Oracle PL/SQL

[Oracle Database10g PL/SQL]

http://otndnld.oracle.co.jp/document/products/oracle10 g/102 /doc_cd/appdev.102 /B19257 -01 /overview.html#1310

特長

・SQL のデータ操作機能と手続き型言語の処理機能の両方が利用できる。
・プログラム・フローを制御できる。
・変数の宣言、プロシージャとファンクションの定義、ランタイム・エラーのトラップが可能。
・複数のアプリケーション間でその定義済みコードを再利用できる。
・単純な SQL で問題が解決できる場合、PL/SQL プログラム内で SQL コマンドを直接発行できる。
・データ型は、SQL の列型と対応している。

ブロック構造

・プログラムを構成する基本単位（プロシージャ、ファンクションおよび無名ブロック）
・内部で相互にネストできる論理ブロック。
・宣言はブロックの中で局所的に有効。

変数の宣言

・任意の SQL データ型 (CHAR、DATE、NUMBER など) や、BOOLEAN、PLS_INTEGER などの PL/SQL 固有のデータ型を持つことがでる。
・宣言部の各行の最後には、セミコロン（;）を付ける。
・コンポジット・データ型 TABLE、VARRAY、RECORD を使用して、ネストした表、可変サイズの配列（VARRAY）およびレコードも宣言できる。

```
DECLARE
  part_no    NUMBER(6);
  part_name  VARCHAR2(20);
  in_stock   BOOLEAN;
  part_price NUMBER(6,2);
  part_desc  VARCHAR2(50);
```

変数への値の代入

変数に値を代入する方法は 3 つある。

コロンに等号を付けた代入演算子（:=）を使用する。

変数は演算子の左に、ファンクション・コールを含む式は演算子の右に置きます。変数を宣言するときに、変数に値を代入できます。

```
DECLARE
   wages          NUMBER;
   hours_worked   NUMBER := 40;
   hourly_salary  NUMBER := 22.50;
   bonus          NUMBER := 150;
   country        VARCHAR2(128);
   counter        NUMBER := 0;
```

```
   done           BOOLEAN;
   valid_id       BOOLEAN;
   emp_rec1       employees%ROWTYPE;
   emp_rec2       employees%ROWTYPE;
   TYPE commissions IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
   comm_tab       commissions;
BEGIN
   wages := (hours_worked * hourly_salary) + bonus;
   country := 'France';
   country := UPPER('Canada');
   done := (counter > 100);
   valid_id := TRUE;
   emp_rec1.first_name := 'Antonio';
   emp_rec1.last_name := 'Ortiz';
   emp_rec1 := emp_rec2;
   comm_tab(5) := 20000 * 0.15;
END;
/
```

データベース値を選択またはフェッチして代入する方法。

```
DECLARE
  bonus  NUMBER(8,2);
  emp_id NUMBER(6) := 100;
BEGIN
  SELECT salary * 0.10 INTO bonus FROM employees
    WHERE employee_id = emp_id;
END;
/
```

値を OUT パラメータまたは IN OUT パラメータとしてサブプログラムに渡し、サブプログラム内で代入する方法。

```
DECLARE
  new_sal NUMBER(8,2);
  emp_id  NUMBER(6) := 126;
  PROCEDURE adjust_salary(emp_id NUMBER, sal IN OUT NUMBER) IS
    emp_job VARCHAR2(10);
    avg_sal NUMBER(8,2);
    BEGIN
      SELECT job_id INTO emp_job FROM employees WHERE employee_id = emp_id;
      SELECT AVG(salary) INTO avg_sal FROM employees WHERE job_id = emp_job;
      DBMS_OUTPUT.PUT_LINE ('The average salary for ' || emp_job
                            || ' employees: ' || TO_CHAR(avg_sal));
      sal := (sal + avg_sal)/2; -- adjust sal value which is returned
    END;
BEGIN
  SELECT AVG(salary) INTO new_sal FROM employees;
  DBMS_OUTPUT.PUT_LINE ('The average salary for all employees: '
                        || TO_CHAR(new_sal));
  adjust_salary(emp_id, new_sal); -- assigns a new value to new_sal
  DBMS_OUTPUT.PUT_LINE ('The adjusted salary for employee ' || TO_CHAR(emp_id)
                        || ' is ' || TO_CHAR(new_sal)); -- sal has new value
END;
/
```

バインド変数

- SQL の INSERT、UPDATE、DELETE または SELECT 文を直接埋め込むと、WHERE 句および VALUES 句内の変数を自動的にバインド変数に変換する。
- 動的 SQL の場合は、変数を通常使用する部分（WHERE 句や VALUES 句など）にバインド変数を指定する。
  - リテラルおよび変数値を連結して単一の文字列にするかわりに、変数をバインド変数の名前（先頭にコロンを追加したもの）に置き換え、USING 句を使用して対応する PL/SQL 変数を指定します。

```
'DELETE FROM employees WHERE employee_id = :id' USING emp_id;
```

■定数の宣言

・キーワード CONSTANT を付ける
・定数に直接値を代入する必要がある。

```
credit_limit CONSTANT NUMBER := 5000.00;
```