Встроенный динамический SQL

Динамический SQL

PL/SQL использует раннее связывание для выполнения операторов SQL. Следствием этого является то, что только операторы DML могут непосредственно включаться в блоки PL/SQL. Однако можно решить эту проблему с помощью динамического SQL.

Динамический SQL разбирается и исполняется во время выполнения, а не синтаксического разбора блока PL/SQL.

Динамический SQL

Существуют два способа выполнения динамического SQL в PL/SQL.

1. Первый применяет модуль DBMS_SQL.
2. Второй способ был введен в Oracle8i и предлагает использование встроенного динамического SQL. Встроенный динамический SQL является составной частью самого языка. Вследствие этого он значительно проще в применении и быстрее, чем модуль DBMS_SQL.

EXECUTE IMMEDIATE.

Базовым оператором, используемым в не содержащих запросов операторах (DML и DDL) и блоках PL/SQL, является оператор EXECUTE IMMEDIATE.

Выполняемая строка может задаваться как литерал, заключенный в одиночные кавычки или как переменная типа символьной строки PL/SQL.

Завершающая точка с запятой не нужна для операторов DML и DDL, но указывается для анонимных блоков.

EXECUTE IMMEDIATE. Пример.

В этом примере показаны различные способы использования EXECUTE IMMEDIATE: для выполнения DDL, DML и анонимных блоков PL/SQL.

```
BEGIN
EXECUTE IMMEDIATE
        'CREATE TABLE execute_table (call VARCHAR2(10))';
FOR v_Counter IN 1..10 LOOP
    v_SQLString :=
        'INSERT INTO execute_table
        VALUES ("Row' || v_Counter || '")';
    EXECUTE IMMEDIATE v_SQLString;
END LOOP;
v_PLSQLBlock :=
    'BEGIN
        FOR v_Rec IN (SELECT * FROM execute_table) LOOP
        DBMS_OUTPUT.PUT_LINE(v_Rec.call);
        END LOOP;
    END;';
EXECUTE IMMEDIATE v_PLSQLBlock;
EXECUTE IMMEDIATE 'DROP TABLE execute_table ';
END;
```

EXECUTE IMMEDIATE.

EXECUTE IMMEDIATE используется также для выполнения операторов со связанными переменными.

В этом случае выполняемая строка содержит *специальные позиции,* помеченные двоеточием.

Позиции предназначены для размещения переменных PL/SQL, которые указываются в предложении USING оператора EXECUTE IMMEDIATE,

EXECUTE IMMEDIATE. Пример 2..

```
BEGIN
- Вставим ECN 103 в таблицу classes, используя строку
символов
- для оператора SQL.
v_SQLString :=
   'INSERT INTO CLASSES (department, course,
description,
      max_students, current_students,
      num_credits)
   VALU€S(:dep, :course, :descr, :max_s, :cur_s,
:num_c)';
EXECUTE IMMEDIATE v_SQLString USING
   'ECN', 103, 'Economics 103', 10, 0, 3;
- Зарегистрируем всех выбравших Economics в новой
группе.
FOR v_StudentRec IN c_EconMajor LOOP
   - Здесь мы имеем литеральный оператор SQL, а
переменные PL/SOL
   - находятся в предложении USING.
ECECUTE IMMEDIATE
   'INSERT INTO registered_students
      (student_ID, department, course, grade)
   VALUES (:id, :dep, :course, NULL)'
```

OPEN FOR

Запросы выполняются с помощью оператора OPEN FOR аналогично курсорным переменным. Различие состоит в том, что строка, содержащая запрос, может быть переменной PL/SQL, а не литералом.

К получаемой курсорной переменной можно обращаться так же, как и к любой другой переменной.

Для связывания используется предложение USING, так же как в операторе EXECUTE IMMEDIATE.

```
BEGIN
v_SQLStatement := 'SELECT * FROM students ' || p_WhereClause;
OPEN v_ReturnCursor FOR v_SQLStatement;

v_SQLStatement := 'SELECT * FROM students WHERE major = :m';
OPEN v_ReturnCursor FOR v_SQLStatement USING p_Major;

END;
```

Массовые соединения

Операторы SQL в блоках PL/SQL пересылаются системе поддержки SQL, которая в свою очередь может передавать данные назад системе поддержки PL/SQL (как результат запроса). Во многих случаях данные, которые вносятся или обновляются в базе данных, помещаются сначала в сборную конструкцию PL/SQL, и затем эта сборная конструкция просматривается с помощью цикла FOR для отправки информации системе поддержки SQL. Это приводит к переключению контекста между PL/SQL и SQL для каждой строки в сборной конструкции.

Oracle8i и выше позволяет передавать все строки сборной конструкции системе поддержки SQL с помощью одной операции, оставляя только одно переключение контекста. Это называется *массовым соединением,* оно выполняется с помощью оператора FORALL.

Массовые соединения. Пример.

```
DECLARE
   TYPE t_Numbers IS TABLE OF temp_table.num_col%TYPE;
   TYPE t_Strings IS TABLE OF temp_table.char_col%TYPE;
   v_Numbers t_Numbers := t_Numbers(1);
   v_Strings t_Strings := t_Strings(1);
- Печатаем общее число строк таблицы temp_table.
PROCEDURE PrintTotalRows (p_Message IN VARCHAR2) IS
   v_Count NUMBER:
BEGIN
   SELECT COUNT(*)
      INTO v_Count
      FROM temp_table;
   DBMS_OUTPUT.PUT_LINE(p_Message || ': Count is ' ||
v_Count);
END PrintTotalRows;
BEGIN
DELETE FROM temp_table;
-- Заполняем вложенные таблицы PL/SQL, используя 1000
значений.
v_Numbers.EXTEND(1000);
v_Strings.EXTEND(1000);
FOR v_Count IN 1..1000 LOOP
   v_Numbers(v_Count) := v_Count;
   v_Strings(v_Count) := 'Element #' || v_Count;
```

Массовые соединения. Пример (продолжение).

```
-- Внесем в базу данных все 1000 элементов с помощью оператора FORALL.
FORALL v_Count IN 1..1000
    INSERT INTO temp_table VALUES
        (v_Numbers(v_Count), v_Strings(v_Count));
- Теперь должно быть 1000 строк.
   PrintTotalRows('After first insert');
-- Снова внесем в базу данных элементы с 501 по 1000.
FORALL v_Count IN 501..1000
   INSERT INTO temp_table VALUES
   (v_Numbers(v_count), v_Strings(v_Count));
-- Теперь у нас должно быть 1500 строк.
   PrintTotalRows('After second insert');
-- Обновим все строки.
FORALL v_Count IN 1..1000
   UPDATE temp_table
      SET char_col = 'Changed!'
      WHERE num_col = v_Numbers(v_Count);
- Несмотря на то, что имеется только 1000 элементов, этот оператор
- обновляет 1500 строк, так как предложение WHERE соответствует
- 2 строкам для каждой из последних 500 строк.
DBMS_OUTPUT.PUT_LINE(
```

Массовые соединения. Пример (продолжение).

```
-- Аналогично, этот DELETE удалит 300 строк.
FORALL V_Count IN 401..600
      DELETE FROM tempjtable
      WHERE nun_col = v_Numbers(v_Count);
-- Поэтому должно остаться 1200 строк.
PrintTotalRows('After delete');
END;
```

Результатом выполнения примера будет следующее:

```
After first insert: Count is 1000
After second insert: Count is 1500
Update processed 1500 rows.
After delete: Count is 1200
```

FORALL синтаксически аналогичен циклу FOR. Он может использоваться для сборных конструкций любого типа и для операторов INSERT, DELETE и UPDATE. Определяемый в FORALL диапазон должен быть непрерывным, и все элементы в этом диапазоне должны существовать.

Особенности использования транзакций

Если в массовой операции DML при обработке одной из строк возникает ошибка, то откатывается только эта строка. Предыдущие строки будут обработаны.

В Oracle9i можно указать в операторе FORALL новую конструкцию SAVE EXCEPTIONS. При этом любая ошибка, возникшая во время пакетной обработки, будет сохранена, а обработка будет продолжена.

Для просмотра исключений можно использовать новый атрибут SQL%BULK_EXCEPTIONS, который действует как таблица PL/SQL.

DBMS_SQL

DBMS_SQL используется для выполнения динамического SQL в PL/SQL. Он не встроен непосредственно в язык и поэтому менее эффективен, чем встроенный динамический SQL (который доступен в Oracle8г и выше).

Модуль DBMS_SQL позволяет непосредственно управлять обработкой операторов в курсоре, выполнять синтаксический разбор оператора, связывать входные переменные и определять выходные переменные.

DBMS_SQL. Пример.

```
CREATE OR REPLACE PROCEDURE UpdateClasses(
/* Использует DBMS_SQL для обновления таблицы учебных групп,
задания
 числа зачетов для всех групп на указанном факультете.
*/
p_Department IN classes.department%TYPE,
p_NewCredits IN classes. num_credits%TYPE,
p_RowsUpdated OUT INTEGER) AS
v_CursorID INTEGER;
v_UpdateStmt VARCHAR2(100);

BEGIN
- Откроем курсор для обработки.
v_CursorID := DBMS_SQL.OPEN_CURSOR;
- Определим строку SQL.
v_UpdateStmt :=
      'UPDATE classes
            SET num_credits = :nc
            WHERE department = :dept';
```

DBMS_SQL. Пример. (продолжение)

```
  -- Выполним синтаксический разбор оператора.
  DBMS_SQL.PARSE(v_CursorID, v_UpdateStrat, DBMS_SQL.NATIVE);
  -- Свяжем p_NewCredits с позицией :nc. Эта перегруженная версия
  -- BIND_VARIABLE привяжет p_NewCredits как NUMBER,
  -- поскольку он так объявлен.
  DBMS_SQL.BIND_VARIABLE(v_CursorID, ':nc', p_NewCredits);
  -- Свяжем p_Department с позицией :dept. Эта перегруженная
версия
  -- BIND_VARIABLE привяжет p_Department как CHAR, поскольку он
  -- так объявлен.
  DBMS_SQL.BIND_VARIABLE_CHAR(v_CursorID, ':dept ', p_Department);
  -- Выполним оператор
    p_RowsUpdated := DBMS_SQL.EXECUTE(v_CursorID);
  -- Закроем курсор
  DBMS_SQL.CLOSE_CURSOR(v_CursorID);
EXCEPTION
     WHEN OTHERS THEN
       -- Закроем курсор и снова инициируем ошибку.
     DBMS_SQL.CLOSE_CURSOR(v_CursorID);
     RAISE;
END UpdateClasses;
```

Задания

1. Напишите процедуру, возвращающую список товаров, количество каждого товара и его цену. Процедура должна иметь два входных параметра - название фирмы и название товара. Если указано название фирмы, выдаётся список всех купленных ею товаров. Если указано название фирмы и дополнительно указано наименование товара, то выдаётся товар, купленный фирмой, наименование которого совпадает с заданным наименованием. Если указано только наименование товара, то выдаётся указанный товар.

   Процедуру написать с использованием позиций предназначенных для размещения переменных PL/SQL, указываемых в предложении USING оператора EXECUTE IMMEDIATE.
2. Выполните первое задание используя DBMS_SQL.

Задания

3. Модифицируйте задание 1 так,чтобы в том случае, если выдаётся один товар, то он бы записывался 200 раз в таблицу test_item. Таблица test_item содержит два поля:

- name – наименование товара;
- num – порядковый номер в блоке (т.е. оно от 1 до 200)

Процедуру написать с использованием FORALL.