Динамический SQL

(использование в Oracle)

Виды предложений SQL

Метод	Тип предложения	Требуемые вызовы пакета DBMS_SQL
1	Незапросные, нет базовых переменных, выполняется однократно	открытие курсора (open cursor), разборка (parse), выполнение (execute), закрытие курсора (close cursor)
2	Незапросные, известное число базовых переменных, выполняются один или несколько раз	открытие курсора, разборка, связывание переменных (bind variables), выполнение, закрытие курсора
3	Запросные, известное число операторов SELECT и базовых переменных	открытие курсора, разборка, связывание переменных, определение колонок (define columns), выполнение, выборка строк (fetch rows), получение значений колонок (get column values), обновление (refretch), ... закрытие курсора
4	Запросные, неизвестное число операторов SELECT и базовых переменных колонок	открытие курсора, разборка, связывание переменных, определение колонок, выполнение, выборка строк, получение значений колонок, обновление, ... закрытие курсора

Пакет DBMS_SQL. Основные функции и процедуры

- **OPEN_CURSOR** RETURN INTEGER

  Открывает новый курсор. По окончании работы необходимо закрыть открытый курсор, используя функцию CLOSE_CURSOR.

- **PARSE**

  (cursor IN INTEGER,
  statement IN VARCHAR2,
  language IN INTEGER)

  Немедленная разборка предложения в курсоре. Если разбираемое предложение является DDL-предложением, процедура PARSE также выполняет это DDL-предложение.

- **EXECUTE**

  (cursor IN INTEGER) RETURN INTEGER

  Выполняет предложение, находящееся в курсоре, и возвращает число обработанных в процессе выполнения строк.

- **CLOSE_CURSOR**

  (cursor IN OUT INTEGER)

  Закрывает открытый курсор.

Пакет DBMS_SQL. Основные функции и процедуры

- **BIND_VARIABLE**
  (cursor IN INTEGER,
  variable IN VARCHAR2,
  value IN NUMBER | VARCHAR2 | DATE | MLSLABEL
  [, size IN INTEGER])

  Связывает значение с переменной в предложении, разбираемом в курсоре. Когда переменная является входной или входной/выходной, связанное значение должно быть правильно определено. Если переменная – выходная, вызов игнорирует связанное значение.
- **DEFINE_COLUMN**
  (cursor IN INTEGER,
  possition IN INTEGER,
  value IN NUMBER | VARCHAR2 | DATE | MLSLABEL
  [, size IN INTEGER])

  Определяет колонку, указанную в курсоре. Эта процедура используется только в SELECT-курсорах.

Пакет DBMS_SQL. Основные функции и процедуры

- **FETCH_ROWS**
  (cursor IN INTEGER) RETURN INTEGER
  Извлекает очередную строку из курсора. После извлечения строки в локальный буфер (с помощью FETCH_ROWS) программа должна вызвать процедуру COLUMN_VALUE, чтобы прочитать извлеченную строку. Программа может неоднократно использовать функцию FETCH_ROWS, чтобы извлекать строки из курсора, до тех пор пока не будут исчерпаны все строки.
- **COLUMN_VALUE**
  (cursor IN INTEGER,
  possition IN INTEGER,
  value OUT NUMBER | VARCHAR2 | DATE | MLSLABEL
  [, error OUT NUMBER
  , length OUT INTEGER])
  Получает значение колонки. Эта процедура применяется для доступа к данным, предварительно выбранным посредством вызова функции FETCH_ROWS.

Пакет DBMS_SQL. Вспомогательные функции

- **IS_OPEN** (cursor IN INTEGER) RETURN BOOLEAN
  Проверяет, открыт ли курсор.
- **LAST_ERROR_POSITION** RETURN INTEGER
  Если имеет место ошибка во время выполнения операции, находящейся в курсоре, функция возвращает относительную позицию колонки в курсорном предложении, которая послужила причиной ошибки.
- **LAST_ROW_COUNT** RETURN INTEGER
  Функция возвращает суммарное количество строк, извлеченных до сих пор из курсора.
- **LAST_ROW_ID** RETURN ROWID
  Функция возвращает значение ROWID последней строки, обработанной в курсоре.
- **LAST_SQL_FUNCTION_COD**E RETURN INTEGER
  Функция возвращает код функции SQL-предложения.

Пример 1. Процедура создания таблицы с помощью пакета dbms_sql (метод 1)

```
CREATE PROCEDURE create_temp_dept(tname IN OUT VARCHAR2) AS
cur INTEGER;     -- хранит идентификатор (ID) курсора
ret INTEGER;     -- хранит возвращаемое по вызову значение
str VARCHAR2(250);   -- хранит команды
BEGIN
   /* генерация временной таблицы по имени DEPT, используя заранее
    * заданное (hard-coded) имя и возврат значения функции
    * DBMS_SESSION.UNIQUE_SESSION_ID */
   tname := 'dept' || dbms_session.unique_session_id;
   -- генерация команды CREATE TABLE по заранее заданному тексту
   -- и переменной tname
   str := 'CREATE TABLE '||tname
         || ' (deptno INTEGER,'
         || ' dname VARCHAR2(14),'
         || ' loc VARCHAR2(13), '
         || 'TABLESPACE temp '
         || 'STORAGE ('|| 'INITIAL 10K NEXT 10K MAXEXTENTS 2 )';
   -- Динамически формируемое DDL SQL-предложение по методу 1
   cur := dbms_sql.open_cursor;
   dbms_sql.parse(cur, str, dbms_sql.v7);
   ret := dbms_sql.execute(cur);
   dbms_sql.close_cursor(cur);
END;
```

Пример 2. Использование пакета dbms_sql (методы 1,2,3)

Перечень действий примера:

1. **Создать таблицу demo_tbl. (Незапросное предложение, DDL.)**
   Структура таблицы следующая:
   ID NUMBER(3),
   NAME VARCHAR2(20)
2. **Вставить в неё 5 строк с ID 1..5. (Незапросное предложение, DML.)**
3. **Удалить строку с номером 4. (Незапросное предложение, DML.)**
4. **Считать и распечатать строки из таблицы с ID=2,3,5.**
   (Запросное предложение.)
5. **Удалить таблицу demo_tbl. (Незапросное предложение, DDL.)**

Текст примера 2 (использование пакета dbms_sql)

```
DECLARE
  -- курсор
  vCursor NUMBER;
  -- оператор для создания таблицы
  vCreateTable VARCHAR2(200) := 'CREATE TABLE demo_tbl (ID NUMBER(3),
                                NAME VARCHAR2(50))';
  -- оператор для удаления таблицы
  vDropTable VARCHAR2(200) := 'DROP TABLE demo_tbl';
  -- оператор для вставки в таблицу
  vInsertTable VARCHAR2(200) := 'INSERT INTO demo_tbl(ID, NAME)
                                VALUES(:id, :name)';
  -- оператор для удаления строки из таблицы
  vDeleteTable VARCHAR2(200) := 'DELETE FROM demo_tbl WHERE ID=:id';
  -- оператор для считывания строк из таблицы
  vSelectTable VARCHAR2(200) := 'SELECT ID, NAME FROM demo_tbl
                                WHERE ID BETWEEN :id_l AND :id_h';
  vResult INTEGER;
  -- выходные переменные для оператора SELECT
  vId NUMBER(3);
  vName VARCHAR2(50);
```

Продолжение текста примера 2 (использование пакета dbms_sql)

```
BEGIN
  vCursor := DBMS_SQL.OPEN_CURSOR; /* создание таблицы */
  DBMS_SQL.PARSE(vCursor, vCreateTable, DBMS_SQL.V7);
  /* добавление данных в таблицу */
  DBMS_SQL.PARSE(vCursor, vInsertTable, DBMS_SQL.V7);
  FOR vId IN 1..5 LOOP
      -- установка значений для переменных привязки
      DBMS_SQL.BIND_VARIABLE(vCursor, ':id', vId);
      DBMS_SQL.BIND_VARIABLE(vCursor, ':name', 'Name '||vId);
      -- выполнение оператора Insert
      vResult:=DBMS_SQL.EXECUTE(vCursor);
  END LOOP;
  COMMIT; -- фиксация изменений
  /* удаление строки с ID=4 */
  DBMS_SQL.PARSE(vCursor, vDeleteTable, DBMS_SQL.V7);
  -- установка значения для переменной привязки
  DBMS_SQL.BIND_VARIABLE(vCursor, ':id', 4);
  -- выполнение оператора Insert
  vResult:=DBMS_SQL.EXECUTE(vCursor);
  COMMIT; -- фиксация изменений
```

Завершение текста примера 2 (использование пакета dbms_sql)

```
  /* считывание строк с ID=2,3,4 */
  DBMS_SQL.PARSE(vCursor, vSelectTable, DBMS_SQL.V7);
  -- установка значений для переменных привязки
  DBMS_SQL.BIND_VARIABLE(vCursor, ':id_l', 2);
  DBMS_SQL.BIND_VARIABLE(vCursor, ':id_h', 5);
  -- определение выходных переменных
  DBMS_SQL.DEFINE_COLUMN(vCursor, 1, vId);
  DBMS_SQL.DEFINE_COLUMN(vCursor, 2, vName, 50);
  -- выполнение оператора Select
  vResult:=DBMS_SQL.EXECUTE(vCursor);
  LOOP
      EXIT WHEN DBMS_SQL.FETCH_ROWS(vCursor) = 0;
      -- считывание строки из буфера в переменные PL/SQL
      DBMS_SQL.COLUMN_VALUE(vCursor, 1, vId);
      DBMS_SQL.COLUMN_VALUE(vCursor, 2, vName);
      -- распечатка полученных данных
      DBMS_OUTPUT.PUT_LINE(vId||' '||vName);
  END LOOP;
  /* удаление таблицы */
  DBMS_SQL.PARSE(vCursor, vDropTable, DBMS_SQL.V7);
  DBMS_SQL.CLOSE_CURSOR(vCursor);
EXCEPTION WHEN OTHERS THEN DBMS_SQL.CLOSE_CURSOR(vCursor);
      RAISE;
END;
```

Встроенный SQL (начиная с версии 8i) (Native dynamic SQL, NDS)

При работе со встроенным динамическим SQL для выполнения предложений, не возвращающих множественные результаты, используется следующая конструкция:

EXECUTE IMMEDIATE *предложение_SQL*
[**INTO** { *переменная1* [, *переменная2*] ... | *запись* }]
[**USING** [IN | OUT | IN OUT] *связанный_аргумент1* [,
[IN | OUT | IN OUT] *связанный_аргумент2*] ...];

INTO – определяет приёмник результата.
USING – определяет типы и порядок аргументов, используемых для передачи значений в запрос и приёма данных результата. Замена аргументов на их значения происходит позиционно.

Ограничения на передаваемые аргументы:

- Нельзя передавать значения типа boolean, index-by таблицы, записи.
- Нельзя передавать NULL как литерал (только переменную со значением ***null*** или функцию, которая возвращает ***null***).

Примеры использования NDS

Пример 1. Удаление таблицы.

```
execute immediate 'drop table temp_table';
```

Пример 2. Изменение зарплаты сотрудникам определенного отдела.

```
execute immediate 'update EMP set salary=salary*:num where depno=:did'
   USING  v_num,  v_did;
```

Пример 3. Получение информации о количестве записей в таблице.

```
   …
   v_name  varchar2(40);
   v_cnt  number;
begin
   execute immediate 'select count(*) from ' || v_name   INTO  v_cnt;
```

Пример 4. Получение информации об определенном сотруднике.

```
   …
   v_emp  emp%ROWTYPE;
begin
   execute immediate 'select count(*) from ' || name
         into var1;
```

Специальные конструкции NDS

Специальные конструкции для работы с запросами, порождающими множественные результаты:

OPEN { *курсорная_переменная* | *:хост_переменная* }

FOR *предложение_SQL*
[**USING** *связанный_аргумент1* [, *связанный_аргумент2*] ...];

*Оператор **OPEN** исполняет запрос, позиционирует курсор на первую запись и устанавливает %ROWCONT в 0.*

FETCH { *курсорная_переменная* | *:хост_переменная* }

INTO { *переменная1* [, *переменная2*] ... | *запись* };

*Оператор **FETCH** извлекает очередную строку (запись) из курсора и увеличивает значение %ROWCONT на 1. Если строка считана, устанавливает %FOUND в TRUE; если курсор исчерпан, устанавливает %NOTFOUND в TRUE.*

CLOSE { *курсорная_переменная* | *:хост_переменная* };

*Оператор **CLOSE** закрывает открытый курсор.*

Примеры пакетного и встроенного динамического SQL

```
-- Решение с помощью пакета DBMS_SQL
CREATE OR REPLACE PROCEDURE   runddl (ddl_in  IN VARCHAR2)  IS
   cur INTEGER:= DBMS_SQL.OPEN_CURSOR;
   fdbk INTEGER;
BEGIN DBMS_SQL.PARSE (cur, ddl_in, DBMS_SQL.NATIVE);
     fdbk := DBMS_SQL.EXECUTE (cur);
     DBMS_SQL.CLOSE_CURSOR (cur);
  EXCEPTION WHEN OTHERS THEN
     DBMS_OUTPUT.PUT_LINE ( 'RunDDL Failure on ' || ddl_in);
     DBMS_OUTPUT.PUT_LINE (SQLERRM);
     DBMS_SQL.CLOSE_CURSOR (cur);
END;
/
-- Решение с помощью встроенного SQL
CREATE OR REPLACE PROCEDURE    runddl81 (ddl_in  IN VARCHAR2)
     AUTHID CURRENT_USER  IS
BEGIN
     EXECUTE IMMEDIATE ddl_in;
END;
/
```

Реализация примера 2 с помощью NDS

```
DECLARE
    -- оператор для создания таблицы
    vCreateTable VARCHAR2(200) := 'CREATE TABLE demo_tbl
                                  (ID NUMBER(3), NAME VARCHAR2(50))';
    -- оператор для удаления таблицы
    vDropTable VARCHAR2(200) := 'DROP TABLE demo_tbl';
    -- оператор для вставки в таблицу
    vInsertTable VARCHAR2(200) := 'INSERT INTO demo_tbl(ID, NAME)
                                  VALUES(:id, :name)';
    -- оператор для удаления строки из таблицы
    vDeleteTable VARCHAR2(200) := 'DELETE FROM demo_tbl WHERE ID=:id';
    -- оператор для считывания строк из таблицы
    vSelectTable VARCHAR2(200) := 'SELECT ID, NAME FROM demo_tbl
                                  WHERE ID BETWEEN :id_l AND :id_h';
    vResult INTEGER;
    -- выходные переменные для оператора SELECT
    vId NUMBER(3);
    vName VARCHAR2(50);
    -- объявление курсорной переменной
    TYPE ref_cur  IS  REF CURSOR;
    c  ref_cur;
```

Реализация на NDS примера 2 (продолжение)

```
BEGIN
    EXECUTE IMMEDIATE vCreateTable;   -- создаём таблицу
    FOR vId IN 1..5 LOOP              -- добавляем данные
       EXECUTE IMMEDIATE vInsertTable USING vId, 'Name ╪'||vId ;
    END LOOP;
    COMMIT;
    EXECUTE IMMEDIATE vDeleteTable USING 3;   -- удаляем строку с ID=3
    COMMIT;
    OPEN c FOR vSelectTable USING 2, 5;        -- считаем строки с ID=2,3,5
    LOOP  FETCH c INTO vId, vName;
        EXIT WHEN c%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE(vId||' '||vName);
    END LOOP;
    CLOSE c;
    -- если запрос возвращает не более одной строки, можно записать так:
    EXECUTE IMMEDIATE  vSelectTable  INTO  vId, vName  USING 3, 4;
    DBMS_OUTPUT.PUT_LINE('-------------------');
    DBMS_OUTPUT.PUT_LINE(vId||' '||vName);
    EXECUTE IMMEDIATE vDropTable;             -- удаляем таблицу
END;
```

Сравнение возможностей пакетного и встроенного динамического SQL

❖**Встроенный динамический SQL:**

✔ Работает со всеми без исключения типами данных Oracle, включая и типы объектов, заданные пользователем, и типы коллекции (переменные массивы, вложенные таблицы, индексированные таблицы).

✔ Позволяет извлекать множественные данные (серию строк) непосредственно в конструкцию PL/SQL.

✔ Допускает использование RETURNING только в единственном запросе.

❖**Пакетный динамический SQL:**

✔ Поддерживает “Метод 4” пакетного SQL, при котором во время компиляции не фиксируется число извлекаемых столбцов или число переменных привязки.

✔ Позволяет описывать столбцы динамического курсора так, чтобы те получали значения из столбцов индексированной (index-by) таблицы записей.

✔ Работать с SQL-предложениями длиной более 32К.

✔ Возвращать данные с помощью RETURNING в массив переменных.

✔ Повторно использовать динамические курсоры, что улучшает производительность.

✔ Выполняться на клиентской части приложения, например, в Oracle Developer.

✔ DBMS_SQL позволяет работать лишь с типами данных, совместимыми с Oracle7.

✔ В DBMS_SQL данные извлекаются построчно в отдельную запись.