Курс «Базы данных»

Тема. Программирование на языке PL/SQL. Часть 2

Барабанщиков
Игорь Витальевич

План лекции

1. Использование SQL в PL/SQL
2. Неявные курсоры
3. Управляющие конструкции

Использование SQL в PL/SQL

- **В процедурах PL/SQL можно использовать команды SQL.**
- **Выборка строки данных из БД** выполняется командой **SELECT**. Должна возвращаться только одна строка.
- **Внесение изменений** в строки таблиц БД выполняется командами **DML (Insert, Update, Delete)**.
- **Управление транзакциями** выполняется командами **COMMIT, ROLLBACK, SAVEPOINT**.

Команды SQL в PL/SQL

- **Ключевое слово END означает конец блока**, а не завершение транзакции.
- Один **блок может содержать несколько транзакций**.
- Одна транзакция может охватывать несколько блоков.
- **PL/SQL не поддерживает прямо команды DDL** (create table…, alter table…).
- **PL/SQL не поддерживает прямо команды DCL** (grant, revoke)

Команда SELECT в PL/SQL

- Значения, выбираемые по команде SELECT, **должны запоминаться в переменных**, указанных во фразе **INTO**.
- **Предложение INTO обязательно**.
- Запрос должен возвращать **одну и только одну строку**.
- Отсутствие возвращаемых строк или возврат нескольких строк приводят к ошибке.
- Количество и типы переменных во фразе **INTO** должно совпадать с количеством и типами столбцов во фразе SELECT.

Пример команды SELECT

```
DECLARE
  v_name  VARCHAR2(20);
  v_date    employees.hiredate%TYPE;
BEGIN
   -- выборка данных из таблицы в переменную
   SELECT  last_name,  hiredate
       INTO  v_name,  v_date
       FROM  employees
       WHERE emp_id  = 100;
    -- вывод значения переменной
   dbms_output.put_line('Имя: '||v_name);
END;
```

Манипулирование данными

```
DECLARE
   v_sal     employees.salary%TYPE;
BEGIN
   SELECT salary INTO v_sal
     FROM emploees  WHERE emp_id = 105;
   v_sal := v_sal + 1000;
   UPDATE  employees  SET  salary = v_sal
        WHERE  emp_id = 299;
END;
```

Обработка команд SQL

- Сервер БД выделяет для обработки команд SQL приватную область памяти, называемую ***контекстной областью***.
- В этой области команда SQL синтаксически разбирается и выполняется.
- В этой области сохраняются результаты обработки.
- Пользователь не имеет доступа к этой области памяти, ей управляет сервер

Явный и неявный курсоры

- **Курсор** – это указатель на контекстную область.
- Когда в блоке выполняется команда SQL, то создается **неявный курсор**.
- Управление неявным курсором *автоматически* выполняет сервер Oracle.
- Если требуется выбрать из БД несколько строк и обработать их последовательно, то программист создает **явный курсор**.

Атрибуты курсора

SQL%FOUND	**Возвращает значение «ИСТИННО» (TRUE), если последняя команда SQL обработала одну или несколько строк.**
SQL%NOTFOUND	Возвращает значение «ИСТИННО» (TRUE), если последняя команда SQL не обработала ни одной строки.
SQL%ROWCOUNT	Возвращает количество строк (число), обработанных последней командой SQL.

Атрибуты курсора

- **Атрибуты курсора позволяют выяснить что произошло при выполнении последней команды SQL.**
- Атрибуты курсора можно применять в командах PL/SQL подобно функциям.
- Использовать атрибуты курсора в командах SQL нельзя.
- Во всех атрибутах присутствует префикс **SQL%**. Это указывает на *неявный курсор*.

Пример

```
DECLARE
  v_count  BINARY_INTEGER;
BEGIN
  UPDATE employees SET salary=salary+1000
      WHERE deptno = 10;
  v_count := SQL%ROWCOUNT;
  dbms_output.put_line('обновлено'||v_count);
END;
```

Функции SQL в PL/SQL

Большинство встроенных функций SQL доступно и в выражениях PL/SQL:

- Однострочные числовые функции
- Однострочные символьные функции
- Функции преобразования типов данных
- Функции для работы с датами

Недопустимые функции:

- DECODE
- Групповые (AVG, MIN, MAX, SUM, COUNT)

Примеры использования функций

- Определение длины строки

  v_name := ‘Иванов’;

  v_size := **LENGTH**(v_name);
- Преобразование в верхний регистр

  v_uname := **UPPER**(v_name);
- Преобразование типа данных

  v_date := **TO_CHAR**(**SYSDATE**, ‘dd.mm.yyyy’);

Управление потоком выполнения

Изменить поток выполнения в блоке PL/SQL можно с помощью набора управляющих структур:

- Оператор условного управления **IF**
- Оператор **CASE**
- Команды цикла **LOOP**

Команда условного управления IF

```
IF  условие THEN
    команды;
[ ELSIF условие THEN  -- необязательная часть
    команды; ]
[ ELSIF условие THEN  -- необязательная часть
    команды; ]
[ ELSE                -- необязательная часть
    команды; ]
```

Примеры IF

Пример 1:

```
DECLARE
  v_num number(5) := 10;
BEGIN
  IF v_num > 5 THEN
      v_num := v_num – 1;
   END IF;
END;
```

Пример 2:

```
DECLARE
  v_num number(5) := 0;
BEGIN
  IF v_num = 0 THEN
      v_num := v_num + 1;
   ELSE
      v_num := 100 / v_num;
   END IF;
END;
```

Пример IF…ELSIF…ELSE

```
DECLARE
  v_age  NUMBER(3) := 50;
BEGIN
  IF v_age < 12 THEN
      dbms_output.put_line('Ребенок');
  ELSIF v_age <= 18 THEN
      dbms_output.put_line('Подросток');
  ELSIF v_age < 60 THEN
      dbms_output.put_line('Взрослый');
  ELSE
      dbms_output.put_line('Пенсионер');
   END IF;
END;
```

Простой оператор CASE

Выражение-переключатель сравнивается со значением в предложении WHEN.

```
CASE выражение-переключатель
  WHEN значение1 THEN команда1;
  WHEN значение2 THEN команда2;
  WHEN значение3 THEN команда3;
  [ ELSE команда4; ]
END CASE;
```

Пример простого CASE

```
CASE  v_znak
   WHEN  ‘+’  THEN  v_rezult := v_arg1 + v_arg2;
   WHEN  ‘-’  THEN  v_rezult := v_arg1 – v_arg2;
   WHEN  ‘/’  THEN  v_rezult := v_arg1 / v_arg2;
   WHEN  ‘*’  THEN  v_rezult := v_arg1 * v_arg2;
   ELSE  v_rezult := 0;
END  CASE;
```

Поисковый оператор CASE

Не имеет переключателя. Вместо этого каждое предложение WHEN содержит логическое выражение.

```
CASE
  WHEN выражение1 THEN команда1;
  WHEN выражение2 THEN команда2;
  WHEN выражение3 THEN команда3;
  [ ELSE команда4;]
END CASE;
```

Пример поискового CASE

```
CASE
  WHEN  region_id = ‘Урал’
            THEN mgr := ‘Иванов’;
  WHEN  division_id = ‘Бухгалтерия’
            THEN mgr := ‘Петров’;
  ELSE
            mgr := ‘Сидоров’;
END CASE;
```

Операторы цикла в PL/SQL

PL/SQL предоставляет следующие типы циклов:

- **Простой цикл** – повторное выполнение действий без каких-либо условий.
- **Цикл WHILE** – повторное выполнение действий на основе условия.
- **Цикл FOR** – выполнение действий определенное количество раз.

Простой цикл

Это группа повторно выполняемых команд, ограниченных словами **LOOP … END LOOP**.

Если отсутствует команда EXIT, то цикл будет выполняться бесконечно.

```
LOOP
   команда1;
   . . .
   EXIT  [ WHEN  выражение ];
END LOOP;
```

Пример простого цикла

```
DECLARE
   v_count  BINARY_INTEGER  := 0;
BEGIN
   LOOP
      v_count  :=  v_count  +  1;
      dbms_output.put_line('номер='||v_count);
      EXIT  WHEN v_count > 5;
   END  LOOP;
END;
```

Цикл WHILE

Используется для повторного выполнения последовательности команд, пока **ИСТИННО** заданное условие.

```
WHILE условие LOOP
    команда1;
    команда2;
    . . .
END LOOP;
```

Пример цикла WHILE

```
DECLARE
   v_count  BINARY_INTEGER  := 0;
BEGIN
   WHILE  v_count <= 5  LOOP
      v_count  :=  v_count  +  1;
      dbms_output.put_line('номер='||v_count);
   END  LOOP;
END;
```

Цикл FOR

- **Объявление индекса** (переменной цикла) **не требуется**, он объявляется неявно.
- **Обязательно надо задать нижнюю и верхнюю границы**.

```
FOR индекс IN [ REVERSE ] ниж_гр .. верх_гр
  LOOP
     команда1
     команда2;
     . . .
  END LOOP;
```

Пример цикла FOR

```
DECLARE
   /*  переменную цикла объявлять
       не надо!!!  */
BEGIN
   FOR  v_count  IN  1..5  LOOP
      dbms_output.put_line('номер='||v_count);
   END  LOOP;
END;
```

Итоги

Были изучены вопросы:

- Использование команд SQL в PL/SQL
- Использование функций SQL в PL/SQL
- Неявные курсоры
- Операторы управления потоком выполнения.
- Операторы цикла.