**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.

# 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!”

**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.

# 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)

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

c. Значение переменной weight в позиции (2)

d. Значение переменной message в позиции (2)

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

**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:
2. Значение и тип данных переменной credit\_rating во вложенном блоке:
3. Значение и тип данных переменной customer в основном (внешнем) блоке:
4. Значение и тип данных переменной name в основном блоке:

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

**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:



**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):



**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:



**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.

# 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.

**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:



# 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:



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

**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:



**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:



# 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:



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

**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:



**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:



# 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.

**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»).