はじめに

■コース概要と目的

コンポジット型やバルク配列処理、動的な SQL 文、パッケージなど、Oracle 独自のプログラミング言語「PL/SQL」の応用的な記述方法について、実習を通して習得します。

■受講対象者

これから PL/SQL を使用してアプリケーション開発をされる方。

■前提条件

「SQL トレーニング」「PL/SQL プログラミング I 」コースを受講された方。もしくは、同等の知識をお持ちの方。

※DML 文、トランザクションについては既に習得済みであること。

■テキスト内の記述について

▼構文

[　]	省略可能
{ A | B }	A または B のどちらかを選択
n	数値の指定
_	デフォルト値

▼マーク

	指定バージョンからの新機能 （左記の場合、Oracle 12cR1 からの新機能）
	知っておいたほうが良いテクニック、もしくは注意事項
	参照ページ
	データ・ディクショナリ・ビュー

第 3 章

システム固有の動的 SQL

この章では、動的 SQL を使用した SQL 文の記述方法について説明します。

１.システム固有の動的 SQL

EXECUTE IMMEDIATE 文を使用するとシステム固有の動的 SQL を実行できます。動的 SQL はプログラム実行時に SQL を組立てることができるため、アクセス対象のオブジェクトが不明な場合などに有効です。

（１）静的 SQL と動的 SQL

PL/SQL プログラムの中で使用する SQL には、「静的 SQL」「動的 SQL」の２種類の記述方法があります。

１）静的 SQL

静的 SQL とは、プログラム内で指定された特定のオブジェクト（表や列）にアクセスする SQL です。SQL 自体はプログラムを通して変化することがありません。

２）動的 SQL

動的 SQL とは、プログラム内で指定するオブジェクトが実行時に変化する SQL です。検索条件を増減させることもできます。

■静的 SQL および動的 SQL の特徴

	静的 SQL	動的 SQL
SQL の内容	固定	可変
オブジェクト名	固定	可変
PL/SQL でサポートする SQL	SELECT 文 DML 文 トランザクション制御文	SELECT 文 DML 文 トランザクション制御文 DDL 文 セッション制御文 システム制御文

（２）動的 SQL 使用時の注意事項

静的 SQL はプログラムを通して変化しないため、参照オブジェクトのチェック、権限のチェックといったコンパイルはプログラム作成時に行われます。
一方、動的 SQL は実行するまでアクセスするオブジェクトなどが未定であるため、作成時ではなく実行時にコンパイルが行われます。そのため、実行時のオーバーヘッドが高くなり、静的 SQL に比べパフォーマンスが劣ります。したがって、静的 SQL では記述できない処理のみに動的 SQL を使用します。

■静的SQL

```
SELECT * FROM emp WHERE deptno = 条件値 ;
```

アクセス対象のオブジェクトは定まっており、パラメータを使用した条件値の受渡しができる。

■動的SQL

```
SELECT * FROM 表名 WHERE 列名 = 条件値 ;
```

条件値だけでなく、アクセス対象のオブジェクトを未定にすることができる。
パラメータを使用して条件値やオブジェクト名を受渡し、実行時にSQLを組立てることができる。

```
SELECT * FROM 表名 WHERE 列名 = 条件値
                         AND  列名 = 条件値 ;
```

条件値やアクセス対象のオブジェクトだけでなく、検索条件を増減させることもできる。

2.EXECUTE IMMEDIATE文

動的SQLを実行するにはEXECUTE IMMEDIATE文を使用します。EXECUTE IMMEDIATE文は静的SQLでは実行できないSQLも実行できます。

(1)動的SQLで実行できるSQL文

・動的SQL
  - DML文
  - SELECT...INTO文（複数行の結果を戻すSELECT文はカーソル変数を使用します）
・DDL文（CREATE文など）
・セッション制御文（ALTER SESSION文など）
・システム制御文（ALTER SYSTEM文）

「カーソル変数」（4章）

(2)EXECUTE IMMEDIATE文

EXECUTE IMMEDIATE文に続く文字列（テキスト）をSQLとして実行します。

```
EXECUTE IMMEDIATE SQLテキスト ;
```

■SQL組立て時の注意事項

EXECUTE IMMEDIATEに続く文字列はSQLとして実行されるため、組立てられた文字列がSQLの構文規則に即している必要があります。そのため、SQLを組立てる際は以下の点に注意してください。

・SQLとして認識させる文字列（SQLテキスト）を単一引用符（'）で囲みます。変数やパラメータに対しては必要ありません。
・SQL文中に変数やパラメータが含まれる場合は、コンカチネーション（||）で連結します。
・SQLテキストと変数・パラメータを連結する際、文字間に空白を設けないと実行時にエラーが発生します。

例）パラメータで表名と条件値を指定し、条件に合致した表の対象行を削除して画面に表示するプログラムを作成する。

```
SQL> CREATE OR REPLACE PROCEDURE delete_records(t_name VARCHAR2,clause VARCHAR2 DEFAULT NULL)
  2  IS
  3     where_clause   VARCHAR2(100) := ' WHERE ' || clause;
  4  BEGIN
  5     IF clause IS NULL
  6        THEN where_clause := NULL;
  7     END IF;
  8        EXECUTE IMMEDIATE 'DELETE FROM ' || t_name || where_clause;
  9           DBMS_OUTPUT.PUT_LINE('DELETE FROM ' || t_name || where_clause);
 10  END;
 11  /
```

文字データとパラメータを連結して DELETE文を組立て、実行する

例）上記プログラムを実行する。

```
SQL> EXECUTE delete_records('EMP','deptno = 10')
DELETE FROM EMP WHERE deptno = 10
```

組立てたSQLを画面に表示

■SQL の組立て方法

SQL テキスト　　変数・パラメータ

```
EXECUTE IMMEDIATE   'DELETE FROM ' || t_name ;
```

コンカチネーションで連結

・空白がないためエラー

```
EXECUTE IMMEDIATE 'DELETE FROM' || t_name;    ⇒   DELETE FROMEMP;
```

FROM と表名が連続している

・空白があるため実行可能

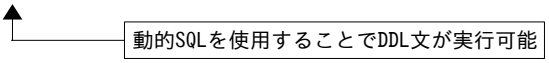
```
EXECUTE IMMEDIATE 'DELETE FROM ' || t_name;   ⇒   DELETE FROM EMP;
```

（３）DDL 文、セッション制御文、システム制御文の記述

EXECUTE IMMEDIATE 文を使用すると、PL/SQL プログラム内で DDL 文、セッション制御文、システム制御文を記述できます。例えば、PL/SQL プログラム内でオブジェクトを削除したい場合や、ALTER SESSION 文を使用して、そのセッションだけ初期化パラメータの値を変更したい場合に便利です。

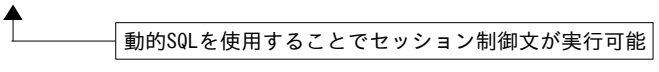
例）指定したオブジェクトを削除するプログラムを作成し、実行する。

```
SQL> CREATE OR REPLACE PROCEDURE drop_object(object_type VARCHAR2,name VARCHAR2)
  2  IS
  3  BEGIN
  4     EXECUTE IMMEDIATE 'DROP '|| object_type || name;
  5  END;
  6  /

プロシージャが作成されました。

SQL> EXECUTE drop_object('TABLE ','t1')

PL/SQL プロシージャが正常に完了しました。
```



例）PL/SQL プログラムの中で初期化パラメータ plsql_code_type の値を変更する。

```
BEGIN
   EXECUTE IMMEDIATE 'ALTER SESSION SET plsql_code_type = ''NATIVE''';
…省略…
END;
```



※単一引用符内で単一引用符を認識させるためには、二重にして指定する必要があります。

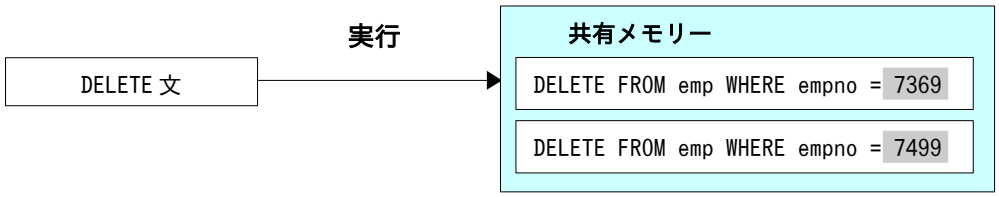
３．動的 SQL の効率化

静的・動的に関わらず、SQL は解析結果を共有できればデータベース全体のパフォーマンスが向上します。動的 SQL を使用する場合、プレースホルダを使用することで SQL 解析結果の共有率を高めることができます。

（１）SQL の解析結果の共有化

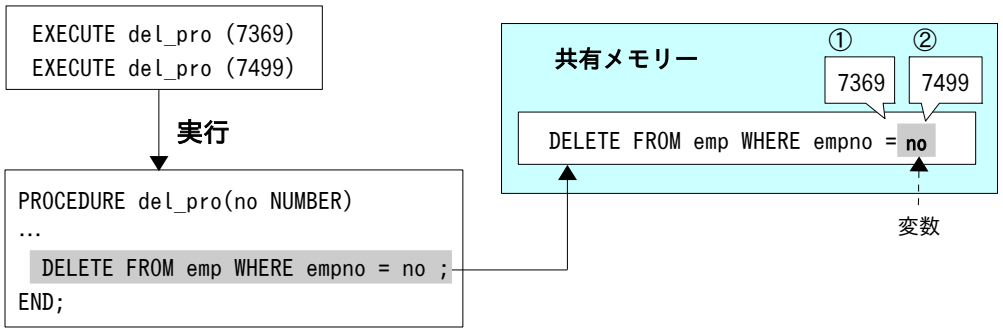
SQL を発行すると、SQL は解析・実行されます。SQL の解析結果はすぐには解放されず、しばらく共有メモリー上に保持されます。その後、再度同じ SQL を発行すると、残された解析結果を再利用できるため、パフォーマンスが向上します。
また、SQL の条件値だけを変更して SQL を実行する場合、条件値に変数を用いると解析結果を共有できます。

■変数を使用しない場合（SQL 解析結果を共有しにくい）



異なる条件値ごとに SQL 文を解析するため、条件値ごとに解析する分、パフォーマンスが低下する。

■変数を使用する場合（SQL 解析結果を共有しやすい）



①変数を含めて SQL を解析するため、条件値が未定な状態で解析結果が共有メモリー上に保持される。
②条件値に変数を用いている SQL の場合は、共有メモリー上に残されている解析結果を再利用できるため、解析にかかるオーバーヘッドを低減でき、パフォーマンスが向上する。

（２）動的 SQL の解析結果の共有化

動的 SQL 実行時に SQL 解析結果を共有するには、プレースホルダを使用する必要があります。

１）プレースホルダ

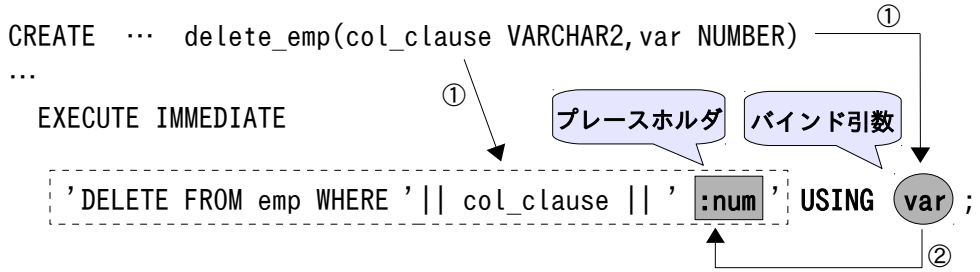
プレースホルダは動的 SQL に使用する変数のことです。宣言部で定義せずに、プレースホルダとする文字列の直前にコロン（:）を付けて使用します。

２）バインド引数

バインド引数はプレースホルダに代入する値のことです。USING 句を使用して、プレースホルダに受渡す値を指定します。
バインド引数は、SQL 中に指定したプレースホルダと位置で対応付いており、複数のプレースホルダを使用する場合はバインド引数の順序をプレースホルダに対応させる必要があります。

例）プレースホルダを用いて、パラメータで指定した社員を EMP 表から削除するプログラムを作成する。

```
CREATE OR REPLACE PROCEDURE delete_emp(col_clause VARCHAR2,var NUMBER)
IS
BEGIN
   EXECUTE IMMEDIATE 'DELETE FROM emp WHERE '|| col_clause || ':num' USING var;
END;
```



①ストアド・プログラムのパラメータを受渡し、実行 SQL の解析を行う。
②解析を行った後にバインド引数をプレースホルダに受渡す。

補足：下記例では、条件値を受渡した後に解析するため、解析結果を共有しにくい。

```
CREATE OR REPLACE PROCEDURE delete_emp(col_clause VARCHAR2,var NUMBER)
…
   EXECUTE IMMEDIATE 'DELETE FROM emp WHERE '|| col_clause || var;
…
```