# Лабораторная работа № 6

Введение в PL/SQL

|  |
| --- |
| Настройка SQL. Необходимо включить режим ECHO и вывести протокол лаб. работы в файл *<Фамилия студента>.txt*. Этот файл является отчетом о проделанной лаб. работе.  SQL> set echo on  SQL> spool c:\spool. txt  ……………………………!!! ОБРАТИТЕ ВНИМАНИЕ – косая черта в командной строке эквивалентна команде “run” !!!…………………………………..  Завершить протокол лаб. работы (команда spool off) и направить результаты преподавателю.  SQL> spool off |

**Задание 1:** Программа на PL/SQL. По давней традиции, восходящей к языку C, учебники по языкам программирования начинаются с программы «Hello, World!» и описания типов данных изучаемого языка. Не будем нарушать эту традицию, но внесем в нее одно изменение. Так как язык PL/SQL предназначен для работы с базами данных Oracle, то строку «Hello, World!» не будем задавать статически в исходном коде, а возьмем из

таблицы базы данных.

SQL> SET SERVEROUTPUT ON

SQL> CREATE TABLE hello\_world\_table (message VARCHAR2(30));

SQL> INSERT INTO hello\_world\_table VALUES('Hello, World!');

Создадим файл сценария в БЛОКНОТЕ – c именем test0.sql

|  |
| --- |
| Declare  J integer := 0;  Today date := sysdate; -- присвоение переменной значения системной даты  Name varchar2(30); /\* объявление переменной без присвоения значения \*/  greetings varchar2(20);  Begin  J := J + 1;  Name := 'Иванов И.И.';  /\* присвоение greetings:='HELLO WORLD' из таблицы \*/  SELECT MESSAGE INTO greetings FROM hello\_world\_table;  DBMS\_OUTPUT.PUT\_LINE(j);  DBMS\_OUTPUT.PUT\_LINE(GREETINGS);  DBMS\_OUTPUT.PUT\_LINE(today);  End;  / |

Выполним файл сценария: SQL> @c:\test0

**Задание 2:** Добавьте в файл сценария код

|  |
| --- |
| DBMS\_OUTPUT.PUT\_LINE(today||greetings||'\*\*\*\* temprecher\*\*\*\* температура =ln (j)= '||LN(j)); |

Выполните сценарий: SQL> @c:\test0

Выполните код:

SQL> INSERT INTO hello\_world\_table VALUES(‘Today cold');

Выполним файл сценария: SQL> @c:\test0

Объясните ошибку.

Задание 3: Измените таблицу (добавьте в таблицу столбец <TEMP> (температура), …)

SQL> alter table hello\_world\_table add temp number(4,2);

SQL> insert into hello\_world\_table values ('today cold',-5);

SQL> insert into into hello\_world\_table values ('today hot',20);

/\* изменить \*/

SQL> update hello\_world\_table set temp=0 where message='hello word';

SQL> select \* from hello\_world\_table;

**Задание 3:** Добавьте в файл сценария код

|  |
| --- |
| lamda NUMBER; -- temprecher  text1 VARCHAR2(30) := 'Cold';  text2 VARCHAR2(30) := 'Hot';  -- исполняемый раздел  lamda := &Input\_Data;  DBMS\_OUTPUT.PUT\_LINE('');  IF (lamda < 0)  THEN DBMS\_OUTPUT.PUT\_LINE(text1);  ELSIF (lamda > 0)  THEN DBMS\_OUTPUT.PUT\_LINE(text2);  END IF;  !!!!!!!!!!!!!!!!! заключите в комментарий /\* SELECT MESSAGE INTO greetings FROM hello\_world\_table;\*/ |

SQL> set echo off

Выполним файл сценария: SQL> @c:\test0

**Задание 4:** Замените в файле сценария код lamda := &Input\_Data; на ---------------------------------

SELECT temp INTO lamda FROM hello\_world\_table where temp>0; !!! [<=>] !!!

Перепишем пример с использованием поисковой команды CASE вместо команды IF (сравните с

предыдущей версией):

|  |
| --- |
| SELECT temp INTO lamda FROM hello\_world\_table where temp>0;  CASE  WHEN (lamda > 0)  THEN DBMS\_OUTPUT.PUT\_LINE(text1);  ELSE  DBMS\_OUTPUT.PUT\_LINE(text2);  END CASE; |

Выполним файл сценария: SQL> @c:\test0

**Задание 5:**

Команда перехода GOTO

Команда перехода GOTO позволяет осуществить переход по метке, присутствующей в коде PL/SQL. С помощью уникального идентификатора, заключенного в двойные угловые скобки (метки), можно пометить любую часть исполняемого блока PL/SQL для перехода в это место.

SQL> DECLARE

2 s NUMBER := 1;

3 BEGIN

4 IF s = 1 THEN

5 GOTO mybranch; -- переход по метке mybranch

6 ELSE

7 s := 1;

8 END IF;

9 <<mybranch>> -- установка метки mybranch

10 NULL;

11 END;

12 /

PL/SQL procedure successfully completed

**Задание 6:** Команда NULL

Команда NULL («пустая» команда) обычно используется как «заглушка» в месте, где надо написать какую-нибудь команду, потому что ничего не написать там нельзя по требованиям синтаксиса PL/SQL. Потом, по мере появления определенности, «заглушка» заменяется на функциональный код:

CASE sex

WHEN 'М' THEN

sex\_decoded := 'male';

WHEN 'F' THEN

sex\_decoded := 'female';

ELSE

NULL; -- toDo: write code for exception sex not in list {F,M} ;))

END CASE;

**Задание 7:** Простой цикл

Простой цикл рассмотрим на примере определения числа, факториал которого является наимень-шим числом, впервые превышающим заданную константу (1 000 000 000).

SQL> DECLARE

2 arg NUMBER; -- Переменная для вычисления факториала

3 i NUMBER; -- Переменная-счетчик

4 limit NUMBER := 1000000000; -- Граница

5 text1 VARCHAR2(80) := 'n! числа, впервые превышающий 1000000000';

6

7 BEGIN

8 i := 0;

9 arg := 1;

10 LOOP

11 EXIT WHEN arg > limit;

12 arg := arg\*(i+1);

13 i := i + 1;

14 END LOOP;

15 DBMS\_OUTPUT.PUT\_LINE(text1);

16 DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(arg));

17 DBMS\_OUTPUT.PUT\_LINE('Искомое число = '||TO\_CHAR(i));

18 END;

/

================================================================

n! числа, впервые превышающий 1000000000

6227020800

Искомое число = 13

PL/SQL procedure successfully completed.

Из любого цикла в PL/SQL можно выйти командой EXIT с указанием логического условия выхода. В основном команда EXIT используется в простых циклах, потому что в циклах FOR и WHILE и так явно указываются условия окончания цикла, а иметь в коде больше одного условия окончания для цикла является плохим стилем программирования. Если происходит зацикливание (выполнение бесконечного цикла без выхода из него), то программа PL/SQL «уходит в себя» («повисает»). Для прекращения выполнения такой программы следует нажать на клавиатуре комбинацию клавиш Ctrl+C:

SQL> BEGIN LOOP NULL; END LOOP; END;

2 /

^C

**Задание 8:**

Цикл WHILE

Цикл WHILE с предусловием позволяет выполнить одну и ту же последовательность команд PL/SQL пока истинно проверяемое предусловие. С помощью цикла WHILE найдем число, факториал которого является наименьшим числом, впервые превышающим 1 000 000 000 000:

SQL> DECLARE

2 arg NUMBER; -- Переменная для вычисления факториала

3 i NUMBER; -- Переменная-счетчик

4 limit NUMBER := 1000000000000;-- Граница

5 text1 VARCHAR2(80):='n! числа, впервые превышающий 1000000000000;

6

7 BEGIN

8 i := 0;

9 arg := i;

10 WHILE arg < 1000000000000 LOOP

11 arg := arg\*(i+1);

12 i := i + 1;

13 END LOOP;

14 DBMS\_OUTPUT.PUT\_LINE(text1);

15 DBMS\_OUTPUT.PUT\_LINE(TO\_CHAR(arg));

16 DBMS\_OUTPUT.PUT\_LINE('Искомое число = '||TO\_CHAR(i));

17 END;

/

n! числа,впервые превышающий 1000000000000

1307674368000

Искомое число = 15

PL/SQL procedure successfully completed.

Отметим, что если условие цикла WHILE изначально ложно (FALSE), то цикл не выполнится ни разу.

**Задание 9:**

Цикл FOR

Цикл FOR («цикл со счетчиком»), используется в том случае, когда известно, сколько раз нужно выполнить итерацию цикла. Приведем пример вычисления факториала заданного числа.

SQL> DECLARE

2 arg NUMBER := 1;

3 n NUMBER := 20;

4 text1 VARCHAR2(30) := 'Факториал числа '||n||' = ';

5 BEGIN

6 FOR i IN 1..n LOOP

7 arg := arg\*i;

8 END LOOP;

9 DBMS\_OUTPUT.PUT\_LINE(text1||TO\_CHAR(arg));

10 END;

/

Факториал числа 20 = 2432902008176640000

PL/SQL procedure successfully completed.

=====================================================