**Практические занятия по курсу**

**«Разработка программных единиц PL/SQL»**

**Часть 1**

Оглавление

[Практическое занятие 1. Введение в PL/SQL 2](#_Toc411281281)

[Практическое занятие 2. Объявление переменных PL/SQL 3](#_Toc411281282)

[Практическое занятие 3. Команды исполняемой секции 5](#_Toc411281283)

[Практическое занятие 4. Взаимодействие с сервером Oracle 8](#_Toc411281284)

[Практическое занятие 5. Управляющие конструкции 9](#_Toc411281285)

[Практическое занятие 6. Работа с составными типами данных 10](#_Toc411281286)

[Практическое занятие 7. Использование явных курсоров 12](#_Toc411281287)

[Практическое занятие 8. Обработка исключений 15](#_Toc411281288)

# Практическое занятие 1. Введение в PL/SQL

* Типы блоков
* Структура блока
* Выполнение анонимного блока
* Вывод сообщений при помощи пакета DBMS\_OUTPUT

**1**. Какой из перечисленных блоков кода выполнится успешно?

BEGIN

END;

DECLARE

amount INTEGER(10);

END;

DECLARE

BEGIN

END;

DECLARE

amount INTEGER(10);

BEGIN

DBMS\_OUTPUT.PUT\_LINE(amount);

END;

**2**. Напишите и выполните анонимный блок кода для вывода фразы “Hello, World!”.

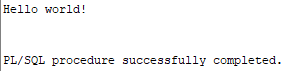
Сохраните созданный скрипт в файле lab\_01\_02.sql.

BEGIN

dbms\_output.put\_line('Hello world!');

END;

/



# Практическое занятие 2. Объявление переменных PL/SQL

* Требования к именам переменных
* Основные скалярные типы данных PL\SQL
* Атрибут %TYPE
* Связанные переменные

**1**. Укажите правильные и неправильные имена переменных:

a. today

b. last\_name

c. today’s date

d. Number\_of\_days\_February\_this\_year

e. IsLeap$year

f. #number

g. NUMBER#

h. number1to7

**2**. Укажите правильные и неправильные инструкции декларирования и инициализации:

a. Numbers\_of\_copies PLS\_INTEGER;

b. printer\_name constant VARCHAR2(10);

c. deliver\_to VARCHAR2(10):=Jonson;

d. by\_when DATE:=SYSDATE+1;

**3**. Проанализируйте следующий анонимный блок и выберите подходящее утверждение:

**SET SERVEROUTPUT ON**

**DECLARE**

**fname VARCHAR(20);**

**lname VARCHAR(15) DEFAULT ‘fernandez’;**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE(FNAME || ‘ ‘ || lname);**

**END;**

a. Блок выполнится успешно и распечатает ‘ fernandez’

b. Блок выдаст ошибку из-за того, что переменная fname используется без инициализации

c. Блок выполнится успешно и напечатает ‘null fernandez’

d. Блок выдаст ошибку, потому что нельзя использовать инструкцию DEFAULT для инициализации переменных типа VARCHAR2

e. Блок выдаст ошибку, потому что переменная FNAME не декларирована

**4**. Загрузите в редактор скрипт для создания анонимного блока из пункта 2 первой Лабораторной работы, который вы сохранили в файле: lab\_01\_02.sql.

a. Добавьте в блок секцию декларирования переменных и включите в нее:

i. Переменную с именем today, типом данных DATE и с инициализацией по умолчанию с помощью системной функции SYSDATE.

ii. Переменную tomorrow такого же типа данных, что и переменная today. Воспользуйтесь при этом директивой %TYPE.

b. В исполняемой секции блока инициализируйте переменную tomorrow с помощью выражения прибавляющего один день к текущему значению переменной today.

c. Распечатайте значения переменных today и tomorrow после фразы “Hello, World!”

**DECLARE**

**today DATE := SYSDATE;**

**tomorrow today%TYPE;**

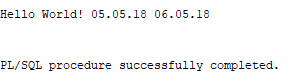
**BEGIN**

**tomorrow := today + 1;**

**DBMS\_OUTPUT.PUT\_LINE('Hello World! ' || today || ' ' || tomorrow);**

**END;**

**/**



**5**. Добавьте в скрипт 2 связанных (bind) переменных: basic\_perсent и pf\_percent типа NUMBER;

a. В исполняемой секции кода присвойте этим переменным значения 45 и 12 соответственно.

b. Отделите исполняемый блок от остальной части скрипта инструкцией «/» и выведите значения связанных переменных с помощью инструкции PRINT.

c. Сохраните скрипт под именем lab\_02\_05.sql – он пригодится вам в следующих работах.

**VARIABLE basic\_percent NUMBER**

**VARIABLE pf\_percent NUMBER**

**DECLARE**

**BEGIN**

**:basic\_percent := 45;**

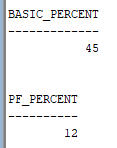
**:pf\_percent := 12;**

**END;**

**/**

**PRINT basic\_percent**

**PRINT pf\_percent**



# Практическое занятие 3. Команды исполняемой секции

* Виды преобразований типов данных
* Вложенные блоки и области видимости
* Метки блоков

**1**. Проанализируйте блок кода и дополните его трассировочными инструкциями, необходимыми для ответа на вопросы:

**DECLARE**

**weight NUMBER(3) := 600;**

**message VARCHAR2(255) := 'Product 10012';**

**BEGIN**

**DECLARE**

**weight NUMBER(3) := 1;**

**message VARCHAR2(255) := 'Product 11001';**

**new\_locn VARCHAR2(50) := 'Europe';**

**BEGIN**

**weight := weight + 1;**

**new\_locn := 'Western ' || new\_locn;**

**/\*(1)\*/**

**END;**

**weight := weight + 1;**

**message := message || ' is in stock';**

**new\_locn := 'Western ' || new\_locn;**

**/\*(2)\*/**

**END;**

**/**

a. Значение переменной weight в позиции (1) 2

b. Значение переменной new\_locn в позиции (1) Western Europe

c. Значение переменной weight в позиции (2) 601

d. Значение переменной message в позиции (2) Product 10012

e. Значение переменной new\_locn в позиции (2) – не определено

**2**. Проанализируйте блок кода и ответьте на вопросы (добавьте необходимые трассировочные инструкции):

**DECLARE**

**customer VARCHAR2(50) := 'Womansport';**

**credit\_rating VARCHAR2(50) := 'EXCELLENT';**

**BEGIN**

**DECLARE**

**customer NUMBER(7) := 201;**

**name VARCHAR2(25) := 'Unisports';**

**BEGIN**

**credit\_rating :='GOOD';**

**…**

**END;**

**…**

**END;**

**/**

1. Значение и тип данных переменной customer во вложенном блоке:

201, NUMBER(7)

1. Значение и тип данных переменной credit\_rating во вложенном блоке:

‘GOOD’, VARCHAR2(50)

1. Значение и тип данных переменной customer в основном (внешнем) блоке:

‘Womansport’, VARCHAR2(50)

1. Значение и тип данных переменной name в основном блоке:

Не определено

e. Значение и тип данных переменной credit\_rating в основном блоке:

‘EXCELLENT’, VARCHAR2(50)

**3**. Откройте скрипт, сохраненный в файле lab\_02\_05.sql. Выполните скрипт.

Посмотрите на результат его работы. Теперь в него надо внести изменения:

**Замечание: по смыслу и в соответствии с дальнейшими заданиями, необходимо**

**закомментировать декларацию и присвоение для переменных today и tomorrow.**

a. Закомментируйте строчку, где выполняется декларирование связанных переменных.

b. Закомментируйте инструкции, в которых связанным переменным присваиваются значения.

c. Декларируйте две переменные: fname типа VARCHAR2(15), и emp\_sal типа NUMBER(10).

d. В исполняемую секцию включите инструкцию выборки имени и зарплаты 110 сотрудника из таблицы employees.

SELECT first\_name, salary

INTO fname, emp\_sal FROM employees

WHERE employee\_id=110;

e. Поменяйте инструкцию, которая печатала «Hello, World!» на инструкцию, которая печатает «Hello, » и имя сотрудника.

f. Подсчитайте налоговый вычет на основе зарплаты сотрудника.

Используйте для этого следующую схему: налоговый вычет составляет 12% от базовой части зарплаты, а базовая часть зарплаты – это 45% от зарплаты.

g. После приветствия сотрудника по имени – распечатайте его зарплату, и подсчитанный налоговый вычет.

h. Результат должен выглядеть примерно так:



**--VARIABLE basic\_percent NUMBER**

**--VARIABLE pf\_percent NUMBER**

**DECLARE**

**fname VARCHAR2(15);**

**emp\_sal NUMBER(10);**

**BEGIN**

**-- :basic\_percent := 45;**

**-- :pf\_percent := 12;**

**SELECT first\_name, salary**

**INTO fname, emp\_sal**

**FROM emp**

**WHERE employee\_id = 110;**

**DBMS\_OUTPUT.PUT\_LINE('Hello ' || fname);**

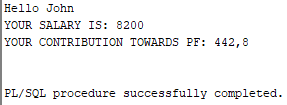
**DBMS\_OUTPUT.PUT\_LINE('YOUR SALARY IS: ' || emp\_sal);**

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

**ROUND(:pf\_percent \* (:basic\_percent \* emp\_sal / 100) / 100, 2));**

**END;**

**//**



**4**. В этом задании вы будете работать с переменными подстановки, запрашивая их значения в run-time. Продолжайте модифицировать скрипт из 3 пункта.

a. Добавьте в скрипт инструкцию PROMPT с информационным сообщением «Укажите номер сотрудника».

b. Добавьте декларацию переменной empno так, чтобы она инициализировалась значением, полученным у пользователя.

c. Измените инструкцию SELECT так, чтобы при поиске информации о сотруднике использовать значение переменной empno.

d. Выполните скрипт и сохраните его под именем lab\_03\_04\_soln.sql.

e. Результат должен выглядеть примерно так (при empno = 100):



**DECLARE**

**fname VARCHAR2(15);**

**emp\_sal NUMBER(10);**

**BEGIN**

**SELECT first\_name, salary**

**INTO fname, emp\_sal**

**FROM emp**

**WHERE employee\_id = &emp\_id;**

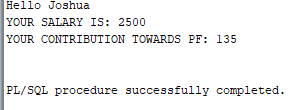
**DBMS\_OUTPUT.PUT\_LINE('Hello ' || fname);**

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

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

**ROUND(:pf\_percent \* (:basic\_percent \* emp\_sal / 100) / 100, 2));END;**

**/**



**5**. Выполните скрипт lab\_03\_05.sql (CREATE TABLE employee\_details AS SELECT \* FROM EMPLOYEES). Будет создана таблица EMPLOYEE\_DETAILS.

a. Таблицы employees и employee\_details имеют одинаковую структуру и данные. Все дальнейшие модификации выполняйте над данными из таблицы employee\_details.

# Практическое занятие 4. Взаимодействие с сервером Oracle

* Раздел INTO для сохранения результатов команд SQL
* Явные и неявные курсоры
* Атрибуты неявных курсоров

**1**. Создайте скрипт для выборки максимального номера отдела, сохраните его в переменной max\_deptno, распечатайте полученное значение:

a. Декларируйте переменную max\_deptno типа NUMBER.

b. В исполняемой секции блока выберите максимальное значение номера отдела и сохраните его в переменной

c. Распечатайте значение переменной в конце исполняемой секции.

d. Сохраните результат в файле lab\_04\_01\_soln.sql

e. Результат должен выглядеть примерно так:



**DECLARE**

**max\_deptno employee\_details.department\_id%TYPE;**

**BEGIN**

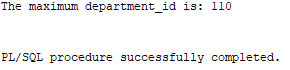
**SELECT MAX(department\_id) INTO max\_deptno**

**FROM employee\_details;**

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

**END;**

**/**



**2**. Измените скрипт из упражнения 1.

a. В скрипте lab\_04\_01\_soln.sql декларируйте две переменные: dept\_name такого же типа, что и столбец departments.department\_name и связанную переменную dept\_id типа NUMBER.

b. Присвойте переменной dept\_name значение «Education» в секции декларации.

c. Прибавьте к полученному максимальному значению номера отдела 10 и присвойте полученную цифру в переменную dept\_id.

d. Воспользуйтесь значениями переменных dept\_name и dept\_id в инструкции INSERT в таблицу departments. Для заполнения столбца location\_id используйте NULL значение.

e. С помощью атрибута SQL%ROWCOUNT узнайте количество строк обработанных инструкцией INSERT.

f. С помощью SELECT инструкции убедитесь, что строчка добавлена.

g. Сохраните скрипт в файле lab\_04\_02\_soln.sql.

**3**. С помощью инструкции UPDATE измените значение location\_id, для вновь добавленного отдела на 3000. Продолжите работу с предыдущим скриптом.

a. В исполняемой секции блока выполните инструкцию UPDATE.

b. Выведите новое состояние записи с помощью инструкции SELECT.

c. После проверки удалите добавленную строку.

d. Сохраните скрипт под именем lab\_04\_03\_soln.sql.

**VARIABLE dept\_id NUMBER**

**DECLARE**

**max\_deptno employee\_details.department\_id%TYPE;**

**dept\_name departments.department\_name%TYPE := 'Education';**

**BEGIN**

**SAVEPOINT before\_update;**

**SELECT MAX(department\_id) INTO max\_deptno**

**FROM departments;**

**:dept\_id := max\_deptno + 10;**

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

**UPDATE departments**

**SET location\_id = 3000**

**WHERE department\_id = max\_deptno;**

**DBMS\_OUTPUT.PUT\_LINE(SQL%ROWCOUNT);**

**-- ROLLBACK to before\_update;**

**END;**

**/**

# Практическое занятие 5. Управляющие конструкции

* Виды структур управления
* Значение NULL в условии IF
* Обработка значений NULL
* Виды циклов в PL\SQL
* Вложенные циклы и метки
* Команда CONTINUE

**1**. Выполните скрипт lab\_05\_01.sql для создания таблицы messages . Напишите блок кода для заполнения таблицы:

a. Добавьте числа в интервале от 1 до 10, пропустив 6 и 8. По одному числу на строку.

b. Добавьте инструкцию COMMIT в конце блока.

c. Проверьте результат с помощью инструкции SELECT.

**DROP TABLE messages;**

**CREATE TABLE messages (results VARCHAR2(80));**

**DECLARE**

**i PLS\_INTEGER;**

**BEGIN**

**FOR i IN 1 .. 10**

**LOOP**

**IF i = 6 OR i = 8**

**THEN continue;**

**ELSE**

**INSERT INTO messages**

**VALUES (i);**

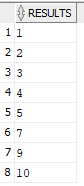
**END IF;**

**END LOOP;**

**COMMIT;**

**END;**

**/**



**2**. Выполните скрипт lab\_05\_02.sql. Этот скрипт создает реплику таблицы employees с именем emp и добавляет в нее новый столбец stars типа VARCHR2(50). Вам надо создать PL\SQL блок, который добавляет по одной «\*» за каждую 1000$ в зарплате сотрудника.

a. С помощью инструкции DEFINE определите переменную empno и присвойте ей значение 176.

b. В начале исполняемого блока инициализируйте empno с помощью переменной подстановки.

c. Декларируйте переменную asteriks типа emp.stars и присвойте ей значение NULL; Декларируйте переменную sal типа emp.salary.

d. В исполняемой секции напишите блок кода, с помощью которого переменная asteriks получает по одной звездочке за каждую $1000 в зарплате сотрудника.

e. Обновите столбец stars сотрудника с помощью полученного значения строчной переменной asteriks.

f. Добавьте в конец исполняемого блока инструкцию COMMIT.

g. С помощью инструкции SELECT выведите из таблицы измененную строчку.

h. Сохраните скрипт в файле lab\_05\_02\_soln.sql

i. Результат должен выглядеть примерно так:



**DEFINE empno = 176;**

**DECLARE**

**asteriks emp2.stars%TYPE := NULL;**

**sal emp2.salary%TYPE;**

**BEGIN**

**SELECT salary INTO sal**

**FROM emp2**

**WHERE employee\_id = '&empno';**

**-- FOR i IN 1 .. TRUNC(sal / 1000)**

**-- LOOP**

**-- asteriks := concat (asteriks, '\*');**

**-- END LOOP;**

**asteriks := RPAD(' ', TRUNC(sal / 1000), '\*');**

**UPDATE emp2**

**SET stars = asteriks**

**WHERE employee\_id = '&empno';**

**dbms\_output.put\_line(SQL%ROWCOUNT);**

**END;**

**/**

select employee\_id, salary, stars from emp2

where employee\_id = 176;



# Практическое занятие 6. Работа с составными типами данных

* Композитные типы данных
* Записи PL\SQL
* Коллекции PL\SQL
* Методы коллекций

**1**. Напишите скрипт для распечатки информации о выбранной стране:

a. Декларируйте переменную country\_record типа countries%ROWTYPE.

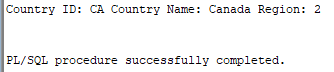
b. С помощью директивы DEFINE определите переменную countryid и присвойте ей значение CA по умолчанию. В исполняемой секции кода присвойте ей значение через переменную подстановки.

c. Извлеките информацию из таблицы countries для страны с заданным значением countryid в переменную, и выведите данные с помощью переменной country\_record.

d. Результат должен выглядеть примерно так:



1. Протестируйте работу скрипта на странах с идентификаторами DE, UK, US.



**DEFINE countryid = 'CA'**

**DECLARE**

**country\_record countries%ROWTYPE;**

**BEGIN**

**SELECT \* INTO country\_record**

**FROM countries**

**WHERE country\_id = '&countryid';**

**dbms\_output.put\_line ('Country ID: ' || country\_record.country\_id ||**

**' Country Name: ' || country\_record.country\_name ||**

**' Region: ' || country\_record.region\_id);**

**NULL;**

**END;**

**/**

**2**. Напишите блок кода для выборки имени департамента из специальной структуры хранения INDEX BY … TABLE. Сохраните скрипт как lab\_06\_02\_soln.sql.

a. Декларируйте тип INDEX BY TABLE с именем dept\_table\_type предназначенный для хранения значений типа departments.department\_name%TYPE. Декларируйте переменную my\_dept\_table типа dept\_table\_type.

b. Декларируйте две переменные loop\_count и deptno типа NUMBER. Присвойте им значения по умолчанию 10 и 0 соответственно.

c. С помощью цикла, получите из таблицы departments имена 10 отделов и сохраните их в my\_dept\_table. Начните с dept\_id равного 10. Увеличивайте счетчик и номер отдела соответственно с шагом в 10 единиц. Номера и названия отделов, которые необходимо скопировать в индексированную таблицу, показаны на рисунке:



d. С помощью другого цикла, извлеките значения из индексированной таблицы и распечатайте их.

e. Сохраните скрипт под именем lab\_06\_02\_soln.sql. Результат работы показан на рисунке:



**DECLARE**

**TYPE dept\_table\_type IS TABLE OF**

**departments.department\_name%TYPE**

**INDEX BY PLS\_INTEGER;**

**my\_dept\_table dept\_table\_type;**

**loop\_count NUMBER DEFAULT 10;**

**deptno NUMBER DEFAULT 0;**

**BEGIN**

**FOR i IN 1 .. loop\_count**

**LOOP**

**deptno := i \* loop\_count;**

**SELECT departments.department\_name INTO my\_dept\_table(deptno)**

**FROM departments**

**WHERE department\_id = deptno;**

**END LOOP;**

**FOR k IN my\_dept\_table.FIRST .. my\_dept\_table.LAST**

**LOOP**

**IF MOD(k,loop\_count) != 0 THEN continue;**

**ELSE**

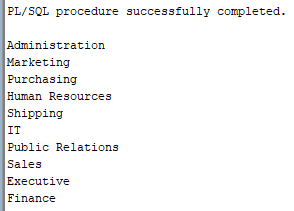
**dbms\_output.put\_line(my\_dept\_table(k));**

**END IF;**

**END LOOP;**

**END;**

**/**



**3**. Измените скрипт из второго пункта, таким образом, чтобы в индексированной таблице хранились записи (record), содержащие всю информацию об отделах.

a. Продолжайте работать с предыдущим скриптом.

b. Измените декларирование типа dept\_table\_type так, чтобы теперь в индексированной таблице хранились элементы типа departments%ROWTYPE.

c. Измените инструкцию SELECT таким образом, чтобы извлекать все колонки из таблицы departments для каждой записи.

d. Измените второй цикл, чтобы вывести все данные сохраненные в индексированной таблице.

e. Сохраните скрипт под новым именем. Результат должен выглядеть примерно так:



**DECLARE**

**TYPE dept\_table\_type IS TABLE OF**

**departments%ROWTYPE**

**INDEX BY PLS\_INTEGER;**

**my\_dept\_table dept\_table\_type;**

**loop\_count NUMBER DEFAULT 10;**

**deptno NUMBER DEFAULT 0;**

**BEGIN**

**FOR dept\_rec IN (SELECT \* FROM departments)**

**LOOP**

**deptno := deptno + 10;**

**my\_dept\_table(deptno) := dept\_rec;**

**END LOOP;**

**FOR k IN my\_dept\_table.FIRST .. my\_dept\_table.LAST**

**LOOP**

**IF MOD(k,loop\_count) != 0 THEN continue;**

**ELSE**

**dbms\_output.put\_line('Dept num: ' || my\_dept\_table(k).department\_id ||**

**' Dept name: ' || my\_dept\_table(k).department\_name ||**

**' Manager ID: ' || my\_dept\_table(k).manager\_id ||**

**' Loc ID: ' || my\_dept\_table(k).location\_id);**

**END IF;**

**END LOOP;**

**END;**

**/**

Dept num: 10 Dept name: Administration Manager ID: 200 Loc ID: 1700

Dept num: 20 Dept name: Marketing Manager ID: 201 Loc ID: 1800

Dept num: 30 Dept name: Purchasing Manager ID: 114 Loc ID: 1700

Dept num: 40 Dept name: Human Resources Manager ID: 203 Loc ID: 2400

Dept num: 50 Dept name: Shipping Manager ID: 121 Loc ID: 1500

Dept num: 60 Dept name: IT Manager ID: 103 Loc ID: 1400

Dept num: 70 Dept name: Public Relations Manager ID: 204 Loc ID: 2700

Dept num: 80 Dept name: Sales Manager ID: 145 Loc ID: 2500

Dept num: 90 Dept name: Executive Manager ID: 100 Loc ID: 1700

Dept num: 100 Dept name: Finance Manager ID: 108 Loc ID: 1700

Dept num: 110 Dept name: Accounting Manager ID: 205 Loc ID: 1700

Dept num: 120 Dept name: Treasury Manager ID: Loc ID: 1700

Dept num: 130 Dept name: Corporate Tax Manager ID: Loc ID: 1700

Dept num: 140 Dept name: Control And Credit Manager ID: Loc ID: 1700

Dept num: 150 Dept name: Shareholder Services Manager ID: Loc ID: 1700

Dept num: 160 Dept name: Benefits Manager ID: Loc ID: 1700

Dept num: 170 Dept name: Manufacturing Manager ID: Loc ID: 1700

Dept num: 180 Dept name: Construction Manager ID: Loc ID: 1700

Dept num: 190 Dept name: Contracting Manager ID: Loc ID: 1700

Dept num: 200 Dept name: Operations Manager ID: Loc ID: 1700

Dept num: 210 Dept name: IT Support Manager ID: Loc ID: 1700

Dept num: 220 Dept name: NOC Manager ID: Loc ID: 1700

Dept num: 230 Dept name: IT Helpdesk Manager ID: Loc ID: 1700

Dept num: 240 Dept name: Government Sales Manager ID: Loc ID: 1700

Dept num: 250 Dept name: Retail Sales Manager ID: Loc ID: 1700

Dept num: 260 Dept name: Recruiting Manager ID: Loc ID: 1700

Dept num: 270 Dept name: Payroll Manager ID: Loc ID: 1700

PL/SQL procedure successfully completed.

# Практическое занятие 7. Использование явных курсоров

* Явные курсоры
* Атрибуты явных курсоров
* Разделы FOR UPDATE и CURRENT OF

**1**. Напишите PL\SQL блок для определения n самых высоких зарплат сотрудников.

a. Выполните скрипт lab\_07\_01.sql для создания новой таблицы top\_salaries, предназначенной для хранения зарплаты сотрудников.

b. Запросите у пользователя значение переменной n. Пользователи могут захотеть увидеть зарплату 5-ти, 10-ти и так далее самых высокооплачиваемых сотрудников. С помощью инструкции DEFINE заранее определите переменную p\_num для хранения значения n.

c. Декларируйте переменную num типа NUMBER для хранения значения p\_num, переменную sal типа employees.salary, курсор emp\_cursor, который выбирает зарплату сотрудников в порядке убывания. Помните, что повторов значений быть не должно.

d. В исполняемой секции кода откройте цикл и выберите первые n записей из курсора и добавьте их в таблицу top\_salaries. Не забывайте про атрибуты %ROWCOUNT и %FOUND.

e. После добавления значений в таблицу просмотрите их с помощью инструкции SELECT.

Пример результата:



1. Протестируйте работу скрипта на различных значениях n: 0, 5, 10, 100, 1000. Не забывайте очищать таблицу top\_salaries после каждого теста.

**DEFINE v\_num**

**DECLARE**

**num NUMBER;**

**CURSOR emp\_cursor IS**

**SELECT DISTINCT salary FROM emp**

**ORDER BY 1 desc**

**FETCH FIRST '&&v\_num' ROWS ONLY;**

**sal emp\_cursor%ROWTYPE;**

**BEGIN**

**-- :v\_num := &v\_num;**

**open emp\_cursor;**

**-- FOR i IN 1 .. '&v\_num'**

**LOOP**

**FETCH emp\_cursor INTO sal;**

**EXIT WHEN emp\_cursor%NOTFOUND;**

**INSERT INTO top\_salaries**

**VALUES (sal.salary);**

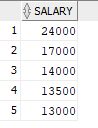
**dbms\_output.put\_line(sal.salary);**

**END LOOP;**

**close emp\_cursor;**

**END;**

**/**



**2**. Создайте PL\SQL блок, который делает следующее:

a. С помощью директивы DEFINE определите переменную p\_deptno для хранения department\_id значения.

b. Декларируйте переменную deptno типа NUMBER и присвойте ей значение p\_deptno.

c. Декларируйте курсор emp\_cursor для выборки last\_name, salary и manager\_id сотрудников, работающих в отделе с указанным значением deptno.

d. В исполняемой секции с помощью инструкции FOR и цикла пройдитесь по курсору и если salary меньше 5000 и manager\_id 101 или 124 отобразите сообщение “<<last\_name>> Due for a raise” иначе покажите сообщение “<<last\_name>> Not due for a raise”.

e. Примерный результат:



**DEFINE p\_deptno**

**DECLARE**

**deptno NUMBER := '&p\_deptno';**

**CURSOR emp\_cursor IS**

**SELECT**

**last\_name**

**, salary**

**, manager\_id**

**FROM emp**

**WHERE**

**department\_id = deptno;**

**BEGIN**

**FOR emp\_rec IN emp\_cursor**

**LOOP**

**IF emp\_rec.salary < 5000 AND (emp\_rec.manager\_id = 101 OR emp\_rec.manager\_id = 124) THEN**

**dbms\_output.put\_line (emp\_rec.last\_name || ' Due for a raise');**

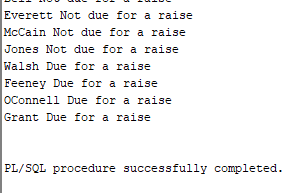
**ELSE dbms\_output.put\_line (emp\_rec.last\_name || ' Not due for a raise');**

**END IF;**

**END LOOP;**

**END;**

**/**



**3**. Напишите PL\SQL для работы с параметризованным курсором.

a. Декларируйте курсор dept\_cursor для извлечения department\_id, department\_name для отделов с номером меньше 100. Отсортируйте выборку по возрастанию номеров отделов.

b. Декларируйте другой курсор emp\_cursor, который принимает deptartment\_id на вход в качестве параметра и извлекает last\_name, job\_id, hire\_date и salary для тех сотрудников, у которых employee\_id меньше 120 и работающих в соответствующем отделе.

c. Декларируйте переменные, для хранения значений, извлеченных из каждого курсора. Воспользуйтесь директивой %TYPE.

d. Откройте dept\_cursor и с помощью простого цикла и инструкции fetch выберите значения в соответствующую переменную. Распечатайте имя и номер отдела.

e. Для каждого отдела отройте emp\_cursor. С помощью вложенного цикла пробегитесь по нему, и извлеките значения в соответствующую переменную и распечатайте их.

f. Закройте все циклы и курсоры в правильной последовательности. Выполните скрипт. Результат должен быть похож на рисунок:



**DECLARE**

**CURSOR dept\_cursor IS**

**SELECT department\_id, department\_name**

**FROM departments**

**WHERE department\_id < 100**

**ORDER BY 1;**

**CURSOR emp\_cursor (dep NUMBER) IS**

**SELECT**

**last\_name**

**, job\_id**

**, hire\_date**

**, salary**

**FROM emp**

**WHERE**

**employee\_id < 120**

**AND**

**department\_id = dep;**

**dept\_rec dept\_cursor%ROWTYPE;**

**emp\_rec emp\_cursor%ROWTYPE;**

**BEGIN**

**OPEN dept\_cursor;**

**LOOP**

**FETCH dept\_cursor INTO dept\_rec;**

**EXIT WHEN dept\_cursor%NOTFOUND;**

**dbms\_output.put\_line('Department number: ' ||**

**dept\_rec.department\_id || ' ' ||**

**'Department name: ' || dept\_rec.department\_name);**

**OPEN emp\_cursor(dept\_rec.department\_id);**

**LOOP**

**FETCH emp\_cursor INTO emp\_rec;**

**EXIT WHEN emp\_cursor%NOTFOUND;**

**dbms\_output.put\_line( emp\_rec.last\_name || ' ' ||**

**emp\_rec.job\_id || ' ' ||**

**to\_char(emp\_rec.hire\_date**

**, 'DD-MON-YYYY'**

**, 'nls\_date\_language=american') || ' '**

**|| emp\_rec.salary);**

**END LOOP;**

**CLOSE emp\_cursor;**

**dbms\_output.put\_line('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_');**

**END LOOP;**

**CLOSE dept\_cursor;**

**END;**

**/**

# Практическое занятие 8. Обработка исключений

* Предопределенные исключения Oracle
* Функция PRAGMA\_EXCEPTION\_INIT
* Функции SQLCODE и SQLERRM
* Распространение исключений в подблоках
* Процедура RAISE\_APPLICATION\_ERROR
* Команда RAISE

**1**. Основная задача этого упражнения – продемонстрировать использование предопределенных исключений. Напишите PL\SQL блок для выборки сотрудников с определенной зарплатой.

a. Удалите все записи из таблицы messages. С помощью директивы DEFINE определите переменную sal и инициализируйте ее значением 6000.

b. В секции декларации исполняемого блока определите переменную ename типа employees.last\_name%TYPE и переменную emp\_sal типа employees.salary%TYPE и инициализируйте ее с помощью подстановочной переменной &sal.

c. В исполняемой секции выберите фамилии тех сотрудников, чьи зарплаты равны значению переменной emp\_sal. **Не используйте явных курсоров!** Если выборка вернет одну строку – добавьте в таблицу messages имя и зарплату сотрудника.

d. Если для заданного значения зарплаты не найдено соответствующих записей или найдено несколько записей, обработайте соответствующие исключительные ситуации, и добавьте в таблицу messages записи «No employee with a salary of <salary>» или «More than one employee with a salary of <salary>».

e. Просмотрите таблицу messages, чтобы оценить успешно или нет отработал PL\SQL блок.

**DEFINE sal = 6000;**

**DECLARE**

**ename emp.last\_name%TYPE;**

**emp\_sal emp.salary%TYPE := &sal;**

**BEGIN**

**SELECT last\_name INTO ename**

**from emp where salary = emp\_sal;**

**dbms\_output.put\_line(ename);**

**insert into messages**

**values (ename || emp\_sal);**

**EXCEPTION WHEN NO\_DATA\_FOUND**

**THEN**

**dbms\_output.put\_line ('No employee with a salary of ' ||**

**emp\_sal);**

**WHEN TOO\_MANY\_ROWS**

**THEN**

**dbms\_output.put\_line ('More than one employee with a salary of ' ||**

**emp\_sal);**

**WHEN OTHERS**

**THEN**

**dbms\_output.put\_line ('Oops something bad happens.');**

**END;**

**/**

**2**. В этом упражнении вы посмотрите, как создавать именованные исключения для стандартных системных ошибок Oracle сервера ORA-02292 (нарушение ограничений целостности).

a. В декларативной секции объявите childrecord\_exists типа EXCEPTION. Проассоциируйте эту переменную со стандартной ошибкой Oracle -02292. Воспользуйтесь инструкцией PRAGMA EXCEPTION\_INIT.

b. В исполняемой секции блока напечатайте сообщение «Deleting department 40........» и выполните инструкцию DELETE для удаления отдела с department\_id равным 40.

c. Добавьте секцию перехвата и обработки ошибок и поймайте в ней childrecord\_exists. При обработке ошибке выведите соответствующее сообщение («Cannot delete this department. There are employees in this department»).

**DECLARE**

**childrecord\_exists EXCEPTION;**

**PRAGMA EXCEPTION\_INIT (childrecord\_exists,-02292);**

**BEGIN**

**dbms\_output.put\_line('Deleting department 40......');**

**delete from dept**

**where department\_id = 40;**

**EXCEPTION**

**WHEN childrecord\_exists**

**THEN**

**dbms\_output.put\_line('Cannot delete this department.**

**There are employees in this department');**

**WHEN OTHERS**

**THEN**

**NULL;**

**END;**

**/**