Динамический SQL

- **пакет DBMS_SQL (Oracle7)**
- **внутренний динамический SQL, интегрированный с языком PL/SQL (Oracle8i).**

2. Использование DBMS_SQL

2.1. Обработка операторов DDL

1. **Открытие курсора (OPEN_CURSOR)**
2. **Грамматический разбор оператора (PARSE)**
3. **Закрытие курсора (CLOSE_CURSOR)**

```
CREATE OR REPLACE PROCEDURE Create_table  (
   p_Description IN VARCHAR2)   IS
   v_Cursor        NUMBER;
   v_CreateString VARCHAR2(100);
BEGIN
   v_Cursor := DBMS_SQL.OPEN_CURSOR;
   v_CreateString := 'CREATE TABLE  MyTab ' || p_Description;
   DBMS_SQL.PARSE(v_Cursor, v_CreateString, DBMS_SQL.NATIVE);
   DBMS_SQL.CLOSE_CURSOR(v_Cursor);
EXCEPTION
   WHEN  OTHERS  THEN
   DBMS_SQL. CLOSE_CURSOR(v_Cursor);
   RAISE;
END Create_table;
```

2.2. Обработка операторов Alter session и операторов управления транзакциями

1. **Открытие курсора (OPEN_CURSOR)**
2. **Грамматический разбор оператора (PARSE)**
3. **Обработка оператора (EXECUTE)**
4. **Закрытие курсора (CLOSE_CURSOR)**

2.3. Обработка операторов DML

1. **Открытие курсора (OPEN_CURSOR)**
2. **Грамматический разбор оператора (PARSE)**
3. **Привязка всех входных переменных (BIND_VARIABLE)**
4. **Обработка оператора (EXECUTE)**
5. **Закрытие курсора (CLOSE_CURSOR)**

2.4. Обработка запросов

1. **Открытие курсора (OPEN_CURSOR)**
2. **Грамматический разбор оператора (PARSE)**
3. **Привязка всех входных переменных (BIND_VARIABLE)**
4. **Описание элементов списка выбора (DEFINE_COLUMN)**
5. **Обработка оператора (EXECUTE)**
6. **Считывание строки (FETCH_ROWS)**
7. **Запись результата в переменные PL/SQL (COLUMN_VALUE)**
8. **Закрытие курсора (CLOSE_CURSOR)**

```
CREATE OR REPLACE PROCEDURE DynaQuery (
      p_City IN VARCHAR2)   IS
      v_Cursor         NUMBER;
      v_SelectStm     VARCHAR2(100);
      v_Name           Sal.sname%TYPE;
      v_Dummy         NUMBER;
   BEGIN
      v_Cursor := DBMS_SQL.OPEN_CURSOR;
      v_SelectStm := 'SELECT  sname FROM sal WHERE city=:s
                        ORDER BY sname';
      DBMS_SQL.PARSE(v_Cursor, v_SelectStm, DBMS_SQL.NATIVE);
      DBMS_SQL.BIND_VARIABLE(v_Cursor, ':s', p_City);
      DBMS_SQL.DEFINE_COLUMN(v_Cursor, 1, v_Name, 10);
      v_Dummy := DBMS_SQL.EXECUTE(v_Cursor);
      LOOP
         IF DBMS_SQL.FETCH_ROWS(v_Cursor)=0  THEN   EXIT;  END IF;
         DBMS_SQL.COLUMN_VALUE(v_Cursor, 1, v_Name);
         INSERT INTO Temp_tab (char_col) VALUES(v_Name);
      END LOOP;
      DBMS_SQL.CLOSE_CURSOR(v_Cursor);
      COMMIT;
   EXCEPTION
      WHEN  OTHERS  THEN
      DBMS_SQL. CLOSE_CURSOR(v_Cursor);
      RAISE;
   END DynaQuery;
```

2.5. Обработка анонимных блоков PL/SQL

1. **Открытие курсора (OPEN_CURSOR)**
2. **Грамматический разбор оператора (PARSE)**
3. **Привязка всех входных переменных (BIND_VARIABLE)**
4. **Обработка оператора (EXECUTE)**
5. **Считывание значений выходных переменных (VARIABLE_VALUE)**
6. **Закрытие курсора (CLOSE_CURSOR)**

3. Использование внутреннего динамического SQL

- **EXECUTE IMMEDIATE**
- **OPEN FOR (для запросов, возвращающих более одной строки)**

3.1. Обработка операторов, не являющихся запросами, и для блоков PL/SQL

```
EXECUTE IMMEDIATE строка_символов_sql
   [INTO {перемменная[, переменная]…| запись}]
   [USING [IN | OUT | IN OUT] аргумент
         [,  [IN | OUT | IN OUT] аргумент ]…];
```

3.1.1. Операторы без переменных привязки

```
DECLARE
   v_SQLString     VARCHAR2(200);
   v_PLSQLBlock VARCHAR2(500);
BEGIN
   EXECUTE IMMEDIATE 'CREATE TABLE exe_tab (col1 VARCHAR2(10))';

   FOR  v_Cnt  IN  1..5  LOOP
      v_SQLString := 'INSERT INTO exe_tab VALUES (''Row '|| v_Cnt || ''')';
      EXECUTE IMMEDIATE  v_SQLString;
   END LOOP;

   v_PLSQLBlock :=
      'BEGIN
         FOR v_Rec IN (SELECT * FROM exe_tab) LOOP
            DBMS_OUTPUT.PUT_LINE(v_Rec.col1);
         END LOOP;
      END;';
   EXECUTE IMMEDIATE  v_PLSQLBlock;

   EXECUTE IMMEDIATE 'DROP TABLE exe_tab';
END;
```

3.1.2. Операторы с переменными привязки

```
DECLARE
      v_SQLString     VARCHAR2(200);
BEGIN
      v_SQLString :=
            'INSERT INTO cust(cnum,cname,city,rating,snum)
              VALUES(:cn, :cnm, :ct, :rt, :sn)';
      EXECUTE IMMEDIATE v_SQLString
            USING  2020,  'Tom',  'Berlin', 100, 1002;
END;
```

3.2. Обработка запросов

3.2.1. Многострочные запросы

OPEN *курсорная_переменная* FOR *строка_символов_sql*
[USING *аргумент* [, *аргумент*]…];

```
DECLARE
    TYPE t_SalCur IS REF CURSOR;
    v_SalCur    t_SalCur;
    v_Sal       sal%ROWTYPE;
    v_SQLSring  VARCHAR2(200);
BEGIN
    v_SQLSring := 'SELECT * FROM sal WHERE city=''London''';
    OPEN v_SalCur FOR v_SQLSring;
    LOOP
        FETCH v_SalCUR INTO v_Sal;
        EXIT WHEN v_SalCur%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Fetched sname is ' || v_Sal.sname);
    END LOOP;
    CLOSE v_SalCur;
END;
/
```

3.2.2. Однострочные запросы

EXECUTE IMMEDIATE *строка_символов_sql*
INTO *список_определения*
[USING *список_привязки*];

```
DECLARE
   v_SQLQuery VARCHAR2(200);
   v_Sal      sal%ROWTYPE;
   v_City     VARCHAR2(10);
   v_Cnt      NUMBER;
BEGIN
   v_City := 'London';
   v_SQLQuery :=
       'SELECT count(*) FROM sal WHERE city =''' || v_City || '''';

   EXECUTE IMMEDIATE v_SQLQuery INTO v_Cnt;

   DBMS_OUTPUT.PUT_LINE('Fetched count = ' || v_Cnt);
END;
/
```

3.3. Передача значений параметров

3.3.1. Подстановка и конкатенация

```
EXECUTE IMMEDIATE
      'UPDATE ' || tab || 'SET sal = :new_sal '
USING v_sal;

EXECUTE IMMEDIATE
'UPDATE ' || tab || 'SET sal = ' || v_sal;
```

3.3.2. Ограничения на подстановку

- **В качестве параметров можно использовать только значения данных (литералы, переменные, выражения)**
- **Нельзя использовать имена объектов БД (таблиц, столбцов и т.п.)**

3.3.3. Режимы использования параметров

- **В режиме IN – любой элемент: литерал, константа, переменная или выражение**
- **В режимах OUT и IN OUT – использовать переменную**
- **Тип данных подставляемых значений должен поддерживаться SQL (например, нельзя BOOLEAN)**

3.3.4. Дублирование формальных параметров

- **В динамическом SQL (DML- и DDL-строки, не оканчивающиеся точкой с запятой) подстановка задается для каждого формального параметра, *с учетом их повторений***
- **В динамическом блоке PL/SQL (строки, заканчивающиеся точкой с запятой) параметры подстановки задаются только для каждого *уникального* формального параметра**

```
BEGIN
   EXECUTE IMMEDIATE
      'UPDATE emp SET col_1 = :val
      WHERE hiredate BETWEEN :ndate AND :kdate
      AND :val IS NOT NULL'
   USING val_in, start_in, end_in, val_in;
END;
```

```
BEGIN
   EXECUTE IMMEDIATE
      'BEGIN
         UPDATE emp SET col_1 = :val
         WHERE hiredate BETWEEN :ndate AND :kdate
         AND :val IS NOT NULL
      END;'
   USING val_in, start_in, end_in;
END;
```

3.3.5. Передача значения NULL

Нельзя написать:

```
   EXECUTE IMMEDIATE
      'UPDATE emp SET salary = :val
      WHERE hiredate IS NOT NULL'
   USING NULL;
```

- **Скрыть в переменной**
- **Использовать функцию преобразования типов**

```
DECLARE
   val_in NUMBER;
BEGIN
   EXECUTE IMMEDIATE
      'UPDATE emp SET salary = :val
      WHERE hiredate IS NOT NULL'
   USING val_in;
END;

BEGIN
   EXECUTE IMMEDIATE
      'UPDATE emp SET salary = :val
      WHERE hiredate IS NOT NULL'
   USING TO_NUMBER(NULL);
END;
```

3.3.6. Программы с правами вызывающего

```
CREATE OR REPLACE PROCEDURE execDDL (ddl_str IN VARCHAR2)
IS
BEGIN
    EXECUTE IMMEDIATE ddl_str;
END;


SQL> exec STUDY. ExecDDL('create table tmp (dt date)');


CREATE OR REPLACE PROCEDURE execDDL (ddl_str IN VARCHAR2)
    AUTHID CURRENT_USER  IS
BEGIN
    EXECUTE IMMEDIATE ddl_str;
END;
```

Сравнение возможностей NDS и пакета DBMS_SQL

Встроенный динамический SQL (NDS)

- **эффективней, чем DBMS_SQL**
- **проще в применении, программный код короче**
- **работает со всеми типами данных SQL, включая пользовательские объекты и коллекции (DBMS_SQL – только типы данных, совместимые с Oracle7)**
- **позволяет выбирать данные из нескольких столбцов прямо в запись, а не только в отдельные переменные**

Пакет DBMS_SQL следует использовать, если

- **во время компиляции не известно количество запрошенных столбцов и переменных привязки**
- **необходимо выполнить код SQL и PL/SQL произвольной длины (для NDS – не более 32 Кбайт)**
- **динамический SQL требуется вызывать из клиентского кода PL/SQL, например из библиотек Oracle Forms**