

#####

## ЛАБОРАТОРНАЯ РАБОТА N2

ТЕМА: Язык PL/SQL. Курсоры. Последовательности.

Используемые объекты:

1. Учебная база данных - таблицы SAL, CUST, ORD
2. Описание учебной базы данных - таблица OPDB

Переменные в PL/SQL предназначены для размещения одного экземпляра элементов данных. С этим связана проблема, возникающая, когда командой SELECT ... INTO ... из базы данных выбирается не одна, а несколько строк. Такая ситуация обрабатывается с использованием исключения TOO\_MANY\_ROWS.

Если из БД нужно выбрать несколько строк и обрабатывать их по отдельности, можно использовать курсор. Курсор это набор записей, выбранных из таблиц БД с помощью команды SELECT, связанной с курсором. К этому набору записей можно обращаться для считывания из него данных в программные переменные.

Курсор определяется в разделе DECLARE, где задается его имя и связанная с ним команда SELECT. Например:

```
...  
DECLARE  
    CURSOR cur1 IS  
        SELECT sname, comm FROM sal WHERE city <> 'London';
```

После определения курсора, для фактического выполнения запроса и извлечения требуемых строк нужно его открыть в основном блоке:

```
...  
    OPEN cur1;
```

В PL/SQL имеется системная переменная %ISOPEN (атрибут курсора), которую полезно использовать для определения того, был ли курсор ранее открыт. В приведенном ниже фрагменте кода курсор открывается, если он не был открыт ранее:

```
...  
    if not cur1%ISOPEN then  
        OPEN cur1;  
    end if;
```

В результате открытия курсора формируется набор записей, но требуемые нам данные недоступны, т.к. они еще не перенесены в программные переменные. Для чтения данных из курсора и пересылки их в переменные используется команда FETCH. Эта команда читает одну, очередную строку из набора и переносит данные из этой строки в указанные переменные. В следующем примере показано, как имя и комиссионные из очередной строки помещаются в переменные vname и vcomm соответственно:

```
...  
    FETCH cur1 INTO vname, vcomm;  
...
```

После обработки данных, полученных из первой строки, переслать в программные переменные данные следующей строки можно, выполнив снова команду FETCH.

В PL/SQL существуют атрибуты курсора, которые позволяют управлять работой с ним:

- %FOUND,
- %NOTFOUND,
- %ROWCOUNT.

Первые два дают возможность определить, прочитана ли (%FOUND) или не прочитана (%NOTFOUND) строка из курсора. Использовать эти переменные удобно при обработке строк курсора в цикле: их можно включать в выражение для проверки выхода из цикла. Для цикла LOOP такую проверку нужно делать обязательно, иначе цикл может быть бесконечным. В следующем примере работа с курсором прекращается, когда очередная строка не найдена, т.е. строки окончились:

```
...
LOOP
  FETCH cur1 INTO vname, vcomm;
  IF cur1%NOTFOUND THEN
    exit;
  END IF;
  INSERT INTO tab VALUES (vname, vcomm*1.1);
END LOOP;
```

Переменная %ROWCOUNT накапливает количество строк, прочитанных командой FETCH. Ее также можно использовать для выхода из цикла, когда, например, нужно обработать только заданное число строк курсора.

### ЗАДАНИЕ 1.

Напишите сценарий для вывода имен и комиссионных первых двух продавцов, работающих не в Лондоне.

В тех случаях, когда нет необходимости вручную работать с курсором, можно использовать специальный курсорный цикл FOR. Вот пример, как, используя такой цикл, можно считать и вывести данные о заказах и их суммах:

```
...
FOR v_z IN c_zak LOOP
  DBMS_OUTPUT.PUT_LINE('заказ № ||v_z.num||' на сумму '||v_z.sm);
END LOOP;
```

При этом курсор c\_zak определяется обычным образом, а переменную v\_z нашего курсорного цикла предварительно определять не нужно. В данном случае курсор открывается и закрывается автоматически, причем, если специально не прервать цикл, то он завершится, когда кончатся все записи курсора.

В курсорах можно использовать параметры, как при работе с подпрограммами или процедурами. Для этого в описании курсора нужно указать формальные параметры и их типы. Фактические параметры задаются при вызове курсора. Вот пример описания курсора с параметром

```

DECLARE
  cursor c_ord(p_Date VARCHAR2) is
    select onum,amt from ord
    where to_char(odate,'dd.mm.yy')=p_Date;

```

Здесь параметр используется для задания даты, и, таким образом, будет производиться выборка заказов за указанное число. Для выполнения этого в теле блока можно указать следующий код:

```

FOR v_ord IN c_ord('03.01.2010') LOOP
  DBMS_OUTPUT.PUT_LINE('Заказ '||v_ord.onum||' сумма '||v_ord.amt);
END LOOP;

```

## ЗАДАНИЕ 2.

Напишите сценарий, в котором при помощи курсора выбираются все заказы после 4-го числа, и эта дата задается в качестве параметра. Параметр курсора определите типа даты, а не строки.

Одна из интересных функций языка SQL Oracle - это DECODE. Она выполняется как условный оператор и имеет следующий синтаксис:

```

DECODE (exp, val1, trans1, val2, trans2, ... , default)

```

Здесь функция DECODE будет сравнивать выражение exp со значением val1 и, если они равны, то DECODE вернет значение trans1. В противном случае выражение exp будет сравниваться на равенство со значением val2, и при успешном сравнении будет возвращено trans2, и так далее. Если же ни одно из сравнений не выполнится, то функция вернет последнее из указанных значений - default.

Одно из полезных применений функции DECODE - это вывод данных из таблиц БД в виде матриц. В следующем примере показан способ вывода количества заказов за 3-е, 4-е и 10-е число с помощью одного запроса.

```

select
  count(decode(to_char(odate,'dd.mm.yyyy'),'03.01.2010',onum,null)) "3-е",
  count(decode(to_char(odate,'dd.mm.yyyy'),'04.01.2010',onum,null)) "4-е",
  count(decode(to_char(odate,'dd.mm.yyyy'),'10.01.2010',onum,null)) "10-е"
from ord;

```

## ЗАДАНИЕ 3.

Выполните приведенный выше запрос с функцией DECODE.  
 Напишите и выполните запрос для вывода в виде матрицы суммарных стоимостей заказов за каждое число для каждого продавца (для каждого числа – отдельный столбец, для каждого продавца - отдельная строка).

Последовательность Oracle - это объект базы данных, который может генерировать последовательный список чисел для числовых столбцов таблиц БД. Простейший способ создания последовательности представлен следующей командой:

```
CREATE SEQUENCE myseq;
```

В результате выполнения команды создается последовательность MYSEQ, которая по умолчанию будет генерировать ряд целых чисел, начиная с единицы до значения 10 в степени 27 с шагом 1.

Последовательность может быть использована многими пользователями (если им дать на это привилегии) пока не будет уничтожена командой DROP SEQUENCE.

Для использования последовательностей в командах SQL и PL/SQL применяют два псевдостолбца: sequence.NEXTVAL и sequence.CURRVAL. Sequence.NEXTVAL выдает очередное значение последовательности, sequence.CURRVAL показывает последнее сгенерированное число. Для демонстрации работы генератора последовательностей удобно использовать специальную системную таблицу SYS.DUAL. Например, следующие две команды покажут два сгенерированных кода, а третья – последний сгенерированный код:

```
...  
SELECT myseq.NEXTVAL FROM sys.dual;  
SELECT myseq.NEXTVAL FROM sys.dual;  
SELECT myseq.CURRVAL FROM sys.dual;
```

#### ЗАДАНИЕ 4.

Используя SQL Developer, создайте свою последовательность и свою таблицу. Вставьте в таблицу три строки, причем в командах вставки для занесения значений в одно из числовых полей таблицы используйте созданную последовательность.

Командой CREATE SEQUENCE и командой ALTER SEQUENCE можно задать значения параметров последовательности, отличные от значений по умолчанию. В следующем примере создается последовательность с начальным значением 5, шагом 3, максимальным значением 50:

```
...  
CREATE SEQUENCE seq INCREMENT BY 3  
  START WITH 5  
  MAXVALUE 50;
```

#### ЗАДАНИЕ 5.

Составьте сценарий, в котором создайте новую таблицу для занесения имен продавцов и минимальных стоимостей их заказов. В сценарии измените параметры последовательности из Задания 4 так, чтобы она формировала четные числа, начиная с 5000. Используя курсор, заполните созданную таблицу, причем, при вставке строк для формирования уникальных значений идентификаторов примените свою последовательность.

Сценарий должен заканчиваться выводом данных из  
заполненной таблицы. Выполните сценарий.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

=====

Контрольные вопросы выдает преподаватель после выполнения всех заданий.