーおよびロールに付与します。grant_option パラメータを使用すると、権限を付与されたユーザーおよびロールが、他のユーザーおよびロールに対して指定した権限を付与できるようになります。

構文

```
DBMS_WM.GrantSystemPriv(
   priv_types       IN VARCHAR2,
   grantee          IN VARCHAR2
   [, grant_option  IN VARCHAR2 DEFAULT 'NO']
   [, auto_commit   IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-23 GrantSystemPriv プロシージャのパラメータ

パラメータ	説明
priv_types	権限を表す 1 つ以上のキーワードで構成された文字列 （Workspace Manager の権限については、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。権限のキーワードは、カンマで区切られます。使用可能なキーワードは、ACCESS_ANY_WORKSPACE、MERGE_ANY_WORKSPACE、CREATE_ANY_WORKSPACE、REMOVE_ANY_WORKSPACE、ROLLBACK_ANY_WORKSPACE および FREEZE_ANY_WORKSPACE です。
grantee	priv_types を付与するユーザー名（PUBLIC ユーザー・グループの指定も可能）またはロール。
grant_option	権限受領者が権限付与オプションを使用できるようにするには YES、使用できないようにするには NO（デフォルト）を指定します。権限付与オプションを使用すると、grantee が他のユーザーおよびロールに対し priv_types で指定された権限を付与できるようになります。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

このプロシージャを GrantWorkspacePriv プロシージャと対比してください。GrantWorkspacePriv プロシージャは、何も含まないキーワードを使用して作業領域レベルで Workspace Manager の権限を付与し、workspace パラメータを所有しています。

ユーザーが複数のソースから権限を取得し、いずれかのソースにその権限の付与オプションが設定されている場合、ユーザーにはその権限の付与オプションが設定されます。たとえば、SCOTT というユーザーに ACCESS_ANY_WORKSPACE 権限が付与され、grant_option が NO に指定されているとします。しかし、PUBLIC ユーザー・グループには ACCESS_ANY_WORKSPACE 権限が付与され、grant_option が YES に指定されている場合、SCOTT というユーザーが PUBLIC のメンバーであるため、SCOTT には付与オプションで ACCESS_ANY_WORKSPACE 権限が付与されています。

WM_ADMIN_ROLE ロールには、付与オプションで Workspace Manager のすべての権限が付与されます。WM_ADMIN_ROLE ロールは、自動的に DBA ロールに割り当てられます。

ACCESS_WORKSPACE または ACCESS_ANY_WORKSPACE 権限は、その他すべての Workspace Manager 権限に必要です。

システムレベル権限を取り消すには、RevokeSystemPriv プロシージャを使用します。

次の条件が 1 つ以上該当する場合、例外が発生します。

- grantee がデータベースで有効なユーザーまたはロールではない。
- priv_types を 付与する権限を持っていない。

例

次の例では、Smith というユーザーにデータベース内の作業領域へのアクセスを許可しています。ただし、Smith は他のユーザーに ACCESS_ANY_WORKSPACE 権限を付与できません。

```
EXECUTE DBMS_WM.GrantSystemPriv ('ACCESS_ANY_WORKSPACE', 'Smith', 'NO');
```

GrantWorkspacePriv プロシージャ

ユーザーおよびロールに作業領域レベルの権限を付与します。grant_option パラメータを使用すると、権限を付与されたユーザーおよびロールが、他のユーザーおよびロールに対して指定した権限を付与できるようになります。

構文

```
DBMS_WM.GrantWorkspacePriv(
   priv_types       IN VARCHAR2,
   workspace        IN VARCHAR2,
   grantee          IN VARCHAR2
   [, grant_option  IN VARCHAR2 DEFAULT 'NO']
   [, auto_commit   IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-24 GrantWorkspacePriv プロシージャのパラメータ

パラメータ	説明
priv_types	権限を表す 1 つ以上のキーワードで構成された文字列 （Workspace Manager の権限については、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照)。権限のキーワードは、カンマで区切られます。使用可能なキーワードは、ACCESS_WORKSPACE、MERGE_WORKSPACE、CREATE_WORKSPACE、REMOVE_WORKSPACE、ROLLBACK_WORKSPACE および FREEZE_WORKSPACE です。
workspace	作業領域名。大文字と小文字が区別されます。
grantee	priv_types を付与するユーザー名（PUBLIC ユーザー・グループの指定も可能）またはロール。
grant_option	権限受領者が権限付与オプションを使用できるようにするには YES、使用できないようにするには NO（デフォルト）を指定します。権限付与オプションを使用すると、grantee は、workspace で指定された作業領域において、priv_types で指定された権限を他のユーザーおよびロールに付与できるようになります。
auto_commit	ブール値（TRUE または FALSE)。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

このプロシージャを GrantSystemPriv プロシージャと対比してください。GrantSystemPriv プロシージャは、*xxx*_ANY_WORKSPACE（ACCESS_ANY_WORKSPACE、MERGE_ANY_WORKSPACE など）の形式のキーワードを持つシステムレベルの Workspace Manager 権限を付与します。

ユーザーが複数のソースから権限を取得し、いずれかのソースにその権限の付与オプションが設定されている場合、ユーザーにはその権限の付与オプションが設定されます。たとえば、SCOTT というユーザーに ACCESS_WORKSPACE 権限が付与され、grant_option が NO に指定されているとします。しかし、このような場合でも、PUBLIC ユーザー・グループには ACCESS_WORKSPACE 権限が付与され、grant_option が YES に指定されている場合、SCOTT というユーザーが PUBLIC のメンバーであるため、SCOTT には付与オプションで ACCESS_WORKSPACE 権限が付与されています。

WM_ADMIN_ROLE ロールには、付与オプションで Workspace Manager のすべての権限が付与されます。WM_ADMIN_ROLE ロールは、自動的に DBA ロールに割り当てられます。

ACCESS_WORKSPACE または ACCESS_ANY_WORKSPACE 権限は、その他すべての Workspace Manager 権限に必要です。

作業領域レベルの権限を取り消すには、RevokeWorkspacePriv プロシージャを使用します。

次の条件が 1 つ以上該当する場合、例外が発生します。

- grantee がデータベースで有効なユーザーまたはロールではない。
- priv_types を付与する権限を持っていない。

例

次の例では、Smith というユーザーに NEWWORKSPACE 作業領域へのアクセスおよびその作業領域内での変更のマージを許可しています。また、Smith は、NEWWORKSPACE において、この 2 種類の権限を他のユーザーに付与できます。

```
DBMS_WM.GrantWorkspacePriv ('ACCESS_WORKSPACE, MERGE_WORKSPACE', 'NEWWORKSPACE',
'Smith', 'YES');
```

IsWorkspaceOccupied ファンクション

作業領域にアクティブなセッションがあるかどうかをチェックします。

構文

```
DBMS_WM.IsWorkspaceOccupied(
   workspace  IN VARCHAR2) RETURN VARCHAR2;
```

パラメータ

表 80-25 IsWorkspaceOccupied ファンクションのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。

使用上の注意

このファンクションは、作業領域にアクティブなセッションがある場合は YES、アクティブなセッションがない場合は NO を戻します。

LIVE 作業領域が指定されている場合またはユーザーに作業領域へのアクセス権限が付与されていない場合には、例外が発生します。

例

次の例は、B_focus_2 作業領域にセッションがあるかどうかをチェックしています。

```
SELECT DBMS_WM.IsWorkspaceOccupied('B_focus_2') FROM DUAL;

DBMS_WM.ISWORKSPACEOCCUPIED('B_FOCUS_2')
--------------------------------------------------------------------------------
YES
```

LockRows プロシージャ

指定された表におけるバージョン化された行およびその親作業領域の対応行に対するアクセスを制御します。

構文

```
DBMS_WM.LockRows(
   workspace        IN VARCHAR2,
   table_name       IN VARCHAR2
   [, where_clause  IN VARCHAR2 DEFAULT '']
   [, lock_mode     IN VARCHAR2 DEFAULT 'E']);
```

パラメータ

表 80-26 LockRows プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。作業領域から表示可能な行の最新バージョンがロックされます。この作業領域で行が変更されていない場合は、ロックされたバージョンを祖先の作業領域に入れることができます。大文字と小文字が区別されます。
table_name	行がロックされる表の名前。大文字と小文字は区別されません。
where_clause	ロックされる行を識別する WHERE 句（WHERE キーワードを除く）。 例 :'department_id = 20' WHERE 句で指定できるのは、主キーの列のみです。WHERE 句に副問合せを含めることはできません。 where_clause が指定されていない場合、table_name のすべての行がロックされます。
lock_mode	ロックに設定されるモード。E（排他的）または S（共有）のいずれかです。デフォルトは E です。

使用上の注意

このプロシージャは、標準の Oracle サーバーのロックに加えて発生する Workspace Manager のロックに影響を与えます。Workspace Manager のロックの説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

このプロシージャは、Workspace Manager のロックの設定には影響を与えません。この設定がオンまたはオフにされるかどうかは、SetLockingON プロシージャおよび SetLockingOFF プロシージャにより決定されます。

行のロックを解除するには、UnlockRows プロシージャを使用します。

例

次の例では、NEWWORKSPACE 作業領域で last_name = 'Smith' である EMPLOYEES 表の行をロックしています。

```
EXECUTE DBMS_WM.LockRows ('NEWWORKSPACE', 'employees', 'last_name = ''Smith''');
```

MergeTable プロシージャ

作業領域の表（すべての行または WHERE 句で指定した行）への変更をその親作業領域に適用します。

構文

```
DBMS_WM.MergeTable(
   workspace            IN VARCHAR2,
   table_id             IN VARCHAR2
   [, where_clause      IN VARCHAR2 DEFAULT '']
   [, create_savepoint  IN BOOLEAN DEFAULT FALSE]
   [, remove_data       IN BOOLEAN DEFAULT FALSE]
   [, auto_commit       IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-27 MergeTable プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
table_id	親作業領域にマージする行を含む表の名前。大文字と小文字は区別されません。
where_clause	親作業領域にマージされる行を識別する WHERE 句（WHERE キーワードを除く）。例 :'department_id = 20'
	WHERE 句で指定できるのは、主キーの列のみです。WHERE 句に副問合せを含めることはできません。
	where_clause が指定されていない場合、table_name のすべての行がマージされます。
create_savepoint	ブール値（TRUE または FALSE）。
	TRUE の場合、マージ操作の前に親作業領域に暗黙的セーブポイントが作成されます （暗黙的および明示的セーブポイントの説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。
	FALSE（デフォルト）の場合、マージ操作の前に親作業領域に暗黙的セーブポイントが作成されません。

表 80-27 MergeTable プロシージャのパラメータ（続き）

パラメータ	説明
remove_data	ブール値（TRUE または FALSE）。 TRUE の場合は、子作業領域の表のデータ（where_clause により指定）が削除されます。このオプションは、workspace に子作業領域がない場合（つまり、リーフ作業領域の場合）のみ指定できます。 FALSE（デフォルト）の場合、子作業領域の表のデータは削除されません。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

workspace で table_name バージョン対応表の where_clause を満たすデータはすべて、workspace の親作業領域に適用されます。

マージされている行が保持するロックはすべて、解放されます。

マージされている作業領域とその親作業領域との間で競合が発生する場合は、マージ操作が失敗するため、<table_name>_CONF ビューを使用して手動で競合を解消する必要があります （競合の解消は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています）。

LIVE 作業領域には親作業領域がないため、表をマージできません。

オープンしているデータベース・トランザクションが表に影響を与える場合は、表をマージまたはリフレッシュできません。

table_id へのアクセス権、workspace への MERGE_WORKSPACE 権限または MERGE_ANY_WORKSPACE 権限がユーザーに付与されていない場合は、例外が発生します。

例

次の例では、NEWWORKSPACE で last_name = 'Smith' である EMP 表（USER3 スキーマ内）の変更を親作業領域にマージしています。

```
EXECUTE DBMS_WM.MergeTable ('NEWWORKSPACE', 'user3.emp', 'last_name = ''Smith''');
```

MergeWorkspace プロシージャ

作業領域で行われた変更をすべてその親作業領域に適用します。またオプションで、作業領域を削除します。

構文

```
DBMS_WM.MergeWorkspace(
   workspace            IN VARCHAR2
   [, create_savepoint  IN BOOLEAN DEFAULT FALSE]
   [, remove_workspace  IN BOOLEAN DEFAULT FALSE]
   [, auto_commit       IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-28 MergeWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
create_savepoint	ブール値（TRUE または FALSE）。 TRUE の場合、マージ操作の前に親作業領域に暗黙的セーブポイントが作成されます （暗黙的および明示的セーブポイントの説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。 FALSE（デフォルト）の場合、マージ操作の前に親作業領域に暗黙的セーブポイントが作成されません。
remove_workspace	ブール値（TRUE または FALSE）。 TRUE の場合は、マージ操作の後に workspace が削除されます。 FALSE（デフォルト）の場合は、マージ操作の後に workspace が削除されません。つまり、作業領域はそのまま存続します。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

workspace のすべてのバージョン対応表のすべてのデータが workspace の親作業領域にマージされます。このとき、remove_workspace が TRUE に設定されていると、workspace が削除されます。

プロシージャが実行されている間、現在の作業領域は NO_ACCESS モードにアクセス制限され、親作業領域は READ_ONLY モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

マージされている作業領域とその親作業領域との間で競合が発生する場合は、マージ操作が失敗するため、<table_name>_CONF ビューを使用して手動で競合を解消する必要があります （競合の解消は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています)。

remove_workspace パラメータ値が TRUE の場合は、マージされる作業領域がリーフ作業領域であることが必要です。つまり、その作業領域に子作業領域がある場合はマージできません （作業領域の階層の説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください)。

ユーザーが workspace への MERGE_WORKSPACE 権限または MERGE_ANY_WORKSPACE 権限を付与されていない場合は、例外が発生します。

例

次の例では、NEWWORKSPACE の変更をその親作業領域にマージし、NEWWORKSPACE を削除（デフォルト）しています。

```
EXECUTE DBMS_WM.MergeWorkspace ('NEWWORKSPACE');
```

RecoverAllMigratingTables プロシージャ

Workspace Manager の移行手順が失敗して一貫性のない状態になっているすべての表で、移行プロセスの完了を試行します。

構文

```
DBMS_WM.RecoverAllMigratingTables(
   [, ignore_last_error IN BOOLEAN DEFAULT FALSE]);
```

パラメータ

表 80-29 RecoverAllMigratingTables プロシージャのパラメータ

パラメータ	説明
ignore_last_error	ブール値（TRUE または FALSE）。 TRUE に設定すると、移行プロセスの間に発生した最新のエラー（ある場合）を無視します。最新のエラーに関する情報は、USER_WM_VT_ERRORS および ALL_WM_VT_ERRORS メタデータ・ビューに格納されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています（詳細は、「使用上の注意」を参照）。 FALSE（デフォルト）に設定すると、移行プロセスの間に発生した最新のエラー（ある場合）を無視しません。

使用上の注意

Workspace Manager の現行リリースをアップグレード中にエラーが発生した場合は、1 つ以上のバージョン対応表が一貫性のない状態のままである可能性があります（現行リリースのアップグレードについては、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。アップグレード手順に失敗した場合は、エラーの原因を修正（USER_WM_VT_ERRORS または ALL_WM_VT_ERRORS メタデータ・ビューを検証して SQL 文とエラー・メッセージを確認します）してから、デフォルトの ignore_last_error パラメータ値 FALSE を設定した RecoverMigratingTable プロシージャ（単一の表の場合）または RecoverAllMigratingTables プロシージャ（すべての表の場合）をコールして、アップグレード・プロセスの完了を試みます。

ただし、コールが再度失敗し、エラーの原因を修正できない場合、このエラーを無視することが安全かつ適切であることが確実であれば、ignore_last_error パラメータ値を TRUE に設定して RecoverMigratingTable プロシージャまたは RecoverAllMigratingTables プロシージャをコールし、エラーを無視できます。エラーを無視することが安全かつ適切であるかどうかは必ず確認してください。

例

次の例では、アップグレード手順に失敗して一貫性のない状態になっているすべてのバージョン対応表をリカバリします。

```
EXECUTE DBMS_WM.RecoverAllMigratingTables;
```

次の例では、アップグレード手順に失敗して一貫性のない状態になっているすべてのバージョン対応表をリカバリし、アップグレード手順の失敗原因となった最新のエラーを無視します。

```
EXECUTE DBMS_WM.RecoverAllMigratingTables(TRUE);
```

RecoverMigratingTable プロシージャ

Workspace Manager の移行手順が失敗して一貫性のない状態になっている単一の表で、移行プロセスの完了を試行します。

構文

```
DBMS_WM.RecoverMigratingTable(
   table_name           IN VARCHAR2
   [, ignore_last_error IN BOOLEAN DEFAULT FALSE]);
```

パラメータ

表 80-30 RecoverMigratingTable プロシージャのパラメータ

パラメータ	説明
table_name	移行時のエラーからリカバリするバージョン対応表の名前。大文字と小文字は区別されません。
ignore_last_error	ブール値（TRUE または FALSE）。 TRUE に設定すると、移行プロセスの間に発生した最新のエラー（ある場合）を無視します。最新のエラーに関する情報は、USER_WM_VT_ERRORS および ALL_WM_VT_ERRORS メタデータ・ビューに格納されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています （詳細は、「使用上の注意」を参照）。 FALSE（デフォルト）に設定すると、移行プロセスの間に発生した最新のエラー（ある場合）を無視しません。

使用上の注意

Workspace Manager の現行リリースをアップグレード中にエラーが発生した場合、1 つ以上のバージョン対応表が一貫性のない状態のままである可能性があります（現行リリースのアップグレードについては、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。アップグレード手順が失敗した場合、エラーの原因を修正し（USER_WM_VT_ERRORS または ALL_WM_VT_ERRORS メタデータ・ビューを検証して SQL 文とエラー・メッセージを確認します）、ignore_last_error パラメータ値を FALSE に設定した RecoverMigratingTable プロシージャ（単一の表の場合）または RecoverAllMigratingTables プロシージャ（すべての表の場合）をコールして、アップグレード・プロセスの完了を試みます。

ただし、コールが再度失敗し、エラーの原因を修正できない場合、このエラーを無視することが安全かつ適切であることが確実であれば、ignore_last_error パラメータ値を TRUE に設定して RecoverMigratingTable プロシージャまたは RecoverAllMigratingTables プロシージャをコールし、エラーを無視できます。エラーを無視することが安全かつ適切であるかどうかは必ず確認してください。

table_name が存在しない場合、またはバージョン対応でない場合は、例外が発生します。

例

次の例では、アップグレード手順失敗の原因となったエラーから COLA_MARKETING_BUDGET 表のリカバリを試みます。

```
EXECUTE DBMS_WM.RecoverMigratingTable('COLA_MARKETING_BUDGET');
```

次の例では、COLA_MARKETING_BUDGET 表のリカバリを試み、アップグレード手順失敗の原因となった最新のエラーを無視します。

```
EXECUTE DBMS_WM.RecoverMigratingTable('COLA_MARKETING_BUDGET', TRUE);
```

RefreshTable プロシージャ

親作業領域の表（すべての行または WHERE 句で指定された行）に行われた変更をすべて作業領域に適用します。

構文

```
DBMS_WM.RefreshTable(
   workspace        IN VARCHAR2,
   table_id         IN VARCHAR2
   [, where_clause  IN VARCHAR2 DEFAULT '']
   [, auto_commit   IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-31 RefreshTable プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
table_id	親作業領域の値を使用してリフレッシュする行を含む表の名前。大文字と小文字は区別されません。
where_clause	親作業領域からリフレッシュされる行を識別する WHERE 句（WHERE キーワードを除く）。例:'department_id = 20' WHERE 句で指定できるのは、主キーの列のみです。WHERE 句に副問合せを含めることはできません。 where_clause が指定されていない場合、table_name のすべての行がリフレッシュされます。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

このプロシージャは、workspace が作成された時点または最後にリフレッシュされた時点から、親作業領域の table_id バージョン対応表で where_clause を満たす行のすべての変更を workspace に適用します。

リフレッシュされている作業領域とその親作業領域との間で競合が発生する場合は、リフレッシュ操作が失敗するため、<table_name>_CONF ビューを使用して手動で競合を解消する必要があります （競合の解消は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています)。

LIVE 作業領域には親作業領域がないため、表をリフレッシュできません。

オープンしているデータベース・トランザクションが表に影響を与える場合は、表をマージまたはリフレッシュできません。

ユーザーが table_id へのアクセス権、workspace への MERGE_WORKSPACE 権限または MERGE_ANY_WORKSPACE 権限を付与されていない場合は、例外が発生します。

例

次の例では、親作業領域で last_name = 'Smith' である EMPLOYEES 表への変更を適用することにより、NEWWORKSPACE をリフレッシュしています。

```
EXECUTE DBMS_WM.RefreshTable ('NEWWORKSPACE', 'employees', 'last_name = ''Smith''');
```

RefreshWorkspace プロシージャ

親作業領域で行われた変更をすべて作業領域に適用します。

構文

```
DBMS_WM.RefreshWorkspace(
   workspace      IN VARCHAR2
   [, auto_commit IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-32 RefreshWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

このプロシージャは、workspace が作成された時点または最後にリフレッシュされた時点から親作業領域のバージョン対応表に加えられたすべての変更を workspace に適用します。

リフレッシュされている作業領域とその親作業領域との間で競合が発生する場合は、リフレッシュ操作が失敗するため、<table_name>_CONF ビューを使用して手動で競合を解消する必要があります （競合の解消は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています）。

指定された作業領域およびその親作業領域は、READ_ONLY モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

LIVE 作業領域には親作業領域がないため、表をリフレッシュできません。

ユーザーが workspace への MERGE_WORKSPACE 権限または MERGE_ANY_WORKSPACE 権限を付与されていない場合は、例外が発生します。

例

次の例では、親作業領域に加えられた変更を適用することにより、NEWWORKSPACE をリフレッシュしています。

```
EXECUTE DBMS_WM.RefreshWorkspace ('NEWWORKSPACE');
```

RelocateWriterSite プロシージャ

Workspace Manager のレプリケーション環境で、nonwriter サイトの 1 つを新規 writer サイトにします（古い writer サイトは nonwriter サイトになります）。

構文

```
DBMS_WM.RelocateWriterSite(
   newwritersite           IN VARCHAR2,
   oldwritersiteavailable  IN BOOLEAN);
```

パラメータ

表 80-33 RelocateWriterSite プロシージャのパラメータ

パラメータ	説明
newwritersite	Workspace Manager のレプリケーション環境での新規 writer サイトにする、現行の nonwriter サイトの名前（データベース・リンク）。
oldwritersiteavailable	ブール値（TRUE または FALSE）。 TRUE の場合は、古い writer サイトが更新され、writer サイトの変更が反映されます。 FALSE の場合、古い writer サイトは更新されず、writer サイトの変更は反映されません。この場合、古い writer サイトを使用可能にするには、SynchronizeSite プロシージャを使用する必要があります。

使用上の注意

このプロシージャを使用するには、レプリケーションの Workspace Manager オブジェクトへの適用方法を理解する必要があります。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。また、Oracle Replication の主要な概念と技術についても理解する必要があります。詳細は、『Oracle9*i* アドバンスト・レプリケーション』および『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

このプロシージャは、レプリケーション管理者として実行する必要があります。このプロシージャは、任意のマスター・サイトで実行できます。

現在、古い writer サイトが使用可能な場合は、oldwritersiteavailable パラメータを TRUE に指定する必要があります。oldwritersiteavailable パラメータを FALSE に指定した場合は、古い writer サイトが使用可能になった後に SynchronizeSite プロシージャを実行して、そのサイトを最新にする必要があります。

このプロシージャは、次の操作を実行します。

- oldwritersiteavailable が TRUE の場合は、古い writer サイトにあるすべてのバージョン対応表について、作業領域操作、DML 操作および DDL 操作を使用禁止にします。
- 新規の writer サイトにあるバージョン対応表について、作業領域操作、DML 操作および DDL 操作を使用可能にします。
- Replication API プロシージャを起動し、マスター・グループおよびすべてのバージョン対応表のマスター・グループについて、マスター定義サイトを newwritersite に再配置します。

例

次の例では、ある企業の Workspace Manager 環境の writer サイトを BACKUP-SITE1 に再配置します。

```
DBMS_WM.RelocateWriterSite(
    newwritersite           =>  'BACKUP-SITE1.ACME.COM');
    oldwritersiteavailable  =>  TRUE);
```

RemoveWorkspace プロシージャ

作業領域に関連付けられたすべての行バージョンを廃棄し、その作業領域も削除します。

構文

```
DBMS_WM.RemoveWorkspace(
   workspace       IN VARCHAR2
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-34 RemoveWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

RemoveWorkspace 操作は、リーフ作業領域（階層分岐の最下層作業領域）でのみ実行できます。データベース作業領域の階層の説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

削除する作業領域には、他のユーザーがいないようにしてください。

ユーザーが workspace への REMOVE_WORKSPACE 権限または REMOVE_ANY_WORKSPACE 権限を付与されていない場合は、例外が発生します。

例

次の例では、NEWWORKSPACE を削除しています。

```
EXECUTE DBMS_WM.RemoveWorkspace('NEWWORKSPACE');
```

RemoveWorkspaceTree プロシージャ

作業領域およびその子作業領域に関連付けられたすべての行バージョンを廃棄し、影響を受ける作業領域を削除します。

構文

```
DBMS_WM.RemoveWorkspaceTree(
   workspace       IN VARCHAR2
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-35 RemoveWorkspaceTree プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

RemoveWorkspaceTree 操作を実行すると、サポート構造が削除され、作業領域またはそのリーフ作業領域までのすべての子作業領域での変更がロールバックされるため、使用する場合は十分に注意してください。データベース作業領域の階層の説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

workspace またはその子作業領域には、他のユーザーがいないようにしてください。

ユーザーが workspace またはその子作業領域への REMOVE_WORKSPACE 権限を付与されていない場合は、例外が発生します。

例

次の例は、NEWWORKSPACE およびその子作業領域すべてを削除しています。

```
EXECUTE DBMS_WM.RemoveWorkspaceTree('NEWWORKSPACE');
```

ResolveConflicts プロシージャ

作業領域間の競合を解消します。

構文

```
DBMS_WM.ResolveConflicts(
   workspace     IN VARCHAR2,
   table_name    IN VARCHAR2,
   where_clause  IN VARCHAR2,
   keep          IN VARCHAR2);
```

パラメータ

表 80-36 ResolveConflicts プロシージャのパラメータ

パラメータ	説明
workspace	他の作業領域との競合の有無をチェックする作業領域の名前。大文字と小文字が区別されます。
table_name	競合をチェックする表の名前。大文字と小文字は区別されません。
where_clause	親作業領域からリフレッシュされる行を識別する WHERE 句（WHERE キーワードを除く）。例 :'department_id = 20' WHERE 句で指定できるのは、主キーの列のみです。WHERE 句に副問合せを含めることはできません。
keep	競合を解消する作業領域。PARENT、CHILD または BASE のいずれかです。 PARENT の場合、親作業領域の行が子作業領域にコピーされます。 CHILD の場合、子作業領域の行が即座に親作業領域にコピーされることはありません。ただし、競合は解消したとみなされ、子作業領域がマージされたときに子作業領域の行が親作業領域にコピーされます。 BASE の場合、ベース行が子作業領域にコピーされます。親作業領域にはコピーされません。ただし、競合は解消したとみなされ、子作業領域がマージされたときにベース行が親作業領域にコピーされます。ベース行が存在しない行の挿入競合の場合、BASE は無視されます。この場合、keep パラメータ値は PARENT または CHILD に指定する必要があります。

使用上の注意

このプロシージャは、table_name および where_clause により識別される条件をチェックし、workspace およびその親作業領域の行の値に競合が発生していないかどうかを確認します。このプロシージャは、keep パラメータで指定した、親作業領域または子作業領域の行の値を使用して競合を解消します。ただし、競合の解消を実行しても、トランザクションをコミット（標準のデータベース・コミット操作）し、CommitResolve プロシージャをコールして競合解消セッションを終了するまでは、実際にマージされません （競合解消のプロセスの概要を含めた詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。

たとえば、Department 20（DEPARTMENT_ID = 20）について、LIVE および Workspace1 作業領域で MANAGER_NAME が Tom であると想定します。この場合、次の操作が発生します。

1. Department 20 の manager_name が LIVE データベース作業領域で Tom から Mary に変更されます。
2. 変更がコミットされます（標準のデータベース・コミット操作）。
3. Department 20 の manager_name が Workspace1 で Tom から Franco に変更されます。
4. MergeWorkspace プロシージャがコールされ、Workspace1 の変更が LIVE 作業領域にマージされます。

   ただし、この時点では、Workspace1 の Department 20 の MANAGER_NAME（Franco が LIVE 作業領域で Mary と競合）に対して競合があるため、MergeWorkspace プロシージャをコールしても成功しません。
5. ResolveConflicts プロシージャのコールに使用するパラメータは、'Workspace1'、'department'、'department_id = 20'、'child' です。

   ステップ 7 の MergeWorkspace プロシージャ操作の後、Workspace1 および LIVE 作業領域の両方で MANAGER_NAME の値が Franco になります。
6. 変更がコミットされます（標準のデータベース・コミット操作）。
7. MergeWorkspace プロシージャがコールされ、Workspace1 の変更が LIVE 作業領域にマージされます。

競合解消の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

例

次の例では、Workspace1 の DEPARTMENT 表の行に関連する競合を解消しています。DEPARTMENT_ID は 20 です。また、子作業領域ではこの値を使用して関連する競合をすべて解消します。次に、トランザクションを最初にコミット（標準コミット）し、MergeWorkspace プロシージャをコールすることにより、競合解消の結果をマージします。

```
EXECUTE DBMS_WM.BeginResolve ('Workspace1');
EXECUTE  DBMS_WM.ResolveConflicts ('Workspace1', 'department', 'department_id = 20',
'child');
COMMIT;
EXECUTE DBMS_WM.CommitResolve ('Workspace1');
```

RevokeSystemPriv プロシージャ

ユーザーおよびロールからシステムレベルの権限を取り消し（削除）ます。

構文

```
DBMS_WM.RevokeSystemPriv(
   priv_types      IN VARCHAR2,
   grantee         IN VARCHAR2
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-37 RevokeSystemPriv プロシージャのパラメータ

パラメータ	説明
priv_types	権限を表す 1 つ以上のキーワードで構成された文字列 （Workspace Manager の権限については、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。権限のキーワードは、カンマで区切られます。使用可能なキーワードは、ACCESS_ANY_WORKSPACE、MERGE_ANY_WORKSPACE、CREATE_ANY_WORKSPACE、REMOVE_ANY_WORKSPACE および ROLLBACK_ANY_WORKSPACE です。
grantee	priv_types を取り消すユーザー名（PUBLIC ユーザー・グループの指定も可能）またはロール。

表 80-37 RevokeSystemPriv プロシージャのパラメータ（続き）

パラメータ	説明
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

このプロシージャを RevokeWorkspacePriv プロシージャと対比してください。RevokeWorkspacePriv プロシージャは、*xxx*_WORKSPACE（ACCESS_WORKSPACE、MERGE_WORKSPACE など）の形式のキーワードを持つ作業領域レベルの Workspace Manager 権限を取り消します。

システムレベル権限を付与するには、GrantSystemPriv プロシージャを使用します。

次の条件が 1 つ以上該当する場合、例外が発生します。

- grantee がデータベースで有効なユーザーまたはロールではない。
- grantee に対する priv_types の権限付与者でない。

例

次の例は、Smith というユーザーの作業領域へのアクセスおよび作業領域の変更のマージを取り消します。

```
EXECUTE DBMS_WM.RevokeSystemPriv ('ACCESS_ANY_WORKSPACE, MERGE_ANY_WORKSPACE',
'Smith');
```

RevokeWorkspacePriv プロシージャ

指定された作業領域のユーザーおよびロールから作業領域レベルの権限を取り消し（削除）ます。

構文

```
DBMS_WM.RevokeWorkspacePriv(
   priv_types      IN VARCHAR2,
   workspace       IN VARCHAR2,
   grantee         IN VARCHAR2
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-38 RevokeWorkspacePriv プロシージャのパラメータ

パラメータ	説明
priv_types	権限を表す 1 つ以上のキーワードで構成された文字列 （Workspace Manager の権限については、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。権限のキーワードは、カンマで区切られます。使用可能なキーワードは、ACCESS_WORKSPACE、MERGE_WORKSPACE、CREATE_WORKSPACE、REMOVE_WORKSPACE および ROLLBACK_WORKSPACE です。
workspace	作業領域名。大文字と小文字が区別されます。
grantee	priv_types を取り消すユーザー名（PUBLIC ユーザー・グループの指定も可能）またはロール。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

このプロシージャを RevokeSystemPriv プロシージャと対比してください。RevokeSystemPriv プロシージャは、*xxx*_ANY_WORKSPACE（ACCESS_ANY_WORKSPACE、MERGE_ANY_WORKSPACE など）の形式のキーワードを持つシステムレベルの Workspace Manager 権限を取り消します。

作業領域レベルの権限を付与するには、GrantWorkspacePriv プロシージャを使用します。

次の条件が 1 つ以上該当する場合、例外が発生します。

- grantee がデータベースで有効なユーザーまたはロールではない。
- grantee に対する priv_types の権限付与者でない。

例

次の例では、Smith というユーザーの NEWWORKSPACE 作業領域へのアクセスおよび作業領域の変更のマージを取り消します。

```
EXECUTE DBMS_WM.RevokeWorkspacePriv ('ACCESS_WORKSPACE, MERGE_WORKSPACE',
'NEWWORKSPACE', 'Smith');
```

RollbackDDL プロシージャ

指定した表の DDL（データ定義言語）セッション中に加えられた DDL 変更をロールバック（取消し）して、その DDL セッションを終了します。

構文

```
DBMS_WM.RollbackDDL(
   table_name  IN VARCHAR2);
```

パラメータ

表 80-39 RollbackDDL プロシージャのパラメータ

パラメータ	説明
table_name	バージョン対応表の名前。大文字と小文字は区別されません。

使用上の注意

このプロシージャは、DDL セッション中に、バージョン対応表に加えられた変更、およびバージョン対応表に基づく任意の索引とトリガーに加えられた変更をロールバック（取消し）します。また、このプロシージャは、BeginDDL プロシージャで作成された特殊な *<table-name>_LTS* 表（スケルトン表）を削除します。

バージョン対応表に関連した DDL 操作の実行方法、および Oracle Replication 環境でのバージョン対応表に対する DDL 操作の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

次の条件が 1 つ以上該当する場合、例外が発生します。

- table_name が存在しない、またはバージョン対応でない。
- table_name について、オープンした DDL セッションが存在しない（つまり、この表を指定して BeginDDL プロシージャがコールされていない、または CommitDDL プロシージャか RollbackDDL プロシージャがこの表を指定してコールされている）。

例

次の例では、DDL セッションを開始し、COLA_MARKETING_BUDGET_LTS という名前のスケルトン表を使用して COMMENTS 列を COLA_MARKETING_BUDGET 表に追加し、変更を取り消して DDL セッションを終了します。

```
EXECUTE DBMS_WM.BeginDDL('COLA_MARKETING_BUDGET');
ALTER TABLE cola_marketing_budget_lts ADD (comments VARCHAR2(100));
EXECUTE DBMS_WM.RollbackDDL('COLA_MARKETING_BUDGET');
```

RollbackResolve プロシージャ

競合解消セッションを終了し、BeginResolve プロシージャが実行された後に作業領域に行われた変更をすべて廃棄します。

構文

```
DBMS_WM.RollbackResolve(
   workspace  IN VARCHAR2);
```

パラメータ

表 80-40 RollbackResolve プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。

使用上の注意

このプロシージャは、BeginResolve プロシージャで開始された現行の競合解消セッションを終了し、競合解消セッションが開始された後に作業領域で行われた変更をすべて廃棄します。このプロシージャを、変更をすべて保存する CommitResolve プロシージャと対比してください。

競合解消セッションがロールバックされている間、作業領域は 1WRITER モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

競合解消の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

次の条件が 1 つ以上該当する場合、例外が発生します。

- workspace で 1 つ以上のデータベース・トランザクションがオープンしている。
- WM_ADMIN_ROLE ロールを付与されていないユーザー、または workspace で BeginResolve プロシージャを実行しなかったユーザーによりプロシージャがコールされた。

例

次の例では、Workspace1 で競合解消セッションを終了し、変更をすべて廃棄します。

```
EXECUTE  DBMS_WM.RollbackResolve ('Workspace1');
```

RollbackTable プロシージャ

作業領域において指定した表（すべての行または WHERE 句で指定された行）に対して行われた変更をすべて廃棄します。

構文

```
DBMS_WM.RollbackTable(
   workspace        IN VARCHAR2,
   table_id         IN VARCHAR2,
   [, sp_name       IN VARCHAR2 DEFAULT '']
   [, where_clause  IN VARCHAR2 DEFAULT '']
   [, remove_locks  IN BOOLEAN DEFAULT TRUE]
   [, auto_commit   IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-41 RollbackTable プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
table_id	廃棄する行を含む表の名前。大文字と小文字は区別されません。
sp_name	ロールバックするセーブポイントの名前。大文字と小文字が区別されます。デフォルトでは、すべての変更が廃棄されます（つまり、セーブポイントがすべて無視されます）。
where_clause	廃棄される行を識別する WHERE 句（WHERE キーワードを除く）。 例:'department_id = 20' WHERE 句で指定できるのは、主キーの列のみです。WHERE 句に副問合せを含めることはできません。 where_clause が指定されていない場合、他のパラメータの条件に一致する行はすべて廃棄されます。
remove_locks	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、where_clause での条件を満たす親作業領域および子作業領域がバージョニングされていない親作業領域の行のロックが解放されます。sp_name パラメータでセーブポイントが指定されている場合、このオプションは無効になります。 FALSE の場合、親作業領域のロックが一切解放されません。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

指定したセーブポイントの後に暗黙的セーブポイントが作成されている場合は、暗黙的セーブポイントを作成した子作業領域を最初にマージまたは削除しないかぎり、セーブポイントにロールバックできません。

次の条件が 1 つ以上該当する場合、例外が発生します。

- workspace が存在しない。
- workspace または影響を受ける表にロールバックする権限を付与されていない。
- table_id に影響を与えるデータベース・トランザクションが workspace でオープンしている。

例

次の例では、NEWWORKSPACE が作成された後に加えられた EMP 表（USER3 スキーマ内）への変更をすべてロールバックしています。

```
EXECUTE DBMS_WM.RollbackTable ('NEWWORKSPACE', 'user3.emp');
```

RollbackToSP プロシージャ

指定されたセーブポイント以降、作業領域でバージョン対応表に対して行われたデータ変更をすべて廃棄します。

構文

```
DBMS_WM.RollbackToSP(
   workspace       IN VARCHAR2,
   savepoint_name  IN VARCHAR2
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-42 RollbackToSP プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
savepoint_name	ロールバックを変更するセーブポイントの名前。大文字と小文字が区別されます。

表 80-42 RollbackToSP プロシージャのパラメータ（続き）

パラメータ	説明
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

このプロシージャを実行している間、作業領域は NO_ACCESS モードにアクセス制限されます。

このプロシージャを、作業領域を作成した後に加えられたすべての変更をロールバックする RollbackWorkspace プロシージャと対比してください。

指定したセーブポイントの後に暗黙的セーブポイントが作成されている場合は、暗黙的セーブポイントを作成した子作業領域を最初にマージまたは削除しないかぎり、セーブポイントにロールバックできません。

次の条件が 1 つ以上該当する場合、例外が発生します。

- workspace が存在しない。
- savepoint_name が存在しない。
- savepoint_name の後、workspace に 1 つ以上の暗黙的セーブポイントが作成されており、暗黙的セーブポイントを作成させた子作業領域がまだ存在している。
- workspace または影響を受ける表にロールバックする権限を付与されていない。
- workspace にセッションがある。

例

次の例では、Savepoint1 が作成された後、NEWWORKSPACE 作業領域ですべての表に加えられたすべての変更をロールバックしています。

```
EXECUTE DBMS_WM.RollbackToSP ('NEWWORKSPACE', 'Savepoint1');
```

RollbackWorkspace プロシージャ

作業領域でバージョン対応表に対して行われたデータ変更をすべて廃棄します。

構文

```
DBMS_WM.RollbackWorkspace(
   workspace       IN VARCHAR2
   [, auto_commit  IN BOOLEAN DEFAULT TRUE]);
```

パラメータ

表 80-43 RollbackWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。
auto_commit	ブール値（TRUE または FALSE）。 TRUE（デフォルト）の場合、終了後にコミットされる自律型のデータベース・トランザクションとして操作が実行されます。 FALSE の場合、コール元がオープンしているデータベース・トランザクションが存在すると、その一部として実行されます。データベース・トランザクションがオープンされていない場合、新しいデータベース・トランザクションで操作が実行されます。いずれの場合も、コール元がトランザクションをコミットします。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

使用上の注意

ロールバックできるのは、リーフ作業領域のみです。つまり、子作業領域を持つ作業領域はロールバックできません （作業領域の階層の説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。

このプロシージャを、指定されたセーブポイントに変更をロールバックする RollbackToSP プロシージャと対比してください。

RemoveWorkspace プロシージャと同様、RollbackWorkspace は作業領域のデータを削除します。また、RemoveWorkspace プロシージャとは異なり、Workspace Manager の作業領域構造を削除しません。

このプロシージャの実行中、指定した作業領域は NO_ACCESS モードにアクセス制限されます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

次の条件が 1 つ以上該当する場合、例外が発生します。

- workspace に子作業領域がある。
- workspace が存在しない。
- workspace または影響を受ける表にロールバックする権限を付与されていない。
- workspace にアクティブなセッションがある。

例

次の例では、NEWWORKSPACE 作業領域が作成された後に加えられたすべての変更をロールバックしています。

```
EXECUTE DBMS_WM.RollbackWorkspace ('NEWWORKSPACE');
```

SetConflictWorkspace プロシージャ

作業領域とその親との間に競合が発生しているかどうかを判別します。

構文

```
DBMS_WM.SetConflictWorkspace(
   workspace  IN VARCHAR2);
```

パラメータ

表 80-44 SetConflictWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。

使用上の注意

このプロシージャは、workspace とその親作業領域との間の競合をチェックし、必要に応じて、<table_name>_CONF ビュー（『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）の内容を変更します。

作業領域内で変更されたすべての表に対し、<table_name>_CONF ビューから SELECT 操作を実行すると、その作業領域で親作業領域と競合する行がすべて表示されます。（現行の競合作業領域の設定に対する競合がある表のリストを取得するには、SQL 文の SELECT * FROM ALL_WM_VERSIONED_TABLES WHERE conflict = 'YES'; を使用します。SQL 文 SELECT * FROM <table_name>_CONF を実行すると、現行の作業領域とその親作業領域との <table_name> の競合が表示されます。）

作業領域をマージまたはリフレッシュするには、最初に競合を解消する必要があります。競合を解消するには、ResolveConflicts プロシージャを使用します（次に、MergeWorkspace プロシージャを使用して解消結果をマージします）。

例

次の例では、B_focus_2 とその親作業領域との競合をチェックし、必要に応じて <table_name>_CONF ビューの内容を変更しています。

```
EXECUTE DBMS_WM.SetConflictWorkspace ('B_focus_2');
```

SetDiffVersions プロシージャ

バージョン対応表における 2 つのセーブポイントおよび共通祖先（ベース）について、値の差異を検出します。このような差異を示す差異ビューの内容を変更します。

構文

```
DBMS_WM.SetDiffVersions(
   workspace1  IN VARCHAR2,
   workspace2  IN VARCHAR2);
```

または

```
DBMS_WM.SetDiffVersions(
   workspace1  IN VARCHAR2,
   savepoint1  IN VARCHAR2,
   workspace2  IN VARCHAR2,
   savepoint2  IN VARCHAR2);
```

パラメータ

表 80-45 SetDiffVersions プロシージャのパラメータ

パラメータ	説明
workspace1	バージョン対応表の差異を最初にチェックされる作業領域の名前。大文字と小文字が区別されます。
savepoint1	値をチェックする workspace1 のセーブポイントの名前。大文字と小文字が区別されます。 savepoint1 および savepoint2 が指定されていない場合、各作業領域ではバージョン対応表の行の LATEST セーブポイントがチェックされます（LATEST セーブポイントの説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。

表 80-45 SetDiffVersions プロシージャのパラメータ（続き）

パラメータ	説明
workspace2	バージョン対応表の差異を 2 番目にチェックされる作業領域の名前。大文字と小文字が区別されます。
savepoint2	値をチェックする workspace2 のセーブポイントの名前。大文字と小文字が区別されます。

使用上の注意

このプロシージャは、差異ビュー（xxx_DIFF）の内容を変更します。差異ビューの詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。プロシージャをコールすると、3 つの行のうち 1 つ以上のセットが移入されます。各セットの構成は次のとおりです。

- 共通祖先の値
- workspace1 の値（savepoint1 または LATEST セーブポイントの値）
- workspace2 の値（savepoint2 または LATEST セーブポイントの値）

適切な xxx_DIFF ビューから行を選択し、2 つのセーブポイントおよびその共通祖先で比較可能な表の値をチェックできます。xxx_DIFF ビューの行では、共通祖先（またはベース）は DiffBase として識別されます。

例

次の例では、B_focus_1 および B_focus_2 作業領域のバージョン対応表の差異をチェックしています（出力は読みやすく再フォーマットされます）。

```
SQL> -- Add rows to difference view: COLA_MARKETING_BUDGET_DIFF
SQL> EXECUTE DBMS_WM.SetDiffVersions ('B_focus_1', 'B_focus_2');

SQL> -- View the rows that were just added.
SQL> SELECT * from COLA_MARKETING_BUDGET_DIFF;

PRODUCT_ID  PRODUCT_NAME  MANAGER  BUDGET  WM_DIFFVER          WMCODE
----------  ------------  -------  ------  -----------        --------
       1        cola_a    Alvarez   2      DiffBase           NC
       1        cola_a    Alvarez   1.5    B_focus_1, LATEST  U
       1        cola_a    Alvarez   2      B_focus_2, LATEST  NC
       2        cola_b    Burton    2      DiffBase           NC
       2        cola_b    Beasley   3      B_focus_1, LATEST  U
       2        cola_b    Burton    2.5    B_focus_2, LATEST  U
       3        cola_c    Chen      1.5    DiffBase           NC
       3        cola_c    Chen      1      B_focus_1, LATEST  U
       3        cola_c    Chen      1.5    B_focus_2, LATEST  NC
```

```
4        cola_d    Davis     3.5    DiffBase            NC
4        cola_d    Davis     3      B_focus_1, LATEST   U
4        cola_d    Davis     2.5    B_focus_2, LATEST   U

12 rows selected.
```

差異（xxx_DIFF）ビューの情報の解釈および使用方法は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』で説明されています。

SetLockingOFF プロシージャ

現行のセッションにおける Workspace Manager のロックを使用禁止にします。

構文

```
DBMS_WM.SetLockingOFF();
```

パラメータ

なし。

使用上の注意

このプロシージャは、SetLockingON プロシージャにより設定された Workspace Manager のロックを解除します。このセッションで適用された既存のロックは、解除されずにそのまま残ります。このセッションで加えられた変更はすべて、ロックされません。

例

次の例では、セッションのロック解除を設定しています。

```
EXECUTE DBMS_WM.SetLockingOFF;
```

SetLockingON プロシージャ

現行のセッションにおける Workspace Manager のロックを使用可能にします。

構文

```
DBMS_WM.SetLockingON(
   lockmode  IN VARCHAR2);
```

パラメータ

表 80-46 SetLockingON プロシージャのパラメータ

パラメータ	説明
lockmode	ロック・モード。E、S または C のいずれかです。 E（排他）モードは、前のバージョンの行および現行のバージョンで対応する行をロックします。どちらのバージョンでも、他のユーザーは作業領域で値を変更できません。 S（共有）モードは、前のバージョンの行および現行のバージョンで対応する行をロックします。ただし、現行のバージョンでは、他のユーザーは作業領域で値を変更できます。前のバージョンの作業領域では値を変更できません。 C（引継ぎ）モードは、現行の作業領域の行を前のバージョンで対応する行と同じロック・モードでロックします（前のバージョンで行がロックされていない場合は、現行のバージョンで対応する行もロックされません）。

使用上の注意

このプロシージャは、標準の Oracle サーバーのロックに加えて発生する Workspace Manager のロックに影響を与えます。Workspace Manager のロックを使用して、競合を回避できます。ユーザーが行をロックすると、親作業領域で対応する行もロックされます。したがって、この作業領域を親とマージするとき、この行は競合しないことが保証されます。

排他ロックの場合は、1 つ以上の列の異なる値がテストされる *what-if* シナリオを使用できません。したがって、排他ロックが有効ではない場合には、シナリオのテストについて検討してください。

ロックはユーザーのセッション・レベルで使用可能にされます。ロック・モードは、次のいずれかが発生するまで有効のままです。

- セッションが別の作業領域へ移動するか、別のデータベースに接続した場合。どちらの場合も、SetWorkspaceLockModeON プロシージャを使用して別のロック・モードが指定されている場合を除き、C（引継ぎ）に設定されます。
- セッションが SetLockingOFF プロシージャを実行した場合。

UnlockRows プロシージャによりロックが解除されないかぎり、作業領域の継続期間はロックの有効性が持続します（既存のロックは SetLockingOFF プロシージャによる影響を受けません）。

ロックに関連付けられている特定の権限はありません。作業領域に移動できるセッションはすべて、ロックを設定できます。

例

次の例では、セッションに排他ロックを設定しています。

```
EXECUTE DBMS_WM.SetLockingON ('E');
```

このユーザーがロックした行はすべて、作業領域がマージまたはロールバックされるまで、ロックされたままとなります。

SetMultiWorkspaces プロシージャ

指定された単数または複数の作業領域を、バージョン対応表の複数作業領域ビューで表示できるようにします。

構文

```
DBMS_WM.SetMultiWorkspaces(
   workspaces  IN VARCHAR2);
```

パラメータ

表 80-47 SetMultiWorkspaces プロシージャのパラメータ

パラメータ	説明
workspaces	複数作業領域ビューに情報が追加される 1 つ以上の作業領域（『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照）。作業領域名は大文字と小文字が区別されます。 複数（8 以下）の作業領域を指定するには、作業領域名をカンマで区切ります。たとえば、'workspace1,workspace2' のようにします。

使用上の注意

このプロシージャは、複数作業領域ビュー（xxx_MW）に行を追加します。これらのビューの内容および使用方法の詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

複数作業領域ビューで作業領域名を表示可能にするには、GetMultiWorkspaces ファンクションを使用します。

次の条件が 1 つ以上該当する場合、例外が発生します。

- ユーザーが workspaces という名前がつく作業領域に 1 つ以上移動する権限を付与されていない。
- workspaces という名前がつく作業領域が有効ではない。

例

次の例では、B_focus_1 作業領域でバージョン対応表の複数作業領域ビューに情報を追加しています。

```
SQL> EXECUTE DBMS_WM.SetMultiWorkspaces ('B_focus_1');
```

SetWoOverwriteOFF プロシージャ

EnableVersioning プロシージャまたは SetWoOverwriteON プロシージャにより使用可能にされた VIEW_WO_OVERWRITE 履歴オプションを使用禁止にし、VIEW_W_OVERWRITE（上書き）に変更します。

構文

```
DBMS_WM.SetWoOverwriteOFF();
```

パラメータ

なし。

使用上の注意

このプロシージャは、VIEW_WO_OVERWRITE オプションを VIEW_W_OVERWRITE に変更することにより、<table_name>_HIST という名前のビューの履歴情報の記録に影響を与えます。つまり、このプロシージャを使用すると、表の同じバージョンへの最新の変更のみがビューに表示されます。バージョンに対する変更の履歴は保持されません。つまり、同じバージョンの行に対する後続の変更が前の変更に上書きされます。

このプロシージャが影響を与えるのは、EnableVersioning プロシージャをコールしたときに VIEW_WO_OVERWRITE に設定された hist パラメータを使用してバージョン対応となった表のみです。

<table_name>_HIST ビューの説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。VIEW_WO_OVERWRITE および VIEW_W_OVERWRITE オプションの詳細は、「EnableVersioning プロシージャ」を参照してください。

履歴オプションは、GotoDate プロシージャの動作に影響を与えます。プロシージャの詳細は、「使用上の注意」を参照してください。

SetWoOverwriteOFF プロシージャを実行した場合、現行のセッションの間のみ効果が持続します。このプロシージャの効果を元に戻すには、SetWoOverwriteON プロシージャを使用します。

例

次の例では、VIEW_WO_OVERWRITE 履歴オプションを使用禁止にしています。

```
EXECUTE DBMS_WM.SetWoOverwriteOFF;
```

SetWoOverwriteON プロシージャ

SetWoOverwriteOFF プロシージャにより使用禁止にされた VIEW_WO_OVERWRITE 履歴オプションを使用可能にします。

構文

```
DBMS_WM.SetWoOverwriteON();
```

パラメータ

なし。

使用上の注意

このプロシージャは、VIEW_W_OVERWRITE オプションを VIEW_WO_OVERWRITE（上書きなし）に変更することにより、<table_name>_HIST という名前のビューの履歴情報の記録に影響を与えます。つまり、このプロシージャを使用すると、表の同じバージョンへのすべての変更がビューに表示されます。バージョンに対する変更の履歴が保持されます。つまり、同じバージョンの行に対して後続の変更が行われても、前の変更に上書きされません。

このプロシージャは、以前の SetWoOverwriteOFF プロシージャのコールにより影響を受けた表にのみ影響を与えます。

<table_name>_HIST ビューの説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。VIEW_WO_OVERWRITE および VIEW_W_OVERWRITE オプションの詳細は、「EnableVersioning プロシージャ」を参照してください。

CompressWorkspace プロシージャまたは CompressWorkspaceTree プロシージャを使用して compress_view_wo_overwrite パラメータを TRUE に指定して作業領域が圧縮されている場合は、VIEW_WO_OVERWRITE 履歴オプションを優先させることができます。

履歴オプションは、GotoDate プロシージャの動作に影響を与えます。プロシージャの詳細は、「使用上の注意」を参照してください。

このプロシージャの効果を元に戻すには、SetWoOverwriteOFF プロシージャを使用します。

例

次の例では、VIEW_WO_OVERWRITE 履歴オプションを使用可能にしています。

```
EXECUTE DBMS_WM.SetWoOverwriteON;
```

SetWorkspaceLockModeOFF プロシージャ

指定された作業領域における Workspace Manager のロックを使用禁止にします。

構文

```
DBMS_WM.SetWorkspaceLockModeOFF(
   workspace  IN VARCHAR2);
```

パラメータ

表 80-48 SetWorkspaceLockModeOFF プロシージャのパラメータ

パラメータ	説明
workspace	ロック・モードを解除する作業領域の名前。大文字と小文字が区別されます。

使用上の注意

このプロシージャは、SetWorkspaceLockModeON プロシージャにより設定された Workspace Manager のロックを解除します。このセッションで適用された既存のロックは、解除されずにそのまま残ります。このセッションまたは後続のセッションで加えられた変更はすべてロックされません。ただし、セッションで SetLockingON プロシージャを実行し、ロックを有効にした場合を除きます。

次のいずれかが発生した場合、例外が発生します。

- ユーザーに WM_ADMIN_ROLE ロールが割り当てられていない。または workspace の所有者ではない。
- workspace でデータベース・トランザクションがオープンしている。
- workspace が継続してリフレッシュされた作業領域である（CreateWorkspace プロシージャの isrefreshed パラメータの説明を参照）。

例

次の例では、NEWWORKSPACE という名前の作業領域のロック解除を設定しています。

```
EXECUTE DBMS_WM.SetWorkspaceLockModeOFF('NEWWORKSPACE');
```

SetWorkspaceLockModeON プロシージャ

指定された作業領域における Workspace Manager のロックを使用可能にします。

構文

```
DBMS_WM.SetWorkspaceLockModeON(
   workspace    IN VARCHAR2,
   lockmode     IN VARCHAR2
   [, override  IN BOOLEAN DEFAULT FALSE]);
```

パラメータ

表 80-49 SetWorkspaceLockModeON プロシージャのパラメータ

パラメータ	説明
workspace	Workspace Manager のロックを使用可能にする作業領域の名前。大文字と小文字が区別されます。
lockmode	行レベル・ロックのデフォルトのロック・モード。E、S または C のいずれかです。 E（排他）モードは、親作業領域の行および現行の作業領域で対応する行をロックします。どちらの作業領域でも、他のユーザーは作業領域で値を変更できません。 S（共有）モードは、親作業領域の行および現行の作業領域で対応する行をロックします。ただし、現行の作業領域では、他のユーザーは作業領域で値を変更できます。親作業領域では値を変更できません。 C（引継ぎ）モードは、現行の作業領域の行を親作業領域で対応する行と同じロック・モードでロックします（親作業領域で行がロックされていない場合は、子作業領域で対応する行もロックされません）。
override	ブール値（TRUE または FALSE）。 TRUE の場合、作業領域のセッションで SetLockingON プロシージャおよび SetLockingOFF プロシージャを使用することにより、lockmode の値を変更できます。 FALSE（デフォルト）の場合、作業領域のセッションでは lockmode の値を変更できません。

使用上の注意

このプロシージャは、標準の Oracle サーバーのロックに加えて発生する Workspace Manager のロックに影響を与えます。Workspace Manager のロックを使用して、競合を回避できます。ユーザーが行をロックすると、親作業領域で対応する行もロックされます。したがって、この作業領域を親とマージするとき、この行は競合しないことが保証されます。

排他ロックの場合は、1 つ以上の列の異なる値がテストされる *what-if* シナリオを使用できません。したがって、排他ロックが有効ではない場合には、シナリオのテストについて検討してください。

override パラメータ値が TRUE の場合、SetLockingON プロシージャおよび SetLockingOFF プロシージャをそれぞれ使用して、ユーザー・セッション・レベルでもロック設定の有効と無効を切り替えられます。

このセッションまたは後続のセッションで加えられた新しい変更はすべてロックされます。ただし、セッションで SetLockingOFF プロシージャを実行し、ロックを無効にした場合を除きます。

次のいずれかが発生した場合、例外が発生します。

- ユーザーに WM_ADMIN_ROLE ロールが割り当てられていない。または workspace の所有者ではない。
- workspace でデータベース・トランザクションがオープンしている。
- workspace が継続してリフレッシュされた作業領域である（CreateWorkspace プロシージャの isrefreshed パラメータの説明を参照）。

例

次の例では、NEWWORKSPACE という名前の作業領域に排他ロックを設定しています。

```
EXECUTE DBMS_WM.SetWorkspaceLockModeON ('NEWWORKSPACE', 'E');
```

ロックした行はすべて、作業領域がマージまたはロールバックされるまで、ロックされたままとなります。

SynchronizeSite プロシージャ

RelocateWriterSite プロシージャを使用した writer サイトの移動後に、Workspace Manager レプリケーション環境のローカル・サイト（古い writer サイト）を最新にします。

構文

```
DBMS_WM.SynchronizeSite(
   newwritersite  IN VARCHAR2);
```

パラメータ

表 80-50 SynchronizeSite プロシージャのパラメータ

パラメータ	説明
newwritersite	ローカル・サイトを最新にする必要がある新しい writer サイト（データベース・リンク）の名前。

使用上の注意

このプロシージャを使用するには、レプリケーションの Workspace Manager オブジェクトへの適用方法を理解する必要があります。詳細は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。また、Oracle Replication の主要な概念と技術についても理解する必要があります。詳細は、『Oracle9*i* アドバンスト・レプリケーション』および『Oracle9*i* レプリケーション・マネージメント API リファレンス』を参照してください。

このプロシージャは、レプリケーション管理者として実行する必要があります。

oldwritersiteavailable パラメータを FALSE に指定して RelocateWriterSite プロシージャを実行した場合は、このプロシージャを古い writer サイトで実行する必要があります。

例

次の例では、Workspace Manager のレプリケーション環境で、新しい writer サイト（BACKUP-SITE1.ACME.COM）を使用して、ローカル・システムを最新にします。

```
DBMS_WM.SynchronizeSite('BACKUP-SITE1.ACME.COM');
```

UnfreezeWorkspace プロシージャ

FreezeWorkspace プロシージャの効果を元に戻し、作業領域へのアクセスと変更を可能にします。

構文

```
DBMS_WM.UnfreezeWorkspace(
   workspace  IN VARCHAR2);
```

パラメータ

表 80-51 UnfreezeWorkspace プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。大文字と小文字が区別されます。

使用上の注意

workspace にセッションがある場合、操作は失敗します。

次のいずれかに該当する場合のみ、作業領域のアクセス制限を解除できます。

- 指定した作業領域の所有者である。
- 指定した作業領域に対して、WM_ADMIN_ROLE、FREEZE_ANY_WORKSPACE 権限または FREEZE_WORKSPACE 権限を持っている。

例

次の例では、NEWWORKSPACE 作業領域のアクセス制限を解除しています。

```
EXECUTE DBMS_WM.UnfreezeWorkspace ('NEWWORKSPACE');
```

UnlockRows プロシージャ

指定された表におけるバージョン化された行およびその親作業領域の対応行にアクセスできるようにします。

構文

```
DBMS_WM.UnlockRows(
   workspace        IN VARCHAR2,
   table_name       IN VARCHAR2
   [, where_clause  IN VARCHAR2 DEFAULT '']
   [, all_or_user   IN VARCHAR2 DEFAULT 'USER']
   [, lock_mode     IN VARCHAR2 DEFAULT 'ES']);
```

パラメータ

表 80-52 UnlockRows プロシージャのパラメータ

パラメータ	説明
workspace	作業領域名。この作業領域でロックされた行および親作業領域で対応する行は、残りのパラメータで指定されたとおり、ロック解除されます。大文字と小文字が区別されます。
table_name	行をロック解除する表の名前。大文字と小文字は区別されません。
where_clause	ロック解除される行を識別する WHERE 句（WHERE キーワードを除く）。 例 :'department_id = 20' WHERE 句で指定できるのは、主キーの列のみです。WHERE 句に副問合せを含めることはできません。 where_clause が指定されていない場合、table_name のすべての行がアクセス可能になります。
all_or_user	要求の範囲。ALL または USER のいずれかです。 ALL: 指定した作業領域でユーザーがアクセス可能なすべてのロックが対象となります。 USER（デフォルト）: 指定した作業領域でユーザーが所有するロックのみが対象となります。
lock_mode	ロック・モード。E、S または ES のいずれかです。 E: 排他モードのロックのみが考慮されます。 S: 共有モードのロックのみが考慮されます。 ES（デフォルト）: 排他モードおよび共有モードのロックが考慮されます。

使用上の注意

このプロシージャは、標準の Oracle サーバーのロックに加えて発生する Workspace Manager のロックに影響を与えます。Workspace Manager のロックの説明は、『Oracle9*i* アプリケーション開発者ガイド - Workspace Manager』を参照してください。

このプロシージャは、以前からロックされている行のロックを解除します（「LockRows プロシージャ」を参照）。Workspace Manager のロックの設定には影響を与えません。この設定がオンまたはオフにされるかどうかは、SetLockingON プロシージャおよび SetLockingOFF プロシージャにより決定されます。

例

次の例では、NEWWORKSPACE 作業領域で last_name = 'Smith' である EMPLOYEES 表の行をロック解除しています。

```
EXECUTE DBMS_WM.UnlockRows ('employees', 'NEWWORKSPACE', 'last_name = ''Smith''');
```

81

DBMS_XDB

DBMS_XDB パッケージには、PL/SQL のリソース管理とアクセス制御 API が含まれています。

> **関連項目：** 詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

DBMS_XDB の概要

DBMS_XDB パッケージによって、PL/SQL アプリケーション開発者は、Oracle XML DB 階層でリソースを管理し、Oracle XML DB のアクセス制御リスト（ACL）セキュリティと Oracle XML DB の構成管理をサポートする API を使用できます。

Oracle XML DB のリソース管理機能では、`Link()`、`LockResource()`、`GetLockToken()`、`UnlockResource()`、`CreateResource()`、`CreateFolder()`、`DeleteResource()`、`Link()` およびファンクションが提供されます。これらのメソッドは、RESOURCE_VIEW で提供される機能とともに使用します。

ACL ベースのセキュリティ・メカニズムは、階層内 ACL（Oracle XML DB のリソース API 経由で格納された ACL）またはメモリー内 ACL（Oracle XML DB 以外のユーザーが格納）のいずれかとともに使用できます。これらのメソッドの一部は、Oracle XML DB リソースおよび任意のデータベース・オブジェクトの両方に使用できます。

アクセス制御セキュリティ機能では、Oracle XML DB のリソース用に、`checkPrivileges()`、`getAclDocument()`、`changePrivileges()` および `getPrivileges()` ファンクションが提供されます。`AclCheckPrivileges()` ファンクションによって、データベース・ユーザーは、オブジェクトを Oracle XML DB 階層に格納せずに、Oracle XML DB の ACL ベースのセキュリティ・メカニズムにアクセスできます。

Oracle XML DB の構成管理では、`CFG_Refresh()`、`CFG_Get()` および `CFG_Update()` が提供されます。

DBMS_XDB のファンクションとプロシージャ

表 81-1 DBMS_XDB のファンクションとプロシージャの要約

ファンクション / プロシージャ	説明
「getAclDocument()」81-3 ページ	パス名を指定して、リソースを保護する ACL ドキュメントを取得します。
「getPrivileges()」81-3 ページ	指定した Oracle XML DB リソースに対して現行ユーザーに付与されている権限をすべて取得します。
「changePrivileges()」81-4 ページ	指定した ACE を、指定したリソースの ACL に追加します。
「checkPrivileges()」81-5 ページ	指定した Oracle XML DB リソースに対して現行ユーザーに付与されているアクセス権限をチェックします。
「setacl()」81-6 ページ	指定の Oracle XML DB リソースに対する ACL を、指定の ACL に設定します。
「AclCheckPrivileges()」81-6 ページ	指定の ACL ドキュメントで指定されている現行ユーザーに付与されたアクセス権限をチェックします。このアクセス権限は、所有者が owner パラメータで指定されているリソースに対するものです。

表 81-1 DBMS_XDB のファンクションとプロシージャの要約（続き）

ファンクション / プロシージャ	説明
「LockResource()」81-7 ページ	リソースへのパスを指定して、リソースに対する WebDAV スタイルのロックを取得します。
「GetLockToken()」81-7 ページ	リソースへのパスを指定して、現行ユーザーのリソースのロック・トークンを戻します。
「UnlockResource()」81-7 ページ	ロック・トークンとリソースへのパスを指定して、リソースのロックを解除します。
「CreateResource()」81-8 ページ	新規リソースを作成します。
「CreateFolder()」81-9 ページ	階層内に新規フォルダ・リソースを作成します。
「DeleteResource()」81-9 ページ	階層からリソースを削除します。
「Link()」81-10 ページ	既存のリソースへのリンクを作成します。
「CFG_Refresh()」81-10 ページ	最新の構成情報をデータベースに反映します。
「CFG_Get()」81-10 ページ	セッションの構成情報を取得します。
「CFG_Update()」81-11 ページ	構成情報を更新します。

getAclDocument()

パス名を指定して、リソースを保護する ACL ドキュメントを取得し、ACL ドキュメントの xmltype を戻します。

構文

```
FUNCTION getAclDocument( abspath  IN  VARCHAR2)
                         RETURN sys.xmltype;
```

パラメータ	IN / OUT	説明
abspath	(IN)	ACL ドキュメントが必要なリソースのパス名。

getPrivileges()

指定した Oracle XML DB リソースに対して現行ユーザーに付与されている権限をすべて取得します。<privilege> 要素の XMLType インスタンスを戻します。これには、このリソースに対して現行ユーザーに付与されたすべてのリーフ権限のリストが含まれています。次に例を示します。

```
<privilege xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                              http://xmlns.oracle.com/xdb/acl.xsd"
      <read-contents/>
      <read-properties/>
      <resolve/>
      <read-acl/>
 </privilege>
```

構文

```
FUNCTION getPrivileges( res_path IN VARCHAR2) RETURN sys.xmltype;
```

パラメータ	IN / OUT	説明
res_path	(IN)	Oracle XML DB リソースの階層内の絶対パス。

changePrivileges()

指定した ACE を、指定したリソースの ACL に追加します。ACL が正常に変更された場合は、正の整数を戻します。次に例を示します。

```
<ace  xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:dav="DAV:"
          xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                              http://xmlns.oracle.com/xdb/acl.xsd
                              DAV:http://xmlns.oracle.com/xdb/dav.xsd"
      <grant>true</grant>
      <principal>SCOTT</principal>
      <privilege>
          <read-contents/>
          <read-properties/>
          <resolve/>
          <dav:waste/>
      </privilege>
</ace>
```

構文

```
FUNCTION changePrivileges( res_path  IN  VARCHAR2,
                           ace       IN  xmltype)
                           RETURN pls_integer;
```

パラメータ	IN / OUT	説明
res_path	(IN)	権限を変更する必要がある Oracle XML DB リソースのパス名。
ace	(IN)	<ace> 要素の XMLType インスタンス。この要素は、<principal>、<grant> 操作および権限リストを指定します。前述のコード例を参照してください。

同じ principal および同じ操作（grant/deny）を持つ ACE が ACL に存在しない場合、新規の ACE が ACL の最後に追加されます。

checkPrivileges()

指定した Oracle XML DB リソースに対して現行ユーザーに付与されているアクセス権限をチェックします。要求されたすべての権限が付与されている場合は、正の整数を戻します。たとえば、次の <privilege> XMLType インスタンスを使用して、<read.contents>、<read.properties> および <dav:waste> 権限をチェックします。

```
<privilege xmlns="http://xmlns.oracle.com/xdb/acl.xsd"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:dav="DAV:"
          xsi:schemaLocation="http://xmlns.oracle.com/xdb/acl.xsd
                              http://xmlns.oracle.com/xdb/acl.xsd
                              DAV: http://xmlns.oracle.com/xdb/dav.xsd"
     <read-contents/>
     <read-properties/>
     <resolve/>
     <dav:waste/>
</privilege>
```

構文

```
FUNCTION checkPrivileges( res_path   IN  VARCHAR2,
                          privs      IN  xmltype)
                          RETURN pls_integer;
```

パラメータ	IN / OUT	説明
res_path	(IN)	Oracle XML DB リソースの階層内の絶対パス。
privs	(IN)	privilege 要素の XMLType インスタンス。要求された一連のアクセス権限を指定します。前述のコード例を参照してください。

setacl()

指定の Oracle XML DB リソースに対する ACL を、パスで指定した ACL に設定します。ユーザーには、リソースに対する <write-acl> 権限が必要です。

構文

```
PROCEDURE setacl( res_path   IN  VARCHAR2,
                  acl_path   IN  VARCHAR2);
```

パラメータ	IN / OUT	説明
res_path	(IN)	Oracle XML DB リソースの階層内の絶対パス。
acl_path	(IN)	Oracle XML DB ACL の階層内の絶対パス。

AclCheckPrivileges()

指定の ACL ドキュメントで指定されている現行ユーザーに付与されたアクセス権限をチェックします。このアクセス権限は、所有者が owner パラメータで指定されているリソースに対するものです。要求されたすべての権限が付与されている場合は、正の整数を戻します。

構文

```
FUNCTION AclCheckPrivileges( acl_path  IN  VARCHAR2,
                             owner     IN  VARCHAR2,
                             privs     IN  xmltype)
                             RETURN pls_integer;
```

パラメータ	IN / OUT	説明
acl_path	(IN)	ACL ドキュメントの階層内の絶対パス。
owner	(IN)	リソースの所有者名。疑似ユーザー「DAV:owner」は、ACL 権限の解決時にこのユーザーに置き換えられます。
privs	(IN)	privilege 要素の XMLType インスタンス。要求された一連のアクセス権限を指定します。 checkPrivileges()checkPrivileges() の説明を参照してください。

LockResource()

リソースへのパスを指定して、リソースに対する WebDAV スタイルのロックを取得します。操作に成功した場合は TRUE を戻し、失敗した場合は FALSE を戻します。ユーザーには、リソースに対する UPDATE 権限が必要です。

構文

```
FUNCTION LockResource( path      IN  VARCHAR2,
                       depthzero IN  BOOLEAN,
                       shared    IN  boolean)
                       RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
path	(IN)	ロックするリソースのパス名。
depthzero	(IN)	現在はサポートされていません。現在、このファンクションでロックされるのは、指定したリソースのみです。今後のリリースでは、FALSE を指定して、すべての深さのロックを取得します。
shared	(IN)	TRUE を指定すると、共有書込みロックを取得します。

GetLockToken()

リソースへのパスを指定して、現行ユーザーのリソースのロック・トークンを戻します。ユーザーには、リソースに対する READPROPERTIES 権限が必要です。

構文

```
PROCEDURE GetLockToken( path      IN  VARCHAR2,
                        locktoken OUT VARCHAR2);
```

パラメータ	IN / OUT	説明
path	(IN)	リソースへのパス名。
locktoken	(OUT)	ログインしたユーザーのリソースに対するロック・トークン。

UnlockResource()

ロック・トークンとリソースへのパスを指定して、リソースのロックを解除します。操作に成功した場合は TRUE を戻し、失敗した場合は FALSE を戻します。ユーザーには、リソースに対する UPDATE 権限が必要です。

構文

```
FUNCTION UnlockResource( path     IN  VARCHAR2,
                         deltoken IN  VARCHAR2)
                         RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
path	(IN)	リソースへのパス名。
deltoken	(IN)	削除するロック・トークン。

CreateResource()

新規リソースを作成します。操作に成功した場合は TRUE を戻し、失敗した場合は FALSE を戻します。次の表では、オプションについて説明します。

構文	説明
FUNCTION CreateResource(path IN VARCHAR2, data IN VARCHAR2) RETURN BOOLEAN;	指定した文字列を内容として使用し、新規リソースを作成します。
FUNCTION CreateResource(path IN VARCHAR2, data IN SYS.XMLTYPE) RETURN BOOLEAN;	指定した XMLType データを内容として使用し、新規リソースを作成します。
FUNCTION CreateResource(path IN VARCHAR2, datarow IN REF SYS.XMLTYPE) RETURN BOOLEAN;	既存の XMLType 行に REF を指定し、その行を指す内容を持つリソースを作成します。この行は、別のリソース内に事前に存在させることはできません。
FUNCTION CreateResource(path IN VARCHAR2, data IN CLOB) RETURN BOOLEAN;	指定した CLOB を内容として使用し、新規リソースを作成します。
FUNCTION CreateResource(path IN VARCHAR2, data IN BFILE) RETURN BOOLEAN;	指定した BFILE を内容として使用し、新規リソースを作成します。

パラメータ	IN / OUT	説明
path	(IN)	作成するリソースのパス名。パス名の親フォルダは、階層内にすでに存在している必要があります。たとえば、'/foo/bar.txt' と指定する場合は、フォルダ '/foo' がすでに存在している必要があります。
data	(IN)	新規リソースの内容。データは解析され、スキーマに基づいた XML 文書が含まれているかどうかがチェックされます。含まれている場合、内容はスキーマのデフォルト表にスキーマに基づいたものとして格納されます。含まれていない場合は、バイナリ・データとして格納されます。
datarow	(IN)	内容として使用する XMLType 行への参照。

CreateFolder()

階層内に新規フォルダ・リソースを作成します。操作に成功した場合は TRUE を戻し、失敗した場合は FALSE を戻します。指定したパス名の親フォルダは、階層内にすでに存在している必要があります。たとえば、path パラメータに '/folder1/folder2' を指定する場合は、'/folder1' がすでに存在している必要があります。

構文

```
FUNCTION CreateFolder( path   IN  VARCHAR2)
                       RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
path	(IN)	新規フォルダのパス名。

DeleteResource()

階層からリソースを削除します。

構文

```
PROCEDURE DeleteResource( path   IN  VARCHAR2);
```

パラメータ	IN / OUT	説明
path	(IN)	削除するリソースのパス名。

Link()

既存のリソースへのリンクを作成します。このプロシージャは、UNIX におけるハード・リンクの作成に似ています。

構文

```
PROCEDURE Link( srcpath    IN  VARCHAR2,
                linkfolder IN  VARCHAR2,
                linkname   IN  VARCHAR2);
```

パラメータ	IN / OUT	説明
srcpath	(IN)	リンクを作成するリソースのパス名。
linkfolder	(IN)	新規リンクを格納するフォルダ。
linkname	(IN)	新規リンクの名前。

CFG_Refresh()

最新の構成情報をデータベースに反映します。

構文

```
PROCEDURE CFG_Refresh;
```

CFG_Get()

セッションの構成情報を XMLType インスタンスとして取得します。

構文

```
FUNCTION CFG_Get RETURN SYS.XMLType;
```

CFG_Update()

構成情報を更新して、変更をコミットします。

構文

```
PROCEDURE CFG_Update( xdbconfig   IN  SYS.XMLTYPE);
```

パラメータ	IN / OUT	説明
xdbconfig	(IN)	新規の構成データ。

82

DBMS_XDBT

DBMS_XDBT パッケージによって、管理者は、XML DB 階層に ConText 索引を作成し、自動メンテナンス用に構成できます。

> **関連項目：** 詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

BMS_XDBT の概要

DBMS_XDBT パッケージは、管理者に対して、Oracle XML DB 階層に ConText 索引を設定するための便利なメカニズムを提供します。このパッケージには、デフォルトのプリファレンスの作成、索引の作成、および ConText 索引の自動同期化の設定を行うためのプロシージャが含まれています。

また、DBMS_XDBT パッケージには、索引の構成設定を指定する一連のパッケージ変数も含まれています。これらのパッケージ変数は、インストールに必要な基本カスタマイズで使用されますが、完全なセットではありません。

DBMS_XDBT パッケージは、次のように使用できます。

- パッケージをカスタマイズして、適切な構成を設定します。
- dropPreferences() プロシージャを使用して、既存の索引のプリファレンスを削除します。
- createPreferences() プロシージャを使用して、索引の新規のプリファレンスを作成します。
- createIndex() プロシージャを使用して、ConText 索引を作成します。
- configureAutoSync() プロシージャを使用して、索引の自動同期化を設定します。

BMS_XDBT のファンクションとプロシージャ

表 82-1 DBMS_XDBT のファンクションとプロシージャの要約

プロシージャ / ファンクション	説明
「dropPreferences()」 82-3 ページ	既存のプリファレンスを削除します。
「createPreferences()」 82-3 ページ	XML DB 階層の ConText 索引に必要なプリファレンスを作成します。
「createDatastorePref()」 82-3 ページ	ConText 索引用の USER データストア・プリファレンスを作成します。
「createFilterPref()」 82-4 ページ	ConText 索引用のフィルタ・プリファレンスを作成します。
「createLexerPref()」 82-4 ページ	ConText 索引用のレクサー・プリファレンスを作成します。
「createWordlistPref()」 82-5 ページ	ConText 索引用のストップリストを作成します。
「createStoplistPref()」 82-5 ページ	ConText 索引用のセクション・グループを作成します。

表 82-1 DBMS_XDBT のファンクションとプロシージャの要約 （続き）

プロシージャ / ファンクション	説明
「createStoragePref()」 82-5 ページ	ConText 索引用のワードリスト・プリファレンスを作成します。
「createSectiongroupPref()」 82-6 ページ	ConText 索引用の記憶域プリファレンスを作成します。
「createIndex()」 82-6 ページ	XML DB 階層の ConText 索引を作成します。
「configureAutoSync()」 82-7 ページ	自動メンテナンス（同期化）用の ConText 索引を構成します。

dropPreferences()

このプロシージャは、XML DB 階層の ConText 索引用に以前に作成されたプリファレンスを削除します。

構文

```
PROCEDURE dropPreferences;
```

createPreferences()

このプロシージャは、構成設定に基づいて、一連のデフォルトのプリファレンスを作成します。

構文

```
PROCEDURE createPreferences;
```

createDatastorePref()

このプロシージャは、XML DB 階層の ConText 索引用の USER データストア・プリファレンスを作成します。

- データストア・プリファレンスの名前は変更できます。DatastorePref 構成設定を参照してください。
- デフォルトの USER データストア・プロシージャは、渡されたドキュメントのフィルタも行います。DBMS_XDBT パッケージは、フィルタ処理を制御する一連の構成設定を提供します。

- SkipFilter_Types 配列には、標準の式のリストが含まれています。いずれかの式と一致する MIME タイプのドキュメントは、索引付けされません。ドキュメント・メタデータの一部のプロパティ（作成者など）も索引付けされません。
- NullFilter_Types 配列には、標準の式のリストが含まれています。いずれかの式と一致する MIME タイプのドキュメントは、フィルタ処理は行われませんが、索引付けは行われます。これは、HTML、XML、プレーン・テキストなど、テキストベースのドキュメントで使用するためです。
- 他のすべてのドキュメントでは、IFILTER API を経由して INSO フィルタが使用されます。

構文

```
PROCEDURE createDatastorePref;
```

createFilterPref()

このプロシージャは、XML DB 階層の ConText 索引用の NULL フィルタ・プリファレンスを作成します。

- フィルタ・プリファレンスの名前は変更できます。FilterPref 構成設定を参照してください。
- USER データストア・プロシージャは、渡されるドキュメントのフィルタを行います。詳細は、「createDatastorePref」を参照してください。

構文

```
PROCEDURE createFilterPref;
```

createLexerPref()

このプロシージャは、XML DB 階層の ConText 索引用の基本 lexer プリファレンスを作成します。

- lexer プリファレンスの名前は変更できます。LexerPref 構成設定を参照してください。これ以外の構成設定はありません。
- MultiLexer プリファレンスはサポートされていません。
- デフォルトでは、ベース文字変換はオンになっています。

構文

```
PROCEDURE createLexerPref;
```

createWordlistPref()

このプロシージャは、XML DB 階層の ConText 索引用のワードリスト・プリファレンスを作成します。

- ワードリスト・プリファレンスの名前は変更できます。WordlistPref 構成設定を参照してください。これ以外の構成設定はありません。
- FUZZY_MATCH 属性と STEMMER 属性は、AUTO（自動言語検出）に設定されます。

構文

```
PROCEDURE createWordlistPref;
```

createStoplistPref()

このプロシージャは、XML DB 階層の ConText 索引用のストップリストを作成します。

- ストップリストの名前は変更できます。StoplistPref 構成設定を参照してください。
- 数値は索引付けされません。
- StopWords 配列は、ストップワードの構成可能なリストです。このリストには、CTXSYS.DEFAULT_STOPLIST の一連のストップワード以外のストップワードが含まれます。

構文

```
PROCEDURE createStoplistPref;
```

createStoragePref()

このプロシージャは、XML DB 階層の ConText 索引用の BASIC_STORAGE プリファレンスを作成します。

- 記憶域プリファレンスの名前は変更できます。StoragePref 構成設定を参照してください。
- ConText 索引を構成する表と索引用の表領域を指定できます。IndexTablespace 構成設定を参照してください。
- デフォルトでは、接頭辞とサブストリングの索引付けはオンになっていません。
- I_INDEX_CLAUSE では、キー圧縮が使用されます。

構文

```
PROCEDURE createStoragePref;
```

createSectiongroupPref()

このプロシージャは、XML DB 階層の ConText 索引用のセクション・グループを作成します。

- セクション・グループの名前は変更できます。SectiongroupPref 構成設定を参照してください。
- デフォルトでは、HTML セクショナが使用されます。デフォルトでは、ゾーン・セクションは作成されません。大部分のドキュメントが XML の場合は、AUTO_SECTION_GROUP または PATH_SECTION_GROUP の使用を検討してください。SectionGroup 構成設定を参照してください。

構文

```
PROCEDURE createSectiongroupPref;
```

createIndex()

このプロシージャは、XML DB 階層の ConText 索引を作成します。

- 索引の名前は変更できます。IndexName 構成設定を参照してください。
- LogFile 構成パラメータを設定すると、索引作成時に ROWID ロギングが使用可能になります。
- IndexMemory 構成パラメータを設定すると、索引作成および後の同期化で使用するメモリー量を判断できます。

構文

```
PROCEDURE createIndex;
```

configureAutoSync()

このプロシージャは、ConText 索引の自動同期化を行うためのジョブを設定します。

- システムは、自動同期化のジョブ・キュー用に構成する必要があります。ジョブは、USER_JOBS カタログ・ビューを使用して表示できます。
- AutoSyncPolicy 構成パラメータを設定して、適切な同期ポリシーを選択できます。

同期化は、次のいずれかを基準にして実行できます。

同期化の基準	説明
SYNC_BY_PENDING_COUNT	同期化は、ペンディング・キュー内のドキュメント数がしきい値を超過するとトリガーされます（MaxPendingCount 構成設定を参照）。ペンディング・キューは定期的にポーリングされ（CheckPendingCountInterval 構成パラメータを参照）、ドキュメント数がしきい値を超過しているかどうかが判別されます。
SYNC_BY_TIME	同期化は、定期的にトリガーされます （SyncInterval 構成パラメータを参照）。
SYNC_BY_PENDING_COUNT_AND_TIME	前述の 2 つの基準の組合せ。

構文

```
PROCEDURE configureAutoSync;
```

DBMS_XDBT パッケージのカスタマイズ

DBMS_XDBT パッケージは、次のいずれかの方法でカスタマイズできます。

- PL/SQL プロシージャまたは無名ブロックを使用して関連するパッケージ変数（構成設定）を設定してから、このパッケージ内のプロシージャを実行します。
- 一般的な方法として、このパッケージを適切に変更してカスタマイズして導入するか、またはコピーとして導入します。

システムは、ジョブ・キューが使用できるように構成されている必要があることに注意してください。ジョブは、USER_JOBS カタログ・ビューを使用して表示できます。

ここでは、DBMS_XDBT パッケージのカスタマイズに使用できる構成設定（パッケージ変数）について説明します。

一般的な索引付け設定

次の表は、一般的な索引付けに関連する構成設定のリストです。

パラメータ	デフォルト値	説明
IndexName	XDB$CI	ConText 索引の名前。
IndexTablespace	XDB$RESINFO	ConText 索引を構成する表と索引で使用する表領域。
IndexMemory	128M	索引作成と同期化で使用するメモリー量。この値は、MAX_INDEX_MEMORY システム・パラメータの値以下であることが必要です。MAX_INDEX_MEMORY システム・パラメータ（CTX_ADMIN パッケージを参照）の値は、IndexMemory の設定値以下であることが必要です。
LogFile	'XdbCtxLog'	索引作成時に ROWID ロギングで使用するログ・ファイル。LOG_DIRECTORY システム・パラメータは設定済みであることが必要です。このパラメータを NULL に設定すると、ROWID ロギングはオフになります。LOG_DIRECTORY システム・パラメータ（CTX_ADMIN パッケージを参照）は、ROWID ロギングを使用できるように設定する必要があります。

フィルタ設定

次の表は、XML DB 階層でドキュメントのフィルタ処理を制御する構成設定のリストです。

パラメータ	デフォルト値	説明
SkipFilter_Types	image/%, audio/%, video/%, model/%	索引付けが必要な MIME タイプのリスト。
NullFilter_Types	text/plain, text/html, text/xml	INSO フィルタを使用する必要がない MIME タイプのリスト。このパラメータは、テキストベースのドキュメントで使用します。
FilterPref	XDB$CI_FILTER	フィルタ・プリファレンスの名前。

セクションとセクション・グループ設定

次の表は、セクショナに関連する構成設定のリストです。

パラメータ	デフォルト値	説明
SectionGroup	HTML_SECTION_GROUP	デフォルトで使用するセクショナ。リポジトリに格納されているのが主に XML 文書の場合は、PATH_SECTION_GROUP または AUTO_SECTION_GROUP の使用を検討してください。
SectiongroupPref	XDB$CI_SECTIONGROUP	セクション・グループの名前。

ストップリスト設定

次の表は、ストップリストの構成設定のリストです。

パラメータ	デフォルト値	説明
StoplistPref	XDB$CI_STOPLIST	ストップリストの名前。
StopWords	0..9 'a'..'z' 'A'..'Z'	CTXSYS.DEFAULT_STOPLIST に指定されているストップワード以外のストップワードのリスト。

その他のプリファレンス設定

次の表は、その他の索引プリファレンス用の設定のリストです。

パラメータ	デフォルト値	説明
DatastorePref	XDB$CI_DATASTORE	データストア・プリファレンスの名前。
StoragePref	XDB$CI_STORAGE	記憶域プリファレンスの名前。
WordlistPref	XDB$CI_WORDLIST	ワードリスト・プリファレンスの名前。
DefaultLexerPref	XDB$CI_DEFAULT_LEXER	デフォルトのレクサー・プリファレンスの名前。

索引の同期化の設定

次の表は、ConText 索引の同期化の時期と方法を制御する設定のリストです。

パラメータ	デフォルト値	説明
AutoSyncPolicy	SYNC_BY_PENDING_COUNT	索引を同期化する時期を指定します。次のいずれかを指定できます。 - SYNC_BY_PENDING_COUNT - SYNC_BY_TIME - SYNC_BY_PENDING_COUNT_AND_TIME
MaxPendingCount	2	索引の同期化がトリガーされる前に、この索引の CTX_USER_PENDING キューに含まれるドキュメントの最大数。AutoSyncPolicy が次のいずれかの場合のみ適用されます。 - SYNC_BY_PENDING_COUNT - SYNC_BY_PENDING_COUNT_AND_TIME
CheckPending CountInterval	10 minutes	ペンディング・キューのチェック頻度を分単位で指定します。AutoSyncPolicy が次のいずれかの場合のみ適用されます。 - SYNC_BY_PENDING_COUNT - SYNC_BY_PENDING_COUNT_AND_TIME
SyncInterval	60 minutes	索引を同期化する頻度を分単位で指定します。AutoSyncPolicy が次のいずれかの場合のみ適用されます。 - SYNC_BY_TIME - SYNC_BY_PENDING_COUNT_AND_TIME

83

DBMS_XDB_VERSION

Oracle XML DB バージョニング API は、DBMS_XBD_VERSION パッケージに含まれています。

> **関連項目：** 詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

DBMS_XDB_VERSION の概要

DBMS_XDB_VERSION のファンクションとプロシージャは、バージョン履歴内の VCR の作成とバージョンの管理に役立ちます。

DBMS_XDB_VERSION のファンクションとプロシージャ

表 83-1 DBMS_XDB_VERSION のファンクションとプロシージャの要約

ファンクション / プロシージャ	説明
「MakeVersioned()」83-3 ページ	パス名を指定した標準リソースをバージョン管理されたリソース（VCR）に変換します。
「Checkout()」83-3 ページ	VCR を更新または削除する前にチェックアウトします。
「Checkin()」83-4 ページ	チェックアウトされた VCR をチェックインし、新規作成されたバージョンのリソース ID を戻します。
「Uncheckout()」83-4 ページ	チェックアウトされたリソースをチェックインし、リソースがチェックアウトされる前にバージョンのリソース ID を戻します。
「GetPredecessors()」83-5 ページ	先行リソースのリストをパス名を指定することによって取得します。
「GetPredsByResId()」83-5 ページ	先行リソースのリストをリソース ID を指定することによって取得します。
「GetResourceByResId()」83-5 ページ	リソース・オブジェクト ID を指定して、リソースを XMLType として取得します。
「GetSuccessors()」83-5 ページ	後続リソースのリストをパス名を指定することによって取得します。
「GetSuccsByResId()」83-6 ページ	後続リソースのリストをリソース ID を指定することによって取得します。

MakeVersioned()

パス名を指定した標準リソースをバージョン管理されたリソースに変換します。複数のパス名が同じリソースにバインドされている場合は、リソースのコピーが作成され、指定したパス名が、新たに作成されたコピーにバインドされます。これによって、この新規リソースはバージョン管理の対象になります。他のすべてのパス名は、元のリソースを参照し続けます。このファンクションは、VCR の最初のバージョン（ルート）のリソース ID を戻します。これは、自動コミットを行う SQL 操作ではありません。

- VCR に対して MakeVersioned() をコールでき、例外や警告は発生しません。
- フォルダ、バージョン履歴、バージョン・リソースおよび ACL に対して MakeVersioned() をコールできます。
- スキーマに基づいたリソースはサポートされていません。

リソースが存在しない場合は、例外が発生します。

構文

```
FUNCTION MakeVersioned( pathname VARCHAR2) RETURN dbms_xdb.resid_type;
```

パラメータ	説明
pathname	バージョン管理の対象にするリソースのパス名。

Checkout()

VCR を更新または削除する前にチェックアウトします。これは、自動コミットの SQL 操作ではありません。同じ作業領域の 2 名のユーザーが、Checkout() を使用して同時に同じ VCR をチェックアウトすることはできません。同時にチェックアウトすると、いずれかのユーザーはロールバックする必要があります。このため、リソースの更新前に Checkout() 操作をコミットし、トランザクションがロールバックされた場合の更新内容の消失を回避することをお薦めします。指定したリソースが VCR でない場合、VCR がすでにチェックアウトされている場合、またはリソースが存在しない場合は、例外が発生します。

構文

```
PROCEDURE Checkout(  pathname VARCHAR2);
```

パラメータ	説明
pathname	チェックアウトする VCR のパス名。

Checkin()

チェックアウトされた VCR をチェックインし、新規作成されたバージョンのリソース ID を戻します。これは、自動コミットを行う SQL 操作ではありません。Checkin() には、Checkout() 操作で渡したパス名と同じパス名を渡す必要はありません。ただし、操作が正しく機能するためには、Checkin() と Checkout() のパス名が同じリソースのパス名であることが必要です。リソースが改名されている場合、変更前の名前はすでに無効になっているか、現在は別のリソースにバインドされているため、Checkin() では新規の名前を使用する必要があります。パス名が存在しない場合は、例外が発生します。パス名が変更されている場合、リソースの Checkin() では新規の名前を使用する必要があります。

構文

```
FUNCTION Checkin( pathname VARCHAR2) RETURN dbms_xdb.resid_type;
```

パラメータ	説明
pathname	チェックアウトされたリソースのパス名。

Uncheckout()

チェックアウトされたリソースをチェックインし、リソースがチェックアウトされる前にバージョンのリソース ID を戻します。これは、自動コミットを行う SQL 操作ではありません。Uncheckout() には、Checkout() 操作で渡したパス名と同じパス名を渡す必要はありません。ただし、操作が正しく行われるには、Uncheckout() と Checkout() のパス名が同じリソースのパス名であることが必要です。リソースが改名されている場合、変更前の名前はすでに無効になっているか、現在は別のリソースにバインドされているため、Uncheckout では新規の名前を使用する必要があります。パス名が存在しない場合は、例外が発生します。パス名が変更されている場合、リソースの Uncheckout() では新規の名前を使用する必要があります。

構文

```
FUNCTION Uncheckout( pathname VARCHAR2) RETURN dbms_xdb.resid_type;
```

パラメータ	説明
pathname	チェックアウトされたリソースのパス名。

GetPredecessors()

先行リソースのリストをパス名を指定することによって取得します。pathname が無効の場合は、例外が発生します。

構文

```
FUNCTION GetPredecessors( pathname VARCHAR2) RETURN resid_list_type;
```

パラメータ	説明
pathname	リソースのパス名。

GetPredsByResId()

先行リソースのリストをリソース ID を指定することによって取得します。先行リソースの取得は、resid 別より pathname 別の方が効率的です。resid が無効の場合は、例外が発生します。

構文

```
FUNCTION GetPredsByResId( resid resid_type) RETURN resid_list_type;
```

パラメータ	説明
resid	リソース ID。

GetResourceByResId()

リソース・オブジェクト ID を指定して、リソースを XMLType として取得します。システムではバージョン用のパス名を作成しないため、このファンクションは、リソース ID を使用してリソースを取得するのに役立ちます。

構文

```
FUNCTION GetResourceByResId( resid resid_type) RETURN XMLType;
```

パラメータ	説明
resid	リソース ID。

GetSuccessors()

バージョン・リソースまたは VCR を指定して、後続リソースのリストをパス名別に取得します。後続リソースの取得は、resid 別より pathname 別の方が効率的です。pathname が無効の場合は、例外が発生します。

構文

```
FUNCTION GetSuccessors( pathname VARCHAR2) RETURN resid_list_type;
```

パラメータ	説明
pathname	リソースのパス名。

GetSuccsByResId()

バージョン・リソースまたは VCR を指定して、後続リソースのリストをリソース ID 別に取得します。後続リソースの取得は、resid 別より pathname 別の方が効率的です。resid が無効の場合は、例外が発生します。

構文

```
FUNCTION GetSuccsByResId( resid resid_type) RETURN resid_list_type;
```

パラメータ	説明
resid	リソース ID。

84

DBMS_XMLDOM

DBMS_XMLDOM を使用して、XMLType オブジェクトにアクセスします。スキーマに基づいたドキュメントとスキーマに基づかないドキュメントの両方にアクセスできます。データベースの起動前には、読込みおよび書込みを行うディレクトリを init.ora ファイルに指定する必要があります。次に例を示します。

```
UTL_FILE_DIR=/mypath/insidemypath
```

読込みおよび書込みを行うファイルは、サーバー・ファイル・システム上に存在する必要があります。

関連項目：

- 第 95 章「UTL_FILE」
- 『Oracle9*i* XML Developer's Kit ガイド - XDK』
- 『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』

この章では、次の項目について説明します。

DBMS_XMLDOM の概要

ドキュメント・オブジェクト・モデル（DOM）は、HTML ドキュメントと XML 文書用の Application Program Interface（API）です。DOM は、ドキュメントの論理構造、およびドキュメントのアクセスと操作の方法を定義します。DOM 仕様では、「ドキュメント」という用語は広い意味で使用されます。XML は、様々なシステムに格納できる様々な種類の情報を表現する方法として広く使用されるようになり、そのほとんどは、従来からドキュメントではなくデータとみなされていました。しかし、XML はデータをドキュメントとして表現し、データの管理に DOM を使用できます。

DOM を使用すると、プログラマはドキュメントを作成してその構造にナビゲートし、要素と内容を追加、変更または削除できます。HTML または XML 文書の内容は、DOM を使用してアクセス、変更、削除または追加を行うことができます（いくつかの例外があります）。特に、XML の内部および外部のサブセットに対する DOM インタフェースは、まだ仕様化されていません。

DOM の W3C 仕様の重要な目的の 1 つは、広範な環境とアプリケーションで使用できる標準プログラミング・インタフェースの提供にあります。DOM は、あらゆるプログラミング言語で使用できるように設計されています。DOM 標準はオブジェクト指向であるため、PL/SQL に適合するために次の変更が必要でした。

- 各種の DOM インタフェース（Node、Element など）には、同等の PL/SQL タイプ（それぞれ、DOMNode、DOMElement など）があります。
- 各種の DOM 例外コード（WRONG_DOCUMENT_ERR、HIERARCHY_REQUEST_ERR など）には、同じ名前の PL/SQL 例外があります。
- 各種の DOM のノード・タイプ・コード（ELEMENT_NODE、ATTRIBUTE_NODE など）には、同じ名前の PL/SQL 定数があります。
- DOM タイプに対して定義されたメソッドは、その DOM タイプをパラメータとして受け取るファンクションまたはプロシージャになります。たとえば、DOM のノード n で appendChild を実行するための PL/SQL ファンクション appendChild()（84-25 ページを参照）が用意されています。

```
FUNCTION appendChild( n DOMNode,
                      newChild IN DOMNode)
                      RETURN DOMNode;
```

また、DOM 要素の elem で setAttribute を実行するための PL/SQL プロシージャ setAttribute()（84-52 ページを参照）が用意されています。

```
PROCEDURE setAttribute( elem DOMElement,
                        name IN VARCHAR2,
                        value IN VARCHAR2);
```

DOM は、継承の階層を定義します。たとえば、ドキュメント、要素および属性は、ノードのサブタイプとして定義されます。したがって、ノード・インタフェースに定義されているメソッドは、これらのサブタイプでも使用可能であることが必要です。このような継承は PL/SQL で直接行うことができないため、makeNode ファンクションは、異なる DOM タイプで起動し、DOMNode に変換する必要があります。DOMNode を受け取る適切なファンクションまたはプロシージャがコールされ、これらのタイプが操作されます。その後、タイプ固有の機能が必要な場合は、make*() ファンクションを使用して、DOMNode をそのタイプに再変換できます。ここでは、DOM* が目的の DOM タイプです。

この PL/SQL DOM インタフェースの実装は、DOM 標準の改訂 REC-DOM-Level-1-19981001 に準拠しています。このマニュアルで説明するタイプとメソッドは、PL/SQL パッケージの DBMS_XMLDOM によって使用可能になります。

- データベースの起動前には、読込みディレクトリと書込みディレクトリを init.ORA ファイルに指定する必要があります。次に例を示します。

```
UTL_FILE_DIR=/mypath/insidemypath
```

- 読込みおよび書込みを行うファイルは、サーバー・ファイル・システム上に存在する必要があります。

DBMS_XMLDOM のタイプ

表 84-1 では、DBMS_XMLDOM.DOMTYPE のタイプが定義されています。

表 84-1 XDB_XMLDOM タイプ

タイプ	説明
DOMNode	DOM の Node インタフェースを実装します。
DOMNamedNodeMap	DOM の NamedNodeMap インタフェースを実装します。
DOMNodeList	DOM の NodeList インタフェースを実装します。
DOMAttr	DOM の Attr インタフェースを実装します。
DOMCDataSection	DOM の CDataSection インタフェースを実装します。
DOMCharacterData	DOM の CharacterData インタフェースを実装します。
DOMComment	DOM の Comment インタフェースを実装します。
DOMDocumentFragment	DOM の DocumentFragment インタフェースを実装します。
DOMElement	DOM の Element インタフェースを実装します。
DOMEntity	DOM の Entity インタフェースを実装します。
DOMEntityReference	DOM の EntityReference インタフェースを実装します。

表 84-1 XDB_XMLDOM タイプ （続き）

タイプ	説明
DOMNotation	DOM の Notation インタフェースを実装します。
DOMProcessingInstruction	DOM の ProcessingInstruction インタフェースを実装します。
DOMText	DOM の Text インタフェースを実装します。
DOMImplementation	DOM の DOMImplementation インタフェースを実装します。
DOMDocumentType	DOM の DocumentType インタフェースを実装します。
DOMDocument	DOM の Document インタフェースを実装します。

DBMS_XMLDOM の事前定義定数

表 84-2 に、DBMS_XMLDOM に対して定義されている定数を示します。たとえば、getNodeType(myNode) などの要求が行われると、戻されるタイプは次のいずれかの定数になります。

表 84-2 DBMS_XMLDOM の事前定義定数

定数	説明
ELEMENT_NODE	ノードは要素です。
ATTRIBUTE_NODE	ノードは属性です。
TEXT_NODE	ノードはテキスト・ノードです。
CDATA_SECTION_NODE	ノードは CDataSection です。
ENTITY_REFERENCE_NODE	ノードは実体参照です。
ENTITY_NODE	ノードはエンティティです。
PROCESSING_INSTRUCTION_NODE	ノードは処理命令です。
COMMENT_NODE	ノードはコメントです。
DOCUMENT_NODE	ノードはドキュメントです。
DOCUMENT_TYPE_NODE	ノードは Document Type Definition（DTD）です。
DOCUMENT_FRAGMENT_NODE	ノードはドキュメント・フラグメントです。
NOTATION_NODE	ノードは表記法です。

DBMS_XMLDOM の例外

表 84-3 に、DBMS_XMLDOM に対して定義されている例外を示します。

表 84-3 DBMS_XMLDOM の例外

例外	説明
INDEX_SIZE_ERR	索引またはサイズが負数か、許容値を超えている場合。
DOMSTRING_SIZE_ERR	指定したテキスト範囲が DOMString に適合しない場合。
HIERARCHY_REQUEST_ERR	ノードが属していない場所にノードが挿入された場合。
WRONG_DOCUMENT_ERR	ノードを作成したドキュメント以外のドキュメント（そのノードをサポートしていない）でそのノードが使用された場合。
INVALID_CHARACTER_ERR	名前などに指定した文字が無効な場合。有効な文字の定義については XML 仕様の生成規則を、有効な名前文字の定義については生成規則を参照してください。
NO_DATA_ALLOWED_ERROR	データをサポートしないノードにデータが指定された場合。
NO_MODIFICATION_ALLOWED_ERR	変更不可のオブジェクトを変更しようとした場合。
NO_FOUND_ERR	ノードが存在しないコンテキストでノードを参照しようとした場合。
NOT_SUPPORTED_ERR	要求されたタイプのオブジェクトまたは操作が実装でサポートされていない場合。
INUSE_ATTRIBUTE_ERR	すでに他で使用中の属性を追加しようとした場合。

DBMS_XMLDOM のファンクションとプロシージャ

DBMS_XMLDOM サブプログラムは、W3C インタフェースに従って複数のグループに分割されます。

表 84-4 DBMS_XMLDOM のファンクションとプロシージャの要約

グループ / メソッド	説明
DOMNode のメソッド	
「isNull()」84-13 ページ	ノードが NULL かどうかをテストします。
「makeAttr()」84-13 ページ	ノードを属性に変換します。
「makeCDataSection()」84-13 ページ	ノードを CDataSection に変換します。
「makeCharacterData()」84-14 ページ	ノードを CharacterData に変換します。
「makeComment()」84-14 ページ	ノードをコメントに変換します。
「makeDocumentFragment()」84-14 ページ	ノードを DocumentFragment に変換します。
「makeDocumentType()」84-15 ページ	ノードをドキュメント・タイプに変換します。
「makeElement()」84-15 ページ	ノードを要素に変換します。
「makeEntity()」84-15 ページ	ノードをエンティティに変換します。
「makeEntityReference()」84-16 ページ	ノードを EntityReference に変換します。
「makeNotation()」84-16 ページ	ノードを表記法に変換します。
「makeProcessingInstruction()」84-16 ページ	ノードを DOMProcessingInstruction に変換します。
「makeText()」84-17 ページ	ノードを DOMText に変換します。
「makeDocument()」84-17 ページ	ノードを DOMDocument に変換します。
「writeToFile()」84-17 ページ	ノードの内容をファイルに書き込みます。
「writeToBuffer()」84-18 ページ	ノードの内容をバッファに書き込みます。
「writeToClob()」84-19 ページ	ノードの内容を CLOB に書き込みます。
「getNodeName()」84-19 ページ	ノードの名前を取得します。
「getNodeValue()」84-20 ページ	ノードの値を取得します。
「setNodeValue()」84-20 ページ	ノードの値を設定します。
「getNodeType()」84-20 ページ	ノードのタイプを取得します。

表 84-4 DBMS_XMLDOM のファンクションとプロシージャの要約（続き）

グループ / メソッド	説明
「getParentNode()」84-21 ページ	ノードの親を取得します。
「getChildNodes()」84-21 ページ	ノードの子を取得します。
「getFirstChild()」84-21 ページ	ノードの最初の子を取得します。
「getLastChild()」84-22 ページ	ノードの最後の子を取得します。
「getPreviousSibling()」84-22 ページ	ノードの以前の兄弟関係を取得します。
「getNextSibling()」84-22 ページ	ノードの次の兄弟関係を取得します。
「getAttributes()」84-23 ページ	ノードの属性を取得します。
「getOwnerDocument()」84-23 ページ	ノードの所有者ドキュメントを取得します。
「insertBefore()」84-23 ページ	参照先の子の前に子を挿入します。
「replaceChild()」84-24 ページ	古い子を新規の子に置換します。
「removeChild()」84-24 ページ	指定した子をノードから削除します。
「appendChild()」84-25 ページ	新規の子をノードに追加します。
「hasChildNodes()」84-25 ページ	ノードに子ノードがあるかどうかをテストします。
「cloneNode()」84-26 ページ	ノードをクローニングします。
DOMNamedNodeMap のメソッド	
「isNull()」84-26 ページ	NamedNodeMap が NULL かどうかをテストします。
「getNamedItem()」84-27 ページ	指定した名前の項目を取得します。
「setNamedItem()」84-27 ページ	指定した名前の項目をマップ内に設定します。
「removeNamedItem()」84-28 ページ	指定した名前の項目を削除します。
「item()」84-28 ページ	索引を指定してマップ内の項目を取得します。
「getLength()」84-29 ページ	マップ内の項目の数を取得します。
DOMNodelist のメソッド	
「isNull()」84-29 ページ	Nodelist が NULL かどうかをテストします。
「item()」84-29 ページ	索引を指定して Nodelist 内の項目を取得します。
「getLength()」84-30 ページ	リスト内の項目の数を取得します。

表 84-4 DBMS_XMLDOM のファンクションとプロシージャの要約（続き）

グループ / メソッド	説明
DOMAttr のメソッド	
「isNull()」84-30 ページ	属性ノードが NULL かどうかをテストします。
「makeNode()」84-31 ページ	属性をノードに変換します。
「getQualifiedName()」84-31 ページ	属性の修飾名を取得します。
「getNamespace()」84-31 ページ	属性の名前空間 URI を取得します。
「getLocalName()」84-32 ページ	属性のローカル名を取得します。
「getExpandedName()」84-32 ページ	属性の拡張名を取得します。
「getName()」84-32 ページ	属性の名前を取得します。
「getSpecified()」84-33 ページ	所有する要素に属性が指定されているかどうかをテストします。
「getValue()」84-33 ページ	属性の値を取得します。
「setValue()」84-33 ページ	属性の値を設定します。
DOMCDataSection のメソッド	
「isNull()」84-34 ページ	CDataSection が NULL かどうかをテストします。
「makeNode()」84-34 ページ	CDatasection をノードに変換します。
DOMCharacterData のメソッド	
「isNull()」84-35 ページ	CharacterData が NULL かどうかをテストします。
「makeNode()」84-35 ページ	CharacterData をノードに変換します。
「getData()」84-35 ページ	ノードのデータを取得します。
「setData()」84-36 ページ	データをノードに設定します。
「getLength()」84-36 ページ	データの長さを取得します。
「substringData()」84-36 ページ	データのサブストリングを取得します。
「appendData()」84-37 ページ	指定したデータをノード・データに追加します。
「insertData()」84-37 ページ	ノード内のデータを指定の offSets に挿入します。
「deleteData()」84-38 ページ	指定の offSets からデータを削除します。
「replaceData()」84-38 ページ	指定の offSets のデータを置換します。

表 84-4 DBMS_XMLDOM のファンクションとプロシージャの要約（続き）

グループ / メソッド	説明
DOMComment のメソッド	
「isNull()」84-39 ページ	コメントが NULL かどうかをテストします。
「makeNode()」84-39 ページ	コメントをノードに変換します。
DOMImplementation のメソッド	
「isNull()」84-40 ページ	DOMImplementation ノードが NULL かどうかをテストします。
「hasFeature()」84-40 ページ	DOMImplementation が指定の機能を実装しているかどうかをテストします。
DOMDocumentFragment のメソッド	
「isNull()」84-41 ページ	DocumentFragment が NULL かどうかをテストします。
「makeNode()」84-41 ページ	ドキュメント・フラグメントをノードに変換します。
DOMDocumentType のメソッド	
「isNull()」84-42 ページ	ドキュメント・タイプが NULL かどうかをテストします。
「makeNode()」84-42 ページ	ドキュメント・タイプをノードに変換します。
「findEntity()」84-42 ページ	ドキュメント・タイプで指定のエンティティを検索します。
「findNotation()」84-43 ページ	ドキュメント・タイプで指定の表記法を検索します。
「getPublicId()」84-43 ページ	ドキュメント・タイプの公開識別子を取得します。
「getSystemId()」84-44 ページ	ドキュメント・タイプのシステム ID を取得します。
「writeExternalDTDToFile()」84-44 ページ	Document Type Definition（DTD）をファイルに書き込みます。
「writeExternalDTDToBuffer()」84-45 ページ	Document Type Definition（DTD）をバッファに書き込みます。
「writeExternalDTDToClob()」84-45 ページ	Document Type Definition（DTD）を CLOB に書き込みます。
「getName()」84-46 ページ	ドキュメント・タイプの名前を取得します。
「getEntities()」84-46 ページ	ドキュメント・タイプのエンティティのノードマップを取得します。

表 84-4 DBMS_XMLDOM のファンクションとプロシージャの要約（続き）

グループ / メソッド	説明
「getNotations()」84-47 ページ	ドキュメント・タイプの表記法のノードマップを取得します。
DOMElement のメソッド	
「isNull()」84-47 ページ	要素が NULL かどうかをテストします。
「makeNode()」84-47 ページ	要素をノードに変換します。
「getQualifiedName()」84-48 ページ	要素の修飾名を取得します。
「getNamespace()」84-48 ページ	要素の名前空間 URI を取得します。
「getLocalName()」84-48 ページ	要素のローカル名を取得します。
「getExpandedName()」84-49 ページ	要素の拡張名を取得します。
「getChildrenByTagName()」84-49 ページ	要素の子をタグ名別に取得します。
「getElementsByTagName()」84-50 ページ	サブツリー内の要素をタグ名別に取得します。
「resolveNamespacePrefix()」84-50 ページ	接頭辞を名前空間 URI に解決します。
「getTagName()」84-51 ページ	要素のタグ名を取得します。
「getAttribute()」84-51 ページ	指定した名前の属性を取得します。
「setAttribute()」84-52 ページ	指定した名前の属性を設定します。
「removeAttribute()」84-52 ページ	指定した名前の属性を削除します。
「getAttributeNode()」84-53 ページ	指定した名前の属性ノードを取得します。
「setAttributeNode()」84-53 ページ	属性ノードを要素に設定します。
「removeAttributeNode()」84-54 ページ	要素内の属性ノードを削除します。
「normalize()」84-54 ページ	要素のテキストの子を正規化します。
DOMEntity のメソッド	
「isNull()」84-54 ページ	エンティティが NULL かどうかをテストします。
「makeNode()」84-55 ページ	エンティティをノードに変換します。
「getPublicId()」84-55 ページ	エンティティの公開識別子を取得します。
「getSystemId()」84-55 ページ	エンティティのシステム ID を取得します。

表 84-4 DBMS_XMLDOM のファンクションとプロシージャの要約（続き）

グループ / メソッド	説明
「getNotationName()」84-56 ページ	エンティティの表記法の名前を取得します。
DOMEntityReference のメソッド	
「isNull()」84-56 ページ	実体参照が NULL かどうかをテストします。
「makeNode()」84-56 ページ	実体参照をノードに変換します。
DOMNotation のメソッド	
「isNull()」84-57 ページ	表記法が NULL かどうかをテストします。
「makeNode()」84-57 ページ	表記法をノードに変換します。
「getPublicId()」84-57 ページ	表記法の公開識別子を取得します。
「getSystemId()」84-58 ページ	表記法のシステム ID を取得します。
DOMProcessingInstruction のメソッド	
「isNull()」84-58 ページ	処理命令が NULL かどうかをテストします。
「makeNode()」84-58 ページ	処理命令をノードに変換します。
「getData()」84-59 ページ	処理命令のデータを取得します。
「getTarget()」84-59 ページ	処理命令のターゲットを取得します。
「setData()」84-59 ページ	処理命令のデータを設定します。
DOMText のメソッド	
「isNull()」84-60 ページ	テキストが NULL かどうかをテストします。
「makeNode()」84-60 ページ	テキストをノードに変換します。
「splitText()」84-60 ページ	テキスト・ノードの内容を 2 つのテキスト・ノードに分割します。
DOMDocument のメソッド	
「isNull()」84-61 ページ	ドキュメントが NULL かどうかをテストします。
「makeNode()」84-61 ページ	ドキュメントをノードに変換します。
「newDOMDocument()」84-61 ページ	新規ドキュメントを作成します。
「freeDocument()」84-62 ページ	ドキュメントを解放します。
「getVersion()」84-62 ページ	ドキュメントのバージョンを取得します。
「setVersion()」84-62 ページ	ドキュメントのバージョンを設定します。

表 84-4 DBMS_XMLDOM のファンクションとプロシージャの要約（続き）

グループ / メソッド	説明
「getCharset()」84-63 ページ	ドキュメントのキャラクタ・セットを取得します。
「setCharset()」84-63 ページ	ドキュメントのキャラクタ・セットを設定します。
「getStandalone()」84-63 ページ	ドキュメントがスタンドアロンとして指定されているかどうかを取得します。
「setStandalone()」84-64 ページ	ドキュメントをスタンドアロンに設定します。
「writeToFile()」84-64 ページ	ドキュメントをファイルに書き込みます。
「writeToBuffer()」84-65 ページ	ドキュメントをバッファに書き込みます。
「writeToClob()」84-65 ページ	ドキュメントを CLOB に書き込みます。
「writeExternalDTDToFile()」84-66 ページ	ドキュメントの DTD をファイルに書き込みます。
「writeExternalDTDToBuffer()」84-67 ページ	ドキュメントの DTD をバッファに書き込みます。
「writeExternalDTDToClob()」84-67 ページ	ドキュメントの DTD を CLOB に書き込みます。
「getDoctype()」84-68 ページ	ドキュメントの DTD を取得します。
「getImplementation()」84-68 ページ	DOM の実装を取得します。
「getDocumentElement()」84-69 ページ	ドキュメントのルート要素を取得します。
「createElement()」84-69 ページ	新規の要素を作成します。
「createDocumentFragment()」84-69 ページ	新規のドキュメント・フラグメントを作成します。
「createTextNode()」84-70 ページ	テキスト・ノードを作成します。
「createComment()」84-70 ページ	コメント・ノードを作成します。
「createCDATASection()」84-70 ページ	CDataSection ノードを作成します。
「createProcessingInstruction()」84-71 ページ	処理命令を作成します。
「createAttribute()」84-71 ページ	属性を作成します。
「createEntityReference()」84-72 ページ	実体参照を作成します。
「getElementsByTagName()」84-72 ページ	要素のノード・リストをタグ名別に取得します。

DOMNode のメソッド

isNull()

指定した DOMNode が NULL かどうかをチェックします。NULL の場合は TRUE を戻し、そうでない場合は FALSE を戻します。

構文

```
FUNCTION isNull( n DOMNode) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
n	(IN)	チェックする DOMNode。

makeAttr()

指定した DOMNode を DOMAttr に変換し、その DOMAttr を戻します。

構文

```
FUNCTION makeAttr( n DOMNode) RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeCDataSection()

指定した DOMNode を DOMCDataSection に変換し、その DOMCDataSection を戻します。

構文

```
FUNCTION makeCDataSection( n DOMNode) RETURN DOMCDataSection;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeCharacterData()

指定した DOMNode を DOMCharacterData に変換し、その DOMCharacterData を戻します。

構文

```
FUNCTION makeCharacterData( n DOMNode) RETURN DOMCharacterData;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeComment()

指定した DOMNode を DOMComment に変換し、その DOMComment を戻します。

構文

```
FUNCTION makeComment(n DOMNode) RETURN DOMComment;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeDocumentFragment()

指定した DOMNode を DOMDocumentFragment に変換し、その DOMDocumentFragment を戻します。

構文

```
FUNCTION makeDocumentFragment( n DOMNode) RETURN DOMDocumentFragment;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeDocumentType()

指定した DOMNode を DOMDocumentType に変換し、その DOMDocumentType を戻します。

構文

```
FUNCTION makeDocumentType(n DOMNode) RETURN DOMDocumentType;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeElement()

指定した DOMNode を DOMElement に変換し、その DOMElement を戻します。

構文

```
FUNCTION makeElement(n DOMNode) RETURN DOMElement;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeEntity()

指定した DOMNode を DOMEntity に変換し、その DOMEntity を戻します。

構文

```
FUNCTION makeEntity(n DOMNode) RETURN DOMEntity;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeEntityReference()

指定した DOMNode を DOMEntityReference に変換し、その DOMEntityReference を戻します。

構文

```
FUNCTION makeEntityReference(n DOMNode) RETURN DOMEntityReference;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeNotation()

指定した DOMNode を DOMNotation に変換し、その DOMNotation を戻します。

構文

```
FUNCTION makeNotation(n DOMNode) RETURN DOMNotation;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeProcessingInstruction()

指定した DOMNode を DOMProcessingInstruction に変換し、その DOMProcessingInstruction を戻します。

構文

```
FUNCTION makeProcessingInstruction( n DOMNode)
                                    RETURN DOMProcessingInstruction;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeText()

指定した DOMNode を DOMText に変換し、その DOMText を戻します。

構文

```
FUNCTION makeText(n DOMNode) RETURN DOMText;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

makeDocument()

指定した DOMNode を DOMDocument に変換し、その DOMDocument を戻します。

構文

```
FUNCTION makeDocument(n DOMNode) RETURN DOMDocument;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。

writeToFile()

XML ノードを指定のファイルに書き込みます。指定するオプションは、次の表のとおりです。

構文	説明

`PROCEDURE writeToFile(`
`  n DOMNode,`
`  fileName VARCHAR2);`
	データベース・キャラクタ・セットを使用して、XML ノードを指定のファイルに書き込みます。

`PROCEDURE writeToFile(`
`  n DOMNode,`
`  fileName VARCHAR2,`
`  charset  VARCHAR2);`
	指定したキャラクタ・セットを使用して、XML ノードを指定のファイルに書き込みます。このキャラクタ・セットは、個別のパラメータとして渡されます。

パラメータ	IN / OUT	説明
n	(IN)	DOMNode。
fileName	(IN)	書込み先のファイル。
charset	(IN)	指定したキャラクタ・セット。

writeToBuffer()

XML ノードを指定のバッファに書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2);	データベース・キャラクタ・セットを使用して、XML ノードを指定のバッファに書き込みます。
PROCEDURE writeToBuffer(n DOMNode, buffer IN OUT VARCHAR2, charset VARCHAR2);	指定したキャラクタ・セットを使用して、XML ノードを指定のバッファに書き込みます。このキャラクタ・セットは、個別のパラメータとして渡されます。

パラメータ	IN / OUT	説明
n	(IN)	DOMNode。
buffer	(IN/OUT)	書込み先のバッファ。
charset	(IN)	指定したキャラクタ・セット。

writeToClob()

XML ノードを指定の CLOB に書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeToClob(n DOMNode, cl IN OUT CLOB);	データベース・キャラクタ・セットを使用して、XML ノードを指定の CLOB に書き込みます。
PROCEDURE writeToClob(n DOMNode, cl IN OUT CLOB, charset VARCHAR2);	指定したキャラクタ・セットを使用して、XML ノードを指定の CLOB に書き込みます。このキャラクタ・セットは、個別のパラメータとして渡されます。

パラメータ	IN / OUT	説明
n	(IN)	DOMNode。
cl	(IN/OUT)	書込み先の CLOB。
charset	(IN)	指定したキャラクタ・セット。

getNodeName()

ノードのタイプに応じて、ノードの名前を取得します。

構文

```
FUNCTION getNodeName(n DOMNode) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getNodeValue()

ノードのタイプに応じて、ノードの値を取得します。

構文

```
FUNCTION getNodeValue(n DOMNode) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

setNodeValue()

ノードのタイプに応じて、ノードの値を設定します。ノードの値が NULL に定義されている場合は、この値を設定しても無効になります。

構文

```
PROCEDURE setNodeValue( n DOMNode,
                        nodeValue IN VARCHAR2);
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。
nodeValue	IN	ノードに設定する値。

getNodeType()

基礎となるオブジェクトのタイプを表すコードを取得します。

構文

```
FUNCTION getNodeType(n DOMNode) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getParentNode()

ノードの親を取得します。すべてのノード（Attr、Document、DocumentFragment、Entity および Notation を除く）は親を持つことができます。ただし、作成したノードがツリーに追加されていない場合、またはノードがツリーから削除されている場合は、NULL が取得されます。

構文

```
FUNCTION getParentNode(n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getChildNodes()

このノードのすべての子が含まれた NodeList を取得します。子がない場合は、ノードを含まない NodeList が取得されます。

構文

```
FUNCTION getChildNodes(n DOMNode) RETURN DOMNodeList;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getFirstChild()

ノードの最初の子を取得します。該当するノードがない場合は、NULL を戻します。

構文

```
FUNCTION getFirstChild( n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getLastChild()

ノードの最後の子を取得します。該当するノードがない場合は、NULL を戻します。

構文

```
FUNCTION getLastChild( n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getPreviousSibling()

このノードの直前のノードを取得します。該当するノードがない場合は、NULL を戻します。

構文

```
FUNCTION getPreviousSibling( n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getNextSibling()

このノードの直後に続くノードを取得します。該当するノードがない場合は、NULL を戻します。

構文

```
FUNCTION getNextSibling( n DOMNode) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getAttributes()

ノードの属性を含む NamedNodeMap（ノードが要素の場合）、または NULL（それ以外の場合）を取得します。

構文

```
FUNCTION getAttributes( n DOMNode) RETURN DOMNamedNodeMap;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

getOwnerDocument()

ノードに関連付けられたドキュメント・オブジェクトを取得します。このドキュメント・オブジェクトは、新規ノードの作成にも使用されます。このノードが、いずれのドキュメントでも未使用の Document または DocumentType である場合は、NULL が取得されます。

構文

```
FUNCTION getOwnerDocument( n DOMNode) RETURN DOMDocument;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

insertBefore()

ノード newChild を、既存の子ノード refChild の前に挿入します。refChild が NULL の場合、newChild は子リストの最後に挿入されます。

newChild が DocumentFragment オブジェクトの場合は、すべての子が、同じ順序で refChild の前に挿入されます。newChild がすでにツリーに追加されている場合は、そのノードが最初に削除されます。

構文

```
FUNCTION insertBefore( n DOMNode,
                       newChild IN DOMNode,
                       refChild IN DOMNode)
                       RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。
newChild	IN	DOMNode n に挿入する子。
refChild	IN	newChild を挿入する直前の参照ノード。

replaceChild()

子ノード oldChild を、子ノード・リスト内の newChild に置換し、その oldChild ノードを戻します。

newChild が DocumentFragment オブジェクトの場合、oldChild は、すべての DocumentFragment の子に置換（同じ順序で挿入）されます。newChild がすでにツリーに追加されている場合は、そのノードが最初に削除されます。

構文

```
FUNCTION replaceChild( n DOMNode,
                       newChild IN DOMNode,
                       oldChild IN DOMNode)
                       RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。
newChild	IN	oldChild と置換する新規の子。
oldChild	IN	置換するノード n の子。

removeChild()

oldChild が示す子ノードを子ノード・リストから削除し、その子ノードを戻します。

構文

```
FUNCTION removeChild( n DOMNode,
                      oldChild IN DOMNode)
                      RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。
oldCHild	IN	削除するノード n の子。

appendChild()

ノード newChild を、このノードの子ノード・リストの最後に追加します。newChild がすでにツリーに追加されている場合は、そのノードが最初に削除されます。

構文

```
FUNCTION appendChild( n DOMNode,
                      newChild IN DOMNode)
                      RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。
newChild	IN	ノード n の子ノード・リストに追加する子。

hasChildNodes()

このノードに子があるかどうかを戻します。

構文

```
FUNCTION hasChildNodes( n DOMNode) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。

cloneNode()

このノードの複製を戻します。つまり、ノードの汎用コピー・コンストラクタとして機能します。複製のノードに親はないため、parentNode は NULL になります。

要素のクローニングによって、すべての属性とその値（デフォルトの属性を示すために XML プロセッサで生成された値を含む）がコピーされます。ただし、このメソッドは、ディープ・クローンの場合を除き、格納されているテキストをコピーしません。これは、テキストが子のテキスト・ノードに格納されているためです。属性の直接クローニングでは、要素のクローニング操作の一部としてクローニングする場合とは対照的に、指定した属性（TRUE に指定）が戻されます。他のタイプのノードをクローニングすると、単にそのノードのコピーが戻されます。

読取り専用のサブツリーのクローニングでは、読取り / 書込み可能なコピーが作成されます。ただし、EntityReference のクローンの子は、読取り専用になります。さらに、未指定の Attr ノードのクローンが指定されます。Document、DocumentType、Entity および Notation の各ノードは、実装に依存します。

構文

```
FUNCTION cloneNode( n DOMNode,
                    deep boolean)
                    RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	IN	DOMNode。
deep	IN	子がクローニング対象かどうかを判断するブール値。

DOMNamedNodeMap のメソッド

isNull()

指定した DOMNamedNodeMap が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( nnm DOMNamedNodeMap) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
nnm	(IN)	チェックする DOMNameNodeMap。

getNamedItem()

指定した名前のノードを取得します。

構文

```
FUNCTION getNamedItem( nnm DOMNamedNodeMap,
                       name IN VARCHAR2)
                       RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nnm	IN	DOMNamedNodeMap。
name	IN	取得する項目の名前。

setNamedItem()

nodeName 属性を使用して、ノードを追加します。指定した名前のノードがすでにマップ内に存在する場合、そのノードは新規のノードに置換されます。

nodeName 属性は、ノードが格納されている名前を導出するために使用します。したがって、特定タイプの複数のノード（特殊な文字列値を持つ）は、名前がクラッシュするため格納できません。このため、ノードに別名を付けることをお薦めします。

構文

```
FUNCTION setNamedItem( nnm DOMNamedNodeMap,
                       arg IN DOMNode)
                       RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nnm	IN	DOMNamedNodeMap。
arg	IN	NodeName 属性を使用して追加するノード。

removeNamedItem()

指定した名前のノードを削除します。要素に添付した属性がこのマップに含まれ、削除した属性にデフォルト値があることが判明している場合、属性は、デフォルト値、対応する名前空間 URI、ローカル名および接頭辞（該当する場合）とともに表示されます。

構文

```
FUNCTION removeNamedItem( nnm DOMNamedNodeMap,
                          name IN VARCHAR2)
                          RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nnm	IN	DOMNamedNodeMap。
name	IN	マップから削除する項目の名前。

item()

index パラメータに対応する、マップ内の項目を戻します。索引がマップ内のノード数以上の場合は、NULL を戻します。

構文

```
FUNCTION item( nnm DOMNamedNodeMap,
               index IN NUMBER)
               RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nnm	IN	DOMNamedNodeMap。
index	IN	取得する項目があるノード・マップ内の索引。

getLength()

マップ内のノード数。有効な子ノード索引の範囲は、0 ～（長さ -1）です。

構文

```
FUNCTION getLength( nnm DOMNamedNodeMap) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
nnm	IN	DOMNamedNodeMap。

DOMNodeList のメソッド

isNull()

指定した DOMNodeList が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( nl DOMNodeList) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
nl	(IN)	チェックする DOMNodeList。

item()

index パラメータに対応する、コレクション内の項目を戻します。索引がリスト内のノード数以上の場合は、NULL を戻します。

構文

```
FUNCTION item( nl DOMNodeList,
               index IN NUMBER)
               RETURN DOMNode;
```

パラメータ	IN / OUT	説明
nl	IN	DOMNodeList。
index	IN	取得する項目のノード・リスト内の索引。

getLength()

リスト内のノード数。有効な子ノード索引の範囲は、0 ～（長さ -1）です。

構文

```
FUNCTION getLength( nl DOMNodeList) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
nl	IN	DOMNodeList。

DOMAttr のメソッド

isNull()

指定した DOMAttr が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( a DOMAttr) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
a	(IN)	チェックする DOMAttr。

makeNode()

指定した DOMAttr を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( a DOMAttr) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
a	(IN)	変換する DOMAttr。

getQualifiedName()

DOMAttr の修飾名を戻します。

構文

```
FUNCTION getQualifiedName( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	(IN)	DOMAttr。

getNamespace()

DOMAttr の名前空間を戻します。

構文

```
FUNCTION getNamespace( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	(IN)	DOMAttr。

getLocalName()

DOMAttr のローカル名を戻します。

構文

```
FUNCTION getLocalName( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	(IN)	DOMAttr。

getExpandedName()

DOMAttr の拡張名を戻します。

構文

```
FUNCTION getExpandedName( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	(IN)	DOMAttr。

getName()

属性の名前を戻します。

構文

```
FUNCTION getName( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	IN	DOMAttr。

getSpecified()

元のドキュメントで属性の値が明示的に指定されている場合は TRUE、それ以外の場合は FALSE になります。

構文

```
FUNCTION getSpecified( a DOMAttr) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
a	IN	DOMAttr。

getValue()

属性の値を取得します。

構文

```
FUNCTION getValue( a DOMAttr) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
a	IN	DOMAttr。

setValue()

属性の値を設定します。

構文

```
PROCEDURE setValue( a DOMAttr,
                    value IN VARCHAR2);
```

パラメータ	IN / OUT	説明
a	IN	DOMAttr。
value	IN	属性に設定する値。

DOMCDataSection のメソッド

isNull()

指定した DOMCDataSection が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( cds DOMCDataSection) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
cds	(IN)	チェックする DOMCDataSection。

makeNode()

DOMCDataSection を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( cds DOMCDataSection) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
cds	(IN)	変換する DOMCDataSection。

DOMCharacterData のメソッド

isNull()

指定した DOMCharacterData が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( cd DOMCharacterData) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
cd	(IN)	チェックする DOMCharacterData。

makeNode()

指定の DOMCharacterData を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( cd DOMCharacterData) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
cd	(IN)	変換する DOMCharacterData。

getData()

インタフェースを実装するノードの文字データを取得します。

構文

```
FUNCTION getData( cd DOMCharacterData) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
cd	IN	DOMCharacterData。

setData()

インタフェースを実装するノードの文字データを設定します。

構文

```
PROCEDURE setData( cd DOMCharacterData,
                   data IN VARCHAR2);
```

パラメータ	IN / OUT	説明
cd	IN	DOMCharacterData。
data	IN	ノードに設定するデータ。

getLength()

データと substringData() メソッドで使用可能な 16 ビット単位の数。0（ゼロ）の値を設定できます。この場合、CharacterData ノードは空になります。

構文

```
FUNCTION getLength( cd DOMCharacterData) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
cd	IN	DOMCharacterData。

substringData()

データの範囲をノードから抽出します。

構文

```
FUNCTION substringData( cd DOMCharacterData,
                        offset IN NUMBER,
                        cnt IN NUMBER)
                        RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
cd	IN	DOMCharacterData。
offset	IN	取得するデータの開始オフセット。
cnt	IN	取得するデータの（オフセットからの）文字数。

appendData()

ノードの文字データの最後に文字列を追加します。データが正常に追加されると、連結データと指定の文字列引数にアクセスできます。

構文

```
PROCEDURE appendData( cd DOMCharacterData,
                      arg IN VARCHAR2);
```

パラメータ	IN / OUT	説明
cd	IN	DOMCharacterData。
arg	IN	既存のデータに追加するデータ。

insertData()

指定した 16 ビット単位のオフセットに文字列を挿入します。

構文

```
PROCEDURE insertData( cd DOMCharacterData,
                      offset IN NUMBER,
                      arg IN VARCHAR2);
```

パラメータ	IN / OUT	説明
cd	IN	DOMCharacterData。
offset	IN	データを挿入するオフセット。
arg	IN	挿入する値。

deleteData()

16 ビット単位の範囲をノードから削除します。正常に削除されると、データと長さに変更が反映されます。

構文

```
PROCEDURE deleteData( cd DOMCharacterData,
                      offset IN NUMBER,
                      cnt IN NUMBER);
```

パラメータ	IN / OUT	説明
cd	IN	DOMCharacterData。
offset	IN	データを削除するオフセット。
cnt	IN	削除する（オフセット以降の）文字数。

replaceData()

16 ビット単位の範囲を置換します。正常に置換されると、データと長さに変更が反映されます。

構文

```
PROCEDURE replaceData( cd DOMCharacterData,
                       offset IN NUMBER,
                       cnt IN NUMBER,
                       arg IN VARCHAR2);
```

パラメータ	IN / OUT	説明
cd	IN	DOMCharacterData。
offset	IN	置換するオフセット。
cnt	IN	置換する文字数。
arg	IN	置換する値。

DOMComment のメソッド

isNull()

指定した DOMComment が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( com DOMComment) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
com	(IN)	チェックする DOMComment。

makeNode()

指定した DOMComment を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( com DOMComment) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
com	(IN)	変換する DOMComment。

DOMImplementation のメソッド

isNull()

指定した DOMImplementation が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( di DOMImplementation) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
di	(IN)	チェックする DOMImplementation。

hasFeature()

DOMImplementation で特定の機能が実装されているかどうかをテストします。

構文

```
FUNCTION hasFeature( di DOMImplementation,
                    feature IN VARCHAR2,
                    version IN VARCHAR2)
                    RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
di	IN	DOMImplementation。
feature	IN	チェックする機能。
version	IN	チェックする DOM のバージョン。

DOMDocumentFragment のメソッド

isNull()

指定した DOMDocumentFragment が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( df DOMDocumentFragment) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
df	(IN)	チェックする DOMDocumentFragment。

makeNode()

指定した DOMDocumentFragment を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( df DOMDocumentFragment) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
df	(IN)	変換する DOMDocumentFragment。

DOMDocumentType のメソッド

isNull()

指定した DOMDocumentType が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( dt DOMDocumentType) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
dt	(IN)	チェックする DOMDocumentType。

makeNode()

指定した DOMDocumentType を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( dt DOMDocumentType) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
dt	(IN)	変換する DOMDocumentType。

findEntity()

指定の DTD でエンティティを検索し、見つかった場合はそのエンティティを戻します。

構文

```
FUNCTION findEntity( dt      DOMDocumentType,
                     name    VARCHAR2,
                     par     BOOLEAN)
                     RETURN  DOMEntity;
```

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
name	(IN)	検索するエンティティ。
par	(IN)	エンティティのタイプを示すフラグ。パラメータ・エンティティの場合は TRUE、通常のエンティティの場合は FALSE になります。

findNotation()

指定の DTD で表記法を検索し、見つかった場合はその表記法を戻します。

構文

```
FUNCTION findNotation( dt   DOMDocumentType,
                       name VARCHAR2)
                       RETURN DOMNotation;
```

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
name	(IN)	検索する表記法。

getPublicId()

指定した DTD の公開識別子を戻します。

構文

```
FUNCTION getPublicId( dt DOMDocumentType) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
dt	(IN)	DTD。

getSystemId()

指定した DTD のシステム ID を戻します。

構文

```
FUNCTION getSystemId( dt DOMDocumentType) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
dt	(IN)	DTD。

writeExternalDTDToFile()

DTD を指定のファイルに書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2);	データベース・キャラクタ・セットを使用して、DTD を指定のファイルに書き込みます。
PROCEDURE writeExternalDTDToFile(dt DOMDocumentType, fileName VARCHAR2, charset VARCHAR2);	指定したキャラクタ・セットを使用して、DTD を指定のファイルに書き込みます。

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
fileName	(IN)	書込み先のファイル。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToBuffer()

DTD を指定のバッファに書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2);	データベース・キャラクタ・セットを使用して、DTD を指定のバッファに書き込みます。
PROCEDURE writeExternalDTDToBuffer(dt DOMDocumentType, buffer IN OUT VARCHAR2, charset VARCHAR2);	指定したキャラクタ・セットを使用して、DTD を指定のバッファに書き込みます。

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
buffer	(IN/OUT)	書込み先のバッファ。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToClob()

DTD を指定の CLOB に書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB);	データベース・キャラクタ・セットを使用して、DTD を指定の CLOB に書き込みます。
PROCEDURE writeExternalDTDToClob(dt DOMDocumentType, cl IN OUT CLOB, charset VARCHAR2);	指定したキャラクタ・セットを使用して、DTD を指定の CLOB に書き込みます。

パラメータ	IN / OUT	説明
dt	(IN)	DTD。
cl	(IN/OUT)	書込み先の CLOB。
charset	(IN)	キャラクタ・セット。

getName()

DTD の名前、または DOCTYPE キーワードの直後の名前を取得します。

構文

```
FUNCTION getName( dt DOMDocumentType) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
dt	IN	DOMDocumentType。

getEntities()

DTD で宣言された一般的なエンティティ（外部と内部の両方）を含む NamedNodeMap を取得します。

構文

```
FUNCTION getEntities( dt DOMDocumentType) RETURN DOMNamedNodeMap;
```

パラメータ	IN / OUT	説明
dt	IN	DOMDocumentType。

getNotations()

DTD で宣言された表記法を含む NamedNodeMap を取得します。

構文

```
FUNCTION getNotations( dt DOMDocumentType) RETURN DOMNamedNodeMap;
```

パラメータ	IN / OUT	説明
dt	IN	DOMDocumentType。

DOMElement のメソッド

isNull()

指定した DOMElement が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( elem DOMElement) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
elem	(IN)	チェックする DOMElement。

makeNode()

指定した DOMElement を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( elem DOMElement) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
elem	(IN)	変換する DOMElement。

getQualifiedName()

DOMElement の修飾名を戻します。

構文

```
FUNCTION getQualifiedName( elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。

getNamespace()

DOMElement の名前空間を戻します。

構文

```
FUNCTION getNamespace( elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。

getLocalName()

DOMElement のローカル名を戻します。

構文

```
FUNCTION getLocalName( elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。

getExpandedName()

DOMElement の拡張名を戻します。

構文

```
FUNCTION getExpandedName( elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。

getChildrenByTagName()

DOMElement の子を戻します。指定するオプションは、次の表のとおりです。

構文	説明
FUNCTION getChildrenByTagName(elem DOMElement, name IN VARCHAR2) RETURN DOMNodeList;	タグ名を指定して、DOMElement の子を戻します。
FUNCTION getChildrenByTagName(elem DOMElement, name IN VARCHAR2, ns VARCHAR2) RETURN DOMNodeList;	タグ名と名前空間を指定して、DOMElement の子を戻します。

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
name	(IN)	タグ名。* によってすべてのタグが指定されます。
ns	(IN)	名前空間。

getElementsByTagName()

DOMElement の要素の子を戻します。指定するオプションは、次の表のとおりです。

構文	説明
FUNCTION getElementsByTagName(elem DOMElement, name IN VARCHAR2) RETURN DOMNodeList;	タグ名を指定して、DOMElement の要素の子を戻します。
FUNCTION getElementsByTagName(elem DOMElement, name IN VARCHAR2, ns VARCHAR2) RETURN DOMNodeList;	タグ名と名前空間を指定して、DOMElement の要素の子を戻します。

パラメータ

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
name	(IN)	タグ名。* によってすべてのタグが指定されます。
ns	(IN)	名前空間。

resolveNamespacePrefix()

指定した名前空間の接頭辞を解決し、解決済みの名前空間を戻します。

構文

```
FUNCTION resolveNamespacePrefix( elem DOMElement,
                                 prefix VARCHAR2)
                                 RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
prefix	(IN)	名前空間の接頭辞。

getTagName()

DOMElement の名前を戻します。

構文

```
FUNCTION getTagName(elem DOMElement) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。

getAttribute()

DOMElement の属性値を名前別に戻します。

構文

```
FUNCTION getAttribute( elem DOMElement,
                       name IN VARCHAR2)
                       RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
name	(IN)	属性名。* によってすべての属性が指定されます。

setAttribute()

DOMElement の属性値を名前別に設定します。

構文

```
PROCEDURE setAttribute( elem DOMElement,
                        name IN VARCHAR2,
                        value IN VARCHAR2);
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
name	(IN)	属性名。* によってすべての属性が指定されます。
value	(IN)	属性値。

removeAttribute()

DOMElement から属性を名前別に削除します。

構文

```
PROCEDURE removeAttribute( elem DOMElement,
                           name IN VARCHAR2);
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
name	(IN)	属性名。* によってすべての属性が指定されます。

getAttributeNode()

DOMElement の属性ノードを名前別に戻します。

構文

```
FUNCTION getAttributeNode( elem DOMElement,
                           name IN VARCHAR2)
                           RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
name	(IN)	属性名。* によってすべての属性が指定されます。

setAttributeNode()

新規の属性ノードを DOMElement に追加します。

構文

```
FUNCTION setAttributeNode( elem DOMElement,
                           newAttr IN DOMAttr)
                           RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
newAttr	(IN)	新規の DOMAttr。

removeAttributeNode()

指定した属性ノードを DOMElement から削除します。

構文

```
FUNCTION removeAttributeNode( elem DOMElement,
                              oldAttr IN DOMAttr)
                              RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。
oldAttr	(IN)	古い DOMAttr。

normalize()

DOMElement のテキストの子を正規化します。

構文

```
PROCEDURE normalize( elem DOMElement);
```

パラメータ	IN / OUT	説明
elem	(IN)	DOMElement。

DOMEntity のメソッド

isNull()

指定した DOMEntity が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( ent DOMEntity) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
ent	(IN)	チェックする DOMEntity。

makeNode()

指定した DOMEntity を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( ent DOMEntity) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
ent	(IN)	変換する DOMEntity。

getPublicId()

DOMEntity の公開識別子を戻します。

構文

```
FUNCTION getPublicId( ent DOMEntity) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
ent	(IN)	DOMEntity。

getSystemId()

DOMEntity のシステム ID を戻します。

構文

```
FUNCTION getSystemId( ent DOMEntity) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
ent	(IN)	DOMEntity。

getNotationName()

DOMEntity の表記法の名前を戻します。

構文

```
FUNCTION getNotationName( ent DOMEntity) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
ent	(IN)	DOMEntity。

DOMEntityReference のメソッド

isNull()

指定した DOMEntityReference が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( eref DOMEntityReference) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
eref	(IN)	チェックする DOMEntityReference。

makeNode()

DOMEntityReference を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( eref DOMEntityReference) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
eref	(IN)	変換する DOMEntityReference。

DOMNotation のメソッド

isNull()

指定した DOMNotation が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( n DOMNotation) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
n	(IN)	チェックする DOMNotation。

makeNode()

DOMNotation を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( n DOMNotation) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNotation。

getPublicId()

DOMNotation の公開識別子を戻します。

構文

```
FUNCTION getPublicId( n DOMNotation) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
n	(IN)	DOMNotation。

getSystemId()

DOMNotation のシステム ID を戻します。

構文

```
FUNCTION getSystemId( n DOMNotation) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
n	(IN)	DOMNotation。

DOMProcessingInstruction のメソッド

isNull()

指定した DOMProcessingInstruction が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( pi DOMProcessingInstruction) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
pi	(IN)	チェックする DOMProcessingInstruction。

makeNode()

DOMProcessingInstruction を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( pi DOMProcessingInstruction) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
pi	(IN)	変換する DOMProcessingInstruction。

getData()

DOMProcessingInstruction のコンテンツ・データを戻します。

構文

```
FUNCTION  getData( pi DOMProcessingInstruction) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
pi	(IN)	DOMProcessingInstruction。

getTarget()

DOMProcessingInstruction のターゲットを戻します。

構文

```
FUNCTION getTarget( pi DOMProcessingInstruction) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
pi	(IN)	DOMProcessingInstruction。

setData()

DOMProcessingInstruction のコンテンツ・データを設定します。

構文

```
PROCEDURE setData( pi DOMProcessingInstruction,
                   data IN VARCHAR2);
```

パラメータ	IN / OUT	説明
pi	(IN)	DOMProcessingInstruction。
data	(IN)	新規の処理命令のコンテンツ・データ。

DOMText のメソッド

isNull()

指定した DOMText が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( t DOMText) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
t	(IN)	チェックする DOMText。

makeNode()

DOMText を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( t DOMText) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
t	(IN)	変換する DOMText。

splitText()

指定したオフセットで、DOMText ノードを 2 つの DOMText ノードに分割します。

構文

```
FUNCTION splitText( t DOMText,
                    offset IN NUMBER)
                    RETURN DOMText;
```

パラメータ	IN / OUT	説明
t	(IN)	DOMText。
offset	(IN)	分割を行うオフセット。

DOMDocument のメソッド

isNull()

指定した DOMDocument が NULL かどうかをチェックし、NULL の場合は TRUE を、それ以外の場合は FALSE を戻します。

構文

```
FUNCTION isNull( doc DOMDocument) RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
doc	(IN)	チェックする DOMDocument。

makeNode()

DOMDocument を DOMNode に変換し、その DOMNode を戻します。

構文

```
FUNCTION makeNode( doc DOMDocument) RETURN DOMNode;
```

パラメータ	IN / OUT	説明
doc	(IN)	変換する DOMDocument。

newDOMDocument()

新規の DOMDocument インスタンスを戻します。

構文

```
FUNCTION newDOMDocument RETURN DOMDocument;
```

freeDocument()

DOMDocument オブジェクトを解放します。

構文

```
PROCEDURE freeDocument( doc DOMDocument);
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。

getVersion()

XML 文書のバージョン情報を戻します。

構文

```
FUNCTION getVersion( doc DOMDocument) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。

setVersion()

XML 文書のバージョン情報を設定します。

構文

```
PROCEDURE setVersion( doc     DOMDocument,
                      version VARCHAR2);
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
version	(IN)	バージョン情報。

getCharset()

XML 文書のキャラクタ・セットを取得します。

構文

```
FUNCTION getCharset( doc DOMDocument) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。

setCharset()

XML 文書のキャラクタ・セットを設定します。

構文

```
PROCEDURE setCharset( doc DOMDocument,
                      charset VARCHAR2);
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
charset	(IN)	キャラクタ・セット。

getStandalone()

XML 文書のスタンドアロン情報を取得します。

構文

```
FUNCTION getStandalone( doc DOMDocument) RETURN VARCHAR2;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。

setStandalone()

XML 文書のスタンドアロン情報を設定します。

構文

```
PROCEDURE setStandalone( doc DOMDocument,
                         value VARCHAR2);
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
value	(IN)	スタンドアロン情報。

writeToFile()

XML 文書を指定のファイルに書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2);	データベース・キャラクタ・セットを使用して、XML 文書を指定のファイルに書き込みます。
PROCEDURE writeToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);	指定したキャラクタ・セットを使用して、XML 文書を指定のファイルに書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
filename	(IN)	書込み先のファイル。
charset	(IN)	キャラクタ・セット。

writeToBuffer()

XML 文書を指定のバッファに書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);	データベース・キャラクタ・セットを使用して、XML 文書を指定のバッファに書き込みます。
PROCEDURE writeToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);	指定したキャラクタ・セットを使用して、XML 文書を指定のバッファに書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
buffer	(IN/OUT)	書込み先のバッファ。
charset	(IN)	キャラクタ・セット。

writeToClob()

XML 文書を指定の CLOB に書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB);	データベース・キャラクタ・セットを使用して、XML 文書を指定の CLOB に書き込みます。
PROCEDURE writeToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);	指定したキャラクタ・セットを使用して、XML 文書を指定の CLOB に書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
cl	(IN/OUT)	書込み先のバッファ。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToFile()

外部 DTD を指定のファイルに書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2);	データベース・キャラクタ・セットを使用して、外部 DTD を指定のファイルに書き込みます。
PROCEDURE writeExternalDTDToFile(doc DOMDocument, fileName VARCHAR2, charset VARCHAR2);	指定したキャラクタ・セットを使用して、外部 DTD を指定のファイルに書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
fileName	(IN)	書込み先のファイル。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToBuffer()

外部 DTD を指定のバッファに書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2);	データベース・キャラクタ・セットを使用して、外部 DTD を指定のバッファに書き込みます。
PROCEDURE writeExternalDTDToBuffer(doc DOMDocument, buffer IN OUT VARCHAR2, charset VARCHAR2);	指定したキャラクタ・セットを使用して、外部 DTD を指定のバッファに書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
buffer	(IN/OUT)	書込み先のバッファ。
charset	(IN)	キャラクタ・セット。

writeExternalDTDToClob()

外部 DTD を指定の CLOB に書き込みます。指定するオプションは、次の表のとおりです。

構文	説明
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB);	データベース・キャラクタ・セットを使用して、外部 DTD を指定の CLOB に書き込みます。
PROCEDURE writeExternalDTDToClob(doc DOMDocument, cl IN OUT CLOB, charset VARCHAR2);	指定したキャラクタ・セットを使用して、外部 DTD を指定の CLOB に書き込みます。

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
cl	(IN)	書込み先の CLOB。
charset	(IN)	キャラクタ・セット。

getDoctype()

DOMDocument に関連付けられた DTD を戻します。

構文

```
FUNCTION getDoctype( doc DOMDocument) RETURN DOMDocumentType;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。

getImplementation()

DOMDocument を処理する DOMImplementation オブジェクトを戻します。

構文

```
FUNCTION getImplementation( doc DOMDocument) RETURN DOMImplementation;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。

getDocumentElement()

DOMDocument の子ノード（ドキュメント要素）を戻します。

構文

```
FUNCTION getDocumentElement( doc DOMDocument) RETURN DOMElement;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。

createElement()

DOMElement を作成します。

構文

```
FUNCTION createElement( doc DOMDocument,
                        tagName IN VARCHAR2)
                        RETURN DOMElement;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
tagName	(IN)	新規の DOMElement のタグ名。

createDocumentFragment()

DOMDocumentFragment を作成します。

構文

```
FUNCTION createDocumentFragment( doc DOMDocument) RETURN DOMDocumentFragment;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。

createTextNode()

DOMText ノードを作成します。

構文

```
FUNCTION createTextNode( doc DOMDocument,
                         data IN VARCHAR2)
                         RETURN DOMText;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
data	(IN)	DOMText ノードの内容。

createComment()

DOMComment ノードを作成します。

構文

```
FUNCTION createComment( doc DOMDocument,
                        data IN VARCHAR2)
                        RETURN DOMComment;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
data	(IN)	DOMComment ノードの内容。

createCDATASection()

DOMCDATASection ノードを作成します。

構文

```
FUNCTION createCDATASection( doc DOMDocument,
                             data IN VARCHAR2)
                             RETURN DOMCDATASection;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
data	(IN)	DOMCDATASection ノードの内容。

createProcessingInstruction()

DOMProcessingInstruction ノードを作成します。

構文

```
FUNCTION createProcessingInstruction( doc DOMDocument,
                                      target IN VARCHAR2,
                                      data IN VARCHAR2)
                                      RETURN DOMProcessingInstruction;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
target	(IN)	新規の処理命令のターゲット。
data	(IN)	新規の処理命令のコンテンツ・データ。

createAttribute()

DOMAttr ノードを作成します。

構文

```
FUNCTION createAttribute( doc DOMDocument,
                          name IN VARCHAR2)
                          RETURN DOMAttr;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
name	(IN)	新規の属性名。

createEntityReference()

DOMEntityReference ノードを作成します。

構文

```
FUNCTION createEntityReference( doc DOMDocument,
                                name IN VARCHAR2)
                                RETURN DOMEntityReference;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
name	(IN)	新規の実体参照の名前。

getElementsByTagName()

指定したタグ名を持つすべての要素の DOMNodeList を戻します。

構文

```
FUNCTION getElementsByTagName( doc DOMDocument,
                               tagname IN VARCHAR2)
                               RETURN DOMNodeList;
```

パラメータ	IN / OUT	説明
doc	(IN)	DOMDocument。
tagname	(IN)	一致させるタグの名前。

85

DBMS_XMLGEN

DBMS_XMLGEN は、SQL 問合せの結果を標準的な XML 形式に変換します。パッケージは、任意の SQL 問合せを入力として XML 形式に変換し、その結果を CLOB として戻します。

このパッケージは、C で記述されてカーネルにコンパイルされる点を除き、DBMS_XMLQUERY パッケージに類似しています。このパッケージを実行できるのは、データベース上のみです。

> **関連項目：** XML サポートの詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。
>
> XML サポートの詳細および DBMS_XMLGEN の使用方法の例は、『Oracle9*i* XML データベース開発者ガイド - Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

DMS_XMLGEN の概要

DBMS_XMLGEN は、SQL 問合せの結果を標準的な XML 形式に変換します。パッケージは、任意の SQL 問合せを入力として XML 形式に変換し、その結果を CLOB として戻します。

このパッケージは、C で記述されてカーネルにコンパイルされる点を除き、DBMS_XMLQUERY パッケージに類似しています。このパッケージを実行できるのは、データベース内のみです。

DBMS_XMLGEN のファンクションとプロシージャ

表 85-1 DBMS_XMLGEN のファンクションとプロシージャの要約

ファンクション / プロシージャ	説明
「newContext()」85-3 ページ	新しいコンテキスト・ハンドルを作成します。
「setRowTag()」85-3 ページ	結果の各行を囲む要素の名前を設定します。デフォルトのタグは ROW です。
「setRowSetTag ()」85-4 ページ	結果全体を囲む要素の名前を設定します。デフォルトのタグは ROWSET です。
「getXML()」85-4 ページ	XML 文書を取得します。
「getNumRowsProcessed()」85-5 ページ	getXML への最後のコールで処理された SQL 行の数を取得します。
「setMaxRows()」85-6 ページ	一度にフェッチされる行の最大数を設定します。
「setSkipRows()」85-6 ページ	XML を生成する前に毎回スキップされる行数を設定します。デフォルトは 0（ゼロ）です。
「setConvertSpecialChars()」85-7 ページ	XML 文字ではない $ などの特殊文字をエスケープ表示に変換するかどうかを設定します。デフォルトでは、変換します。
「convert()」85-7 ページ	XML を同等のエスケープまたはエスケープ解除の XML に変換します。
「useItemTagsForColl()」85-8 ページ	コレクション要素に対し、tag _ITEM が追加された列名のコレクションを使用するように強制します。デフォルトでは、基礎となっているオブジェクト・タイプ名をコレクションのベース要素として設定します。
「restartQUERY()」85-8 ページ	問合せを再起動し、最初からフェッチを開始します。
「closeContext()」85-9 ページ	コンテキストをクローズし、リソースをすべて解放します。

newContext()

新しいコンテキスト・ハンドルを生成して戻します。このコンテキスト・ハンドルは、結果から XML を戻すために、getXML() およびその他のファンクションで使用します。指定できるオプションは、次の表のとおりです。

構文	説明
DBMS_XMLGEN.newContext (query IN VARCHAR2) RETURN ctxHandle;	問合せから新しいコンテキスト・ハンドルを生成します。
DBMS_XMLGEN.newContext (queryString IN SYS_REFCURSOR) RETURN ctxHandle;	問合せ文字列から、新しいコンテキスト・ハンドルを PL/SQL REF カーソルの形式で生成します。

パラメータ	IN / OUT	説明
query	(IN)	VARCHAR 形式の問合せ。結果は XML に変換する必要があります。
queryString	(IN)	PL/SQL REF カーソル形式の問合せ文字列。結果は XML に変換する必要があります。

setRowTag()

すべての行を分割する要素の名前を設定します。デフォルトの名前は ROW です。ユーザーは、この名前を NULL に設定して、ROW 要素自体を抑制できます。ただし、行と行セットの両方が NULL で、出力に複数の列または行がある場合は、エラーが発生します。これは、生成された XML に最上位レベルの囲みタグがないため、無効になるためです。

構文

```
DBMS_XMLGEN.setRowTag (
   ctx     IN ctxHandle,
   rowTag  IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得したコンテキスト・ハンドル。
rowTag	(IN)	ROW 要素の名前。NULL を渡すと、ROW 要素を表示しないことを示します。

setRowSetTag ()

ドキュメントのルート要素名を設定します。デフォルトの名前は ROWSET です。ユーザーは、rowSetTag を NULL に設定して、この要素の出力を抑制できます。ただし、行および行セットの両方が NULL で、出力に複数の列または行がある場合は、エラーが発生します。これは、生成された XML に最上位レベルの囲みタグがないため、無効になるためです。

構文

```
DBMS_XMLGEN.setRowSetTag (
   ctx        IN ctxHandle,
   rowSetTag  IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得したコンテキスト・ハンドル。
rowSetTag	(IN)	ドキュメントの要素名。NULL を渡すと、ROWSET 要素を表示しないことを示します。

getXML()

XML 文書を取得します。setSkipRows() のコールにより示された行がスキップされた場合は、setMaxRows() のコールにより指定された行の最大数（指定されていない場合は結果全体）がフェッチされ、XML に変換されます。行が取り出されたかどうかをチェックするには、getNumRowsProcessed() を使用してください。指定できるオプションは、次の表のとおりです。

構文	説明
FUNCTION DBMS_XMLGEN.getXML (ctx IN ctxHandle, clobval IN OUT NCOPY clob, dtdOrSchema IN number := NONE) RETURN boolean;	このプロシージャは、指定した行の最大数をフェッチすることにより、XML 文書を取得します。XML 文書は、渡された CLOB に追加されます。余分な CLOB のコピーを回避し、後続のコールで同じ CLOB を再利用する場合は、このバージョンの getXML() を使用してください。CLOB が再利用できるため、この getXML() コールは効果的です。
FUNCTION DBMS_XMLGEN.getXML (ctx IN ctxHandle, dtdOrSchema IN number := NONE) RETURN clob;	XML 文書を生成して一時 CLOB として戻します。このファンクションから取得した一時 CLOB は、DBMS_LOB.FREETEMPORARY のコールを使用して解放する必要があります。

構文	説明
FUNCTION DBMS_XMLGEN.getXML (sqlQuery IN VARCHAR2, dtdOrSchema IN number := NONE) RETURN clob;	SQL 問合せ文字列からの結果を XML 形式に変換し、その XML を一時 CLOB として戻します。この一時 CLOB は、後で DBMS_LOB.FREETEMPORARY のコールを使用して解放する必要があります。
FUNCTION DBMS_XMLGEN.getXMLType (ctx IN ctxhandle, dtdOrSchema IN number := NONE) RETURN sys.XMLType;	XML 文書を生成し、sys.XMLType として戻します。XMLType 操作は、ExistsNode および Extract を含む結果に対して実行できます。これは、結果のサイズが 4KB 未満の場合に、getStringVal() ファンクションを使用して、結果を文字列として取得することもできます。
FUNCTION DBMS_XMLGEN.getXMLType (sqlQuery IN VARCHAR2, dtdOrSchema IN number := NONE) RETURN sys.XMLType	SQL 問合せ文字列からの結果を XML 形式に変換し、sys.XMLType として戻します。XMLType 操作は、ExistsNode および Extract を含む結果に対して実行できます。これは、結果のサイズが 4KB 未満の場合に、getStringVal() ファンクションを使用して、結果を文字列として取得するための一方法でもあります。

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得したコンテキスト・ハンドル。
clobval	(IN/OUT)	XML 文書が追加される CLOB。
sqlQuery	(IN)	SQL 問合せ文字列。
dtdOrSchema	(IN)	DTD またはスキーマの生成を示すブール値。現在サポートされているオプションは、NONE のみです。

getNumRowsProcessed()

getXML のコールを使用した XML の生成時に処理された SQL 行の数を取得します。XML の生成前にスキップされた行は含まれません。getXML() をループでコールしている場合に、終了条件を決定するために使用します。getXML() は行がない場合でも、常に XML 文書を生成します。

構文

```
DBMS_XMLGEN.getNumRowsProcessed (
   ctx  IN ctxHandle)
RETURN NUMBER;
```

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得したコンテキスト・ハンドル。

setMaxRows()

getXML のコールを起動するたびに SQL の問合せ結果からフェッチする行の最大数を設定します。ページ番号付きの結果を生成する場合に使用します。たとえば、XML データや HTML データのページの生成時に、この maxRows パラメータを設定して、XML や HTML に変換する行の数を抑制します。

構文

```
DBMS_XMLGEN.setMaxRows (
   ctx      IN ctxHandle,
   maxRows  IN NUMBER);
```

パラメータ	IN / OUT	説明
ctx	(IN)	実行した問合せに対応するコンテキスト・ハンドル。
maxRows	(IN)	getXML へのコールごとに取得する行の最大数。

setSkipRows()

getXML ルーチンへのコールを実行するたびに、XML 出力を生成する前に任意の行数をスキップします。このユーティリティを使用して、ページ番号付きの結果をステートレス Web ページ用に生成する場合に使用します。たとえば、XML または HTML データの最初のページを生成する場合は、skipRows を 0（ゼロ）に設定します。次のセットでは、skipRows を最初のケースで取得した行数に設定します。「getNumRowsProcessed()」を参照してください。

構文

```
DBMS_XMLGEN.setSkipRows (
   ctx       IN ctxHandle,
   skipRows  IN NUMBER);
```

パラメータ	IN / OUT	説明
ctx	(IN)	実行した問合せに対応するコンテキスト・ハンドル。
skipRows	(IN)	getXML へのコールごとにスキップする行数。

setConvertSpecialChars()

XML データの特殊文字を同等のエスケープ XML に変換するかどうかを設定します。たとえば、< は < に変換されます。デフォルトでは、変換します。入力データに <、>、"、' などのエスケープが不可欠な特殊文字を含めることができない場合は、XML 処理のパフォーマンスが改善されます。文字データをスキャンして特殊文字に置換する作業は、データが多い場合は特に不経済です。

構文

```
DBMS_XMLGEN.setConvertSpecialChars (
   ctx   IN ctxHandle,
   conv  IN boolean);
```

パラメータ	IN / OUT	説明
ctx	(IN)	newContext をコールして取得したコンテキスト・ハンドル。
conv	(IN)	TRUE は、変換が必要であることを示します。

convert()

XML データを同等のエスケープまたはエスケープ解除の XML に変換し、XML CLOB データをエンコードまたはデコードされた形式で戻します。ENTITY_ENCODE を指定すると、XML データがエスケープされます。たとえば、< は < に変換されます。エスケープの解除では、これとは逆に変換されます。指定できるオプションは、次の表のとおりです。

構文	説明
DBMS_XMLGEN.convert (xmlData IN VARCHAR2, flag IN NUMBER := ENTITY_ENCODE) RETURN VARCHAR2;	文字列形式（VARCHAR2）で xmlData を使用します。
DBMS_XMLGEN.convert (xmlData IN CLOB, flag IN NUMBER := ENTITY_ENCODE) RETURN CLOB;	CLOB 形式で xmlData を使用します。

パラメータ	IN / OUT	説明
xmlData	(IN)	エンコードまたはデコードする XML CLOB データ。
flag	(IN)	フラグ設定。エンコードの場合は ENTITY_ENCODE（デフォルト）、デコードの場合は ENTITY_DECODE に設定します。

useItemTagsForColl()

コレクション要素のデフォルト名を上書きします。コレクション要素のデフォルト名は、そのタイプ名です。このファンクションを使用すると、デフォルトに上書きして _ITEM タグを追加した列名を使用できます。NUMBER のコレクションがある場合、コレクション要素のデフォルトのタグ名は NUMBER です。このファンクションを使用すると、ユーザーはこの動作を上書きして _ITEM タグを追加したコレクションの列名を生成できます。

構文

```
DBMS_XMLGEN.useItemTagsForColl (
   ctx  IN ctxHandle);
```

パラメータ	IN / OUT	説明
ctx	(IN)	コンテキスト・ハンドル。

restartQUERY()

問合せを再起動し、最初の行から XML を生成します。新しいコンテキストを作成せずに、問合せを再度実行する場合に使用できます。

構文

```
DBMS_XMLGEN.restartQUERY (ctx  IN ctxHandle);
```

パラメータ	IN / OUT	説明
ctx	(IN)	現行の問合せに対応するコンテキスト・ハンドル。

closeContext()

任意のコンテキストをクローズし、SQL カーソルなどそのコンテキストに関連付けられたすべてのリソースを解放し、バッファのバインドおよび定義を行います。このコールの後、後続の DBMS_XMLGEN ファンクション・コールでハンドルは使用できません。

構文

```
DBMS_XMLGEN.closeContext ( ctx  IN ctxHandle);
```

パラメータ	IN / OUT	説明
ctx	(IN)	クローズするコンテキスト・ハンドル。

86

DBMS_XMLPARSER

DBMS_XMLPARSER を使用すると、XML 文書の内容と構造にアクセスできます。

> **関連項目：** 詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

DBMS_XMLPARSER の概要

eXtensible Markup Language（XML）は、XML 文書と呼ばれるデータ・オブジェクトのクラスを記述します。XML 文書を処理するコンピュータ・プログラムの動作を記述する場合もあります。XML は、アプリケーション・プロファイル、つまり Standard Generalized Markup Language（SGML）の制限された形式です。XML 文書の構成は、SGML ドキュメントに準拠しています。

XML 文書は、エンティティと呼ばれる記憶単位で構成され、このエンティティには解析済みまたは未解析のデータが含まれています。解析済みデータは文字で構成され、一部は文字データを、他の一部はマークアップを形成します。マークアップは、ドキュメントの記憶域レイアウトと論理構造の記述をエンコードします。XML によって、記憶域レイアウトと論理構造に対して制約を加えるメカニズムが提供されます。

XML 文書の読込み、その内容と構造へのアクセスには、XML プロセッサと呼ばれるソフトウェア・モジュールが使用されます。XML プロセッサは、別のモジュール（アプリケーション）にかわって処理を実行しているとみなされます。XML プロセッサ（またはパーサー）の PL/SQL 実装は、W3C XML 仕様（改訂 REC-xml-19980210）に準拠しています。また、XML データの読込み方法とアプリケーションに提供する情報に関して、XML プロセッサに要求される動作が含まれています。

この PL/SQL XML パーサーのデフォルト動作は、次のとおりです。

- DOM API からアクセス可能な解析ツリーが作成されます。
- DTD が検出された場合、パーサーは検証を行い、検出されなかった場合は検証は行いません。
- エラー・ログが指定されていない場合、エラーは記録されません。ただし、解析に失敗すると、アプリケーション・エラーが発生します。

DBMS_XMLPARSER のファンクションとプロシージャ

表 86-1 DBMS_XMLPARSER のファンクションとプロシージャの要約

サブプログラム	説明
「parse()」86-4 ページ	指定の URL およびファイルに格納された XML を解析します。
「newParser()」86-4 ページ	新規のパーサー・インスタンスを戻します。
「parseBuffer()」86-4 ページ	指定のバッファに格納された XML を解析します。
「parseClob()」86-5 ページ	指定の CLOB に格納された XML を解析します。
「parseDTD()」86-5 ページ	指定の URL およびファイルに格納された DTD を解析します。
「parseDTDBuffer()」86-6 ページ	指定のバッファに格納された DTD を解析します。
「parseDTDClob()」86-6 ページ	指定の CLOB に格納された DTD を解析します。
「setBaseDir()」86-7 ページ	相対 URL の解決に使用するベース・ディレクトリを設定します。
「showWarnings()」86-7 ページ	警告をオンまたはオフにします。
「setErrorLog()」86-7 ページ	指定したファイルに送信するエラーを設定します。
「setPreserveWhitespace()」86-8 ページ	空白の保持モードを設定します。
「setValidationMode()」86-8 ページ	妥当性チェック・モードを設定します。
「getValidationMode()」86-9 ページ	妥当性チェック・モードを戻します。
「setDoctype()」86-9 ページ	DTD を設定します。
「getDoctype()」86-9 ページ	DTD を取得します。
「getDocument()」86-10 ページ	DOM のドキュメントを取得します。
「freeParser()」86-10 ページ	パーサー・オブジェクトを解放します。
「getReleaseVersion()」86-10 ページ	Oracle XML Parser for PL/SQL のリリース・バージョンを戻します。

parse()

指定の URL およびファイルに格納された XML を解析します。解析に失敗すると、アプリケーション・エラーが発生します。指定できるオプションは、次の表のとおりです。

構文	説明
FUNCTION parse(url VARCHAR2) RETURN DOMDocument;	作成された DOM のドキュメントを戻します。この構文は、パーサーのデフォルト動作が受入れ可能で、URL およびファイルの解析のみが必要な場合に使用します。
PROCEDURE parse(p Parser, url VARCHAR2);	パーサーのデフォルト動作に対する変更は、このプロシージャのコール前に有効にしておく必要があります。

パラメータ	IN / OUT	説明
url	(IN)	解析する URL およびファイルの完全パス。
p	(IN)	パーサー・インスタンス。

newParser()

新規のパーサー・インスタンスを戻します。このファンクションは、他の解析メソッドを使用する必要がある場合、パーサーのデフォルト動作の変更前にコールする必要があります。

構文

```
FUNCTION newParser RETURN Parser;
```

parseBuffer()

指定のバッファに格納された XML を解析します。パーサーのデフォルト動作に対する変更は、このプロシージャのコール前に有効にしておく必要があります。解析に失敗すると、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseBuffer( p   Parser,
                       doc VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
doc	(IN)	解析する XML 文書のバッファ。

parseClob()

指定の CLOB に格納された XML を解析します。パーサーのデフォルト動作に対する変更は、このプロシージャのコール前に有効にしておく必要があります。解析に失敗すると、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseClob( p   Parser,
                     doc CLOB);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
doc	(IN)	解析する CLOB 内の XML。

parseDTD()

指定の URL およびファイルに格納された DTD を解析します。パーサーのデフォルト動作に対する変更は、このプロシージャのコール前に有効にしておく必要があります。解析に失敗すると、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseDTD( p     Parser,
                    url   VARCHAR2,
                    root  VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
url	(IN)	解析する URL およびファイルの完全パス。
root	(IN)	ルート要素名。

parseDTDBuffer()

指定のバッファに格納された DTD を解析します。パーサーのデフォルト動作に対する変更は、このプロシージャのコール前に有効にしておく必要があります。解析に失敗すると、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseDTDBuffer( p    Parser,
                          dtd  VARCHAR2,
                          root VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
dtd	(IN)	解析する DTD のバッファ。
root	(IN)	ルート要素の名前。

parseDTDClob()

指定の CLOB に格納された DTD を解析します。パーサーのデフォルト動作に対する変更は、このプロシージャのコール前に有効にしておく必要があります。解析に失敗すると、アプリケーション・エラーが発生します。

構文

```
PROCEDURE parseDTDClob( p    Parser,
                        dtd  CLOB,
                        root VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
dtd	(IN)	解析する DTD の CLOB。
root	(IN)	ルート要素の名前。

setBaseDir()

相対 URL の解決に使用するベース・ディレクトリを設定します。解析に失敗すると、アプリケーション・エラーが発生します。

構文

```
PROCEDURE setBaseDir( p   Parser,
                      dir VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
dir	(IN)	ベース・ディレクトリとして使用するディレクトリ。

showWarnings()

警告をオンまたはオフにします。

構文

```
PROCEDURE showWarnings( p   Parser,
                        yes BOOLEAN);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
yes	(IN)	設定するモード。警告を表示する場合は TRUE に、表示しない場合は FALSE に設定します。

setErrorLog()

指定したファイルに送信するエラーを設定します。

構文

```
PROCEDURE setErrorLog( p        Parser,
                       fileName VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
fileName	(IN)	エラー・ログとして使用するファイルの完全パス。

setPreserveWhitespace()

空白の保持モードを設定します。

構文

```
PROCEDURE setPreserveWhitespace( p   Parser,
                                 yes BOOLEAN);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
yes	(IN)	設定するモード。保持する場合は TRUE に、保持しない場合は FALSE に設定します。

setValidationMode()

妥当性チェック・モードを設定します。

構文

```
PROCEDURE setValidationMode( p   Parser,
                             yes BOOLEAN);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
yes	(IN)	設定するモード。妥当性チェックを行う場合は TRUE に、妥当性チェックを行わない場合は FALSE に設定します。

getValidationMode()

妥当性チェック・モードを取得します。妥当性チェックを行う場合は TRUE、妥当性チェックを行わない場合は FALSE になります。

構文

```
FUNCTION getValidationMode( p Parser)
                          RETURN BOOLEAN;
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。

setDoctype()

パーサーが妥当性チェックで使用する DTD を設定します。ドキュメントの解析前にコールする必要があります。

構文

```
PROCEDURE setDoctype( p   Parser,
                      dtd DOMDocumentType);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。
dtd	(IN)	設定する DTD。

getDoctype()

解析済みの DTD を戻します。このファンクションをコールできるのは、DTD の解析後のみです。

構文

```
FUNCTION getDoctype( p Parser)
                    RETURN DOMDocumentType;
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。

getDocument()

パーサーによって作成された DOM のツリー・ドキュメントのルートを戻します。このファンクションをコールできるのは、ドキュメントの解析後のみです。

構文

```
FUNCTION getDocument( p Parser)
                    RETURN DOMDocument;
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。

freeParser()

パーサー・オブジェクトを解放します。

構文

```
PROCEDURE freeParser( p Parser);
```

パラメータ	IN / OUT	説明
p	(IN)	パーサー・インスタンス。

getReleaseVersion()

Oracle XML Parser for PL/SQL のリリース・バージョンを戻します。

構文

```
PROCEDURE getReleaseVersion RETURN VARCHAR2;
```

87

DBMS_XMLQUERY

DBMS_XMLGEN は C の組込み済みパッケージです。一般に、可能なかぎり DBMS_XMLQUERY のかわりに DBMS_XMLGEN を使用します。DBMS_XMLQUERY は、データベースから XML タイプへの変換の機能性を提供します。

> **関連項目：** 詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

- DBMS_XMLQuery のタイプ
- DBMS_XMLQuery の定数
- DBMS_XMLQuery のファンクションとプロシージャ

DBMS_XMLQuery の概要

この API は、DB_to_XML タイプの機能を提供します。

DBMS_XMLQuery のタイプ

表 87-1 DBMS_XMLQuery のタイプ

タイプ	説明
ctxType	問合せコンテキスト・ハンドルのタイプ。newContext() の戻り型です。

DBMS_XMLQuery の定数

表 87-2 DBMS_XMLQuery の定数

定数	説明
DB_ENCODING	DB 文字エンコーディングの使用を示します。
DEFAULT_ROWSETTAG	結果セットから生成された XML を囲む要素のタグ名（ほとんどの場合、ルート・ノードのタグ名）-- ROWSET。
DEFAULT_ERRORTAG	発生したエラーを囲むデフォルトのタグ -- ERROR。
DEFAULT_ROWIDATTR	データベース・レコードに対応する XML 要素のカーディナリティ属性のデフォルト名 -- NUM。
DEFAULT_ROWTAG	データベース・レコードに対応する要素のデフォルト・タグ名 -- ROW。
DEFAULT_DATE_FORMAT	デフォルトの日付マスク -- 'MM/dd/yyyy HH:mm:ss'。
ALL_ROWS	ALL_ROWS パラメータは、出力ですべての行が必要であることを示します。
NONE	出力に XML メタデータを含めないことを指定します（DTD や Schema などがない）。
DTD	DTD の生成が必要であることを指定します。
SCHEMA	XML SCHEMA の生成が必要であることを指定します。
LOWER_CASE	小文字のタグ名を使用します。
UPPER_CASE	大文字のタグ名を使用します。

DBMS_XMLQuery のファンクションとプロシージャ

表 87-3 DBMS_XMLQuery のファンクションとプロシージャの要約

ファンクション / プロシージャ	説明
「newContext()」87-5 ページ	問合せコンテキストを作成し、コンテキスト・ハンドルを戻します。
「closeContext()」87-5 ページ	特定の問合せコンテキストのクローズまたは割当て解除を行います。
「setRowsetTag()」87-5 ページ	XML データセットを囲むタグを設定します。
「setRowTag()」87-6 ページ	データベースに対応する XML 要素を囲むタグを設定します。
「setErrorTag()」87-6 ページ	XML エラー・ドキュメントを囲むタグを設定します。
「setRowIdAttrName()」87-6 ページ	タグを囲む行の ID 属性名を設定します。
「setRowIdAttrValue()」87-7 ページ	タグを囲む行の ID 属性に値を割り当てるスカラー列を指定します。
「setCollIdAttrName()」87-7 ページ	コレクション要素のセパレータ・タグの ID 属性名を設定します。
「useNullAttributeIndicator()」87-8 ページ	NULL を示すために XML 属性を使用するかどうかを指定します。
「useTypeForCollElemTag()」87-8 ページ	コレクション要素のタグ名としてコレクション要素のタイプ名を使用することを XSU に指示します。
「setTagCase()」87-9 ページ	生成された XML タグの大文字または小文字を指定します。
「setDateFormat()」87-9 ページ	XML 文書で生成された日付の書式を設定します。
「setMaxRows()」87-10 ページ	XML に変換される行の最大数を設定します。
「setSkipRows()」87-10 ページ	スキップする行数を設定します。
「setStylesheetHeader()」87-10 ページ	スタイルシートのヘッダーを設定します。
「setXSLT()」87-11 ページ	生成された XML に適用されるスタイルシートを登録します。
「setXSLTParam()」87-12 ページ	最上位レベルのスタイルシート・パラメータの値を設定します。
「removeXSLTParam()」87-12 ページ	特定の最上位レベルのスタイルシート・パラメータを削除します。
「setBindValue()」87-13 ページ	特定のバインド名の値を設定します。

表 87-3 DBMS_XMLQuery のファンクションとプロシージャの要約（続き）

ファンクション / プロシージャ	説明
「setMetaHeader()」87-13 ページ	XML メタ・ヘッダーを設定します。
「setDataHeader()」87-14 ページ	XML データ・ヘッダーを設定します。
「setEncodingTag()」87-14 ページ	XML 文書でのエンコーディング処理命令を設定します。
「setRaiseException()」87-15 ページ	発生した例外をスローするように XSU に指示します。
「setRaiseNoRowsException()」87-15 ページ	生成された XML 文書がなんらかの理由で空である場合、`OracleXMLNoRowsException` をスローするかどうかを XSU に指示します。
「setSQLToXMLNameEscaping()」87-16 ページ	XML 識別子にマップされた SQL オブジェクト名が有効な XML 識別子でない場合、XML タグのエスケープをオンまたはオフにします。
「propagateOriginalException()」87-16 ページ	例外が発生してスローする場合、OracleXMLSQLException を使用して例外を折り返すのではなく、発生した例外をスローするように XSU に指示します。
「getExceptionContent()」87-17 ページ	このメソッドは、引数を使用して、スローした例外のエラー・コードとエラー・メッセージを戻します。
「getDTD()」87-17 ページ	DTD を生成します。
「getNumRowsProcessed()」87-18 ページ	問合せで処理された行の数を戻します。
「getVersion()」87-18 ページ	使用中の XSU のバージョンを出力します。
「getXML()」87-19 ページ	XML 文書を生成します。

newContext()

問合せコンテキストを作成し、コンテキスト・ハンドルを戻します。指定できるオプションは、次の表のとおりです。

構文	説明
`FUNCTION newContext( sqlQuery IN VARCHAR2) RETURN ctxType`	文字列から問合せコンテキストを作成します。
`FUNCTION newContext( sqlQuery IN CLOB) RETURN ctxType`	CLOB から問合せコンテキストを作成します。

パラメータ	IN / OUT	説明
sqlQuery	(IN)	SQL 問合せ。結果は XML に変換する必要があります。

closeContext()

特定の問合せコンテキストのクローズまたは割当て解除を行います。

構文

```
PROCEDURE closeContext( ctxHdl IN ctxType);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

setRowsetTag()

XML データセットを囲むタグを設定します。

構文

```
PROCEDURE setRowsetTag(ctxHdl IN ctxType, tag IN VARCHAR2)
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setRowTag()

データベース・レコードに対応する XML 要素を囲むタグを設定します。

構文

```
PROCEDURE setRowTag(ctxHdl IN ctxType, tag IN VARCHAR2)
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setErrorTag()

XML エラー・ドキュメントを囲むタグを設定します。

構文

```
PROCEDURE setErrorTag( ctxHdl IN ctxType,
                       tag IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setRowIdAttrName()

タグを囲む行の ID 属性名を設定します。NULL または空の文字列をタグに渡すことで、行の ID 属性を省略します。

構文

```
PROCEDURE setRowIdAttrName( ctxHdl IN ctxType,
                            attrName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setRowIdAttrValue()

タグを囲む行の ID 属性に値を割り当てるスカラー列を指定します。colName に NULL または空の文字列を渡すと、行カウント値（0、1、2 など）に割り当てられている行の ID 属性となります。

構文

```
PROCEDURE setRowIdAttrValue( ctxHdl IN ctxType,
                             colName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
colName	(IN)	値が行の ID 属性に割り当てられる列。

setCollIdAttrName()

コレクション要素のセパレータ・タグの ID 属性名を設定します。

構文

```
PROCEDURE setCollIdAttrName( ctxHdl IN ctxType,
                             attrName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
attrName	(IN)	属性名。

useNullAttributeIndicator()

NULL を示すために XML 属性を使用するか、XML 文書で特定のエンティティのインクルードを省略するかを指定します。

構文

```
PROCEDURE useNullAttributeIndicator( ctxHdl IN ctxType,
                                     flag IN BOOLEAN);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	NULL を示すために属性を使用するかどうか。

useTypeForCollElemTag()

デフォルトでは、コレクション要素のタグ名は、コレクションのタグ名に "_item" が付いた名前です。このメソッドは、引数の TRUE を使用してコールすると、コレクション要素のタイプ名をコレクション要素のタグ名として使用するように XSU に指示します。

構文

```
PROCEDURE useTypeForCollElemTag( ctxHdl IN ctxType,
                                 flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
colName	(IN)	タイプ名の使用をオンにするかどうか。

setTagCase()

生成された XML タグの大文字または小文字を指定します。

構文

```
PROCEDURE setTagCase( ctxHdl IN ctxType,
                      tCase IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tCase	(IN)	タグの大文字または小文字（0 はそのまま、1 は小文字、2 は大文字）。

setDateFormat()

XML 文書で生成された日付の書式を設定します。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件に準拠する必要があります。マスクを NULL または空の文字列に設定すると、デフォルトのマスク（DEFAULT_DATE_FORMAT）が使用されます。

構文

```
PROCEDURE setDateFormat( ctxHdl IN ctxType,
                         mask IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
mask	(IN)	日付マスク。

setMaxRows()

XML に変換される行の最大数を設定します。デフォルトでは、最大数は設定されていません。

構文

```
PROCEDURE setMaxRows ( ctxHdl IN ctxType,
                       rows IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
rows	(IN)	生成する行の最大数。

setSkipRows()

スキップする行数を設定します。デフォルトでは、スキップされる行はありません。

構文

```
PROCEDURE setSkipRows( ctxHdl IN ctxType,
                       rows IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
rows	(IN)	スキップする行の最大数。

setStylesheetHeader()

生成された XML 文書のスタイルシートのヘッダー（スタイルシート処理命令）を設定します。

注意 : URI 引数に NULL を渡すと、スタイルシート・ヘッダーおよびスタイルシート・タイプの設定が解除されます。

構文

```
PROCEDURE setStylesheetHeader( ctxHdl IN ctxType,
                               uri IN VARCHAR2,
                               type IN VARCHAR2 := 'text/xsl');
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
uri	(IN)	スタイルシートの URI。
type	(IN)	スタイルシート・タイプ。デフォルトは、text/xsl です。

setXSLT()

生成された XML に適用されるスタイルシートを登録します。スタイルシートがすでに登録されている場合は、新しいものに置換されます。指定できるオプションは、次の表のとおりです。

構文	説明
PROCEDURE setXSLT(ctxHdl IN ctxType, uri IN VARCHAR2, ref IN VARCHAR2 := null);	スタイルシートを登録解除するには、URI に NULL を渡します。
PROCEDURE setXSLT(ctxHdl IN ctxType, stylesheet CLOB, ref IN VARCHAR2 := null);	スタイルシートを登録解除するには、スタイルシートに NULL または空の文字列を渡します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
uri	(IN)	スタイルシートの URI。
stylesheet	(IN)	スタイルシート。
ref	(IN)	エンティティに含めたりインポートする他、外部エンティティの URL。

setXSLTParam()

最上位レベルのスタイルシート・パラメータの値を設定します。パラメータ値には、有効な XPath 式を指定します（このため、文字列リテラル値は明示的に引用されることが必要です）。

注意：登録されているスタイルシートがない場合、このメソッドは操作できません。

構文

```
PROCEDURE setXSLTParam( ctxHdl IN ctxType,
                        name IN VARCHAR2,
                        value IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
name	(IN)	最上位レベルのスタイルシート・パラメータの名前。
value	(IN)	スタイルシート・パラメータに割り当てられる値。

removeXSLTParam()

最上位レベルのスタイルシート・パラメータの値を削除します。

注意：登録されているスタイルシートがない場合、このメソッドは操作できません。

構文

```
PROCEDURE removeXSLTParam( ctxHdl IN ctxType,
                           name IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
name	(IN)	最上位レベルのスタイルシート・パラメータの名前。

setBindValue()

特定のバインド名の値を設定します。

構文

```
PROCEDURE setBindValue( ctxHdl IN ctxType,
                        bindName IN VARCHAR2,
                        bindValue IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
bindName	(IN)	バインド名。
bindValue	(IN)	バインド値。

setMetaHeader()

XML メタ・ヘッダーを設定します。設定すると、このオブジェクトにより生成された各 XML 文書のメタデータ・パート（DTD または XMLSchema）の最初の部分にヘッダーが挿入されます。最後に指定したメタ・ヘッダーが使用されます。さらに、ヘッダーに NULL を渡すと、パラメータによってメタ・ヘッダーの設定が解除されます。

構文

```
PROCEDURE setMetaHeader( ctxHdl IN ctxType,
                         header IN CLOB := null);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
Header	(IN)	ヘッダー。

setDataHeader()

XML データ・ヘッダーを設定します。データ・ヘッダーとは、問合せにより生成された XML エンティティ（行セット）の始めに追加される XML エンティティです。2 つのエンティティは、docTag 引数を使用して指定されたタグで囲まれます。最後に指定したデータ・ヘッダーが使用されます。さらに、ヘッダーに NULL を渡すと、パラメータによりデータ・ヘッダーの設定が解除されます。

構文

```
PROCEDURE setDataHeader( ctxHdl IN ctxType,
                  header IN CLOB := null,
                  tag IN VARCHAR2 := null);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
header	(IN)	ヘッダー。
tag	(IN)	データ・ヘッダーおよび行セットを囲むタグ。

setEncodingTag()

XML 文書でのエンコーディング処理命令を設定します。

構文

```
PROCEDURE setEncodingTag( ctxHdl IN ctxType,
                          enc IN VARCHAR2 := DB_ENCODING);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
enc	(IN)	使用するエンコーディング。

setRaiseException()

発生した例外をスローするように XSU に指示します。このコールが行われない場合またはフラグ引数に FALSE が渡された場合は、XSU で SQL の例外が発生し、例外のメッセージから XML 文書が生成されます。

構文

```
PROCEDURE setRaiseException( ctxHdl IN ctxType,
                             flag IN BOOLEAN);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	発生した例外をスローするかどうかの有無。スローする場合は TRUE、スローしない場合は FALSE を設定します。

setRaiseNoRowsException()

生成された XML 文書がなんらかの理由で空である場合、OracleXMLNoRowsException をスローするかどうかを XSU に指示します。デフォルトでは、例外はスローされません。

構文

```
PROCEDURE setRaiseNoRowsException( ctxHdl IN ctxType,
                                   flag IN BOOLEAN);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	データがない場合に OracleXMLNoRowsException をスローするかどうか。スローする場合は TRUE、スローしない場合は FALSE を設定します。

setSQLToXMLNameEscaping()

XML 識別子にマップされた SQL オブジェクト名が有効な XML 識別子でない場合、XML タグのエスケープをオンまたはオフにします。

構文

```
PROCEDURE setSQLToXMLNameEscaping( ctxHdl IN ctxType,
                                   flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	エスケープをオンにするかどうか。エスケープする場合は TRUE、エスケープしない場合は FALSE を設定します。

propagateOriginalException()

例外が発生してスローする場合、OracleXMLSQLException を使用して例外を折り返すのではなく、発生した例外をスローするように XSU に指示します。

構文

```
PROCEDURE propagateOriginalException( c txHdl IN ctxType,
                                      flag IN BOOLEAN);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	元の例外を伝播するかどうか。伝播する場合は TRUE、伝播しない場合は FALSE を設定します。

getExceptionContent()

このメソッドは、引数を使用して、スローした例外のエラー・コードとエラー・メッセージ（SQL エラー・コード）を戻します。これは、何の例外が発生した場合でも、JVM が例外をスローさせるという事実を避けるために行われます。したがって、PL/SQL のレンダリングにより、元の例外にアクセスできなくなります。

構文

```
PROCEDURE getExceptionContent( ctxHdl IN ctxType,
                               errNo OUT NUMBER,
                               errMsg OUT VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
errNo	(IN)	エラー番号。
errMsg	(IN)	エラー・メッセージ。

getDTD()

コンテキストの初期化に使用する SQL 問合せに基づいて DTD を生成して戻します。指定できるオプションは、次の表のとおりです。

構文	説明
FUNCTION getDTD(ctxHdl IN ctxType, withVer IN BOOLEAN := false) RETURN CLOB;	コンテキストの初期化に使用する SQL 問合せに基づいて DTD を生成するファンクション。
PROCEDURE getDTD(ctx IN ctxType, xDoc IN CLOB, withVer IN BOOLEAN := false);	CLOB のコンテキストと xDOC の初期化に使用する SQL 問合せに基づいて DTD を生成するプロシージャ。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
withVer	(IN)	バージョン情報を生成するかどうか。生成する場合は TRUE、生成しない場合は FALSE を設定します。
xDoc	(IN)	生成された XML 文書の書込み先 CLOB。

getNumRowsProcessed()

問合せで処理された行の数を戻します。

構文

```
FUNCTION getNumRowsProcessed( ctx IN ctxType) RETURN NUMBER;
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

getVersion()

使用中の XSU のバージョンを出力します。

構文

```
PROCEDURE getVersion();
```

getXML()

新規コンテキストを作成して問合せを実行し、XML を戻してコンテキストをクローズします。これは便利なファンクションです。コンテキストは、明示的にオープンまたはクローズする必要はありません。指定できるオプションは、次の表のとおりです。

構文	説明
FUNCTION getXML(sqlQuery IN VARCHAR2, metaType IN NUMBER := NONE) RETURN CLOB;	このファンクションでは、sqlQuery が文字列形式で使用されます。
FUNCTION getXML(sqlQuery IN CLOB, metaType IN NUMBER := NONE) RETURN CLOB;	このファンクションでは、sqlQuery が CLOB 形式で使用されます。
FUNCTION getXML(ctxHdl IN ctxType, metaType IN NUMBER := NONE); RETURN CLOB	このファンクションは、コンテキストの初期化に使用する SQL 問合せに基づいて XML 文書を生成します。
PROCEDURE getXML(ctxHdl IN ctxType, xDoc IN CLOB, metaType IN NUMBER := NONE);	このプロシージャは、コンテキストの初期化に使用する SQL 問合せに基づいて XML 文書を生成します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
sqlQuery	(IN)	SQLQuery。
metaType	(IN)	XML メタデータ・タイプ（NONE、DTD または SCHEMA）。
sDoc	(IN)	生成された XML 文書の書込み先 CLOB。

88

DBMS_XMLSAVE

DBMS_XMLSAVE は、XML からデータベース・タイプへの変換の機能を提供します。

> **関連項目：** 詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

DBMS_XMLSave の概要

この API は、XML_to_DB タイプの機能を提供します。

DBMS_XMLSave のタイプ

表 88-1 DBMS_XMLSave のタイプ

タイプ	説明
ctxType	問合せコンテキスト・ハンドルのタイプ。newContext() の戻り型です。

DBMS_XMLSave の定数

表 88-2 DBMS_XMLSave の定数

定数	説明
DEFAULT_ROWTAG	データベース・レコードに対応する要素のデフォルト・タグ名 -- ROW。
DEFAULT_DATE_FORMAT	デフォルトの日付マスク -- 'MM/dd/yyyy HH:mm:ss'。
MATCH_CASE	XML 要素のデータベース・エンティティへのマップ時に、XSU は大 / 小文字区別が必要であることを指定します。
IGNORE_CASE	XML 要素のデータベース・エンティティへのマップ時に、XSU は大 / 小文字区別が不要なことを指定します。

DBMS_XMLSave のファンクションとプロシージャ

表 88-3 DBMS_XMLSave のファンクションとプロシージャの要約

ファンクション / プロシージャ	説明
「newContext()」88-4 ページ	保存コンテキストを作成し、コンテキスト・ハンドルを戻します。
「closeContext()」88-4 ページ	特定の保存コンテキストのクローズまたは割当て解除を行います。
「setRowTag()」88-5 ページ	XML 文書で使用されたタグに名前を付け、データベースに対応する XML 要素を囲みます。
「setIgnoreCase()」88-5 ページ	XSU が XML 要素をデータベースにマップします。
「setDateFormat()」88-6 ページ	XML 文書の日付書式を XSU に記述します。
「setBatchSize()」88-6 ページ	DML の操作中に使用されたバッチ・サイズを変更します。
「setCommitBatch()」88-7 ページ	コミットするバッチ・サイズを設定します。
「setSQLToXMLNameEscaping()」88-7 ページ	XML 識別子にマップされた SQL オブジェクト名が有効な XML 識別子でない場合、XML タグのエスケープをオンまたはオフにします。
「setUpdateColumn()」88-8 ページ	更新列リストに列を追加します。
「clearUpdateColumnList()」88-8 ページ	更新列リストをクリアします。
「setPreserveWhitespace()」88-8 ページ	空白を保持するかどうかを XSU に指示します。
「setKeyColumn()」88-9 ページ	このメソッドは、キー列リストに列を追加します。
「clearKeyColumnList()」88-9 ページ	キー列リストをクリアします。
「setXSLT()」88-10 ページ	保存する XML に追加する XSL 変換を登録します。
「setXSLTParam()」88-10 ページ	最上位レベルのスタイルシート・パラメータの値を設定します。
「removeXSLTParam()」88-11 ページ	最上位レベルのスタイルシート・パラメータの値を削除します。
「insertXML()」88-11 ページ	コンテキストの作成時に指定した表に XML 文書を挿入します。

表 88-3 DBMS_XMLSave のファンクションとプロシージャの要約（続き）

ファンクション / プロシージャ	説明
「updateXML()」88-12 ページ	XML 文書を指定して、表を更新します。
「deleteXML()」88-13 ページ	XML 文書のデータで指定したレコードを、コンテキスト作成時に指定した表から削除します。
「propagateOriginalException()」88-13 ページ	例外が発生してスローする場合、OracleXMLSQLException を使用して例外を折り返すのではなく、発生した例外をスローするように XSU に指示します。
「getExceptionContent()」88-14 ページ	このメソッドは、引数を使用して、スローした例外のエラー・コードとエラー・メッセージを戻します。

newContext()

保存コンテキストを作成し、コンテキスト・ハンドルを戻します。

構文

```
FUNCTION newContext(t argetTable IN VARCHAR2) RETURN ctxType;
```

パラメータ	IN / OUT	説明
targetTable	(IN)	XML 文書のロード先となる表。

closeContext()

特定の保存コンテキストのクローズまたは割当て解除を行います。

構文

```
PROCEDURE closeContext(ctxHdl IN ctxType);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

setRowTag()

XML 文書で使用されたタグに名前を付け、データベース・レコードに対応する XML 要素を囲みます。

構文

```
PROCEDURE setRowTag( ctxHdl IN ctxType,
                     tag IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
tag	(IN)	タグ名。

setIgnoreCase()

XSU が要素名（XML タグ）に基づいて、XML 要素をデータベースの列または属性にマップします。このファンクションは、大 / 小文字を区別しないでマップするように XSU に指示します。

構文

```
PROCEDURE setIgnoreCase( ctxHdl IN ctxType,
                         flag IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	XML 文書でタグの大 / 小文字を無視するかどうか。0（ゼロ）の場合は FALSE、1 の場合は TRUE です。

setDateFormat()

XML 文書の日付書式を XSU に記述します。日付書式パターン（日付マスク）の構文は、java.text.SimpleDateFormat クラスの要件に従う必要があります。マスクを NULL または空の文字列に設定すると、デフォルトのマスク（OracleXMLCore.DATE_FORMAT）が使用されます。

構文

```
PROCEDURE setDateFormat( ctxHdl IN ctxType,
                         mask IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
mask	(IN)	日付マスク。

setBatchSize()

DML の操作中に使用されたバッチ・サイズを変更します。挿入、更新または削除を実行する場合は、複数の操作をバッチ化することで、それぞれを別の文として実行しないで、1 回ですべてを実行できます。ただし、すべてのバインド値をバッファに設定するために必要なメモリー量が増加します。バッチが使用されると、バッチが実行された後にのみコミットが発生することに注意してください。したがって、バッチ内の文のいずれかが失敗した場合は、バッチ全体がロールバックされます。このロールバックは、パフォーマンス向上には少し役立ちますが、回避する場合は、バッチ・サイズを 1 に設定してください。

構文

```
PROCEDURE setBatchSize( ctxHdl IN ctxType,
                        batchSize IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
batchSize	(IN)	バッチ・サイズ。

setCommitBatch()

コミットするバッチ・サイズを設定します。コミットするバッチ・サイズは、数値またはコミットの前に挿入されたレコードを参照します。commitBatch が 1 未満の場合、またはセッションが自動コミット・モードの場合、XSU は明示的なコミットを行いません。デフォルトでは、コミットのバッチ・サイズは 0（ゼロ）です。

構文

```
PROCEDURE setCommitBatch( ctxHdl IN ctxType,
                          batchSize IN NUMBER);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
batchSize	(IN)	コミットするバッチ・サイズ。

setSQLToXMLNameEscaping()

XML 識別子にマップされた SQL オブジェクト名が有効な XML 識別子でない場合、XML タグのエスケープをオンまたはオフにします。

構文

```
PROCEDURE setSQLToXMLNameEscaping( ctxHdl IN ctxType,
                                   flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	エスケープをオンにするかどうか。

setUpdateColumn()

更新列リストに列を追加します。挿入の場合、デフォルトでは表のすべての列に値が挿入されます。これに対して、更新の場合、デフォルトでは XML 文書の ROW 要素に存在するタグに対応する列のみが更新されます。更新列リストが指定されている場合、このリストに含まれる列のみが更新または挿入の対象となります。

構文

```
PROCEDURE setUpdateColumn( ctxHdl IN ctxType,
                           colName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
colName	(IN)	更新列リストに追加される列。

clearUpdateColumnList()

更新列リストをクリアします。

構文

```
PROCEDURE clearUpdateColumnList( ctxHdl IN ctxType);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

setPreserveWhitespace()

空白を保持するかどうかを XSU に指示します。

構文

```
PROCEDURE setPreserveWhitespace( ctxHdl IN ctxType,
                                 flag IN BOOLEAN := true);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	XSU が空白を保持する必要があるかどうか。

setKeyColumn()

このメソッドは、キー列リストに列を追加します。更新または削除の場合、更新または削除文の WHERE 句を構成するのはキー列リストの列です。更新を実行するには、キー列リストを先に指定する必要があります。削除操作においては、キー列リストはオプションです。

構文

```
PROCEDURE setKeyColumn( ctxHdl IN ctxType,
                        colName IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
colName	(IN)	キー列リストに追加される列。

clearKeyColumnList()

キー列リストをクリアします。

構文

```
PROCEDURE clearKeyColumnList( ctxHdl IN ctxType);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。

setXSLT()

保存する XML に追加する XSL 変換を登録します。スタイルシートがすでに登録されている場合は、新しいものに置換されます。スタイルシートを登録解除するには、URI に NULL を渡します。指定できるオプションは、次の表のとおりです。

構文	説明
PROCEDURE setXSLT(ctxHdl IN ctxType, uri IN VARCHAR2, ref IN VARCHAR2 := null);	URI を介してスタイルシートを渡します。
PROCEDURE setXSLT(ctxHdl IN ctxType, stylesheet IN CLOB, ref IN VARCHAR2 := null);	CLOB を介してスタイルシートを渡します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
uri	(IN)	登録するスタイルシートの URI。
ref	(IN)	エンティティに含めたりインポートする他、外部エンティティの URL。
stylesheet	(IN)	登録するスタイルシートを含む CLOB。

setXSLTParam()

最上位レベルのスタイルシート・パラメータの値を設定します。パラメータには、有効な XPath 式を指定します（このため、文字列リテラル値は明示的に引用されることが必要です）。

構文

```
PROCEDURE setXSLTParam( ctxHdl IN ctxType,
                        name IN VARCHAR2,
                        value IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
name	(IN)	パラメータ名。
value	(IN)	XPath 式として指定するパラメータ値。

removeXSLTParam()

最上位レベルのスタイルシート・パラメータの値を削除します。

構文

```
PROCEDURE removeXSLTParam( ctxHdl IN ctxType,
                           name IN VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
name	(IN)	パラメータ名。

insertXML()

コンテキストの作成時に指定した表に XML 文書を挿入し、挿入された行数を戻します。指定できるオプションは、次の表のとおりです。

構文	説明
FUNCTION insertXML(ctxHdl IN ctxType, xDoc IN VARCHAR2) RETURN NUMBER;	xDoc パラメータを VARCHAR2 として渡します。
FUNCTION insertXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER;	xDoc パラメータを CLOB として渡します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
xDoc	(IN)	XML 文書を含む文字列。

updateXML()

コンテキストの作成時に指定した表を XML 文書のデータで更新し、更新された行数を戻します。指定できるオプションは、次の表のとおりです。

構文	説明
FUNCTION updateXML(ctxHdl IN ctxType, xDoc IN VARCHAR2) RETURN NUMBER;	xDoc パラメータを VARCHAR2 として渡します。
FUNCTION updateXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER;	xDoc パラメータを CLOB として渡します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
xDoc	(IN)	XML 文書を含む文字列。

deleteXML()

XML 文書のデータで指定したレコードを、コンテキスト作成時に指定した表から削除し、削除した行数を戻します。指定できるオプションは、次の表のとおりです。

構文	説明
FUNCTION deleteXML(ctxHdl IN ctxPType, xDoc IN VARCHAR2) RETURN NUMBER;	xDoc パラメータに VARCHAR2 タイプを使用します。
FUNCTION deleteXML(ctxHdl IN ctxType, xDoc IN CLOB) RETURN NUMBER;	xDoc パラメータに CLOB タイプを使用します。

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
xDoc	(IN)	XML 文書を含む文字列。

propagateOriginalException()

例外が発生してスローする場合、OracleXMLSQLException を使用して例外を折り返すのではなく、発生した例外をスローするように XSU に指示します。

構文

```
PROCEDURE propagateOriginalException( ctxHdl IN ctxType,
                                      flag IN BOOLEAN);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
flag	(IN)	元の例外を伝播するかどうか。0（ゼロ）の場合は FALSE、1 の場合は TRUE です。

getExceptionContent()

このメソッドは、引数を使用して、スローした例外のエラー・コードおよびエラー・メッセージ（SQL エラー・コード）を戻します。これは、すでに発生している例外に加えて、JVM が例外をスローし、結果として、PL/SQL のレンダリングによる元の例外へのアクセスができなくなることを避けるために行われます。

構文

```
PROCEDURE getExceptionContent( ctxHdl IN ctxType,
                               errNo OUT NUMBER,
                               errMsg OUT VARCHAR2);
```

パラメータ	IN / OUT	説明
ctxHdl	(IN)	コンテキスト・ハンドル。
errNo	(IN)	エラー番号。
errMsg	(IN)	エラー・メッセージ。

89

DBMS_XMLSchema

DBMS_XMLSchema パッケージは、XML Schema の登録および削除を行うプロシージャを提供します。

> **関連項目：** 詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

- DBMS_XMLSCHEMA の定数
- DBMS_XMLSCHEMA のファンクションとプロシージャ

DBMS_XMLSCHEMA の概要

このパッケージは、Oracle XML DB のインストール時にスクリプト dbmsxsch.sql によって作成されます。このパッケージは、XML Schema の登録および削除を行うプロシージャを提供します。

DBMS_XMLSCHEMA の定数

表 89-1 DBMS_XMLSCHEMA の定数

定数	説明
DELETE_RESTRICT	CONSTANT NUMBER := 1;
DELETE_INVALIDATE	CONSTANT NUMBER := 2;
DELETE_CASCADE	CONSTANT NUMBER := 3;
DELETE_CASCADE_FORCE	CONSTANT NUMBER := 4;

DBMS_XMLSCHEMA のファンクションとプロシージャ

表 89-2 DBMS_XMLSCHEMA のファンクションとプロシージャの要約

ファンクション / プロシージャ	説明
「registerSchema()」89-3 ページ	Oracle で使用する指定のスキーマを登録します。このスキーマは、このスキーマに準拠するドキュメントを格納するために使用できます。
「registerURI()」89-5 ページ	URI 名で指定した XML Schema を登録します。
「deleteSchema()」89-6 ページ	Oracle XML DB からスキーマを削除します。
「compileSchema()」89-7 ページ	登録済みの XML Schema を再コンパイルします。これは、無効な状態のスキーマを有効な状態にする場合に役立ちます。
「generateSchema()」89-8 ページ	Oracle のタイプ名から XML Schema を生成します。

registerSchema()

Oracle XML DB で使用する指定のスキーマを登録します。指定できるオプションは、次の表のとおりです。

構文	説明

`procedure registerSchema(schemaURL IN varchar2,`
`schemaDoc IN VARCHAR2,`
`local IN BOOLEAN := TRUE,`
`genTypes IN BOOLEAN := TRUE,`
`genbean IN BOOLEAN := FALSE,`
`genTables IN BOOLEAN := TRUE,`
`force IN BOOLEAN := FALSE,`
`owner IN VARCHAR2 := null);`
	VARCHAR2 として指定したスキーマを登録します。

`procedure registerSchema(schemaURL IN varchar2,`
`schemaDoc IN CLOB,`
`local IN BOOLEAN := TRUE,`
`genTypes IN BOOLEAN := TRUE,`
`genbean IN BOOLEAN := FASLE,`
`force IN BOOLEAN := FALSE,`
`owner IN VARCHAR2 := null);`
	CLOB として指定したスキーマを登録します。

`procedure registerSchema(schemaURL IN varchar2,`
`schemaDoc IN BFILE,`
`local IN BOOLEAN := TRUE,`
`genTypes IN BOOLEAN := TRUE,`
`genbean IN BOOLEAN := FALSE,`
`force IN BOOLEAN := FALSE,`
`owner IN VARCHAR2 := null);`
	BFILE として指定したスキーマを登録します。

構文	説明
procedure registerSchema(schemaURL IN varchar2, schemaDoc IN SYS.XMLType, local IN BOOLEAN := TRUE, genTypes IN BOOLEAN := TRUE, genbean IN BOOLEAN := FALSE, force IN BOOLEAN := FALSE, owner IN VARCHAR2 := null);	XMLType として指定したスキーマを登録します。
procedure registerSchema(schemaURL IN varchar2, schemaDoc IN SYS.URIType, local IN BOOLEAN := TRUE, genTypes IN BOOLEAN := TRUE, genbean IN BOOLEAN := FALSE, force IN BOOLEAN := FALSE, owner IN VARCHAR2 := null);	URIType として指定したスキーマを登録します。

パラメータ	IN / OUT	説明
schemaURL	(IN)	スキーマ・ドキュメントを一意に識別する URL。この値は、Oracle XML DB 階層内のスキーマ・ドキュメントのパス名を導出するために使用します。
schemaDoc	(IN)	有効な XML Schema ドキュメント。
local	(IN)	スキーマがローカルかグローバルか。 デフォルトでは、すべてのスキーマはローカル・スキーマとして、/sys/schemas/<username/... の下に登録されます。 グローバルとして登録されるスキーマは、 /sys/schemas/PUBLIC/... の下に登録されます。 スキーマをグローバルとして登録するには、前述のディレクトリに対する書込み権限が必要です。
genTypes	(IN)	スキーマ・コンパイラがオブジェクト・タイプを生成する必要があるかどうか。デフォルトは TRUE です。
genbean	(IN)	このリリースでは常に FALSE です。
genTables	(IN)	スキーマ・コンパイラがデフォルトの表を生成する必要があるかどうか。デフォルトは TRUE です。

パラメータ	IN / OUT	説明
force	(IN)	このパラメータを TRUE に設定した場合、スキーマ登録によるエラーは発生しません。エラーの場合は、かわりに、無効な XML Schema オブジェクトが作成されます。このパラメータのデフォルト値は FALSE です。
owner	(IN)	このパラメータは、XML Schema オブジェクトを所有するデータベース・ユーザーの名前を指定します。デフォルトでは、スキーマを登録したユーザーが XML Schema オブジェクトを所有します。このパラメータを使用すると、別のデータベース・ユーザーが所有する XML Schema を登録できます。

registerURI()

URI 名で指定した XML Schema を登録します。

構文

```
procedure registerURI(schemaURL IN varchar2,
                      schemaDocURI IN varchar2,
                      local IN BOOLEAN := TRUE,
                      genTypes IN BOOLEAN := TRUE,
                      genbean IN BOOLEAN := FALSE,
                      genTables IN BOOLEAN := TRUE,
                      force IN BOOLEAN := FALSE,
                      owner IN VARCHAR2 := null);
```

パラメータ	IN / OUT	説明
schemaURL	(IN)	スキーマ・ドキュメントを一意に識別する名前。
schemaDocURI	(IN)	スキーマ・ドキュメントの物理的な位置に対応するパス名（URI）。URI パスは、HTTP、FTP、DB または Oracle XML DB プロトコルに基づく場合があります。このファンクションは、URIFactory を使用して URIType インスタンスを構成し、registerSchema() ファンクションを起動します。
local	(IN)	スキーマがローカルかグローバルか。 デフォルトでは、すべてのスキーマはローカル・スキーマとして、/sys/schemas/<username/... の下に登録されます。 グローバルとして登録されるスキーマは、/sys/schemas/PUBLIC/... の下に追加されます。 ユーザーがスキーマをグローバルとして登録するには、前述のディレクトリに対する書込み権限が必要です。

パラメータ	IN / OUT	説明
genTypes	(IN)	スキーマ・コンパイラがオブジェクト・タイプを生成する必要があるかどうか。デフォルトは TRUE です。
genbean	(IN)	このリリースでは常に FALSE です。
genTables	(IN)	スキーマ・コンパイラがデフォルトの表を生成する必要があるかどうか。デフォルトは TRUE です。
force	(IN)	このパラメータを TRUE に設定した場合、スキーマ登録によるエラーは発生しません。エラーの場合は、かわりに、無効な XML Schema オブジェクトが作成されます。このパラメータのデフォルト値は FALSE です。
owner	(IN)	このパラメータは、XML Schema オブジェクトを所有するデータベース・ユーザーの名前を指定します。デフォルトでは、スキーマを登録したユーザーが XML Schema オブジェクトを所有します。このパラメータを使用すると、別のデータベース・ユーザーが所有する XML Schema を登録できます。

deleteSchema()

URL で指定した XMLSchema を削除します。ORA-31001 例外「リソース・ハンドルまたはパス名が無効です」が発生する場合があります。

構文

```
procedure deleteSchema(schemaURL IN varchar2,
                       delete_option IN pls_integer := DELETE_RESTRICT);
```

パラメータ	IN / OUT	説明
schemaURL	(IN)	削除するスキーマを識別する URL。
delete_option	(IN)	スキーマを削除するためのオプション。

delete_option パラメータのオプション

オプション	説明
DELETE_RESTRICT	このスキーマに依存する表またはスキーマが存在する場合、スキーマの削除は失敗します。
DELETE_INVALIDATE	依存関係が存在する場合でも、スキーマの削除に失敗することはありません。かわりに、依存するすべてのオブジェクトは無効になります。
DELETE_CASCADE	スキーマを削除すると、デフォルトの SQL タイプと表も削除されます。ただし、このスキーマに準拠するインスタンスが格納されている場合、削除は失敗します。
DELETE_CASCADE_FORCE	CASCADE オプションと同じですが、このスキーマに準拠するインスタンスが格納されているかどうかはチェックしません。また、すべてのエラーは無視されます。

compileSchema()

このプロシージャは、登録済みの XML Schema を再コンパイルするために使用できます。これは、無効な状態のスキーマを有効な状態にする場合に役立ちます。ORA-31001 例外「リソース・ハンドルまたはパス名が無効です」が発生する場合があります。

構文

```
procedure compileSchema( schemaURL IN varchar2);
```

パラメータ	IN / OUT	説明
schemaURL	(IN)	スキーマを識別する URL。

generateSchema()

このファンクションは、Oracle のタイプ名から XML Schema を生成します。 ORA-31001 例外「リソース・ハンドルまたはパス名が無効です」が発生する場合があります。指定できるオプションは、次の表のとおりです。

構文	説明
function generateSchemas(schemaName IN varchar2, typeName IN varchar2, elementName IN varchar2 := NULL, schemaURL IN varchar2 := NULL, annotate IN BOOLEAN := TRUE, embedColl IN BOOLEAN := TRUE) return sys.XMLSequenceType;	XMLType のコレクションを戻します。データベース・スキーマごとに 1 つの XMLSchema ドキュメントになります。
function generateSchema(schemaName IN varchar2, typeName IN varchar2, elementName IN varchar2 := NULL, recurse IN BOOLEAN := TRUE, annotate IN BOOLEAN := TRUE, embedColl IN BOOLEAN := TRUE) return sys.XMLType;	1 つのスキーマ（XMLType）にすべてインラインします。

パラメータ	IN / OUT	説明
schemaName	(IN)	タイプを含むデータベース・スキーマの名前。
typeName	(IN)	Oracle タイプの名前。
elementName	(IN)	typeName にデフォルト設定された XMLSchema の最上位レベルの要素の名前。
schemaURL	(IN)	インポート文の最上位レベルのスキーマに必要な、スキーマが格納されるベース URL を指定します。
recurse	(IN)	指定したタイプで参照されるすべてのタイプのスキーマを生成するかどうかを指定します。

パラメータ	IN / OUT	説明
annotate	(IN)	XMLSchema に SQL 注釈を設定するかどうかを指定します。
embedColl	(IN)	コレクションを参照するタイプにコレクションを埋め込むか、または complexType を作成するか。注釈がオンの場合は、FALSE に設定できません。

カタログ・ビュー

表 89-3 カタログ・ビュー・スキーマの要約

スキーマ	説明
「USER_XML_SCHEMAS」 89-10 ページ	ユーザーが所有するすべての登録済み XML Schema。
「ALL_XML_SCHEMAS」 89-10 ページ	現行ユーザーが使用できるすべての登録済み XML Schema。
「DBA_XML_SCHEMAS」 89-11 ページ	Oracle XML DB のすべての登録済み XML Schema。
「DBA_XML_TABLES」 89-11 ページ	システム内のすべての XMLType 表。
「USER_XML_TABLES」 89-11 ページ	現行ユーザーが所有するすべての XMLType 表。
「ALL_XML_TABLES」 89-12 ページ	現行ユーザーが使用できるすべての XMLType 表。
「DBA_XML_TAB_COLS」 89-12 ページ	システム内のすべての XMLType 表列。
「USER_XML_TAB_COLS」 89-12 ページ	現行ユーザーが所有する表内のすべての XMLType 表列。
「ALL_XML_TAB_COLS」 89-13 ページ	現行ユーザーが使用できる表内のすべての XMLType 表列。
「DBA_XML_VIEWS」 89-13 ページ	システム内のすべての XMLType ビュー。
「USER_XML_VIEWS」 89-13 ページ	現行ユーザーが所有するすべての XMLType ビュー。
「ALL_XML_VIEWS」 89-14 ページ	現行ユーザーが使用できるすべての XMLType ビュー。
「DBA_XML_VIEW_COLS」 89-14 ページ	システム内のすべての XMLType ビュー列。

表 89-3 カタログ・ビュー・スキーマの要約（続き）

スキーマ	説明
「USER_XML_VIEW_COLS」 89-14 ページ	現行ユーザーが所有するビュー内のすべての XMLType ビュー列。
「ALL_XML_VIEW_COLS」 89-15 ページ	現行ユーザーが使用できるビュー内のすべての XMLType ビュー列。

USER_XML_SCHEMAS

現行ユーザーに属するすべてのスキーマ（ローカルおよびグローバル）をリストします。

列	データ・タイプ	説明
SCHEMA_URL	VARCHAR2	XML Schema の URL。
LOCAL	VARCHAR2	ローカル・スキーマ（YES/NO）。
SCHEMA	XMLTYPE	XML Schema ドキュメント。

ALL_XML_SCHEMAS

現行ユーザーに属するすべてのローカル・スキーマおよびグローバル・スキーマをリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	XML Schema を所有するデータベース・ユーザー。
SCHEMA_URL	VARCHAR2	XML Schema の URL。
LOCAL	VARCHAR2	ローカル・スキーマ（YES/NO）。
SCHEMA	XMLTYPE	XML Schema ドキュメント。

DBA_XML_SCHEMAS

システム内の登録済みのローカルとグローバルのスキーマをすべてリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	XML Schema を所有するデータベース・ユーザー。
SCHEMA_URL	VARCHAR2	XML Schema の URL。
LOCAL	VARCHAR2	ローカル・スキーマ（YES/NO）。
SCHEMA	XMLTYPE	XML Schema ドキュメント。

DBA_XML_TABLES

システム内のすべての XMLType 表をリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	表を所有するデータベース・ユーザー。
TABLE_NAME	VARCHAR2	XMLType 表の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。
STORAGE_TYPE	VARCHAR2	格納タイプ : CLOB / OBJECT-RELATIONAL。

USER_XML_TABLES

現行ユーザーに属するすべての XMLType 表をリストします。

列	データ・タイプ	説明
TABLE_NAME	VARCHAR2	XMLType 表の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。
STORAGE_TYPE	VARCHAR2	格納タイプ : CLOB / OBJECT-RELATIONAL。

ALL_XML_TABLES

現行ユーザーに属するすべてのローカル XMLType 表、および現行ユーザーが参照できるすべてのグローバル表をリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	表を所有するデータベース・ユーザー。
TABLE_NAME	VARCHAR2	XMLType 表の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。
STORAGE_TYPE	VARCHAR2	格納タイプ : CLOB / OBJECT-RELATIONAL。

DBA_XML_TAB_COLS

システム内のすべての XMLType 列をリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	表を所有するデータベース・ユーザー。
TABLE_NAME	VARCHAR2	表の名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。
STORAGE_TYPE	VARCHAR2	格納タイプ : CLOB / OBJECT-RELATIONAL。

USER_XML_TAB_COLS

現行ユーザーに属する表内のすべての XMLType 列をリストします。

列	データ・タイプ	説明
TABLE_NAME	VARCHAR2	表の名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。
STORAGE_TYPE	VARCHAR2	格納タイプ : CLOB / OBJECT-RELATIONAL。

ALL_XML_TAB_COLS

現行ユーザーに属する表内のすべての XMLType 列、および現行ユーザーが参照できるすべてのグローバル表をリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	表を所有するデータベース・ユーザー。
TABLE_NAME	VARCHAR2	表の名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。
STORAGE_TYPE	VARCHAR2	格納タイプ : CLOB / OBJECT-RELATIONAL。

DBA_XML_VIEWS

システム内のすべての XMLType ビューをリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	ビューを所有するデータベース・ユーザー。
VIEW_NAME	VARCHAR2	XMLType ビューの名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。

USER_XML_VIEWS

現行ユーザーに属するすべての XMLType ビューをリストします。

列	データ・タイプ	説明
VIEW_NAME	VARCHAR2	XMLType ビューの名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。

ALL_XML_VIEWS

現行ユーザーに属するすべてのローカル XMLType ビュー、および現行ユーザーが参照できるすべてのグローバル・ビューをリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	ビューを所有するデータベース・ユーザー。
VIEW_NAME	VARCHAR2	XMLType ビューの名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。

DBA_XML_VIEW_COLS

システム内のすべての XMLType 列をリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	ビューを所有するデータベース・ユーザー。
VIEW_NAME	VARCHAR2	ビューの名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。

USER_XML_VIEW_COLS

現行ユーザーに属するビュー内のすべての XMLType 列をリストします。

列	データ・タイプ	説明
VIEW_NAME	VARCHAR2	ビューの名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。

ALL_XML_VIEW_COLS

現行ユーザーに属するビュー内のすべての XMLType 列、および現行ユーザーが参照できるすべてのグローバル・ビューをリストします。

列	データ・タイプ	説明
OWNER	VARCHAR2	ビューを所有するデータベース・ユーザー。
VIEW_NAME	VARCHAR2	ビューの名前。
COLUMN_NAME	VARCHAR2	XMLType 列の名前。
XMLSCHEMA	VARCHAR2	XML Schema の URL。
ELEMENT_NAME	VARCHAR2	XML Schema の要素。

90

DBMS_XPLAN

DBMS_XPLAN パッケージによって、EXPLAIN PLAN コマンドの出力を書式設定する簡単な方法が提供されます。EXPLAIN PLAN コマンドの詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

このパッケージは、パッケージ所有者（SYS）ではなく、コール・ユーザーの権限で実行されます。

この章では、次の項目について説明します。

DBMS_XPLAN の使用

DBMS_XPLAN パッケージには、次の例に示すように、PLAN TABLE の内容を書式設定して表示するための DISPLAY テーブル・ファンクションが用意されています。

DBMS_XPLAN.DISPLAY を使用した PLAN TABLE の表示 : 例

```
Rem
Rem  Execute an explain plan command on a SELECT statement
Rem
EXPLAIN PLAN FOR
SELECT *
FROM emp e, dept d
WHERE e.deptno = d.deptno
  AND e.ename='benoit';

Rem
Rem Display the plan using the DBMS_XPLAN.DISPLAY() table function
Rem
SET LINESIZE 130
SET PAGESIZE 0
SELECT * FROM table(DBMS_XPLAN.DISPLAY);
```

この問合せによって次の出力が作成されます。

```
---------------------------------------------------------------------
| Id  | Operation            |  Name       | Rows  | Bytes | Cost  |
---------------------------------------------------------------------
|   0 | SELECT STATEMENT     |             |     1 |    50 |     3 |
|*  1 | HASH JOIN            |             |     1 |    50 |     3 |
|*  2 | TABLE ACCESS FULL    | EMP         |     1 |    32 |     1 |
|   3 | TABLE ACCESS FULL    | DEPT        |     4 |    72 |     1 |
---------------------------------------------------------------------
Predicate Information (identified by operation id)
1 - access("E1"."DEPTNO"="D1"."DEPTNO")
2 - filter("E1"."ENAME"='benoit')
```

DBMS_XPLAN サブプログラムの要約

表 90-1 DBMS_XPLAN パッケージのサブプログラム

サブプログラム	説明
「DISPLAY ファンクション」 90-3 ページ	PLAN TABLE の内容を表示します。

DISPLAY ファンクション

このファンクションは、PLAN TABLE の内容を表示します。

構文

```
DBMS_XPLAN.DISPLAY(
   table_name    IN  VARCHAR2  DEFAULT 'PLAN_TABLE',
   statement_id  IN  VARCHAR2  DEFAULT  NULL,
   format        IN  VARCHAR2  DEFAULT  'TYPICAL');
```

パラメータ

表 90-2 DISPLAY ファンクションのパラメータ

パラメータ	説明
table_name	プランが格納されている表の名前を指定します。このパラメータのデフォルトは PLAN_TABLE です。これは EXPLAIN PLAN コマンドに対するデフォルトの PLAN TABLE です。
statement_id	表示するプランの statement_id を指定します。このパラメータのデフォルトは NULL です。これは、EXPLAIN PLAN コマンドを set statement_id 句を使用せずに実行する場合のデフォルトです。
format	プランの詳細レベルを制御します。次の 4 つの値を指定できます。 ■ BASIC: プランの最小限の情報として、操作 ID、オブジェクト名および操作オプションを表示します。 ■ TYPICAL: これがデフォルトです。プランに関する最も一般的な情報を表示します。パーティション・プルーニング、並列性および述語は、使用可能な場合のみ表示されます。 ■ ALL: 最高の詳細レベルです。TYPICAL レベルで表示される情報に加えて、パラレル実行サーバー用に生成された SQL 文（パラレルの場合のみ）が表示されます。 ■ SERIAL: TYPICAL と同じですが、プランがパラレルで実行される場合でもパラレル情報は表示されません。

結果の表示 : 例

PLAN TABLE に格納されている最新の EXPLAIN PLAN コマンドの結果を表示するには、次のように入力します。

```
SELECT * FROM table(DBMS_XPLAN.DISPLAY);
```

デフォルトの PLAN TABLE 以外の PLAN TABLE から表示するには、例えば「my_plan_table」の場合、次のように入力します。

```
SELECT * FROM table(DBMS_XPLAN.DISPLAY('my_plan_table'));
```

最小限のプラン情報を表示するには、次のように入力します。

```
SELECT * FROM table(DBMS_XPLAN.DISPLAY('plan_table', null,'basic'));
```

statement_id = foo など、foo で識別される文のプランを表示するには、次のように入力します。

```
SELECT * FROM table(DBMS_XPLAN.DISPLAY('plan_table', 'foo'));
```

使用上の注意

デフォルトでは、DISPLAY テーブル・ファンクションでレポートされるのは関連情報のみです。90-2 ページの「DBMS_XPLAN.DISPLAY を使用した PLAN TABLE の表示 : 例」で示す問合せは、パラレルで実行されていません。したがって、プランのパラレル化に関連する情報はレポートされません。次の例のように、問合せがパラレルで実行された場合のみ、パラレル情報がレポートされます。

パラレル情報を伴う PLAN TABLE の表示 : 例

```
Rem
Rem Execute an explain plan command for a parallel query
Rem
ALTER TABLE emp PARALLEL;
EXPLAIN PLAN for
SELECT * FROM emp e, dept d
   WHERE e.deptno = d.deptno
   AND e.ename    ='benoit'
   ORDER BY e.empno;

Rem
Rem Display the plan using the dbms_xplan.display() table function
Rem
SET LINESIZE 130
SET PAGESIZE 0
SELECT * FROM table(DBMS_XPLAN.DISPLAY);
```

この EXPLAIN PLAN の出力は、次のようになります。

```
-------------------------------------------------------------------------------
| Id  | Operation         |Name | Rows | Bytes | Cost | TQ    |IN-OUT|PQ Distrib |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |     |    1 |    50 |    3 |67,60  |      |           |
|   1 | SORT ORDER BY     |     |    1 |    50 |    3 |67,61  | P->S | QC(ORDER) |
|   2 | MERGE JOIN        |     |    1 |    50 |    3 |67,62  | P->P | RANGE     |
|   3 | SORT JOIN         |     |    4 |    72 |    3 |67,63  | PCWP |           |
|   4 | TABLE ACCESS FULL|DEPT  |    4 |    72 |    2 |67,64  | S->P | BROADCAST |
|*  5 | SORT JOIN         |     |    1 |    32 |    2 |67,65  | PCWP |           |
|*  6 | TABLE ACCESS FULL|EMP   |    1 |    32 |    2 |67,66  | PCWP |           |
-------------------------------------------------------------------------------
Predicate Information (identified by operation id)
   5 - access("E1"."DEPTNO"="D1"."DEPTNO")
       filter("E1"."DEPTNO"="D1"."DEPTNO")
   6 - filter("E1"."ENAME"'benoit')
```

問合せがパラレルの場合、並列性に関連する情報として、テーブル・キュー番号（TQ 列）、テーブル・キュー・タイプ（IN-OUT）およびキュー・テーブル配布方式（PQ Distrib）がレポートされます。

デフォルトでは、PLAN TABLE 内の複数のプランが、DISPLAY テーブル・ファンクションに渡された statement_id パラメータ（デフォルト値は NULL）と一致した場合に表示されるのは、最新の EXPLAIN PLAN コマンドに対応するプランのみです。したがって、各 EXPLAIN PLAN の後で、PLAN TABLE をパージする必要はありません。ただし、DISPLAY テーブル・ファンクションの最適な実行パフォーマンスを確保するため、たとえば TRUNCATE TABLE コマンドを使用して、PLAN TABLE を定期的にパージする必要があります。

使いやすくするため、DISPLAY テーブル・ファンクション以外にビューを定義し、そのビューを使用して、次の例のように、EXPLAIN PLAN コマンドの出力を表示できます。

ビューを使用した出力の表示 : 例

```
# define plan view
create view plan as select * from table(dbms_xplan.display);

# display the output of the last explain plan command
select * from plan;
```

91

DBMS_XSLPROCESSOR

DBMS_XSLPROCESSOR を使用すると、XML 文書の内容と構造にアクセスできます。

> **関連項目：** 詳細は、『Oracle9*i* XML API リファレンス - XDK および Oracle XML DB』を参照してください。

この章では、次の項目について説明します。

DBMS_XSLPROCESSOR の概要

eXtensible Stylesheet Language Transformation（XSLT）は、ソース・ツリーを結果ツリーに変換するルールを記述します。XSLT で表現する変換は、スタイルシートと呼ばれます。指定した変換は、パターンをスタイルシートで定義したテンプレートに関連付けることで実行されます。テンプレートは、結果ツリーの一部を作成するためにインスタンス化されます。XSL Processor の PL/SQL 実装は、W3C XSLT 草案（改訂 WD-xslt-19990813）に準拠しています。また、XSLT スタイルシートの読込み方法と影響を与える変換に関して、XSL Processor に要求される動作が含まれています。

この PL/SQL XSL Processor のデフォルト動作は、次のとおりです。

- DOM API からアクセス可能な結果ツリーが作成されます。
- エラー・ログが指定されていない場合、エラーは記録されません。ただし、解析に失敗すると、アプリケーション・エラーが発生します。

DBMS_XSLPROCESSOR のサブプログラム

表 91-1 DBMS_XSLPROCESSOR のサブプログラムの要約

サブプログラム	説明
「newProcessor()」91-3 ページ	新規のプロセッサ・インスタンスを戻します。
「processXSL()」91-3 ページ	入力された XML 文書を変換します。
「showWarnings()」91-6 ページ	警告をオンまたはオフにします。
「setErrorLog()」91-6 ページ	指定したファイルに送信するエラーを設定します。
「newStylesheet()」91-7 ページ	指定の入力と参照 URL を使用して新規のスタイルシートを作成します。
「transformNode()」91-7 ページ	指定のスタイルシートを使用して DOM ツリー内のノードを変換します。
「selectNodes()」91-8 ページ	指定のパターンと一致する DOM ツリーからノードを選択します。
「selectSingleNodes()」91-8 ページ	指定のパターンと一致するツリーから最初のノードを選択します。
「valueOf()」91-8 ページ	指定のパターンと一致するツリーから最初のノードの値を取得します。
「setParam()」91-9 ページ	スタイルシートに最上位レベルのパラメータを設定します。

表 91-1 DBMS_XSLPROCESSOR のサブプログラムの要約 （続き）

サブプログラム	説明
「removeParam()」91-9 ページ	最上位レベルのスタイルシート・パラメータを削除します。
「resetParams()」91-10 ページ	最上位レベルのスタイルシート・パラメータをリセットします。
「freeStylesheet()」91-10 ページ	スタイルシート・オブジェクトを解放します。
「freeProcessor()」91-10 ページ	プロセッサ・オブジェクトを解放します。

newProcessor()

新規のプロセッサ・インスタンスを戻します。このファンクションは、他のプロセッサ・メソッドを使用する必要がある場合、プロセッサのデフォルト動作の変更前にコールする必要があります。

構文

```
FUNCTION newProcessor RETURN Processor;
```

processXSL()

入力された XML 文書を変換します。プロセッサのデフォルト動作に対する変更は、このプロシージャのコール前に有効にしておく必要があります。処理に失敗すると、アプリケーション・エラーが発生します。指定できるオプションは、次の表のとおりです。

構文	説明

`FUNCTION processXSL(`
`  p  Processor,`
`  ss  Stylesheet,`
`  xmldoc DOMDocument),`
` RETURN DOMDocumentFragment;`
	指定の DOMDocument とスタイルシートを使用して、入力された XML 文書を変換し、結果のドキュメント・フラグメントを戻します。

`FUNCTION processXSL(`
`  p  Processor,`
`  ss  Stylesheet,`
`  url  VARCHAR2,`
` RETURN DOMDocumentFragment;`
	指定のドキュメント（URL として使用）とスタイルシートを使用して、入力された XML 文書を変換し、結果のドキュメント・フラグメントを戻します。

構文	説明

`FUNCTION processXSL(`
`  p  Processor,`
`  ss  Stylesheet,`
`  clb  CLOB)`
` RETURN DOMDocumentFragment;`
	指定のドキュメント（CLOB として使用）とスタイルシートを使用して、入力された XML 文書を変換し、結果のドキュメント・フラグメントを戻します。

`PROCEDURE processXSL(`
`  p  Processor,`
`  ss  Stylesheet,`
`  xmldoc  DOMDocument,`
`  dir  VARCHAR2,`
`  fileName VARCHAR2);`
	指定の DOMDocument とスタイルシートを使用して、入力された XML 文書を変換し、出力を指定のファイルに書き込みます。

`PROCEDURE processXSL(`
`  p  Processor,`
`  ss  Stylesheet,`
`  url  VARCHAR2,`
`  dir  VARCHAR2,`
`  fileName VARCHAR2);`
	指定の URL とスタイルシートを使用して、入力された XML 文書を変換し、出力を指定のディレクトリ内の指定したファイルに書き込みます。

`PROCEDURE processXSL(`
`  p  Processor,`
`  ss  Stylesheet,`
`  xmldoc  DOMDocument,`
`  cl IN OUT CLOB);`
	指定の DOMDocument とスタイルシートを使用して、入力された XML 文書を変換し、出力を CLOB に書き込みます。

`FUNCTION processXSL(`
`  p  Processor,`
`  ss  Stylesheet,`
`  xmldf  DOMDocumentFragment)`
` RETURN DOMDocumentFragment;`
	指定の DOMDocumentFragment とスタイルシートを使用して、入力された XML DocumentFragment を変換し、結果のドキュメント・フラグメントを戻します。

`PROCEDURE processXSL(`
`  p  Processor,`
`  ss  Stylesheet,`
`  xmldf  DOMDocumentFragment,`
`  dir  VARCHAR2,`
`  fileName VARCHAR2);`
	指定の DOMDocumentFragment とスタイルシートを使用して、入力された XML DocumentFragment を変換し、出力を指定のディレクトリ内の指定したファイルに書き込みます。

構文	説明
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldf DOMDocumentFragment, buf IN OUT VARCHAR2);	指定の DOMDocumentFragment とスタイルシートを使用して、入力された XML DocumentFragment を変換し、出力をバッファに書き込みます。
PROCEDURE processXSL(p Processor, ss Stylesheet, xmldf DOMDocumentFragment, cl IN OUT CLOB);	指定の DOMDocumentFragment とスタイルシートを使用して、入力された XML DocumentFragment を変換し、出力を CLOB に書き込みます。

パラメータ	IN / OUT	説明
p	(IN)	プロセッサ・インスタンス。
ss	(IN)	スタイルシート・インスタンス。
xmldoc	(IN)	変換する XML 文書。
url	(IN)	変換する情報の URL。
clb	(IN)	変換する情報が含まれる CLOB。
dir	(IN)	処理対象の出力ファイルを保存するディレクトリ。
fileName	(IN)	処理対象の出力ファイル。
cl	(IN/OUT)	処理対象の出力ファイルを保存する CLOB。
buf	(IN/OUT)	処理対象の出力ファイルを保存するバッファ。
xmldf	(IN)	変換する XML 文書フラグメント。

showWarnings()

警告をオン（TRUE）またはオフ（FALSE）にします。

構文

```
PROCEDURE showWarnings( p Processor,
                        yes BOOLEAN);
```

パラメータ	IN / OUT	説明
p	(IN)	プロセッサ・インスタンス。
yes	(IN)	設定するモード。警告を表示する場合は TRUE、表示しない場合は FALSE を設定します。

setErrorLog()

指定したファイルに送信するエラーを設定します。

構文

```
PROCEDURE setErrorLog( p Processor,
                       fileName VARCHAR2);
```

パラメータ	IN / OUT	説明
p	(IN)	プロセッサ・インスタンス。
fileName	(IN)	エラー・ログとして使用するファイルの完全パス。

newStylesheet()

新規のスタイルシート・インスタンスを作成して戻します。指定できるオプションは、次の表のとおりです。

構文	説明
FUNCTION newStylesheet(xmldoc DOMDocument, ref VARCHAR2) RETURN Stylesheet;	指定の DOMDocument と参照 URL を使用して新規のスタイルシート・インスタンスを作成し、戻します。
FUNCTION newStylesheet(inp VARCHAR2, ref VARCHAR2) RETURN Stylesheet;	指定の入力と参照 URL を使用して新規のスタイルシート・インスタンスを作成し、戻します。

パラメータ	IN / OUT	説明
xmldoc	(IN)	構成に使用する DOMDocument。
inp	(IN)	構成に使用する入力 URL。
ref	(IN)	参照 URL。

transformNode()

指定のスタイルシートを使用して DOM ツリー内のノードを変換し、結果を DOMDocumentFragment として戻します。

構文

```
FUNCTION transformNode( n DOMNode,
                        ss Stylesheet)
                        RETURN DOMDocumentFragment;
```

パラメータ	IN / OUT	説明
n	(IN)	変換する DOMNode。
ss	(IN)	使用するスタイルシート。

selectNodes()

指定のパターンと一致するノードを DOM ツリーから選択し、選択の結果を戻します。

構文

```
FUNCTION selectNodes( n DOMNode,
                      pattern VARCHAR2)
                      RETURN DOMNodeList;
```

パラメータ	IN / OUT	説明
n	(IN)	ツリーのルートの DOMNode。
pattern	(IN)	使用するパターン。

selectSingleNodes()

指定のパターンと一致するツリーから最初のノードを選択し、そのノードを戻します。

構文

```
FUNCTION selectSingleNodes( n DOMNode,
                            pattern VARCHAR2)
                            RETURN DOMNode;
```

パラメータ	IN / OUT	説明
n	(IN)	ツリーのルートの DOMNode。
pattern	(IN)	使用するパターン。

valueOf()

指定のパターンと一致するツリーから最初のノードの値を取得します。

構文

```
PROCEDURE valueOf( n DOMNode,
                   pattern VARCHAR2,
                   val OUT VARCHAR2);
```

パラメータ	IN / OUT	説明
n	(IN)	ツリーのルートの DOMNode。
pattern	(IN)	使用するパターン。
val	(OUT)	取得された値。

setParam()

スタイルシートに最上位レベルのパラメータを設定します。パラメータ値は、有効な XPath 式であることが必要です。必ずリテラル文字列値を引用します。

構文

```
PROCEDURE setParam( ss Stylesheet,
                    name VARCHAR2,
                    value VARCHAR2);
```

パラメータ	IN / OUT	説明
ss	(IN)	スタイルシート。
name	(IN)	パラメータの名前。
value	(IN)	パラメータの値。

removeParam()

最上位レベルのスタイルシート・パラメータを削除します。

構文

```
PROCEDURE removeParam( ss Stylesheet,
                       name VARCHAR2);
```

パラメータ	IN / OUT	説明
ss	(IN)	スタイルシート。
name	(IN)	パラメータの名前。

resetParams()

最上位レベルのスタイルシート・パラメータをリセットします。

構文

```
PROCEDURE resetParams( ss Stylesheet);
```

パラメータ	IN / OUT	説明
ss	(IN)	スタイルシート。

freeStylesheet()

スタイルシート・オブジェクトを解放します。

構文

```
PROCEDURE freestylesheet( ss Stylesheet);
```

パラメータ	IN / OUT	説明
ss	(IN)	スタイルシート。

freeProcessor()

プロセッサ・オブジェクトを解放します。

構文

```
PROCEDURE freeProccessor( p Processor);
```

パラメータ	IN / OUT	説明
p	(IN)	プロセッサ。

92

DEBUG_EXTPROC

DEBUG_EXTPROC パッケージによって、セッション内の extproc エージェントを起動できます。このユーティリティ・パッケージは、外部プロシージャをデバッグするのに役立ちます。

この章では、次の項目について説明します。

- DEBUG_EXTPROC の要件およびインストール時の注意
- DEBUG_EXTPROC の使用
- DBMS_EXTPROC サブプログラムの要約

DEBUG_EXTPROC の要件およびインストール時の注意

要件

Oracle アカウントに、パッケージに対する EXECUTE 権限と、CREATE LIBRARY 権限が必要です。

> **注意：** DEBUG_EXTPROC は、実行プロセスに連結できるデバッガを使用して、プラットフォームでのみ稼働します。

インストール時の注意

パッケージをインストールするには、スクリプト DBGEXTP.SQL を実行します。

- 'extproc' プロセスをデバッグする Oracle USER に、このパッケージをインストールまたはロードします。
- DEBUG_EXTPROC パッケージに対する EXECUTE 権限があることを確認します。

```
SELECT SUBSTR(OBJECT_NAME, 1, 20)
FROM USER_OBJECTS
WHERE OBJECT_NAME = 'DEBUG_EXTPROC';
```

- パッケージに対して EXECUTE 権限がある場合は、他のユーザーとしてこのパッケージをインストールできます。

DEBUG_EXTPROC の使用

使用の前提条件

外部プロシージャ 'extproc' エージェントを起動するには、リスナーが適切に構成されていることが前提です。

また、デバッグ処理を支援するために、デバッグ記号が付いた共有ライブラリを作成していることも前提です。C コンパイラのマニュアルをチェックして、適切な C コンパイラ・スイッチで、デバッグ記号が付いた共有ライブラリを作成してください。

使用上の注意

- Oracle に接続して、SQL*Plus または OCI プログラムから新規の Oracle セッションを起動します。
- プロシージャ DEBUG_EXTPROC.STARTUP_EXTPROC_AGENT を実行して、このセッションで extproc エージェントを起動します。たとえば、DEBUG_EXTPROC.STARTUP_EXTPROC_AGENT を実行します。extproc エージェントが終了してしまうので、このセッションは終了しないでください。
- このセッションで起動した extproc エージェントの PID を判別します。
- デバッガ（たとえば、gdb、dbx またはシステム固有なデバッガ）を使用して、extproc 実行ファイルをロードし、実行プロセスに連結します。
- ファンクション 'pextproc' にブレーク・ポイントを設定し、デバッガが実行を継続できるようにします。
- 最初に DEBUG_EXTPROC.STARTUP_EXTPROC_AGENT を実行した同じセッションで、外部プロシージャを実行します。
- デバッガがファンクション 'pextproc' でブレークします。この時点で、PL/SQL 外部ファンクションで参照する共有ライブラリがロードされ、ファンクションが解決されます。C ファンクションにブレーク・ポイントを設定し、デバッガが実行を継続できるようにします。

PL/SQL は、実行時に共有ライブラリをロードするため、使用するデバッガは、共有ライブラリから新規の記号を自動的に追跡管理できる場合とできない場合があります。デバッガ・コマンドをいくつか発行して、記号をロードできます（たとえば、gdb 内の 'share'）。

- デバッガは、C ファンクションでブレークします。デバッグ記号が付いた共有ライブラリを作成しておくことが前提です。
- デバッグを続行します。

DBMS_EXTPROC サブプログラムの要約

DEBUG_EXTPROC には、STARTUP_EXTPROC_AGENT プロシージャという 1 つのサブプログラムが含まれています。これにより、セッション内で extproc エージェント・プロセスを起動します。

STARTUP_EXTPROC_AGENT プロシージャ

このプロシージャは、セッションで extproc エージェント・プロセスを起動します。これにより、実行プロセスの PID を取得できます。この PID は、デバッガを使用して実行プロセスに連結するために必要です。

構文

```
DEBUG_EXTPROC.STARTUP_EXTPROC_AGENT;
```

93

UTL_COLL

UTL_COLL パッケージによって、問合せや更新を行うためのコレクション・ロケータを PL/SQL プログラムで使用できます。

この章では、次の項目について説明します。

UTL_COLL サブプログラムの要約

現在、このパッケージでサポートしているファンクションは、IS_LOCATOR のみです。

IS_LOCATOR ファンクション

このファンクションは、コレクション項目が実際にロケータかどうかを判別します。

構文

```
UTL_COLL.IS_LOCATOR (
   collection IN ANY)
  RETURNS BOOLEAN;
```

パラメータ

表 93-1 IS_LOCATOR ファンクションのパラメータ

パラメータ	説明
collection	ネストした表または VARRAY 項目

戻り値

表 93-2 IS_LOCATOR ファンクションの戻り値

戻り値	説明
1	コレクション項目はロケータです。
0	コレクション項目はロケータではありません。

プラグマ

WNDS、WNPS および RNPS の各プラグマの断言。

例

```
CREATE OR REPLACE TYPE list_t as TABLE OF VARCHAR2(20);
/

CREATE OR REPLACE TYPE phone_book_t AS OBJECT (
  pno  number,
  ph   list_t );
/

CREATE TABLE phone_book OF phone_book_t
      NESTED TABLE ph STORE AS nt_ph;
CREATE TABLE phone_book1 OF phone_book_t
      NESTED TABLE ph STORE AS nt_ph_1 RETURN LOCATOR;

INSERT INTO phone_book VALUES(1, list_t('650-633-5707','650-323-0953'));
INSERT INTO phone_book1 VALUES(1, list_t('415-555-1212'));

CREATE OR REPLACE PROCEDURE chk_coll IS
  plist list_t;
  plist1 list_t;
BEGIN
  SELECT ph INTO plist FROM phone_book WHERE pno=1;

  SELECT ph INTO plist1 FROM phone_book1 WHERE pno=1;

  IF (UTL_COLL.IS_LOCATOR(plist)) THEN
    DBMS_OUTPUT.PUT_LINE('plist is a locator');
  ELSE
    DBMS_OUTPUT.PUT_LINE('plist is not a locator');
  END IF;

  IF (UTL_COLL.IS_LOCATOR(plist1)) THEN
    DBMS_OUTPUT.PUT_LINE('plist1 is a locator');
  ELSE
    DBMS_OUTPUT.PUT_LINE('plist1 is not a locator');
  END IF;

END chk_coll;

SET SERVEROUTPUT ON
EXECUTE chk_coll;
```

94

UTL_ENCODE

UTL_ENCODE パッケージは、ホスト間でのデータ転送が可能になるように、RAW データを標準形式にエンコードするファンクションを提供します。UTL_ENCODE ファンクションを使用して、電子メール・テキストの本文をエンコードできます。また、このパッケージには、エンコード・ファンクションの対としてデコード・ファンクションも含まれています。デコード・ファンクションはエンコードの標準に準拠しており、送受信側において Oracle 以外のユーティリティに対応しています。

この章では、次の項目について説明します。

UTL_ENCODE サブプログラムの要約

表 94-1 UTL_ENCODE サブプログラム

サブプログラム	説明
「BASE64_ENCODE ファンクション」94-2 ページ	RAW 値のバイナリ表現を BASE 64 要素にエンコードし、RAW 文字列の形式で戻します。
「BASE64_DECODE ファンクション」94-3 ページ	BASE 64 にエンコードされた RAW 入力文字列を読み込み、元の RAW 値にデコードします。
「UUENCODE ファンクション」94-4 ページ	RAW 入力文字列を読み込み、対応する uuencode 形式の文字列にエンコードします。
「UUDECODE ファンクション」94-5 ページ	RAW の uuencode 形式の入力文字列を読み込み、対応する RAW 文字列にデコードします。
「QUOTED_PRINTABLE_ENCODE ファンクション」94-6 ページ	RAW 入力文字列を読み込み、引用符付きの出力可能な形式の対応文字列にエンコードします。
「QUOTED_PRINTABLE_DECODE ファンクション」94-7 ページ	引用符付きの出力可能な形式の varchar2 の入力文字列を読み込み、対応する RAW 文字列にデコードします。

BASE64_ENCODE ファンクション

このファンクションは、RAW 値のバイナリ表現を BASE 64 要素にエンコードし、RAW 文字列の形式で戻します。

構文

```
UTL_ENCODE.BASE64_ENCODE (
   r   IN RAW)
RETURN RAW;
```

プラグマ

```
pragma RESTRICT_REFERENCES(base64_encode, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 94-2 BASE64_ENCODE ファンクションのパラメータ

パラメータ	説明
r	エンコードされる RAW 値。デフォルトのパラメータもオプションのパラメータも設定されていません。

戻り値

表 94-3 BASE64_ENCODE ファンクションの戻り値

戻り値	説明
RAW	エンコードされた BASE 64 要素を含みます。

BASE64_DECODE ファンクション

このファンクションは、BASE 64 にエンコードされた RAW 入力文字列を読み込み、元の RAW 値にデコードします。

構文

```
UTL_ENCODE.BASE64_DECODE (
   r  IN RAW)
RETURN RAW;
```

プラグマ

```
pragma RESTRICT_REFERENCES(base64_decode, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 94-4 BASE64_DECODE ファンクションのパラメータ

パラメータ	説明
r	BASE 64 にエンコードされたデータを含む RAW 文字列。デフォルトのパラメータもオプションのパラメータも設定されていません。

戻り値

表 94-5 BASE64_DECODE ファンクションの戻り値

戻り値	説明
RAW	デコードされた文字列を含みます。

UUENCODE ファンクション

このファンクションは、RAW 入力文字列を読み込み、対応する uuencode 形式文字列にエンコードします。このファンクションの出力は累積されます。つまり、このファンクションを使用すると、大容量のデータ・ストリームをエンコードする場合、データ・ストリームを許容サイズの RAW 値に分割してエンコードし、単一のエンコードされた文字列に連結することができます。94-5 ページの「UUDECODE ファンクション」も参照してください。

構文

```
UTL_ENCODE.UUENCODE (
   r          IN RAW,
   type       IN PLS_INTEGER DEFAULT 1,
   filename   IN VARCHAR2 DEFAULT NULL,
   permission IN VARCHAR2 DEFAULT NULL) RETURN RAW;
```

プラグマ

```
pragma RESTRICT_REFERENCES(uuencode, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 94-6 UUENCODE ファンクションのパラメータ

パラメータ	説明
r	RAW 文字列。
type	uuencode にエンコードされた出力のタイプを含むオプションの数値パラメータ。 オプション： complete － 1 の値を持つ PL/SQL の事前定義定数 （デフォルト）。 header_piece middle_piece end_piece

表 94-6 UUENCODE ファンクションのパラメータ（続き）

パラメータ	説明
filename	uuencode ファイル名を含むオプションの varchar2 パラメータ。デフォルトは uuencode.txt です。
permission	アクセス権モードを含むオプションの varchar2 パラメータ。デフォルトは 0（テキスト文字列のゼロ）です。

戻り値

表 94-7 UUENCODE ファンクションの戻り値

戻り値	説明
RAW	uuencode 形式の文字列を含みます。

UUDECODE ファンクション

このファンクションは、RAW の uuencode 形式の入力文字列を読み込み、対応する RAW 文字列にデコードします。UUENCODE および UUDECODE のデータ・ストリームの累積性の説明は、94-4 ページの「UUENCODE ファンクション」を参照してください。

構文

```
UTL_ENCODE.UUDECODE (
   r  IN RAW)
RETURN RAW;
```

プラグマ

```
pragma RESTRICT_REFERENCES(uudecode, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 94-8 DUDECODE ファンクションのパラメータ

パラメータ	説明
r	uuencode にエンコードされたデータ文字列を含む RAW 文字列。デフォルトのパラメータもオプションのパラメータも設定されていません。

戻り値

表 94-9 UUDECODE ファンクションの戻り値

戻り値	説明
RAW	デコードされた RAW 文字列。

QUOTED_PRINTABLE_ENCODE ファンクション

このファンクションは、RAW 入力文字列を読み込み、引用符付きの出力可能な形式の対応文字列にエンコードします。

構文

```
UTL_ENCODE.QUOTED_PRINTABLE_ENCODE (
   r  IN RAW )
RETURN RAW;
```

プラグマ

```
pragma RESTRICT_REFERENCES(quoted_printable_encode, WNDS, RNDS,WNPS, RNPS);
```

パラメータ

表 94-10 QUOTED_PRINTABLE_ENCODE ファンクションのパラメータ

パラメータ	説明
r	RAW 文字列。デフォルトのパラメータもオプションのパラメータも設定されていません。

戻り値

表 94-11 QUOTED_PRINTABLE_ENCODE ファンクションの戻り値

戻り値	説明
RAW	引用符付きの出力可能な文字列を含みます。

QUOTED_PRINTABLE_DECODE ファンクション

このファンクションは、引用符付きの出力可能な形式の varchar2 の入力文字列を読み込み、対応する RAW 文字列にデコードします。

構文

```
UTL_ENCODE.QUOTED_PRINTABLE_DECODE (
   r  IN RAW )
RETURN RAW;
```

プラグマ

```
pragma RESTRICT_REFERENCES(quoted_printable_decode, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 94-12 QUOTED_PRINTABLE_DECODE ファンクションのパラメータ

パラメータ	説明
r	引用符付きの出力可能なデータ文字列を含む RAW 文字列。デフォルトのパラメータもオプションのパラメータも設定されていません。

戻り値

表 94-13 QUOTED_PRINTABLE_DECODE ファンクションの戻り値

戻り値	説明
RAW	デコードされた文字列。

95

UTL_FILE

UTL_FILE パッケージによって、ユーザーは PL/SQL プログラムでオペレーティング・システムのテキスト・ファイルの読込みと書込みができます。UTL_FILE は、オペレーティング・システムのストリーム・ファイル I/O の制約付きバージョンを提供します。

UTL_FILE の I/O 機能は、標準のオペレーティング・システムに備えられたストリーム・ファイルの I/O（OPEN、GET、PUT、CLOSE）機能に類似していますが、いくつかの点で制限があります。たとえば、FOPEN ファンクションをコールすると、ファイル・ハンドルが戻されます。後続の GET_LINE または PUT のコールでこのファイル・ハンドルを使用し、ファイルへのストリーム I/O を実行します。ファイルの I/O が完了した場合、FCLOSE をコールして出力を完了し、そのファイルに関連付けられたリソースを解放します。

> **注意：** UTL_FILE パッケージは、Oracle Procedure Builder が提供しているクライアント側の TEXT_IO パッケージと類似しています。サーバー・インプリメンテーションに対する制約事項には、UTL_FILE と TEXT_IO の間である程度の API に差異が必要です。PL/SQL のファイル I/O では、PL/SQL 例外を使用して、ユーザーにエラーが戻されます。

この章では、次の項目について説明します。

セキュリティ

UTL_FILE は、PL/SQL のクライアント側とサーバー側の両方で使用可能です。クライアント側の実装（テキスト I/O）は、通常のオペレーティング・システムでのファイル・アクセス権のチェックに準じています。ただし、サーバー側の実装は、権限を付与されたモードで実行される場合があります。この場合、アクセス可能なディレクトリへの制限が要求されます。

これまでは UTL_FILE ファンクションのアクセス可能なディレクトリは、UTL_FILE_DIR パラメータを使用して初期化ファイルに指定してきました。しかし、UTL_FILE_DIR へのアクセスはお薦めしません。UTL_FILE_DIR の置換には、CREATE DIRECTORY 機能の使用をお薦めします。UTL_FILE のアプリケーション管理者による柔軟できめ細かい制御を可能にするディレクトリ・オブジェクトは、動的に（データベースを停止しないで）メンテナンスでき、Oracle の他のツール製品との一貫性を保持しています。CREATE DIRECTORY 権限がデフォルトで付与されるのは、SYS と SYSTEM のみです。

注意： ディレクトリのアクセス検証には、UTL_FILE_DIR ではなく、CREATE DIRECTORY 機能を使用してください。

ファイルの所有権と保護

UNIX システムでは、FOPEN ファンクションで作成したファイルにはそのファイルの所有者、つまりインスタンスを実行するシャドウ・プロセスの所有者がいます。通常、この所有者は ORACLE です。FOPEN で作成したファイルは、UTL_FILE サブプログラムを使用して常に読込みと書込みができますが、非権限ユーザーがこれらのファイルを PL/SQL 以外で読み込む場合は、システム管理者によるアクセス権の設定が必要です。

例（UNIX 固有）

次の文があるとします。

```
SQL> CREATE DIRECTORY log_dir AS '/appl/gl/log';
SQL> GRANT READ ON DIRECTORY log_dir TO DBA;

SQL> CREATE DIRECTORY out_dir AS '/appl/gl/user'';
SQL> GRANT READ ON DIRECTORY user_dir TO PUBLIC;
```

有効でアクセス可能なファイルの場所とファイル名は、次のとおりです。

ファイルの場所	ファイル名	アクセスできるユーザー
/appl/gl/log	L12345.log	DBA 権限を持つユーザー
/appl/gl/user	u12345.tmp	すべてのユーザー

次のファイルの場所とファイル名は無効です。

ファイルの場所	ファイル名	無効の理由
/appl/gl/log/backup	L12345.log	# サブディレクトリにアクセスできません。
/APPL/gl/log	L12345.log	# ディレクトリの文字列は、O/S で要求されている大 / 小文字区別ルールに従う必要があります。
/appl/gl/log	backup/L1234.log	# ファイル名にディレクトリ・パスの一部が含まれていない可能性があります。
/user/tmp	L12345.log	# 対応する CREATE DIRECTORY コマンドが発行されていません。

注意： ユーザー・レベルのファイル許可はありません。UTL_FILE ディレクトリ・オブジェクト権限によって、ユーザーには、特定ディレクトリ内のすべてのファイルに対する読取りと書込みのアクセス権限が与えられます。

例外

表 95-1 UTL_FILE パッケージの例外

例外名	説明
INVALID_PATH	ファイルの場所が無効です。
INVALID_MODE	FOPEN の open_mode パラメータが無効です。
INVALID_FILEHANDLE	ファイル・ハンドルが無効です。
INVALID_OPERATION	要求どおりにファイルをオープンできないか、または操作できません。
READ_ERROR	読込み操作中にオペレーティング・システムのエラーが発生しました。
WRITE_ERROR	書込み操作中にオペレーティング・システムのエラーが発生しました。
INTERNAL_ERROR	PL/SQL 内の未指定エラー。

表 95-1 UTL_FILE パッケージの例外（続き）

例外名	説明
CHARSETMISMATCH	ファイルは FOPEN_NCHAR を使用してオープンされていますが、後の I/O 操作では PUTF または GET_LINE などの nonchar ファンクションを使用します。
FILE_OPEN	要求した操作は、ファイルがオープンしているため失敗しました。
INVALID_MAXLINESIZE	FOPEN() の MAX_LINESIZE 値が無効です。1 ～ 32767 の範囲内の値にしてください。
INVALID_FILENAME	ファイル名パラメータが無効です。
ACCESS_DENIED	ファイルの場所に対するアクセス許可が拒否されました。
INVALID_OFFSET	FSEEK() の ABSOLUTE_OFFSET パラメータが無効です。0（ゼロ）より大きく、ファイル内の合計バイド数より小さい値にしてください。
DELETE_FAILED	要求したファイルの削除操作に失敗しました。
RENAME_FAILED	要求したファイルの改名操作に失敗しました。

UTL_FILE 内のプロシージャが、NO_DATA_FOUND や VALUE_ERROR などの事前定義済みの PL/SQL 例外を発生させることもあります。

タイプ

FILE_TYPE の内容は、UTL_FILE パッケージ専用です。このレコードのコンポーネントを参照または変更しないでください。

```
TYPE file_type IS RECORD (
   id       BINARY_INTEGER,
   datatype BINARY_INTEGER);
```

UTL_FILE サブプログラムの要約

表 95-2 UTL_FILE サブプログラム

サブプログラム	説明
「FOPEN ファンクション」 95-7 ページ	入力用または出力用にファイルをオープンします。
「FOPEN_NCHAR ファンクション」 95-8 ページ	入力用または出力用にファイルを Unicode でオープンします。
「IS_OPEN ファンクション」 95-9 ページ	ファイル・ハンドルが、オープンしているファイルを参照しているかどうかを判別します。
「FCLOSE プロシージャ」 95-9 ページ	ファイルをクローズします。
「FCLOSE_ALL プロシージャ」 95-10 ページ	オープンしているファイル・ハンドルをすべてクローズします。
「GET_LINE プロシージャ」 95-11 ページ	オープンしているファイルからテキストを読み込みます。
「GET_LINE_NCHAR プロシージャ」 95-12 ページ	オープンしているファイルからテキストを Unicode で読み込みます。
「GET_RAW プロシージャ」 95-13 ページ	RAW 文字列値をファイルから読み込み、読み込んだバイトの数だけ、ファイルのポインタを前方に調整します。
「PUT プロシージャ」 95-13 ページ	ファイルに 1 つの文字を書き込みます。
「PUT_NCHAR プロシージャ」 95-14 ページ	ファイルに 1 つの Unicode 文字を書き込みます。
「PUT_RAW プロシージャ」 95-15 ページ	RAW データ値を入力として受け入れ、出力バッファに書き込みます。
「NEW_LINE プロシージャ」 95-16 ページ	1 つ以上の OS 固有の行終了記号をファイルに書き込みます。
「PUT_LINE プロシージャ」 95-16 ページ	ファイルに 1 行を書き込みます。OS 固有の行終了記号が追加されます。
「PUT_LINE_NCHAR プロシージャ」 95-17 ページ	ファイルに Unicode 行を 1 行書き込みます。
「PUTF プロシージャ」 95-18 ページ	書式付きの PUT プロシージャです。

表 95-2 UTL_FILE サブプログラム（続き）

サブプログラム	説明
「PUTF_NCHAR プロシージャ」95-20 ページ	書式付きの PUT_NCHAR プロシージャです。ファイルに 1 つの Unicode 文字を書式付きで書き込みます。
「FFLUSH プロシージャ」95-21 ページ	保留中のすべての出力データを物理的にファイルへ書き込みます。
「FSEEK プロシージャ」95-21 ページ	指定したバイトの数だけ、ファイル内でポインタを前方または後方に調整します。
「FREMOVE プロシージャ」95-22 ページ	ユーザーに十分な権限があるという前提で、ディスク・ファイルを削除します。
「FCOPY プロシージャ」95-23 ページ	ファイルの連続部分を新規に作成したファイルにコピーします。
「FGETPOS ファンクション」95-24 ページ	ファイル内の現行の相対オフセット位置をバイト数で戻します。
「FGETATTR プロシージャ」95-24 ページ	ディスク・ファイルの属性を読み込んで戻します。
「FRENAME プロシージャ」95-25 ページ	UNIX の mv ファンクションと同じように、既存のファイルに新規の名前を指定します。

注意： ファイルの場所とファイル名の各パラメータは、別々の文字列として FOPEN ファンクションに指定されるため、ファイルの場所は、アクセス可能なディレクトリ・オブジェクトの ALL_DIRECTORIES ビューで指定したアクセス可能なディレクトリのリストと照合してチェックできます。ファイルの場所と名前でシステム上の正しいファイル名を示し、そのディレクトリがアクセス可能であることが必要です。アクセス可能なディレクトリのサブディレクトリは、必ずしもアクセス可能である必要はありません。サブディレクトリも、ALL_DIRECTORIES オブジェクトと一致する完全パス名を使用して指定する必要があります。

UNIX での C シェル環境変数などのオペレーティング・システム固有のパラメータは、ファイルの場所またはファイル名のパラメータでは使用できません。

FOPEN ファンクション

このファンクションは、ファイルをオープンします。最大行サイズを指定でき、最大 50 ファイルまで同時にオープンできます。95-8 ページの「FOPEN_NCHAR ファンクション」も参照してください。

構文

```
UTL_FILE.FOPEN (
   location     IN VARCHAR2,
   filename     IN VARCHAR2,
   open_mode    IN VARCHAR2,
   max_linesize IN BINARY_INTEGER)
  RETURN file_type;
```

パラメータ

表 95-3 FOPEN ファンクションのパラメータ

パラメータ	説明
location	ファイルのディレクトリ位置。
filename	拡張子（ファイル・タイプ）も含めたファイル名。ディレクトリ・パスはありません。UNIX でのファイル名は、/ で終了できません。
open_mode	ファイルのオープン方法を指定します。次のモードがあります。 r ーテキストの読込み w ーテキストの書込み a ーテキストの追加 open_mode の値を使用して、存在しないファイルをオープンしようとすると、そのファイルは write モードで作成されます。
max_linesize	改行文字を含むこのファイルの 1 行当たりの最大文字数。（最小値は 1、最大値は 32767）。デフォルトは約 1000 バイトです。

戻り値

FOPEN は、そのファイルを操作する後続プロシージャすべてに渡す必要のあるファイル・ハンドルを戻します。ファイル・ハンドルの特定の内容は、UTL_FILE パッケージ専用であり、UTL_FILE のユーザーは、個々のコンポーネントの参照または変更を行わないでください。

表 95-4 FOPEN ファンクションの戻り値

戻り値	説明
file_type	オープン・ファイルのハンドル。

例外

INVALID_PATH: ファイルの位置または名前が無効です。
INVALID_MODE: open_mode 文字列が無効です。
INVALID_OPERATION: ファイルが要求どおりにオープンされませんでした。
INVALID_MAXLINESIZE: 指定した max_linesize は、大きすぎるか、小さすぎます。

FOPEN_NCHAR ファンクション

このファンクションは、指定した最大行サイズで入力用または出力用ファイルを Unicode でオープンします。最大 50 ファイルまで同時にオープンできます。このファンクションを使用すると、データベース・キャラクタ・セットではなく Unicode でテキスト・ファイルの読込みまたは書込みを実行できます。95-7 ページの「FOPEN ファンクション」も参照してください。

構文

```
UTL_FILE.FOPEN_NCHAR (
   location     IN VARCHAR2,
   filename     IN VARCHAR2,
   open_mode    IN VARCHAR2,
   max_linesize IN BINARY_INTEGER)
RETURN file_type;
```

パラメータ

表 95-5 FOPEN_NCHAR ファンクションのパラメータ

パラメータ	説明
location	ファイルのディレクトリ位置。
filename	ファイル名（拡張子を含む）。
open_mode	オープン・モード（r、w、a）。
max_linesize	改行文字を含むこのファイルの 1 行当たりの最大文字数。（最小値は 1、最大値は 32767）。

IS_OPEN ファンクション

このファンクションは、オープン・ファイルをファイル・ハンドルが識別しているかどうかをテストします。IS_OPEN は、ファイル・ハンドルが、オープン状態でクローズしていないファイルを示しているかどうかを通知するのみです。このファンクションは、ユーザーがファイル・ハンドルを使用しようとしたときに、オペレーティング・システムのエラーが発生しないことを保証するものではありません。

構文

```
UTL_FILE.IS_OPEN (
   file  IN FILE_TYPE)
  RETURN BOOLEAN;
```

パラメータ

表 95-6 IS_OPEN ファンクションのパラメータ

パラメータ	説明
file	FOPEN または FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。

戻り値

TRUE または FALSE

例外

なし。

FCLOSE プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルをクローズします。FCLOSE の実行時に、まだ書き込んでいないデータがバッファに残っていると、ファイルのクローズ時に WRITE_ERROR 例外を受け取る場合があります。

構文

```
UTL_FILE.FCLOSE (
   file IN OUT FILE_TYPE);
```

パラメータ

表 95-7 FCLOSE プロシージャのパラメータ

パラメータ	説明
file	FOPEN または FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。

例外

```
WRITE_ERROR
INVALID_FILEHANDLE
```

FCLOSE_ALL プロシージャ

このプロシージャは、セッションでオープンしているすべてのファイル・ハンドルをクローズします。これは、PL/SQL プログラムの例外終了などの非常時のクリーンアップ・プロシージャとして使用します。

> **注意：** FCLOSE_ALL は、ユーザーが保持しているオープン・ファイル・ハンドルの状態は変更しません。つまり、ファイルはクローズされていても、FCLOSE_ALL コール後のファイル・ハンドルの IS_OPEN テストでは TRUE が戻されます。FCLOSE_ALL の前にオープンされたファイルには、以降の読込み操作や書込み操作を行うことができません。

構文

```
UTL_FILE.FCLOSE_ALL;
```

パラメータ

なし。

例外

```
WRITE_ERROR
```

GET_LINE プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルからテキストを 1 行読み込んで、出力バッファ・パラメータに配置します。テキストは、ファイルまたは linesize パラメータの最後まで読み込まれますが、行の終了記号は含まれません。FOPEN に指定されている max_linesize を超える読込みはできません。

行がバッファに収まらない場合は、VALUE_ERROR 例外が発生します。ファイルの終わりに到達したためにテキストが読み込まれなかった場合は、NO_DATA_FOUND 例外が発生します。

行終了記号の文字はバッファに読み込まれないため、ブランク行を読み込むと空の文字列が戻されます。

buffer パラメータの最大サイズは、より小さいサイズを FOPEN に指定しないかぎり、32767 バイトです。使用しているプラットフォームにもよりますが、デフォルトは約 1000 バイトです。95-12 ページの「GET_LINE_NCHAR プロシージャ」も参照してください。

構文

```
UTL_FILE.GET_LINE (
   file        IN  FILE_TYPE,
   buffer      OUT VARCHAR2,
   linesize    IN  NUMBER,
   len         IN  PLS_INTEGER DEFAULT NULL);
```

パラメータ

表 95-8 GET_LINE プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル。 ファイルは読込み用（モード r）としてオープンする必要があります。そうでない場合は、INVALID_OPERATION 例外が発生します。
buffer	ファイルから読み込まれた行を受け取るデータ・バッファ。
linesize	読み込むバイトの最大数を指定します。
len	ファイルから読み込むバイト数。デフォルトは NULL です。NULL の場合、len は RAW の最大長とみなされます。

例外

```
INVALID_FILEHANDLE
INVALID_OPERATION
READ_ERROR
NO_DATA_FOUND
VALUE_ERROR
```

GET_LINE_NCHAR プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルからテキストを 1 行読み込んで、出力バッファ・パラメータに配置します。このファンクションを使用すると、データベース・キャラクタ・セットではなく Unicode でテキスト・ファイルの読込みを実行できます。95-11 ページの「GET_LINE プロシージャ」も参照してください。

構文

```
UTL_FILE.GET_LINE_NCHAR (
   file        IN  FILE_TYPE,
   buffer      OUT NVARCHAR2,
   len         IN  PLS_INTEGER DEFAULT NULL);
```

パラメータ

表 95-9 GET_LINE_NCHAR プロシージャのパラメータ

パラメータ	説明
file	FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。このファイルは読込み用（モード r）にオープンされる必要があります。ファイルを FOPEN_NCHAR ではなく FOPEN でオープンした場合は、CHARSETMISMATCH 例外が発生します。
buffer	ファイルから読み込まれた行を受け取るデータ・バッファ。
len	ファイルから読み込むバイト数。デフォルトは NULL です。NULL の場合、len は RAW の最大長とみなされます。

GET_RAW プロシージャ

このプロシージャは、RAW 文字列値をファイルから読み込み、読み込んだバイトの数だけ、ファイルのポインタを前方に調整します。

構文

```
UTL_FILE.GET_RAW (
   fid  IN  utl_file.file_type,
   r    OUT NOCOPY RAW,
   len  IN  PLS_INTEGER DEFAULT NULL);
```

パラメータ

表 95-10 GET_RAW プロシージャのパラメータ

パラメータ	説明
fid	ファイル ID。
r	RAW データ。
len	ファイルから読み込むバイト数。デフォルトは NULL です。NULL の場合、len は RAW の最大長とみなされます。

PUT プロシージャ

PUT プロシージャは、ファイル・ハンドルが示すオープン・ファイルに、バッファ・パラメータ内に格納されているテキスト文字列を書き込みます。このファイルは書込み操作用にオープンされる必要があります。PUT は、行終了記号を追加しません。行の終了には NEW_LINE を使用するか、または PUT_LINE を使用して行終了記号付きの完全な 1 行を書き込んでください。95-14 ページの「PUT_NCHAR プロシージャ」も参照してください。

buffer パラメータの最大サイズは、より小さいサイズを FOPEN に指定しないかぎり、32767 バイトです。使用しているプラットフォームにもよりますが、デフォルトは約 1000 バイトです。連続した PUT コールの全合計は、途中でバッファ・フラッシュをしないかぎり、32767 を超えることはできません。

構文

```
UTL_FILE.PUT (
   file      IN FILE_TYPE,
   buffer    IN VARCHAR2);
```

パラメータ

表 95-11 PUT プロシージャのパラメータ

パラメータ	説明
file	FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。このファイルは書込み用（モード w）にオープンされる必要があります。ファイルを FOPEN_NCHAR ではなく FOPEN でオープンした場合は、CHARSETMISMATCH 例外が発生します。
buffer	ファイルに書き込むテキストを含んだバッファ。 ファイルはモード w またはモード a を使用してオープンする必要があります。これ以外のモードを使用すると、INVALID_OPERATION 例外が発生します。

例外

```
INVALID_FILEHANDLE
INVALID_OPERATION
WRITE_ERROR
```

PUT_NCHAR プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルに、バッファ・パラメータ内に格納されているテキスト文字列を書き込みます。このファンクションを使用すると、データベース・キャラクタ・セットではなく Unicode でテキスト・ファイルの書込みを実行できます。95-13 ページの「PUT プロシージャ」も参照してください。

buffer パラメータの最大サイズは、より小さいサイズを FOPEN に指定しないかぎり、32767 バイトです。使用しているプラットフォームにもよりますが、デフォルトは約 1000 バイトです。連続した PUT コールの全合計は、途中でバッファ・フラッシュをしないかぎり、32767 を超えることはできません。

構文

```
UTL_FILE.PUT_NCHAR (
   file      IN FILE_TYPE,
   buffer    IN NVARCHAR2);
```

パラメータ

表 95-12 PUT_NCHAR プロシージャのパラメータ

パラメータ	説明
file	FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。ファイルを FOPEN_NCHAR ではなく FOPEN でオープンした場合は、CHARSETMISMATCH 例外が発生します。
buffer	ファイルに書き込むテキストを含んだバッファ。 ファイルはモード w またはモード a を使用してオープンする必要があります。これ以外のモードを使用すると、INVALID_OPERATION 例外が発生します。

PUT_RAW プロシージャ

このプロシージャは、RAW データ値を入力として受け入れ、出力バッファに書き込みます。3 番目の引数を TRUE に設定することで、バッファの自動フラッシュを要求できます。

buffer パラメータの最大サイズは、より小さいサイズを FOPEN に指定しないかぎり、32767 バイトです。使用しているプラットフォームにもよりますが、デフォルトは約 1000 バイトです。連続した PUT コールの全合計は、途中でバッファ・フラッシュをしないかぎり、32767 を超えることはできません。

構文

```
UTL_FILE.PUT_RAW (
   fid       IN utl_file.file_type,
   r         IN RAW,
   autoflush IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 95-13 PUT_RAW プロシージャのパラメータ

パラメータ	説明
fid (IN)	ファイル ID。
r (IN)	バッファに書き込まれた RAW データ。
autoflush (IN)	TRUE の場合は、出力バッファに値を書き込んだ後でフラッシュを実行します。デフォルトは FALSE です。

NEW_LINE プロシージャ

このプロシージャは、入力ファイル・ハンドルが示すファイルに、1 つ以上の行終了記号を書き込みます。行終了記号はプラットフォーム固有の文字や文字列であるため、このプロシージャは PUT とは異なります。

構文

```
UTL_FILE.NEW_LINE (
   file     IN FILE_TYPE,
   lines    IN NATURAL := 1);
```

パラメータ

表 95-14 NEW_LINE プロシージャのパラメータ

パラメータ	説明
file	FOPEN または FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。
lines	ファイルに書き込む行終了記号の数。

例外

```
INVALID_FILEHANDLE
INVALID_OPERATION
WRITE_ERROR
```

PUT_LINE プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルに、バッファ・パラメータ内に格納されているテキスト文字列を書き込みます。このファイルは書込み操作用にオープンされる必要があります。PUT_LINE は、プラットフォーム固有の行終了文字または文字列で行を終了します。

buffer パラメータの最大サイズは、より小さいサイズを FOPEN に指定しないかぎり、32767 バイトです。使用しているプラットフォームにもよりますが、デフォルトは約 1000 バイトです。連続した PUT コールの全合計は、途中でバッファ・フラッシュをしないかぎり、32767 を超えることはできません。

95-17 ページの「PUT_LINE_NCHAR プロシージャ」も参照してください。

構文

```
UTL_FILE.PUT_LINE (
   file      IN FILE_TYPE,
   buffer    IN VARCHAR2,
   autoflush IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 95-15 PUT_LINE プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル。
buffer	ファイルに書き込む行を含んだテキスト・バッファ。
autoflush	WRITE の後で、バッファをディスクにフラッシュします。

例外

```
INVALID_FILEHANDLE
INVALID_OPERATION
WRITE_ERROR
```

PUT_LINE_NCHAR プロシージャ

このプロシージャは、ファイル・ハンドルが示すオープン・ファイルに、バッファ・パラメータ内に格納されているテキスト文字列を書き込みます。このファンクションを使用すると、データベース・キャラクタ・セットではなく Unicode でテキスト・ファイルの書込みを実行できます。95-16 ページの「PUT_LINE プロシージャ」も参照してください。

buffer パラメータの最大サイズは、より小さいサイズを FOPEN に指定しないかぎり、32767 バイトです。使用しているプラットフォームにもよりますが、デフォルトは約 1000 バイトです。連続した PUT コールの全合計は、途中でバッファ・フラッシュをしないかぎり、32767 を超えることはできません。

構文

```
UTL_FILE.PUT_LINE_NCHAR (
   file    IN FILE_TYPE,
   buffer  IN NVARCHAR2);
```

パラメータ

表 95-16 PUT_LINE_NCHAR プロシージャのパラメータ

パラメータ	説明
file	FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。このファイルは書込み用（モード w）にオープンされる必要があります。ファイルを FOPEN_NCHAR ではなく FOPEN でオープンした場合は、CHARSETMISMATCH 例外が発生します。
buffer	ファイルに書き込む行を含んだテキスト・バッファ。

PUTF プロシージャ

このプロシージャは、書式化された PUT プロシージャです。これは、制限付きの printf() のように動作します。書式文字列には任意のテキストを指定できますが、文字列 %s と \n には次のような特別な意味があります。

文字列	意味
%s	この文字列を引数リスト内の次の引数の文字列値に置き換えます。
\n	適切なプラットフォーム固有の行終了記号に置き換えます。

95-20 ページの「PUTF_NCHAR プロシージャ」も参照してください。

構文

```
UTL_FILE.PUTF (
   file    IN FILE_TYPE,
   format  IN VARCHAR2,
   [arg1   IN VARCHAR2  DEFAULT NULL,
   . . .
   arg5    IN VARCHAR2  DEFAULT NULL]);
```

パラメータ

表 95-17 PUTF プロシージャのパラメータ

パラメータ	説明
file	FOPEN コールが戻すアクティブなファイル・ハンドル。
format	テキストや書式文字 \n と %s を含むことができる書式文字列。
arg1..arg5	1 ～ 5 個までのオプションの引数文字列。 引数文字列は、書式文字列内の %s 書式指定文字に、順序正しく置き換えられます。 引数より多い書式指定文字が書式パラメータ文字列内にある場合は、引数のない各 %s は空の文字列に置き換えられます。

例

次に、行を書き込む例を示します。

```
Hello, world!
I come from Zork with greetings for all earthlings.

my_world  varchar2(4) := 'Zork';
...
PUTF(my_handle, 'Hello, world!\nI come from %s with %s.\n',
                my_world,
                'greetings for all earthlings');
```

引数より多い %s 書式指定文字が書式パラメータ内にある場合は、対応する引数のない %s は空の文字列に置き換えられます。

例外

```
INVALID_FILEHANDLE
INVALID_OPERATION
WRITE_ERROR
```

PUTF_NCHAR プロシージャ

このプロシージャは、書式化された PUT_NCHAR プロシージャです。このファンクションを使用すると、データベース・キャラクタ・セットではなく Unicode でテキスト・ファイルの書込みを実行できます。95-18 ページの「PUTF プロシージャ」および 95-16 ページの「PUT_LINE プロシージャ」を参照してください。

buffer パラメータの最大サイズは、より小さいサイズを FOPEN に指定しないかぎり、32767 バイトです。使用しているプラットフォームにもよりますが、デフォルトは約 1000 バイトです。連続した PUT コールの全合計は、途中でバッファ・フラッシュをしないかぎり、32767 を超えることはできません。

構文

```
UTL_FILE.PUTF_NCHAR (
   file    IN FILE_TYPE,
   format  IN NVARCHAR2,
   [arg1   IN NVARCHAR2  DEFAULT NULL,
   . . .
   arg5    IN NVARCHAR2  DEFAULT NULL]);
```

パラメータ

表 95-18 PUTF_NCHAR プロシージャのパラメータ

パラメータ	説明
file	FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。このファイルは書込み用（モード w）にオープンされる必要があります。ファイルを FOPEN_NCHAR ではなく FOPEN でオープンした場合は、CHARSETMISMATCH 例外が発生します。
format	テキストや書式文字 \n と %s を含むことができる書式文字列。
arg1..arg5	1 ～ 5 個までのオプションの引数文字列。 引数文字列は、書式文字列内の %s 書式指定文字に、順序正しく置き換えられます。 引数より多い書式指定文字が書式パラメータ文字列内にある場合は、引数のない各 %s は空の文字列に置き換えられます。

FFLUSH プロシージャ

FFLUSH は、ファイル・ハンドルが示すファイルに、保留中のデータを物理的に書き込みます。ファイルに書き込むデータは通常バッファリングされます。FFLUSH プロシージャは、バッファリングされているデータを強制的にファイルに書き込みます。データは改行文字で終了する必要があります。

フラッシュは、まだオープンしているファイルを読み込む必要がある場合に役立ちます。たとえば、デバッグ・メッセージをファイルにフラッシュして、即時に読み込むことができます。

構文

```
UTL_FILE.FFLUSH (
   file  IN FILE_TYPE);
```

パラメータ

表 95-19 FFLUSH プロシージャのパラメータ

パラメータ	説明
file	FOPEN または FOPEN_NCHAR コールが戻すアクティブなファイル・ハンドル。

例外

```
INVALID_FILEHANDLE
INVALID_OPERATION
WRITE_ERROR
```

FSEEK プロシージャ

このプロシージャは、指定したバイトの数だけ、ファイル内でポインタを前方または後方に調整します、

オフセットの場合、このファンクションはバイト・オフセットを検索します。試行の前に、ファイルの最後または先頭に達した場合、このプロシージャは最後または先頭の行をそれぞれ戻します。

loc の場合、このプロシージャは、バイト数で指定された絶対位置を検索します。

構文

```
UTL_FILE.FSEEK (
   fid             IN utl_file.file_type,
   absolute_offset IN PL_INTEGER DEFAULT NULL,
   relative_offset IN PLS_INTEGER DEFAULT NULL);
```

パラメータ

表 95-20 FSEEK プロシージャのパラメータ

パラメータ	説明
fid (in)	ファイル ID。
absolute_offset (IN)	検索先の絶対位置。デフォルト = NULL。
relative_offset (IN)	前方または後方に検索するバイト数。整数 = 前方、負数 = 後方、0（ゼロ）= 現在の位置、デフォルト = NULL。

注意

このプロシージャを使用すると、最初にファイルのクローズと再オープンを行わずに、ファイル内の前の行を読み込むことができます。ナビゲートするバイト数を把握しておく必要があります。

FREMOVE プロシージャ

このプロシージャは、ユーザーに十分な権限があるという前提で、ディスク・ファイルを削除します。

構文

```
UTL_FILE.FREMOVE (
   location IN VARCHAR2,
   filename IN VARCHAR2);
```

パラメータ

表 95-21 FREMOVE プロシージャのパラメータ

パラメータ	説明
location (IN)	ファイルのディレクトリ位置。ALL_DIRECTORIES の DIRECTORY_NAME（大 / 小文字が区別されます）。
filename (IN)	削除されるファイルの名前。

注意

FREMOVE ファンクションでは、ファイルの削除前に権限の検証は行われません。ファイルとディレクトリに対するアクセス権限は、O/S によって検証されます。障害時には、例外が戻されます。

FCOPY プロシージャ

このプロシージャは、ファイルの連続部分を新規に作成したファイルにコピーします。start_line パラメータと end_line パラメータが省略されている場合、デフォルトでは、ファイル全体がコピーされます。ソース・ファイルは読取りモードでオープンします。宛先ファイルは書込みモードでオープンします。ソース・ファイルのコピー操作では、ファイルの一部を選択するために、開始および終了の行番号をオプションで指定できます。

構文

```
UTL_FILE.FCOPY (
   location   IN VARCHAR2,
   filename   IN VARCHAR2,
   dest_dir   IN VARCHAR2,
   dest_file  IN VARCHAR2,
   start_line IN PLS_INTEGER DEFAULT 1,
   end_line   IN PLS_INTEGER DEFAULT NULL);
```

パラメータ

表 95-22 FCOPY プロシージャのパラメータ

パラメータ	説明
location (IN)	ソース・ファイルのディレクトリ位置。ALL_DIRECTORIES ビューの DIRECTORY_NAME（大 / 小文字が区別されます）。
filename (IN)	コピー対象のソース・ファイル。
dest_dir (IN)	宛先ファイルが作成される宛先ディレクトリ。
dest_file (N)	ソース・ファイルから作成された宛先ファイル。
start_line (IN)	コピー操作を開始する行の番号。デフォルトは、最初の行の 1 です。
end_line (IN)	コピー操作を停止する行の番号。デフォルトは、ファイルの最後を示す NULL です。

FGETPOS ファンクション

このファンクションは、ファイル内の現行の相対オフセット位置をバイト数で戻します。

構文

```
UTL_FILE.FGETPOS (
   fileid IN file_type)
 RETURN PLS_INTEGER;
```

パラメータ

表 95-23 FGETPOS のパラメータ

パラメータ	説明
fileid (IN)	ソース・ファイルのディレクトリ位置。

戻り値

FGETPOS は、オープン・ファイルの相対的なオフセット位置をバイト数で戻します。ファイルがオープンしていない場合は、例外が発生します。ファイルの先頭では、0（ゼロ）が戻されます。

FGETATTR プロシージャ

このプロシージャは、ディスク・ファイルの属性を読み込んで戻します。

構文

```
UTL_FILE.FGETATTR(
   location    IN VARCHAR2,
   filename    IN VARCHAR2,
   exists      OUT BOOLEAN,
   file_length OUT NUMBER,
   blocksize   OUT NUMBER);
```

パラメータ

表 95-24 FGETATTR プロシージャのパラメータ

パラメータ	説明
location	ソース・ファイルのディレクトリ位置。ALL_DIRECTORIES ビューの DIRECTORY_NAME（大 / 小文字が区別されます）。
filename	ソース・ファイルの名前。
exists	ファイルの存在を示すブール値。
file_length	ファイルの長さ（バイト単位）。ファイルが存在しない場合は NULL です。
blocksize	ファイル・システムのブロック・サイズ（バイト単位）。ファイルが存在しない場合は NULL です。

FRENAME プロシージャ

このプロシージャは、UNIX の mv ファンクションと同じように、既存のファイルに新規の名前を指定します。ソース・ディレクトリと宛先ディレクトリの両方に対するアクセス権の付与が必要です。宛先ディレクトリにファイルがある場合は、overwrite パラメータを使用して、そのファイルを上書きするかどうかを指定できます。デフォルトは、上書きしない場合の FALSE です。

構文

```
UTL_FILE.FRENAME (
   location  IN VARCHAR2,
   filename  IN VARCHAR2,
   dest_dir  IN VARCHAR2,
   dest_file IN VARCHAR2,
   overwrite IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 95-25 FRENAME のパラメータ

パラメータ	説明
location (IN)	ソース・ファイルのディレクトリ位置。ALL_DIRECTORIES ビューの DIRECTORY_NAME（大 / 小文字が区別されます）。
filename (IN)	改名対象のソース・ファイル。
dest_dir (IN)	宛先ファイルの宛先ディレクトリ。ALL_DIRECTORIES ビューの DIRECTORY_NAME（大 / 小文字が区別されます）。
dest_file (N)	ファイルの新しい名前。
overwrite (IN)	デフォルトは FALSE です。

96

UTL_HTTP

UTL_HTTP パッケージは、SQL と PL/SQL から Hypertext Transfer Protocol（HTTP）のコールアウトを行います。これを使用すると、HTTP を経由してインターネット上のデータにアクセスできます。

UTL_HTTP を使用すると、Web（HTTP）サーバーと通信する PL/SQL プログラムを記述できます。また、UTL_HTTP には、SQL 問合せで使用できるファンクションが含まれています。また、このパッケージは、HTTPS と呼ばれる Secure Sockets Layer（SSL）プロトコルを使用して、直接または HTTP プロキシ経由で HTTP をサポートします。インターネット関連の他のデータアクセス・プロトコル（ファイル転送プロトコル（FTP）または Gopher プロトコル）は、これらのプロトコルをサポートする HTTP プロキシ・サーバーを使用してサポートされます。

パッケージで HTTPS を使用して Web サイトからデータをフェッチする場合は、Oracle Wallet Manager を使用して Oracle Wallet を設定する必要があります。HTTPS 以外を使用したフェッチの場合は、Oracle Wallet は必要ありません。

関連項目：

- 第 102 章「UTL_URL」
- 第 100 章「UTL_SMTP」
- Wallet Manager の詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

この章では、次の項目について説明します。

UTL_HTTP の定数、タイプおよびフロー

UTL_HTTP の定数

表 96-1 UTL_HTTP の定数

定数および構文	用途
`HTTP_VERSION_1_0 CONSTANT VARCHAR2(10) := 'HTTP/1.0';`	`begin_request` ファンクションで使用できる HTTP バージョン 1.0 を示します。
`HTTP_VERSION_1 CONSTANT VARCHAR2(10) := 'HTTP/1.1';`	`begin_request` ファンクションで使用できる HTTP バージョン 1.1 を示します。
`DEFAULT_HTTP_PORT CONSTANT PLS_INTEGER := 80;`	Web サーバーまたはプロキシ・サーバーがリスニングするデフォルトの TCP/IP ポート（80）。
`DEFAULT_HTTPS_PORT CONSTANT PLS_INTEGER := 443;`	HTTPS Web サーバーがリスニングするデフォルトの TCP/IP ポート（443）。

HTTP 1.1 のすべてのステータス・コードは次のとおりです。

`HTTP_CONTINUE CONSTANT PLS_INTEGER := 100;`	—
`HTTP_SWITCHING_PROTOCOLS CONSTANT PLS_INTEGER := 101;`	
`HTTP_OK CONSTANT PLS_INTEGER := 200;`	—
`HTTP_CREATED CONSTANT PLS_INTEGER := 201;`	—
`HTTP_ACCEPTED CONSTANT PLS_INTEGER := 202;`	—
`HTTP_NON_AUTHORITATIVE_INFO CONSTANT PLS_INTEGER := 203;`	
`HTTP_NO_CONTENT CONSTANT PLS_INTEGER := 204;`	—
`HTTP_RESET_CONTENT CONSTANT PLS_INTEGER := 205;`	—
`HTTP_PARTIAL_CONTENT CONSTANT PLS_INTEGER := 206;`	—
`HTTP_MULTIPLE_CHOICES CONSTANT PLS_INTEGER := 300;`	—
`HTTP_MOVED_PERMANENTLY CONSTANT PLS_INTEGER := 301;`	
`HTTP_FOUND CONSTANT PLS_INTEGER := 302;`	—
`HTTP_SEE_OTHER CONSTANT PLS_INTEGER := 303;`	—
`HTTP_NOT_MODIFIED CONSTANT PLS_INTEGER := 304;`	—
`HTTP_USE_PROXY CONSTANT PLS_INTEGER := 305;`	—
`HTTP_TEMPORARY_REDIRECT CONSTANT PLS_INTEGER := 307;`	

表 96-1 UTL_HTTP の定数（続き）

定数および構文	用途
HTTP_BAD_REQUEST CONSTANT PLS_INTEGER := 400;	—
HTTP_UNAUTHORIZED CONSTANT PLS_INTEGER := 401;	—
HTTP_PAYMENT_REQUIRED CONSTANT PLS_INTEGER := 402;	—
HTTP_FORBIDDEN CONSTANT PLS_INTEGER := 403;	—
HTTP_NOT_FOUND CONSTANT PLS_INTEGER := 404;	—
HTTP_NOT_ACCEPTABLE CONSTANT PLS_INTEGER := 406;	—
HTTP_PROXY_AUTH_REQUIRED CONSTANT PLS_INTEGER := 407;	
HTTP_REQUEST_TIME_OUT CONSTANT PLS_INTEGER := 408;	—
HTTP_CONFLICT CONSTANT PLS_INTEGER := 409;	—
HTTP_GONE CONSTANT PLS_INTEGER := 410;	—
HTTP_LENGTH_REQUIRED CONSTANT PLS_INTEGER := 411;	—
HTTP_PRECONDITION_FAILED CONSTANT PLS_INTEGER := 412;	
HTTP_REQUEST_ENTITY_TOO_LARGE CONSTANT PLS_INTEGER := 413;	
HTTP_REQUEST_URI_TOO_LARGE CONSTANT PLS_INTEGER := 414;	
HTTP_UNSUPPORTED_MEDIA_TYPE CONSTANT PLS_INTEGER := 415;	
HTTP_REQ_RANGE_NOT_SATISFIABLE CONSTANT PLS_INTEGER := 416;	
HTTP_EXPECTATION_FAILED CONSTANT PLS_INTEGER := 417;	
HTTP_NOT_IMPLEMENTED CONSTANT PLS_INTEGER := 501;	—
HTTP_BAD_GATEWAY CONSTANT PLS_INTEGER := 502;	—
HTTP_SERVICE_UNAVAILABLE CONSTANT PLS_INTEGER := 503;	
HTTP_GATEWAY_TIME_OUT CONSTANT PLS_INTEGER := 504;	—
HTTP_VERSION_NOT_SUPPORTED CONSTANT PLS_INTEGER := 505;	—

UTL_HTTP のタイプ

UTL_HTTP では次のタイプを使用します。

REQ タイプ

この PL/SQL レコード・タイプを使用して、HTTP 要求を表します。

構文

```
TYPE req IS RECORD (
   url           VARCHAR2(32767),
   method        VARCHAR2(64),
   http_version  VARCHAR2(64),
);
```

パラメータ

表 96-2 REQ タイプのパラメータ

パラメータ	説明
url	HTTP 要求の URL。begin_request により要求が作成された後に設定されます。
method	URL により識別されたリソースで実行されるメソッド。begin_request により要求が作成された後に設定されます。
http_version	要求の送信に使用される HTTP プロトコル。begin_request により要求が作成された後に設定されます。

使用上の注意

API begin_request から REQ に戻される情報は読取り専用です。レコードのフィールド値を変更しても、要求には影響はありません。

REQ レコード・タイプには名前が private_ という接頭辞で始まる他のフィールドがあります。このフィールドはプライベートで、UTL_HTTP パッケージの実装で使用することを目的としています。これらのフィールドは変更しないでください。

RESP タイプ

この PL/SQL レコード・タイプは、HTTP 応答を表します。

構文

```
TYPE resp IS RECORD (
   status_code    PLS_INTEGER,
   reason_phrase  VARCHAR2(256),
   http_version   VARCHAR2(64),
);
```

パラメータ

表 96-3 RESP タイプのパラメータ

パラメータ	説明
status_code	Web サーバーにより戻されるステータス・コード。Web サーバーにより処理される HTTP 要求の結果を示す 3 桁の整数です。get_response により応答が処理された後に設定されます。
reason_phrase	ステータス・コードを記述する Web サーバーにより戻される短いテキスト・メッセージ。Web サーバーにより処理される HTTP 要求の結果を簡潔に説明します。get_response により応答が処理された後に設定されます。
http_version	HTTP 応答に使用される HTTP プロトコルのバージョン。get_response により応答が処理された後に設定されます。

使用上の注意

API get_response から RESP に戻される情報は読取り専用です。RESP レコード・タイプには名前が private_ という接頭辞で始まる他のフィールドがあります。このフィールドはプライベートで、UTL_HTTP パッケージの実装で使用することを目的としています。これらのフィールドは変更しないでください。

COOKIE および COOKIE_TABLE タイプ

COOKIE タイプは、HTTP Cookie を表す PL/SQL レコード・タイプです。COOKIE_TABLE タイプは、HTTP Cookie のコレクションを表す PL/SQL 索引付き表タイプです。

構文

```
TYPE cookie IS RECORD (
   name  VARCHAR2(256),
   value  VARCHAR2(1024),
   domain  VARCHAR2(256),
   expire  TIMESTAMP WITH TIME ZONE,
   path  VARCHAR2(1024),
   secure  BOOLEAN,
   version  PLS_INTEGER,
   comment  VARCHAR2(1024)
);
TYPE cookie_table IS TABLE OF cookie INDEX BY binary_integer;
```

COOKIE レコード・タイプのフィールド

表 96-4 に、COOKIE および COOKIE_TABLE レコード・タイプのフィールドを示します。

表 96-4 COOKIE および COOKIE_TABLE タイプのフィールド

フィールド	説明
name	HTTP Cookie の名前。
value	Cookie の値。
domain	Cookie が有効なドメイン。
expire	Cookie の期限が切れる時刻。
path	Cookie を適用する URL のサブセット。
secure	セキュリティ保護された手段でのみ Web サーバーに Cookie を戻す必要があります。
version	Cookie が準拠する HTTP Cookie 仕様のバージョン。Netscape の Cookie の場合、このフィールドは NULL です。
comment	Cookie の用途を記述したコメント。Netscape の Cookie の場合、このフィールドは NULL です。

使用上の注意

PL/SQL プログラムは通常、UTL_HTTP パッケージに格納された Cookie 情報を調査または変更しません。Cookie は、パッケージが透過的に維持します。UTL_HTTP パッケージ内部で維持され、データベース・セッションの期間のみ存続します。データベース・セッションの期間以降も Cookie の維持を必要とする PL/SQL アプリケーションは、get_cookies を使用して Cookie を読み込み、データベース表に永続的に格納し、次のデータベース・セッションで add_cookies を使用してパッケージに Cookie を戻し、再度格納できます。cookie レコードのすべてのフィールド（ただし comment フィールドは除く）は格納する必要があります。Cookie 情報は変更しないでください。変更すると、Web サーバーでアプリケーション・エラーが発生したり、PL/SQL および Web サーバー・アプリケーションのセキュリティに障害が発生する場合があります。96-15 ページの「例 : Cookie の取出しおよびリストア」を参照してください。

CONNECTION タイプ

この PL/SQL レコード・タイプを使用して、HTTP 1.1 プロトコルの仕様に従い、HTTP 要求が完了した後に永続的に保持されるネットワーク接続のリモート・ホストおよび TCP/IP ポートを表します。永続的なネットワーク接続は、後続の HTTP 要求により同じホストおよびポートに対して再利用できる可能性があります。ネットワーク接続の待機時間が回避されるため、後続の HTTP 要求を高速に処理できる場合があります。connection_table は、接続の PL/SQL 表です。

Web サーバーに直接 HTTP を永続的に接続する場合は、host および port フィールドに Web サーバーのホスト名および TCP/IP ポート番号が含まれます。proxy_host および proxy_port フィールドは設定されません。プロキシを使用して Web サーバーへの接続に以前使用されていた HTTP に永続的に接続する場合は、proxy_host および proxy_port フィールドにプロキシ・サーバーのホスト名および TCP/IP ポート番号が含まれます。host および port フィールドは設定されません。これは、プロキシ・サーバーに接続している間、永続的接続が特定のターゲット Web サーバーにバインドされないことを示します。プロキシ・サーバーへの HTTP の永続的接続を使用して、プロキシを使用するターゲット Web サーバーにアクセスできます。

ssl フィールドは、Secure Sockets Layer（SSL）が HTTP の永続的接続で使用されているかどうかを示します。HTTPS 要求は、SSL を介して作成された HTTP 要求です。プロキシを使用して接続した HTTPS（SSL）の永続的接続の場合は、host および port フィールドにターゲット HTTPS Web サーバーのホスト名および TCP/IP ポート番号が含まれます。これらのフィールドは常に設定されます。プロキシ・サーバーを使用して HTTPS Web サーバーに永続的に HTTPS 接続を行った場合は、同じターゲット Web サーバーに別の要求を行う場合にのみ再利用できます。

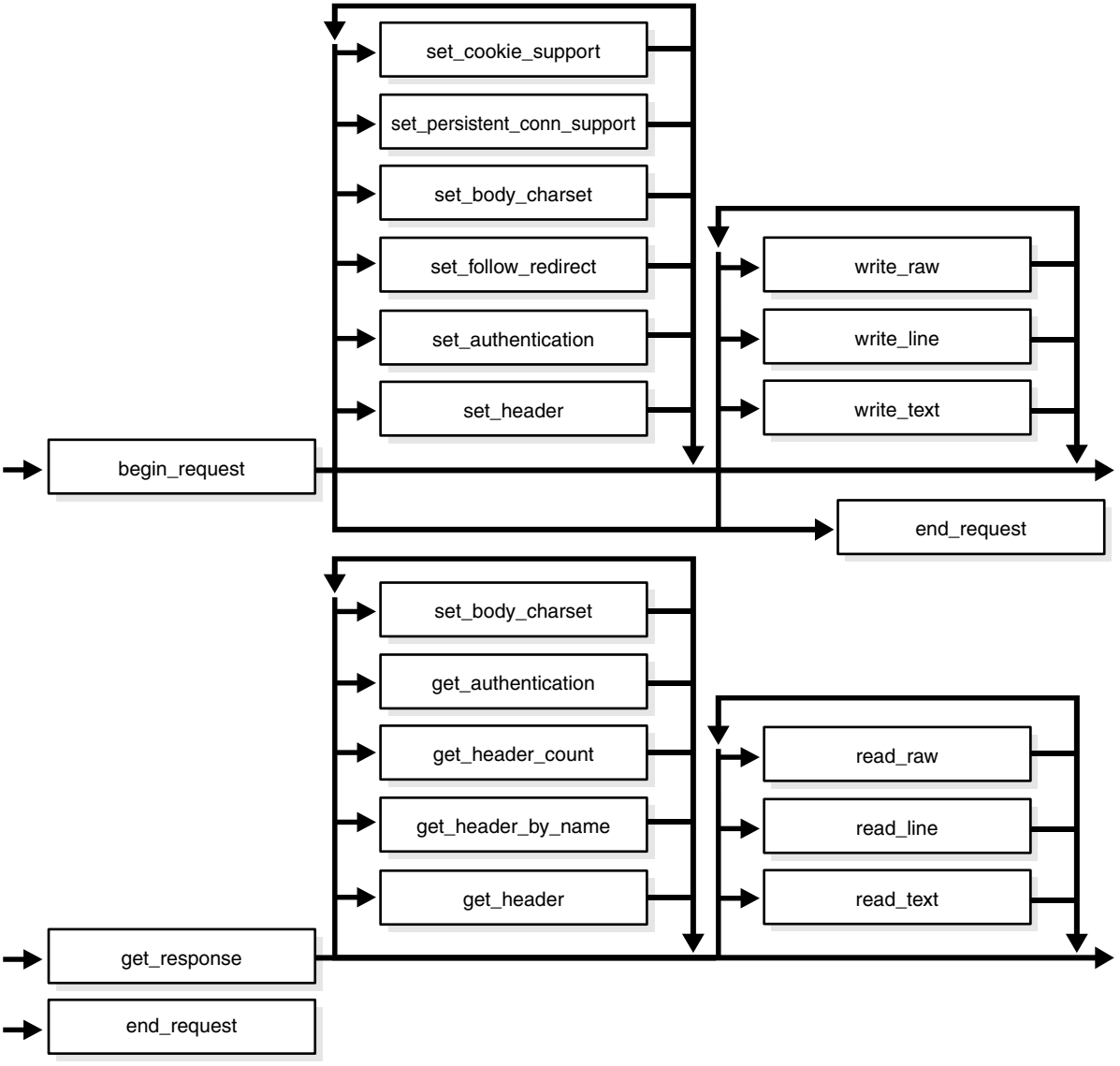
構文

```
TYPE connection IS RECORD (
   host  VARCHAR2(256),
   port  PLS_INTEGER,
   proxy_host  VARCHAR2(256),
   proxy_port  PLS_INTEGER,
   ssl  BOOLEAN
);
TYPE connection_table IS TABLE OF connection INDEX BY BINARY_INTEGER;
```

UTL_HTTP のフロー

UTL_HTTP パッケージは HTTP プロトコルへのアクセスを提供します。API は、図 96-1 に表示されている順序でコールしてください。異なる順序でコールすると例外が発生します。

図 96-1 Core UTL_HTTP パッケージのフロー



随時、次をコールできます。

- Cookie を操作する非プロトコル API
  - get_cookie_count
  - get_cookies
  - add_cookies
  - clear_cookies
- 永続的接続
  - get_persistent_conn_count
  - get_persistent_conns
  - close_persistent_conn
  - close_persistent_conns
- UTL_HTTP パッケージの属性および構成を現行のセッションで操作する API
  - set_proxy
  - get_proxy
  - set_cookie_support
  - get_cookie_support
  - set_follow_redirect
  - get_follow_redirect
  - set_body_charset
  - get_body_charset
  - set_persistent_conn_support
  - get_persistent_conn_support
  - set_detailed_excp_support
  - get_detailed_excp_support
  - set_wallet
  - set_transfer_timeout
  - get_transfer_timeout

- UTL_HTTP パッケージの最後の詳細な例外コードおよびメッセージを現行のセッションで取り出す API
  - get_detailed_sqlcode
  - get_detailed_sqlerrm

注意： 要求および応答の API によっては、現行のセッションでパッケージの属性および構成を操作する API と同じ名前を設定できる場合があります。これらは、要求または応答を操作する API のオーバーロード・バージョンです。

UTL_HTTP の例外

表 96-5 に、UTL_HTTP パッケージの API が呼び出す可能性のある例外を示します。デフォルトでは、要求の実行に失敗すると UTL_HTTP により request_failed 例外が発生します。set_detailed_excp_support により詳細な例外が発生するようにパッケージが設定されている場合、残りの例外は直接発生します。ただし、end_of_body 例外は除きます。この例外は設定にかかわらず、read_text、read_line および read_raw により呼び出されます。

表 96-5 UTL_HTTP の例外

例外	エラー・コード	理由	発生した場所
request_failed	29273	要求の実行に失敗しました。	detailed_exception が無効である場合、HTTP 要求または応答の API。
bad_argument	29261	API に渡された引数が不適切です。	detailed_exception が有効である場合、HTTP 要求または応答の API。
bad_url	29262	要求された URL の構成が不適切です。	detailed_exception が有効な場合の begin_request。
protocol_error	29263	Web サーバーとの通信で HTTP プロトコルのエラーが発生しました。	detailed_exception が有効な場合の set_header、get_response、read_raw、read_text および read_line。
unknown_scheme	29264	要求された URL のスキームが不明です。	detailed_exception が有効な場合の begin_request および get_response。

表 96-5 UTL_HTTP の例外（続き）

例外	エラー・コード	理由	発生した場所
header_not_found	29265	ヘッダーが見つかりません。	detailed_exception が有効な場合の get_header および get_header_by_name。
end_of_body	29266	HTTP 応答本体の終わりに到達しています。	detailed_exception が有効な場合の read_raw、read_text および read_line。
illegal_call	29267	HTTP 要求の現在の状態では UTL_HTTP へのコールが不正です。	detailed_exception が有効な場合の set_header、set_authentication および set_persistent_conn_support。
http_client_error	29268	get_response で、応答ステータス・コードがクライアントでのエラー発生を示しています（4xx のステータス・コード）。または、begin_request で、HTTP プロキシが、プロキシ経由の HTTP 要求時に 4xx のステータス・コードを戻しています。	detailed_exception が有効な場合の begin_request および get_response。
http_server_error	29269	get_response で、応答ステータス・コードがクライアントでのエラー発生を示しています（5xx のステータス・コード）。または、begin_request で、HTTP プロキシが、プロキシ経由の HTTP 要求時に 5xx のステータス・コードを戻しています。	detailed_exception が有効な場合の begin_request および get_response。
too_many_requests	29270	オープンされている要求または応答が多すぎます。	detailed_exception が有効な場合の begin_request。
partial_multibyte_exception	29275	完全に読み込まれた文字はありません。応答本体の終わりにマルチバイト・キャラクタの一部が検出されました。	detailed_exception が有効な場合の read_text および read_line。
transfer_timeout	29276	データが読み込まれず、読込みタイムアウトが発生しました。	detailed_exception が有効な場合の read_text および read_line。

注意： partial_multibyte_char および transfer_timeout の例外は、UTL_TCP で定義された同じ例外の複製です。これらは、パッケージを使用するときに UTL_TCP の知識を必要としなくても済むように、このパッケージで定義されています。これらの例外は複製であるため、このパッケージの partial_multibyte_char および transfer_timeout 例外を補足する例外ハンドルは、UTL_TCP の例外も補足します。

REQUEST および REQUEST_PIECES() では、なんらかの例外が発生し detailed_exception が無効な場合に request_failed 例外が発生します。

UTL_HTTP の例

次の例では、UTL_HTTP の使用方法を示します。

例 : UTL_HTTP の使用

```
SET serveroutput ON SIZE 40000

DECLARE
  req   utl_http.req;
  resp  utl_http.resp;
  value VARCHAR2(1024);
BEGIN

  utl_http.set_proxy('proxy.my-company.com', 'corp.my-company.com');

  req := utl_http.begin_request('http://www-hr.corp.my-company.com');
  utl_http.set_header(req, 'User-Agent', 'Mozilla/4.0');
  resp := utl_http.get_response(req);
  LOOP
    utl_http.read_line(resp, value, TRUE);
    dbms_output.put_line(value);
  END LOOP;
  utl_http.end_response(resp);
EXCEPTION
  WHEN utl_http.end_of_body THEN
    utl_http.end_response(resp);
END;
```

例 : HTTP 応答ヘッダーの取出し

```
SET serveroutput ON SIZE 40000

DECLARE
  req   utl_http.req;
  resp  utl_http.resp;
  name  VARCHAR2(256);
  value VARCHAR2(1024);
BEGIN

  utl_http.set_proxy('proxy.my-company.com', 'corp.my-company.com');

  req := utl_http.begin_request('http://www-hr.corp.my-company.com');
  utl_http.set_header(req, 'User-Agent', 'Mozilla/4.0');
  resp := utl_http.get_response(req);

  dbms_output.put_line('HTTP response status code: ' || resp.status_code);
  dbms_output.put_line('HTTP response reason phrase: ' || resp.reason_phrase);

  FOR i IN 1..utl_http.get_header_count(resp) LOOP
    utl_http.get_header(resp, i, name, value);
    dbms_output.put_line(name || ': ' || value);
  END LOOP;
  utl_http.end_response(resp);
END;
```

例 : HTTP 認証の処理

```
SET serveroutput ON SIZE 40000

CREATE OR REPLACE PROCEDURE get_page (url      IN VARCHAR2,
                                      username IN VARCHAR2 DEFAULT NULL,
                                      password IN VARCHAR2 DEFAULT NULL,
                                      realm    IN VARCHAR2 DEFAULT NULL) AS
  req       utl_http.req;
  resp      utl_http.resp;
  my_scheme VARCHAR2(256);
  my_realm  VARCHAR2(256);
  my_proxy  BOOLEAN;
BEGIN

  -- Turn off checking of status code. We will check it by ourselves.
  utl_http.http_response_error_check(FALSE);

  req := utl_http.begin_request(url);
  IF (username IS NOT NULL) THEN
    utl_http.set_authentication(req, username, password); -- Use HTTP Basic Authen.
```

```
Scheme
  END IF;

  resp := utl_http.get_response(req);
  IF (resp.status_code = utl_http.HTTP_UNAUTHORIZED) THEN
    utl_http.get_authentication(resp, my_scheme, my_realm, my_proxy);
    IF (my_proxy) THEN
       dbms_output.put_line('Web proxy server is protected.');
       dbms_output.put('Please supplied the required ' || my_scheme || '
authentication username/password for realm ' || my_realm || ' for the proxy
server.');
    ELSE
      dbms_output.put_line('Web page ' || url || ' is protected.');
      dbms_output.put('Please supplied the required ' || my_scheme || '
authentication username/password for realm ' || my_realm || ' for the Web page.');
    END IF;
    utl_http.end_response(resp);
    RETURN;
  END IF;

  FOR i IN 1..utl_http.get_header_count(resp) LOOP
    utl_http.get_header(resp, i, name, value);
    dbms_output.put_line(name || ': ' || value);
  END LOOP;
  utl_http.end_response(resp);

END;
```

例 : Cookie の取出しおよびリストア

```
CREATE TABLE my_cookies (
    session_id  BINARY_INTEGER,
    name        VARCHAR2(256),
    value       VARCHAR2(1024),
    domain      VARCHAR2(256),
    expire      DATE,
    path        VARCHAR2(1024),
    secure      VARCHAR2(1),
    version     BINARY_INTEGER
);

CREATE SEQUENCE session_id;

SET serveroutput ON SIZE 40000

REM Retrieve cookies from UTL_HTTP
```

```
CREATE OR REPLACE FUNCTION save_cookies RETURN BINARY_INTEGER AS
  cookies        utl_http.cookie_table;
  my_session_id  BINARY_INTEGER;
  secure         VARCHAR2(1);
BEGIN

  /* assume that some cookies have been set in previous HTTP requests. */

  utl_http.get_cookies(cookies);
  select session_id.nextval into my_session_id from dual;

  FOR i in 1..cookies.count LOOP
    IF (cookies(i).secure) THEN
      secure := 'Y';
    ELSE
      secure := 'N';
    END IF;
    insert into my_cookies
    value (my_session_id, cookies(i).name, cookies(i).value, cookies(i).domain,
           cookies(i).expire, cookies(i).path, secure, cookies(i).version);
  END LOOP;

  RETURN my_session_id;

END;

REM Retrieve cookies from UTL_HTTP

CREATE OR REPLACE PROCEDURE restore_cookies (this_session_id IN BINARY_INTEGER) AS
  cookies        utl_http.cookie_table;
  cookie         utl_http.cookie;
  i              PLS_INTEGER := 0;
  CORSOR c (c_session_id BINARY_INTEGER) IS
    SELECT * FROM my_cookies WHERE session_id = c_session_id;
BEGIN

  FOR r IN c(this_session_id) LOOP
    i := i + 1;
    cookie.name   := r.name;
    cookie.value  := r.value;
    cookie.domain := r.domain;
    cookie.expire := r.expire;
    cookie.path   := r.path;
    IF (r.secure = 'Y') THEN
      cookie.secure := TRUE;
    ELSE
      cookie.secure := FALSE;
```

```
    END IF;
    cookie.version := r.version;
    cookies(i) := cookie;
  END LOOP;

  utl_http.clear_cookies;
  utl_http.add_cookies(cookies);

END;
```

UTL_HTTP サブプログラムの要約

表 96-6 UTL_HTTP サブプログラムー単一コールでの単純な HTTP のフェッチ

サブプログラム	説明
「REQUEST ファンクション」96-22 ページ	指定した URL から取り出したデータの最初の 2000 バイトまでを戻します。このファンクションは、SQL 問合せで直接使用できます。
「REQUEST_PIECES ファンクション」96-25 ページ	指定した URL から取り出したデータについて、2000 バイト・ピースの PL/SQL 表を戻します。

表 96-7 UTL_HTTP サブプログラムーセッションの設定

サブプログラム	説明
「SET_PROXY プロシージャ」96-28 ページ	HTTP または他のプロトコルの要求で使用するプロキシを設定します。
「GET_PROXY プロシージャ」96-29 ページ	現在のプロキシ設定を取り出します。
「SET_COOKIE_SUPPORT プロシージャ」96-30 ページ	将来の HTTP 要求が HTTP Cookie をサポートするかどうかを設定し、現在のデータベース・ユーザー・セッションで維持される Cookie の最大数を設定します。
「GET_COOKIE_SUPPORT プロシージャ」96-31 ページ	現在の Cookie のサポート設定を取り出します。
「SET_FOLLOW_REDIRECT プロシージャ」96-32 ページ	`get_response` ファンクションでの将来の要求に対する HTTP 応答において、HTTP リダイレクトの指示に `UTL_HTTP` が従う最大回数を設定します。
「GET_FOLLOW_REDIRECT プロシージャ」96-33 ページ	現行のセッションでの follow_redirect の設定を取り出します。
「SET_BODY_CHARSET プロシージャ」96-33 ページ	メディア・タイプが `text` で、キャラクタ・セットが `Content-Type` ヘッダーで指定されていない場合、将来のすべての HTTP 要求本体におけるデフォルトのキャラクタ・セットを設定します。

表 96-7 UTL_HTTP サブプログラムーセッションの設定（続き）

サブプログラム	説明
「GET_BODY_CHARSET プロシージャ」96-34 ページ	将来の HTTP 要求すべての本体におけるデフォルトのキャラクタ・セットを取り出します。
「SET_PERSISTENT_CONN_SUPPORT プロシージャ」96-34 ページ	将来の HTTP 要求が HTTP 1.1 の永続的接続をサポートするかどうかを設定し、現在のデータベース・ユーザー・セッションで維持される永続的接続の最大回数を設定します。
「GET_PERSISTENT_CONN_SUPPORT プロシージャ」96-37 ページ	永続的接続のサポートが有効かどうかをチェックし、現行のセッションでの永続的接続の最大回数を取得します。
「SET_RESPONSE_ERROR_CHECK プロシージャ」96-37 ページ	Web サーバーがエラーを示すステータス・コード（4xx または 5xx）を戻した場合に、get_response が例外を呼び出すかどうかを設定します。
「GET_RESPONSE_ERROR_CHECK プロシージャ」96-38 ページ	応答エラー・チェックが設定されているかどうかをチェックします。
「SET_DETAILED_EXCP_SUPPORT プロシージャ」96-38 ページ	UTL_HTTP パッケージが詳細な例外を呼び出すように設定します。
「GET_DETAILED_EXCP_SUPPORT プロシージャ」96-39 ページ	UTL_HTTP パッケージが詳細な例外を呼び出すかどうかをチェックします。
「SET_WALLET プロシージャ」96-39 ページ	Secure Sockets Layer（SSL）、つまり、HTTPS を経由する HTTP 要求すべてに使用する Oracle Wallet を設定します。
「SET_TRANSFER_TIMEOUT プロシージャ」96-41 ページ	UTL_HTTP が Web サーバーまたはプロキシ・サーバーから HTTP 応答を読み込むタイムアウト値を設定します。
「GET_TRANSFER_TIMEOUT プロシージャ」96-41 ページ	現在のネットワーク転送のタイムアウト値を取り出します。

表 96-8 UTL_HTTP サブプログラムー HTTP 要求

サブプログラム	説明
「BEGIN_REQUEST ファンクション」96-42 ページ	新しい HTTP 要求を開始します。UTL_HTTP はターゲット Web サーバーまたはプロキシ・サーバーへのネットワーク接続を確立し、HTTP 要求行を送信します。
「SET_HEADER プロシージャ」96-43 ページ	HTTP 要求ヘッダーを設定します。要求ヘッダーは、設定後ただちに Web サーバーに送信されます。
「SET_AUTHENTICATION プロシージャ」96-44 ページ	HTTP 要求ヘッダーの HTTP 認証情報を設定します。Web サーバーが要求を認証するにはこの情報が必要です。

表 96-8 UTL_HTTP サブプログラム― HTTP 要求（続き）

サブプログラム	説明
「SET_COOKIE_SUPPPORT プロシージャ」96-45 ページ	要求の HTTP Cookie のサポートを有効または無効にします。
「SET_FOLLOW_REDIRECT プロシージャ」96-46 ページ	GET_RESPONSE ファンクションでこの要求に対する HTTP 応答において、HTTP リダイレクトの指示に UTL_HTTP が従う最大回数を設定します。
「SET_BODY_CHARSET プロシージャ」96-47 ページ	メディア・タイプが text で、キャラクタ・セットが Content-Type ヘッダーで指定されていない場合は、要求本体のデフォルトのキャラクタ・セットを設定します。
「SET_PERSISTENT_CONN_SUPPORT プロシージャ」96-47 ページ	要求の HTTP 1.1 の永続的接続のサポートを有効または無効にします。
「WRITE_TEXT プロシージャ」96-50 ページ	HTTP 要求本体にテキスト・データを書き込みます。
「WRITE_LINE プロシージャ」96-51 ページ	HTTP 要求本体にテキスト行を書き込み、改行文字（UTL_TCP で定義される CRLF）を使用して行を終了します。
「WRITE_RAW プロシージャ」96-52 ページ	HTTP 要求本体にバイナリ・データを書き込みます。
「END_REQUEST プロシージャ」96-53 ページ	HTTP 要求を終了します。

表 96-9 UTL_HTTP サブプログラム― HTTP 応答

サブプログラム	説明
「GET_RESPONSE ファンクション」96-54 ページ	HTTP 応答を読み込みます。ファンクションが戻ると、ステータス行および HTTP 応答ヘッダーが読み込まれて処理されます。
「GET_HEADER_COUNT ファンクション」96-54 ページ	応答で戻された HTTP 応答ヘッダーの数を戻します。
「GET_HEADER プロシージャ」96-55 ページ	応答で戻された n 回目の HTTP 応答ヘッダーの名前および値を戻します。
「GET_HEADER_BY_NAME プロシージャ」96-56 ページ	特定のヘッダー名を持つ応答に戻された HTTP 応答ヘッダーの値を戻します。
「GET_AUTHENTICATION プロシージャ」96-57 ページ	Web サーバーが受け入れる要求に必要な HTTP 認証情報を、HTTP 応答ヘッダーの指示に従って取り出します。

表 96-9 UTL_HTTP サブプログラム― HTTP 応答（続き）

サブプログラム	説明
「SET_BODY_CHARSET プロシージャ」96-58 ページ	メディア・タイプが text で、キャラクタ・セットが Content-Type ヘッダーで指定されていない場合は、応答本体のキャラクタ・セットを設定します。
「READ_TEXT プロシージャ」96-59 ページ	HTTP 応答本体をテキスト形式で読み込み、コール元が指定したバッファに出力を戻します。
「READ_LINE プロシージャ」96-60 ページ	HTTP 応答本体を行末までテキスト形式で読み込み、コール元が提供したバッファに出力を戻します。
「READ_RAW プロシージャ」96-62 ページ	HTTP 応答本体をバイナリ形式で読み込み、コール元が指定したバッファに出力を戻します。
「END_RESPONSE プロシージャ」96-63 ページ	HTTP 応答を終了します。HTTP 要求および応答を完了します。

表 96-10 UTL_HTTP サブプログラム― HTTP Cookie

サブプログラム	説明
「GET_COOKIE_COUNT ファンクション」96-63 ページ	すべての Web サーバーにより設定された UTL_HTTP パッケージが現在維持する Cookie の数を戻します。
「GET_COOKIES ファンクション」96-64 ページ	すべての Web サーバーにより設定された UTL_HTTP パッケージが現在維持するすべての Cookie を戻します。
「ADD_COOKIES プロシージャ」96-64 ページ	UTL_HTTP に維持される Cookie を追加します。
「CLEAR_COOKIES プロシージャ」96-64 ページ	UTL_HTTP パッケージに維持される Cookie をすべてクリアします。

表 96-11 UTL_HTTP サブプログラム― HTTP の永続的接続

サブプログラム	説明
「GET_PERSISTENT_CONN_COUNT ファンクション」96-65 ページ	UTL_HTTP パッケージにより現在永続的に維持されているネットワーク接続数を Web サーバーに戻します。
「GET_PERSISTENT_CONNS プロシージャ」96-65 ページ	UTL_HTTP パッケージにより現在永続的に維持されているネットワーク接続をすべて Web サーバーに戻します。
「CLOSE_PERSISTENT_CONN プロシージャ」96-66 ページ	現在のデータベース・セッションで UTL_HTTP パッケージが維持する HTTP の永続的接続をクローズします。
「CLOSE_PERSISTENT_CONNS プロシージャ」96-66 ページ	現在のデータベース・セッションで UTL_HTTP パッケージが維持する HTTP の永続的接続のグループをクローズします。

表 96-12 UTL_HTTP サブプログラム―エラー状態

サブプログラム	説明
「GET_DETAILED_SQLCODE ファンクション」96-68 ページ	最後に発生した例外の詳細な SQLCODE を取り出します。
「GET_DETAILED_SQLERRM ファンクション」96-68 ページ	最後に発生した例外の詳細な SQLERRM を取り出します。

単純な HTTP のフェッチ

REQUEST および REQUEST_PIECES は、Uniform Resource Locator（URL）の文字列を使用し、サイトに接続して、そのサイトで取得したデータ（通常は HTML）を戻します。

同一マシン（同一の権限や環境変数など）でブラウザを使用して URL に接続できない場合、REQUEST または REQUEST_PIECES で、その URL への接続が成功することは期待できません。

REQUEST または REQUEST_PIECES に失敗した場合（たとえば、例外が発生した場合や HTML 形式のエラー・メッセージが戻された場合で、URL 引数は正しいと考えられる場合）は、ブラウザで同じ URL に接続を試みて、ユーザーのマシンからネットワークの可用性を検証します。ユーザーのブラウザにプロキシ・サーバー・セットがある場合、そのセットは、オプションの proxy パラメータを使用して REQUEST または REQUEST_PIECES の各コールごとに設定する必要があります。

注意： UTL_HTTP では、環境変数を使用してそのプロキシ動作を指定することもできます。たとえば、UNIX では、環境変数 http_proxy を URL に設定することによって、そのサービスを HTTP 要求のプロキシ・サーバーとして使用するように指定します。また、環境変数 no_proxy をドメイン名に設定することによって、そのドメインの URL では HTTP プロキシ・サーバーを使用しないように指定します。UTL_HTTP パッケージが Oracle データベース・サーバーで実行される場合、データベース・インスタンスの開始時に設定された環境設定が使用されます。

REQUEST ファンクション

このファンクションは、指定した URL から取り出したデータの最初の 2000 バイトまでを戻します。このファンクションは、SQL 問合せで直接使用できます。

構文

```
UTL_HTTP.REQUEST (
   url              IN VARCHAR2,
   proxy            IN VARCHAR2 DEFAULT NULL),
   wallet_path      IN VARCHAR2 DEFAULT NULL,
   wallet_password  IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

プラグマ

```
pragma restrict_references (request, wnds, rnds, wnps, rnps);
```

パラメータ

表 96-13 REQUEST ファンクションのパラメータ

パラメータ	説明
url	URL。
proxy	（オプション）HTTP 要求時に使用するプロキシ・サーバーを指定します。プロキシ設定の完全な形式については、set_proxy を参照してください。

表 96-13 REQUEST ファンクションのパラメータ（続き）

パラメータ	説明
wallet_path	（オプション）クライアント側の Wallet を指定します。クライアント側の Wallet には、HTTPS 要求に必要な信頼されている認証局のリストが含まれています。wallet_path の形式は、PC では file:c:¥WINNT¥Profiles¥<username>¥WALLETS、UNIX では file:/home/<username>/wallets のようになります。 UTL_HTTP パッケージが Oracle データベース・サーバーで実行される場合は、データベース・サーバーから Wallet にアクセスします。したがって、Wallet のパスにデータベース・サーバーからアクセスできる必要があります。Oracle Wallet の設定方法の説明は、set_wallet を参照してください。HTTPS 以外の要求の場合は、Oracle Wallet は必要ありません。
wallet_password	（オプション）Wallet のオープンに必要なパスワードを指定します。

戻り値

戻りタイプは長さ 2000 以下の文字列で、引数 URL への HTTP 要求から戻された HTML 結果から最初の 2000 バイトまでが含まれます。

例外

```
INIT_FAILED
REQUEST_FAILED
```

使用上の注意

このファンクションに引数として渡された URL は、URL 仕様 RFC 2396 に従った不正文字（空白など）の検証は行われません。コール元は、UTL_URL パッケージを使用して、このような文字をエスケープする必要があります。URL で有効な文字のリストについては、パッケージのコメントを参照してください。URL は US-ASCII 文字のみで構成する必要があります。それ以外の文字の URL での使用は避けてください。

Oracle Wallet の使用については、set_wallet ファンクションのマニュアルを参照してください。Oracle Wallet は、HTTPS Web サーバーへのアクセスに必要です。

応答エラー・チェックがオンにされないかぎり、Web サーバーから 4xx または 5xx の応答を受信してもこのファンクションが例外を呼び出すことはありません。そのかわりに、Web サーバーからフォーマット・エラー・メッセージが戻されます。

```
<HTML>
<HEAD>
<TITLE>Error Message</TITLE>
</HEAD>
<BODY>
<H1>Fatal Error 500</H1>
Can't Access Document:  http://home.nothing.comm.
<P>
<B>Reason:</B> Can’t locate remote host:  home.nothing.comm.
<P>
<P><HR>
<ADDRESS><A HREF="http://www.w3.org">
CERN-HTTPD3.0A</A></ADDRESS>
</BODY>
</HTML>
```

例

```
SQLPLUS> SELECT utl_http.request('http://www.my-company.com/') FROM dual;
UTL_HTTP.REQUEST('HTTP://WWW.MY-COMPANY.COM/')
<html>
<head><title>My Company Home Page</title>
<!--changed Jan. 16, 19
1 row selected.
```

firewall の内側にいる場合は、proxy パラメータを含めます。たとえば、Oracle firewall 内には、www-proxy.my-company.com という名前のプロキシ・サーバーがある場合があります。

```
SQLPLUS> SELECT
utl_http.request('http://www.my-company.com', 'www-proxy.us.my-company.com') FROM
dual;
```

REQUEST_PIECES ファンクション

このファンクションは、指定した URL から取り出したデータについて、2000 バイト・ピースの PL/SQL 表を戻します。

構文

```
type html_pieces is table of varchar2(2000) index by binary_integer;

UTL_HTTP.REQUEST_PIECES (
   url             IN VARCHAR2,
   max_pieces      IN NATURAL DEFAULT 32767,
   proxy           IN VARCHAR2 DEFAULT NULL,
   wallet_path     IN VARCHAR2 DEFAULT NULL,
   wallet_password IN VARCHAR2 DEFAULT NULL)
RETURN html_pieces;
```

プラグマ

```
pragma restrict_references (request_pieces, wnds, rnds, wnps, rnps);
```

パラメータ

表 96-14 REQUEST_PIECES ファンクションのパラメータ

パラメータ	説明
url	URL。
max_pieces	（オプション）REQUEST_PIECES が戻すピースの最大数（長さは各 2000 文字、最後のピースはこれより短くなる場合があります）。指定する場合、引数は正の整数を指定します。
proxy	（オプション）HTTP 要求時に使用するプロキシ・サーバーを指定します。プロキシ設定の完全な形式については、set_proxy を参照してください。

表 96-14 REQUEST_PIECES ファンクションのパラメータ（続き）

パラメータ	説明
wallet_path	（オプション）クライアント側の Wallet を指定します。クライアント側の Wallet には、HTTPS 要求に必要な信頼されている認証局のリストが含まれています。wallet_path の書式は、'file:/<local-dir-for-client-side-wallet>' です。 wallet_path の形式は、PC では file:c:¥WINNT¥Profiles¥<username>¥WALLETS、UNIX では file:/home/<username>/wallets です。UTL_HTTP パッケージが Oracle データベース・サーバーで実行される場合は、データベース・サーバーから Wallet にアクセスします。したがって、Wallet のパスにデータベース・サーバーからアクセスできる必要があります。 Oracle Wallet の設定方法の説明は、set_wallet を参照してください。HTTPS 以外の要求の場合は、Oracle Wallet は必要ありません。
wallet_password	（オプション）Wallet のオープンに必要なパスワードを指定します。

戻り値

REQUEST_PIECES は、UTL_HTTP.HTML_PIECES タイプの PL/SQL 表を戻します。PL/SQL 表の各要素は、最大長 2,000 の文字列です。REQUEST_PIECES により戻される PL/SQL 表の要素は、その URL に対して HTTP 要求から取得した連続データです。

例外

```
INIT_FAILED
REQUEST_FAILED
```

使用上の注意

このファンクションに引数として渡された URL は、URL 仕様 RFC 2396 に従った不正文字（空白など）の検証は行われません。コール元は、UTL_URL パッケージを使用して、このような文字をエスケープする必要があります。URL で有効な文字のリストについては、パッケージのコメントを参照してください。URL は US-ASCII 文字のみで構成する必要があります。それ以外の文字の URL での使用は避けてください。

このファンクションにより戻される PL/SQL 表の各エントリ（情報の断片）は、容量を完全に満たさない場合があります。このファンクションでは、前の断片が完全に満たされる前に、次の断片にデータが到着し始めます。

Oracle Wallet の使用については、set_wallet ファンクションのマニュアルを参照してください。Oracle Wallet は、HTTPS Web サーバーへのアクセスに必要です。

応答エラー・チェックがオンにされないかぎり、Web サーバーから 4xx または 5xx の応答を受信してもこのファンクションが例外を呼び出すことはありません。そのかわりに、Web サーバーからフォーマット・エラー・メッセージが戻されます。

```
<HTML>
<HEAD>
<TITLE>Error Message</TITLE>
</HEAD>
<BODY>
<H1>Fatal Error 500</H1>
Can't Access Document:  http://home.nothing.comm.
<P>
<B>Reason:</B> Can’t locate remote host:  home.nothing.comm.
<P>
<P><HR>
<ADDRESS><A HREF="http://www.w3.org">
CERN-HTTPD3.0A</A></ADDRESS>
</BODY>
</HTML>
```

例

```
SET SERVEROUTPUT ON


DECLARE
   x   utl_http.html_pieces;
   len PLS_INTEGER;
BEGIN
   x := utl_http.request_pieces('http://www.oracle.com/', 100);
   dbms_output.put_line(x.count || ' pieces were retrieved.');
   dbms_output.put_line('with total length ');
   IF x.count < 1 THEN
      dbms_output.put_line('0');
  ELSE
   len := 0;
   FOR i in 1..x.count LOOP
      len := len + length(x(i));
   END LOOP;
   dbms_output.put_line(i);
  END IF;
END;
/
-- Output
Statement processed.
```

```
4 pieces were retrieved.
with total length
7687
```

セッションの設定

HTTP 要求がデータベースのユーザー・セッション内で実行される場合、セッションの設定により UTL_HTTP の構成およびデフォルトの動作を操作します。要求が作成されると、現行のセッションの HTTP Cookie サポート、リダイレクトへの準拠、本体のキャラクタ・セット、永続的接続のサポートおよび転送タイムアウトのデフォルト設定が継承されます。これらの設定は、要求 API をコールすることにより後で変更できます。要求に対する応答が作成されると、この要求からこれらの設定が継承されます。後で、要求 API をコールすることにより変更できるのは、本体のキャラクタ・セットのみです。

SET_PROXY プロシージャ

このプロシージャは、HTTP プロトコルまたはその他のプロトコルの要求に使用するプロキシを設定します。no_proxy_domains で指定されたドメインに属するホストへの要求は除外されます。プロキシには、プロキシ・サーバーがリスニングするオプションの TCP/IP ポート番号を含めることができます。構文は、[http://]host[:port][/]（www-proxy.my-company.com:80 など）です。プロキシのポートを指定していない場合は、ポート 80 とみなされます。no_proxy_domains は、プロキシ・サーバーを経由せずに、宛先の HTTP サーバーに HTTP 要求が直接送信されるドメインまたはホストをカンマ、セミコロンまたは空白で区切られたリストです。オプションで、ドメインまたはホストごとにポート番号を指定できます。ポート番号が指定されていると、非プロキシの制限が適用されるのは、特定のドメインまたはホスト（corp.my-company.com、eng.my-company.com:80 など）のポートでの要求のみです。no_proxy_domains の値が NULL でプロキシが設定されている場合、要求はすべてプロキシを経由して送信されます。プロキシが設定されていない場合、UTL_HTTP により要求がターゲット Web サーバーに直接送信されます。

構文

```
UTL_HTTP.set_proxy (
   proxy             IN VARCHAR2,
   no_proxy_domains  IN VARCHAR2);
```

パラメータ

表 96-15 SET_PROXY プロシージャのパラメータ

パラメータ	説明
proxy (IN)	UTL_HTTP パッケージにより使用されるプロキシ（ホストおよびオプションのポート番号）。
no_proxy_domains (IN)	プロキシをすべての要求に対して使用しないホストおよびドメインのリスト。

使用上の注意

データベース・サーバー・インスタンスの開始時にプロキシが設定されている場合は、環境変数 http_proxy および no_proxy のプロキシ設定が想定されます。このプロシージャにより設定されたプロキシ設定は、初期の設定より優先されます。

GET_PROXY プロシージャ

このプロシージャは、現在のプロキシ設定を取り出します。

構文

```
UTL_HTTP.get_proxy (
   proxy            OUT NOCOPY VARCHAR2,
   no_proxy_domains  OUT NOCOPY VARCHAR2);
```

パラメータ

表 96-16 GET_PROXY プロシージャのパラメータ

パラメータ	説明
proxy (OUT)	UTL_HTTP パッケージにより現在使用されているプロキシ（ホストおよびオプションのポート番号）。
no_proxy_domains (OUT)	プロキシをすべての要求に対して使用しないホストおよびドメインのリスト。

SET_COOKIE_SUPPORT プロシージャ

このプロシージャは次の項目を設定します。

- 将来の HTTP 要求が HTTP Cookie をサポートするかどうか。
- 現在のデータベース・ユーザー・セッションで維持される Cookie の最大数。

Cookie が HTTP 要求に対して有効である場合、現行のセッションで保存され、要求に適用可能な Cookie はすべて、HTTP Cookie の仕様標準に基づき、その要求で Web サーバーに戻されます。Cookie のサポートが要求に対して有効である場合は、要求に応答して設定された Cookie が現行のセッションに保存され、後続の要求で Web サーバーに戻されます。Cookie のサポートが HTTP 要求に対して無効である場合は、その要求で Web サーバーに Cookie は戻されず、要求に応答して設定された Cookie は現行のセッションで保存されません。`Set-Cookie` HTTP ヘッダーを応答から取り出すことは可能です。

データベース・ユーザー・セッションでは、すべての HTTP 要求に対して Cookie のサポートがデフォルトで有効にされています。Cookie のサポートのデフォルト設定（有効または無効）が影響するのは将来の要求に対してのみであり、既存の要求には影響ありません。要求が作成された後、要求を操作するその他の `set_cookie_support` プロシージャを使用して、Cookie のサポートの設定を変更できます。

現行のセッションにデフォルトで保存される Cookie の最大数は、各サイトにつき 20、合計で 300 です。

構文

```
UTL_HTTP.set_cookie_support (
   enable       IN BOOLEAN,
   max_cookies  IN PLS_INTEGER DEFAULT 300,
   max_cookies_per_site  IN PLS_INTEGER DEFAULT 20);
```

パラメータ

表 96-17 SET_COOKIE SUPPORT プロシージャのパラメータ

パラメータ	説明
`enable (IN)`	将来の HTTP 要求が HTTP Cookie をサポートするかどうかを設定します（`TRUE` の場合はサポートし、`FALSE` の場合はサポートしません）。
`max_cookies (IN)`	現行のセッションで維持される Cookie の最大合計数を設定。
`max_cookies_per_site (IN)`	現行のセッションで Web サイト当たりで維持される Cookie の最大合計数を設定します。

使用上の注意

Cookie の最大合計数または各 Web サイトに対する Cookie の最大数を減少させる場合は、減少後の最大数まで最も古い Cookie からパージされます。現行のセッションで保存された HTTP Cookie はデータベース・セッションの期間のみ存続します。永続的に格納される Cookie はありません。Cookie のサポートを無効にすると、現行のセッションで保存された Cookie はクリアされません。

get_cookies および add_cookies を使用した Cookie の取出し、保存およびリストアを行う方法については、96-13 ページの「UTL_HTTP の例」を参照してください。

GET_COOKIE_SUPPORT プロシージャ

このプロシージャは、現在の Cookie サポート設定を取り出します。

構文

```
UTL_HTTP.get_cookie_support (
   enable               OUT BOOLEAN,
   max_cookies          OUT PLS_INTEGER,
   max_cookies_per_site  OUT PLS_INTEGER);
```

パラメータ

表 96-18 GET_COOKIE SUPPORT プロシージャのパラメータ

パラメータ	説明
enable (OUT)	将来の HTTP 要求が HTTP Cookie をサポートするかどうかを設定します（TRUE の場合はサポートし、FALSE の場合はサポートしません）。
max_cookies (OUT)	現在のデータベース・ユーザー・セッションで維持される Cookie の最大合計数を示します。
max_cookies_per_site (OUT)	現行のセッションで各 Web サイトについて維持される Cookie の最大合計数を示します。

SET_FOLLOW_REDIRECT プロシージャ

このプロシージャは、get_response ファンクションでの将来の要求に対する HTTP 応答において、HTTP リダイレクトの指示に UTL_HTTP が従う最大回数を設定します。

max_redirects が正の値に設定されている場合、get_response は、HTTP 要求のステータス・コード 301、302 および 307（HTTP HEAD メソッドおよび HTTP GET メソッド）、303（すべての HTTP メソッド）に対してリダイレクトされた URL に自動的に従います。さらに HTTP 要求（要求メソッドはステータス・コード 303 の HTTP GET に変更されます）が新しい場所で再試行されます。リダイレクトされていない最後の場所に到達するか、エラーが発生するか、または無限のループを回避するために設定されたリダイレクトの最大回数に到達するまでリダイレクトが実行されます。REQ レコードの URL および method フィールドは、最後にリダイレクトされた URL およびその URL へのアクセスに使用されたメソッドに更新されます。リダイレクトの最大回数を 0（ゼロ）に設定し、自動リダイレクトを無効にします。

データベース・ユーザー・セッションにおけるリダイレクトのデフォルト最大回数は 3 です。デフォルト値が影響を与えるのは将来の要求に対してのみであり、既存の要求に影響はありません。

要求が作成された後、要求を操作するその他の set_follow_redirect プロシージャを使用して、リダイレクトの最大回数を変更できます。

構文

```
UTL_HTTP.set_follow_redirect (
   max_redirects  IN PLS_INTEGER DEFAULT 3);
```

パラメータ

表 96-19 SET_FOLLOW_REDIRECT プロシージャのパラメータ

パラメータ	説明
max_redirects (IN)	リダイレクトの最大回数。リダイレクトを無効にするには 0（ゼロ）に設定します。

使用上の注意

現行のセッションでは自動リダイレクトに従わないように設定していますが、個別に HTTP 要求を指定して follow_redirect ファンクションのリダイレクト指示に従うように設定できます。その逆の設定も可能です。

GET_FOLLOW_REDIRECT プロシージャ

このプロシージャは、現行のセッションでの follow_redirect 設定を取り出します。

構文

```
UTL_HTTP.get_follow_redirect (
   max_redirects  OUT PLS_INTEGER);
```

パラメータ

表 96-20 GET_FOLLOW_REDIRECT プロシージャのパラメータ

パラメータ	説明
max_redirects (OUT)	将来の HTTP 要求すべてに対するリダイレクトの最大回数。

SET_BODY_CHARSET プロシージャ

このプロシージャは、メディア・タイプが text で、キャラクタ・セットが Content-Type ヘッダーで指定されていない場合、将来の HTTP 要求すべての本体におけるデフォルトのキャラクタ・セットを設定します。HTTP プロトコルの標準仕様に準拠していても、要求または応答のメディア・タイプが text で、キャラクタ・セット情報が Content-Type ヘッダーに含まれていない場合は、要求または応答の本体のキャラクタ・セットをデフォルトの ISO-8859-1 に設定する必要があります。要求に対して作成された応答は、現行のセッションの本体キャラクタ・セットではなく、要求のデフォルトの本体キャラクタ・セットを継承します。

データベース・ユーザー・セッションでは、デフォルトの本体キャラクタ・セットは ISO-8859-1 です。デフォルトの本体キャラクタ・セットの設定は将来の要求に対してのみ影響を与え、既存の要求に影響はありません。

要求が作成された後、要求を操作するその他の set_body_charset プロシージャを使用して、本体キャラクタ・セットを変更できます。

構文

```
UTL_HTTP.set_body_charset (
   charset  IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 96-21 SET_BODY_CHARSET プロシージャのパラメータ

パラメータ	説明
charset（IN）	要求本体のデフォルトのキャラクタ・セット。キャラクタ・セットは、Oracle または Internet Assigned Numbers Authority（IANA）のネーミング規則で命名できます。NULL の場合、データベース・キャラクタ・セットとみなされます。

GET_BODY_CHARSET プロシージャ

このプロシージャは、将来の HTTP 要求すべての本体におけるデフォルトのキャラクタ・セットを取り出します。

構文

```
UTL_HTTP.get_body_charset (
   charset  OUT NOCOPY VARCHAR2);
```

パラメータ

表 96-22 GET_BODY_CHARSET プロシージャのパラメータ

パラメータ	説明
charset （OUT）	将来の HTTP 要求すべての本体におけるデフォルトのキャラクタ・セット。

SET_PERSISTENT_CONN_SUPPORT プロシージャ

このプロシージャは次の項目を設定します。

- 将来の HTTP 要求が HTTP 1.1 の永続的接続をサポートするかどうか。
- 現在のデータベース・ユーザー・セッションで維持される永続的接続の最大数。

永続的接続が HTTP 要求に対して有効である場合、要求が完了した後、パッケージでオープン Web サーバーまたはプロキシ・サーバーへのネットワーク接続が維持されます。同じサーバーへの後続の要求で HTTP 1.1 の永続的接続を使用できます。永続的接続がサポートされている場合、ネットワーク接続の待機時間が回避されるため、後続の HTTP 要求は高速に処理されます。永続的接続が HTTP 要求に対して無効である場合、要求が完了した後、HTTP ヘッダー Connection: close が HTTP 要求に自動的に送信され、ネットワーク接続がクローズされます。この設定は、HTTP 1.0 プロトコルに準拠した HTTP 要求には影響ありません。これは、要求が完了した後、ネットワーク接続が常にクローズされるためです。

要求が作成されると、ターゲット Web サーバー（プロキシ・サーバー）が使用可能であれば、パッケージは常に既存の永続的接続を再利用しようとします。使用可能なサーバーがない場合、新しいネットワーク接続が開始されます。要求に対する永続的接続サポートの設定が影響するのは、要求が完了した後にネットワーク接続をクローズするかどうかということについてのみです。

データベース・ユーザー・セッションでは、すべての HTTP 要求に対して永続的接続がデフォルトで無効にされています。現行のセッションで保存される永続的接続のデフォルト最大数は 0（ゼロ）です。永続的接続サポートのデフォルト設定（有効または無効）が影響するのは将来の要求に対してのみであり、既存の要求に影響はありません。

要求が作成された後、要求を操作する他の set_persistent_conn_support プロシージャを使用して、永続的接続サポートの設定を変更できます。

UTL_HTTP で永続的接続を使用すると、同じサーバーから複数の Web ページをフェッチする回数を軽減する一方で、データベース・サーバーのシステム・リソース（ネットワーク接続）が浪費されます。データベース・サーバーで多くのネットワーク接続がオープンされたままである場合、永続的接続を過度に使用すると、データベース・サーバーのスケーラビリティが低下します。ネットワーク接続は、後続の要求により即時使用される場合にのみオープンにし、不要な場合にはクローズしてください。通常、セッションでは永続的接続サポートを無効にし、個々の HTTP 要求で永続的接続を有効にします。詳細は、96-36 ページの「例 : SET_PERSISTENT_CONN_SUPPORT の使用」を参照してください。

構文

```
UTL_HTTP.set_persistent_conn_support (
   enable     IN BOOLEAN,
   max_conns  IN PLS_INTEGER DEFAULT 0);
```

パラメータ

表 96-23 SET_PERSISTENT_CONN_SUPPORT プロシージャのパラメータ

パラメータ	説明
enable (IN)	永続的接続サポートを有効（TRUE に設定）または無効（FALSE に設定）にします。
max_conns (IN)	現行のセッションで維持される永続的接続の最大数を設定します。

使用上の注意

データベース・セッションで保存される永続的接続のデフォルト値は 0（ゼロ）です。永続的接続を実際に有効にするには、永続的接続の最大数を正の値に設定する必要があります。そうでない場合には、接続は永続的に保持されません。

例 : SET_PERSISTENT_CONN_SUPPORT の使用

```
DECLARE
  TYPE vc2_table IS TABLE OF VARCHAR2(256) INDEX BY binary_integer;
  paths vc2_table;

  PROCEDURE fetch_pages(paths IN vc2_table) AS
    url_prefix VARCHAR2(256) := 'http://www.my-company.com/';
    req   utl_http.req;
    resp  utl_http.resp;
    data  VARCHAR2(1024);
  BEGIN
    FOR i IN 1..paths.count LOOP
      req := utl_http.begin_request(url_prefix || paths(i));

      -- Use persistent connection except for the last request
      IF (i < paths.count) THEN
        utl_http.set_persistent_conn_support(req, TRUE);
      END IF;

      resp := utl_http.get_response(req);

      BEGIN
        LOOP
          utl_http.read_text(resp, data);
          -- do something with the data
        END LOOP;
      EXCEPTION
        WHEN utl_http.end_of_body THEN
          NULL;
      END;
      utl_http.end_response(resp);
    END LOOP;
  END;

BEGIN
  utl_http.set_persistent_conn_support(FALSE, 1);
  paths(1) := '...';
  paths(2) := '...';
  ...
  fetch_pages(paths);
END;
```

GET_PERSISTENT_CONN_SUPPORT プロシージャ

このプロシージャは次の項目をチェックします。

- 永続的接続サポートが有効にされているかどうか。
- 現行のセッションでの永続的接続の最大数の取得。

構文

```
UTL_HTTP.get_persistent_conn_support (
   enable     OUT BOOLEAN,
   max_conns  OUT PLS_INTEGER);
```

パラメータ

表 96-24 GET_PERSISTENT_CONN_SUPPORT プロシージャのパラメータ

パラメータ	説明
enable (OUT)	永続的接続サポートが有効にされている場合は TRUE、そうでない場合は FALSE です。
max_conns (OUT)	現行のセッションで維持される永続的接続の最大数。

SET_RESPONSE_ERROR_CHECK プロシージャ

このプロシージャは、Web サーバーがエラーを示すステータス・コード（4xx または 5xx）を戻した場合に、get_response が例外を呼び出すかどうかを設定します。たとえば、要求した URL が宛先の Web サーバーで見つからない場合、404（ドキュメントが見つからない）の応答ステータス・コードが戻されます。ステータス・コードがエラーを示す 4xx または 5xx で、このプロシージャが有効である場合は、get_response が HTTP_CLIENT_ERROR または HTTP_SERVER_ERROR 例外を呼び出します。SET_RESPONSE_ERROR_CHECK が FALSE に設定されている場合は、ステータス・コードがエラーを示しても get_response は例外を呼び出しません。デフォルトでは、応答エラー・チェックはオフになっています。

構文

```
UTL_HTTP.set_response_error_check (
   enable  IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 96-25 SET_RESPONSE_ERROR_CHECK プロシージャのパラメータ

パラメータ	説明
enable (IN)	応答エラーをチェックする場合は TRUE、そうでない場合は FALSE です。

使用上の注意

SET_RESPONSE_ERROR_CHECK を FALSE に設定すると、get_response ファンクションがその他の例外を呼び出す可能性があります。

GET_RESPONSE_ERROR_CHECK プロシージャ

このプロシージャは、応答エラー・チェックが設定されているかどうかをチェックします。

構文

```
UTL_HTTP.get_response_error_check (
   enable  OUT BOOLEAN);
```

パラメータ

表 96-26 GET_RESPONSE_ERROR_CHECK プロシージャのパラメータ

パラメータ	説明
enable (OUT)	応答エラー・チェックが設定されている場合は TRUE、そうでない場合は FALSE です。

SET_DETAILED_EXCP_SUPPORT プロシージャ

このプロシージャは、UTL_HTTP パッケージが詳細な例外を呼び出すように設定します。デフォルトでは、UTL_HTTP が要求の実行に失敗すると request_failed 例外が発生します。エラーの詳細情報を参照するには、GET_DETAILED_SQLCODE および GET_DETAILED_SQLEERM を使用します。

構文

```
UTL_HTTP.set_detailed_excp_support (
   enable  IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 96-27 SET_DETAILED_EXCP_SUPPORT プロシージャのパラメータ

パラメータ	説明
enable (IN)	UTL_HTTP に詳細な例外を直接発生させるように指示する場合は TRUE、そうでない場合は FALSE に設定します。

GET_DETAILED_EXCP_SUPPORT プロシージャ

このプロシージャは、UTL_HTTP パッケージが詳細な例外を呼び出すかどうかをチェックします。

構文

```
UTL_HTTP.get_detailed_excp_support (
   enable  OUT BOOLEAN);
```

パラメータ

表 96-28 GET_DETAILED_EXCP_SUPPORT プロシージャのパラメータ

パラメータ	説明
enable (OUT)	UTL_HTTP に詳細な例外を発生させる場合は TRUE、そうでない場合は FALSE に設定します。

SET_WALLET プロシージャ

このプロシージャは、Secure Sockets Layer（SSL）、つまり HTTPS を経由する HTTP 要求すべてに使用する Oracle Wallet を設定します。UTL_HTTP パッケージが SSL を介して HTTP サーバーと通信する場合、HTTP サーバーは認証局が署名したデジタル証明書を UTL_HTTP パッケージに示し、身分を証明します。Oracle Wallet には、UTL_HTTP パッケージのユーザーが信頼する認証局のリストが含まれています。Oracle Wallet は、HTTPS 要求を作成する必要があります。

Oracle Wallet を設定するには、Oracle Wallet Manager を使用して Wallet を作成します。HTTPS 要求を成功させるには、リモート HTTPS Web サーバーの証明書に署名した認証局が Wallet に設定されたトラスト・ポイントであることが必要です。Wallet が作成されると、トラスト・ポイントとして認識された認証局に移入されます。リモート HTTPS Web サーバーの証明書に署名した認証局がトラスト・ポイントにない場合、または新しいルートの証明書を持っている場合は、認証局のルート証明書を取得し、Oracle Wallet Manager を使用して Wallet のトラスト・ポイントとしてインストールする必要があります。

関連項目： Wallet Manager の詳細は、『Oracle Advanced Security 管理者ガイド』を参照してください。

構文

```
UTL_HTTP.set_wallet (
   path      IN VARCHAR2,
   password  IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 96-29 SET_WALLET プロシージャのパラメータ

パラメータ	説明
path (IN)	Oracle Wallet を含むディレクトリ・パス。形式は file:<directory-path> です。 wallet_path の形式は、PC では file:c:¥WINNT¥Profiles¥<username>¥WALLETS、UNIX では file:/home/<username>/wallets です。UTL_HTTP パッケージが Oracle データベース・サーバーで実行される場合は、データベース・サーバーから Wallet にアクセスします。したがって、Wallet のパスにデータベース・サーバーからアクセスできる必要があります。
password (IN)	Wallet のオープンに必要なパスワード。パスワードなしの場合、Wallet ディレクトリ内の Wallet の 2 番目のコピーでオープンする場合があります。この 2 番目のコピーは読取り専用です。password が NULL の場合、UTL_HTTP パッケージが Wallet の 2 番目の読取り専用コピーをオープンします。

SET_TRANSFER_TIMEOUT プロシージャ

将来の HTTP 要求すべてにおいて、UTL_HTTP パッケージが試行する、Web サーバーまたはプロキシ・サーバーから HTTP 応答を読み込むまでのデフォルトのタイムアウト値を設定します。このタイムアウト値を使用して、Web サーバーから Web ページを取り出す間、ビジー状態の Web サーバーや通信量の多いネットワークにより PL/SQL プログラムがブロックされることを回避します。タイムアウトのデフォルト値は 60 秒です。

構文

```
UTL_HTTP.set_transfer_timeout (
   timeout  IN PLS_INTEGER DEFAULT 60);
```

パラメータ

表 96-30 SET_TRANSFER_TIMEOUT プロシージャのパラメータ

パラメータ	説明
TIMEOUT （IN）	ネットワーク転送のタイムアウト値（秒）。

GET_TRANSFER_TIMEOUT プロシージャ

このプロシージャは、将来の HTTP 要求に対するデフォルトのタイムアウト値を取り出します。

構文

```
UTL_HTTP.get_transfer_timeout (
   timeout  OUT PLS_INTEGER);
```

パラメータ

表 96-31 GET_TRANSFER_TIMEOUT プロシージャのパラメータ

パラメータ	説明
TIMEOUT （OUT）	ネットワーク転送のタイムアウト値（秒）。

HTTP 要求

次の API は HTTP 要求を開始し、属性を操作し、要求情報を Web サーバーに送信します。要求が作成されると、現行のセッションの HTTP Cookie サポート、リダイレクトへの準拠、本体のキャラクタ・セット、永続的接続のサポートおよび転送タイムアウトのデフォルト設定が継承されます。これらの設定は、要求 API をコールすると変更できます。

BEGIN_REQUEST ファンクション

このファンクションは、新しい HTTP 要求を開始します。UTL_HTTP はターゲット Web サーバーまたはプロキシ・サーバーへのネットワーク接続を確立し、HTTP 要求行を送信します。PL/SQL プログラムは、他の API をコールして要求を継続し、完了します。

構文

```
UTL_HTTP.begin_request (
   url           IN VARCHAR2,
   method        IN VARCHAR2 DEFAULT 'GET',
   http_version  IN VARCHAR2 DEFAULT NULL)
RETURN req;
```

パラメータ

表 96-32 BEGIN_REQUEST ファンクションのパラメータ

パラメータ	説明
url (IN)	HTTP 要求の URL。
method (IN)	URL により識別されたリソースで実行されるメソッド。
http_version (IN)	要求の送信に使用される HTTP プロトコルのバージョン。プロトコル・バージョンの形式は HTTP/major-version.minor-version です。major-version および minor-version は正の値です。このパラメータが NULL の場合、UTL_HTTP は、サポートされている HTTP プロトコルの最新バージョンを使用して要求を送信します。パッケージがサポートする最新バージョンは 1.1 ですが、次のバージョンにアップグレード可能です。デフォルトは NULL です。

使用上の注意

このファンクションに引数として渡された URL については、URL 仕様 RFC 2396 による不正な文字（空白など）の検証は行われません。コールを実行する場合は、UTL_URL パッケージを使用してこのような文字をエスケープする必要があります。URL は US-ASCII 文字のみで構成する必要があります。URL で有効な文字のリストは、第 102 章「UTL_URL」を参照してください。URL は US-ASCII 文字のみで構成する必要があることに注意してください。それ以外の文字の URL での使用は避けてください。

HTTPS を介して Web サーバーにアクセスするには、Oracle Wallet を設定する必要があります。Oracle Wallet の設定方法は、set_wallet プロシージャを参照してください。

SET_HEADER プロシージャ

このプロシージャは、HTTP 要求ヘッダーを設定します。要求ヘッダーは、設定後ただちに Web サーバーに送信されます。

構文

```
UTL_HTTP.set_header (
   r      IN OUT NOCOPY req,
   name   IN VARCHAR2,
   value  IN VARCHAR2);
```

パラメータ

表 96-33 SET_HEADER プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 要求。
name (IN)	HTTP 要求ヘッダーの名前。
value (IN)	HTTP 要求ヘッダーの値。

使用上の注意

HTTP プロトコルの標準では、複数の HTTP ヘッダーに同じ名前を付けることができます。したがって、既存のヘッダーと同じ名前を付けても置換されません。

HTTP 1.1 を使用して要求を作成する場合、UTL_HTTP によりホストのヘッダーが自動的に設定されます。

このプロシージャを使用して Content-Type ヘッダーを設定すると、UTL_HTTP はヘッダー値からキャラクタ・セット情報を探します。キャラクタ・セット情報が存在する場合は、要求本体のキャラクタ・セットとして設定されます。set_body_charset プロシージャを使用すると、後でこの設定を変更できます。

chunked の値を使用して Transfer-Encoding ヘッダーを設定すると、UTL_HTTP が write_text、write_line および write_raw プロシージャにより書き込まれた要求本体を自動的にエンコードします。HTTP 1.1 ベースの Web サーバーまたは CGI プログラムは、HTTP 1.1 chunked transfer-encoding 形式でエンコードされた要求本体をサポートしていません。

SET_AUTHENTICATION プロシージャ

このプロシージャは、HTTP 要求ヘッダーの HTTP 認証情報を設定します。Web サーバーが要求を認証するにはこの情報が必要です。

構文

```
UTL_HTTP.set_authentication(
   r  IN OUT NOCOPY req,
   username  IN VARCHAR2,
   password  IN VARCHAR2,
   scheme    IN VARCHAR2 DEFAULT 'Basic',
   for_proxy IN BOOLEAN  DEFAULT FALSE);
```

パラメータ

表 96-34 SET_AUTHENTICATION プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 要求。
username (IN)	HTTP 認証のユーザー名。
password (IN)	HTTP 認証のパスワード。
scheme (IN)	HTTP 認証方式。デフォルトの BASIC は、HTTP 基本認証方式を示します。
for_proxy (IN)	HTTP 認証情報が Web サーバーではなく HTTP プロキシ・サーバーにアクセスするためのものであるかどうかを識別します。デフォルトは FALSE です。

使用上の注意

HTTP 基本認証方式のみがサポートされています。

SET_COOKIE_SUPPPORT プロシージャ

このプロシージャは、要求の HTTP Cookie のサポートを有効または無効にします。Cookie が HTTP 要求に対して有効である場合、現行のセッションで保存され、要求に適用可能な Cookie はすべて、HTTP Cookie の仕様標準に基づき、その要求で Web サーバーに戻されます。Cookie サポートが要求に対して有効である場合は、要求に応答して設定された Cookie が現行のセッションに保存され、後続の要求で Web サーバーに戻されます。Cookie サポートが HTTP 要求に対して無効である場合は、その要求で Web サーバーに Cookie は戻されず、要求に応答して設定された Cookie は現行のセッションで保存されません。ただし、`Set-Cookie` HTTP ヘッダーは応答からの取出しが可能です。

このプロシージャを使用して、要求がセッションのデフォルト設定を継承する Cookie のサポート設定を変更します。

構文

```
UTL_HTTP.set_cookie_support(
   r       IN OUT NOCOPY req,
   enable  IN BOOLEAN DEFAULT TRUE);
```

パラメータ

表 96-35 SET_COOKIE_SUPPORT プロシージャのパラメータ

パラメータ	説明
`r (IN/OUT)`	HTTP 要求。
`enable (IN)`	HTTP Cookie サポートを有効にする場合は `TRUE`、無効に設定する場合は `FALSE` に設定します。

使用上の注意

現行のセッションで保存された HTTP Cookie はデータベース・セッションの期間にのみ存続します。永続的に格納される Cookie はありません。`get_cookies` および `add_cookies` を使用した Cookie の取出し、保存およびリストアを行う方法については、96-13 ページの「UTL_HTTP の例」を参照してください。

SET_FOLLOW_REDIRECT プロシージャ

このプロシージャは、GET_RESPONSE ファンクションでの将来の要求に対する HTTP 応答において、HTTP リダイレクトの指示に UTL_HTTP が従う最大回数を設定します。

max_redirects が正の値に設定されている場合、GET_RESPONSE は、HTTP 要求のステータス・コード 301、302 および 307（HTTP HEAD メソッドおよび HTTP GET メソッド）、303（すべての HTTP メソッド）に対してリダイレクトされた URL に自動的に従います。さらに HTTP 要求（要求メソッドはステータス・コード 303 の HTTP GET に変更されます）が新しい場所で再試行されます。リダイレクトされていない最後の場所に到達するか、エラーが発生するか、または無限のループを回避するために設定されたリダイレクトの最大回数に到達するまでリダイレクトが実行されます。REQ レコードの url および method フィールドは、最後にリダイレクトされた URL およびその URL へのアクセスに使用されたメソッドに更新されます。リダイレクトの最大回数を 0（ゼロ）に設定し、自動リダイレクトを無効にします。

このプロシージャを使用して、要求がセッションのデフォルト設定を継承するリダイレクトの最大数を変更します。

構文

```
UTL_HTTP.set_follow_redirect(
   r              IN OUT NOCOPY req,
   max_redirects  IN PLS_INTEGER DEFAULT 3);
```

パラメータ

表 96-36 SET_FOLLOW_REDIRECT プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 要求。
max_redirects (IN)	リダイレクトの最大回数。リダイレクトを無効にするには 0（ゼロ）に設定します。

使用上の注意

リダイレクトを有効にするには、GET_RESPONSE の前に SET_FOLLOW_REDIRECT プロシージャをコールする必要があります。

SET_BODY_CHARSET プロシージャ

このプロシージャは、メディア・タイプが text で、キャラクタ・セットが Content-Type ヘッダーで指定されていない場合は、要求のキャラクタ・セットを設定します。要求または応答のメディア・タイプが text で、キャラクタ・セット情報が Content-Type ヘッダーにない場合は、HTTP プロトコルの標準仕様に従って、要求または応答の本体のキャラクタ・セットのデフォルトに ISO-8859-1 を設定する必要があります。

このプロシージャを使用して、要求がセッションのデフォルト設定を継承するデフォルトの本体キャラクタ・セットを変更します。

構文

```
UTL_HTTP.set_body_charset(
   r        IN OUT NOCOPY req,
   charset  IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 96-37 SET_BODY_CHARSET プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 要求。
charset (IN)	要求本体のデフォルトのキャラクタ・セット。キャラクタ・セットは、Oracle または Internet Assigned Numbers Authority (IANA) のネーミング規則で命名できます。charset が NULL の場合、データベース・キャラクタ・セットと想定されます。

SET_PERSISTENT_CONN_SUPPORT プロシージャ

このプロシージャは、要求の HTTP 1.1 の永続的接続のサポートを有効または無効にします。

永続的接続が HTTP 要求に対して有効である場合、要求が完了した後、パッケージは Web サーバーまたはプロキシ・サーバーへのネットワーク接続をオープンのまま維持し、同じサーバーへの後続の要求で HTTP 1.1 プロトコルの各仕様を再利用します。永続的接続がサポートされている場合、ネットワーク接続の待機時間が回避されるため、後続の HTTP 要求は高速に処理されます。永続的接続が HTTP 要求に対して無効である場合、要求が完了した後、Connection: close HTTP ヘッダーが HTTP 要求に自動的に送信され、ネットワーク接続がクローズされます。この設定は、HTTP 1.0 プロトコルに準拠した HTTP 要求には影響ありません。これは、要求が完了した後、ネットワーク接続が常にクローズされるためです。

要求が作成されると、ターゲット Web サーバー（プロキシ・サーバー）が使用可能であれば、パッケージは常に既存の永続的接続を再利用しようとします。使用可能なサーバーがない場合、新しいネットワーク接続が開始されます。要求に対する永続的接続サポートの設定が影響するのは、要求が完了した後にネットワーク接続をクローズするかどうかということについてのみです。

このプロシージャを使用して、要求がセッションのデフォルト設定を継承する永続的接続のサポート設定を変更します。

UTL_HTTP で永続的接続を使用すると、同じサーバーから複数の Web ページをフェッチする回数を軽減する一方で、データベース・サーバーのシステム・リソース（ネットワーク接続）が浪費されます。また、データベース・サーバーで多くのネットワーク接続がオープンされたままである場合、永続的接続を過度に使用すると、データベース・サーバーのスケーラビリティが低下します。ネットワーク接続は、後続の要求により即時使用される場合にのみオープンにし、不要な場合にはクローズしてください。通常、セッションでは永続的接続サポートを無効にし、個々の HTTP 要求で永続的接続を有効にします。詳細は、96-49 ページの「例 : HTTP 要求での SET_PERSISTENT_CONN_SUPPORT の使用」を参照してください。

構文

```
UTL_HTTP.set_persistent_conn_support(
   r       IN OUT NOCOPY req,
   enable  IN BOOLEAN DEFAULT FALSE);
```

パラメータ

表 96-38 SET_PERSISTENT_CONN_SUPPORT プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 要求。
enable (IN)	ネットワークを永続的に接続するには TRUE に設定します。そうでない場合は FALSE に設定します。

使用上の注意

データベース・セッションで保存される永続的接続のデフォルト値は 0（ゼロ）です。永続的接続を実際に有効にするには、永続的接続の最大数を正の値に設定する必要があります。そうでない場合には、接続は永続的に保持されません。

例 : HTTP 要求での SET_PERSISTENT_CONN_SUPPORT の使用

```
DECLARE
  TYPE vc2_table IS TABLE OF VARCHAR2(256) INDEX BY binary_integer;
  paths vc2_table;

UTL_HTTP.fetch_pages(paths IN vc2_table) AS
    url_prefix VARCHAR2(256) := 'http://www.my-company.com/';
    req   utl_http.req;
    resp  utl_http.resp;
    data  VARCHAR2(1024);
  BEGIN
    FOR i IN 1..paths.count LOOP
      req := utl_http.begin_request(url_prefix || paths(i));

      -- Use persistent connection except for the last request
      IF (i < paths.count) THEN
        utl_http.set_persistent_conn_support(req, TRUE);
      END IF;

      resp := utl_http.get_response(req);

      BEGIN
        LOOP
          utl_http.read_text(resp, data);
          -- do something with the data
        END LOOP;
      EXCEPTION
        WHEN utl_http.end_of_body THEN
          NULL;
      END;
      utl_http.end_response(resp);
    END LOOP;
  END;

BEGIN
  utl_http.set_persistent_conn_support(FALSE, 1);
  paths(1) := '...';
  paths(2) := '...';
  ...
  fetch_pages(paths);
END;
```

WRITE_TEXT プロシージャ

このプロシージャは、HTTP 要求本体にテキスト・データを書き込みます。HTTP 要求本体と同じデータが送信されると、HTTP 要求ヘッダーのセクションはただちに完了します。テキスト・データは、データベース・キャラクタ・セットから要求本体のキャラクタ・セットに自動的に変換されます。

構文

```
UTL_HTTP.write_text(
   r     IN OUT NOCOPY req,
   data  IN VARCHAR2);
```

パラメータ

表 96-39 WRITE_TEXT プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 要求。
data (IN)	HTTP 要求本体に組み込まれて送信されるテキスト・データ。

使用上の注意

HTTP クライアントはリモート Web サーバーに対し、送信する要求本体の長さを常に知らせる必要があります。データ量があらかじめわかっている場合、Content-Length ヘッダーを要求に設定できます。この場合、コンテンツの長さが文字数ではなくバイト数で計測されます。要求の長さが不明の場合は、HTTP 1.1 chunked transfer-encoding 形式を使用して要求本体を送信できます。要求本体はチャンクで送信されます。このとき、チャンク自体が送信される前に各チャンクの長さが送信されます。Transfer-Encoding: chunked ヘッダーが設定されると、UTL_HTTP は要求本体に対し、chunked transfer-encoding を実行します。HTTP 1.1 ベースの Web サーバーまたは CGI プログラムは、HTTP 1.1 chunked transfer-encoding 形式でエンコードされた要求本体をサポートしていません。詳細は、set_header プロシージャを参照してください。

Content-Length ヘッダーを送信する場合は、データベース・キャラクタ・セットから要求本体のキャラクタ・セットに変換された後、ヘッダーで指定した長さとテキストの要求本体のバイト数での長さを同じにする必要があります。2 つのキャラクタ・セットのうちいずれかがマルチバイト・キャラクタ・セットである場合、要求本体のキャラクタ・セットの正確なバイト数での長さをあらかじめ認識することはできません。この場合、明示的にキャラクタ・セットの変換を実行し、結果のバイト数の長さを判別し、Content-Length ヘッダーを送信し、write_raw プロシージャを使用してキャラクタ・セットの自動変換を回避します。または、リモート Web サーバーまたは CGI プログラムが対応する場合は、HTTP 1.1 chunked transfer-encoding 形式を使用して要求本体を送信できます。UTL_HTTP はチャンクの長さを透過的に処理します。

WRITE_LINE プロシージャ

このプロシージャは、HTTP 要求本体にテキスト行を書き込み、改行文字（UTL_TCP で定義される CRLF）を使用して行を終了します。HTTP 要求本体と同じデータが送信されると、HTTP 要求ヘッダーのセクションはただちに完了します。テキスト・データは、データベース・キャラクタ・セットから要求本体のキャラクタ・セットに自動的に変換されます。

構文

```
UTL_HTTP.write_line(
   r     IN OUT NOCOPY req,
   data  IN VARCHAR2);
```

パラメータ

表 96-40 WRITE_LINE プロシージャのパラメータ

パラメータ	説明
r （IN/OUT）	HTTP 要求。
data（IN）	HTTP 要求本体に組み込まれて送信されるテキスト行。

使用上の注意

HTTP クライアントはリモート Web サーバーに対し、送信する要求本体の長さを常に知らせる必要があります。データ量があらかじめわかっている場合、Content-Length ヘッダーを要求に設定できます。この場合、コンテンツの長さが文字数ではなくバイト数で計測されます。要求の長さが不明の場合は、HTTP 1.1 chunked transfer-encoding 形式を使用して要求本体を送信できます。要求本体はチャンクで送信されます。このとき、チャンク自体が送信される前に各チャンクの長さが送信されます。Transfer-Encoding: chunked ヘッダーが設定されると、UTL_HTTP パッケージは要求本体に対し、chunked transfer-encoding を実行します。HTTP 1.1 ベースの Web サーバーまたは CGI プログラムは、HTTP 1.1 chunked transfer-encoding 形式でエンコードされた要求本体をサポートしていません。詳細は、set_header プロシージャを参照してください。

Content-Length ヘッダーを送信する場合は、データベース・キャラクタ・セットから要求本体のキャラクタ・セットに変換された後、ヘッダーで指定した長さとテキストの要求本体のバイト数での長さを同じにする必要があります。2 つのキャラクタ・セットのうちいずれかがマルチバイト・キャラクタ・セットである場合、要求本体のキャラクタ・セットの正確なバイト数での長さをあらかじめ認識することはできません。この場合、明示的にキャラクタ・セットの変換を実行し、結果のバイト数の長さを判別し、Content-Length ヘッダーを送信し、write_raw プロシージャを使用してキャラクタ・セットの自動変換を回避します。または、リモート Web サーバーまたは CGI プログラムが対応する場合は、HTTP 1.1 chunked transfer-encoding 形式を使用して要求本体を送信できます。UTL_HTTP はチャンクの長さを透過的に処理します。

WRITE_RAW プロシージャ

このプロシージャは、HTTP 要求本体にバイナリ・データを書き込みます。HTTP 要求本体と同じデータが送信されると、HTTP 要求ヘッダーのセクションはただちに完了します。

構文

```
UTL_HTTP.write_raw(
   r     IN OUT NOCOPY req,
   data  IN RAW);
```

パラメータ

表 96-41 WRITE_RAW プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 要求。
data (IN)	HTTP 要求本体に組み込まれて送信されるバイナリ・データ。

使用上の注意

HTTP クライアントはリモート Web サーバーに対し、送信する要求本体の長さを常に知らせる必要があります。データ量があらかじめわかっている場合、Content-Length ヘッダーを要求に設定できます。この場合、コンテンツの長さが文字数ではなくバイト数で計測されます。要求の長さが不明の場合は、HTTP 1.1 chunked transfer-encoding 形式を使用して要求本体を送信できます。要求本体はチャンクで送信されます。このとき、チャンク自体が送信される前に各チャンクの長さが送信されます。`Transfer-Encoding: chunked` ヘッダーが設定されると、`UTL_HTTP` は要求本体に対し、chunked transfer-encoding を実行します。HTTP 1.1 ベースの Web サーバーまたは CGI プログラムは、HTTP 1.1 chunked transfer-encoding 形式でエンコードされた要求本体をサポートしていません。詳細は、`set_header` プロシージャを参照してください。

END_REQUEST プロシージャ

このプロシージャは、HTTP 要求を終了します。要求の完了および応答の待機を行わずに HTTP 要求を終了するために、プログラムでこのプロシージャをコールできます。または、プログラムは要求の開始、応答の待機、応答のクローズという通常の順序を経て要求を終了することもできます。ネットワーク接続は常にクローズされ、再利用されることはありません。

構文

```
UTL_HTTP.end_request (
   r  IN OUT NOCOPY req);
```

パラメータ

表 96-42 END_REQUEST プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 要求。

HTTP 応答

次の API は、GET_RESPONSE から取得した HTTP 応答を操作し、Web サーバーから応答情報を受信します。応答が要求に対して作成されると、現行のセッションの HTTP Cookie サポート、follow_redirect、本体のキャラクタ・セット、永続的接続のサポートおよび転送タイムアウトについてデフォルト設定が継承されます。このとき要求 API をコールすることにより後で変更できるのは、本体のキャラクタ・セットのみです。

GET_RESPONSE ファンクション

このファンクションは、HTTP 応答を読み込みます。ファンクションが戻ると、ステータス行および HTTP 応答ヘッダーが読み込まれて処理されます。ステータス・コード、理由、HTTP プロトコルのバージョンは応答レコードに格納されます。このファンクションは、HTTP ヘッダー・セクションを完了します。

構文

```
UTL_HTTP.get_response (
   r  IN OUT NOCOPY req)
RETURN resp;
```

パラメータ

表 96-43 GET_RESPONSE プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。

GET_HEADER_COUNT ファンクション

このファンクションは、応答で戻された HTTP 応答ヘッダーの数を戻します。

構文

```
UTL_HTTP.get_header_count (
   r  IN OUT NOCOPY resp)
RETURN PLS_INTEGER;
```

パラメータ

表 96-44 GET_HEADER_COUNT ファンクションのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。

使用上の注意

リモート Web サーバーから戻された応答本体が chunked transfer encoding 形式でエンコードされている場合、応答本体の末尾で戻される追跡ヘッダーが応答に追加され、応答ヘッダーのカウントが更新されます。応答本体の末尾に到達した後、応答を終了する前に追加ヘッダーを取り出すことができます。

GET_HEADER プロシージャ

このプロシージャは、応答で戻された n 回目の HTTP 応答ヘッダーの名前および値を戻します。

構文

```
UTL_HTTP.get_header (
   r      IN OUT NOCOPY resp,
   n      IN PLS_INTEGER,
   name   OUT NOCOPY VARCHAR2,
   value  OUT NOCOPY VARCHAR2);
```

パラメータ

表 96-45 GET_HEADER プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。
n (IN)	n 番目に戻されるヘッダー。
name (OUT)	HTTP 応答ヘッダーの名前。
value (OUT)	HTTP 応答ヘッダーの値。

使用上の注意

リモート Web サーバーから戻された応答本体が chunked transfer encoding 形式でエンコードされている場合、応答本体の末尾で戻される追跡ヘッダーが応答に追加され、応答ヘッダーのカウントが更新されます。応答本体の末尾に到達した後、応答を終了する前に追加ヘッダーを取り出すことができます。

GET_HEADER_BY_NAME プロシージャ

このプロシージャは、特定のヘッダー名を持つ応答に戻された HTTP 応答ヘッダーの値を戻します。

構文

```
UTL_HTTP.get_header_by_name(
   r      IN OUT NOCOPY resp,
   name   IN VARCHAR2,
   value  OUT NOCOPY VARCHAR2,
   n      IN PLS_INTEGER DEFAULT 1);
```

パラメータ

表 96-46 GET_HEADER_BY_NAME プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。
n (IN)	戻すように指定した名前により n 番目に発生する HTTP 応答ヘッダー。デフォルトは 1 です。
name (IN)	値が戻される HTTP 応答ヘッダーの名前。
value (OUT)	HTTP 応答ヘッダーの値。

使用上の注意

リモート Web サーバーから戻された応答本体が chunked transfer encoding 形式でエンコードされている場合、応答本体の末尾で戻される追跡ヘッダーが応答に追加され、応答ヘッダーのカウントが更新されます。応答本体の末尾に到達した後、応答を終了する前に追加ヘッダーを取り出すことができます。

GET_AUTHENTICATION プロシージャ

このプロシージャは、Web サーバーが受け付ける要求に必要とされる HTTP 認証情報を HTTP 応答ヘッダーの指示に従って取り出します。

構文

```
UTL_HTTP.get_authentication(
   r          IN OUT NOCOPY resp,
   scheme     OUT VARCHAR2,
   realm      OUT VARCHAR2,
   for_proxy  IN BOOLEAN  DEFAULT FALSE);
```

パラメータ

表 96-47 GET_AUTHENTICATION プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。
scheme (OUT)	要求された HTTP 認証のスキーム。
realm (OUT)	要求された HTTP 認証のレルム。
for_proxy (IN)	Web サーバーではなく HTTP プロキシ・サーバーにアクセスするために必要な HTTP 認証情報を戻します。デフォルトは FALSE です。

使用上の注意

ドキュメントが保護されていないことを Web クライアントが認識していない場合、そのドキュメントを取り出すには少なくとも 2 つの HTTP 要求が必要です。最初の HTTP 要求で、Web クライアントは必要な認証情報を提供せずに要求を作成します。したがって、この要求は拒否されます。Web クライアントは、get_authentication をコールすることにより、認証対象となる要求に必要な認証情報を判別します。Web クライアントは 2 回目の要求を行い、set_authorization を使用して必要な認証情報を提供します。Web サーバーが認証情報を検証できれば、要求は正常に処理され、要求したドキュメントが戻されます。要求を行う前に、認証情報が必要であることを Web クライアントが認識している場合は、最初の要求で必要な認証情報を提供できます。これにより、余分な要求を行わずに済みます。

SET_BODY_CHARSET プロシージャ

このプロシージャは、メディア・タイプが text で、キャラクタ・セットが Content-Type ヘッダーで指定されていない場合は、応答本体のキャラクタ・セットを設定します。要求または応答のメディア・タイプが text で、キャラクタ・セット情報が Content-Type ヘッダーにない場合は、HTTP プロトコルの標準仕様に従って、要求または応答の本体のキャラクタ・セットのデフォルトに ISO-8859-1 を設定する必要があります。

このプロシージャを使用して、応答が要求から継承するデフォルトの本体キャラクタ・セットを変更します。

構文

```
UTL_HTTP.set_body_charset(
   r        IN OUT NOCOPY resp,
   charset  IN VARCHAR2 DEFAULT NULL);
```

パラメータ

表 96-48 SET_BODY_CHARSET プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。
charset (IN)	応答本体のデフォルトのキャラクタ・セット。キャラクタ・セットは、Oracle または Internet Assigned Numbers Authority (IANA) のネーミング規則で命名できます。charset が NULL の場合、データベース・キャラクタ・セットと想定されます。

READ_TEXT プロシージャ

このプロシージャは、HTTP 応答本体をテキスト形式で読み込み、コール元が提供したバッファに出力を戻します。HTTP 応答本体の終端に到達した場合、end_of_body 例外が発生します。テキスト・データは、データベース・キャラクタ・セットから応答本体のデータベース・キャラクタ・セットに自動的に変換されます。

構文

```
UTL_HTTP.read_text(
   r     IN OUT NOCOPY resp,
   data  OUT NOCOPY VARCHAR2,
   len   IN PLS_INTEGER DEFAULT NULL);
```

パラメータ

表 96-49 READ_TEXT プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。
data (OUT)	テキスト形式の HTTP 応答本体。
len (IN)	読み込まれるデータの最大文字数。len が NULL の場合、このプロシージャは、data に割り当てられたバッファを最大限に活用して入力を読み込みます。HTTP 応答本体の末尾に到達するまで、または transfer_timeout の時間量が経過するまでに使用できるデータが少ない場合は、実際に戻されるデータ量が指定した量より減る可能性があります。デフォルトは NULL です。

使用上の注意

UTL_HTTP パッケージは、HTTP 1.1 chunked transfer-encoding をサポートしています。応答ヘッダーで指定された chunked transfer-encoding 形式で応答本体が戻された場合、パッケージは自動的にチャンクをデコードし、チャンクを解除した形式の応答本体を戻します。

転送のタイムアウトがこの応答の要求で設定されている場合、read_text は、タイムアウトが発生するまでの間、各データ・パケットが読み込まれるのを待機します。タイムアウトが発生すると、このプロシージャは読込みを停止し、正常に読み込んだデータをすべて戻します。正常に読み込まれたデータがない場合は、transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

マルチバイト・キャラクタの一部が応答本体の終わりで検出された場合、read_text は読込みを中止し、完全に読み込まれたマルチバイト・キャラクタをすべて戻します。正常に読み込まれた文字がない場合は、partial_multibyte_char 例外が発生します。read_raw プロシージャを使用して例外を処理し、部分的なマルチバイト・キャラクタのバイトをバイナリとして読み込むことができます。マルチバイト・キャラクタの一部が渡されていない状

態でタイムアウトが発生し、応答本体の途中でマルチバイト・キャラクタの残りの部分が表示された場合は、transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

Content-Type 応答ヘッダーによって指定されている応答本体のキャラクタ・セットが不明で、Oracle ではサポートされていない場合は、応答本体をテキストとして読み込もうとすると、「ORA-01482: 指定されたキャラクタ・セットはサポートされていません。」という例外が発生します。応答本体は、READ_RAW プロシージャを使用してバイナリとして読み込むか、SET_BODY_CHARSET プロシージャを使用して、その応答本体のキャラクタ・セットを明示的に設定することで再度テキストとして読み込むことができます。

READ_LINE プロシージャ

このプロシージャは、HTTP 応答本体を行末までテキスト形式で読み込み、コール元が提供したバッファに出力を戻します。行末は、UTL_TCP の read_line ファンクションで定義されます。HTTP 応答本体の終端に到達した場合、end_of_body 例外が発生します。テキスト・データは、データベース・キャラクタ・セットから応答本体のデータベース・キャラクタ・セットに自動的に変換されます。

構文

```
UTL_HTTP.read_line(
   r           IN OUT NOCOPY resp,
   data        OUT NOCOPY  VARCHAR2,
   remove_crlf  IN  BOOLEAN DEFAULT FALSE);
```

パラメータ

表 96-50 READ_LINE プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。
data (OUT)	テキスト形式の HTTP 応答本体。
remove_crlf (IN)	TRUE の場合、改行文字が削除されます。

使用上の注意

UTL_HTTP パッケージは、HTTP 1.1 chunked transfer-encoding をサポートしています。応答ヘッダーで指定された chunked transfer-encoding 形式で応答本体が戻された場合、パッケージは自動的にチャンクをデコードし、チャンクを解除した形式の応答本体を戻します。

転送のタイムアウトがこの応答の要求で設定されている場合、read_line は、タイムアウトが発生するまでの間、各データ・パケットが読み込まれるのを待機します。タイムアウトが発生すると、このプロシージャは読込みを停止し、正常に読み込んだデータをすべて戻します。正常に読み込まれたデータがない場合は、transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

マルチバイト・キャラクタの一部が応答本体の終わりで検出された場合、read_line は読込みを中止し、完全に読み込まれたマルチバイト・キャラクタをすべて戻します。正常に読み込まれた文字がない場合は、partial_multibyte_char 例外が発生します。read_raw プロシージャを使用して例外を処理し、部分的なマルチバイト・キャラクタのバイトをバイナリとして読み込むことができます。マルチバイト・キャラクタの一部が渡されていない状態でタイムアウトが発生し、応答本体の途中でマルチバイト・キャラクタの残りの部分が表示された場合は、transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

Content-Type 応答ヘッダーによって指定されている応答本体のキャラクタ・セットが不明で、Oracle ではサポートされていない場合は、応答本体をテキストとして読み込もうとすると、「ORA-01482: 指定されたキャラクタ・セットはサポートされていません。」という例外が発生します。応答本体は、READ_RAW プロシージャを使用してバイナリとして読み込むか、SET_BODY_CHARSET プロシージャを使用して、その応答本体のキャラクタ・セットを明示的に設定することで再度テキストとして読み込むことができます。

READ_RAW プロシージャ

このプロシージャは、HTTP 応答本体をバイナリ形式で読み込み、コール元が提供したバッファに出力を戻します。HTTP 応答本体の終端に到達した場合、end_of_body 例外が発生します。

構文

```
UTL_HTTP.read_raw(
   r     IN OUT NOCOPY resp,
   data  OUT NOCOPY RAW,
   len   IN PLS_INTEGER DEFAULT NULL);
```

パラメータ

表 96-51 READ_RAW プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。
data (OUT)	バイナリ形式の HTTP 応答本体。
len (IN)	読み込むデータのバイト数。len が NULL の場合、このプロシージャは、data に割り当てられたバッファを最大限に活用して入力を読み込みます。HTTP 応答本体の末尾に到達するまで、または transfer_timeout の時間量が経過するまでに使用できるデータが少ない場合は、実際に戻されるデータ量が指定した量より減る可能性があります。デフォルトは NULL です。

使用上の注意

UTL_HTTP パッケージは、HTTP 1.1 chunked transfer-encoding をサポートしています。応答ヘッダーで指定された chunked transfer-encoding 形式で応答本体が戻された場合、パッケージは自動的にチャンクをデコードし、チャンクを解除した形式の応答本体を戻します。

転送のタイムアウトがこの応答の要求で設定されている場合、read_raw は、タイムアウトが発生するまでの間、各データ・パケットが読み込まれるのを待機します。タイムアウトが発生すると、read_raw は読込みを停止し、正常に読み込んだデータをすべて戻します。正常に読み込まれたデータがない場合は、transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

END_RESPONSE プロシージャ

このプロシージャは、HTTP 応答を終了します。HTTP 要求および応答を完了します。この要求で HTTP 1.1 の永続的接続を使用している場合を除き、ネットワーク接続もクローズされます。

構文

```
UTL_HTTP.end_response (
   r  IN OUT NOCOPY resp);
```

パラメータ

表 96-52 END_RESPONSE プロシージャのパラメータ

パラメータ	説明
r (IN/OUT)	HTTP 応答。

HTTP Cookie

次の API を使用して、HTTP Cookie を操作します。

GET_COOKIE_COUNT ファンクション

このファンクションは、すべての Web サーバーにより設定された UTL_HTTP パッケージが現在維持する Cookie の数を戻します。

構文

```
UTL_HTTP.get_cookie_count
RETURN PLS_INTEGER;
```

GET_COOKIES ファンクション

このファンクションは、すべての Web サーバーに設定された UTL_HTTP パッケージが現在維持する Cookie をすべて戻します。

構文

```
UTL_HTTP.get_cookies (
   cookies  IN OUT NOCOPY cookie_table);
```

パラメータ

表 96-53 GET_COOKIES プロシージャのパラメータ

パラメータ	説明
cookies (IN/OUT)	戻される Cookie。

ADD_COOKIES プロシージャ

このプロシージャは、UTL_HTTP に維持される Cookie を追加します。

構文

```
UTL_HTTP.add_cookies (
   cookies  IN cookie_table);
```

パラメータ

表 96-54 ADD_COOKIES プロシージャのパラメータ

パラメータ	説明
cookies (IN/OUT)	追加される Cookie。

使用上の注意

パッケージが現在維持する Cookie は、新しい Cookie が追加されるまでクリアされません。

CLEAR_COOKIES プロシージャ

このプロシージャは、UTL_HTTP パッケージに維持される Cookie をすべてクリアします。

構文

```
UTL_HTTP.clear_cookies;
```

HTTP の永続的接続

次のファンクションを使用して、永続的接続を操作します。

GET_PERSISTENT_CONN_COUNT ファンクション

このファンクションは、UTL_HTTP パッケージにより現在永続的に維持されているネットワーク接続数を Web サーバーに戻します。

構文

```
UTL_HTTP.get_persistent_conn_count
RETURN PLS_integer;
```

使用上の注意

異なる TCP/IP ポートでの同じ Web サーバーへの接続は、個別にカウントされます。Web サーバーのホスト名は、元の HTTP 要求の URL で指定されたとおりに識別されます。したがって、ドメイン名を持つ完全修飾されたホスト名は、ドメイン名のないホスト名とは別にカウントされます。

GET_PERSISTENT_CONNS プロシージャ

このプロシージャは、UTL_HTTP パッケージにより現在永続的に維持されているネットワーク接続を Web サーバーにすべて戻します。

構文

```
UTL_HTTP.get_persistent_conns (
   connections  IN OUT NOCOPY connection_table);
```

パラメータ

表 96-55 GET_PERSISTENT_CONNS プロシージャのパラメータ

パラメータ	説明
connections (IN/OUT)	ネットワーク接続が永続的に維持されます。

使用上の注意

異なる TCP/IP ポートでの同じ Web サーバーへの接続は、個別にカウントされます。Web サーバーのホスト名は、元の HTTP 要求の URL で指定されたとおりに識別されます。したがって、ドメイン名を持つ完全修飾されたホスト名は、ドメイン名のないホスト名とは別にカウントされます。

CLOSE_PERSISTENT_CONN プロシージャ

このプロシージャは、現在のデータベース・セッションで UTL_HTTP パッケージが維持する HTTP の永続的接続をクローズします。

構文

```
UTL_HTTP.close_persistent_conn (
   conn  IN connection);
```

パラメータ

表 96-56 CLOSE_PERSISTENT_CONN プロシージャのパラメータ

パラメータ	説明
conn (IN)	クローズする HTTP 永続的接続

CLOSE_PERSISTENT_CONNS プロシージャ

このプロシージャは、現在のデータベース・セッションで UTL_HTTP パッケージが維持する HTTP の永続的接続のグループをクローズします。このプロシージャは、パターンを一致させる方法を使用して、クローズする永続的接続を判別します。

共通のプロパティ（特定のホストへのすべての接続やすべての SSL 接続など）を共有する HTTP 永続的接続のグループをクローズするには、特定のパラメータを設定して残りのパラメータを NULL にします。このプロシージャがコールされたときに特定のパラメータが NULL に設定されていると、そのパラメータはクローズする接続の判別に使用されません。

たとえば、次のコールをプロシージャに使用して、foobar への永続的接続をすべてクローズします。

```
utl_http.close_persistent_conns(host => 'foobar');
```

プロシージャに次のコールを使用すると、foobar を介して TCP/IP ポート 80 の永続的接続をすべてクローズします。

```
utl_http.close_persistent_conns(proxy_host => 'foobar',
                                proxy_port => 80);
```

次のコールをプロシージャに使用して、永続的接続をすべてクローズします。

```
utl_http.close_persistent_conns;
```

構文

```
UTL_HTTP.close_persistent_conns (
   host         IN VARCHAR2 DEFAULT NULL,
   port         IN PLS_INTEGER DEFAULT NULL,
   proxy_host   IN VARCHAR2 DEFAULT NULL,
   proxy_port   IN PLS_INTEGER DEFAULT NULL,
   ssl          IN BOOLEAN DEFAULT NULL);
```

パラメータ

表 96-57 CLOSE_PERSISTENT_CONNS プロシージャのパラメータ

パラメータ	説明
host（IN）	永続的接続をクローズするホスト。
port（IN）	永続的接続をクローズするポート番号。
proxy_host （IN）	永続的接続をクローズするプロキシ・ホスト。
proxy_port （IN）	永続的接続をクローズするプロキシ・ポート。
ssl （IN）	永続的な SSL 接続をクローズします。

使用上の注意

異なる TCP/IP ポートでの同じ Web サーバーへの接続は、個別にカウントされます。Web サーバーのホスト名は、元の HTTP 要求の URL で指定されたとおりに識別されます。したがって、ドメイン名を持つ完全修飾されたホスト名は、ドメイン名のないホスト名とは別にカウントされます。

このプロシージャがコールされたときにパラメータで NULL 値を使用すると、パッケージでクローズする永続的接続を決定したときに、コール元がその値を無視することを意味します。特定の永続的接続をクローズする場合に永続的接続で NULL に設定されているパラメータの値のみを NULL に設定するには、close_persistent_conn プロシージャを使用して、特定の永続的接続をクローズします。

エラー状態

次の API がエラー情報を取り出します。

GET_DETAILED_SQLCODE ファンクション

このファンクションは、最後に発生した例外の詳細な SQLCODE を取り出します。

構文

```
UTL_HTTP.get_detailed_sqlcode
RETURN PLS_INTEGER;
```

GET_DETAILED_SQLERRM ファンクション

このファンクションは、最後に発生した例外の詳細な SQLERRM を取り出します。

構文

```
UTL_HTTP.get_detailed_sqlerrm
RETURN VARCHAR2;
```

97

UTL_INADDR

UTL_INADDR は、インターネット・アドレッシングをサポートするための PL/SQL プロシージャを提供します。ホスト名を取り出す API およびローカル・ホストとリモート・ホストの IP アドレスを提供します。

この章では、次の項目について説明します。

例外

表 97-1 インターネット・アドレス・パッケージからの例外

例外	説明
UNKNOWN_HOST	ホストが不明です。

UTL_INADDR サブプログラムの要約

表 97-2 UTL_INADDR サブプログラム

サブプログラム	説明
「get_host_name ファンクション」97-2 ページ	IP アドレスが指定されたローカルまたはリモート・ホストの名前を取り出します。
「get_host_address ファンクション」97-3 ページ	名前が指定されたローカルまたはリモート・ホストの IP アドレスを取り出します。

get_host_name ファンクション

このファンクションは、IP アドレスが指定されたローカルまたはリモート・ホストの名前を取り出します。

構文

```
UTL_INADDR.GET_HOST_NAME (
   ip  IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

パラメータ

表 97-3 get_host_name ファンクションのパラメータ

パラメータ	説明
ip	ホスト名を決定するために使用するホストの IP アドレス。ip が NULL でない場合は、ドメイン名を使用したホストの正式名が戻されます。NULL の場合は、ローカル・ホスト名が戻されます。名前にはローカル・ホストが属するドメインは含まれません。

戻り値

指定した IP アドレスのローカルまたはリモート・ホスト名。

例外

unknown_host。指定した IP アドレスが不明です。

get_host_address ファンクション

このファンクションは、ホストの IP アドレスを取り出します。

構文

```
UTL_INADDR.GET_HOST_ADDRESS (
   host  IN VARCHAR2 DEFAULT NULL)
RETURN VARCHAR2;
```

パラメータ

表 97-4 get_host_address ファンクションのパラメータ

パラメータ	説明
host（IN）	IP アドレスを取り出すホストの名前。host が NULL の場合、このファンクションはローカル・ホストの IP アドレスを戻します。

98

UTL_RAW

UTL_RAW パッケージは、RAW データ・タイプを操作するための SQL ファンクションを提供します。通常の SQL ファンクションは複数の RAW で作動せず、PL/SQL は RAW データ・タイプと CHAR データ・タイプの間でのオーバーロードができないため、このパッケージが必要になります。UTL_RAW には、各種の COBOL 数値形式を複数の RAW の間で変換するサブプログラムも含まれています。

UTL_RAW は、データベース環境に固有ではなく、他の環境でもデータベース環境と同じように実際に使用できます。このため、DBMS のかわりに、UTL という接頭辞がパッケージに付けられます。

この章では、次の項目について説明します。

使用上の注意

UTL_RAW によって、RAW レコードは多くの要素で構成できます。RAW データ・タイプを使用すると、キャラクタ・セット変換は実行されず、RAW は、リモート・プロシージャ・コールを介して転送されるときに元の形式で保持されます。

また、RAW ファンクションによって、以前は hextoraw ファンクションと rawtohex ファンクションに限定されていたバイナリ・データを操作できます。

UTL_RAW サブプログラムの要約

表 98-1 UTL_RAW サブプログラム

サブプログラム	説明
「CAST_FROM_BINARY_INTEGER ファンクション」98-4 ページ	（RAW の）BINARY_INTEGER のバイナリ表現を戻します。
「CAST_FROM_NUMBER ファンクション」98-5 ページ	（RAW の）NUMBER のバイナリ表現を戻します。
「CAST_TO_BINARY_INTEGER ファンクション」98-6 ページ	（RAW の）BINARY_INTEGER のバイナリ表現を BINARY_INTEGER に変換します。
「CAST_TO_NUMBER ファンクション」98-7 ページ	（RAW の）NUMBER のバイナリ表現を NUMBER に変換します。
「CAST_TO_RAW ファンクション」98-8 ページ	n データ・バイトを使用して表した VARCHAR2 を、n データ・バイトを持つ RAW に変換します。
「CAST_TO_VARCHAR2 ファンクション」98-9 ページ	n データ・バイトを使用して表した RAW を、n データ・バイトを持つ VARCHAR2 に変換します。
「CONCAT ファンクション」98-10 ページ	最大 12 までの RAW を単一の RAW に連結します。
「LENGTH ファンクション」98-11 ページ	RAW r の長さをバイトで戻します。
「SUBSTR ファンクション」98-12 ページ	RAW r の pos から len バイトを戻します。
「TRANSLATE ファンクション」98-13 ページ	from_set と to_set のバイト列変換に従って、入力 RAW r 内のバイト列を変換します。
「TRANSLITERATE ファンクション」98-15 ページ	from_set と to_set のバイト列変換に従って、入力 RAW r 内のバイト列を変換します。

表 98-1 UTL_RAW サブプログラム（続き）

サブプログラム	説明
「OVERLAY ファンクション」98-17 ページ	target RAW の指定部分を overlay_str RAW でオーバーレイし、target の位置 pos バイトから始まる len バイト分を処理します。
「COPIES ファンクション」98-19 ページ	r を n 回のコピーで連結したものを戻します。
「XRANGE ファンクション」98-20 ページ	値 start_byte で始まり値 end_byte で終わる、連続した有効な 1 バイト・コードをすべて含む RAW を戻します。
「REVERSE ファンクション」98-21 ページ	RAW r のバイトの順序を、最後から最初に逆転させます。
「COMPARE ファンクション」98-22 ページ	RAW r1 と RAW r2 を比較します。
「CONVERT ファンクション」98-23 ページ	RAW r をキャラクタ・セット from_charset からキャラクタ・セット to_charset に変換し、結果の RAW を戻します。
「BIT_AND ファンクション」98-24 ページ	RAW r1 と RAW r2 の値でビット単位の論理演算 AND を実行し、AND 演算後の結果 RAW を戻します。
「BIT_OR ファンクション」98-25 ページ	RAW r1 と RAW r2 の値でビット単位の論理演算 OR を実行し、OR 演算後の結果 RAW を戻します。
「BIT_XOR ファンクション」98-26 ページ	RAW r1 と RAW r2 の値でビット単位の論理演算 XOR を実行し、XOR 演算後の結果 RAW を戻します。
「BIT_COMPLEMENT ファンクション」98-27 ページ	RAW r の値でビット単位の論理演算補数を実行し、補数演算後の結果 RAW を戻します。

CAST_FROM_BINARY_INTEGER ファンクション

このファンクションは、（RAW の）BINARY_INTEGER のバイナリ表現を戻します。

構文

```
UTL_RAW.CAST_FROM_BINARY_INTEGER (
   n          IN BINARY_INTEGER
   endianess  IN PLS_INTEGER DEFAULT BIG_ENDIAN)
RETURN RAW;
```

プラグマ

```
pragma restrict_references(cast_from_binary_integer, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-2 CAST_FROM_BINARY_INTEGER ファンクションのパラメータ

パラメータ	説明
n	BINARY_INTEGER の値。
endianess	big-endian または little-endian のアーキテクチャを表す PLS_INTEGER。デフォルトは big-endian です。

戻り値

BINARY_INTEGER の値のバイナリ表現。

CAST_FROM_NUMBER ファンクション

このファンクションは、(RAW の) NUMBER のバイナリ表現を戻します。include_length が TRUE の場合、戻される RAW の最初のバイトは、数値の有効なバイト数をエンコードします。このとき長さはバイト数に含まれません。また、その結果は任意のデータを使用して 22 バイトの固定長に拡張されます。include_length が FALSE の場合、戻される RAW は最大 21 バイトの可変長です。

構文

```
UTL_RAW.CAST_FROM_NUMBER (
   n              IN NUMBER
   include_length  IN BOOLEAN)
RETURN RAW;
```

プラグマ

```
pragma restrict_references(cast_from_number, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-3 CAST_FROM_NUMBER ファンクションのパラメータ

パラメータ	説明
n	NUMBER の値。

戻り値

NUMBER の値のバイナリ表現。

CAST_TO_BINARY_INTEGER ファンクション

このファンクションは、（RAW の）BINARY_INTEGER のバイナリ表現を BINARY_INTEGER に変換します。

構文

```
UTL_RAW.CAST_TO_BINARY_INTEGER (
   r          IN RAW
   endianess  IN PLS_INTEGER DEFAULT BIG_ENDIAN)
RETURN BINARY_INTEGER;
```

プラグマ

```
pragma restrict_references(cast_to_binary_integer, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-4 CAST_TO_BINARY_INTEGER ファンクションのパラメータ

パラメータ	説明
r	BINARY_INTEGER のバイナリ表現。
endianess	big-endian または little-endian のアーキテクチャを表す PLS_INTEGER。デフォルトは big-endian です。

戻り値

BINARY_INTEGER の値。

CAST_TO_NUMBER ファンクション

このファンクションは、（RAW の）NUMBER のバイナリ表現を NUMBER に変換します。include_length が TRUE の場合、r の最初のバイトは、r の有効なバイト数をエンコードします。このとき長さはバイト数に含まれません。有効なバイト数とは、最大 21 バイトに長さのバイトを付加したものです。

構文

```
UTL_RAW.CAST_TO_NUMBER (
   r               IN RAW
   include_length  IN BOOLEAN)
RETURN NUMBER;
```

プラグマ

```
pragma restrict_references(cast_to_number, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-5 CAST_TO_NUMBER ファンクションのパラメータ

パラメータ	説明
r	NUMBER のバイナリ表現。

戻り値

NUMBER の値。

CAST_TO_RAW ファンクション

このファンクションは、n データ・バイトを使用して表した VARCHAR2 を、n データ・バイトの RAW に変換します。データは変更されませんが、データ・タイプのみ RAW データ・タイプに変換されます。

構文

```
UTL_RAW.CAST_TO_RAW (
   c  IN VARCHAR2)
RETURN RAW;
```

プラグマ

```
pragma restrict_references(cast_to_raw, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-6 CAST_TO_RAW ファンクションのパラメータ

パラメータ	説明
c	RAW に変換される VARCHAR2。

戻り値

表 98-7 CAST_TO_RAW ファンクションの戻り値

戻り値	説明
RAW	先行する長さのフィールドのない、入力 VARCHAR2 と同じバイト長の同じデータ。
NULL	入力パラメータ c が NULL の場合。

CAST_TO_VARCHAR2 ファンクション

このファンクションは、n データ・バイトを使用して表した RAW を、n データ・バイトの VARCHAR2 に変換します。

> **注意：** VARCHAR2 への変換時、その VARCHAR2 内の文字に対して現行のグローバリゼーション・サポート・キャラクタ・セットが使用されます。

構文

```
UTL_RAW.CAST_TO_VARCHAR2 (
   r IN RAW)
  RETURN VARCHAR2;
```

プラグマ

```
pragma restrict_references(cast_to_varchar2, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-8 CAST_TO_VARCHAR2 ファンクションのパラメータ

パラメータ	説明
r	VARCHAR2 に変更する RAW（先行する長さのフィールドなし）。

戻り値

表 98-9 CAST_TO_VARCHAR2 ファンクションの戻り値

戻り値	説明
VARCHAR2	入力 RAW と同じデータ。
NULL	入力パラメータ r が NULL の場合。

CONCAT ファンクション

このファンクションは、最大 12 までの RAW を単一の RAW に連結します。連結したサイズが 32K を超える場合は、エラーが戻ります。

構文

```
UTL_RAW.CONCAT (
   r1  IN RAW DEFAULT NULL,
   r2  IN RAW DEFAULT NULL,
   r3  IN RAW DEFAULT NULL,
   r4  IN RAW DEFAULT NULL,
   r5  IN RAW DEFAULT NULL,
   r6  IN RAW DEFAULT NULL,
   r7  IN RAW DEFAULT NULL,
   r8  IN RAW DEFAULT NULL,
   r9  IN RAW DEFAULT NULL,
   r10 IN RAW DEFAULT NULL,
   r11 IN RAW DEFAULT NULL,
   r12 IN RAW DEFAULT NULL)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(concat, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

r1....r12 は、連結する RAW 項目です。

戻り値

表 98-10 CONCAT ファンクションの戻り値

戻り値	説明
RAW	連結された項目。

エラー

入力値の合計の長さが RAW の最大許容長である 32767 バイトを超えると、エラーが発生します。

LENGTH ファンクション

このファンクションは、RAW r の長さをバイトで戻します。

構文

```
UTL_RAW.LENGTH (
   r  IN RAW)
RETURN NUMBER;
```

プラグマ

```
pragma restrict_references(length, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-11 LENGTH ファンクションのパラメータ

パラメータ	説明
r	長さを測定する RAW バイト・ストリーム。

戻り値

表 98-12 LENGTH ファンクションの戻り値

戻り値	説明
NUMBER	RAW の現行の長さ。

SUBSTR ファンクション

このファンクションは、RAW r の pos から len バイトを戻します。

構文

```
UTL_RAW.SUBSTR (
   r   IN RAW,
   pos IN BINARY_INTEGER,
   len IN BINARY_INTEGER DEFAULT NULL)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(substr, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

pos が正の値の場合、SUBSTR は r の初めからカウントして最初のバイトを検索します。pos が負の値の場合、SUBSTR は r の最後から逆方向にカウントします。値 pos は 0（ゼロ）に指定できません。

len が省略された場合、SUBSTR は r の終わりまですべてのバイトを戻します。値 len は 1 未満に指定できません。

表 98-13 SUBSTR ファンクションのパラメータ

パラメータ	説明
r	一部分を抽出する RAW バイト列。
pos	r 内で抽出を開始するバイト位置。
len	r から抽出する、pos からのバイト数（オプション）。

デフォルトとオプション・パラメータ

表 98-14 SUBSTR ファンクションのオプション・パラメータ

オプション・パラメータ	説明
len	位置 pos から r の終わりまでの長さ。

戻り値

表 98-15 SUBSTR ファンクションの戻り値

戻り値	説明
portion of r	pos から始まる len バイト長。
NULL	入力パラメータ r が NULL の場合。

エラー

表 98-16 SUBSTR ファンクションのエラー

エラー	説明
VALUE_ERROR	pos = 0 または len < 0 のいずれかです。

TRANSLATE ファンクション

このファンクションは、from_set と to_set のバイト列変換に従って、入力 RAW r 内のバイト列を変換します。r 内のバイト列が from_set 内のバイト列と一致すると、to_set 内の対応する位置にあるバイト列に置換され、一致しないと削除されます。

r 内のバイト列が from_set で未定義の場合は、結果にコピーされます。from_set にある最初（最左端）のバイト列のみ使用されます。後続の複製部分はスキャンされずに無視されます。to_set が from_set より短い場合は、from_set の余分なバイト列に対応する変換はなく、r 内に一致するバイト列はありません。

注意： TRANSLITERATE とは、次の点で異なります。

— 変換 RAW にデフォルトはありません。

— to_set の変換 RAW で未定義の r バイト列は削除されます。

— 結果 RAW は、入力 RAW r より短い場合があります。

構文

```
UTL_RAW.TRANSLATE (
   r        IN RAW,
   from_set IN RAW,
   to_set   IN RAW)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(translate, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-17 TRANSLATE ファンクションのパラメータ

パラメータ	説明
r	変換する RAW ソース・バイト列。
from_set	変換する RAW バイト・コード（r にある場合）。
to_set	対応する from_str バイト列が変換される RAW バイト・コード。

戻り値

表 98-18 TRANSLATE ファンクションの戻り値

戻り値	説明
RAW	変換されたバイト列。

エラー

表 98-19 TRANSLATE ファンクションのエラー

エラー	説明
VALUE_ERROR	次のいずれかです。 － r が NULL または長さ 0 です。 － from_set が NULL または長さ 0 です。 － to_set が NULL または長さ 0 です。

TRANSLITERATE ファンクション

このファンクションは、from_set と to_set のバイト列変換に従って、入力 RAW r 内のバイト列を変換します。r 内の連続するバイト列が from_set 内で検索され、見つからない場合は、変更しないまま結果 RAW にコピーされます。見つかった場合、そのバイト列は、to_set の対応するバイト列、対応するバイト列が存在しない場合は pad バイト列のいずれかに結果 RAW 内で置換されます。

r 内のバイト列が from_set で未定義の場合は、結果にコピーされます。from_set にある最初（最左端）のバイト列のみ使用されます。後続の複製部分はスキャンされずに無視されます。結果 RAW は、常に r と同じ長さになります。

to_set が from_set より短い場合、選択した from_set バイト列に対応する to_set バイトがないと、pad バイト列が結果 RAW に埋め込まれます（pad バイト列を使用して、to_set が from_set と同じ長さまで拡張されたようになります）。

注意： TRANSLATE とは、次の点で異なります。

— to_set で未定義の r バイト列が埋め込まれます。

— 結果 RAW は、常に入力 RAW r と同じ長さになります。

構文

```
UTL_RAW.TRANSLITERATE (
   r        IN RAW,
   to_set   IN RAW DEFAULT NULL,
   from_set IN RAW DEFAULT NULL,
   pad      IN RAW DEFAULT NULL)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(transliterate, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-20 TRANSLITERATE ファンクションのパラメータ

パラメータ	説明
r	変換する RAW 入力バイト列。
from_set	変換する RAW バイト・コード（r にある場合）（任意の長さ）。
to_set	対応する from_set バイト列が変換される RAW バイト・コード（任意の長さ）。
pad	to_set が from_set より短い場合に使用する 1 バイト列。

デフォルトとオプション・パラメータ

表 98-21 TRANSLITERATE ファンクションのオプション・パラメータ

オプション・パラメータ	説明
from_set	x'00' ～ x'ff'。
to_set	NULL 文字列まで。実際は、必要に応じて pad を使用して、from_set の長さまで拡張されます。
pad	x'00'。

戻り値

表 98-22 TRANSLITERATE ファンクションの戻り値

戻り値	説明
RAW	変換されたバイト列。

エラー

表 98-23 TRANSLITERATE ファンクションのエラー

エラー	説明
VALUE_ERROR	r が NULL または長さ 0 です。

OVERLAY ファンクション

このファンクションは、target RAW の指定部分を overlay_str RAW でオーバーレイし、target の位置 pos バイトから始まる len バイト分を処理します。

overlay_str が len バイト未満の場合は、pad バイト列を使用して len バイトまで拡張されます。overlay_str が len バイトを超える場合は、overlay_str の余分なバイトは無視されます。target の位置 pos から始まる len バイトが target の長さを超える場合、overlay_str 全体を含む長さまで拡張されます。

len が指定されている場合は、0（ゼロ）以上である必要があります。pos が指定されている場合は、1 以上である必要があります。pos が target の長さを超えている場合、target は pad バイト列を使用して位置 pos まで埋め込まれ、さらに target は overlay_str バイト列を使用して拡張されます。

構文

```
UTL_RAW.OVERLAY (
   overlay_str IN RAW,
   target      IN RAW,
   pos         IN BINARY_INTEGER DEFAULT 1,
   len         IN BINARY_INTEGER DEFAULT NULL,
   pad         IN RAW            DEFAULT NULL)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(overlay, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-24 OVERLAY ファンクションのパラメータ

パラメータ	説明
overlay_str	target をオーバーレイするために使用するバイト列。
target	オーバーレイするバイト列。
pos	オーバーレイを開始する、target 内での位置（1 から番号付けされている）。
len	オーバーレイする target バイトの数。
pad	オーバーレイ len が overlay_str 長を超えた場合、または pos が target の長さを超えた場合に使用する pad バイト列。

デフォルトとオプション・パラメータ

表 98-25 OVERLAY ファンクションのオプション・パラメータのデフォルト

オプション・パラメータ	説明
pos	1
len	overlay_str の長さまで
pad	x'00'

戻り値

表 98-26 OVERLAY ファンクションの戻り値

戻り値	説明
RAW	指定したとおりにオーバーレイされた target の byte_string。

エラー

表 98-27 OVERLAY ファンクションのエラー

エラー	説明
VALUE_ERROR	次のいずれかです。 － overlay_str が NULL または長さ 0 です。 － target が不明か、未定義です。 － target の長さが、RAW の最大長を超えました。 － len < 0 － pos < 1

COPIES ファンクション

このファンクションは、r を n 回コピーして連結したものを戻します。

構文

```
UTL_RAW.COPIES (
   r IN RAW,
   n IN NUMBER)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(copies, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-28 COPIES ファンクションのパラメータ

パラメータ	説明
r	コピーする RAW。
n	RAW をコピーする回数（必ず正の数で指定）。

戻り値

このファンクションは、n 回コピーした RAW を戻します。

エラー

表 98-29 COPIES ファンクションのエラー

エラー	説明
VALUE_ERROR	次のいずれかです。 － r が不明、NULL または長さ 0 です。 － n < 1 － 結果の長さが、RAW の最大長を超えました。

XRANGE ファンクション

このファンクションは、値 start_byte で始まり値 end_byte で終わる、連続した有効な 1 バイト・コードをすべて含む RAW を戻します。start_byte が end_byte より大きい場合、結果バイトの連続は start_byte で始まり、x'FF' から x'00' に折り返して end_byte で終わります。start_byte と end_byte を指定する場合は、シングル・バイトの RAW である必要があります。

構文

```
UTL_RAW.XRANGE (
   start_byte IN RAW DEFAULT NULL,
   end_byte   IN RAW DEFAULT NULL)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(xrange, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-30 XRANGE ファンクションのパラメータ

パラメータ	説明
start_byte	戻される連続値の最初のバイト・コード値。
end_byte	戻される連続値の最後のバイト・コード値。

デフォルトとオプション・パラメータ

```
start_byte - x'00'
end_byte   - x'FF'
```

戻り値

表 98-31 XRANGE ファンクションの戻り値

戻り値	説明
RAW	連続した有効な 1 バイトの 16 進数コード。

REVERSE ファンクション

このファンクションは、RAW r のバイトの順序を、最後から最初に逆転させます。たとえば、x'0102F3' は x'F30201' になり、'xyz' は 'zyx' に逆転されます。結果は、入力 RAW と同じ長さになります。

構文

```
UTL_RAW.REVERSE (
   r IN RAW)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(reverse, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-32 REVERSE ファンクションのパラメータ

パラメータ	説明
r	逆転する RAW。

戻り値

表 98-33 REVERSE ファンクションの戻り値

戻り値	説明
RAW	r の逆転値を含んでいます。

エラー

表 98-34 REVERSE ファンクションのエラー

エラー	説明
VALUE_ERROR	r が NULL または長さ 0 です。

COMPARE ファンクション

このファンクションは、RAW r1 と RAW r2 を比較します。r1 と r2 の長さが異なる場合、短い方の RAW は、必要に応じて pad バイト列を使用して右側に拡張されます。

構文

```
UTL_RAW.COMPARE (
   r1  IN RAW,
   r2  IN RAW,
   pad IN RAW DEFAULT NULL)
  RETURN NUMBER;
```

プラグマ

```
pragma restrict_references(compare, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-35 COMPARE ファンクションのパラメータ

パラメータ	説明
r1	比較する 1 番目の RAW で、NULL または長さ 0 が可能。
r2	比較する 2 番目の RAW で、NULL または長さ 0 が可能。
pad	r1 または r2 の短い方を拡張するためのバイト列。

デフォルトとオプション・パラメータ

```
pad - x'00'
```

戻り値

表 98-36 COMPARE ファンクションの戻り値

戻り値	説明
NUMBER	RAW バイト列が両方とも NULL または等しい場合は、0（ゼロ）。 または、最初に不一致になったバイト位置（1 から番号付けされている）と等しい番号。

CONVERT ファンクション

このファンクションは、RAW r をキャラクタ・セット from_charset からキャラクタ・セット to_charset に変換し、結果の RAW を戻します。

from_charset と to_charset は両方とも、Oracle サーバーに定義されているサポート・キャラクタのセットである必要があります。

構文

```
UTL_RAW.CONVERT (
   r            IN RAW,
   to_charset   IN VARCHAR2,
   from_charset IN VARCHAR2)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(convert, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-37 CONVERT ファンクションのパラメータ

パラメータ	説明
r	変換する RAW バイト列。
to_charset	r が変換されるグローバリゼーション・サポート・キャラクタ・セットの名前。
from_charset	r が変換される前のグローバリゼーション・サポート・キャラクタ・セットの名前。

戻り値

表 98-38 CONVERT ファンクションの戻り値

戻り値	説明
RAW	指定したキャラクタ・セットに従って変換されたバイト列 r。

エラー

表 98-39 CONVERT ファンクションのエラー

エラー	説明
VALUE_ERROR	次のいずれかです。 − r が不明、NULL または長さ 0 です。 − from_charset または to_charset が不明、NULL または長さ 0 です。 − from_charset または to_charset の名前が無効か、またはサポートされていません。

BIT_AND ファンクション

このファンクションは、RAW r1 と RAW r2 の値でビット単位の論理演算 AND を実行し、AND 演算後の結果 RAW を戻します。

r1 と r2 の長さが異なる場合、AND 演算は、短い方の RAW の最終バイト後に終了し、長い方の RAW の未処理部分は部分結果に追加されます。したがって、結果の長さは、2 つの入力 RAW の長い方と同じ長さになります。

構文

```
UTL_RAW.BIT_AND (
   r1 IN RAW,
   r2 IN RAW)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(bit_and, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-40 BIT_AND ファンクションのパラメータ

パラメータ	説明
r1	r2 と AND 演算をする RAW。
r2	r1 と AND 演算をする RAW。

戻り値

表 98-41 BIT_AND ファンクションの戻り値

戻り値	説明
RAW	r1 と r2 の AND 演算結果を含んでいます。
NULL	入力パラメータ r1 または r2 が NULL の場合。

BIT_OR ファンクション

このファンクションは、RAW r1 と RAW r2 の値でビット単位の論理演算 OR を実行し、OR 演算後の結果 RAW を戻します。

r1 と r2 の長さが異なる場合、OR 演算は、短い方の RAW の最終バイト後に終了し、長い方の RAW の未処理部分は部分結果に追加されます。したがって、結果の長さは、2 つの入力 RAW の長い方と同じ長さになります。

構文

```
UTL_RAW.BIT_OR (
   r1 IN RAW,
   r2 IN RAW)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(bit_or, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-42 BIT_OR ファンクションのパラメータ

パラメータ	説明
r1	r2 と OR 演算をする RAW。
r2	r1 と OR 演算をする RAW。

戻り値

表 98-43 BIT_OR ファンクションの戻り値

戻り値	説明
RAW	r1 と r2 の OR 演算結果を含んでいます。
NULL	入力パラメータ r1 または r2 が NULL の場合。

BIT_XOR ファンクション

このファンクションは、RAW r1 と RAW r2 の値でビット単位の論理演算 XOR を実行し、XOR 演算後の結果 RAW を戻します。

r1 と r2 の長さが異なる場合、XOR 演算は、短い方の RAW の最終バイト後に終了し、長い方の RAW の未処理部分は部分結果に追加されます。したがって、結果の長さは、2 つの入力 RAW の長い方と同じ長さになります。

構文

```
UTL_RAW.BIT_XOR (
   r1 IN RAW,
   r2 IN RAW)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(bit_xor, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-44 BIT_XOR ファンクションのパラメータ

パラメータ	説明
r1	r2 と XOR 演算をする RAW。
r2	r1 と XOR 演算をする RAW。

戻り値

表 98-45 BIT_XOR ファンクションの戻り値

戻り値	説明
RAW	r1 と r2 の XOR 演算結果を含んでいます。
NULL	入力パラメータ r1 または r2 が NULL の場合。

BIT_COMPLEMENT ファンクション

このファンクションは、RAW r の値でビット単位の論理演算補数を実行し、補数演算後の結果 RAW を戻します。結果の長さは、入力 RAW r の長さと等しくなります。

構文

```
UTL_RAW.BIT_COMPLEMENT (
   r IN RAW)
  RETURN RAW;
```

プラグマ

```
pragma restrict_references(bit_complement, WNDS, RNDS, WNPS, RNPS);
```

パラメータ

表 98-46 BIT_COMPLEMENT ファンクションのパラメータ

パラメータ	説明
r	補数演算を実行する RAW。

戻り値

表 98-47 BIT_COMPLEMENT ファンクションの戻り値

戻り値	説明
RAW	r1 の補数演算結果を含んでいます。
NULL	入力パラメータ r が NULL の場合。

99

UTL_REF

Oracle9*i* は、ユーザー定義のコンポジット・タイプまたはオブジェクト・タイプをサポートします。オブジェクト・タイプのインスタンスをオブジェクトと呼びます。オブジェクト・タイプは、列のタイプとしてまたは表のタイプとして使用できます。

オブジェクト表では、表の各行にオブジェクトが格納されています。オブジェクト表にあるオブジェクトは、オブジェクト識別子で一意に識別できます。

参照はオブジェクトへの持続ポインタで、各参照にはオブジェクト識別子を含めることができます。参照は、オブジェクト・タイプの属性にしたり、表の列に格納することができます。参照を指定して、オブジェクトを取り出すことができます。

UTL_REF パッケージは、参照ベースの操作をサポートするための PL/SQL プロシージャを提供します。SQL と異なり、UTL_REF プロシージャでは、オブジェクト表名が不明でも汎用タイプ・メソッドを書き込むことができます。

この章では、次の項目について説明します。

- 要件
- UTL_REF のデータ・タイプ、例外およびセキュリティ
- UTL_REF サブプログラムの要約

要件

このパッケージを使用するには、プロシージャ・オプションが必要です。このパッケージは、SYS（SYSDBA で接続）によって作成される必要があります。このパッケージが提供する操作は、パッケージ所有者 SYS ではなく、現行のコール・ユーザーのもとで実行されます。

UTL_REF のデータ・タイプ、例外およびセキュリティ

データ・タイプ

オブジェクト・タイプは、ユーザーが定義するか、またはライブラリ・タイプとして提供されたコンポジット・データ・タイプです。次の構文を使用して、オブジェクト・タイプ employee_type を作成できます。

```
CREATE TYPE employee_type AS OBJECT (
   name    VARCHAR2(20),
   id      NUMBER,

member function GET_ID
    (name VARCHAR2)
   RETURN MEMBER);
```

オブジェクト・タイプ employee_type はユーザー定義型で、name と id の 2 つの属性およびメンバー・ファンクションの GET_ID() が含まれています。

次の SQL 構文を使用して、オブジェクト表を作成できます。

```
CREATE TABLE employee_table OF employee_type;
```

例外

UTL_REF ファンクションの実行中に、様々な理由で例外が戻る場合があります。たとえば、次の使用例では例外が発生します。

- 選択したオブジェクトが存在しません。これには、次のいずれかの理由が考えられます。
  1. オブジェクトが削除されたか、または指定した参照が無効です。
  2. オブジェクト表が削除されたか、または存在しません。
- シリアル化可能トランザクションで、オブジェクトを変更またはロックできません。オブジェクトは、シリアル化可能トランザクションの開始後に、別のトランザクションで変更されました。
- オブジェクトを選択または変更する権限がありません。UTL_REF サブプログラムのコール元は、選択または変更するオブジェクトに対する適切な権限が必要です。

表 99-1 UTL_REF の例外

例外	説明
errnum == 942	権限が不十分です。
errnum == 1031	権限が不十分です。
errnum == 8177	シリアル化可能トランザクションの場合は、シリアル化できません。
errnum == 60	デッドロックが検出されました。
errnum == 1403	データが見つかりません（REF が NULL である場合など）。

UTL_REF パッケージは、名前の付いた例外を定義しません。特定の例外を捕捉して適切に処理するために、ブロックを処理する例外を定義できます。

セキュリティ

UTL_REF パッケージは、サーバー上のストアド PL/SQL プロシージャまたはパッケージとクライアント側の PL/SQL コードからも同様に使用できます。

サーバー上の PL/SQL プロシージャまたはパッケージから起動した場合、UTL_REF は、REF が指すオブジェクトに対する適切なアクセス権限が実行者にあることを検証します。

注意： これは、定義者の権限で操作する、サーバー上の PL/SQL パッケージまたはプロシージャとは対照的で、パッケージ所有者には、必要な操作を実行するための適切な権限が必要です。

したがって、UTL_REF がユーザーの SYS 権限で定義された場合、ユーザー A が参照からオブジェクトを選択するために UTL_REF.SELECT を起動すると、ユーザー A（実行者）は、権限をチェックする必要があります。

UTL_REF は、クライアント側 PL/SQL コードから起動すると、PL/SQL が実行されているクライアント・セッションの権限で操作が行われます。

UTL_REF サブプログラムの要約

表 99-2 UTL_REF サブプログラム

サブプログラム	説明
「SELECT_OBJECT プロシージャ」 99-4 ページ	参照を指定してオブジェクトを選択します。
「LOCK_OBJECT プロシージャ」 99-5 ページ	参照を指定してオブジェクトをロックします。
「UPDATE_OBJECT プロシージャ」 99-6 ページ	参照を指定してオブジェクトを更新します。
「DELETE_OBJECT プロシージャ」 99-7 ページ	参照を指定してオブジェクトを削除します。

SELECT_OBJECT プロシージャ

このプロシージャは、参照を指定してオブジェクトを選択します。選択されたオブジェクトはデータベースから取り出され、その値は PL/SQL の変数 'object' に入れられます。このサブプログラムの意味は、次の SQL 文に似ています。

```
SELECT VALUE(t)
INTO object
FROM object_table t
WHERE REF(t) = reference;
```

前述の SQL 文と異なり、このサブプログラムでは、オブジェクトが常駐しているオブジェクト表名を指定する必要はありません。

構文

```
UTL_REF.SELECT_OBJECT (
   reference IN REF "<typename>",
   object    IN OUT "<typename>");
```

パラメータ

表 99-3 SELECT_OBJECT プロシージャのパラメータ

パラメータ	説明
reference	選択または取り出すオブジェクトへの参照。
object	選択したオブジェクトを格納する PL/SQL 変数。この変数は、参照されたオブジェクトと同じオブジェクト・タイプである必要があります。

例外

発生する可能性があります。

LOCK_OBJECT プロシージャ

このプロシージャは、参照を指定してオブジェクトをロックします。さらに、このプロシージャでは、プログラムによって、ロックされたオブジェクトを選択できます。このサブプログラムの意味は、次の SQL 文に似ています。

```
SELECT VALUE(t)
  INTO object
  FROM object_table t
  WHERE REF(t) = reference
  FOR UPDATE;
```

前述の SQL 文と異なり、このサブプログラムでは、オブジェクトが常駐しているオブジェクト表名を指定する必要はありません。オブジェクトの更新または削除前に、オブジェクトをロックする必要はありません。

構文

```
UTL_REF.LOCK_OBJECT (
   reference IN REF "<typename>");

UTL_REF.LOCK_OBJECT (
   reference IN REF "<typename>",
   object    IN OUT "<typename>");
```

パラメータ

表 99-4 LOCK_OBJECT プロシージャのパラメータ

パラメータ	説明
reference	ロックするオブジェクトの参照。
object	ロックされたオブジェクトを格納する PL/SQL 変数。この変数は、ロックされたオブジェクトと同じオブジェクト・タイプである必要があります。

例外

発生する可能性があります。

UPDATE_OBJECT プロシージャ

このプロシージャは、参照を指定してオブジェクトを更新します。参照されたオブジェクトは、PL/SQL 変数 'object' に含まれている値で更新されます。このサブプログラムの意味は、次の SQL 文に似ています。

```
UPDATE object_table t
SET VALUE(t) = object
WHERE REF(t) = reference;
```

前述の SQL 文と異なり、このサブプログラムでは、オブジェクトが常駐しているオブジェクト表名を指定する必要はありません。

構文

```
UTL_REF.UPDATE_OBJECT (
   reference IN REF "<typename>",
   object    IN     "<typename>");
```

パラメータ

表 99-5 UPDATE_OBJECT プロシージャのパラメータ

パラメータ	説明
reference	更新するオブジェクトの参照。
object	オブジェクトの新規の値を含める PL/SQL 変数。この変数は、更新されたオブジェクトと同じオブジェクト・タイプである必要があります。

例外

発生する可能性があります。

DELETE_OBJECT プロシージャ

このプロシージャは、参照を指定してオブジェクトを削除します。このサブプログラムの意味は、次の SQL 文に似ています。

```
DELETE FROM object_table
WHERE REF(t) = reference;
```

前述の SQL 文と異なり、このサブプログラムでは、オブジェクトが常駐しているオブジェクト表名を指定する必要はありません。

構文

```
UTL_REF.DELETE_OBJECT (
   reference IN REF "<typename>");
```

パラメータ

表 99-6 DELETE_OBJECT プロシージャのパラメータ

パラメータ	説明
reference	削除するオブジェクトの参照。

例外

発生する可能性があります。

例

この例は、次のシナリオを実行するための UTL_REF パッケージの使用方法を示しています。会社の従業員が自宅住所の変更を上司に連絡するとします。

... Address_t の宣言など ...

```
CREATE OR REPLACE TYPE Person_t (
   name    VARCHAR2(64),
   gender  CHAR(1),
   address Address_t,
   MEMBER PROCEDURE setAddress(addr IN Address_t)
);

CREATE OR REPLACE TYPE BODY Person_t (
   MEMBER PROCEDURE setAddress(addr IN Address_t) IS
   BEGIN
      address := addr;
   END;
);
```

```
CREATE OR REPLACE TYPE Employee_t (
```

Person_t で、REF を使用した Person_t への継承の実装と setAddress の委任をシミュレーションします。

```
   thePerson  REF Person_t,
   empno      NUMBER(5),
   deptREF    Department_t,
   mgrREF     Employee_t,
   reminders  StringArray_t,
   MEMBER PROCEDURE setAddress(addr IN Address_t),
   MEMBER procedure addReminder(reminder VARCHAR2);
);

CREATE TYPE BODY Employee_t (
   MEMBER PROCEDURE setAddress(addr IN Address_t) IS
      myMgr Employee_t;
      meAsPerson Person_t;
   BEGIN
```

責任を thePerson に委任して、アドレスを更新します。個人オブジェクトを参照からロックして、それをまた選択します。

```
      UTL_REF.LOCK_OBJECT(thePerson, meAsPerson);
      meAsPerson.setAddress(addr);
```

thePerson に委任します。

```
        UTL_REF.UPDATE_OBJECT(thePerson, meAsPerson);
        if mgr is NOT NULL THEN
```

マネージャに覚書きを渡します。

```
         UTL_REF.LOCK_OBJECT(mgr);
         UTL_REF.SELECT_OBJECT(mgr, myMgr);
         myMgr.addReminder
         ('Update address in the employee directory for' ||
         thePerson.name || ', new address: ' || addr.asString);
         UTL_REF.UPDATE_OBJECT(mgr, myMgr);
      END IF;
   EXCEPTION
      WHEN OTHERS THEN
      errnum := SQLCODE;
      errmsg := SUBSTR(SQLERRM, 1, 200);
```

100

UTL_SMTP

UTL_SMTP は、Simple Mail Transfer Protocol（SMTP）を使用した電子メールの送信向けに設計されています。SMTP を使用して電子メールを送信するように、SMTP サーバーをメール・クライアントに実装する機能は備えていません。

SMTP パッケージへのインタフェースの多くが、ファンクションおよびプロシージャとして記述されています。ファンクション形式の場合、クライアントで処理するための応答はサーバーから戻されます。プロシージャ形式の場合、応答は廃棄されますが、一時的なエラー（400 番台の応答コード）または永続的なエラー（500 番台の応答コード）を示す応答のときには例外が発生します。

オリジナルの SMTP プロトコルによる通信には、7 ビット ASCII が使用されることに注意してください。UTL_SMTP を使用すると、すべてのテキスト・データ（つまり、VARCHAR2 タイプデータ）は、サーバーに送信される前に US7ASCII に変換されます。SMTP 拡張 8BITMIME [RFC1652] をサポートしている SMTP サーバーの実装の一部は、クライアントとサーバーの間の完全 8 ビット通信をサポートしています。

DATA コマンドの本体は完全 8 ビットで転送できますが、それ以外の SMTP のコマンドと応答は 7 ビットです。ターゲットの SMTP サーバーが 8BITMIME 拡張要素をサポートしている場合、マルチバイト・データベースのユーザーは、非 US7ASCII のマルチバイト VARCHAR2 データを RAW に変換し、write_raw_data() API を使用して 8 ビット MIME のエンコーディングを使用したマルチバイト・データを送信できます。

UTL_SMTP では、RFC821 に指定されている SMTP 通信が提供されますが、メッセージの内容を RFC822 に従って書式設定するための API（電子メールの件名の設定など）は提供していません。メッセージの適切な書式設定は、ユーザー各自で行ってください。

この章では、次の項目について説明します。

注意： RFC ドキュメントとは Request for Comments ドキュメントのことで、インターネット上の公開閲覧に関する規格を提案する文書です。実際の RFC ドキュメントは、次の URL を参照してください。

```
http://www.ietf.org/rfc/
```

例外、制限事項および応答コード

例外

表 100-1 に、UTL_SMTP パッケージの API が呼び出す可能性のある例外を示します。ネットワーク・エラーは、応答コード 421（サービスは使用不可）になります。

表 100-1 UTL_SMTP の例外

例外	説明
INVALID_OPERATION	無効な操作が行われたときに呼び出されます。つまり、open_data() の後に write_data()、write_raw_data() または close_data() 以外の API をコールした場合、または open_data() をコールせずに、write_data()、write_raw_data() または close_data() をコールした場合です。
TRANSIENT_ERROR	400 番台の応答コードを受け取った場合に呼び出されます。
PERMANENT_ERROR	500 番台の応答コードを受け取った場合に呼び出されます。

制限事項

API による制限または範囲チェックはありません。ただし、SMTP の各種要素については、次のサイズ制限に注意する必要があります。制限を超えるデータが送信されると、サーバーからエラーが戻ります。

表 100-2 SMTP のサイズ制限

要素	サイズ制限
ユーザー	ユーザー名の最大合計長は、64 文字です。
ドメイン	ドメイン名またはドメイン番号の最大合計長は、64 文字です。
パス	reverse-path または forward-path の最大合計長は 256 文字です（句読点と要素のセパレータも含む）。
コマンドライン	コマンド自体と <CRLF> も含めたコマンドラインの最大合計長は、512 文字です。
応答行	応答コードと <CRLF> も含めた応答行の最大合計長は、512 文字です。
テキスト行	<CRLF> を含めたテキスト行の最大合計長は、1000 文字です（ただし、透過性に必要な先行ドットはカウントされません）。
受信バッファ	バッファリングを要する受信者の最大合計数は、100 です。

応答コード

次のリストに SMTP 応答コードを示します。

表 100-3 SMTP 応答コード

応答コード	意味
211	システム・ステータスまたはシステム・ヘルプの応答です。
214	ヘルプ・メッセージ（使用方法や特定の非標準コマンドの意味に関する情報で、この応答は人間であるユーザーにのみ有用です）。
220	<domain> サービスは準備完了です。
221	<domain> サービスは送信チャネルをクローズしています。
250	要求したメール・アクションは正常で、完了しました。
251	ユーザーがローカルではありません。<forward-path> にフォワードします。
252	OK。ノード <node> に対する保留中のメッセージが開始されました。ユーザーを VRFY できません（情報がローカルにないなど）が、このユーザーへのメッセージを送信しようとします。
253	OK。ノード <node> に対する保留中のメッセージ <messages> が開始されました。
354	メール入力を開始し、<CRLF>.<CRLF> で終了します。
355	オクテット・オフセットは、トランザクション・オフセットです。
421	<domain> サービスが利用不可のため、送信チャネルをクローズしています（サービスのシャットダウンが必要な場合には、すべてのコマンドに対してこの応答が使用される可能性があります）。
450	要求したメール・アクションが実行されていません。メールボックスは利用不可です（メールボックス・ビジーなど）。
451	要求したアクションが異常終了しました。処理中にローカル・エラーが発生しました。
452	要求したアクションが実行されていません。システム記憶域不足です。
453	電子メールがありません。
454	一時的に TLS を使用できません。要求した認証メカニズムには暗号化が必要です。
458	ノード <node> に対してメッセージをキューできません。
459	ノード <node> は許可されていません。
500	構文エラーのため、コマンドを認識できません（コマンドラインが長すぎるなどのエラーも含まれます）。

表 100-3 SMTP 応答コード（続き）

応答コード	意味
501	パラメータまたは引数に構文エラーがあります。
502	コマンドが実装されていません。
503	コマンドの順序が不正です。
504	コマンド・パラメータが実装されていません。
521	<Machine> がメールを受け入れません。
530	STARTTLS コマンドを最初に発行する必要があります。要求した認証メカニズムには暗号化が必要です。
534	認証メカニズムが脆弱です。
538	要求した認証メカニズムには暗号化が必要です。
550	要求したアクションが実行されていません。メールボックスは利用不可です（メールボックスが見つからない、アクセスできないなど）。
551	ユーザーがローカルではありません。<forward-path> を試行してください。
552	要求したメール・アクションが異常終了しました。記憶域の割当て超過です。
553	要求したアクションが実行されていません。メールボックス名が使用不可です（メールボックスの構文が正しくないなど）。
554	トランザクションに失敗しました。

UTL_SMTP サブプログラムの要約

表 100-4 UTL_SMTP サブプログラム

サブプログラム	説明
「connection レコード・タイプ」100-7 ページ	SMTP 接続を示す PL/SQL レコード・タイプです。
「reply、replies レコード・タイプ」100-8 ページ	SMTP 応答行を示す PL/SQL レコード・タイプです。
「open_connection ファンクション」100-8 ページ	SMTP サーバーへの接続をオープンします。
「command()、command_replies() ファンクション」100-10 ページ	一般的な SMTP コマンドを実行します。
「helo ファンクション」100-11 ページ	接続後に SMTP サーバーとの初期ハンドシェイクを実行します。
「ehlo ファンクション」100-12 ページ	接続後、戻された詳細な情報を使用して SMTP サーバーとの初期ハンドシェイクを実行します。
「mail ファンクション」100-13 ページ	サーバーとのメール・トランザクションを開始します。宛先はメールボックスです。
「rcpt ファンクション」100-14 ページ	電子メール・メッセージの受信者を指定します。
「data ファンクション」100-15 ページ	電子メール・メッセージの本文を指定します。
「open_data()、write_data()、write_raw_data()、close_data() ファンクション」100-16 ページ	data() API への強力なファイングレイン制御を提供しています。
「rset ファンクション」100-18 ページ	現行のメール・トランザクションを異常終了します。
「vrfy ファンクション」100-18 ページ	宛先の電子メール・アドレスの妥当性を検証します。
「noop() ファンクション」100-19 ページ	NULL のコマンドです。
「quit ファンクション」100-20 ページ	SMTP セッションを終了し、サーバーとの接続を切断します。

connection レコード・タイプ

SMTP 接続を示す PL/SQL レコード・タイプです。

構文

```
TYPE connection IS RECORD (
    host              VARCHAR2(255),      -- remote host name
    port              PLS_INTEGER,        -- remote port number
    tx_timeout        PLS_INTEGER,        -- Transfer time-out (in seconds)
    private_tcp_con   utl_tcp.connection, -- private, for implementation use
    private_state     PLS_INTEGER         -- private, for implementation use
);
```

フィールド

表 100-5 connection レコード・タイプのフィールド

フィールド	説明
host	接続が確立したときのリモート・ホストの名前。接続が確立されない場合は NULL です。
port	接続したリモート SMTP サーバーのポート番号。接続が確立されない場合は NULL です。
tx_timeout	この接続で読込みまたは書込み操作を中止するまでの UTL_SMTP パッケージの待機時間（秒）。読込み操作の場合、読み込めるデータがない場合に操作が即座に中止されます。書込み操作の場合、出力バッファがフルで、ブロックされずにネットワークに送信できるデータがない場合に操作が中止されます。0（ゼロ）は、まったく待機しないことを示します。NULL は、無期限に待機することを示します。
private_tcp_con	実装目的でのみ使用されるプライベートのパラメータ。このフィールドは変更しないでください。
private_state	実装目的でのみ使用されるプライベートのパラメータ。このフィールドは変更しないでください。

使用上の注意

接続レコードの読取り専用フィールドは、open_connection() を使用して接続に成功した後、SMTP 接続の情報を戻すために使用されます。これらのフィールドを変更しても、接続には影響ありません。private_XXX フィールドは、実装目的にのみ使用されます。これらのフィールドは変更しないでください。

reply、replies レコード・タイプ

SMTP 応答行を示す PL/SQL レコード・タイプです。各 SMTP 応答行は、応答コードとそれに続くテキスト・メッセージで構成されています。大半の SMTP コマンドに対しては単一の応答行ですが、一部の SMTP コマンドに対しては複数の応答行となります。このような場合は、複数の応答行を示すために応答レコードの PL/SQL 表が使用されます。

構文

```
TYPE reply IS RECORD (
  code    PLS_INTEGER,       -- 3-digit reply code
  text    VARCHAR2(508)      -- text message
);
TYPE replies IS TABLE OF reply INDEX BY BINARY_INTEGER;   -- multiple reply lines
```

フィールド

表 100-6 reply、replies レコード・タイプのフィールド

フィールド	説明
code	3 桁の応答コード。
text	応答のテキスト・メッセージ。

open_connection ファンクション

このファンクションは、SMTP サーバーへの接続をオープンします。

構文

```
UTL_SMTP.OPEN_CONNECTION (
   host        IN  VARCHAR2,
   port        IN  PLS_INTEGER DEFAULT 25,
   c           OUT connection,
   tx_timeout  IN  PLS_INTEGER DEFAULT NULL)
RETURN reply;
UTL_SMTP.OPEN_+CONNECTION (
   host        IN  VARCHAR2,
   port        IN  PLS_INTEGER DEFAULT 25,
   tx_timeout  IN  PLS_INTEGER DEFAULT NULL)
RETURN connection;
```

パラメータ

表 100-7 open_connection ファンクションのパラメータ

パラメータ	説明
host（IN）	SMTP サーバーのホストの名前。
port（IN）	SMTP サーバーがリスニングするポート番号（通常は 25）。
tx_timeout （IN）	この接続で読込みまたは書込み操作を中止するまでの UTL_SMTP パッケージの待機時間（秒）。読込み操作の場合、読み込めるデータがない場合に操作が即座に中止されます。書込み操作の場合、出力バッファがフルで、ブロックされずにネットワークに送信できるデータがない場合に操作が中止されます。0（ゼロ）は、まったく待機しないことを示します。NULL は、無期限に待機することを示します。

使用上の注意

サーバーからの応答は、ステータス・コード 220 で開始するメッセージになります。

utl_smtp.connection レコードを戻す open_connection() API のバージョンは、実際には、接続が初めて確立された時点で SMTP サーバーが戻す応答コードをチェックする open_connection プロシージャ・バージョンです。

書込み操作のタイムアウト機能は、このパッケージの現行のリリースではサポートされていません。

command()、command_replies() ファンクション

一般的な SMTP コマンドを実行します。

構文

```
UTL_SMTP.COMMAND (
   c    IN connection,
   cmd  IN VARCHAR2,
   arg  IN VARCHAR2 DEFAULT NULL)
RETURN reply;
UTL_SMTP.COMMAND (
   c    IN connection,
   cmd  IN VARCHAR2,
   arg  IN ARCHAR2 DEFAULT NULL);
UTL_SMTP.COMMAND_REPLIES (
   c    IN connection,
   cmd  IN VARCHAR2,
   arg  IN VARCHAR2 DEFAULT NULL)
RETURN replies;
```

パラメータ

表 100-8 command ()、command_replies () ファンクションのパラメータ

パラメータ	説明
c（IN）	SMTP 接続。
cmd（IN）	サーバーに送信する SMTP コマンド。
arg（IN）	SMTP 引数へのオプション引数。cmd と arg の間に空白が挿入されます。

使用上の注意

これらの API は、一般的な SMTP コマンドを起動します。command() は、単一の応答行が予想される場合にのみ使用します。command_replies() は、応答行が複数になる（つまり、EXPN や HELP）場合に使用します。

command() は、SMTP サーバーからの応答が複数行ある場合、最終応答行のみ戻します。

helo ファンクション

このファンクションは、接続後に SMTP サーバーとの初期ハンドシェイクを実行します。

構文

```
UTL_SMTP.HELO (
   c  IN NOCOPY connection, domain IN NOCOPY)
RETURN reply;
UTL_SMTP.HELO (
   c  IN NOCOPY connection, domain IN NOCOPY);
```

パラメータ

表 100-9 helo ファンクションのパラメータ

パラメータ	説明
c (IN NOCOPY)	SMTP 接続。
domain (IN NOCOPY)	ローカル（送信側）ホストのドメイン名。識別の目的で使用されます。

使用上の注意

RFC821 では、接続した後でクライアントはサーバーに対してクライアント自体を識別する必要があると指定されています。このルーチンは、その識別を実行します。このルーチンをコールする前に、open_connection() をコールして接続をオープンしておく必要があります。

サーバーからの応答は、ステータス・コード 250 で開始するメッセージになります。

関連ファンクション

ehlo()

ehlo ファンクション

このファンクションは、接続後、戻された詳細な情報を使用して SMTP サーバーとの初期ハンドシェイクを実行します。

構文

```
UTL_SMTP.EHLO (
   c       IN OUT NOCOPY connection,
   domain  IN NOCOPY)
RETURN replies;
UTL_SMTP.EHLO (
   c       IN OUT NOCOPY connection,
   domain  IN NOCOPY);
```

パラメータ

表 100-10 ehlo ファンクションのパラメータ

パラメータ	説明
c (IN NOCOPY)	SMTP 接続。
domain (IN NOCOPY)	ローカル（送信側）ホストのドメイン名。識別の目的で使用されます。

使用上の注意

ehlo() のインタフェースは、サーバーがその構成に関する詳細な情報を戻せることを除くと helo() と同じです。[RFC1869] には、戻される情報の書式が指定されています。書式は、PL/SQL アプリケーションでこのコールのファンクション形式を使用して取り出すことができます。helo() との互換性のために、サーバーが戻すテキストの各行はステータス・コード 250 で開始されます。

関連ファンクション

helo()

mail ファンクション

このファンクションは、サーバーとのメール・トランザクションを開始します。宛先はメールボックスです。

構文

```
UTL_SMTP.MAIL (
   c           IN OUT NOCOPY connection,
   sender      IN OUT NOCOPY,
   parameters  IN OUT NOCOPY)
RETURN reply;
UTL_SMTP.MAIL (
   c           IN OUT NOCOPY connection,
   sender      IN OUT NOCOPY,
   parameters  IN OUT NOCOPY);
```

パラメータ

表 100-11 Mail ファンクションのパラメータ

パラメータ	説明
c (IN NOCOPY)	SMTP 接続。
sender (IN OUT NOCOPY)	メッセージを送信するユーザーの電子メール・アドレス。
parameters (IN OUT NOCOPY)	[RFC1869] のセクション 6 に定義されている MAIL コマンドへの追加パラメータ。XXX=XXX (XXX=XXX ....) の形式に従ってください。

使用上の注意

このコマンドはメッセージを送信しません。準備を開始するのみです。トランザクションを完了するには、このコマンドの後に rcpt() と data() へのコールが必要です。SMTP サーバーへの接続はオープン状態で、helo() または ehlo() コマンドはすでに送信されている必要があります。

サーバーからの応答は、ステータス・コード 250 で開始するメッセージになります。

rcpt ファンクション

このファンクションは、電子メール・メッセージの受信者を指定します。

構文

```
UTL_SMTP.RCPT (
   c           IN OUT NOCOPY connection,
   recipient   IN OUT NOCOPY,
   parameters  IN OUT NOCOPY)
RETURN reply;
UTL_SMTP.RCPT (
   c           IN OUT NOCOPY connection
   recipient   IN OUT NOCOPY,
   parameters  IN OUT NOCOPY);
```

表 100-12 rcpt ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	SMTP 接続。
recipient （IN OUT NOCOPY）	メッセージの送信先ユーザーの電子メール・アドレス。
parameters（IN OUT NOCOPY）	[RFC1869] のセクション 6 に定義されている RCPT コマンドへの追加パラメータ。XXX=XXX (XXX=XXX ....) の形式に従ってください。

使用上の注意

複数の受信者にメッセージを送信するには、このルーチンを複数回コールします。各起動で 1 つの電子メール・アドレスへの配信がスケジュールされます。メッセージ・トランザクションは、先行する mail() へのコールによって開始されており、メール・サーバーへの接続はすでにオープン状態にあり、先行する open_connection() および helo() または ehlo() へのコールによってそれぞれ初期化されている必要があります。

サーバーからの応答は、ステータス・コード 250 または 251 で開始するメッセージになります。

data ファンクション

このファンクションは、電子メール・メッセージの本文を指定します。

構文

```
UTL_SMTP.DATA (
   c     IN OUT NOCOPY connection
   body  IN OUT NOCOPY)
RETURN reply;
UTL_SMTP.DATA (
   c     IN OUT NOCOPY connection
   body  IN OUT NOCOPY);
```

パラメータ

表 100-13 data ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	SMTP 接続。
body（IN OUT NOCOPY）	送信するメッセージのヘッダーを含めたテキスト（[RFC822] 形式）。

使用上の注意

body パラメータの内容が MIME（RFC822）の仕様に準拠していることをアプリケーションにより確認する必要があります。data() ルーチンは、RFC821 の要求に従って、<CR><LF>.<CR><LF> の順序で（行の開始はシングル・ピリオド）メッセージを終了します。また、body の <CR><LF>.<CR><LF>（シングル・ピリオド）の順序を <CR><LF>..<CR><LF>（ダブル・ピリオド）に変換します。この変換によって、RFC821 のセクション 4.5.2 に記述されている透過性が提供されます。

data() は、必ず open_connection()、helo() または ehlo()、mail() および rcpt() がコールされた後にコールしてください。このルーチンがコールされるときは、SMTP サーバーへの接続がオープン状態で、メール・トランザクションがアクティブ状態である必要があります。

サーバーからの応答は、ステータス・コード 250 で開始するメッセージになります。初期 DATA コマンドから受信した 354 という応答は、コール元に戻されません。

open_data()、write_data()、write_raw_data()、close_data() ファンクション

これら 3 つの API は、data() API、つまり、SMTP DATA 操作により強力なファイングレイン制御を提供しています。open_data() は、DATA コマンドを送信します。その後、write_data() および write_raw_data() で電子メール・メッセージの一部を書き込みます。write_data() および write_raw_data() を繰り返しコールして電子メール・メッセージにデータを追加します。close_data() コールは、<CR><LF>.<CR><LF>（行の開始はシングル・ピリオド）の順序で送信することにより、電子メール・メッセージを終了します。

構文

```
UTL_SMTP.OPEN_DATA (
   c     IN OUT NOCOPY connection)
RETURN reply;
UTL_SMTP.OPEN_DATA (
   c     IN OUT NOCOPY connection);
UTL_SMTP.WRITE_DATA (
   c     IN OUT NOCOPY connection,
   data  IN OUT NOCOPY);
UTL_SMTP.WRITE_RAW_DATA (
   c     IN OUT NOCOPY connection
   data  IN OUT NOCOPY);
UTL_SMTP.CLOSE_DATA (
   c     IN OUT NOCOPY connection)
RETURN reply;
UTL_SMTP.CLOSE_DATA (
   c     IN OUT NOCOPY connection);
```

パラメータ

表 100-14 open_data()、write_data()、write_raw_data()、close_data() ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	SMTP 接続。
data （IN OUT NOCOPY）	送信するメッセージのヘッダーを含めたテキストの一部（[RFC822] フォーマットで）。

使用上の注意

open_data()、write_data()、write_raw_data() および close_data() へのコールは、正しい順序で実行する必要があります。最初に、SMTP サーバーに DATA コマンドを送信するために、open_data() をコールすると、次に、実際のデータを送信するために、write_data() または write_raw_data() を繰り返しコールできます。データは、close_data() をコールして終了します。open_data() がコールされた後にコールできる API は、write_data()、write_raw_data() または close_data() のみです。他の API へのコールは、INVALID_OPERATION 例外の発生原因となります。

body パラメータの内容が MIME（RFC822）の仕様に準拠していることをアプリケーションにより確認する必要があります。data() ルーチンは、RFC821 の要求に従って、<CR><LF>.<CR><LF> の順序で（行の開始はシングル・ピリオド）メッセージを終了します。また、body の <CR><LF>.<CR><LF>（シングル・ピリオド）の順序を <CR><LF>..<CR><LF>（ダブル・ピリオド）に変換します。この変換によって、RFC821 のセクション 4.5.2 に記述されている透過性が提供されます。

この変換は完全でないことに注意してください。このコード・フラグメントを次に示します。

```
utl_smtp.write_data('some message.' || chr(13) || chr(10));
utl_smtp.write_data('.' || chr(13) || chr(10));
```

順序 <CR><LF>.<CR><LF> は、write_data() への 2 回のコールに分割されるため、write_data() の実装によって、データの終了順序は検出されません。したがって、変換も実行されません。このような状態は、ユーザーの責任で処理する必要があります。処理が行われない場合、メッセージ・データの終端が不完全となる可能性があります。

XXX_data() は、必ず open_connection()、helo() または ehlo()、mail() および rcpt() がコールされた後にコールしてください。このルーチンがコールされるときは、SMTP サーバーへの接続がオープン状態で、メール・トランザクションがアクティブ状態である必要があります。

SMTP サーバーからの応答は、close_data() へのコール中にデータの終了が送信されるまで行われないため、write_data() のファンクション形式は存在しないことに注意してください。

write_data() API を使用して送信するテキスト（VARCHAR2）データは、送信前に US7ASCII に変換されます。テキストにマルチバイト・キャラクタが含まれている場合は、テキスト内の US7ASCII に変換できない各マルチバイト・キャラクタが '?' 文字に置き換えられます。EHLO() API を使用して SMTP サーバーとの間で 8BITMIME 拡張要素を処理する場合は、最初に UTL_RAW パッケージでテキストを RAW に変換してから、write_raw_data() を使用して RAW データを送信することによって、マルチバイトの VARCHAR2 データを送信できます。

rset ファンクション

このファンクションは、現行のメール・トランザクションを異常終了します。

構文

```
UTL_SMTP.RSET (
   c  IN OUT NOCOPY connection)
RETURN reply;
UTL_SMTP.RSET (
   c  IN OUT NOCOPY connection);
```

パラメータ

表 100-15 rset ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	SMTP 接続。

使用上の注意

このコマンドによって、クライアントは構成処理中のメール・メッセージを放棄できます。メールは送信されません。クライアントは、SMTP サーバーへの接続が open_connection() でオープンされた後は、いつでも rset() をコールできます。サーバーは、RSET に対してステータス・コード 250 で開始するメッセージで常に応答します。

関連ファンクション

quit()

vrfy ファンクション

このファンクションは、宛先の電子メール・アドレスの妥当性を検証します。

構文

```
UTL_SMTP.VRFY (
   c          IN OUT NOCOPY connection
   recipient  IN OUT NOCOPY)
RETURN reply;
```

パラメータ

表 100-16 vrfy ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	SMTP 接続。
recipient （IN OUT NOCOPY）	検証される電子メール・アドレス。

使用上の注意

サーバーは、宛先アドレス recipient の解決を試みます。解決した場合、サーバーは受信者のフルネームと完全修飾メールボックス・パスを戻します。この要求の実行前には、open_connection() および helo() または ehlo() を介して、サーバーへの接続が確立されている必要があります。

検証が正常に終了した場合は、ステータス・コード 250 または 251 で開始する 1 行以上の行が戻されます。

関連ファンクション

expn()

noop() ファンクション

NULL のコマンドです。

構文

```
UTL_SMTP.NOOP (
   c  IN OUT NOCOPY connection)
RETURN VARCHAR2;
UTL_SMTP.NOOP (
   c  IN OUT NOCOPY connection);
```

パラメータ

表 100-17 noop ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	SMTP 接続。

使用上の注意

このコマンドは、サーバーから正常な応答を引き出すことにのみ有効です。open_connection() でサーバーへの接続を確立した後は、いつでもこのコマンドを発行できます。noop() コマンドは、サーバーが接続され、正しくリスニングが行われていることを検証するために使用します。

このコマンドの応答は、常にステータス・コード 250 で開始する単一の行になります。

quit ファンクション

このファンクションは、SMTP セッションを終了し、サーバーとの接続を切断します。

構文

```
UTL_SMTP.QUIT (
   c  IN OUT NOCOPY connection)
RETURN VARCHAR2;
```

パラメータ

表 100-18 quit ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	SMTP 接続。

使用上の注意

quit() コマンドは、セッションを終了するというクライアントの意図を SMTP サーバーに伝えます。伝えた後に、open_connection() で確立した接続をクローズします。この open_connection() は、このコマンドの実行前にコールされている必要があります。quit() の発行時に、メール・トランザクションが進行中の場合は、rset() と同様に放棄されます。

このコマンドのファンクション形式は、正常終了時にステータス・コード 221 で開始する単一の行を戻します。SMTP サーバーへの接続は、どのような場合にもクローズされます。c の remote_host および remote_port フィールドはリセットされます。

関連ファンクション

rset()

例

次の例では、アプリケーションで UTL_SMTP を使用して電子メールを送信する方法について示します。ポート 25 で SMTP サーバーに接続し、簡単なテキスト・メッセージを送信します。

```
DECLARE
  c utl_smtp.connection;

  PROCEDURE send_header(name IN VARCHAR2, header IN VARCHAR2) AS
  BEGIN
    utl_smtp.write_data(c, name || ': ' || header || utl_tcp.CRLF);
  END;

BEGIN
  c := utl_smtp.open_connection('smtp-server.acme.com');
  utl_smtp.helo(c, 'foo.com');
  utl_smtp.mail(c, 'sender@foo.com');
  utl_smtp.rcpt(c, 'recipient@foo.com');
  utl_smtp.open_data(c);
  send_header('From',    '"Sender" <sender@foo.com>');
  send_header('To',      '"Recipient" <recipient@foo.com>');
  send_header('Subject', 'Hello');
  utl_smtp.write_data(c, utl_tcp.CRLF || 'Hello, world!');
  utl_smtp.close_data(c);
  utl_smtp.quit(c);
EXCEPTION
  WHEN utl_smtp.transient_error OR utl_smtp.permanent_error THEN
    BEGIN
      utl_smtp.quit(c);
    EXCEPTION
      WHEN utl_smtp.transient_error OR utl_smtp.permanent_error THEN
        NULL; -- When the SMTP server is down or unavailable, we don't have
              -- a connection to the server. The quit call will raise an
              -- exception that we can ignore.
    END;
    raise_application_error(-20000,
      'Failed to send mail due to the following error: ' || sqlerrm);
END;
```

101

UTL_TCP

UTL_TCP パッケージおよびそのプロシージャとファンクションにより、TCP/IP を使用して PL/SQL アプリケーションと外部の TCP/IP ベースのサーバーを通信させることができます。インターネット・アプリケーションのプロトコルは多くが TCP/IP に基づいているため、このパッケージはインターネット・プロトコルと電子メールを使用する PL/SQL アプリケーションで使用すると便利です。

UTL_TCP パッケージは、TCP/IP クライアント側のアクセス機能を PL/SQL に提供します。このパッケージで提供される API で使用できるのは、PL/SQL プログラムにより開始される接続のみです。PL/SQL プログラムでは、プログラム外部で開始された接続を受け入れることができません。

この章では、次の項目について説明します。

例外

表 101-1 に、TCP/IP パッケージで発生する例外を示します。

表 101-1 TCP/IP の例外

例外	説明
BUFFER_TOO_SMALL	事前確認に必要な入力バッファが小さすぎます。
END_OF_INPUT	接続先から読込み可能なデータがこれ以上ない場合に発生します。
NETWORK_ERROR	一般的なネットワーク・エラーです。
BAD_ARGUMENT	不正な引数が API コールに渡されました（負のバッファ・サイズなど）。
TRANSFER_TIMEOUT	データが読み込まれず、読込みタイムアウトが発生しました。
PARTIAL_MULTIBYTE_CHAR	完全に読み込まれた文字はありません。入力の終わりにマルチバイト・キャラクタの一部が検出されました。

例

次の例は、HTTP を介して Web ページを取り出すときの TCP/IP パッケージの使用方法を示したコードです。ポート 80（HTTP の標準ポート）でリスニングする Web サーバーに接続し、ルート・ドキュメントを要求します。

```
DECLARE
  c  utl_tcp.connection;  -- TCP/IP connection to the Web server
  ret_val pls_integer;
BEGIN
  c := utl_tcp.open_connection(remote_host => 'www.acme.com',
                               remote_port =>  80,
                               charset     => 'US7ASCII');  -- open connection
  ret_val := utl_tcp.write_line(c, 'GET / HTTP/1.0');    -- send HTTP request
  ret_val := utl_tcp.write_line(c);
  BEGIN
    LOOP
      dbms_output.put_line(utl_tcp.get_line(c, TRUE));  -- read result
    END LOOP;
  EXCEPTION
    WHEN utl_tcp.end_of_input THEN
      NULL; -- end of input
  END;
  utl_tcp.close_connection(c);
END;
```

次の例は、アプリケーションが TCP/IP パッケージを使用して、電子メールを送信する方法を示したコードです（PL/SQL による電子メールとも呼ばれます）。ポート 25 で SMTP サーバーに接続し、簡単なテキスト・メッセージを送信します。

```
PROCEDURE send_mail (sender    IN VARCHAR2,
                     recipient IN VARCHAR2,
                   message   IN VARCHAR2)
IS
    mailhost   VARCHAR2(30) := 'mailhost.mydomain.com';
    smtp_error  EXCEPTION;
    mail_conn   utl_tcp.connection;
    PROCEDURE smtp_command(command IN VARCHAR2,
                           ok      IN VARCHAR2 DEFAULT '250')
    IS
        response varchar2(3);
        len pls_integer;
    BEGIN
        len := utl_tcp.write_line(mail_conn, command);
        response := substr(utl_tcp.get_line(mail_conn), 1, 3);
        IF (response <> ok) THEN
            RAISE smtp_error;
        END IF;
    END;

BEGIN
    mail_conn := utl_tcp.open_connection(remote_host => mailhost,
                                         remote_port => 25,
                                         charset     => 'US7ASCII');
    smtp_command('HELO ' || mailhost);
    smtp_command('MAIL FROM: ' || sender);
    smtp_command('RCPT TO: ' || recipient);
    smtp_command('DATA', '354');
    smtp_command(message);
    smtp_command('QUIT', '221');
    utl_tcp.close_connection(mail_conn);
EXCEPTION
    WHEN OTHERS THEN
        -- Handle the error
END;
```

UTL_TCP サブプログラムの要約

表 101-2 UTL_TCP サブプログラム

サブプログラム	説明
「connection」101-5 ページ	TCP/IP 接続を示す PL/SQL レコード・タイプです。
「CRLF」101-6 ページ	CR-LF 改行文字列。多くの通信規格で一般的に使用されている改行文字列です。
「open_connection ファンクション」101-7 ページ	指定したサービスへの TCP/IP 接続をオープンします。
「available ファンクション」101-9 ページ	TCP/IP 接続からの読込みに使用可能なバイト数を決定します。
「read_raw ファンクション」101-11 ページ	オープン接続中のサービスからバイナリ・データを受信します。
「write_raw ファンクション」101-12 ページ	オープン接続中のサービスにバイナリ・メッセージを送信します。
「read_text ファンクション」101-12 ページ	オープン接続中のサービスからテキスト・データを受け取ります。
「write_text ファンクション」101-14 ページ	オープン接続中のサービスにテキスト・メッセージを送信します。
「read_line ファンクション」101-15 ページ	オープン接続中のサービスからテキスト行を受け取ります。
「write_line ファンクション」101-16 ページ	オープン接続中のサービスにテキスト行を送信します。
「get_raw()、get_text()、get_line() ファンクション」101-17 ページ	読込みファンクションの簡便な形式で、読込みデータ量ではなく読み込んだデータを戻します。
「flush プロシージャ」101-18 ページ	バッファが使用されている場合は、出力バッファの全データをサーバーに即時送信します。
「close_connection プロシージャ」101-18 ページ	オープン状態の TCP/IP 接続をクローズします。
「close_all_connections プロシージャ」101-19 ページ	オープン状態の TCP/IP 接続をすべてクローズします。

connection

TCP/IP 接続を示す PL/SQL レコード・タイプです。

構文

```
TYPE connection IS RECORD (
    remote_host    VARCHAR2(255), -- remote host name
    remote_port    PLS_INTEGER,   -- remote port number
    local_host     VARCHAR2(255), -- local host name
    local_port     PLS_INTEGER,   -- local port number
    charset        VARCHAR2(30),  -- character set for on-the-wire communication
    newline        VARCHAR2(2),   -- newline character sequence
    tx_timeout     PLS_INTEGER,   -- transfer time-out value (in seconds)
    private_sd     PLS_INTEGER,   -- for internal use
    );
```

フィールド

表 101-3 connection レコード・タイプのフィールド

フィールド	説明
remote_host	接続が確立したときのリモート・ホストの名前。接続が確立されない場合は NULL です。
remote_port	接続したリモート・ホストのポート番号。接続が確立されない場合は NULL です。
local_host	接続の確立に使用されるローカル・ホスト名。接続が確立されない場合は NULL です。
local_port	接続の確立に使用されるローカル・ホストのポート番号。接続が確立されない場合は NULL です。
charset	回線用キャラクタ・セット。データベース内のテキスト・メッセージは、回線上とは異なるキャラクタ・セット（つまり、通信プロトコルで指定されているキャラクタ・セットや他の通信上の終端で規定されているキャラクタ・セットなど）にコード化されている可能性があります。したがって、ネットワーク上での送受信時には、データベース内のテキスト・メッセージと回線用キャラクタ・セットの間の変換が行われます。

表 101-3 connection レコード・タイプのフィールド（続き）

フィールド	説明
newline	改行文字列。この改行文字列は、write_line() API で送信されるテキスト行に追加されます。
tx_timeout	この接続で読込みまたは書込み操作を中止するまでの UTL_TCP パッケージの待機時間（秒）。読込み操作の場合、読み込めるデータがない場合に操作が即座に中止されます。書込み操作の場合、出力バッファがフルで、ブロックされずにネットワークに送信できるデータがない場合に操作が中止されます。0（ゼロ）は、まったく待機しないことを示します。NULL は、無期限に待機することを示します。

使用上の注意

connection レコードのフィールドは、多くの場合 open_connection() を使用して、接続情報を戻すために使用されます。これらのフィールドを変更しても、接続には影響はありません。private_XXXX フィールドは、実装目的にのみ使用されます。これらの値は変更しないでください。

現行のリリースの UTL_TCP パッケージでは、open_connection が TCP/IP 接続を確立すると、local_host および local_port パラメータは無視されます。この場合、接続が確立されると、指定したローカル・ホストおよびポート番号が使用されません。local_host および local_port フィールドは、ファンクションにより戻される接続レコードで設定されません。

書込み操作のタイムアウトは、UTL_TCP パッケージの現行のリリースではサポートされていません。

CRLF

CR-LF 改行文字列。多くの通信規格で一般的に使用されている改行文字列です。

構文

```
CRLF varchar2(10);
```

使用上の注意

このパッケージ変数によって、多くのインターネット・プロトコルで一般的に使用されている改行文字列を定義します。これは write_line() に対する改行文字列のデフォルト値で、接続のオープン時に指定されます。このようなプロトコルでは <CR><LF> を使用して新しい行を示しますが、一部の実装では LF のみの使用を選択できます。この場合、ユーザーは、接続のオープン時に異なる改行文字列を指定できます。

CRLF パッケージ変数は、CR-LF 改行文字列を示す定数として使用されます。値を変更しないでください。変更すると、他の PL/SQL アプリケーションでエラーが発生する場合があります。

open_connection ファンクション

このファンクションは、指定したサービスへの TCP/IP 接続をオープンします。

構文

```
UTL_TCP.OPEN_CONNECTION  (remote_host      IN VARCHAR2,
                          remote_port      IN PLS_INTEGER,
                          local_host       IN VARCHAR2 DEFAULT NULL,
                          local_port       IN PLS_INTEGER DEFAULT NULL,
                          in_buffer_size   IN PLS_INTEGER DEFAULT NULL,
                          out_buffer_size  IN PLS_INTEGER DEFAULT NULL,
                          charset          IN VARCHAR2 DEFAULT NULL,
                          newline          IN VARCHAR2 DEFAULT CRLF,
                          tx_timeout       IN PLS_INTEGER DEFAULT NULL)
                          RETURN connection;
```

パラメータ

表 101-4 open_connection ファンクションのパラメータ

パラメータ	説明
remote_host（IN）	サービスを提供するホストの名前。remote_host が NULL の場合は、ローカル・ホストに接続します。
remote_port（IN）	サービスが接続をリスニングするポート番号。
local_host（IN）	サービスを提供するホストの名前。NULL の場合は無視されます。
local_port（IN）	サービスが接続をリスニングするポート番号。NULL の場合は無視されます。

表 101-4 open_connection ファンクションのパラメータ（続き）

パラメータ	説明
in_buffer_size（IN）	入力バッファのサイズ。入力バッファを使用すると、サーバーからデータを受信する際の実行パフォーマンスを高速化できます。バッファの適正サイズは、クライアントとサーバーの間のデータの流れやネットワーク条件によって異なります。0（ゼロ）の値は、バッファを使用しないことを意味します。NULL の値は、バッファが使用されているかどうかをコール元が無視することを意味します。入力バッファの最大サイズは、32767 バイトです。
out_buffer_size（IN）	出力バッファのサイズ。出力バッファを使用すると、サーバーにデータを送信する際の実行パフォーマンスを高速化できます。バッファの適正サイズは、クライアントとサーバーの間のデータの流れやネットワーク条件によって異なります。0（ゼロ）の値は、バッファを使用しないことを意味します。NULL の値は、バッファが使用されているかどうかをコール元が無視することを意味します。出力バッファの最大サイズは、32767 バイトです。
charset（IN）	回線用キャラクタ・セット。データベース内のテキスト・メッセージは、回線上とは異なるキャラクタ・セット（つまり、通信プロトコルで指定されているキャラクタ・セットや他の通信上の終端で規定されているキャラクタ・セットなど）にコード化されている可能性があります。したがって、ネットワーク上での送受信時には、read_text()、read_line()、write_text() および write_line() を使用して、データベース内のテキスト・メッセージと回線用キャラクタ・セットの間の変換が行われます。変換が不要な場合は、このパラメータを NULL に設定してください。
newline（IN）	改行文字列。この改行文字列は、write_line() API で送信されるテキスト行に追加されます。
tx_timeout	この接続で読込みまたは書込み操作を中止するまでの UTL_TCP パッケージの待機時間（秒）。読込み操作の場合、読み込めるデータがない場合に操作が即座に中止されます。書込み操作の場合、出力バッファがフルで、ブロックされずにネットワークに送信できるデータがない場合に操作が中止されます。0（ゼロ）は、まったく待機しないことを示します。NULL は、無期限に待機することを示します。

使用上の注意

この UTL_TCP パッケージでオープンした接続は、オープン状態のまま、あるデータベース・コールから共有サーバー構成内の別のデータベース・コールに渡すことができます。ただし、接続は明示的にクローズする必要があります。接続を格納している PL/SQL レコード変数が PL/SQL プログラムで無効になると、接続はオープン状態のままになります。不要な接続を適切にクローズしないと、ローカルおよびリモートのシステム・リソースが無駄になります。

open_connection により TCP/IP 接続を確立した場合、現在は local_host および local_port パラメータが無視されます。この場合、接続が確立されると、指定したローカル・ホストおよびポート番号が使用されません。

現行のリリースの UTL_TCP パッケージでは、open_connection が TCP/IP 接続を確立すると、local_host および local_port パラメータは無視されます。この場合、接続が確立されると、指定したローカル・ホストおよびポート番号が使用されません。local_host および local_port フィールドは、ファンクションにより戻される接続レコードで設定されません。

書込み操作のタイムアウトは、UTL_TCP パッケージの現行のリリースではサポートされていません。

関連ファンクション

close_connection()、close_all_connections()

available ファンクション

このファンクションは、TCP/IP 接続からの読込みに使用可能なバイト数を決定します。この数は、ブロッキングなしで即時に読み込むことができるバイト数です。接続からデータを読み込めるかどうかを判断します。

構文

```
UTL_TCP.AVAILABLE (
   c         IN OUT NOCOPY connection,
   timeout   IN PLS_INTEGER DEFAULT 0)
RETURN PLS_INTEGER;
```

パラメータ

表 101-5 available ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	読込み可能なデータ量を判断する TCP 接続。
timeout	読込みを中止し、使用可能なデータがないことを報告するまでに待機する時間（秒）。0（ゼロ）は、まったく待機しないことを示します。NULL は、無期限に待機することを示します。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。ユーザーはこの API を使用して、読込み API をコールする前に、データが読込み可能かどうかを確認できるため、入力からデータを読み込む準備が整っていないことでプログラムがブロックされることはありません。

このファンクションでは、読込み可能なバイト数が実際に使用可能なバイト数よりも少なく戻される場合があります。プラットフォームによっては、このファンクションが常に 1 を戻し、使用可能なデータがあることを示します。アプリケーションの移植性を検討する場合、読込み可能なデータがある場合はこのファンクションが正の値を戻し、ない場合は 0（ゼロ）を戻すことを前提としてください。次の例で、移植可能な方法によるファンクションの使用について示します。

```
DECLARE
   c   utl_tcp.connection
   data VARCHAR2(256);
   len  PLS_INTEGER;
BEGIN
   c := utl_tcp.open_connection(...);
   LOOP
      IF (utl_tcp.available(c) > 0) THEN
         len := utl_tcp.read_text(c, data, 256);
      ELSE
         ---do some other things
        . . . .
      END IF
   END LOOP;
END;
```

関連ファンクション

read_raw()、read_text()、read_line()

read_raw ファンクション

このファンクションは、オープン接続中のサービスからバイナリ・データを受信します。

構文

```
UTL_TCP.READ_RAW (c     IN OUT NOCOPY connection,
                  data  IN OUT NOCOPY RAW,
                  len   IN            PLS_INTEGER DEFAULT 1,
                  peek  IN            BOOLEAN     DEFAULT FALSE)
                                      RETURN PLS_INTEGER;
```

パラメータ

表 101-6 read_raw ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	データの受信元 TCP 接続。
data（IN OUT COPY）	受信データ。
len（IN）	受信するデータのバイト数。
peek（IN）	通常、ユーザーはデータを読み込むとそのデータを入力キューから削除、つまり消費しようとします。しかし、入力キューからデータを削除しないで、単にデータを前もって確認、つまり覗くだけにし、次回コール時にも依然としてそのデータが読込み可能な（または覗ける）状態にしておくことが望ましい場合があります。入力キューにデータを保持するには、このフラグを TRUE に設定し、接続のオープン前に入力バッファをセットアップする必要があります。覗く（つまり、読み込んでも入力キューに保持）ことができるデータ量は、入力バッファ・サイズより小さくしてください。
戻り値	実際に受信したデータのバイト数。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。このファンクションは、指定した文字数が読み込まれるまで、または入力の最後に達するまで戻りません。

接続がオープンされたときに転送のタイムアウトが設定されている場合、このファンクションは、タイムアウトが発生するまでの間、各データ・パケットが読み込まれる準備を完了するのを待機します。タイムアウトが発生すると、このファンクションは読込みを停止し、正常に読み込んだデータをすべて戻します。正常に読み込まれたデータがない場合は、transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

関連ファンクション

read_text()、read_line()、available()

write_raw ファンクション

このファンクションは、オープン接続中のサービスにバイナリ・メッセージを送信します。

構文

```
UTL_TCP.WRITE_RAW (c    IN OUT NOCOPY connection,
                   data IN            RAW,
                   len  IN            PLS_INTEGER DEFAULT NULL)
                                      RETURN PLS_INTEGER;
```

表 101-7 write_raw ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	データの送信先 TCP 接続。
data（IN）	送信するデータを含んだバッファ。
len（IN）	送信するデータのバイト数。len が NULL の場合は、データの長さ全体が書き込まれます。実際に書き込まれるデータ量は、ネットワーク条件のために削減される可能性があります。
戻り値	実際に送信したデータのバイト数。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。

関連ファンクション

write_text()、write_line()、flush()

read_text ファンクション

このファンクションは、オープン接続中のサービスからテキスト・データを受信します。

構文

```
UTL_TCP.READ_TEXT (c    IN OUT NOCOPY connection,
                   data IN OUT NOCOPY VARCHAR2,
                   len  IN            PLS_INTEGER DEFAULT 1,
                   peek IN            BOOLEAN     DEFAULT FALSE) RETURN PLS_INTEGER;
```

表 101-8 read_text ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	データの受信元 TCP 接続。
data （IN OUT NOCOPY）	受信データ。
len（IN）	受信データの文字数。
peek（IN）	通常、ユーザーはデータを読み込むとそのデータを入力キューから削除、つまり消費しようとします。しかし、入力キューからデータを削除しないで、単にデータを前もって確認、つまり覗くだけにし、次回コール時にも依然としてそのデータが読込み可能な（または覗ける）状態にしておくことが望ましい場合もあります。入力キューにデータを保持するには、このフラグを TRUE に設定し、接続のオープン時に入力バッファをセットアップする必要があります。覗く（つまり、読み込んでも入力キューに保持）ことができるデータ量は、入力バッファ・サイズより小さくしてください。
戻り値	実際に受信したデータの文字数。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。このファンクションは、指定した文字数が読み込まれるまで、または入力の最後に達するまで戻りません。テキスト・メッセージは、コール元に戻される前に、接続のオープン時に指定した回線用キャラクタ・セットからデータベース・キャラクタ・セットに変換されます。

明示的に無視された場合を除き、VARCHAR2 バッファのサイズはバイト数で指定されます。これに対し、len パラメータは読み込まれる最大文字数を参照します。データベース・キャラクタ・セットにマルチバイトが設定されていると、単一の文字が複数のバイトで構成されている場合は、バッファが最大文字数を保持できることを確認する必要があります。一般に、VARCHAR2 バッファは読み込まれる文字数にデータベース・キャラクタ・セットの 1 文字の最大バイト数を乗算した数と等しくする必要があります。

接続がオープンされたときに転送のタイムアウトが設定されている場合、このファンクションは、タイムアウトが発生するまでの間、各データ・パケットが読み込まれる準備を完了するのを待機します。タイムアウトが発生すると、このファンクションは読込みを停止し、正常に読み込んだデータをすべて戻します。正常に読み込まれたデータがない場合は、transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

マルチバイト・キャラクタの一部が入力の終わりで検出された場合、このファンクションは読込みを中止し、完全に読み込まれたマルチバイト・キャラクタをすべて戻します。正常に読み込まれた文字がない場合は、partial_multibyte_char 例外が発生します。
read_raw ファンクションを使用して例外を処理し、部分的なマルチバイト・キャラクタのバイトをバイナリとして読み込むことができます。マルチバイト・キャラクタの一部が渡されていない状態でタイムアウトが発生したため、マルチバイト・キャラクタの残りの部分が入力途中で表示された場合は、そのかわりに transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

関連ファンクション

read_raw()、read_line()、available()

write_text ファンクション

このファンクションは、オープン接続中のサービスにテキスト・メッセージを送信します。

構文

```
UTL_TCP.WRITE_TEXT (c    IN OUT NOCOPY connection,
                    data IN            VARCHAR2,
                    len  IN            PLS_INTEGER DEFAULT NULL)
                                       RETURN PLS_INTEGER;
```

表 101-9 write_text ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	データの送信先 TCP 接続。
data（IN）	送信するデータを含んだバッファ。
len（IN）	送信するデータの文字数。len が NULL の場合は、データの長さ全体が書き込まれます。実際に書き込まれるデータ量は、ネットワーク条件のために削減される可能性があります。
戻り値	実際に送信したデータの文字数。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。テキスト・メッセージは、回線に送信される前に、接続のオープン時に指定した回線用キャラクタ・セットに変換されます。

関連ファンクション

write_raw()、write_line()、flush()

read_line ファンクション

このファンクションは、オープン接続中のサービスからテキスト行を受け取ります。行は、LF 改行、CR 改行または CR-LF 改行（CR の後に LF が続く）で終了します。

構文

```
UTL_TCP.READ_LINE (c           IN OUT NOCOPY connection,
                   data        IN OUT NOCOPY VARCHAR2,
                   remove_crlf IN            BOOLEAN DEFAULT FALSE,
                   peek        IN            BOOLEAN DEFAULT FALSE)
                                             RETURN PLS_INTEGER;
```

表 101-10 read_line ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	データの受信元 TCP 接続。
data（IN OUT NOCOPY）	受信データ。
remove_crlf（IN）	TRUE の場合は、後に続く CR または LF 文字が受信メッセージから削除されます。
peek（IN）	通常、ユーザーはデータを読み込むとそのデータを入力キューから削除、つまり消費しようとします。しかし、入力キューからデータを削除しないで、単にデータを前もって確認、つまり覗くだけにし、次回コール時にも依然としてそのデータが読込み可能な（または覗ける）状態にしておくことが望ましい場合があります。入力キューにデータを保持するには、このフラグを TRUE に設定し、接続のオープン前に入力バッファをセットアップする必要があります。覗く（つまり、読み込んでも入力キューに保持）ことができるデータ量は、入力バッファ・サイズより小さくしてください。
戻り値	実際に受信したデータの文字数。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。このファンクションは、行の終わりまたは入力の最後に達するまで戻りません。テキスト・メッセージは、コール元に戻される前に、接続のオープン時に指定した回線用キャラクタ・セットからデータベース・キャラクタ・セットに変換されます。

接続がオープンされたときに転送のタイムアウトが設定されている場合、このファンクションは、タイムアウトが発生するまでの間、各データ・パケットが読み込まれる準備を完了するのを待機します。タイムアウトが発生すると、このファンクションは読込みを停止し、正常に読み込んだデータをすべて戻します。正常に読み込まれたデータがない場合は、transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

マルチバイト・キャラクタの一部が入力の終わりで検出された場合、このファンクションは読込みを中止し、完全に読み込まれたマルチバイト・キャラクタをすべて戻します。正常に読み込まれた文字がない場合は、partial_multibyte_char 例外が発生します。read_raw ファンクションを使用して例外を処理し、部分的なマルチバイト・キャラクタのバイトをバイナリとして読み込むことができます。マルチバイト・キャラクタの一部が渡されていない状態でタイムアウトが発生したため、マルチバイト・キャラクタの残りの部分が入力途中で表示された場合は、そのかわりに transfer_timeout 例外が発生します。例外を処理し、読込み操作を後で再試行できます。

関連ファンクション

read_raw()、read_text()、available()

write_line ファンクション

このファンクションは、オープン接続中のサービスにテキスト行を送信します。送信前に、改行文字列がメッセージに追加されます。

構文

```
UTL_TCP.WRITE_LINE (c    IN OUT NOCOPY connection,
                    data IN            VARCHAR2 DEFAULT NULL)
                                       RETURN PLS_INTEGER;
```

表 101-11 write_line ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	データの送信先 TCP 接続。
data（IN）	送信するデータを含んだバッファ。
戻り値	実際に送信したデータの文字数。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。テキスト・メッセージは、回線に送信される前に、接続のオープン時に指定した回線用キャラクタ・セットに変換されます。

関連ファンクション

write_raw()、write_text()、flush()

get_raw()、get_text()、get_line() ファンクション

読込みファンクションの簡便な形式で、読込みデータ量ではなく読み込んだデータを戻します。

構文

```
UTL_TCP.GET_RAW (c     IN OUT NOCOPY connection,
                 len   IN            PLS_INTEGER DEFAULT 1,
                 peek  IN            BOOLEAN     DEFAULT FALSE) RETURN RAW;
UTL_TCP.GET_TEXT (c    IN OUT NOCOPY connection,
                  len  IN            PLS_INTEGER DEFAULT 1,
                  peek IN            BOOLEAN     DEFAULT FALSE) RETURN VARCHAR2;
UTL_TCP.GET_LINE (c           IN OUT NOCOPY connection,
                  remove_crlf IN            BOOLEAN DEFAULT false,
                  peek        IN            BOOLEAN DEFAULT FALSE) RETURN VARCHAR2;
```

表 101-12 get_raw()、get_text()、get_line() ファンクションのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	データの受信元 TCP 接続。
len（IN）	受信するデータのバイト数（または VARCHAR2 の文字数）。デフォルトは 1 です。
peek（IN）	通常、ユーザーはデータを読み込むとそのデータを入力キューから削除、つまり消費しようとします。しかし、入力キューからデータを削除しないで、単にデータを前もって確認、つまり覗くだけにし、次回コール時にも依然としてそのデータが読込み可能な（または覗ける）状態にしておくことが望ましい場合があります。入力キューにデータを保持するには、このフラグを TRUE に設定し、接続のオープン前に入力バッファをセットアップする必要があります。覗く（つまり、読み込んでも入力キューに保持）ことができるデータ量は、入力バッファ・サイズより小さくしてください。
remove_crlf（IN）	TRUE の場合は、後に続く CR または LF 文字が受信メッセージから削除されます。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。

この項で説明した get_* API のすべては、対応する read_* API の読込みタイムアウトに関する箇所を参照してください。get_text および get_line の詳細は、対応する read_* API のキャラクタ・セット変換、バッファ・サイズおよびマルチバイト・キャラクタの箇所を参照してください。

関連ファンクション

read_raw()、read_text()、read_line()

flush プロシージャ

このプロシージャは、バッファが使用されている場合は、出力バッファの全データをサーバーに即時送信します。

構文

```
UTL_TCP.FLUSH (c IN OUT NOCOPY connection);
```

パラメータ

表 101-13 flush プロシージャのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	データの送信先 TCP 接続。

使用上の注意

open_connection() へのコールを介して、接続がすでにオープン状態になっている必要があります。

関連ファンクション

write_raw()、write_text()、write_line()

close_connection プロシージャ

このプロシージャは、オープン状態の TCP/IP 接続をクローズします。

構文

```
UTL_TCP.close_CLOSE_CONNECTION (c IN OUT NOCOPY connection);
```

パラメータ

表 101-14 close_connection プロシージャのパラメータ

パラメータ	説明
c（IN OUT NOCOPY）	クローズする TCP 接続。

使用上の注意

接続は、open_connection() へのコールによって事前にオープンされている必要があります。c の remote_host、remote_port、local_host、local_port および charset フィールドは接続がクローズされた後にリセットされます。

オープン状態の接続は、明示的にクローズする必要があります。オープン状態の接続は、接続を格納している PL/SQL レコード変数が PL/SQL プログラムで無効になると、そのままオープン状態となります。不要な接続を適切にクローズしないと、ローカルおよびリモートのシステム・リソースが無駄になります。

close_all_connections プロシージャ

このプロシージャは、オープン状態の TCP/IP 接続をすべてクローズします。

構文

```
UTL_TCP.CLOSE_ALL_CONNECTIONS;
```

使用上の注意

このコールは、PL/SQL プログラムが参照先のない接続を回避する前に、すべての接続をクローズするために用意されています。

関連ファンクション

open_connection()、close_connection()

102

UTL_URL

UTL_URL パッケージには、ESCAPE および UNESCAPE の 2 つのファンクションがあります。

関連項目： 第 96 章「UTL_HTTP」

この章では、次の項目について説明します。

- UTL_URL パッケージの概要
- UTL_URL の例外
- UTL_URL サブプログラムの要約

UTL_URL パッケージの概要

Uniform Resource Locator（URL）は、ページや画像などの Web リソースを識別する文字列です。URL を使用すると、Hypertext Transfer Protocol（HTTP）を介してこのようなリソースにアクセスできます。たとえば、Oracle の Web サイトの URL は次のとおりです。

```
http://www.oracle.com
```

通常、URL には英語のアルファベット文字、桁および句読点記号が含まれます。このような文字は、未予約文字と呼ばれます。マルチバイト・キャラクタやバイナリ・オクテット・コードなどの他の文字は、Web ブラウザまたは Web サーバーが正確に処理されるようにエスケープする必要があります。ドル記号（$）、疑問符（?）、コロン（:）および等号（=）など一部の句読点文字は、URL のデリミタとして予約されています。このような文字は、予約文字と呼ばれます。このような文字を処理するには、デリミタとして文字を扱うのではなく、エスケープする必要があります。

未予約文字は次のとおりです。

- A 〜 Z、a 〜 z および 0 〜 9。
- ハイフン（-）、アンダースコア（_）、ピリオド（.）、感嘆符（!）チルダ（~）、アスタリスク（*）、アクセント（'）、左カッコ（(）、右カッコ（)）。

予約文字は次のとおりです。

- セミコロン（;）、スラッシュ（/）、疑問符（?）、コロン（:）、アット・マーク（@）、アンパサンド（&）、等号（=）、プラス記号（+）、ドル記号（$）およびカンマ（,）。

UTL_URL パッケージは、URL 文字のエスケープおよびエスケープ解除のメカニズムを提供する 2 つのファンクションを備えています。UTL_HTTP パッケージを経由して URL が Web ページをフェッチする前に、このエスケープ・ファンクションを使用して URL をエスケープします。URL から情報を抽出する前に、エスケープ解除ファンクションを使用して、エスケープされた URL のエスケープを解除します。

詳細は、Request For Comments（RFC）ドキュメントの RFC2396 を参照してください。ここで説明する URL のエスケープおよびエスケープ解除のメカニズムは、HTML の仕様で説明されている、メカニズムをエンコードする x-www-form-urlencoded とは異なります。

```
http://www.w3.org/TR/html
```

x-www-form-urlencoded エンコーディングは、UTL_URL.ESCAPE ファンクションを使用して次のように実装できます。

```
CREATE OR REPLACE FUNCTION form_url_encode (
   data IN VARCHAR2,
   charset IN VARCHAR2)
RETURN VARCHAR2 AS
BEGIN
  RETURN utl_url.escape(data, TRUE, charset); -- note use of TURE
END;
```

form-URL-encode scheme を使用してエンコードしたデータをデコードする場合は、次のファンクションによってデコード・スキームが実装されます。

```
function form_url_decode(
   data in varchar2,
   charset in varchar2)
  return varchar2 as

begin
  return utl_url.unescape(
     replace(data, '+', ' '),
     charset);
end;
```

UTL_URL の例外

表 102-1 に、UTL_URL パッケージの API が起動されたときに、呼び出される可能性のある例外を示します。

表 102-1 UTL_URL の例外

例外	エラー・コード	理由
bad_url	29262	URL に含まれているエスケープ・コード列が不適切です。
bad_fixed_width_charset	29274	固定長マルチバイト・キャラクタ・セットは URL キャラクタ・セットとして許可されていません。

UTL_URL サブプログラムの要約

表 102-2 UTL_URL パッケージのサブプログラム

サブプログラム	説明
「ESCAPE ファンクション」 102-4 ページ	%2-digit-hex-code 形式を使用してエスケープされた、不正な文字（およびオプションで予約文字）を含む URL を戻します。
「UNESCAPE ファンクション」 102-6 ページ	エスケープ文字列をエスケープ解除し、URL の元の形式にします。%XX エスケープ文字列を元の文字に変換します。

ESCAPE ファンクション

このファンクションは、%2-digit-hex-code 形式を使用してエスケープされた、不正な文字（およびオプションで予約文字）を含む URL を戻します。

構文

```
UTL_URL.ESCAPE (
   url                   IN VARCHAR2,
   escape_reserved_chars IN BOOLEAN  DEFAULT FALSE,
   url_charset           IN VARCHAR2 DEFAULT
                                     utl_http.body_charset)
   RETURN VARCHAR2;
```

パラメータ

表 102-3 ESCAPE ファンクションのパラメータ

パラメータ	説明
url (IN)	元の URL。
escape_reserved_chars (IN)	URL の予約文字をエスケープする必要があるかどうかを示します。TRUE に設定すると、URL の予約文字および不正な文字の両方がエスケープされます。FALSE に設定すると、不正な URL 文字のみがエスケープされます。デフォルトは FALSE です。
url_charset (IN)	文字（シングルバイトまたはマルチバイト）のエスケープに際して、%hex-code 形式でエスケープする前に、その文字をどのキャラクタ・セットに変換すべきかを指定します。url_charset が NULL である場合は、データベース・キャラクタ・セットとみなされ、キャラクタ・セットの変換は行われません。現在 UTL_HTTP パッケージの本体キャラクタ・セットのデフォルトである ISO-8859-1 がデフォルト値です。キャラクタ・セットは、Internet Assigned Numbers Authority（IANA）または Oracle のネーミング規則で命名できます。

使用上の注意

このファンクションを使用して、URL 仕様 RFC2396 で定義された不正な文字を含む URL をエスケープします。URL の適切な文字は次のとおりです。

- A 〜 Z、a 〜 z および 0 〜 9。
- ハイフン（-）、アンダースコア（_）、ピリオド（.）、感嘆符（!）チルダ（~）、アスタリスク（*）、アクセント（'）、左カッコ（(）、右カッコ（)）。

予約文字は次のとおりです。

- セミコロン（;）、スラッシュ（/）、疑問符（?）、コロン（:）、アット・マーク（@）、アンパサンド（&）、等号（=）、プラス記号（+）、ドル記号（$）およびカンマ（,）。

予約文字の多くは、URL でデリミタとして使用されます。前述の文字以外については、escape_url を使用してエスケープする必要があります。また、URL の問合せ文字列で名前と値の組合せに予約文字を使用するには、別々に文字をエスケープする必要があります。escape_url は文字をエスケープする必要があるかどうかを認識できません。これは、文字がいったん URL に組み込まれると、実際のデリミタからは認識不可能になるためです。たとえば、$logon=scott/tiger という名前と値の組合せを URL の問合せ文字列に渡すには、$ および / を %24logon=scott%2Ftiger として個別にエスケープし、URL で使用します。

通常は、エスケープの対象でない予約文字（デリミタ）も含めて URL 全体をエスケープします。たとえば、次のようにします。

```
utl_url.escape('http://www.acme.com/a url with space.html')
```

戻り値

```
http://foo.com/a%20url%20with%20space.html
```

他の状況では、予約文字を含む値を使用した問合せ文字列を送信できます。この場合、escape_reserved_chars を TRUE に設定して対象の値のみを完全にエスケープし、URL の残りの部分に連結します。たとえば、次のようにします。

```
url := 'http://www.acme.com/search?check=' || utl_url.escape
('Is the use of the "$" sign okay?', TRUE);
```

この式は、'Is the use of the "$" sign okay?' の疑問符（?）、ドル記号（$）および空白文字をエスケープしていますが、問合せ文字列の使用を示す URL の search の後の ? はエスケープされていません。

Web ページをフェッチする Web サーバーでは、ユーザーのデータベースと異なるキャラクタ・セットを使用することがあります。この場合は、Web サーバーのキャラクタ・セットに url_charset を指定し、エスケープする必要のある文字がターゲットのキャラクタ・セットでエスケープされるようにします。たとえば、ASCII Web サーバーにアクセスする EBCDIC データベースのユーザーは、US7ASCII を使用して URL をエスケープする必要があります。これにより、空白が %40（EBCDIC での空白の hex コード）ではなく、%20（ASCII での空白の hex コード）にエスケープされます。

このファンクションは、URL 形式の妥当性は検証しません。

UNESCAPE ファンクション

このファンクションは、URL に含まれるエスケープ文字列をエスケープ解除して元の形式にし、%XX エスケープ文字列を元の文字に変換します。

構文

```
UTL_URL.UNESCAPE (
   url             IN VARCHAR2,
   url_charset     IN VARCHAR2 DEFAULT utl_http.body_charset)
                   RETURN VARCHAR2;
```

パラメータ

表 102-4 UNESCAPE ファンクションのパラメータ

パラメータ	説明
url (IN)	エスケープ解除する URL。
url_charset (IN)	文字のエスケープが解除された後、その文字は source_charset キャラクタ・セットにあるとみなされ、source_charset からデータベース・キャラクタ・セットに変換されます。その後、URL が戻されます。source_charset が NULL の場合、データベース・キャラクタ・セットとみなされ、キャラクタ・セットの変換は行われません。現在 UTL_HTTP パッケージの本体キャラクタ・セットのデフォルトである ISO-8859-1 がデフォルト値です。キャラクタ・セットは、Internet Assigned Numbers Authority（IANA）または Oracle のネーミング規則で命名できます。

使用上の注意

URL を受信する Web サーバーでは、ユーザーのデータベースと異なるキャラクタ・セットを使用することがあります。この場合は、Web サーバーのキャラクタ・セットに url_charset を指定し、エスケープを解除する必要のある文字がソースのキャラクタ・セットでエスケープ解除されるようにします。たとえば、ASCII Web サーバーから URL を受信する EBCDIC データベースのユーザーは、US7ASCII を使用して URL のエスケープを解除する必要があります。これにより、%20（0x20 は ASCII での空白の hex コード）が ?（0x20 は EBCDIC で有効な文字ではないため）ではなく、空白としてエスケープ解除されます。

このファンクションは、URL 形式の妥当性は検証しません。

103

ANYDATA タイプ

ANYDATA には、指定のタイプのインスタンスおよびそのタイプの説明が含まれています。つまり、ANYDATA は自己記述型です。ANYDATA は、データベースに永続的に格納できます。

LOB がタイプに埋め込まれた ANYDATA インスタンスの永続的な格納については、現時点ではサポートされていません。

この章では、次の項目について説明します。

構成

AnyData を構成するには、2 通りの方法があります。Convert*() コールを使用すると、単一のコールで AnyData 全体を構成できます。Oracle ORDBMS のあらゆるタイプから AnyData まで、明示的な CAST ファンクションとして機能します。

```
STATIC FUNCTION ConvertNumber(num IN NUMBER) RETURN AnyData,
STATIC FUNCTION ConvertDate(dat IN DATE) RETURN AnyData,
STATIC FUNCTION ConvertChar(c IN CHAR) RETURN AnyData,
STATIC FUNCTION ConvertVarchar(c IN VARCHAR) RETURN AnyData,
STATIC FUNCTION ConvertVarchar2(c IN VARCHAR2) RETURN AnyData,
STATIC FUNCTION ConvertRaw(r IN RAW) RETURN AnyData,
STATIC FUNCTION ConvertBlob(b IN BLOB) RETURN AnyData,
STATIC FUNCTION ConvertClob(c IN CLOB) RETURN AnyData,
STATIC FUNCTION ConvertBfile(b IN BFILE) RETURN AnyData,
STATIC FUNCTION ConvertObject(obj IN "<object_type>") RETURN AnyData,
STATIC FUNCTION ConvertRef(rf IN REF "<object_type>") RETURN AnyData,
STATIC FUNCTION ConvertCollection(col IN "<COLLECTION_1>") RETURN AnyData,
```

AnyData を構成するもう 1 つの方法は、個別のアプローチによるものです。構成プロセスは、BeginCreate() コールで開始し、EndCreate() コールで終了します。この 2 つのコールの間では、Set*() コールを使用して、オブジェクト・タイプの各属性またはコレクションの要素を設定できます。オブジェクト属性およびコレクション要素に個別にアクセスするには、Get*() コールの前に PieceWise() コールを起動しておく必要があります。

注意 : AnyData は、最初の属性（またはコレクション要素）から順番に構成またはアクセスする必要があります。BeginCreate() コールは、個別モードで構成を自動的に開始します。BeginCreate() の後、すぐに PieceWise() をコールする必要はありません。構成プロセスを終了するには、EndCreate() をコールする必要があります（その後、アクセス・コールが実行できます）。

ANYDATA サブプログラムの要約

表 103-1 ANYDATA サブプログラム

サブプログラム	説明
「BEGINCREATE 静的プロシージャ」103-3 ページ	新しい AnyData で作成プロセスを開始します。
「PIECEWISE メンバー・プロシージャ」103-4 ページ	現在のデータ値のアクセス・モードを一度に属性に設定します（データ値が TYPECODE_OBJECT の場合）。
「SET メンバー・プロシージャ」103-5 ページ	現行のデータ値を設定します。
ENDCREATE メンバー・プロシージャ 103-7 ページ	AnyData の作成を終了します。
「GETTYPENAME メンバー・ファンクション」103-7 ページ	AnyData の完全修飾名を取得します。
「GETTYPE メンバー・ファンクション」103-8 ページ	AnyData のタイプを取得します。
「GET メンバー・ファンクション」103-9 ページ	現行のデータ値（適切なタイプのデータ値）を取得します。

BEGINCREATE 静的プロシージャ

このプロシージャは、新しい AnyData で作成プロセスを開始します。

構文

```
STATIC PROCEDURE BeginCreate(
   dtype          IN OUT NOCOPY AnyType,
   adata          OUT NOCOPY AnyData);
```

パラメータ

表 103-2 BEGINCREATE プロシージャのパラメータ

パラメータ	説明
dtype	AnyData のタイプ。（OCI_TYPECODE_OBJECT またはコレクション typecode に対応する必要があります）。
adata	構成される AnyData。

例外

DBMS_TYPES.invalid_parameters:dtype は無効です（完全に構成されていないなど）。

使用上の注意

このコールの後、すぐに PieceWise() をコールする必要はありません。この構成プロセスは、個別モードで自動的に開始されます。

PIECEWISE メンバー・プロシージャ

このプロシージャは、現在のデータ値のアクセス・モードを一度に属性に設定します（データ値が TYPECODE_OBJECT の場合）。

データ値のアクセス・モードを一度にコレクション要素に設定します（データ値がコレクション・タイプの場合）。このコールが実行されると、後続の Set*() および Get*() へのコールにより順番に個別の属性またはコレクション要素を取得します。

構文

```
MEMBER PROCEDURE PieceWise(
   self         IN OUT NOCOPY AnyData);
```

パラメータ

表 103-3 BEGINCREATE プロシージャのパラメータ

パラメータ	説明
self	現在のデータ値。

例外

- DBMS_TYPES.invalid_parameters
- DBMS_TYPES.incorrect_usage: 不適切な使用時。

使用上の注意

このコールの実行前は、現行のデータ値は、OBJECT または COLLECTION タイプである必要があります。

オブジェクトまたはコレクション・タイプのネストされた属性の個別の構成またはアクセスはサポートされていません。

SET メンバー・プロシージャ

現行のデータ値を設定します。

現行のデータ値のタイプに応じてコールする必要のあるプロシージャのリストです。データ値のタイプは、個別の構成プロセス中の現在の位置での属性のタイプにする必要があります。

構文

```
MEMBER PROCEDURE SetNumber(
   self        IN OUT NOCOPY AnyData,
   num         IN NUMBER,
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetDate(
   self        IN OUT NOCOPY AnyData,
   dat         IN DATE,
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetChar(
   self        IN OUT NOCOPY AnyData,
   c           IN CHAR,
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetVarchar(
   self        IN OUT NOCOPY AnyData,
   c           IN VARCHAR,
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetVarchar2(
   self        IN OUT NOCOPY AnyData,
   c           IN VARCHAR2,
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetRaw(
   self        IN OUT NOCOPY AnyData,
   r           IN RAW,
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetBlob(
   self        IN OUT NOCOPY AnyData,
   b           IN BLOB,
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetClob(
   self        IN OUT NOCOPY AnyData,
   c           IN CLOB,
   last_elem   IN boolean DEFAULT FALSE)'
MEMBER PROCEDURE SetBfile(
   self        IN OUT NOCOPY AnyData,
   b           IN BFILE,
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetObject(
   self        IN OUT NOCOPY AnyData,
```

```
   obj         IN "<object_type>",
   last_elem   IN boolean DEFAULT FALSE);
MEMBER PROCEDURE SetRef(
   self        IN OUT NOCOPY AnyData,
   rf          IN REF "<object_type>",
   last_elem   IN boolean DEFAULT FALSE),
MEMBER PROCEDURE SetCollection(
   self        IN OUT NOCOPY AnyData,
   col         IN "<collectyion_type>",
   last_elem   IN boolean DEFAULT FALSE);
```

パラメータ

表 103-4 SET*() プロシージャのパラメータ

パラメータ	説明
self	AnyData。
num	設定する数値など。
last_elem	AnyData がコレクションを表す場合にのみ関連します。 コレクションの最後の要素の場合は TRUE、そうでない場合は FALSE に設定します。

例外

- DBMS_TYPES.invalid_parameters: 無効なパラメータ（作成プロセスのこの時点での数値の追加が適切ではない場合）。
- DBMS_TYPES.incorrect_usage: 不適切な使用時。
- DBMS_TYPES.type_mismatch: 予想されたタイプが渡されたタイプと異なる場合。

使用上の注意

BeginCreate() がコールされた場合は、個別モードですでに構成が開始されています。Set*() への後続のコールにより、次の属性値が設定されます。

AnyData がスタンドアロンのコレクションである場合は、Set*() をコールすると次のコレクション要素が設定されます。

ENDCREATE メンバー・プロシージャ

このプロシージャは、AnyData の作成を終了します。このコール以降は、その他の作成ファンクションをコールすることはできません。

構文

```
MEMBER PROCEDURE EndCreate(
   self         IN OUT NOCOPY AnyData);
```

パラメータ

表 103-5 ENDCREATE プロシージャのパラメータ

パラメータ	説明
self	AnyData。

GETTYPENAME メンバー・ファンクション

このファンクションは、AnyData の完全修飾名を取得します。

AnyData が組込みタイプに基づいている場合、このファンクションは NUMBER などを戻します。

ユーザー定義型に基づいている場合、このファンクションは <schema_name>.<type_name>（SCOTT.FOO など）を戻します。

AnyData が一時的な匿名タイプに基づいている場合、このファンクションは NULL を戻します。

構文

```
MEMBER FUNCTION GetTypeName(
   self         IN AnyData)
   RETURN       VARCHAR2;
```

パラメータ

表 103-6 GETTYPENAME ファンクションのパラメータ

パラメータ	説明
self	AnyData。

戻り値

AnyData のタイプ名。

GETTYPE メンバー・ファンクション

このファンクションは、AnyData の typecode を取得します。

構文

```
MEMBER FUNCTION GetType(
   self          IN AnyData,
   typ           OUT NOCOPY AnyType)
   RETURN        PLS_INTEGER;
```

パラメータ

表 103-7 GETTYPE ファンクションのパラメータ

パラメータ	説明
self	AnyData。
typ	AnyData に対応する AnyType。ユーザー定義型を表していない場合は、NULL の場合があります。

戻り値

AnyData のタイプに対応する typecode。

GET メンバー・ファンクション

これらのファンクションは、現行のデータ値（適切なタイプのデータ値）を取得します。

現在のデータ値のタイプは、アクセスするモードにより異なります。つまり、PieceWise() コールを起動しているかどうかにより異なります。

PieceWise() がコールされていない場合は、AnyData 全体にアクセスし、データ値のタイプは AnyData のタイプと一致する必要があります。

PieceWise() がコールされている場合、AnyData に個別にアクセスしています。データ値のタイプは、現在の位置での属性のタイプ（またはコレクション要素）に一致する必要があります。

構文

```
MEMBER FUNCTION GetNumber(
   self         IN AnyData,
   num          OUT NOCOPY NUMBER)
   RETURN       PLS_INTEGER;
  MEMBER FUNCTION GetDate(
   self         IN AnyData,
   dat          OUT NOCOPY DATE)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetChar(
   self         IN AnyData,
   c            OUT NOCOPY CHAR)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetVarchar(
   self         IN AnyData,
   c            OUT NOCOPY VARCHAR)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetVarchar2(
   self         IN AnyData,
   c            OUT NOCOPY VARCHAR2)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetRaw(
   self         IN AnyData,
   r            OUT NOCOPY RAW)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetBlob(
   self         IN AnyData,
   b            OUT NOCOPY BLOB)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetClob(
   self         IN AnyData,
   c            OUT NOCOPY CLOB)
   RETURN       PLS_INTEGER;
```

```
MEMBER FUNCTION GetBfile(
   self         IN AnyData,
   b            OUT NOCOPY BFILE)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetObject(
   self         IN AnyData,
   obj          OUT NOCOPY "<object_type>")
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetRef(
   self         IN AnyData,
   rf           OUT NOCOPY REF "<object_type>")
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetCollection(
   self         IN AnyData,
   col          OUT NOCOPY "<collection_type>")
   RETURN       PLS_INTEGER;
```

パラメータ

表 103-8 GET* ファンクションのパラメータ

パラメータ	説明
self	AnyData。
num	取得する数値など。

戻り値

DBMS_TYPES.SUCCESS または DBMS_TYPES.NO_DATA

PieceWise() がコレクションに対してすでにコールされている場合にのみ戻り値が関連します。このような場合、全要素にアクセスしたときに DBMS_TYPES.NO_DATA はコレクションの終了を示します。

例外

DBMS_TYPES.type_mismatch: 予想されたタイプが渡されたタイプと異なる場合。

DBMS_TYPES.invalid_parameters: 無効なパラメータ（作成プロセスのこの時点での数値の追加が適切ではない場合）。

DBMS_TYPES.incorrect_usage: 不適切な使用時。

104

ANYDATASET タイプ

ANYDATASET タイプには、指定のタイプの説明およびそのタイプの一連のデータ・インスタンスが含まれています。ANYDATASET は、必要に応じて、データベースに永続的に格納したり、自己記述型のデータ・セットと通信するインタフェース・パラメータとして使用できます。このデータ・セットは、すべて特定のタイプに属します。

この章では、次の項目について説明します。

構成

AnyDataSet は、値ごとに順番に構成する必要があります。

AnyDataSet タイプの各データ・インスタンスについて、AddInstance() ファンクションを起動する必要があります。これにより、AnyDataSet に新規データ・インスタンスが追加されます。次に、Set*() をコールして値をすべて設定できます。

構成およびアクセスのモードは、PieceWise() へのコールを実行して属性またはコレクション要素単位で変更できます。

- AnyDataSet のタイプが TYPECODE_OBJECT である場合、後続の Set*() コールを使用して個々の属性が設定されます。アクセスについても同様です。
- 現行のデータ値のタイプがコレクション・タイプである場合は、後続の Set*() コールを使用して、個々のコレクション要素が設定されます。アクセスについても同様です。このコールは AnyData タイプに対して定義される AnyData.PieceWise() コールに非常に似ています。

オブジェクト・タイプまたはコレクション・タイプのネストされた属性（最上位レベルを除く）の個別の構成またはアクセスはサポートされていません。

構成プロセスを終了するには、EndCreate() をコールする必要があります。この後、アクセス・コールを実行できます。

ANYDATASET サブプログラムの要約

表 104-1 ANYDATASET サブプログラム

サブプログラム	説明
「BEGINCREATE 静的プロシージャ」104-3 ページ	AnyDataSet は、値ごとに順番に構成する必要があります。
「BEGINCREATE 静的プロシージャ」104-3 ページ	指定の ANYTYPE の一連のデータ値を作成できる AnyDataSet を新規に作成します。
「ADDINSTANCE メンバー・プロシージャ」104-4 ページ	これにより、AnyDataSet に新規データ・インスタンスを追加します。
「PIECEWISE メンバー・プロシージャ」104-5 ページ	構成モード、属性となるデータ値のアクセスを一度に設定します（データ値が TYPECODE_OBJECT の場合）。
「SET* メンバー・プロシージャ」104-6 ページ	現行のデータ値を設定します。
「ENDCREATE メンバー・プロシージャ」104-8 ページ	AnyDataSet の作成を終了します。このコール以降は、その他の作成ファンクションをコールすることはできません。

表 104-1 ANYDATASET サブプログラム（続き）

サブプログラム	説明
「GETTYPENAME メンバー・ファンクション」104-8 ページ	AnyDataSet でデータ・インスタンスのタイプを説明する AnyType を取得します。
「GETTYPE メンバー・ファンクション」104-9 ページ	現行のデータ値（適切なタイプのデータ値）を取得します。
「GETINSTANCE メンバー・ファンクション」104-10 ページ	AnyDataSet の次のインスタンスを取得します。
「GET* メンバー・ファンクション」104-11 ページ	現行のデータ値（適切なタイプのデータ値）を取得します。
「GETCOUNT メンバー・ファンクション」104-13 ページ	AnyDataSet のデータ・インスタンスの数を取得します。

BEGINCREATE 静的プロシージャ

このプロシージャは、指定の ANYTYPE の一連のデータ値を作成できる AnyDataSet を新規に作成します。

構文

```
STATIC PROCEDURE BeginCreate(
   typecode      IN PLS_INTEGER,
   dtype         IN OUT NOCOPY AnyType,
   aset          OUT NOCOPY AnyDataSet);
```

パラメータ

表 104-2 BEGINCREATE プロシージャのパラメータ

パラメータ	説明
typecode	AnyDataSet のタイプに対する typecode。
dtype	データ値のタイプ。このパラメータは、TYPECODE_OBJECT、Collection typecodes などのユーザー定義型には必須です。
aset	構成される AnyDataSet。

例外

DBMS_TYPES.invalid_parameters:dtype は無効です（完全に構成されていないなど）。

ADDINSTANCE メンバー・プロシージャ

このプロシージャは、AnyDataSet に新規データ・インスタンスを追加します。

構文

```
MEMBER PROCEDURE AddInstance(
   self          IN OUT NOCOPY AnyDataSet);
```

パラメータ

表 104-3 ADDINSTANCE プロシージャのパラメータ

パラメータ	説明
self	構成される AnyDataSet。

例外

DBMS_TYPES.invalid_parameters: 無効なパラメータ。
DBMS_TYPES.incorrect_usage: 不適切な使用時。

使用上の注意

データ・インスタンスは順番に追加する必要があります。新しいデータ・インスタンスを追加する前に、前のデータ・インスタンスが完全に構成されているか、NULL に設定されている必要があります。

このコールを実行しても、構成が個別モードで自動的に設定されることはありません。個別にインスタンスを構成するには、PieceWise() を明示的にコールする必要があります。

PIECEWISE メンバー・プロシージャ

このプロシージャは、構成モード、属性となるデータ値のアクセスを一度に設定します（データ値が TYPECODE_OBJECT の場合）。

構成モード、コレクション要素となるデータ値のアクセスを一度に設定します（データ値がコレクション・タイプの場合）。このコールが実行されると、後続の Set*() および Get*() へのコールにより順番に個別の属性またはコレクション要素を取得します。

構文

```
MEMBER PROCEDURE PieceWise(
   self         IN OUT NOCOPY AnyDataSet);
```

パラメータ

表 104-4 PIECEWISE プロシージャのパラメータ

パラメータ	説明
self	構成される AnyDataSet。

例外

DBMS_TYPES.invalid_parameters: 無効なパラメータ。
DBMS_TYPES.incorrect_usage: 不適切な使用時。

使用上の注意

このコールの実行前は、現行のデータ値は、OBJECT または COLLECTION タイプである必要があります。埋め込まれたオブジェクト・タイプの属性またはネストされたコレクションの個別の構成またはアクセスはサポートされていません。

SET* メンバー・プロシージャ

このプロシージャは、現行のデータ値を設定します。

現行のデータ値のタイプは、構成するモードにより異なります（つまり、PieceWise() コールを起動した方法により異なります）。PieceWise() がコールされていない場合は、現行のデータ・タイプが AnyDataSet のタイプである必要があります。PieceWise() がコールされている場合は、現在の位置での属性のタイプにします。

構文

```
MEMBER PROCEDURE SetNumber(
   self              IN OUT NOCOPY AnyDataSet,
   num               IN NUMBER,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetDate(
   self              IN OUT NOCOPY AnyDataSet,
   dat               IN DATE,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetChar(
   self              IN OUT NOCOPY AnyDataSet,
   c                 IN CHAR,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetVarchar(
   self              IN OUT NOCOPY AnyDataSet,
   c                 IN VARCHAR,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetVarchar2(
   self              IN OUT NOCOPY AnyDataSet,
   c                 IN VARCHAR2,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetRaw(
   self              IN OUT NOCOPY AnyDataSet,
   r                 IN RAW,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetBlob(
   self              IN OUT NOCOPY AnyDataSet,
   b                 IN BLOB,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetClob(
   self              IN OUT NOCOPY AnyDataSet,
   c                 IN CLOB,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetBfile(
   self              IN OUT NOCOPY AnyDataSet,
   b                 IN BFILE,
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetObject(
```

```
   self              IN OUT NOCOPY AnyDataSet,
   obj               IN "<object_type>",
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetRef(
   self              IN OUT NOCOPY AnyDataSet,
   rf                IN REF "<object_type>",
   last_elem boolean DEFAULT FALSE);
MEMBER PROCEDURE SetCollection(
   self              IN OUT NOCOPY AnyDataSet,
   col               IN "<collection_type>",
   last_elem boolean DEFAULT FALSE);
```

パラメータ

表 104-5 SET* プロシージャのパラメータ

パラメータ	説明
self	アクセスされる AnyDataSet。
num	設定する数値など。
last_elem	PieceWise() がコレクションに対してすでにコールされている場合にのみ関連します。コレクションの最後の要素の場合は TRUE、そうでない場合は FALSE に設定します。

例外

- DBMS_TYPES.invalid_parameters: 無効なパラメータ（作成プロセスのこの時点での数値の追加が適切ではない場合）。
- DBMS_TYPES.incorrect_usage: 不適切な使用時。
- DBMS_TYPES.type_mismatch: 予想されたタイプが渡されたタイプと異なる場合。

ENDCREATE メンバー・プロシージャ

このプロシージャは、AnyDataSet の作成を終了します。このコール以降は、その他の作成ファンクションをコールすることはできません。

構文

```
MEMBER PROCEDURE EndCreate(
   self              IN OUT NOCOPY AnyDataSet);
```

パラメータ

表 104-6 ENDCREATE プロシージャのパラメータ

パラメータ	説明
self	構成される AnyDataSet。

GETTYPENAME メンバー・ファンクション

このプロシージャは、AnyDataSet の完全修飾タイプ名を取得します。

AnyDataSet が組込みに基づいている場合、このファンクションは NUMBER などを戻します。

ユーザー定義型に基づいている場合、このファンクションは <schema_name>.<type_name>（SCOTT.FOO など）を戻します。

AnyDataSet が一時的な匿名タイプに基づいている場合、このファンクションは NULL を戻します。

構文

```
MEMBER FUNCTION GetTypeName(
   self           IN AnyDataSet)
   RETURN         VARCHAR2;
```

パラメータ

表 104-7 GETTYPENAME ファンクションのパラメータ

パラメータ	説明
self	構成される AnyDataSet。

戻り値

AnyDataSet のタイプ名。

GETTYPE メンバー・ファンクション

AnyDataSet でデータ・インスタンスのタイプを説明する AnyType を取得します。

構文

```
MEMBER FUNCTION GetType(
   self            IN AnyDataSet,
   typ             OUT NOCOPY AnyType)
   RETURN          PLS_INTEGER;
```

パラメータ

表 104-8 GETTYPE ファンクションのパラメータ

パラメータ	説明
self	AnyDataSet。
typ	AnyData に対応する AnyType。ユーザー定義ファンクションを表していない場合は、NULL になることがあります。

戻り値

AnyData のタイプに対応する typecode。

GETINSTANCE メンバー・ファンクション

このファンクションは、AnyDataSet の次のインスタンスを取得します。AnyDataSet のインスタンスへの順次アクセスのみが許可されます。このファンクションのコール後、Get*() ファンクションを AnyDataSet で起動して現在のインスタンスにアクセスできます。Get*() コールの前に PieceWise() がコールされると、個々の属性（またはコレクション要素）にアクセスできます。

AnyDataSet が完全に作成される前にこのファンクションを起動すると、エラーが発生します。

構文

```
MEMBER FUNCTION GetInstance(
   self           IN OUT NOCOPY AnyDataSet)
   RETURN         PLS_INTEGER;
```

パラメータ

表 104-9 GETINSTANCE ファンクションのパラメータ

パラメータ	説明
self	アクセスされる AnyDataSet。

戻り値

DBMS_TYPES.SUCCESS または DBMS_TYPES.NO_DATA

DBMS_TYPES.NO_DATA は、AnyDataSet の終わり（すべてのインスタンスがアクセスしたこと）を示します。

使用上の注意

このファンクションは、最初のインスタンスにアクセスする前にコールしてください。

GET* メンバー・ファンクション

これらのファンクションは、現行のデータ値（適切なタイプのデータ値）を取得します。

現行のデータ値のタイプは、アクセスするモードにより異なります（つまり、PieceWise() コールを起動した方法により異なります）。PieceWise() がコールされていない場合は、インスタンスに完全にアクセスし、データ値のタイプは AnyDataSet のタイプと一致する必要があります。

PieceWise() がコールされている場合は、インスタンスに個別にアクセスします。データ値のタイプは、現在の位置での属性のタイプ（またはコレクション要素）に一致する必要があります。

構文

```
MEMBER FUNCTION GetNumber(
   self         IN AnyDataSet,
   num          OUT NOCOPY NUMBER)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetDate(
   self         IN AnyDataSet,
   dat          OUT NOCOPY DATE)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetChar(
   self         IN AnyDataSet,
   c            OUT NOCOPY CHAR)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetVarchar(
   self         IN AnyDataSet,
   c            OUT NOCOPY VARCHAR)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetVarchar2(
   self         IN AnyDataSet,
   c            OUT NOCOPY VARCHAR2)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetRaw(
   self         IN AnyDataSet,
   r            OUT NOCOPY RAW)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetBlob(
   self         IN AnyDataSet,
   b            OUT NOCOPY BLOB)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetClob(
   self         IN AnyDataSet,
   c            OUT NOCOPY CLOB)
   RETURN       PLS_INTEGER;
MEMBER FUNCTION GetBfile(
```

```
   self        IN AnyDataSet,
   b           OUT NOCOPY BFILE)
   RETURN      PLS_INTEGER;
MEMBER FUNCTION GetObject(
   self        IN AnyDataSet,
   obj         OUT NOCOPY "<object_type>")
   RETURN      PLS_INTEGER;
MEMBER FUNCTION GetRef(
   self        IN AnyDataSet,
   rf          OUT NOCOPY REF "<object_type>")
   RETURN      PLS_INTEGER;
MEMBER FUNCTION GetCollection(
   self        IN AnyDataSet,
   col         OUT NOCOPY "<collection_type>")
   RETURN      PLS_INTEGER;
```

パラメータ

表 104-10 GET* プロシージャのパラメータ

パラメータ	説明
self	アクセスされる AnyDataSet。
num	取得する数値など。

戻り値

DBMS_TYPES.SUCCESS または DBMS_TYPES.NO_DATA

PieceWise() がコレクションに対してすでにコールされている場合にのみ戻り値が関連します。このような場合、全要素にアクセスしたときに DBMS_TYPES.NO_DATA はコレクションの終了を示します。

例外

DBMS_TYPES.invalid_parameters: 無効なパラメータ（作成プロセスのこの時点での数値の追加が適切ではない場合）。

DBMS_TYPES.incorrect_usage: 不適切な使用。

DBMS_TYPES.type_mismatch: 予想されたタイプが渡されたタイプと異なる場合。

GETCOUNT メンバー・ファンクション

このファンクションは、AnyDataSet のデータ・インスタンスの数を取得します。

構文

```
MEMBER FUNCTION GetCount(
   self         IN AnyDataSet)
   RETURN       PLS_INTEGER;
```

パラメータ

表 104-11 GETCOUNT ファンクションのパラメータ

パラメータ	説明
self	アクセスされる AnyDataSet。

戻り値

データ・インスタンスの数。

105

ANYTYPE タイプ

ANYTYPE には、名前がついているかどうかにかかわらず、オブジェクト・タイプおよびコレクション・タイプなどの永続的な SQL タイプの説明を含めることができます。一時的なタイプの説明を新しく構成する場合にも使用できます。

CREATE TYPE 文を使用して作成できるのは、新規の永続タイプのみです。ANYTYPE インタフェースを使用して作成できるのは、新規の一時タイプのみです。

この章では、次の項目について説明します。

ANYTYPE サブプログラムの要約

表 105-1 ANYTYPE サブプログラム

サブプログラム	説明
「BEGINCREATE 静的プロシージャ」105-3 ページ	一時的なタイプの説明の作成に使用できる新規の ANYTYPE のインスタンスを作成します。
「SETINFO メンバー・プロシージャ」105-4 ページ	COLLECTION または組込みタイプの構成に必要な追加情報を設定します。
「ADDATTR メンバー・プロシージャ」105-5 ページ	DBMS_TYPES.TYPECODE_OBJECT の typecode である ANYTYPE に属性を追加します。
「ENDCREATE メンバー・プロシージャ」105-6 ページ	一時的な AnyType の作成を終了します。このコール以降は、その他の作成ファンクションをコールすることはできません。
「GETPERSISTENT 静的ファンクション」105-7 ページ	CREATE TYPE SQL 文を使用して以前に作成された永続的なタイプに対応する AnyType を戻します。
「GETINFO メンバー・ファンクション」105-8 ページ	AnyType のタイプ情報を取得します。
「GETATTRELEMINFO メンバー・ファンクション」105-9 ページ	タイプの属性のタイプ情報を取得します（TYPECODE_OBJECT である場合）。*self* パラメータがコレクション・タイプのパラメータである場合、コレクションの要素タイプのタイプ情報を取得します。

BEGINCREATE 静的プロシージャ

このプロシージャは、一時的なタイプの説明の作成に使用できる新規の ANYTYPE のインスタンスを作成します。

構文

```
STATIC PROCEDURE BEGINCREATE(
   typecode       IN PLS_INTEGER,
   atype          OUT NOCOPY AnyType);
```

パラメータ

表 105-2 BEGINCREATE プロシージャのパラメータ

パラメータ	説明
typecode	DBMS_TYPES パッケージの定数を使用します。 ユーザー定義型の typecode: DBMS_TYPES.TYPECODE_OBJECT、 DBMS_TYPES.TYPECODE_VARRAY または DBMS_TYPES.TYPECODE_TABLE 組込みタイプの typecode: DBMS_TYPES.TYPECODE_NUMBER など
atype	一時的なタイプの AnyType。

SETINFO メンバー・プロシージャ

このプロシージャは、COLLECTION または組込みタイプの構成に必要な追加情報を設定します。

構文

```
MEMBER PROCEDURE SetInfo(
   self          IN OUT NOCOPY AnyType,
   prec          IN PLS_INTEGER,
   scale         IN PLS_INTEGER,
   len           IN PLS_INTEGER,
   csid          IN PLS_INTEGER,
   csfrm         IN PLS_INTEGER,
   atype         IN ANYTYPE DEFAULT NULL,
   elem_tc       IN PLS_INTEGER DEFAULT NULL,
   elem_count    IN PLS_INTEGER DEFAULT 0);
```

パラメータ

表 105-3 SETINFO プロシージャのパラメータ

パラメータ	説明
self	構成中の一時的な ANYTYPE。
prec、scale（オプション）	typecode が NUMBER を表す場合に必要です。 精度およびスケールを指定します。指定しない場合は無視されます。
len（オプション）	typecode が RAW、CHAR、VARCHAR または VARCHAR2 タイプを表す場合に必要です。長さを指定します。
csid、csfrm（オプション）	typecode が CHAR、VARCHAR、VARCHAR2 または CFILE などの文字情報を必要とするタイプを表す場合に必要です。
atype（オプション）	コレクション要素の typecode が TYPECODE_OBJECT などのユーザー定義型である場合に必要です。また、TYPECODE_REF などのユーザー定義型情報を必要とする組込みタイプにも必要です。それ以外の場合は、このパラメータは必要ありません。
コレクション・タイプに必要なパラメータは次のとおりです。	
elem_tc	DBMS_TYPES パッケージのコレクション要素の typecode である必要があります。
elem_count	self がネストした表（TYPECODE_TABLE）を表す場合は elem_count に 0（ゼロ）を渡します。self が VARRAY を表す場合は、コレクションの件数を渡します。

例外

- DBMS_TYPES.invalid_parameter: 無効なパラメータ（typecode、typeinfo）
- DBMS_TYPES.incorrect_usage: 不適切な使用（EndCreate() のコール後にコールできないなど）

使用上の注意

永続的なユーザー定義型を表す AnyType にこのファンクションをコールするとエラーが発生します。

ADDATTR メンバー・プロシージャ

このプロシージャは、DBMS_TYPES.TYPECODE_OBJECT の typecode である AnyType に属性を追加します。

構文

```
MEMBER PROCEDURE AddAttr(
   self          IN OUT NOCOPY AnyType,
   aname         IN VARCHAR2,
   typecode      IN PLS_INTEGER,
   prec          IN PLS_INTEGER,
   scale         IN PLS_INTEGER,
   len           IN PLS_INTEGER,
   csid          IN PLS_INTEGER,
   csfrm         IN PLS_INTEGER,
   attr_type     IN ANYTYPE DEFAULT NULL);
```

パラメータ

表 105-4 ADDATTR プロシージャのパラメータ

パラメータ	説明
self	構成中の一時的な AnyType。DBMS_TYPES.TYPECODE_OBJECT タイプである必要があります。
aname（オプション）	属性名。NULL を指定できます。
typecode	属性の typecode。組込みまたはユーザー定義の typecode（DBMS_TYPES パッケージ）。
prec、scale（オプション）	typecode が NUMBER を表す場合に必要です。精度およびスケールを指定します。指定しない場合は無視されます。

表 105-4 ADDATTR プロシージャのパラメータ（続き）

パラメータ	説明
len（オプション）	typecode が RAW、CHAR、VARCHAR または VARCHAR2 タイプを表す場合に必要です。長さを指定します。
csid、csfrm（オプション）	typecode が CHAR、VARCHAR、VARCHAR2 または CFILE などの文字情報を必要とするタイプを表す場合に必要です。
attr_type（オプション）	ユーザー定義型に対応する AnyType。このパラメータは、属性がユーザー定義型の場合に必要です。

例外

- DBMS_TYPES.invalid_parameters: 無効なパラメータ（typecode、typeinfo）
- DBMS_TYPES.incorrect_usage: 不適切な使用（EndCreate() のコール後にコールできないなど）

ENDCREATE メンバー・プロシージャ

このプロシージャは、一時的な AnyType の作成を終了します。このコール以降は、その他の作成ファンクションをコールすることはできません。

構文

```
MEMBER PROCEDURE EndCreate(
   self           IN OUT NOCOPY AnyType);
```

パラメータ

表 105-5 ENDCREATE プロシージャのパラメータ

パラメータ	説明
self	構成中の一時的な AnyType。

GETPERSISTENT 静的ファンクション

このプロシージャは、CREATE TYPE SQL 文を使用して以前に作成された永続的なタイプに対応する AnyType を戻します。

構文

```
STATIC FUNCTION GetPersistent(
   schema_name      IN VARCHAR2,
   type_name        IN VARCHAR2,
   version          IN VARCHAR2 DEFAULT NULL)
   RETURN           AnyType;
```

パラメータ

表 105-6 GETPERSISTENT ファンクションのパラメータ

パラメータ	説明
schema_name	タイプのスキーマ名
type_name	タイプ名
version	タイプのバージョン

戻り値

CREATE TYPE SQL 文を使用して以前に作成された永続的なタイプに対応する AnyType。

GETINFO メンバー・ファンクション

このファンクションは、AnyType のタイプ情報を取得します。

構文

```
MEMBER FUNCTION GetInfo (
   self        IN AnyType,
   prec        OUT PLS_INTEGER,
   scale       OUT PLS_INTEGER,
   len         OUT PLS_INTEGER,
   csid        OUT PLS_INTEGER,
   csfrm       OUT PLS_INTEGER,
   schema_name OUT VARCHAR2,
   type_name   OUT VARCHAR2,
   version     OUT varchar2,
   count       OUT PLS_INTEGER)
   RETURN      PLS_INTEGER;
```

パラメータ

表 105-7 GETINFO ファンクションのパラメータ

パラメータ	説明
self	AnyType。
prec、scale	typecode が NUMBER を表す場合。精度およびスケールを指定します。指定しない場合は無視されます。
len	typecode が RAW、CHAR、VARCHAR または VARCHAR2 タイプを表す場合。長さを指定します。
csid、csfrm	typecode が CHAR、VARCHAR、VARCHAR2 または CFILE などの文字情報を必要とするタイプを表す場合。
schema_name、type_name、version	タイプのスキーマ（永続的な場合）、タイプ名およびバージョン。
count	self が VARRAY である場合は、VARRAY 件数が指定されます。self が TYPECODE_OBJECT である場合は、属性の数が指定されます。

戻り値

self の typecode。

例外

- DBMS_TYPES.invalid_parameters: 無効なパラメータ（位置が境界を越えているか、AnyType が適切に構成されていない）。

GETATTRELEMINFO メンバー・ファンクション

このファンクションは、タイプの属性のタイプ情報を取得します（TYPECODE_OBJECT である場合）。self パラメータがコレクション・タイプのパラメータである場合、コレクションの要素タイプのタイプ情報を取得します。

構文

```
MEMBER FUNCTION GetAttrElemInfo (
   self           IN AnyType,
   pos            IN PLS_INTEGER,
   prec           OUT PLS_INTEGER,
   scale          OUT PLS_INTEGER,
   len            OUT PLS_INTEGER,
   csid           OUT PLS_INTEGER,
   csfrm          OUT PLS_INTEGER,
   attr_elt_type  OUT ANYTYPE
   aname          OUT VARRCHAR2)
   RETURN         PLS_INTEGER;
```

パラメータ

表 105-8 GETATTRELEMINFO ファンクションのパラメータ

パラメータ	説明
self	AnyType。
pos	self が TYPECODE_OBJECT である場合は、属性の位置が 1 から指定されます。そうでない場合は無視されます。
prec、scale	属性またはコレクション要素の typecode が NUMBER を表す場合。精度およびスケールを指定します。指定しない場合は無視されます。
len	typecode が RAW、CHAR、VARCHAR または VARCHAR2 タイプを表す場合。長さを指定します。
csid、csfrm	typecode が CHAR、VARCHAR、VARCHAR2 または CFILE などの文字情報を必要とするタイプを表す場合。キャラクタ・セット ID、キャラクタ・セット・フォームを指定します。

表 105-8 GETATTRELEMINFO ファンクションのパラメータ（続き）

パラメータ	説明
attr_elt_type	属性またはコレクション要素の typecode がユーザー定義型を表す場合、それに対応する AnyType を戻します。その後、attr_elt_type を記述できます。
aname	属性名（オブジェクト・タイプの属性の場合。そうでない場合は NULL）。

戻り値

属性またはコレクション要素の typecode。

例外

- DBMS_TYPES.invalid_parameters: 無効なパラメータ（位置が境界を越えているか、AnyType が適切に構成されていない）。

106

アドバンスト・キューイング・タイプ

この章では、次のアドバンスト・キューイング（AQ）パッケージで使用するために設計されたタイプについて説明します。

- DBMS_AQ
- DBMS_AQADM

この章では、次の項目について説明します。

関連項目：

- AQ の使用方法については、『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』を参照してください。
- DBMS_AQ パッケージおよび DBMS_AQADM パッケージの詳細は、第 5 章「DBMS_AQ」および第 6 章「DBMS_AQADM」を参照してください。

アドバンスト・キューイング・タイプ

表 106-1　アドバンスト・キューイング・タイプ

タイプ	説明
「AQ$_AGENT タイプ」106-3 ページ	メッセージのプロデューサまたはコンシューマを識別します。
「AQ$_AGENT_LIST_T タイプ」106-3 ページ	DBMS_AQ.LISTEN がリスニングするエージェントのリストを識別します。
「AQ$_DESCRIPTOR タイプ」106-3 ページ	通知時に AQ PL/SQL のコールバックが受信する AQ 記述子を指定します。
「AQ$_POST_INFO タイプ」106-4 ページ	メッセージの転送先となる匿名サブスクリプションを指定します。
「AQ$_POST_INFO_LIST タイプ」106-5 ページ	メッセージの転送先となる匿名サブスクリプションのリストを識別します。
「AQ$_RECIPIENT_LIST_T タイプ」106-5 ページ	メッセージを受信するエージェントのリストを識別します。
「AQ$_REG_INFO タイプ」106-5 ページ	メッセージのプロデューサまたはコンシューマを識別します。
「AQ$_REG_INFO_LIST タイプ」106-7 ページ	キューに対する登録のリストを識別します。
「AQ$_SUBSCRIBER_LIST_T タイプ」106-7 ページ	キューをサブスクライブするサブスクライバのリストを識別します。
「DEQUEUE_OPTIONS_T タイプ」106-8 ページ	デキュー操作で使用可能なオプションを指定します。
「ENQUEUE_OPTIONS_T タイプ」106-11 ページ	エンキュー操作で使用可能なオプションを指定します。
「MESSAGE_PROPERTIES_T タイプ」106-12 ページ	個々のメッセージの管理で AQ が使用する情報を記述します。

AQ$_AGENT タイプ

メッセージのプロデューサまたはコンシューマを識別します。

構文

```
TYPE SYS.AQ$_AGENT IS OBJECT (
   name       VARCHAR2(30),
   address    VARCHAR2(1024),
   protocol   NUMBER  DEFAULT 0);
```

属性

表 106-2 AQ$_AGENT の属性

属性	説明
name	メッセージのプロデューサまたはコンシューマの名前。名前は、予約語に関して、『Oracle9*i* SQL リファレンス』のオブジェクト名ガイドラインに従う必要があります。
address	受信者のプロトコル固有のアドレス。プロトコルが 0 の場合、アドレスは [*schema.*]*queue*[@*dblink*] の形式になります。 たとえば、サイト dbs1.net の hr キューにある emp_messages という名前のキューのアドレスは、次のようになります。 hr.emp_messages@dbs1.net
protocol	アドレスを解釈し、メッセージを伝播するためのプロトコル。

AQ$_AGENT_LIST_T タイプ

DBMS_AQ.LISTEN がリスニングするエージェントのリストを識別します。

関連項目： 106-3 ページ「AQ$_AGENT タイプ」

構文

```
TYPE SYS.AQ$_AGENT_LIST_T IS TABLE OF SYS.AQ$_AGENT
   INDEX BY BINARY INTEGER;
```

AQ$_DESCRIPTOR タイプ

通知時に AQ PL/SQL のコールバックが受信する AQ 記述子を指定します。

関連項目： 106-12 ページ「MESSAGE_PROPERTIES_T タイプ」

構文

```
TYPE SYS.AQ$_DESCRIPTOR IS OBJECT (
   queue_name      VARCHAR2(61),
   consumer_name   VARCHAR2(30),
   msg_id          RAW(16),
   msg_prop        MSG_PROP_T);
```

属性

表 106-3 AQ$_DESCRIPTOR の属性

属性	説明
queue_name	結果として通知されたメッセージがエンキューされたキューの名前。
consumer_name	マルチ・コンシューマ・キューのコンシューマ名。
msg_id	メッセージの ID 番号。
msg_prop	MSG_PROP_T タイプで指定されるメッセージ・プロパティ。

AQ$_POST_INFO タイプ

メッセージの転送先となる匿名サブスクリプションを指定します。

構文

```
TYPE SYS.AQ$_POST_INFO IS OBJECT (
   name        VARCHAR2(128),
   namespace   NUMBER,
   payload     RAW(2000)  DEFAULT NULL);
```

属性

表 106-4 AQ$_POST_INFO の属性

属性	説明
name	転送先となる匿名サブスクリプションの名前。
namespace	その他のアプリケーションから DBMS_AQ.POST または OCISubscriptionPost() を介して通知を受信するには、ネームスペースが DBMS_AQ.NAMESPACE_ANONYMOUS であることが必要です。
payload	匿名サブスクリプションに転送されるペイロード。

AQ$_POST_INFO_LIST タイプ

メッセージの転送先となる匿名サブスクリプションのリストを識別します。

関連項目： 106-4 ページ「AQ$_POST_INFO タイプ」

構文

```
TYPE SYS.AQ$_POST_INFO_LIST AS VARRAY(1024) OF SYS.AQ$_POST_INFO;
```

AQ$_RECIPIENT_LIST_T タイプ

メッセージを受信するエージェントのリストを識別します。このタイプは、キューが複数デキュー可能な場合のみ使用されます。

関連項目： 106-3 ページ「AQ$_AGENT タイプ」

構文

```
TYPE SYS.AQ$_RECIPIENT_LIST_T IS TABLE OF SYS.AQ$_AGENT
   INDEX BY BINARY_INTEGER;
```

AQ$_REG_INFO タイプ

キューに対する通知の登録者に関する情報を指定します。

構文

```
TYPE SYS.AQ$_REG_INFO IS OBJECT (
   name        VARCHAR2(128),
   namespace   NUMBER,
   callback    VARCHAR2(4000),
   context     RAW(2000)  DEFAULT NULL);
```

属性

表 106-5　AQ 登録情報タイプの属性

属性	説明
name	サブスクリプション名を指定します。 単一コンシューマ・キューの登録の場合は *schema.queue* 形式のサブスクリプション名、マルチ・コンシューマ・キューの登録の場合は *schema.queue:consumer_name* 形式の名前となります。
namespace	サブスクリプションのネームスペースを指定します。 AQ キューから通知を受信するには、ネームスペースが DBMS_AQ.NAMESPACE_AQ であることが必要です。 その他のアプリケーションから DBMS_AQ.POST または OCISubscriptionPost() を介して通知を受信するには、ネームスペースが DBMS_AQ.NAMESPACE_ANONYMOUS であることが必要です。

callback	メッセージ通知時に実行されるアクションを指定します。
	HTTP 通知の場合は、次の形式になります。
	`http://www.company.com:8080`
	電子メール通知の場合は、次の形式になります。
	`mailto://xyz@company.com`
	PLSQLCALLBACK プロシージャ用の RAW メッセージ・ペイロードの場合は、次の形式を使用します。
	`plsql://schema.procedure?PR=0`
	PLSQLCALLBACK プロシージャ用に XML に変換されたユーザー定義型のメッセージ・ペイロードの場合は、次の形式を使用します。
	`plsql://schema.procedure?PR=1`

context	コールバック関数に渡されるコンテキストを指定します。

使用上の注意

通知メカニズムとして、OCI、電子メールまたは PL/SQL コールバックを使用できます。非永続キューに対する通知は、表 106-6 に示すように、指定した通知メカニズムおよびキュー・ペイロード・タイプによって異なります。

表 106-6　非永続キュー

キュー・ペイロード・タイプ	指定表示					
	RAW			XML		
	通知メカニズム			通知メカニズム		
	OCI	電子メール	PL/SQL のコールバック	OCI	電子メール	PL/SQL のコールバック
RAW	コールバックはペイロードにおいて RAW データを受信します。	サポートされていません。	PL/SQL コールバックはペイロードにおいて RAW データを受信します。	コールバックはペイロードにおいて XML データを受信します。	XML データは IDAP メッセージにフォーマットされ、登録済みの電子メール・アドレスに送信されます。	PL/SQL コールバックはペイロードにおいて XML データを受信します。
ユーザー定義型	サポートされていません。	サポートされていません。	サポートされていません。	コールバックはペイロードにおいて XML データを受信します。	XML データは IDAP メッセージにフォーマットされ、登録済みの電子メール・アドレスに送信されます。	PL/SQL コールバックはペイロードにおいて XML データを受信します。

AQ$_REG_INFO_LIST タイプ

キューに対する登録のリストを識別します。

関連項目： 106-5 ページ「AQ$_REG_INFO タイプ」

構文

```
TYPE SYS.AQ$_REG_INFO_LIST AS VARRAY(1024) OF SYS.AQ$_REG_INFO;
```

AQ$_SUBSCRIBER_LIST_T タイプ

キューをサブスクライブするサブスクライバのリストを識別します。

関連項目： 106-3 ページ「AQ$_AGENT タイプ」

構文

```
TYPE SYS.AQ$_SUBSCRIBER_LIST_T IS TABLE OF SYS.AQ$_AGENT
   INDEX BY BINARY_INTEGER;
```

DEQUEUE_OPTIONS_T タイプ

デキュー操作で使用可能なオプションを指定します。

構文

```
TYPE DEQUEUE_OPTIONS_T IS RECORD (
   consumer_name     VARCHAR2(30)    DEFAULT NULL,
   dequeue_mode      BINARY_INTEGER  DEFAULT REMOVE,
   navigation        BINARY_INTEGER  DEFAULT NEXT_MESSAGE,
   visibility        BINARY_INTEGER  DEFAULT ON_COMMIT,
   wait              BINARY_INTEGER  DEFAULT FOREVER,
   msgid             RAW(16)         DEFAULT NULL,
   correlation       VARCHAR2(128)   DEFAULT NULL,
   deq_condition     VARCHAR2(4000)  DEFAULT NULL,
   transformation    VARCHAR2(60)    DEFAULT NULL);
```

属性

表 106-7 DEQUEUE_OPTIONS_T の属性

属性	説明
consumer_name	コンシューマの名前。コンシューマ名が一致するメッセージのみアクセスされます。キューがマルチ・コンシューマ用に設定されていない場合、このフィールドは NULL に設定してください。 キューの安全のため、consumer_name は有効な AQ エージェントであることが必要です。
dequeue_mode	デキューに関連付けるロック動作を指定します。次の値を設定できます。 BROWSE: メッセージのロックを取得せずにメッセージを読み込みます。この仕様は、SELECT 文と同じです。 LOCKED: メッセージを読み込み、その書込みロックを取得します。ロックはトランザクションの継続中有効です。この設定は、SELECT FOR UPDATE 文と同じです。 REMOVE: メッセージを読み込み、そのメッセージを更新または削除します。この設定がデフォルトです。メッセージは保存プロパティに基づいてキュー表に保存されます。 REMOVE_NODATA: メッセージに更新または削除のマークを設定します。メッセージは保存プロパティに基づいてキュー表に保存されます。

表 106-7 DEQUEUE_OPTIONS_T の属性（続き）

属性	説明
navigation	取り出されるメッセージの位置を指定します。最初に、位置が判断されます。次に、検索基準が適用されます。最後に、メッセージが取り出されます。 次の値を設定できます。 NEXT_MESSAGE: 使用可能で検索基準と一致する次のメッセージを取り出します。前のメッセージがメッセージ・グループに属している場合は、検索基準と一致し、そのメッセージ・グループに属している次の使用可能メッセージが取り出されます。この設定がデフォルトです。 NEXT_TRANSACTION: 現行トランザクション・グループの残り（ある場合）をスキップし、次のトランザクション・グループの最初のメッセージを取り出します。この設定は、現行キューに対してメッセージ・グループが使用可能な場合のみ使用できます。 FIRST_MESSAGE: 使用可能で検索基準と一致する最初のメッセージを取り出します。この設定により、位置がキューの先頭に再設定されます。
visibility	新規メッセージを現行トランザクションの一部としてデキューするかどうかを指定します。BROWSE デキュー・モードの使用時は無視されます。 次の値を設定できます。 ON_COMMIT: デキューは現行トランザクションの一部になります。この設定がデフォルトです。 IMMEDIATE: デキュー・メッセージは、現行トランザクションの一部ではありません。独自のトランザクションを構成します。
wait	検索基準と一致する使用可能なメッセージが存在していない場合の待機時間を指定します。 次の値を設定できます。 FOREVER: 使用可能なメッセージが発生するまで待機します。この設定がデフォルトです。 NO_WAIT: 待機しません。 数値 : 待機時間（秒）。
msgid	デキューするメッセージのメッセージ ID を指定します。
correlation	デキューするメッセージの相関 ID を指定します。パーセント記号（%）やアンダースコア（_）などの特殊なパターン一致文字を使用できます。パターンを満たすメッセージが複数ある場合、デキュー順序は未定義です。

表 106-7 DEQUEUE_OPTIONS_T の属性（続き）

属性	説明
deq_condition	メッセージ・プロパティ、メッセージ・データ・プロパティおよびPL/SQL ファンクションに基づいた条件式。 deq_condition は SQL 問合せの WHERE 句と同様の構文を使用して、ブール式で指定されます。このブール式には、メッセージ・プロパティの状態、ユーザー・データのプロパティ（オブジェクト・ペイロードのみ）、および PL/SQL または SQL ファンクション（SQL 問合せの WHERE 句で指定）の状態を組み込むことができます。メッセージ・プロパティには、priority、corrid およびキュー表におけるその他の列が含まれます。 メッセージ・ペイロード（オブジェクト・ペイロード）のデキュー条件を指定するには、句にオブジェクト・タイプの属性を使用します。各属性の前に修飾子として tab.user_data を付加して、ペイロードを格納するキュー表の特定の列を示します。 deq_condition のパラメータは 4000 バイト以内です。
transformation	メッセージをデキューした後に適用される変換を指定します。変換のソース・タイプは、キューのタイプと一致させる必要があります。

ENQUEUE_OPTIONS_T タイプ

エンキュー操作で使用可能なオプションを指定します。

構文

```
TYPE SYS.ENQUEUE_OPTIONS_T IS RECORD (
   visibility            BINARY_INTEGER  DEFAULT ON_COMMIT,
   relative_msgid        RAW(16)         DEFAULT NULL,
   sequence_deviation    BINARY_INTEGER  DEFAULT NULL,
   transformation        VARCHAR2(60)    DEFAULT NULL);
```

属性

表 106-8 ENQUEUE_OPTIONS_T の属性

属性	説明
visibility	エンキュー要求のトランザクション動作を指定します。次の値を設定できます。 ON_COMMIT: エンキューは、現行トランザクションの一部です。操作は、トランザクションがコミットされると完了します。この設定がデフォルトです。 IMMEDIATE: エンキューは、現行トランザクションの一部ではありません。操作は独自のトランザクションを構成します。非永続キューにエンキューするときは、この値のみ許可されます。
relative_msgid	順序逸脱操作で参照されるメッセージのメッセージ ID を指定します。このフィールドは、sequence_deviation に BEFORE が指定されている場合のみ有効です。順序逸脱が指定されていない場合、このパラメータは無視されます。
sequence_deviation	エンキューされるメッセージが、すでにキューにある他のメッセージより前にデキューされる必要があるかどうかを指定します。次の値を設定できます。 BEFORE: メッセージは、relative_msgid で指定されたメッセージより前にエンキューされます。 TOP: メッセージは、他のどのメッセージよりも前にエンキューされます。
transformation	メッセージをエンキューする前に適用される変換を指定します。変換ファンクションの戻りタイプは、キューのタイプと一致させる必要があります。

MESSAGE_PROPERTIES_T タイプ

個々のメッセージの管理で AQ が使用する情報を記述します。これらはエンキュー時に設定され、デキュー時にその値が戻されます。

関連項目： 106-5 ページ「AQ$_RECIPIENT_LIST_T タイプ」

構文

```
TYPE MESSAGE_PROPERTIES_T IS RECORD (
   priority               BINARY_INTEGER  DEFAULT 1,
   delay                  BINARY_INTEGER  DEFAULT NO_DELAY,
   expiration             BINARY_INTEGER  DEFAULT NEVER,
   correlation            VARCHAR2(128)   DEFAULT NULL,
   attempts               BINARY_INTEGER,
   recipient_list         AQ$_RECIPIENT_LIST_T,
   exception_queue        VARCHAR2(51)    DEFAULT NULL,
   enqueue_time           DATE,
   state                  BINARY_INTEGER,
   sender_id              AQ$_AGENT       DEFAULT NULL,
   original_msgid         RAW(16)         DEFAULT NULL);
```

属性

表 106-9 MESSAGE_PROPERTIES_T の属性

属性	説明
priority	メッセージの優先順位を指定します。数値が小さいほど優先順位は高くなります。優先順位には、負数も含めたあらゆる数値を使用できます。
delay	エンキュー・メッセージの遅延を指定します。遅延は、メッセージがデキュー可能になるまでの秒数を表します。msgid によるデキューの場合、delay 操作は上書きされます。遅延設定をしてエンキューされたメッセージは WAITING 状態になり、遅延期間が終了すると READY 状態になります。DELAY 処理では、キュー・モニターを起動する必要があります。delay は、メッセージをエンキューするプロデューサが設定します。 次の値を設定できます。 NO_DELAY: メッセージは、すぐにデキュー可能になります。 数値 : メッセージの遅延秒数。

表 106-9 MESSAGE_PROPERTIES_T の属性（続き）

属性	説明
expiration	メッセージの期限切れまでの時間を指定します。メッセージのデキュー操作可能期間を秒数で指定します。このパラメータは、delay からオフセットされます。期限切れ処理では、キュー・モニターが実行されている必要があります。 次の値を設定できます。 NEVER: メッセージは時間切れになりません。 数値 : メッセージを READY 状態にしておく時間（秒数）。時間切れまでにメッセージがデキューされない場合、メッセージは EXPIRED 状態で例外キューに移されます。
correlation	エンキュー時にメッセージのプロデューサが提供した ID を戻します。
attempts	メッセージのデキューの試行回数を戻します。このパラメータはエンキュー時には設定できません。
recipient_list	このパラメータは、マルチ・コンシューマを許可しているキューに対してのみ有効です。デフォルトの受信者は、キューのサブスクライバです。このパラメータは、デキュー時にコンシューマに戻されません。 タイプ定義の詳細は、106-3 ページの「AQ$_AGENT タイプ」を参照してください。
exception_queue	メッセージが正常に処理されなかった場合のメッセージの移動先キュー名を指定します。 メッセージが自動的に移動されるのは、次の場合です。 ■ キュー作成時に、デキューの試行失敗回数が、DBMS_AQADM.CREATE_QUEUE プロシージャの max_retries パラメータで指定した回数を超えた場合。キューに対する max_retries は、ALL_QUEUES データ・ディクショナリ・ビューで参照できます。 ■ 例外キューのメッセージは、すべて EXPIRED 状態になります。 デフォルトは、キュー表に関連付けられている例外キューです。指定した例外キューが移動時に存在しない場合、そのメッセージは、キュー表に関連付けられているデフォルトの例外キューに移動され、アラート・ファイルに警告ログが記録されます。デフォルトの例外キューが指定されると、デキュー時に NULL 値が戻されます。
enqueue_time	メッセージがエンキューされた時刻を指定します。この値はシステムによって設定され、ユーザーは設定できません。このパラメータはエンキュー時には設定できません。

表 106-9 MESSAGE_PROPERTIES_T の属性（続き）

属性	説明
state	デキュー時にメッセージの状態を指定します。このパラメータはエンキュー時には設定できません。次の状態を設定できます。 0: メッセージは処理可能な状態です。 1: メッセージは遅延の指定秒数に達していません。 2: メッセージは処理され、保存されています。 3: メッセージは例外キューに移されました。
sender_id	アプリケーション指定の送信者 ID を指定します。キューの安全のため、sender_id を指定してメッセージをエンキューする必要があります。
original_msgid	このパラメータは、Oracle AQ がメッセージを伝播するために使用します。

107

JMS タイプ

リリース 2（9.2）では、メンバー・プロシージャとメンバー・ファンクション、および静的ファンクションが既存の JMS PL/SQL タイプ（`aq$_jms_text_message` および `aq$_jms_bytes_message`）に追加されているため、PL/SQL アプリケーションでこれらのタイプの JMS キューを使用できます。aq$_jms_message タイプが追加されているため、様々なタイプの JMS メッセージを同じ AQ キューにエンキューできます。リリース 2（9.2）のメンバー・プロシージャとメンバー・ファンクションを使用すると、PL/SQL からエンキューしたメッセージを OJMS でデキューし、OJMS からエンキューしたメッセージを PL/SQL でデキューできます。

> **関連項目：**『Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング』

この章では、次の項目について説明します。

aq$_jms_message タイプをサポートするための定数

DBMS_AQJMS パッケージには、次の定数が含まれています。

- `JMS_TEXT_MESSAGE  CONSTANT BINARY_INTEGER;`
- `JMS_BYTES_MESSAGE  CONSTANT BINARY_INTEGER;`

詳細は、107-4 ページの「aq$_jms_message タイプ」を参照してください。

JMS タイプの要約

この章では、次の JMS タイプについて説明します。

- aq$_jms_userproperty タイプ
- aq$_jms_userproparray タイプ
- aq$_jms_header タイプ
- aq$_jms_message タイプ
- aq$_jms_text_message タイプ
- aq$_jms_bytes_message タイプ

aq$_jms_userproperty タイプ

このタイプは、各ユーザーが指定した JMS メッセージのユーザー・プロパティを格納するために使用します。

構文

```
TYPE aq$_jms_userproperty AS object(
   name        VARCHAR(100),
   type        INT,
   str_value   VARCHAR(2000),
   num_value   NUMBER,
   java_type   INT);
```

aq$_jms_userproparray タイプ

このタイプは、指定した JMS メッセージに対して、JMS ユーザーが指定したメッセージ・プロパティのリストを格納するために使用します。

構文

```
TYPE aq$_jms_userproparray AS varray(100) of aq$_jms_userproperty;
```

aq$_jms_header タイプ

このタイプは、指定した JMS メッセージに対して、JMS メッセージ・ヘッダー値を格納するために使用します。

構文

```
TYPE aq$_jms_header AS object(
   replyto     sys.aq$_agent,
   type        VARCHAR(100),
   userid      VARCHAR(100),
   appid       VARCHAR(100),
   groupid     VARCHAR(100),
   groupseq    INT,
   properties  aq$_jms_userproparray,
   MEMBER PROCEDURE lookup_property_name (new_property_name IN VARCHAR),
   MEMBER PROCEDURE set_replyto          (replyto           IN sys.aq$_agent),
   MEMBER PROCEDURE set_type             (type              IN VARCHAR),
   MEMBER PROCEDURE set_userid           (userid            IN VARCHAR),
   MEMBER PROCEDURE set_appid            (appid             IN VARCHAR),
   MEMBER PROCEDURE set_groupid          (groupid           IN VARCHAR),
   MEMBER PROCEDURE set_groupseq         (groupseq          IN INT),
   MEMBER PROCEDURE clear_properties,
   MEMBER PROCEDURE set_boolean_property(
      property_name   IN  VARCHAR,
      property_value  IN  BOOLEAN),
   MEMBER PROCEDURE set_byte_property(
     property_name   IN  VARCHAR,
     property_value  IN  INT ),
   MEMBER PROCEDURE set_short_property (
     property_name   IN  VARCHAR,
     property_value  IN  INT ),
   MEMBER PROCEDURE set_int_property (
      property_name   IN  VARCHAR,
      property_value  IN  INT ),
   MEMBER PROCEDURE set_long_property (
      property_name   IN  VARCHAR,
      property_value  IN  NUMBER ),
  MEMBER PROCEDURE set_float_property (
      property_name   IN  VARCHAR,
      property_value  IN  FLOAT ),
  MEMBER PROCEDURE set_double_property (
      property_name   IN  VARCHAR,
      property_value  IN  DOUBLE PRECISION ),
  MEMBER PROCEDURE set_string_property (
      property_name   IN  VARCHAR,
      property_value  IN  VARCHAR ),
```

```
  MEMBER FUNCTION get_replyto  RETURN sys.aq$_agent,
  MEMBER FUNCTION get_type     RETURN VARCHAR,
  MEMBER FUNCTION get_userid   RETURN VARCHAR,
  MEMBER FUNCTION get_appid    RETURN VARCHAR,
  MEMBER FUNCTION get_groupid  RETURN VARCHAR,
  MEMBER FUNCTION get_groupseq RETURN INT,
  MEMBER FUNCTION get_boolean_property (property_name IN  VARCHAR)
    RETURN   BOOLEAN,
  MEMBER FUNCTION get_byte_property    (property_name IN  VARCHAR)
    RETURN   INT,
  MEMBER FUNCTION get_short_property   (property_name IN  VARCHAR)
    RETURN   INT,
  MEMBER FUNCTION get_int_property     (property_name IN  VARCHAR)
    RETURN   INT,
  MEMBER FUNCTION get_long_property    (property_name IN  VARCHAR)
    RETURN   NUMBER,
  MEMBER FUNCTION get_float_property   (property_name IN  VARCHAR)
    RETURN   FLOAT,
  MEMBER FUNCTION get_double_property  (property_name IN  VARCHAR)
    RETURN   DOUBLE PRECISION,
  MEMBER FUNCTION get_string_property  (property_name IN  VARCHAR)
    RETURN   VARCHAR);
```

aq$_jms_message タイプ

このタイプは、JMSText および JMSBytes メッセージ・タイプを格納するために使用するユーザー定義型です。

aq$_jms_message の一部として定義される CONSTRUCT 静的ファンクションは、次のとおりです。

```
STATIC FUNCTION construct ( mtype IN int ) RETURN aq$_jms_message.
```

詳細は、107-29 ページの「CONSTRUCT 静的ファンクション」を参照してください。

構文

```
TYPE aq$_jms_message AS object(
   header        aq$_jms_header,
   senderid      varchar2(100),
   message_type  INT,
   text_len      INT,
   bytes_len     INT,
   text_vc       varchar2(4000),
   bytes_raw     raw(2000),
   text_lob      clob,
   bytes_lob     blob,
```

```
 STATIC FUNCTION construct     (mtype    IN  INT) RETURN aq$_jms_message,
 MEMBER PROCEDURE set_text     (payload  IN  VARCHAR2),
 MEMBER PROCEDURE set_text     (payload  IN  CLOB),
 MEMBER PROCEDURE get_text     (payload  OUT VARCHAR2),
 MEMBER PROCEDURE get_text     (payload  OUT CLOB),
 MEMBER PROCEDURE set_bytes    (payload  IN  RAW),
 MEMBER PROCEDURE set_bytes    (payload  IN  BLOB),
 MEMBER PROCEDURE get_bytes    (payload  OUT RAW),
 MEMBER PROCEDURE get_bytes    (payload  OUT BLOB),
 MEMBER PROCEDURE set_replyto  (replyto  IN  sys.aq$_agent),
 MEMBER PROCEDURE set_type     (type     IN  VARCHAR),
 MEMBER PROCEDURE set_userid   (userid   IN  VARCHAR),
 MEMBER PROCEDURE set_appid    (appid    IN  VARCHAR),
 MEMBER PROCEDURE set_groupid  (groupid  IN  VARCHAR),
 MEMBER PROCEDURE set_groupseq (groupseq IN  INT),
 MEMBER PROCEDURE clear_properties ,
 MEMBER PROCEDURE set_boolean_property(
    property_name   IN      VARCHAR,
    property_value  IN      BOOLEAN),
 MEMBER PROCEDURE set_byte_property(
    property_name   IN      VARCHAR,
    property_value  IN      INT),
 MEMBER PROCEDURE set_short_property(
    property_name   IN      VARCHAR,
    property_value  IN      INT),
 MEMBER PROCEDURE set_int_property(
    property_name   IN      VARCHAR,
    property_value  IN      INT),
MEMBER PROCEDURE set_long_property(
   property_name   IN      VARCHAR,
   property_value  IN      NUMBER),
MEMBER PROCEDURE set_float_property(
   property_name   IN      VARCHAR,
   property_value  IN      FLOAT),
MEMBER PROCEDURE set_double_property(
   property_name   IN      VARCHAR,
   property_value  IN      DOUBLE PRECISION),
MEMBER PROCEDURE set_string_property(
   property_name   IN      VARCHAR,
   property_value  IN      VARCHAR),
MEMBER FUNCTION get_replyto  RETURN sys.aq$_agent,
MEMBER FUNCTION get_type     RETURN VARCHAR,
MEMBER FUNCTION get_userid   RETURN VARCHAR,
MEMBER FUNCTION get_appid    RETURN VARCHAR,
MEMBER FUNCTION get_groupid  RETURN VARCHAR,
MEMBER FUNCTION get_groupseq RETURN INT,
MEMBER FUNCTION get_boolean_property (property_name   IN   VARCHAR)
```

```
  RETURN   BOOLEAN,
 MEMBER FUNCTION get_byte_property    (property_name   IN   VARCHAR)
  RETURN   INT,
 MEMBER FUNCTION get_short_property   (property_name   IN   VARCHAR)
  RETURN   INT,
 MEMBER FUNCTION get_int_property     (property_name   IN   VARCHAR)
  RETURN   INT,
 MEMBER FUNCTION get_long_property    (property_name   IN   VARCHAR)
  RETURN   NUMBER,
 MEMBER FUNCTION get_float_property   (property_name   IN   VARCHAR)
  RETURN   FLOAT,
 MEMBER FUNCTION get_double_property  (property_name   IN   VARCHAR)
  RETURN   DOUBLE PRECISION,
 MEMBER FUNCTION get_string_property  (property_name   IN   VARCHAR)
  RETURN   VARCHAR);
```

aq$_jms_text_message タイプ

このタイプは、AQ キューの JMSText メッセージを格納するために使用するユーザー定義型です。

構文

```
TYPE aq$_jms_text_message AS object(
  header    aq$_jms_header,
  text_len  INT,
  text_vc   varchar2(4000),
  text_lob  clob,
  STATIC FUNCTION construct RETURN aq$_jms_text_message,
  MEMBER PROCEDURE set_text     (payload  IN  VARCHAR2),
  MEMBER PROCEDURE set_text     (payload  IN  CLOB),
  MEMBER PROCEDURE get_text     (payload  OUT VARCHAR2),
  MEMBER PROCEDURE get_text     (payload  OUT CLOB),
  MEMBER PROCEDURE set_replyto  (replyto  IN  sys.aq$_agent),
  MEMBER PROCEDURE set_type     (type     IN  VARCHAR),
  MEMBER PROCEDURE set_userid   (userid   IN  VARCHAR),
  MEMBER PROCEDURE set_appid    (appid    IN  VARCHAR),
  MEMBER PROCEDURE set_groupid  (groupid  IN  VARCHAR),
  MEMBER PROCEDURE set_groupseq (groupseq IN  INT),
  MEMBER PROCEDURE clear_properties,
  MEMBER PROCEDURE set_boolean_property(
     property_name   IN      VARCHAR,
     property_value  IN      BOOLEAN),
  MEMBER PROCEDURE set_byte_property (
     property_name   IN      VARCHAR,
     property_value  IN      INT ),
  MEMBER PROCEDURE set_short_property (
```

```
   property_name   IN      VARCHAR,
   property_value  IN      INT ),
MEMBER PROCEDURE set_int_property (
   property_name   IN      VARCHAR,
   property_value  IN      INT ),
MEMBER PROCEDURE set_long_property (
   property_name   IN      VARCHAR,
   property_value  IN      NUMBER ),
MEMBER PROCEDURE set_float_property (
   property_name   IN      VARCHAR,
   property_value  IN      FLOAT ),
MEMBER PROCEDURE set_double_property (
   property_name   IN      VARCHAR,
   property_value  IN      DOUBLE PRECISION ),
MEMBER PROCEDURE set_string_property (
   property_name   IN      VARCHAR,
   property_value  IN      VARCHAR ),
MEMBER FUNCTION get_replyto  RETURN sys.aq$_agent,
MEMBER FUNCTION get_type     RETURN VARCHAR,
MEMBER FUNCTION get_userid   RETURN VARCHAR,
MEMBER FUNCTION get_appid    RETURN VARCHAR,
MEMBER FUNCTION get_groupid  RETURN VARCHAR,
MEMBER FUNCTION get_groupseq RETURN INT,
MEMBER FUNCTION get_boolean_property (property_name IN  VARCHAR)
  RETURN   BOOLEAN,
MEMBER FUNCTION get_byte_property   (property_name IN  VARCHAR)
  RETURN   INT,
MEMBER FUNCTION get_short_property  (property_name IN  VARCHAR)
  RETURN   INT,
MEMBER FUNCTION get_int_property    (property_name IN  VARCHAR)
  RETURN   INT,
MEMBER FUNCTION get_long_property   (property_name IN  VARCHAR)
  RETURN   NUMBER,
MEMBER FUNCTION get_float_property  (property_name IN  VARCHAR)
  RETURN   FLOAT,
MEMBER FUNCTION get_double_property (property_name IN  VARCHAR)
  RETURN   DOUBLE PRECISION,
MEMBER FUNCTION get_string_property (property_name IN  VARCHAR)
  RETURN   VARCHAR);
```

aq$_jms_bytes_message タイプ

このタイプは、AQ キューの JMSBytes メッセージを格納するために使用するユーザー定義型です。

構文

```
TYPE aq$_jms_bytes_message AS object(
  header     aq$_jms_header,
  bytes_len  INT,
  bytes_raw  raw(2000),
  bytes_lob  blob,
  STATIC FUNCTION construct RETURN aq$_jms_bytes_message,
  MEMBER PROCEDURE set_bytes    (payload  IN RAW),
  MEMBER PROCEDURE set_bytes    (payload  IN BLOB),
  MEMBER PROCEDURE get_bytes    (payload  OUT RAW),
  MEMBER PROCEDURE get_bytes    (payload  OUT BLOB),
  MEMBER PROCEDURE set_replyto  (replyto  IN  sys.aq$_agent),
  MEMBER PROCEDURE set_type     (type     IN  VARCHAR),
  MEMBER PROCEDURE set_userid   (userid   IN  VARCHAR),
  MEMBER PROCEDURE set_appid    (appid    IN  VARCHAR),
  MEMBER PROCEDURE set_groupid  (groupid  IN  VARCHAR),
  MEMBER PROCEDURE set_groupseq (groupseq IN  INT),
  MEMBER PROCEDURE clear_properties,
  MEMBER PROCEDURE set_boolean_property(
     property_name   IN  VARCHAR,
     property_value  IN  BOOLEAN),
  MEMBER PROCEDURE set_byte_property(
     property_name   IN  VARCHAR,
     property_value  IN  INT),
  MEMBER PROCEDURE set_short_property(
     property_name   IN  VARCHAR,
                 property_value  IN INT),
  MEMBER PROCEDURE set_int_property(
                 property_name   IN VARCHAR,
                 property_value  IN INT),
  MEMBER PROCEDURE set_long_property(
                 property_name   IN VARCHAR,
                 property_value  IN NUMBER),
  MEMBER PROCEDURE set_float_property(
                 property_name   IN VARCHAR,
                 property_value  IN FLOAT),
  MEMBER PROCEDURE set_double_property(
                 property_name   IN VARCHAR,
                 property_value  IN DOUBLE PRECISION),
  MEMBER PROCEDURE set_string_property(
                 property_name   IN VARCHAR,
```

```
              property_value  IN VARCHAR),
MEMBER FUNCTION get_replyto RETURN sys.aq$_agent,
MEMBER FUNCTION get_type RETURN VARCHAR,
MEMBER FUNCTION get_userid RETURN VARCHAR,
MEMBER FUNCTION get_appid RETURN VARCHAR,
MEMBER FUNCTION get_groupid RETURN VARCHAR,
MEMBER FUNCTION get_groupseq RETURN INT,
MEMBER FUNCTION get_boolean_property (property_name IN   VARCHAR)
  RETURN   BOOLEAN,
MEMBER FUNCTION get_byte_property    (property_name IN   VARCHAR)
  RETURN   INT,
MEMBER FUNCTION get_short_property   (property_name IN   VARCHAR)
  RETURN   INT,
MEMBER FUNCTION get_int_property     (property_name IN   VARCHAR)
  RETURN   INT,
MEMBER FUNCTION get_long_property    (property_name IN   VARCHAR)
  RETURN   NUMBER,
MEMBER FUNCTION get_float_property   (property_name IN   VARCHAR)
  RETURN   FLOAT,
MEMBER FUNCTION get_double_property  (property_name IN   VARCHAR)
  RETURN   DOUBLE PRECISION,
MEMBER FUNCTION get_string_property  (property_name IN   VARCHAR)
  RETURN   VARCHAR);
```

JMS タイプのメンバーと静的サブプログラムの要約

表 107-1 JMS タイプのメンバーと静的サブプログラム

サブプログラム	説明
LOOKUP_PROPERTY_NAME メンバー・プロシージャ	プロパティに new_property_name が存在するかどうかをチェックします。
SET_REPLYTO メンバー・プロシージャ	JMSReplyTo に対応する replyto パラメータを設定します。
SET_TYPE メンバー・プロシージャ	JMSType に対応する JMS タイプを設定します（任意のテキストを設定できます）。
SET_USERID メンバー・プロシージャ	JMSXUserID に対応する userid を設定します。
SET_APPID メンバー・プロシージャ	JMSXAppID に対応する appid を設定します。
SET_GROUPID メンバー・プロシージャ	JMSXGroupID に対応する groupid を設定します。

表 107-1 JMS タイプのメンバーと静的サブプログラム（続き）

サブプログラム	説明
SET_GROUPSEQ メンバー・プロシージャ	JMSXGroupSeq に対応する groupseq を設定します。
CLEAR_PROPERTIES メンバー・プロシージャ	すべてのプロパティをクリアします。
SET_BOOLEAN_PROPERTY メンバー・プロシージャ	property_name が NULL か存在するかをチェックします。
SET_BYTE_PROPERTY メンバー・プロシージャ	property_name が NULL か存在するかをチェックします。
SET_SHORT_PROPERTY メンバー・プロシージャ	property_name が NULL か存在するかをチェックします。
SET_INT_PROPERTY メンバー・プロシージャ	property_name が NULL か存在するかをチェックします。
SET_LONG_PROPERTY メンバー・プロシージャ	property_name が NULL か存在するかをチェックします。
SET_FLOAT_PROPERTY メンバー・プロシージャ	property_name が NULL か存在するかをチェックします。
SET_DOUBLE_PROPERTY メンバー・プロシージャ	property_name が NULL か存在するかをチェックします。
SET_STRING_PROPERTY メンバー・プロシージャ	property_name が NULL か存在するかをチェックします。
GET_REPLYTO メンバー・ファンクション	JMSReplyTo に対応する replyto を戻します。
GET_TYPE メンバー・ファンクション	JMSType に対応する type を戻します。
GET_USERID メンバー・ファンクション	JMSXUserID に対応する userid を戻します。
GET_APPID メンバー・ファンクション	JMSXAppID に対応する appid を戻します。
GET_GROUPID メンバー・ファンクション	JMSXGroupID に対応する groupid を戻します。
GET_GROUPSEQ メンバー・ファンクション	JMSXGroupSeq に対応する groupseq を戻します。

表 107-1 JMS タイプのメンバーと静的サブプログラム（続き）

サブプログラム	説明
GET_BOOLEAN_PROPERTY メンバー・ファンクション	property_name を検出でき、java_type が BOOLEAN の場合は、BOOLEAN 値を戻します。
GET_BYTE_PROPERTY メンバー・ファンクション	property_name を検出でき、java_type が byte の場合は、byte 値を戻します。
GET_SHORT_PROPERTY メンバー・ファンクション	property_name を検出でき、java_type が short の場合は、short 値を戻します。
GET_INT_PROPERTY メンバー・ファンクション	property_name を検出でき、java_type が INT の場合は、integer 値を戻します。
GET_LONG_PROPERTY メンバー・ファンクション	property_name を検出でき、java_type が long の場合は、number 値を戻します。
GET_FLOAT_PROPERTY メンバー・ファンクション	property_name を検出でき、java_type が FLOAT の場合は、FLOAT 値を戻します。
GET_DOUBLE_PROPERTY メンバー・ファンクション	property_name を検出でき、java_type が DOUBLE の場合は、DOUBLE PRECISION 値を戻します。
GET_STRING_PROPERTY メンバー・ファンクション	property_name を検出でき、java_type が STRING の場合は、VARCHAR 値を戻します。
CONSTRUCT 静的ファンクション	aq$_jms_message のインスタンスを取得します。このインスタンスは、JMS メッセージの特殊なタイプ（JMSText および JMSBytes）を保持できます。
SET_TEXT メンバー・プロシージャ	ペイロードを内部表記に設定します。107-30 ページの「使用上の注意」を参照してください。
GET_TEXT メンバー・プロシージャ	ペイロードの内部表記を VARCHAR2 変数または CLOB 変数のペイロードに設定します。107-31 ページの「使用上の注意」を参照してください。
SET_BYTES メンバー・プロシージャ	ペイロードを内部表記に設定します。107-32 ページの「使用上の注意」を参照してください。
GET_BYTES メンバー・プロシージャ	ペイロードの内部表記を RAW 変数または BLOB 変数のペイロードに設定します。107-33 ページの「使用上の注意」を参照してください。

LOOKUP_PROPERTY_NAME メンバー・プロシージャ

このプロシージャは、new_property_name がプロパティに存在するかどうかをチェックします。

構文

```
LOOKUP_PROPERTY_NAME(
   new_property_name  IN  VARCHAR);
```

パラメータ

表 107-2 LOOKUP_PROPERTY_NAME プロシージャのパラメータ

パラメータ	説明
new_property_name	JMS プロパティ・リストで参照するプロパティ名。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

SET_REPLYTO メンバー・プロシージャ

このプロシージャは、JMSReplyTo に対応する replyto パラメータを設定します。

構文

```
SET_REPLYTO(
   replyto IN SYS.AQ$_AGENT);
```

パラメータ

表 107-3 SET_REPLYTO プロシージャのパラメータ

パラメータ	説明
replyto	クライアントが提供する、JMS メッセージの JMSReplyTo ヘッダー・フィールド。メッセージに対する応答の宛先を指定します。

SET_TYPE メンバー・プロシージャ

このプロシージャは、JMSType に対応する JMS タイプを設定します（任意のテキストを設定できます）。

構文

```
SET_TYPE(
   type IN VARCHAR);
```

パラメータ

表 107-4 SET_TYPE プロシージャのパラメータ

パラメータ	説明
type	JMS メッセージの JMSType ヘッダー・フィールド。クライアントが提供するメッセージ・タイプの識別子です。

SET_USERID メンバー・プロシージャ

このプロシージャは、JMSXUserID に対応する userid を設定します。

構文

```
SET_USERID(
   userid IN VARCHAR);
```

パラメータ

表 107-5 SET_USERID プロシージャのパラメータ

パラメータ	説明
userid	JMS 定義の JMSXUserID メッセージ・プロパティ。送信時に OJMS で設定され、メッセージを送信するユーザーの ID が格納されます。

SET_APPID メンバー・プロシージャ

このプロシージャは、JMSXAppID に対応する appid を設定します。

構文

```
SET_APPID(
   appid IN VARCHAR);
```

パラメータ

表 107-6 SET_APPID プロシージャのパラメータ

パラメータ	説明
appid	JMS 定義の JMSXAppID メッセージ・プロパティ。送信時に OJMS で設定され、メッセージを送信するアプリケーションの ID が格納されます。

SET_GROUPID メンバー・プロシージャ

このプロシージャは、JMSXGroupID に対応する groupid を設定します。

構文

```
SET_GROUPID(
   groupid IN VARCHAR);
```

パラメータ

表 107-7 SET_GROUPID プロシージャのパラメータ

パラメータ	説明
groupid	JMS 定義の JMSXGroupID メッセージ・プロパティ。クライアントによって設定され、現行メッセージが含まれるメッセージ・グループの ID が格納されます。

SET_GROUPSEQ メンバー・プロシージャ

このプロシージャは、JMSXGroupSeq に対応する groupseq を設定します。

構文

```
SET_GROUPSEQ(
   groupseq IN INT);
```

パラメータ

表 107-8 SET_GROUPSEQ プロシージャのパラメータ

パラメータ	説明
groupseq	JMS 定義の JMSXGroupSeq メッセージ・プロパティ。クライアントによって設定され、グループ内の 1 から始まるメッセージ順序番号が格納されます。

CLEAR_PROPERTIES メンバー・プロシージャ

このプロシージャは、すべてのプロパティをクリアします。

構文

```
CLEAR_PROPERTIES;
```

SET_BOOLEAN_PROPERTY メンバー・プロシージャ

このプロシージャは、property_name が NULL か存在するかをチェックします。存在しない場合は、property_value を内部表記（NUMBER タイプ）に格納します。

構文

```
SET_BOOLEAN_PROPERTY(
   property_name  IN VARCHAR,
   property_value IN BOOLEAN);
```

パラメータ

表 107-9 SET_BOOLEAN_PROPERTY プロシージャのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。
property_value	JMS メッセージのユーザー・プロパティまたはシステム・プロパティの値。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

SET_BYTE_PROPERTY メンバー・プロシージャ

このプロシージャは、property_name が NULL か存在するかをチェックします。存在しない場合は、property_value が -127 ～ 127（8 ビット）の範囲内であるかどうかをチェックします。PL/SQL または RDBMS では byte データ・タイプが定義されないため、このチェックが必要です。

構文

```
SET_BYTE_PROPERTY(
   property_name  IN VARCHAR,
   property_value IN INT);
```

パラメータ

表 107-10 SET_BYTE_PROPERTY プロシージャのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。
property_value	JMS メッセージのユーザー・プロパティまたはシステム・プロパティの値。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

プロパティ値が有効範囲を超えている場合は、ORA-24193。

SET_SHORT_PROPERTY メンバー・プロシージャ

このプロシージャは、property_name が NULL か存在するかをチェックします。存在しない場合は、property_value が -32767 ～ 32767（16 ビット）の範囲内であるかどうかをチェックします。PL/SQL または RDBMS では short データ・タイプが定義されないため、このチェックが必要です。

構文

```
SET_SHORT_PROPERTY(
   property_name  IN VARCHAR,
   property_value IN INT);
```

パラメータ

表 107-11 SET_SHORT_PROPERTY プロシージャのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。
property_value	JMS メッセージのユーザー・プロパティまたはシステム・プロパティの値。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

プロパティ値が有効範囲を超えている場合は、ORA-24193。

SET_INT_PROPERTY メンバー・プロシージャ

このプロシージャは、property_name が NULL か存在するかをチェックします。存在しない場合は、property_value が -2147483647 ～ 2147483647（32 ビット）の範囲内であるかどうかをチェックします。PL/SQL および Oracle データベースでの INT データ・タイプは 38 ビットであるため、このチェックが必要です。

構文

```
SET_INT_PROPERTY(
   property_name  IN VARCHAR,
   property_value IN INT);
```

パラメータ

表 107-12 SET_INT_PROPERTY プロシージャのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。
property_value	JMS メッセージのユーザー・プロパティまたはシステム・プロパティの値。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

プロパティ値が有効範囲を超えている場合は、ORA-24193。

SET_LONG_PROPERTY メンバー・プロシージャ

このプロシージャは、property_name が NULL か存在するかをチェックします。存在しない場合は、property_value を格納します。PL/SQL および Oracle データベースでの NUMBER データ・タイプは 32 ビットです。Java での long データ・タイプは 64 ビットです。したがって、値の範囲チェックは必要ありません。

構文

```
SET_LONG_PROPERTY(
   property_name  IN VARCHAR,
   property_value IN NUMBER);
```

パラメータ

表 107-13 SET_LONG_PROPERTY プロシージャのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。
property_value	JMS メッセージのユーザー・プロパティまたはシステム・プロパティの値。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

SET_FLOAT_PROPERTY メンバー・プロシージャ

このプロシージャは、property_name が NULL か存在するかをチェックします。存在しない場合は、property_value を格納します。

構文

```
SET_FLOAT_PROPERTY(
   property_name  IN VARCHAR,
   property_value IN FLOAT);
```

パラメータ

表 107-14 SET_FLOAT_PROPERTY プロシージャのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。
property_value	JMS メッセージのユーザー・プロパティまたはシステム・プロパティの値。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

SET_DOUBLE_PROPERTY メンバー・プロシージャ

このプロシージャは、property_name が NULL か存在するかをチェックします。存在しない場合は、property_value を格納します。

構文

```
SET_DOUBLE_PROPERTY(
   property_name  IN VARCHAR,
   property_value IN DOUBLE PRECISION);
```

パラメータ

表 107-15 SET_DOUBLE_PROPERTY プロシージャのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。
property_value	JMS メッセージのユーザー・プロパティまたはシステム・プロパティの値。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

SET_STRING_PROPERTY メンバー・プロシージャ

このプロシージャは、property_name が NULL か存在するかをチェックします。存在しない場合は、property_value を格納します。

構文

```
SET_STRING_PROPERTY(
   property_name  IN VARCHAR,
   property_value IN VARDHAR);
```

パラメータ

表 107-16 SET_STRING_PROPERTY プロシージャのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。
property_value	JMS メッセージのユーザー・プロパティまたはシステム・プロパティの値。

例外

プロパティ名が存在する場合は、ORA-24191。

プロパティ名が NULL の場合は、ORA-24192。

GET_REPLYTO メンバー・ファンクション

このファンクションは、JMSReplyTo に対応する replyto を戻します。

構文

```
GET_REPLYTO(
   replyto OUT SYS.AQ$_AGENT);
```

戻り値

表 107-17 GET_REPLYTO ファンクションの戻り値

戻り値	説明
replyto	クライアントが提供する、JMS メッセージの JMSReplyTo ヘッダー・フィールド。メッセージに対する応答の宛先を指定します。

GET_TYPE メンバー・ファンクション

このファンクションは、JMSType に対応する type を戻します。

構文

```
GET_TYPE(
   type OUT VARCHAR);
```

戻り値

表 107-18 GET_TYPE ファンクションの戻り値

戻り値	説明
type	JMS メッセージの JMSType ヘッダー・フィールド。クライアントが提供するメッセージ・タイプの識別子です。

GET_USERID メンバー・ファンクション

このファンクションは、JMSXUserID に対応する userid を戻します。

構文

```
GET_USERID(
   userid OUT VARCHAR);
```

戻り値

表 107-19 GET_USERID ファンクションの戻り値

戻り値	説明
userid	JMS 定義の JMSXUserID メッセージ・プロパティ。送信時に OJMS で設定され、メッセージを送信するユーザーの ID が格納されます。

GET_APPID メンバー・ファンクション

このファンクションは、JMSXAppID に対応する appid を戻します。

構文

```
GET_APPID(
   appid OUT VARCHAR);
```

戻り値

表 107-20 GET_APPID ファンクションの戻り値

戻り値	説明
appid	JMS 定義の JMSXAppID メッセージ・プロパティ。送信時に OJMS で設定され、メッセージを送信するアプリケーションの ID が格納されます。

GET_GROUPID メンバー・ファンクション

このファンクションは、JMSXGroupID に対応する groupid を戻します。

構文

```
GET_GROUPID(
   groupid OUT VARCHAR);
```

戻り値

表 107-21 GET_GROUPID ファンクションの戻り値

戻り値	説明
groupid	JMS 定義の JMSXGroupID メッセージ・プロパティ。クライアントによって設定され、現行メッセージが含まれるメッセージ・グループの ID が格納されます。

GET_GROUPSEQ メンバー・ファンクション

このファンクションは、JMSXGroupSeq に対応する groupseq を戻します。

構文

```
GET_GROUPSEQ(
   groupseq OUT INT);
```

戻り値

表 107-22 GET_GROUPSEQ ファンクションの戻り値

戻り値	説明
groupseq	JMS 定義の JMSXGroupSeq メッセージ・プロパティ。クライアントによって設定され、グループ内の 1 から始まるメッセージ順序番号が格納されます。

GET_BOOLEAN_PROPERTY メンバー・ファンクション

このファンクションは、property_name を検出でき、java_type が BOOLEAN の場合は、BOOLEAN 値を戻します。それ以外の場合は、NULL を戻します。

構文

```
GET_BOOLEAN_PROPERTY(
   property_name IN  VARCHAR)
  RETURN BOOLEAN;
```

パラメータ

表 107-23 GET_BOOLEAN_PROPERTY ファンクションのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。

GET_BYTE_PROPERTY メンバー・ファンクション

このファンクションは、property_name を検出でき、java_type が byte の場合は、byte 値を戻します。それ以外の場合は、NULL を戻します。

構文

```
GET_BYTE_PROPERTY(
   property_name IN  VARCHAR)
  RETURN  INT;
```

パラメータ

表 107-24 GET_BYTE_PROPERTY ファンクションのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。

GET_SHORT_PROPERTY メンバー・ファンクション

このファンクションは、property_name を検出でき、java_type が short の場合は、short 値を戻します。それ以外の場合は、NULL を戻します。

構文

```
GET_SHORT_PROPERTY(
   property_name IN  VARCHAR)
  RETURN INT;
```

パラメータ

表 107-25 GET_SHORT_PROPERTY ファンクションのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。

GET_INT_PROPERTY メンバー・ファンクション

このファンクションは、property_name を検出でき、java_type が INT の場合は、integer 値を戻します。それ以外の場合は、NULL を戻します。

構文

```
GET_INT_PROPERTY(
   property_name IN  VARCHAR)
  RETURN INT;
```

パラメータ

表 107-26 GET_INT_PROPERTY ファンクションのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。

GET_LONG_PROPERTY メンバー・ファンクション

このファンクションは、property_name を検出でき、java_type が long の場合は、number 値を戻します。それ以外の場合は、NULL を戻します。

構文

```
GET_LONG_PROPERTY(
   property_name IN  VARCHAR)
  RETURN NUMBER;
```

パラメータ

表 107-27 GET_LONG_PROPERTY ファンクションのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。

GET_FLOAT_PROPERTY メンバー・ファンクション

このファンクションは、property_name を検出でき、java_type が FLOAT の場合は、FLOAT 値を戻します。それ以外の場合は、NULL を戻します。

構文

```
GET_FLOAT_PROPERTY(
   property_name IN  VARCHAR)
  RETURN FLOAT;
```

パラメータ

表 107-28 GET_FLOAT_PROPERTY ファンクションのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。

GET_DOUBLE_PROPERTY メンバー・ファンクション

このファンクションは、property_name を検出でき、java_type が DOUBLE の場合は、DOUBLE PRECISION 値を戻します。それ以外の場合は、NULL を戻します。

構文

```
GET_DOUBLE_PROPERTY(
   property_name IN  VARCHAR)
  RETURN DOUBLE PRECISION;
```

パラメータ

表 107-29 GET_DOUBLE_PROPERTY ファンクションのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。

GET_STRING_PROPERTY メンバー・ファンクション

このファンクションは、property_name を検出でき、java_type が STRING の場合は、VARCHAR 値を戻します。それ以外の場合は、NULL を戻します。

構文

```
GET_STRING_PROPERTY(
   property_name IN  VARCHAR)
  RETURN VARCHAR;
```

パラメータ

表 107-30 GET_STRING_PROPERTY ファンクションのパラメータ

パラメータ	説明
property_name	ユーザー指定の JMS ユーザー・プロシージャまたは JMS 指定の JMS システム・プロパティの名前。

CONSTRUCT 静的ファンクション

このファンクションは、aq$_jms_message のインスタンスを取得するために使用します。このインスタンスは、JMS メッセージの特定のタイプ（JMSText および JMSBytes）を保持できます。各インスタンスが保持できるメッセージ・タイプは、構成メソッドに渡される mtype パラメータによって異なります。構成後のメッセージは、保持用に構成された JMS メッセージ・タイプを格納するためにのみ使用できます。mtype パラメータの有効値は、107-2 ページの「aq$_jms_message タイプをサポートするための定数」に定義されています。詳細は、107-4 ページの「aq$_jms_message タイプ」を参照してください。

構文

```
CONSTRUCT(
   mtype  IN INT)
  RETURN aq$_jms_message;
```

構文

```
CONSTRUCT RETURN aq$_jms_text_message;
```

SET_TEXT メンバー・プロシージャ

このプロシージャは、ペイロード（VARCHAR2）を内部表記に設定します。ペイロードの長さが 4000 以下の場合は、text_vc に設定されます。4000 を超える場合は、text_lob に設定されます。

構文

```
SET_TEXT(
   payload IN VARCHAR2);
```

構文

このプロシージャは、ペイロード（CLOB）を内部表記に設定します。ペイロードは、text_lob に設定されます。

```
SET_TEXT(
   payload IN CLOB);
```

パラメータ

表 107-31 SET_TEXT プロシージャのパラメータ

パラメータ	説明
payload	JMS メッセージのペイロード。

使用上の注意

このプロシージャは、aq$_jms_text_message タイプ（および aq$_jms_message タイプ）には使用できますが、aq$_jms_bytes_message タイプには使用できません。

GET_TEXT メンバー・プロシージャ

このプロシージャは、ペイロードの内部表記を VARCHAR2 変数のペイロードに設定します。text_vc が NULL でない場合は、text_vc を payload に設定し、text_lob が 32767（2**16 -1）以下の場合は、text_lob を payload に転送します。

構文

```
GET_TEXT(
   payload OUT VARCHAR2);
```

構文

このプロシージャは、内部ペイロードを CLOB 変数のペイロードに設定します。text_lob が NULL でない場合は、text_lob を payload に設定するか、または text_vc を payload に転送します。

```
GET_TEXT(
   payload OUT CLOB;
```

パラメータ

表 107-32 GET_TEXT プロシージャのパラメータ

パラメータ	説明
payload	JMS メッセージのペイロード。

例外

ペイロードの長さが 32767（PL/SQL での VARCHAR2 の最大長）を超える場合は、ORA-24190。

使用上の注意

このプロシージャは、aq$_jms_text_message タイプ（および aq$_jms_message タイプ）には使用できますが、aq$_jms_bytes_message タイプには使用できません。

SET_BYTES メンバー・プロシージャ

このプロシージャは、ペイロード（RAW 値）を内部表記（payload の長さが 2000 以下の場合は bytes_raw、2000 を超える場合は bytes_lob）に設定します。

構文

```
SET_BYTES(
   payload IN RAW);
```

構文

このプロシージャは、ペイロード（BLOB）を内部表記（bytes_lob）に設定します。

```
SET_BYTES(
   payload IN BLOB);
```

パラメータ

表 107-33 SET_BYTES プロシージャのパラメータ

パラメータ	説明
payload	JMS メッセージのペイロード。

使用上の注意

このプロシージャは、aq$_jms_bytes_message タイプ（および aq$_jms_message タイプ）には使用できますが、aq$_jms_text_message タイプには使用できません。

GET_BYTES メンバー・プロシージャ

このプロシージャは、ペイロードの内部表記を RAW 変数のペイロードに設定します。bytes_raw が NULL でない場合は、それを payload に設定し、bytes_lob が 32767（2**16 -1）以下の場合は、bytes_lob を payload に転送します。

構文

```
GET_BYTES(
   payload OUT RAW);
```

構文

このプロシージャは、ペイロードの内部表記を BLOB 変数のペイロードに設定します。

```
GET_BYTES(
   payload OUT BLOB);
```

例外

ペイロードの長さが 32767（PL/SQL での VARCHAR2 の最大長）を超える場合は、ORA-24190。

戻り値

表 107-34 GET_BYTES ファンクションの戻り値

戻り値	説明
payload	JMS メッセージのペイロード。

使用上の注意

このプロシージャは、aq$_jms_bytes_message タイプ（および aq$_jms_message タイプ）には使用できますが、aq$_jms_text_message タイプには使用できません。

Oracle JMS 管理インタフェースを使用したエンキュー : 例

次のサンプル・プログラムは、大量のテキスト・メッセージ（JMS ユーザー・プロパティとともに使用）を、JMS の TEXT メッセージを保持するために OJMS 管理インタフェースを使用して作成した AQ キューにエンキューします。この例でエンキューされたテキストとバイト・メッセージは、両方とも OJMS Java クライアントを使用してデキューできます。

```
DECLARE

    text         varchar2(32767);
    agent        sys.aq$_agent   := sys.aq$_agent(' ', null, 0);
    message      sys.aq$_jms_text_message;

    enqueue_options    dbms_aq.enqueue_options_t;
    message_properties dbms_aq.message_properties_t;
    msgid               raw(16);

BEGIN

    message := sys.aq$_jms_text_message.construct;

    message.set_replyto(agent);
    message.set_type('tkaqpet2');
    message.set_userid('jmsuser');
    message.set_appid('plsql_enq');
    message.set_groupid('st');
    message.set_groupseq(1);

    message.set_boolean_property('import', True);
    message.set_string_property('color', 'RED');
    message.set_short_property('year', 1999);
    message.set_long_property('mileage', 300000);
    message.set_double_property('price', 16999.99);
    message.set_byte_property('password', 127);

    FOR i IN 1..500 LOOP
        text := CONCAT (text, '1234567890');
    END LOOP;

    message.set_text(text);

    dbms_aq.enqueue(queue_name => 'jmsuser.jms_text_t1',
                    enqueue_options => enqueue_options,
                    message_properties => message_properties,
                    payload => message,
                    msgid => msgid);
```

```
END;
```

次のサンプル・プログラムは、バイト数の多いメッセージをエンキューします。

```
DECLARE

    text        VARCHAR2(32767);
    bytes       RAW(32767);
    agent       sys.aq$_agent   := sys.aq$_agent(' ', null, 0);
    message     sys.aq$_jms_bytes_message;
    body        BLOB;
    position    INT;

    enqueue_options    dbms_aq.enqueue_options_t;
    message_properties dbms_aq.message_properties_t;
    msgid raw(16);

BEGIN

    message := sys.aq$_jms_bytes_message.construct;

    message.set_replyto(agent);
    message.set_type('tkaqper4');
    message.set_userid('jmsuser');
    message.set_appid('plsql_enq_raw');
    message.set_groupid('st');
    message.set_groupseq(1);

    message.set_boolean_property('import', True);
    message.set_string_property('color', 'RED');
    message.set_short_property('year', 1999);
    message.set_long_property('mileage', 300000);
    message.set_double_property('price', 16999.99);

--  prepare a huge payload into a blob

    FOR i IN 1..1000 LOOP
        text := CONCAT (text, '0123456789ABCDEF');
    END LOOP;

    bytes := HEXTORAW(text);

    dbms_lob.createtemporary(lob_loc => body, cache => TRUE);
    dbms_lob.open (body, DBMS_LOB.LOB_READWRITE);
    position := 1 ;
    FOR i IN 1..10 LOOP
        dbms_lob.write ( lob_loc => body,
```

```
                amount => FLOOR((LENGTH(bytes)+1)/2),
                offset => position,
                buffer => bytes);
            position := position + FLOOR((LENGTH(bytes)+1)/2) ;
        END LOOP;

    -- end of the preparation

        message.set_bytes(body);
        dbms_aq.enqueue(queue_name => 'jmsuser.jms_bytes_t1',
                        enqueue_options => enqueue_options,
                        message_properties => message_properties,
                        payload => message,
                        msgid => msgid);

        dbms_lob.freetemporary(lob_loc => body);
    END;
```

108

論理変更レコードのタイプ

この章では、論理変更レコード（LCR）のタイプについて説明します。Streams における LCR は、データベースに対する変更に関する情報が格納されたメッセージ・ペイロードです。この変更には、データに対する変更（データ操作言語（DML）の変更）とデータベース・オブジェクトに対する変更（データ定義言語（DDL）の変更）を含めることができます。

Streams を使用する場合、取得プロセスでは、LCR の形式で取得した変更をキューにエンキューします。この LCR は、あるデータベースのキューから別のデータベースのキューに伝播できます。最後に、適用プロセスは LCR を宛先データベースに適用できます。LCR は、手動で作成、エンキューおよびデキューすることもできます。

LCR タイプは、オラクル社が提供する次の PL/SQL パッケージで使用します。

- DBMS_APPLY_ADM
- DBMS_AQ
- DBMS_AQADM
- DBMS_CAPTURE_ADM
- DBMS_PROPAGATION_ADM
- DBMS_RULE
- DBMS_RULE_ADM
- DBMS_STREAMS
- DBMS_STREAMS_ADM
- DBMS_TRANSFORM

この章では、次の項目について説明します。

- LCR$_DDL_RECORD タイプ
- LCR$_ROW_RECORD タイプ
- LCR$_ROW_RECORD と LCR$_DDL_RECORD に共通するサブプログラム
- LCR$_ROW_LIST タイプ
- LCR$_ROW_UNIT タイプ

LCR$_DDL_RECORD タイプ

このタイプは、データベース・オブジェクトに対する DDL 変更を表します。

DDL LCR を作成または変更する場合は、ddl_text と base_table_name、base_table_owner、object_type、object_owner、object_name および command_type の各属性との間で一貫性が保たれていることを確認してください。

注意：

- データベース・オブジェクトの名前をパラメータとして LCR コンストラクタに渡すときは、その名前を二重引用符で囲むと、大 / 小文字または小文字をデータベース・オブジェクト用に使用している名前を処理できます。たとえば、名前に小文字が含まれている場合は、その名前を二重引用符で囲む必要があります。
- アプリケーションによる LCR の作成時には、トランザクション識別子または SCN の指定は不要です。これは、それらの値が適用プロセスによって生成され、メモリーに格納されるためです。トランザクション識別子または SCN が LCR に指定されている場合、適用プロセスはその指定を無視し、新しい値を割り当てます。

LCR$_DDL_RECORD コンストラクタ

指定した情報を使用して、SYS.LCR$_DDL_RECORD オブジェクトを作成します。

```
STATIC FUNCTION CONSTRUCT(
   source_database_name       IN VARCHAR2,
   command_type               IN VARCHAR2,
   object_owner               IN VARCHAR2,
   object_name                IN VARCHAR2,
   object_type                IN VARCHAR2,
   ddl_text                   IN CLOB,
   logon_user                 IN VARCHAR2,
   current_schema             IN VARCHAR2,
   base_table_owner           IN VARCHAR2,
   base_table_name            IN VARCHAR2,
   tag                        IN RAW       DEFAULT NULL,
   transaction_id             IN VARCHAR2  DEFAULT NULL,
   scn                        IN NUMBER    DEFAULT NULL)
RETURN SYS.LCR$_DDL_RECORD;
```

LCR$_DDL_RECORD コンストラクタ・ファンクションのパラメータ

表 108-1 LCR$_DDL_RECORD コンストラクタ・ファンクションのパラメータ

パラメータ	説明
source_database_name	DDL 文が発生したデータベース。ドメイン名を含めていない場合は、ローカル・ドメインがデータベース名に自動的に追加されます。たとえば、DBS1 を指定し、ローカル・ドメインが .NET の場合は、DBS1.NET が自動的に指定されます。このパラメータは、NULL 以外の値に設定する必要があります。
command_type	DDL 文で実行されるコマンドのタイプ。このパラメータは、NULL 以外の値に設定する必要があります。 **関連項目：**コマンド・タイプの完全なリストは、『Oracle Call Interface プログラマーズ・ガイド』の「SQL コマンド・コード」の表を参照してください。 次のコマンド・タイプは、DDL LCR ではサポートされていません。 ALTER MATERIALIZED VIEW ALTER MATERIALIZED VIEW LOG ALTER SUMMARY CREATE SCHEMA CREATE MATERIALIZED VIEW CREATE MATERIALIZED VIEW LOG CREATE SUMMARY DROP MATERIALIZED VIEW DROP MATERIALIZED VIEW LOG DROP SUMMARY RENAME マテリアライズド・ビューのコマンド・タイプのスナップショットもサポートされていません。
object_owner	DDL 文が実行されたオブジェクトを所有するユーザー。
object_name	DDL 文が実行されたデータベース・オブジェクト。

表 108-1 LCR$_DDL_RECORD コンストラクタ・ファンクションのパラメータ（続き）

パラメータ	説明
object_type	DDL 文が実行されたオブジェクトのタイプ。 有効なオブジェクト・タイプは次のとおりです。 CLUSTER FUNCTION INDEX LINK OUTLINE PACKAGE PACKAGE BODY PROCEDURE SEQUENCE SYNONYM TABLE TRIGGER TYPE USER VIEW LINK はデータベース・リンクを表します。 NULL も有効なオブジェクト・タイプです。すべてのオブジェクト・タイプをリストしない場合は、NULL を指定します。GET_OBJECT_TYPE メンバー・プロシージャは、オブジェクト・タイプがリストされていない場合、NULL を戻します。
ddl_text	DDL 文のテキスト。このパラメータは、NULL 以外の値に設定する必要があります。
logon_user	DDL 文を実行したセッションのユーザー。
current_schema	変更済みデータベース・オブジェクトに対するスキーマが、ddl_text に明示的に指定されていない場合に使用されるスキーマ。current_schema と異なるスキーマが ddl_text に指定されている場合は、ddl_text に指定されているスキーマが使用されます。 このパラメータは、NULL 以外の値に設定する必要があります。
base_table_owner	DDL 文が表に関連する DDL の場合（CREATE TABLE や ALTER TABLE など）、または DDL 文に表が含まれる場合（表にトリガーを作成する場合など）、base_table_owner では、含まれる表の所有者を指定します。それ以外の場合、base_table_owner は NULL になります。

表 108-1 LCR$_DDL_RECORD コンストラクタ・ファンクションのパラメータ（続き）

パラメータ	説明
base_table_name	DDL 文が表に関連する DDL の場合（CREATE TABLE や ALTER TABLE など）、または DDL 文に表が含まれる場合（表にトリガーを作成する場合など）、base_table_name では、含まれる表の名前を指定します。それ以外の場合、base_table_name は NULL になります。
tag	LCR の追跡を有効にするバイナリ・タグ。たとえば、適用による転送を使用する場合は、DDL 文の元のソース・データベースを判断するためにこのタグを使用します。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。
transaction_id	トランザクションの識別子。
scn	取得した LCR の変更レコードが REDO に書き込まれたときの SCN。SCN の値は、ユーザー作成の LCR に対しては無効です。

LCR$_DDL_RECORD サブプログラムの要約

表 108-2 LCR$_DDL_RECORD のサブプログラム

サブプログラム	説明
共通のサブプログラム	SYS.LCR$_ROW_RECORD タイプと SYS.LCR$_DDL_RECORD タイプに共通するサブプログラムのリストは、108-26 ページの「LCR$_ROW_RECORD と LCR$_DDL_RECORD に共通するサブプログラム」を参照してください。
「EXECUTE メンバー・プロシージャ」108-7 ページ	現行ユーザーのセキュリティ・ドメイン下で LCR を実行します。
「GET_BASE_TABLE_NAME メンバー・ファンクション」108-8 ページ	ベース（依存）表名を戻します。
「GET_BASE_TABLE_OWNER メンバー・ファンクション」108-8 ページ	ベース（依存）表の所有者を戻します。
「GET_CURRENT_SCHEMA メンバー・ファンクション」108-8 ページ	デフォルトのスキーマ（ユーザー）名を戻します。
「GET_DDL_TEXT メンバー・プロシージャ」108-8 ページ	CLOB の DDL テキストを取得します。

表 108-2 LCR$_DDL_RECORD のサブプログラム（続き）

サブプログラム	説明
「GET_LOGON_USER メンバー・ファンクション」108-9 ページ	ログオン・ユーザー名を戻します。
「GET_OBJECT_TYPE メンバー・ファンクション」108-9 ページ	DDL に含まれるオブジェクトのタイプを戻します。
「SET_BASE_TABLE_NAME メンバー・プロシージャ」108-10 ページ	ベース（依存）表名を設定します。
「SET_BASE_TABLE_OWNER メンバー・プロシージャ」108-10 ページ	ベース（依存）表の所有者を設定します。
「SET_CURRENT_SCHEMA メンバー・プロシージャ」108-10 ページ	デフォルトのスキーマ（ユーザー）名を設定します。
「SET_DDL_TEXT メンバー・プロシージャ」108-11 ページ	DDL テキストを設定します。
「SET_LOGON_USER メンバー・プロシージャ」108-11 ページ	ログオン・ユーザー名を設定します。
「SET_OBJECT_TYPE メンバー・プロシージャ」108-12 ページ	オブジェクト・タイプを設定します。

EXECUTE メンバー・プロシージャ

現行ユーザーのセキュリティ・ドメイン下で DDL LCR を実行します。このプロシージャを使用して LCR が適用された場合、LCR に対して実行される適用プロセスのハンドラは実行されません。

> **注意：** EXECUTE メンバー・プロシージャを起動できるのは、適用プロセスの適用ハンドラ内のみです。

構文

```
MEMBER PROCEDURE EXECUTE();
```

GET_BASE_TABLE_NAME メンバー・ファンクション

ベース（依存）表名を戻します。

構文

```
MEMBER FUNCTION GET_BASE_TABLE_NAME RETURN VARCHAR2;
```

GET_BASE_TABLE_OWNER メンバー・ファンクション

ベース（依存）表の所有者を戻します。

構文

```
MEMBER FUNCTION GET_BASE_TABLE_OWNER RETURN VARCHAR2;
```

GET_CURRENT_SCHEMA メンバー・ファンクション

現行のスキーマ名を戻します。

構文

```
MEMBER FUNCTION GET_CURRENT_SCHEMA RETURN VARCHAR2;
```

GET_DDL_TEXT メンバー・プロシージャ

CLOB の DDL テキストを取得します。

次に、このプロシージャを使用して DDL LCR の DDL テキストを取得する PL/SQL プロシージャの例を示します。

```
CREATE OR REPLACE PROCEDURE ddl_in_lcr (ddl_lcr in SYS.LCR$_DDL_RECORD)
IS
  ddl_text   CLOB;
BEGIN
  DBMS_OUTPUT.PUT_LINE( '  ----------------------------------------' );
  DBMS_OUTPUT.PUT_LINE( '  Displaying DDL text in a DDL LCR: ' );
  DBMS_OUTPUT.PUT_LINE( '  ----------------------------------------' );
  DBMS_LOB.CREATETEMPORARY(ddl_text, TRUE);
  ddl_lcr.GET_DDL_TEXT(ddl_text);
  DBMS_OUTPUT.PUT_LINE('DDL text:' || ddl_text);
  DBMS_LOB.FREETEMPORARY(ddl_text);
END;
/
```

注意： GET_DDL_TEXT は、CLOB で使用する領域の管理を容易にするためのメンバー・プロシージャであり、メンバー・ファンクションではありません。前述の例では、CLOB の一時領域が作成され、不要になると解放されることに注意してください。

構文

```
MEMBER FUNCTION GET_DDL_TEXT(
  ddl_text  IN OUT  CLOB);
```

パラメータ

表 108-3 GET_DDL_TEXT プロシージャのパラメータ

パラメータ	説明
ddl_text	DDL LCR の DDL テキスト。

GET_LOGON_USER メンバー・ファンクション

ログオン・ユーザー名を戻します。

構文

```
MEMBER FUNCTION GET_LOGON_USER RETURN VARCHAR2;
```

GET_OBJECT_TYPE メンバー・ファンクション

DDL に含まれるオブジェクトのタイプを戻します。

構文

```
MEMBER FUNCTION GET_OBJECT_TYPE RETURN VARCHAR2;
```

SET_BASE_TABLE_NAME メンバー・プロシージャ

ベース（依存）表名を設定します。

構文

```
MEMBER PROCEDURE SET_BASE_TABLE_NAME(
     base_table_name IN VARCHAR2);
```

パラメータ

表 108-4 SET_BASE_TABLE_NAME プロシージャのパラメータ

パラメータ	説明
base_table_name	ベース表の名前。

SET_BASE_TABLE_OWNER メンバー・プロシージャ

ベース（依存）表の所有者を設定します。

構文

```
MEMBER PROCEDURE SET_BASE_TABLE_OWNER(
     base_table_owner IN VARCHAR2);
```

パラメータ

表 108-5 SET_BASE_TABLE_OWNER プロシージャのパラメータ

パラメータ	説明
base_table_owner	ベース表の所有者の名前。

SET_CURRENT_SCHEMA メンバー・プロシージャ

デフォルトのスキーマ（ユーザー）名を設定します。

構文

```
MEMBER PROCEDURE SET_CURRENT_SCHEMA(
     current_schema IN VARCHAR2);
```

パラメータ

表 108-6 SET_CURRENT_SCHEMA プロシージャのパラメータ

パラメータ	説明
current_schema	現行スキーマとして設定するスキーマの名前。このパラメータは、NULL 以外の値に設定する必要があります。

SET_DDL_TEXT メンバー・プロシージャ

DDL テキストを設定します。

構文

```
MEMBER PROCEDURE SET_DDL_TEXT(
    ddl_text     IN CLOB);
```

パラメータ

表 108-7 SET_DDL_TEXT プロシージャのパラメータ

パラメータ	説明
ddl_text	DDL テキスト。このパラメータは、NULL 以外の値に設定する必要があります。

SET_LOGON_USER メンバー・プロシージャ

ログオン・ユーザー名を設定します。

構文

```
MEMBER PROCEDURE SET_LOGON_USER(
     logon_user IN VARCHAR2);
```

パラメータ

表 108-8 SET_LOGON_USER プロシージャのパラメータ

パラメータ	説明
logon_user	ログオン・ユーザーとして設定するスキーマの名前。

SET_OBJECT_TYPE メンバー・プロシージャ

オブジェクト・タイプを設定します。

構文

```
MEMBER PROCEDURE SET_OBJECT_TYPE(
     object_type    IN VARCHAR2);
```

パラメータ

表 108-9 SET_OBJECT_TYPE プロシージャのパラメータ

パラメータ	説明
object_type	オブジェクト・タイプ。 有効なオブジェクト・タイプは次のとおりです。 CLUSTER FUNCTION INDEX LINK OUTLINE PACKAGE PACKAGE BODY PROCEDURE SEQUENCE SYNONYM TABLE TRIGGER TYPE USER VIEW LINK はデータベース・リンクを表します。 NULL も有効なオブジェクト・タイプです。すべてのオブジェクト・タイプをリストしない場合は、NULL を指定します。GET_OBJECT_TYPE メンバー・プロシージャは、オブジェクト・タイプがリストされていない場合、NULL を戻します。

LCR$_ROW_RECORD タイプ

このタイプは、表内の行に対する DML 変更を表します。このタイプでは、LCR$_ROW_LIST タイプを使用します。

行の LCR を作成または変更する場合は、command_type 属性に、古い列値の有無と新しい列値の有無について一貫性があることを確認してください。

注意：

- データベース・オブジェクトの名前をパラメータとして LCR コンストラクタに渡すときは、その名前を二重引用符で囲むと、大 / 小文字または小文字をデータベース・オブジェクト用に使用している名前を処理できます。たとえば、名前に小文字が含まれている場合は、その名前を二重引用符で囲む必要があります。
- アプリケーションによる LCR の作成時には、トランザクション識別子または SCN の指定は不要です。これは、それらの値が適用プロセスによって生成され、メモリーに格納されるためです。トランザクション識別子または SCN が LCR に指定されている場合、適用プロセスはその指定を無視し、新しい値を割り当てます。

関連項目： 108-32 ページ「LCR$_ROW_LIST タイプ」

LCR$_ROW_RECORD コンストラクタ

指定した情報を使用して、SYS.LCR$_ROW_RECORD オブジェクトを作成します。

```
STATIC FUNCTION CONSTRUCT(
   source_database_name       IN VARCHAR2,
   command_type               IN VARCHAR2,
   object_owner               IN VARCHAR2,
   object_name                IN VARCHAR2,
   tag                        IN RAW                DEFAULT NULL,
   transaction_id             IN VARCHAR2           DEFAULT NULL,
   scn                        IN NUMBER             DEFAULT NULL,
   old_values                 IN SYS.LCR$_ROW_LIST  DEFAULT NULL,
   new_values                 IN SYS.LCR$_ROW_LIST  DEFAULT NULL)
RETURN SYS.LCR$_ROW_RECORD;
```

LCR$_ROW_RECORD コンストラクタ・ファンクションのパラメータ

表 108-10 LCR$_ROW_RECORD コンストラクタ・ファンクションのパラメータ

パラメータ	説明
source_database_name	行の変更が発生したデータベース。ドメイン名を含めていない場合は、ローカル・ドメインがデータベース名に自動的に追加されます。たとえば、DBS1 を指定し、ローカル・ドメインが .NET の場合は、DBS1.NET が自動的に指定されます。このパラメータは、NULL 以外の値に設定する必要があります。
command_type	DML 文で実行されるコマンドのタイプ。このパラメータは、NULL 以外の値に設定する必要があります。 有効な値は次のとおりです。 INSERT UPDATE DELETE LOB ERASE LOB WRITE LOB TRIM INSERT の場合、LCR には、空ではない new_values コレクション、および空または NULL の old_values コレクションが必要です。 UPDATE の場合、LCR には、空ではない new_values コレクション、および空でない old_values コレクションが必要です。 DELETE の場合、LCR には、NULL または空の new_values コレクション、および空ではない old_values コレクションが必要です。 LOB ERASE、LOB WRITE または LOB TRIM の場合、LCR には、空ではない new_values コレクション、および空または NULL の old_values コレクションが必要です。
object_owner	行の変更が発生した表を所有するユーザー。このパラメータは、NULL 以外の値に設定する必要があります。
object_name	DML 文が実行された表。このパラメータは、NULL 以外の値に設定する必要があります。
tag	LCR の追跡を有効にするバイナリ・タグ。たとえば、適用による転送を使用する場合は、DML 変更の元のソース・データベースを判断するためにこのタグを使用します。 **関連項目：**タグの詳細は、『Oracle9*i* Streams』を参照してください。
transaction_id	トランザクションの識別子。

表 108-10 LCR$_ROW_RECORD コンストラクタ・ファンクションのパラメータ（続き）

パラメータ	説明
scn	変更レコードが REDO に書き込まれたときの SCN。
old_values	DML 文の前にある行内の列値（DML 文が UPDATE 文または DELETE 文の場合）。
new_values	DML 文の後にある行内の列値（DML 文が UPDATE 文または INSERT 文の場合）。 補足的にログされた列および関連する LOB 情報（LCR によって LOB 操作が反映される場合）。

LCR$_ROW_RECORD サブプログラムの要約

表 108-11 LCR$_ROW_RECORD サブプログラム

サブプログラム	説明
共通のサブプログラム	SYS.LCR$_ROW_RECORD タイプと SYS.LCR$_DDL_RECORD タイプに共通するサブプログラムのリストは、108-26 ページの「LCR$_ROW_RECORD と LCR$_DDL_RECORD に共通するサブプログラム」を参照してください。
「ADD_COLUMN メンバー・プロシージャ」108-16 ページ	指定した値タイプに応じて、古い値または新しい値として値を列に追加します。
「DELETE_COLUMN メンバー・プロシージャ」108-17 ページ	指定した値タイプに応じて、古い値または新しい値（あるいはその両方）を指定の列から削除します。
「EXECUTE メンバー・プロシージャ」108-18 ページ	現行ユーザーのセキュリティ・ドメイン下で LCR を実行します。
「GET_LOB_INFORMATION メンバー・ファンクション」108-18 ページ	列の LOB 情報を取得します。
「GET_LOB_OFFSET メンバー・ファンクション」108-19 ページ	指定した列の LOB オフセットを戻します。
「GET_LOB_OPERATION_SIZE メンバー・ファンクション」108-20 ページ	LOB 列の操作サイズを取得します。
「GET_VALUE メンバー・ファンクション」108-20 ページ	指定した値タイプに応じて、指定の列の古い値または新しい値を戻します。

表 108-11 LCR$_ROW_RECORD サブプログラム（続き）

サブプログラム	説明
「GET_VALUES メンバー・ファンクション」108-21 ページ	指定した値タイプに応じて、古い値または新しい値のリストを戻します。
「RENAME_COLUMN メンバー・プロシージャ」108-21 ページ	LCR の列名を変更します。
「SET_LOB_INFORMATION メンバー・プロシージャ」108-22 ページ	列の LOB 情報を設定します。
「SET_LOB_OFFSET メンバー・プロシージャ」108-23 ページ	指定した列の LOB オフセットを設定します。
「SET_LOB_OPERATION_SIZE メンバー・プロシージャ」108-24 ページ	LOB 列の操作サイズを設定します。
「SET_VALUE メンバー・プロシージャ」108-24 ページ	指定した列の値を上書きします。
「SET_VALUES メンバー・プロシージャ」108-25 ページ	指定した値タイプに応じて、LCR の既存の古い値または新しい値を置換します。

ADD_COLUMN メンバー・プロシージャ

指定した値タイプに応じて、古い値または新しい値として値を列に追加します。列に同じタイプの値がすでに存在する場合は、エラーが発生します。

すでに存在する列値を設定するには、SET_VALUE を実行します。

関連項目： 108-24 ページ「SET_VALUE メンバー・プロシージャ」

構文

```
MEMBER PROCEDURE ADD_COLUMN(
   value_type     IN VARCHAR2,
   column_name    IN VARCHAR2,
   column_value   IN SYS.AnyData);
```

パラメータ

表 108-12 ADD_COLUMN プロシージャのパラメータ

パラメータ	説明
value_type	列に追加する値のタイプ。列に古い値を追加するには、old を指定します。列に新しい値を追加するには、new を指定します。
column_name	列名。この名前は検証されません。指定した名前が無効な場合は、LCR のアプリケーション中にエラーが発生する場合があります。
column_value	列値。NULL の場合はエラーが発生します。 NULL 値を SYS.AnyData ラッパーにカプセル化すると、列値に NULL を指定できます。

DELETE_COLUMN メンバー・プロシージャ

指定した値タイプに応じて、古い値または新しい値（あるいはその両方）を指定の列から削除します。

構文

```
MEMBER PROCEDURE DELETE_COLUMN(
   column_name    IN VARCHAR2,
   value_type     IN VARCHAR2  DEFAULT '*');
```

パラメータ

表 108-13 DELETE_COLUMN プロシージャのパラメータ

パラメータ	説明
column_name	列名。LCR に列が存在しない場合は、エラーが発生します。
value_type	列から削除する値のタイプ。列の古い値を削除するには、old を指定します。列の新しい値を削除するには、new を指定します。* を指定すると、古い値と新しい値の両方が削除されます。

EXECUTE メンバー・プロシージャ

現行ユーザーのセキュリティ・ドメイン下で行の LCR を実行します。このプロシージャを使用して LCR が適用された場合、LCR に対して実行される適用プロセスのハンドラは実行されません。

> **注意：** EXECUTE メンバー・プロシージャを起動できるのは、適用プロセスの適用ハンドラ内のみです。

構文

```
MEMBER PROCEDURE EXECUTE(
   conflict_resolution    IN BOOLEAN);
```

パラメータ

表 108-14 EXECUTE プロシージャのパラメータ

パラメータ	説明
conflict_resolution	true の場合は、DBMS_APPLY_ADM パッケージの SET_UPDATE_CONFLICT_HANDLER プロシージャを使用して表に定義した競合解消が、LCR 実行の結果として生じた競合の解消に使用されます。 false の場合、競合解消は使用されません。

GET_LOB_INFORMATION メンバー・ファンクション

列の LOB 情報を取得します。

戻り値は次のいずれかになります。

```
DBMS_LCR.NOT_A_LOB        CONSTANT NUMBER := 1;
DBMS_LCR.NULL_LOB         CONSTANT NUMBER := 2;
DBMS_LCR.INLINE_LOB       CONSTANT NUMBER := 3;
DBMS_LCR.EMPTY_LOB        CONSTANT NUMBER := 4;
DBMS_LCR.LOB_CHUNK        CONSTANT NUMBER := 5;
DBMS_LCR.LAST_LOB_CHUNK   CONSTANT NUMBER := 6;
```

指定した列が存在しない場合は、NULL を戻します。

構文

```
MEMBER FUNCTION GET_LOB_INFORMATION(
  value_type   IN VARCHAR2,
  column_name  IN VARCHAR2)
RETURN NUMBER;
```

パラメータ

表 108-15 GET_LOB_INFORMATION ファンクションのパラメータ

パラメータ	説明
value_type	列に戻す値のタイプ。old または new のいずれかです。
column_name	列の名前。

GET_LOB_OFFSET メンバー・ファンクション

指定した列の LOB オフセットを、文字数（CLOB 列の場合）またはバイト数（BLOB 列の場合）で取得します。次のすべての条件を満たす場合のみ、NULL 以外の値を戻します。

- 列の値が存在する場合。
- 列値が行外の LOB の場合。つまり、情報が DBMS_LCR.LAST_LOB_CHUNK または DBMS_LCR.LOB_CHUNK の場合。
- コマンド・タイプが LOB ERASE または LOB WRITE の場合。

これらの条件を満たさない場合は、NULL を戻します。

構文

```
GET_LOB_OFFSET(
   value_type      IN VARCHAR2,
   column_name     IN VARCHAR2)
RETURN NUMBER;
```

パラメータ

表 108-16 GET_LOB_OFFSET ファンクションのパラメータ

パラメータ	説明
value_type	列に戻す値のタイプ。現在は、new のみ指定できます。
column_name	LOB 列の名前。

GET_LOB_OPERATION_SIZE メンバー・ファンクション

LOB 列の操作サイズを、文字数（CLOB 列の場合）またはバイト数（BLOB 列の場合）で取得します。次のすべての条件を満たす場合のみ、NULL 以外の値を戻します。

- 列の値が存在する場合。
- 列値が行外の LOB の場合。
- コマンド・タイプが LOB ERASE または LOB TRIM の場合。
- 情報が DBMS_LCR.LAST_LOB_CHUNK の場合。

これらの条件を満たさない場合は、NULL を戻します。

構文

```
MEMBER FUNCTION GET_LOB_OPERATION_SIZE(
  value_type    IN VARCHAR2,
  column_name   IN VARCHAR2)
RETURN NUMBER,
```

パラメータ

表 108-17 GET_LOB_OPERATION_SIZE ファンクションのパラメータ

パラメータ	説明
value_type	列に戻す値のタイプ。現在は、new のみ指定できます。
column_name	LOB 列の名前。

GET_VALUE メンバー・ファンクション

指定した値タイプに応じて、指定の列の古い値または新しい値を戻します。

構文

```
MEMBER FUNCTION GET_VALUE(
   value_type          IN VARCHAR2,
   column_name         IN VARCHAR2)
RETURN SYS.AnyData;
```

パラメータ

表 108-18 GET_VALUE ファンクションのパラメータ

パラメータ	説明
value_type	列に戻す値のタイプ。列の古い値を取得するには、old を指定します。列の新しい値を取得するには、new を指定します。
column_name	列名。列が存在し、その値が NULL の場合は、NULL 値を含む SYS.AnyData インスタンスを戻します。列値が存在しない場合は、NULL を戻します。

GET_VALUES メンバー・ファンクション

指定した値タイプに応じて、古い値または新しい値のリストを戻します。

構文

```
MEMBER FUNCTION GET_VALUES(
   value_type    IN VARCHAR2)
RETURN SYS.LCR$_ROW_LIST;
```

パラメータ

表 108-19 GET_VALUES ファンクションのパラメータ

パラメータ	説明
value_type	戻す値のタイプ。古い値のリストを戻す場合は、old を指定します。新しい値のリストを戻す場合は、new を指定します。

RENAME_COLUMN メンバー・プロシージャ

LCR の列名を変更します。

構文

```
MEMBER PROCEDURE RENAME_COLUMN(
   from_column_name   IN VARCHAR2,
   to_column_name     IN VARCHAR2,
   value_type         IN VARCHAR2  DEFAULT '*');
```

パラメータ

表 108-20 RENAME_COLUMN プロシージャのパラメータ

パラメータ	説明
from_column_name	既存の列名。
to_column_name	新しい列名。指定した名前の列がすでに存在する場合は、エラーが発生します。
value_type	列名を変更する値のタイプ。 列の古い値の名前を変更するには、old を指定します。古い値が LCR に存在しない場合は、エラーが発生します。 列の新しい値の名前を変更するには、new を指定します。新しい値が LCR に存在しない場合は、エラーが発生します。 * を指定すると、古い列名と新しい列名の両方が変更されます。いずれかの値が LCR に存在しない場合は、エラーが発生します。

SET_LOB_INFORMATION メンバー・プロシージャ

列に LOB 情報を設定します。

構文

```
MEMBER PROCEDURE SET_LOB_INFORMATION(
  value_type          IN      VARCHAR2,
  column_name         IN      VARCHAR2,
  lob_information     IN      NUMBER);
```

パラメータ

表 108-21 SET_LOB_INFORMATION プロシージャのパラメータ

パラメータ	説明
value_type	列に設定する値のタイプ。old または new のいずれかです。old は、lob_information が DBMS_LCR.NOT_A_LOB に設定されている場合のみ指定します。
column_name	列の名前。列値が存在しない場合は、例外が発生します。このパラメータは、LOB 以外の列に対して設定する必要があります。

表 108-21 SET_LOB_INFORMATION プロシージャのパラメータ（続き）

パラメータ	説明
lob_information	次のいずれかの値を指定します。 DBMS_LCR.NOT_A_LOB CONSTANT NUMBER := 1; DBMS_LCR.NULL_LOB CONSTANT NUMBER := 2; DBMS_LCR.INLINE_LOB CONSTANT NUMBER := 3; DBMS_LCR.EMPTY_LOB CONSTANT NUMBER := 4; DBMS_LCR.LOB_CHUNK CONSTANT NUMBER := 5; DBMS_LCR.LAST_LOB_CHUNK CONSTANT NUMBER := 6;

SET_LOB_OFFSET メンバー・プロシージャ

指定した列の LOB オフセットを、文字数（CLOB 列の場合）またはバイト数（BLOB 列の場合）で設定します。

構文

```
SET_LOB_OFFSET(
   value_type      IN VARCHAR2,
   column_name     IN VARCHAR2,
   lob_offset      IN NUMBER);
```

パラメータ

表 108-22 SET_LOB_OFFSET プロシージャのパラメータ

パラメータ	説明
value_type	列に設定する値のタイプ。現在は、new のみ指定できます。
column_name	列名。LCR に列値が存在しない場合は、エラーが発生します。
lob_offset	LOB オフセット番号。有効な値は、NULL または DBMS_LOB.LOBMAXSIZE 以下の正の整数です。

SET_LOB_OPERATION_SIZE メンバー・プロシージャ

LOB 列の操作サイズを、文字数（CLOB 列の場合）またはバイト数（BLOB 列の場合）で設定します。

構文

```
MEMBER PROCEDURE SET_LOB_OPERATION_SIZE(
  value_type          IN      VARCHAR2,
  column_name         IN      VARCHAR2,
  lob_operation_size  IN      NUMBER);
```

パラメータ

表 108-23 SET_LOB_OPERATION_SIZE プロシージャのパラメータ

パラメータ	説明
value_type	列に設定する値のタイプ。現在は、new のみ指定できます。
column_name	LOB 列の名前。LCR に列値が存在しない場合は、例外が発生します。
lob_operation_size	LOB に関する lob_information が DBMS_LCR.LAST_LOB_CHUNK の場合は、有効な LOB ERASE 値または有効な LOB TRIM 値のいずれかに設定できます。LOB_ERASE 値は、DBMS_LOB.LOBMAXSIZE 以下の正の整数であることが必要です。LOB_TRIM 値は、DBMS_LOB.LOBMAXSIZE 以下の負でない整数であることが必要です。 それ以外の場合は、NULL に設定します。

SET_VALUE メンバー・プロシージャ

指定した列の古い値または新しい値を上書きします。

列の古い値を上書きする理由の 1 つは、競合によって発生したエラーを解決するためです。

構文

```
MEMBER PROCEDURE SET_VALUE(
   value_type       IN VARCHAR2,
   column_name      IN VARCHAR2,
   column_value     IN SYS.AnyData);
```

パラメータ

表 108-24 SET_VALUE プロシージャのパラメータ

パラメータ	説明
value_type	設定する値のタイプ。列の古い値を設定するには、old を指定します。列の新しい値を設定するには、new を指定します。
column_name	列名。指定の column_type に対して指定の column_value が LCR に存在しない場合は、エラーが発生します。
column_value	列の新しい値。NULL を指定すると、エラーが発生します。値に NULL を設定するには、NULL を SYS.AnyData インスタンスにカプセル化します。

SET_VALUES メンバー・プロシージャ

指定した値タイプに応じて、LCR のすべての古い値またはすべての新しい値を置換します。

構文

```
MEMBER PROCEDURE SET_VALUES(
   value_type     IN VARCHAR2,
   value_list     IN SYS.LCR$_ROW_LIST);
```

パラメータ

表 108-25 SET_VALUES プロシージャのパラメータ

パラメータ	説明
value_type	置換する値のタイプ。古い値を置換するには、old を指定します。新しい値を置換するには、new を指定します。
value_list	既存のリストを置換する LOV。すべての値を削除するには、NULL または空のリストを使用します。

LCR$_ROW_RECORD と LCR$_DDL_RECORD に共通するサブプログラム

次に、LCR$_ROW_RECORD タイプと LCR$_DDL_RECORD タイプに共通するファンクションとプロシージャを示します。

関連項目： 各タイプ固有のサブプログラムについては、次の項を参照してください。

- 108-6 ページ「LCR$_DDL_RECORD サブプログラムの要約」
- 108-15 ページ「LCR$_ROW_RECORD サブプログラムの要約」

表 108-26 Row タイプと DDL タイプに共通するサブプログラムの要約

サブプログラム	説明
「GET_COMMAND_TYPE メンバー・ファンクション」108-27 ページ	LCR のコマンド・タイプを戻します。
「GET_OBJECT_NAME メンバー・ファンクション」108-27 ページ	LCR によって変更するオブジェクトの名前を戻します。
「GET_OBJECT_OWNER メンバー・ファンクション」108-27 ページ	LCR によって変更するオブジェクトの所有者を戻します。
「GET_SCN メンバー・ファンクション」108-28 ページ	LCR のシステム変更番号（SCN）を戻します。
「GET_SOURCE_DATABASE_NAME メンバー・ファンクション」108-28 ページ	ソース・データベース名を戻します。
「GET_TAG メンバー・ファンクション」108-28 ページ	LCR のタグを戻します。
「GET_TRANSACTION_ID メンバー・ファンクション」108-28 ページ	LCR のトランザクション識別子を戻します。
「IS_NULL_TAG メンバー・ファンクション」108-28 ページ	LCR のタグが NULL の場合は Y を戻し、LCR のタグが NULL でない場合は N を戻します。
「SET_COMMAND_TYPE メンバー・プロシージャ」108-29 ページ	コマンド・タイプを設定します。
「SET_OBJECT_NAME メンバー・プロシージャ」108-29 ページ	LCR によって変更するオブジェクトの名前を設定します。

表 108-26 Row タイプと DDL タイプに共通するサブプログラムの要約（続き）

サブプログラム	説明
「SET_OBJECT_OWNER メンバー・プロシージャ」108-30 ページ	LCR によって変更するオブジェクトの所有者を設定します。
「SET_SOURCE_DATABASE_NAME メンバー・プロシージャ」108-30 ページ	LCR によって変更するオブジェクトのソース・データベース名を設定します。
「SET_TAG メンバー・プロシージャ」108-31 ページ	LCR にタグを設定します。

GET_COMMAND_TYPE メンバー・ファンクション

LCR のコマンド・タイプを戻します。

関連項目： コマンド・タイプの完全なリストは、『Oracle Call Interface プログラマーズ・ガイド』の「SQL コマンド・コード」の表を参照してください。

構文

```
MEMBER FUNCTION GET_COMMAND_TYPE RETURN VARCHAR2;
```

GET_OBJECT_NAME メンバー・ファンクション

LCR によって変更するオブジェクトの名前を戻します。

構文

```
MEMBER FUNCTION GET_OBJECT_NAME RETURN VARCHAR2;
```

GET_OBJECT_OWNER メンバー・ファンクション

LCR によって変更するオブジェクトの所有者を戻します。

構文

```
MEMBER FUNCTION GET_OBJECT_OWNER RETURN VARCHAR2;
```

GET_SCN メンバー・ファンクション

LCR のシステム変更番号（SCN）を戻します。

構文

```
MEMBER FUNCTION GET_SCN RETURN NUMBER;
```

GET_SOURCE_DATABASE_NAME メンバー・ファンクション

ソース・データベース名のグローバル名を戻します。ソース・データベースとは、変更が発生したデータベースです。

構文

```
MEMBER FUNCTION GET_SOURCE_DATABASE_NAME RETURN VARCHAR2;
```

GET_TAG メンバー・ファンクション

LCR のタグを戻します。LCR のタグとは、LCR の追跡を有効にするバイナリ・タグです。たとえば、適用による転送を使用する場合は、DML 変更または DDL 変更の元のソース・データベースを判断するためにこのタグを使用します。

関連項目： タグの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
MEMBER FUNCTION GET_TAG RETURN RAW;
```

GET_TRANSACTION_ID メンバー・ファンクション

LCR のトランザクション識別子を戻します。

構文

```
MEMBER FUNCTION GET_TRANSACTION_ID RETURN VARCHAR2;
```

IS_NULL_TAG メンバー・ファンクション

LCR のタグが NULL の場合は Y を戻し、LCR のタグが NULL でない場合は N を戻します。

関連項目： タグの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
MEMBER FUNCTION IS_NULL_TAG RETURN VARCHAR2;
```

SET_COMMAND_TYPE メンバー・プロシージャ

コマンド・タイプを設定します。指定したコマンド・タイプが解析できない場合は、エラーが発生します。たとえば、INSERT を GRANT に変更すると、エラーが発生します。

関連項目：

- 108-4 ページの「LCR$_DDL_RECORD コンストラクタ・ファンクションのパラメータ」の command_type パラメータの説明
- 108-14 ページの「LCR$_ROW_RECORD コンストラクタ・ファンクションのパラメータ」の command_type パラメータの説明
- コマンド・タイプの完全なリストは、『Oracle Call Interface プログラマーズ・ガイド』の「SQL コマンド・コード」の表を参照してください。

構文

```
MEMBER PROCEDURE SET_COMMAND_TYPE(
   command_type     IN VARCHAR2);
```

パラメータ

表 108-27 SET_COMMAND_TYPE プロシージャのパラメータ

パラメータ	説明
command_type	コマンド・タイプ。このパラメータは、NULL 以外の値に設定する必要があります。

SET_OBJECT_NAME メンバー・プロシージャ

LCR によって変更するオブジェクトの名前を設定します。

構文

```
MEMBER PROCEDURE SET_OBJECT_NAME(
   object_name      IN VARCHAR2);
```

パラメータ

表 108-28 SET_OBJECT_NAME プロシージャのパラメータ

パラメータ	説明
object_name	オブジェクトの名前。

SET_OBJECT_OWNER メンバー・プロシージャ

LCR によって変更するオブジェクトの所有者を設定します。

構文

```
MEMBER PROCEDURE SET_OBJECT_OWNER(
   object_owner IN VARCHAR2);
```

パラメータ

表 108-29 SET_OBJECT_OWNER プロシージャのパラメータ

パラメータ	説明
object_owner	オブジェクトを含むスキーマ。

SET_SOURCE_DATABASE_NAME メンバー・プロシージャ

LCR によって変更するオブジェクトのソース・データベース名を設定します。

構文

```
MEMBER PROCEDURE SET_SOURCE_DATABASE_NAME(
   source_database_name    IN VARCHAR2);
```

パラメータ

表 108-30 SET_SOURCE_DATABASE_NAME プロシージャのパラメータ

パラメータ	説明
source_database_name	変更対象のソース・データベース。ドメイン名を含めていない場合は、ローカル・ドメインがデータベース名に自動的に追加されます。たとえば、DBS1 を指定し、ローカル・ドメインが .NET の場合は、DBS1.NET が自動的に指定されます。このパラメータは、NULL 以外の値に設定する必要があります。

SET_TAG メンバー・プロシージャ

LCR にタグを設定します。LCR のタグとは、LCR の追跡を有効にするバイナリ・タグです。たとえば、適用による転送を使用する場合は、変更対象の元のソース・データベースを判断するためにこのタグを使用します。

関連項目： タグの詳細は、『Oracle9*i* Streams』を参照してください。

構文

```
MEMBER PROCEDURE SET_TAG(
   tag    IN RAW);
```

パラメータ

表 108-31 SET_TAG プロシージャのパラメータ

パラメータ	説明
tag	LCR のバイナリ・タグ。タグ値の最大サイズは 2KB です。

LCR$_ROW_LIST タイプ

表内の行の列値リストを識別します。

このタイプは、LCR$_ROW_UNIT タイプを使用し、LCR$_ROW_RECORD タイプで使用されます。

関連項目：

- 108-33 ページ「LCR$_ROW_UNIT タイプ」
- 108-13 ページ「LCR$_ROW_RECORD タイプ」

構文

```
CREATE TYPE SYS.LCR$_ROW_LIST AS TABLE OF SYS.LCR$_ROW_UNIT;
```

LCR$_ROW_UNIT タイプ

行内の列の値を識別します。

このタイプは、LCR$_ROW_LIST タイプで使用されます。

関連項目： 108-32 ページ「LCR$_ROW_LIST タイプ」

構文

```
CREATE TYPE LCR$_ROW_UNIT AS OBJECT (
  column_name         VARCHAR2(4000),
  data                SYS.AnyData,
  lob_information     NUMBER,
  lob_offset          NUMBER,
  lob_operation_size  NUMBER);
```

属性

表 108-32 LCR$_ROW_UNIT の属性

属性	説明
column_name	列の名前。
data	列に含まれるデータ。
lob_information	列の LOB 情報が含まれ、次のいずれかの値が含まれます。 DBMS_LCR.NOT_A_LOB CONSTANT NUMBER := 1; DBMS_LCR.NULL_LOB CONSTANT NUMBER := 2; DBMS_LCR.INLINE_LOB CONSTANT NUMBER := 3; DBMS_LCR.EMPTY_LOB CONSTANT NUMBER := 4; DBMS_LCR.LOB_CHUNK CONSTANT NUMBER := 5; DBMS_LCR.LAST_LOB_CHUNK CONSTANT NUMBER := 6;

表 108-32 LCR$_ROW_UNIT の属性（続き）

属性	説明
lob_offset	文字数（CLOB 列の場合）またはバイト数（BLOB 列の場合）で指定した LOB オフセット。有効な値は、NULL、または DBMS_LOB.LOBMAXSIZE 以下の正の整数です。
lob_operation_size	LOB の lob_information が DBMS_LCR.LAST_LOB_CHUNK の場合は、有効な LOB ERASE 値または有効な LOB TRIM 値のいずれかに設定できます。LOB_ERASE 値は、DBMS_LOB.LOBMAXSIZE 以下の正の整数であることが必要です。LOB_TRIM 値は、DBMS_LOB.LOBMAXSIZE 以下の負でない整数であることが必要です。 lob_information が DBMS_LCR.LAST_LOB_CHUNK でない場合、およびその他のすべての操作が行われた場合は、NULL になります。

109

ルール・タイプ

この章では、ルール、ルール・セットおよび評価コンテキストで使用するタイプについて説明します。

この章では、次の項目について説明します。

ルール・タイプは、オラクル社が提供する次の PL/SQL パッケージで使用します。

- DBMS_RULE
- DBMS_RULE_ADM
- DBMS_STREAMS_ADM

DBMS_RULE_ADM パッケージは、ルール、ルール・セットおよび評価コンテキストを作成して管理するために使用でき、DBMS_RULE パッケージは、ルールを評価するために使用できます。Streams を使用する場合は、ルールによって、取得プロセスで取得する変更、伝播ジョブで伝播するイベント、および適用プロセスでデキューおよび適用するイベントが判断されます。DBMS_STREAMS_ADM パッケージでは、取得、伝播および適用時に使用するシステム生成ルールが作成されます。

関連項目：

- 『Oracle9*i* Streams』
- 第 63 章「DBMS_RULE」
- 第 64 章「DBMS_RULE_ADM」

ルール・タイプ

表 109-1 DBMS_RULE タイプ

データ構造	説明
「RE$ATTRIBUTE_VALUE タイプ」 109-4 ページ	変数の属性の値を指定します。
「RE$ATTRIBUTE_VALUE_LIST タイプ」 109-4 ページ	ルール評価コンテキストで使用する属性値のリストを識別します。
「RE$COLUMN_VALUE タイプ」 109-5 ページ	表列の値を指定します。
「RE$COLUMN_VALUE_LIST タイプ」 109-5 ページ	ルール評価コンテキストで使用する列値のリストを識別します。
「RE$NAME_ARRAY タイプ」 109-6 ページ	名前のリストを識別します。
「RE$NV_ARRAY タイプ」 109-6 ページ	名前と値の組合せのリストを識別します。
「RE$NV_LIST タイプ」 109-6 ページ	名前と値の組合せリストとそのリストを操作するメソッドを含むオブジェクトを識別します。このオブジェクト・タイプを使用して、ルールに対するイベント・コンテキストとアクション・コンテキストを表します。
「RE$NV_NODE タイプ」 109-8 ページ	名前と値の組合せを識別します。
「RE$RULE_HIT タイプ」 109-9 ページ	評価の結果として検出されたルールを指定します。
「RE$RULE_HIT_LIST タイプ」 109-9 ページ	評価の結果として検出されたルールのリストを識別します。
「RE$TABLE_ALIAS タイプ」 109-10 ページ	ルール評価コンテキストで使用する別名に対応する表を指定します。
「RE$TABLE_ALIAS_LIST タイプ」 109-10 ページ	ルール評価コンテキストで使用する表の別名のリストを識別します。
「RE$TABLE_VALUE タイプ」 109-11 ページ	ROWID を使用して、表の行の値を指定します。
「RE$TABLE_VALUE_LIST タイプ」 109-11 ページ	ルール評価コンテキストで使用する表の値のリストを識別します。
「RE$VARIABLE_TYPE タイプ」 109-12 ページ	ルール評価コンテキストで使用する変数のタイプを指定します。

表 109-1 DBMS_RULE タイプ（続き）

データ構造	説明
「RE$VARIABLE_TYPE_LIST タイプ」 109-13 ページ	ルール評価コンテキストで使用する変数とそのタイプのリストを識別します。
「RE$VARIABLE_VALUE タイプ」 109-14 ページ	変数の値を指定します。
「RE$VARIABLE_VALUE_LIST タイプ」 109-14 ページ	ルール評価コンテキストで使用する変数値のリストを識別します。

RE$ATTRIBUTE_VALUE タイプ

変数の属性の値を指定します。

構文

```
TYPE SYS.RE$ATTRIBUTE_VALUE (
   variable_name       VARCHAR2(32),
   attribute_name      VARCHAR2(4000),
   attribute_value     SYS.AnyData);
```

属性

表 109-2 RE$ATTRIBUTE_VALUE の属性

属性	説明
variable_name	ルールで使用する変数を指定します。
attribute_name	属性名を指定します。
attribute_value	属性値を指定します。

RE$ATTRIBUTE_VALUE_LIST タイプ

ルール評価コンテキストで使用する属性値のリストを識別します。

構文

```
TYPE SYS.RE$ATTRIBUTE_VALUE_LIST AS VARRAY(1024) OF SYS.RE$ATTRIBUTE_VALUE;
```

RE$COLUMN_VALUE タイプ

表列の値を指定します。

構文

```
TYPE SYS.RE$COLUMN_VALUE (
   table_alias       VARCHAR2(32),
   column_name       VARCHAR2(4000),
   column_value      SYS.AnyData);
```

属性

表 109-3 RE$COLUMN_VALUE の属性

属性	説明
table_alias	ルールの表に使用する別名を指定します。
column_name	列名を指定します。
column_value	列値を指定します。

RE$COLUMN_VALUE_LIST タイプ

ルール評価コンテキストで使用する列値のリストを識別します。

構文

```
TYPE SYS.RE$COLUMN_VALUE_LIST AS VARRAY(1024) OF SYS.RE$COLUMN_VALUE;
```

RE$NAME_ARRAY タイプ

名前のリストを識別します。

構文

```
TYPE SYS.RE$NAME_ARRAY AS VARRAY(1024) OF VARCHAR2(30);
```

RE$NV_ARRAY タイプ

名前と値の組合せのリストを識別します。

構文

```
TYPE SYS.RE$NV_ARRAY AS VARRAY(1024) OF SYS.RE$NV_NODE;
```

RE$NV_LIST タイプ

名前と値の組合せリストとそのリストを操作するメソッドを含むオブジェクトを識別します。このオブジェクト・タイプを使用して、ルール・セット評価に対するイベント・コンテキストとルールに対するアクション・コンテキストを表します。

構文

```
TYPE SYS.RE$NV_LIST AS OBJECT(
   actx_list    SYS.RE$NV_ARRAY);
```

属性

表 109-4 RE$NV_LIST の属性

属性	説明
actx_list	名前と値の組合せのリスト。

RE$NV_LIST サブプログラム

この項では、SYS.RE$NV_LIST タイプの次のメンバー・プロシージャおよびメンバー・ファンクションについて説明します。

- ADD_PAIR メンバー・プロシージャ
- GET_ALL_NAMES メンバー・ファンクション
- GET_VALUE メンバー・ファンクション
- REMOVE_PAIR メンバー・プロシージャ

ADD_PAIR メンバー・プロシージャ

名前と値の組合せを名前と値の組合せのリストに追加します。

構文

```
MEMBER PROCEDURE ADD_PAIR(
   name   IN VARCHAR2,
   value  IN SYS.AnyData);
```

パラメータ

表 109-5 ADD_PAIR プロシージャのパラメータ

パラメータ	説明
name	リストに追加する名前と値の組合せの名前。リスト内にすでに名前が存在する場合は、エラーが発生します。
value	リストに追加する名前と値の組合せの値。

GET_ALL_NAMES メンバー・ファンクション

名前と値の組合せのリスト内にあるすべての名前のリストを戻します。

構文

```
MEMBER FUNCTION GET_ALL_NAMES RETURN SYS.RE$NAME_ARRAY;
```

GET_VALUE メンバー・ファンクション

名前と値の組合せのリスト内の指定の名前に対応する値を戻します。

構文

```
MEMBER FUNCTION GET_VALUE(
   name     IN   VARCHAR2)
RETURN SYS.AnyData;
```

パラメータ

表 109-6 GET_VALUE ファンクションのパラメータ

パラメータ	説明
name	値を戻す名前。

REMOVE_PAIR メンバー・プロシージャ

指定した名前の名前と値の組合せを、名前と値の組合せのリストから削除します。

構文

```
MEMBER PROCEDURE REMOVE_PAIR(
   name     IN   VARCHAR2);
```

パラメータ

表 109-7 REMOVE_PAIR プロシージャのパラメータ

パラメータ	説明
name	削除する組合せの名前。

RE$NV_NODE タイプ

名前と値の組合せを識別します。

構文

```
TYPE SYS.RE$NV_NODE (
   nvn_name       VARCHAR2(30),
   nvn_value      SYS.AnyData);
```

属性

表 109-8 RE$NV_NODE の属性

属性	説明
nvn_name	名前と値の組合せの名前を指定します。
nvn_value	名前と値の組合せの値を指定します。

RE$RULE_HIT タイプ

評価の結果として検出されたルールを指定します。

関連項目：

- 64-10 ページ「CREATE_RULE プロシージャ」
- 64-5 ページ「ALTER_RULE プロシージャ」

構文

```
TYPE SYS.RE$RULE_HIT (
   rule_name              VARCHAR2(61),
   rule_action_context    RE$NV_LIST);
```

属性

表 109-9 RE$RULE_HIT の属性

属性	説明
rule_name	*schema_name.rule_name* 形式のルール名。たとえば、hr スキーマ内の employee_rule というルールは、hr.employee_rule の形式で戻されます。
rule_action_context	DBMS_RULE_ADM パッケージの CREATE_RULE プロシージャ、または ALTER_RULE プロシージャで指定したルール・アクション・コンテキスト。

RE$RULE_HIT_LIST タイプ

評価の結果として検出されたルールのリストを識別します。

構文

```
TYPE SYS.RE$RULE_HIT_LIST AS VARRAY(1024) OF SYS.RE$RULE_HIT;
```

RE$TABLE_ALIAS タイプ

ルール評価コンテキストで使用する別名に対応する表を指定します。指定する表名は、スキーマ・オブジェクトのネーミング規則に準拠する必要があります。

関連項目： スキーマ・オブジェクトのネーミング規則については、『Oracle9*i* SQL リファレンス』を参照してください。

構文

```
TYPE SYS.RE$TABLE_ALIAS IS OBJECT(
   table_alias   VARCHAR2(32),
   table_name    VARCHAR2(194));
```

属性

表 109-10 RE$TABLE_ALIAS の属性

属性	説明
table_alias	ルールの表に使用する別名。
table_name	別名によって参照される表名。シノニムも指定できます。表名は、評価コンテキスト・スキーマで解決されます。 形式は次のとおりです。 *schema_name.table_name* たとえば、*schema_name* が hr で、*table_name* が employees の場合は、次のように入力します。 hr.employees

RE$TABLE_ALIAS_LIST タイプ

ルール評価コンテキストで使用する表の別名のリストを識別します。

構文

```
TYPE SYS.RE$TABLE_ALIAS_LIST AS VARRAY(1024) OF SYS.RE$TABLE_ALIAS;
```

RE$TABLE_VALUE タイプ

ROWID を使用して、表の行の値を指定します。

構文

```
TYPE SYS.RE$TABLE_VALUE(
   table_alias       VARCHAR2(32),
   table_rowid       VARCHAR2(18));
```

属性

表 109-11 RE$TABLE_VALUE の属性

属性	説明
table_alias	ルールの表に使用する別名を指定します。
table_rowid	表の行の ROWID を指定します。

RE$TABLE_VALUE_LIST タイプ

ルール評価コンテキストで使用する表の値のリストを識別します。

構文

```
TYPE SYS.RE$TABLE_VALUE_LIST AS VARRAY(1024) OF SYS.RE$TABLE_VALUE;
```

RE$VARIABLE_TYPE タイプ

ルール評価コンテキストで使用する変数のタイプを指定します。指定する変数名は、スキーマ・オブジェクトのネーミング規則に準拠する必要があります。

関連項目： スキーマ・オブジェクトのネーミング規則については、『Oracle9*i* SQL リファレンス』を参照してください。

構文

```
TYPE SYS.RE$VARIABLE_TYPE (
   variable_name             VARCHAR2(32),
   variable_type             VARCHAR2(4000),
   variable_value_function   VARCHAR2(228),
   variable_method_function  VARCHAR2(228));
```

属性

表 109-12 RE$VARIABLE_TYPE の属性

属性	説明
variable_name	ルールで使用する変数名。
variable_type	評価コンテキスト・スキーマで解決するタイプ。有効な Oracle の組込みデータ・タイプ、ユーザー定義型またはオラクル社が提供するタイプを指定できます。これらのタイプの詳細は、『Oracle9*i* SQL リファレンス』を参照してください。
variable_value_function	暗黙的な変数に対して指定できる値関数。シノニムも指定できます。関数名は、評価コンテキスト・スキーマで解決されます。 詳細は、「使用上の注意」を参照してください。
variable_method_function	メソッド起動の結果を戻す値関数を指定します。変数に対してメソッドを起動する単純なルールが多数ある場合は、この関数を指定すると評価を高速化できます。関数には、シノニムまたはリモート関数も指定できます。 関数名は、評価コンテキスト・スキーマで解決されます。この関数は、評価コンテキストを使用するルール・セット、または評価コンテキストを使用するルールを含むルール・セットの所有者にかわって実行されます。 詳細は、「使用上の注意」を参照してください。

使用上の注意

variable_value_function パラメータおよび variable_method_function パラメータの両方で指定する関数は、次の形式になります。

schema_name.package_name.function_name@dblink

たとえば、*schema_name* が hr、*package_name* が var_pac、*function_name* が func_value、*dblink* が dbs1.net の場合は、次のように入力します。

```
hr.var_pac.func_value@dbs1.net
```

次の各項では、関数の署名について説明します。

variable_value_function の署名

この関数には、次の署名が必要です。

```
FUNCTION variable_value_func(
  evaluation_context_schema  IN VARCHAR2,
  evaluation_context_name    IN VARCHAR2,
  variable_name              IN VARCHAR2,
  event_context              IN SYS.RE$NV_LIST )
RETURN SYS.RE$VARIABLE_VALUE;
```

variable_method_function の署名

この関数には、次の署名が必要です。

```
FUNCTION variable_method_function(
  evaluation_context_schema  IN VARCHAR2,
  evaluation_context_name    IN VARCHAR2,
  variable_value             IN SYS.RE$VARIABLE_VALUE,
  method_name                IN VARCHAR2,
  event_context              IN SYS.RE$NV_LIST)
RETURN SYS.RE$ATTRIBUTE_VALUE;
```

RE$VARIABLE_TYPE_LIST タイプ

ルール評価コンテキストで使用する変数とそのタイプのリストを識別します。

構文

```
TYPE SYS.RE$VARIABLE_TYPE_LIST AS VARRAY(1024) OF SYS.RE$VARIABLE_TYPE;
```

RE$VARIABLE_VALUE タイプ

変数の値を指定します。

構文

```
TYPE SYS.RE$VARIABLE_VALUE (
   variable_name       VARCHAR2(32),
   variable_data       SYS.AnyData);
```

属性

表 109-13 RE$VARIABLE_VALUE の属性

属性	説明
variable_name	ルールで使用する変数名を指定します。
variable_data	変数値のデータを指定します。

RE$VARIABLE_VALUE_LIST タイプ

ルール評価コンテキストで使用する変数値のリストを識別します。

構文

```
TYPE SYS.RE$VARIABLE_VALUE_LIST AS VARRAY(1024) OF SYS.RE$VARIABLE_VALUE;
```

索引

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

あ

い

こ

さ

し

す

せ

そ

ち

て

と

ね

は

ひ

ふ

へ

ま

む

め

ゆ

り

る

れ

ろ