***Практика 1: Введение в PL/SQL***

1. Какой из следующих блоков PL/SQL выполнится успешно?
   1. BEGIN

END;

* 1. DECLARE

v\_amount INTEGER(10);

END;

* 1. DECLARE

BEGIN

END;

* 1. DECLARE

v\_amount INTEGER(10);

BEGIN

DBMS\_OUTPUT.PUT\_LINE(v\_amount);

END;

1. Создайте и выполните простой анонимный блок, который выводит фразу «Hello World». Выполните и сохраните скрипт как lab\_01\_02\_soln.sql.

SET SERVEROUTPUT ON

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello World');

END;

***Практика 2: Объявление переменных PL/SQL***

1) Определите допустимые и недопустимые идентификаторы:

a) today

b) last\_name

c) today’s\_date

d) Number\_of\_days\_in\_February\_this\_year

e) Isleap$year

f) #number

g) NUMBER#

h) number1to7

допустимые: a, b, e, g, h

недопустимые: c, d, f

2) Определите допустимые и недопустимые объявление и инициализацию переменных:

a) number\_of\_copies PLS\_NUMBER;

b) PRINTER\_NAME constant VARCHAR2(10);

c) deliver\_to VARCHAR2(10) := Johnson;

d) by\_when DATE := CURRENT\_DATE+1;

допустимые: d

недопустимые: a, b, c

3) Рассмотрите следующий блок, а затем выберите те утверждения, которые являются правдой:

DECLARE

v\_fname VARCHAR2(20);

v\_lname VARCHAR2(15) DEFAULT ‘fernandez’;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(v\_fname ||’ ‘ || v\_lname);

END;

a) Блок выполнится успешно и выведет «fernandez». /если использовать нормальные одинарные кавычки

b) Блок вызовет ошибку, поскольку переменная v\_fname используется без инициализации.

c) Блок выполнится успешно и выведет «null fernandez».

d) Блок вызовет ошибку, поскольку нельзя использовать ключевое слово DEFAULT лоя инициализации переменной типа VARCHAR2.

e) Блок вызовет ошибку, поскольку переменная v\_fname не объявлена.

4) Измените существующий анонимный блок и сохраните его как новый скрипт.

a) Откройте скрипт lab\_01\_02\_soln.sql, который создали в Практике 1.

b) В этом PL/SQL блоке объявите:

1. v\_today типа DATE. Инициализируйте v\_today функцией SYSDATE.

2. v\_tommorow типа переменной v\_today. Используйте атрибут %TYPE для объявления этой переменной.

c) В исполняемой секции:

1. Инициализируйте переменную v\_tommorow выражением, представляющим «завтрашнюю» дату (добавить 1 к значения переменной today).

2. Выведите значения переменных v\_today и v\_tommorow после вывода «Hello World».

d) Сохраните ваш скрипт как lab\_02\_04\_soln.sql, и затем выполните.

Пример вывода:



DECLARE

v\_today DATE := SYSDATE;

v\_tomorrow v\_today%TYPE;

BEGIN

v\_tomorrow := v\_today + 1;

DBMS\_OUTPUT.PUT\_LINE('Hello World');

DBMS\_OUTPUT.PUT\_LINE('TODAY IS: ' || v\_today);

DBMS\_OUTPUT.PUT\_LINE('TOMORROW IS: ' || v\_tomorrow);

END;

5) Отредактируйте скрипт lab\_02\_04\_soln.sql.

a) Добавьте код для создания двух явязанных переменных b\_basic\_percent и b\_pf\_percent типа NUMBER.

b) В исполнимой секции PL/SQL блока присвойте значения 45 и 12 переменным b\_basic\_percent и b\_pf\_percent соответственно.

c) Завершите блок PL/SQL символом «/» и отобразите значения связанных переменных, используя команду PRINT.

d) Выполните и сохраните скрипт как lab\_02\_05\_soln.sql. Пример вывода:



VARIABLE b\_basic\_percent NUMBER;

Variable b\_pf\_percent NUMBER;

DECLARE

v\_today DATE := SYSDATE;

v\_tomorrow v\_today%TYPE;

BEGIN

v\_tomorrow := v\_today + 1;

DBMS\_OUTPUT.PUT\_LINE('Hello World');

DBMS\_OUTPUT.PUT\_LINE('TODAY IS: ' || v\_today);

DBMS\_OUTPUT.PUT\_LINE('TOMORROW IS: ' || v\_tomorrow);

:b\_basic\_percent := 45;

:b\_pf\_percent := 12;

END;

/

PRINT b\_basic\_percent

PRINT b\_pf\_percent

***Практика 3: Написание исполняемых операторов***

1) Проанализируйте следующий PL/SQL блок и определите тип данных и значение каждой из следующих переменных в соответствии с областью видимости:



a) Значение v\_weight в позиции 1 равно: 2, тип NUMBER

b) Значение v\_new\_locn в позиции 1 равно: Western Europe, тип VARCHAR2

c) Значение v\_weight в позиции 2 равно: 601, тип NUMBER

d) Значение v\_message в позиции 2 равно: Product 10012 is in stock, VARCHAR2

e) Значение v\_new\_locn в позиции 2 равно: не определено

2) В следующем PL/SQL блоке определите значения и типы данных для каждого из следующих случаев:



a) Значение v\_customer во вложенном блоке: 201, тип NUMBER

b) Значение v\_name во вложенном блоке: Unisports, тип VARCHAR2

c) Значение v\_credit\_rating во вложенном блоке: GOOD, тип VARCHAR2

d) Значение v\_customer в главном блоке: Womansport, тип VARCHAR2

e) Значение v\_name в главном блоке: не определено

f) Значение v\_credit\_rating в главном блоке: EXCELLENT, тип VARCHAR2

3) Откройте скрипт lab\_02\_05\_soln.sql и отредактируйте его следующим образом:

a) Используя синтаксис комментария одной строки, закомментируйте строки по созданию связанных переменных и включите переменную SERVEROUTPUT.

b) Используя многострочный комментарий, закомментируйте в исполняемой секции строки, которые выполняют присваивание связанных переменных.

c) В секции объявления:

1. Объявите и инициализируйте 2 временные переменные для замены закомментированных связанных переменных.

2. Объявите 2 дополнительные переменные: v\_fname типа VARCHAR2 размером 15 и v\_emp\_sal типа NUMBER размером 10.

d) Включите следующий SQL-оператор в исполняемую секцию:



e) Измените в строке, печатающей «Hello World» на «Hello» и Имя. Затем закомментируйте строки, которые отображали даты и выводили связанные переменные.

f) Рассчитайте вклад сотрудника в резервный фонд (РФ). РФ составляет 12% от базовой зарплаты, а базовая зарплата – 45% от основной. Используйте локальную переменную для подсчёта. Попытайтесь написать расчёт РФ одним выражением. Выведите зарплату сотрудника и его рассчитанный РФ.

g) Выполните и сохраните скрипт как lab\_03\_03\_soln.sql. Пример вывода:



--VARIABLE b\_basic\_percent NUMBER;

--Variable b\_pf\_percent NUMBER;

SET SERVEROUTPUT ON

DECLARE

v\_today DATE := SYSDATE;

v\_tomorrow v\_today%TYPE;

v\_fname VARCHAR(15);

v\_emp\_sal NUMBER (10);

v\_basic\_percent NUMBER := 45;

v\_pf\_percent NUMBER := 12;

BEGIN

-- v\_tomorrow := v\_today + 1;

SELECT first\_name, salary

INTO v\_fname, v\_emp\_sal

FROM employees

WHERE employee\_id = 110;

DBMS\_OUTPUT.PUT\_LINE('Hello '|| v\_fname);

DBMS\_OUTPUT.PUT\_LINE ('YOUR SALARY IS :' || v\_emp\_sal);

DBMS\_OUTPUT.PUT\_LINE ('YOUR CONTRIBUTION TOWARDS PF: ' ||

v\_emp\_sal/100\*v\_basic\_percent/100\*v\_pf\_percent);

--DBMS\_OUTPUT.PUT\_LINE('TODAY IS: ' || v\_today);

--DBMS\_OUTPUT.PUT\_LINE('TOMORROW IS: ' || v\_tomorrow);

/\*:b\_basic\_percent := 45;

:b\_pf\_percent := 12;\*/

END;

/\*

PRINT b\_basic\_percent

PRINT b\_pf\_percent\*/

***Практика 4: Взаимодействие с Oracle Server***

1) Создайте PL/SQL блок, который выбирает максимальный department ID в таблице departments и сохраняет его в переменную v\_max\_depno. Выведите максимальный department ID.

a) Объявите переменную v\_max\_depno типа NUMBER в секции объявлений.

b) Начните исполняемую секцию ключевым словом BEGIN и напишите оператор SELECT для запроса максимального depertment\_id из таблицы departments.

c) Выведите значение v\_max\_deptno и завершите исполнимый блок.

d) Выполните и сохраните скрипт как lab\_04\_01\_soln.sql. Пример вывода:



DECLARE

v\_max\_deptno NUMBER;

BEGIN

SELECT MAX(department\_id)

INTO v\_max\_deptno

FROM departments;

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is: ' || v\_max\_deptno);

END;

2) Измените блок PL/SQL, который был создан в задании 1, для вставки нового отдела в таблицу departments.

a) Загрузите скрипт lab\_04\_01\_soln.sql. Объявите 2 переменные:

v\_dept\_name типа departments.department\_name и

v\_dept\_id типа NUMBER.

Присвойте переменной v\_dept\_name значение 'Education' в секции описания.

b) Вы уже получили текущий максимальный номер отдела из таблицы departments. Добавьте 10 к нему и присвойте результат переменной v\_dept\_id.

c) Напишите оператор INSERT для вставки данных в столбцы department\_name, department\_id и location\_id таблицы departments. Используйте значения v\_dept\_name и v\_dept\_id для столбцов department\_name и department\_id соответственно, и NULL для столбца location\_id.

d) Используйте SQL-атрибут SQL%ROWCOUNT для вывода числа обработанных записей.

e) Выполните оператор SELECT для проверки, добавился ли новый отдел. Для этого можно завершите PL/SQL блок символом “/” и допишите оператор SELECT в ваш скрипт.

f) Выполните и сохраните ваш скрипт как lab\_04\_02\_soln.sql. Пример вывода:



DECLARE

v\_max\_deptno NUMBER;

v\_dept\_name departments.department\_name%TYPE := 'Education';

v\_dept\_id NUMBER;

BEGIN

SELECT MAX(department\_id)

INTO v\_max\_deptno

FROM departments;

v\_dept\_id := 10 + v\_max\_deptno;

INSERT INTO departments (department\_id, department\_name, location\_id)

VALUES (v\_dept\_id, v\_dept\_name, NULL);

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is: ' || v\_max\_deptno);

DBMS\_OUTPUT.PUT\_LINE('SQL%ROWCOUNT gives ' || SQL%ROWCOUNT);

END;

/

SELECT\*FROM departments

WHERE department\_id = 280;

3) В задании 2 вы установили location\_id в NULL. Создайте PL/SQL блок, который обновляет location\_id в значение 3000 для нового отдела.

a) Начните исполняемый блок с ключевого слова BEGIN. Напишите оператор UPDATE, чтобы установить location\_id в значение 3000 для нового отдела (dept\_id = 280).

b) Завершите исполняемый блок ключевым словом END. Закончите PL/SQL блок символом «/» и добавьте оператор SELECT для отображения отдела, который вы изменили.

c) Напишите оператор DELETE, чтобы удалить отдел, который вы добавили.

d) Выполните и сохраните скрипт как lab\_04\_03\_soln.sql. Пример вывода:



DECLARE

v\_max\_deptno NUMBER;

v\_dept\_name departments.department\_name%TYPE := 'Education';

v\_dept\_id NUMBER;

BEGIN

UPDATE departments

SET location\_id = 3000

WHERE department\_id = 280;

SELECT MAX(department\_id)

INTO v\_max\_deptno

FROM departments;

v\_dept\_id := 10 + v\_max\_deptno;

INSERT INTO departments (department\_id, department\_name, location\_id)

VALUES (v\_dept\_id, v\_dept\_name, NULL);

DBMS\_OUTPUT.PUT\_LINE('The maximum department\_id is: ' || v\_max\_deptno);

DBMS\_OUTPUT.PUT\_LINE('SQL%ROWCOUNT gives ' || SQL%ROWCOUNT);

END;

/

SELECT\*FROM departments

WHERE department\_id = 280;

DELETE FROM departments

WHERE department\_id = 280;

***Практика 5: Написание управляющих структур***

Создайте таблицу messages, содержащую 1 столбец results типа NUMBER(6).

1) Напишите PL/SQL блок для вставки данных в таблицу messages.

a) Вставьте числа от 1 до 10, исключая 6 и 8.

b) Зафиксируйте изменения перед концом блока.

c) Выполните оператор SELECT для проверки, что ваш PL/SQL блок работает.

Вы должны увидеть следующий результат:



CREATE TABLE messages (results NUMBER(6));

BEGIN

FOR i in 1..10 LOOP

IF i = 6 or i = 8 THEN CONTINUE;

ELSE INSERT INTO messages(results) VALUES (i);

END IF;

END LOOP;

COMMIT;

END;

/

SELECT \* FROM messages;

2) Создайте таблицу EMP как копию таблицы employees. Измените структуру таблицы EMP, добавив новый столбец stars типа VARCHAR2 размерностью 50. Создайте PL/SQL блок, который добавляет звёздочку в столбец stars для каждых $1000 зарплаты сотрудника. Сохраните скрипт как lab\_05\_02\_soln.sql.

a) В декларативной секции блока объявите переменную v\_empno типа emp.employee\_id и инициализируйте её значением 176. Объявите переменную v\_asterisk типа emp.stars и установите её в NULL. Создайте переменную v\_sal типа emp.salary.

b) В исполняемой секции напишите логику для добавления символа «\*» к строке для каждых $1000 зарплаты. Например, если сотрудник получает $8000, строка должна содержать 8 звёздочек. Если сотрудник зарабатывает $12500, строка должна содержать 13 звёздочек.

c) Измените столбец stars для сотрудника, проставив туда новую строку со звёздочками. Зафиксируйте изменения перед концом блока.

d) Отобразит строку из таблицы emp для проверки, успешно ли выполняется ваш PL/SQL блок.

e) Выполните и сохраните ваш скрипт как lab\_05\_02\_soln.sql. Пример вывода:



CREATE TABLE EMP AS SELECT \* FROM employees;

ALTER TABLE EMP

ADD stars VARCHAR2(50);

DECLARE

v\_empno emp.employee\_id%TYPE:=176;

v\_asterisk emp.stars%TYPE:=Null;

v\_sal emp.salary%TYPE;

BEGIN

    SELECT NVL(ROUND(salary/1000),0) INTO v\_sal

    FROM emp WHERE employee\_id = v\_empno;

    FOR i IN 1..v\_sal

    LOOP v\_asterisk:=v\_asterisk || '\*';

    END LOOP;

    UPDATE emp SET stars = v\_asterisk

    WHERE employee\_id = v\_empno;

    COMMIT;

END;

/

SELECT employee\_id, salary, stars

FROM emp

WHERE employee\_id = 176;

***Практика 6: Работа с составными типами данных***

1) Напишите PL/SQL блок для отображения информации о данной стране.

a) Объявите PL/SQL запись, основанную на структуре таблицы countries.

b) Объявите переменную v\_countryid. Присвойте ей значение «CA».

c) В декларативной секции, используя %ROWTYPE-атрибут, объявите переменную v\_country\_record типа countries.

d) В исполняемой секции получите всю информацию из таблицы countries, используя v\_countryid. Отобразите выбранную информацию о стране. Пример вывода:



e) Выполните и протестируйте PL/SQL блок для стран с ID: DE, UK и US.

SET SERVEROUTPUT ON

SET VERIFY OFF

DECLARE

v\_countryid varchar2(20):= 'CA';

v\_country\_record countries%ROWTYPE;

BEGIN

SELECT \*

INTO v\_country\_record

FROM countries

WHERE country\_id = UPPER(v\_countryid);

DBMS\_OUTPUT.PUT\_LINE('Country ID: ' || v\_country\_record.country\_id ||

' Company Name: ' || v\_country\_record.country\_name || 'Region: ' ||

v\_country\_record.region\_id);

END;

2) Создайте PL/SQL блок для запроса названий некоторых отделов из таблицы departments и вывода каждого названия отдела на экран, используя ассоциативный массив. Сохраните скрипт как lab\_06\_02\_soln.sql.

a) Объявите тип dept\_table\_type как INDEX BY-таблицу типа departments.department\_name. Объявите переменную my\_dept\_table типа dept\_table\_type для временного хранения названий отделов.

b) Объявите две переменные: f\_loop\_count и v\_deptno типа NUMBER. Присвойте в f\_loop\_count значение 10, а в v\_deptno – 0.

c) Используя простой оператор цикла, получите названия 10 отделов и сохраните их в ассоциативном массиве. Начние department\_id с 10. Увеличивайте v\_deptno с шагом 10 для каждой итерации цикла. Следующая таблица показывает department\_id, для которых вы должны запросить department\_name.



d) Используя другой цикл, запросите названия отделов из ассоциативного массива и выведите его на экран.

e) Выполните и сохраните скрипт как lab\_06\_02\_soln.sql. Пример вывода:



SET SERVEROUTPUT ON

DECLARE

TYPE dept\_table\_type is table OF departments.department\_name%TYPE

INDEX BY PLS\_INTEGER;

my\_dept\_table dept\_table\_type;

f\_loop\_count NUMBER:= 10;

v\_deptno NUMBER:= 0;

BEGIN

FOR i IN 1..f\_loop\_count LOOP

v\_deptno := v\_deptno+10;

SELECT department\_name INTO my\_dept\_table(i)

FROM departments

WHERE department\_id = v\_deptno;

END LOOP;

FOR i IN 1..f\_loop\_count LOOP

DBMS\_OUTPUT.PUT\_LINE (my\_dept\_table(i));

END LOOP;

END;

3) Измените блок, который вы создали в Практике 2 для запроса информации о каждом отделе из таблицы departments и отобразите информацию. Используйте ассоциативный массив INDEX BY.

a) Загрузите lab\_06\_02\_soln.sql.

b) Вы объявляли ассоциативный массив как тип departments.department\_name. Измените объявление ассоциативного массива для временного хранения числа, названия и местоположения всех отделов. Используйте %ROWTYPE-атрибут.

c) Измените оператор SELECT для запроса информации по всем отделам из таблицы departments для сохранения значений в ассоциативном массиве.

d) Используя другой цикл, запросите информацию по отделам из ассоциативного массива и выведите её на экран. Пример вывода:



SET SERVEROUTPUT ON

DECLARE

    TYPE dept\_table\_type is table OF departments%ROWTYPE

    INDEX BY PLS\_INTEGER;

    my\_dept\_table dept\_table\_type;

    f\_loop\_count NUMBER:= 10;

    v\_deptno NUMBER:= 0;

BEGIN

    FOR i IN 1..f\_loop\_count LOOP

        v\_deptno := v\_deptno+10;

        SELECT \* INTO my\_dept\_table(i)

        FROM departments

        WHERE department\_id = v\_deptno;

    END LOOP;

    FOR i IN 1..f\_loop\_count LOOP

        DBMS\_OUTPUT.PUT\_LINE ('Department Number: ' ||

        my\_dept\_table(i).department\_name || 'Manager ID: '||

        my\_dept\_table(i).manager\_id || 'Location ID: '||

        my\_dept\_table(i).location\_id);

    END LOOP;

END;

***Практика 7-1: Использование явных курсоров***

В этой практике выполняются два упражнения:

  Первое: используйте явный курсор для обработки нескольких строк из таблицы и заполните другую таблицу результатом с помощью курсора FOR-цикла.

  Второе: вы напишите PL/SQL блок, который обрабатывает информацию двумя курсорами, один из которых использует параметр.

1) Создайте PL/SQL блок для выполнения следующих действий:

a) В описательной секции объявите и инициализируйте переменную v\_deptno типа NUMBER. Присвойте корректное значение department ID (см. таблицу в пункте d).

b) Объявите курсор с именем c\_emp\_cursor, который выбирает last\_name, salary и manager\_id сотрудника, работающего в отделе, указанном в v\_deptno.

c) В исполняемой секции используйте цикл FOR для обработки полученных данных. Если зарплата сотрудника меньше 5000 и его manager ID или 101, или 124, отобразите сообщение «<<*last\_name*>> для повышения». Иначе выведите сообщение «<<*last\_name*>> не для повышения».

d) Проверьте PL/SQL блок для следующих случаев:



DECLARE

v\_deptno NUMBER := 50;

CURSOR c\_emp\_cursor IS

    SELECT last\_name, salary, manager\_id

    FROM employees

    WHERE department\_id = v\_deptno;

BEGIN

    FOR emp\_record IN c\_emp\_cursor LOOP

        IF emp\_record.salary < 5000 AND (emp\_record.manager\_id = 101 OR

        emp\_record.manager\_id = 124)

        THEN DBMS\_OUTPUT.PUT\_LINE(emp\_record.last\_name || ' для повышения');

        ELSE DBMS\_OUTPUT.PUT\_LINE (emp\_record.last\_name || ' не для повышения');

        END IF;

    END LOOP;

END;

2) Напишите PL/SQL блок, который объявляет и использует два курсора: один без параметра, а другой – с параметром. Первый курсор выбирает номер отдела и название отдела из таблицы departments для всех отделов, чей номер ID меньше 100. Второй курсор принимает номер отдела как параметр и запрашивает детальную информацию по сотруднику, кто работает в отделе и чей employee\_id меньше 120.

a) Объявите курсор c\_dept\_cursor для запроса department\_id и department\_name для тех отделов, чей department\_id меньше 100. Отсортируйте по department\_id.

b) Объявите другой курсор c\_emp\_cursor, который принимает номер отдела как параметр и выбирает следующие данные из таблицы employees:

last\_name, job\_id, hire\_date и salary тех сотрудников, кто работает в том же отделе, что и сотрудники с employee\_id менее 120.

c) Объявите переменные для того, чтобы сохранить туда значения, выбираемые каждым курсором. Используйте атрибут %TYPE для объявления переменных.

d) Откройте курсор c\_dept\_cursor и используйте простой цикл для получения значеий в объявленные переменные. Выведите номер и название отдела. Используйте подходящий курсорный атрибут для выхода из цикла.

e) Откройте курсор c\_emp\_cursor, передав текущий номер отдела как параметр. Начните другой цикл и запросите значения c\_emp\_cursor в переменные, и выведите всю запрошенную детальную информацию из таблицы employees.

**Примечание**

* Проверьте, открыт ли уже курсор c\_ emp\_cursor.
* Используйте подходящий курсорный атрибут для выхода из условия.
* Когда цикл выполнится, напечатайте линию после отображённых детальных данных по каждому отделу и закройте курсор c\_emp\_cursor.

f) Завершите первый цикл и закройте c\_dept\_cursor. Затем завершите исполнимую секцию.

g) Выполните скрипт. Пример вывода:



DECLARE

CURSOR c\_dept\_cursor IS

    SELECT department\_id, department\_name

    FROM departments

    WHERE department\_id < 100

    ORDER BY department\_id;

CURSOR c\_emp\_cursor (v\_deptno NUMBER) IS

    SELECT last\_name, job\_id, hire\_date, salary

    FROM employees

    WHERE department\_id = v\_deptno AND employee\_id < 120;

v\_current\_deptno departments.department\_id%TYPE;

v\_current\_dname departments.department\_name%TYPE;

v\_ename employees.last\_name%TYPE;

v\_job employees.job\_id%TYPE;

v\_hiredate employees.hire\_date%TYPE;

v\_sal employees.salary%TYPE;

BEGIN

    OPEN c\_dept\_cursor;

    LOOP

        FETCH c\_dept\_cursor INTO v\_current\_deptno,

        v\_current\_dname;

        EXIT WHEN c\_dept\_cursor%NOTFOUND;

        DBMS\_OUTPUT.PUT\_LINE('Department Number: ' || v\_current\_deptno ||

        ' Department Name: ' || v\_current\_dname);

        IF c\_emp\_cursor%ISOPEN THEN

            CLOSE c\_emp\_cursor;

        END IF;

    OPEN c\_emp\_cursor (v\_current\_deptno);

        LOOP

            FETCH c\_emp\_cursor INTO v\_ename, v\_job, v\_hiredate, v\_sal;

            EXIT WHEN c\_emp\_cursor%NOTFOUND;

            DBMS\_OUTPUT.PUT\_LINE(v\_ename || '   '|| v\_job || '    ' ||

            v\_hiredate || '    ' || v\_sal);

        END LOOP;

        DBMS\_OUTPUT.PUT\_LINE('---------------------------------------------');

        CLOSE c\_emp\_cursor;

    END LOOP;

CLOSE c\_dept\_cursor;

END;

***Практика 7-2: Использование явных курсоров***

Создайте PL/SQL блок, который использует явный курсор для определения наивысших **n** зарплат сотрудников.

1) Создайте таблицу top\_salaries, содержащий один столбец salary типа NUMBER(8,2) для хранения наивысших зарплат сотрудников.

2) В декларативной секции объявите переменную v\_num типа number, которая содержит число **n**, представляющее n наиболее высокооплачиваемых сотрудников таблицы employees. Например, чтобы увидеть 5 наиболее высоких зарплат, присвойте этой переменной значение 5.

Объявите другую переменную sal типа employees.salary. Объявите курсор c\_emp\_cursor, который запрашивает зарплаты сотрудников в убывающем порядке. Примечание: зарплаты не должны повторяться.

3)В исполнимой секции напишите цикл и запросите **n** наибольших зарплат и затем вставьте их в таблицу top\_salaries. Можно использовать простой цикл для обработки данных. Также используйте курсорные атрибуты %ROWCOUNT и %FOUND для выхода из цикла.

**Примечание:** Убедитесь, что вы написали корректно условие выхода, чтобы не было бесконечного цикла.

4) После вставки данных в таблицу top\_salaries отобразите строки с помощью оператора SELECT. Пример вывода информации о пяти наивысших зарплатах в таблице employees:

******

5) Проверьте различные случаи, такие как v\_num = 0 или когда v\_num больше, чем число сотрудников в таблице employees. Очищайте таблицу top\_salaries после каждого теста.

CREATE TABLE top\_salaries (salary NUMBER(8,2));

DECLARE

v\_num NUMBER := 5;

v\_sal employees.salary%TYPE;

CURSOR c\_emp\_cursor IS

    SELECT salary

    FROM employees

    ORDER BY salary DESC;

BEGIN

    OPEN c\_emp\_cursor;

    FETCH c\_emp\_cursor INTO v\_sal;

    WHILE c\_emp\_cursor%ROWCOUNT <= v\_num AND c\_emp\_cursor%FOUND

    LOOP

    INSERT INTO top\_salaries(salary) VALUES (v\_sal);

    FETCH c\_emp\_cursor INTO v\_sal;

    END LOOP;

    CLOSE c\_emp\_cursor;

END;

/

SELECT \*

FROM top\_salaries;

***Практика 8-1: Обработка предопределённых исключений***

1) Удалите все данные таблицы messages. Измените тип данных столбца на varchar2(1000).

2) В декларативной секции объявите две переменные: v\_ename типа employees.last\_name и v\_emp\_sal типа employees.salary. Инициализируйте последнюю в 6000.

3) В исполнимой секции запросите фамилии сотрудников, чьи зарплаты равны значению v\_emp\_sal. Если возвращается только одна строка, вставьте в таблицу messages имя сотрудника и его зарплату.

**Примечание:** Не используйте явный курсор.

4) Если для введённой зарплаты не выводится ни одной строки, обработайте это исключение соответствующим обработчиком и вставьте в таблицу messages сообщение «Нет сотрудника с зарплатой <salary>».

5) Если для указанной зарплаты возвращается несколько записей, обработайте это исключение соответствующим обработчиком и вставьте в таблицу messages сообщение «Более одного сотрудника с зарплатой <salary>».

6) Обработайте любое другое исключение соответствующим обработчиком и вставьте в таблицу messages сообщение «Возникла другая ошибка».

7) Выведите записи из таблицы messages для проверки, выполняется ли правильно PL/SQL блок. Пример вывода:



8) Измените инициализационное значение в переменной v\_emp\_sal на 2000 и выполните блок. Пример вывода:



TRUNCATE TABLE messages;

ALTER TABLE messages

MODIFY (results varchar2(1000));

DECLARE

v\_ename employees.last\_name%TYPE;

v\_emp\_sal employees.salary%TYPE:=6000;

BEGIN

SELECT last\_name INTO v\_ename

FROM employees

WHERE salary = v\_emp\_sal;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

INSERT INTO messages(results) VALUES('Нет сотрудника с зарплатой: '|| v\_emp\_sal);

WHEN TOO\_MANY\_ROWS THEN

INSERT INTO messages(results) VALUES(Более одного сотрудника с зарплатой'||v\_emp\_sal);

WHEN OTHERS THEN

INSERT INTO messages(results) VALUES('Возникла другая ошибка');

END;

/

SELECT \*

FROM messages

***Практика 8-2: Обработка стандартных исключений Oracle Server***

В этой практике вы напишите PL/SQL блок, который объявляет исключение для ошибки сервера Oracle ORA-02292 (integrity constraint violated – child record found).

1) В декларативной секции объявите исключение e\_childrecord\_exists. Ассоциируйте объявленное исключение со стандартной ошибкой сервера Oracle -02292.

2) В исполняемой секции выведите «Удаление отдела 40… ». Напишите оператор DELETE для удаления отедал с department\_id = 40.

3) Напишите секцию обработки исключения для обработки исключения e\_childrecord\_exists и отобразите соответствующее сообщение. Пример вывода:



***Практика 9: Создание и использование хранимых процедур***

В этой практике вы научитесь создавать хранимые в базе данных PL/SQL конструкции, создавать и управлять процедурами и функциями, изучите структуру и необходимые секции для процедур и функций.

1) Загрузите следующий скрипт:

SET SERVEROUTPUT ON

DECLARE

V\_today DATE := SYSDATE;

V\_tommorow v\_today%TYPE;

BEGIN

V\_tommorow := v\_today +1;

DBMS\_OUTPUT.PUT\_LINE(‘TODAY IS:’|| v\_today);

DBMS\_OUTPUT.PUT\_LINE(‘TOMMOROW IS:’|| v\_tommorow);

END;

/

a) Измените скрипт, чтобы преобразовать анонимный блок в процедуру greet.

b) Выполните сценарий, чтобы создать процедуру.



c) Сохраните скрипт как lab\_09\_01\_soln.sql.

d) Нажмите кнопку Clear, чтобы очистить рабочее пространство.

e) Создайте и выполните анонимный блок, чтобы вызвать процедуру greet.



2) Загрузите скрипт lab\_09\_01\_soln.sql.

a) Удалите процедуру greet: DROP PROCEDURE greet;

b) Измените процедуру, чтобы принимать параметр name типа VARCHAR2.

c) Выведите «Hello, <name>» вместо «Hello, World!».

d) Сохраните свой скрипт как lab\_09\_02\_soln.sql.

e) Выполните сценарий, чтобы создать процедуру. Результат должен быть таким:



f) Создайте и выполните анонимный блок, чтобы вызвать процедуру greet с параметром. Блок должен выводить текст. Пример вывода:



CREATE PROCEDURE greet(name VARCHAR2) IS

DECLARE

V\_today DATE := SYSDATE;

V\_tommorow v\_today%TYPE;

BEGIN

V\_tommorow := v\_today +1;

DBMS\_OUTPUT.PUT\_LINE('Hello, '||name||'!');

DBMS\_OUTPUT.PUT\_LINE('TODAY IS: '|| v\_today);

DBMS\_OUTPUT.PUT\_LINE('TOMMOROW IS: '|| v\_tommorow);

END;

EXECUTE GREET('Nancy');