Лабораторные работы по курсу
«СУБД ORACLE»

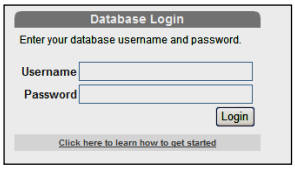
Содержание:

Лабораторная работа № 1. Работа в Oracle Database Express Edition. Создание объектов базы данных в Oracle 1
Лабораторная работа № 2. Создание пользовательских приложений. Управление правами доступа и разрешениями. 4
Лабораторная работа № 3. Освоение программирования с помощью встроенного языка pl/sql в Oracle. Утилита SQLplus. 6
Лабораторная работа № 4. Создание хранимых процедур и функций. 6
Лабораторная работа № 5. Создание триггеров. 9
Лабораторная работа № 6. Разработка web-приложения 12

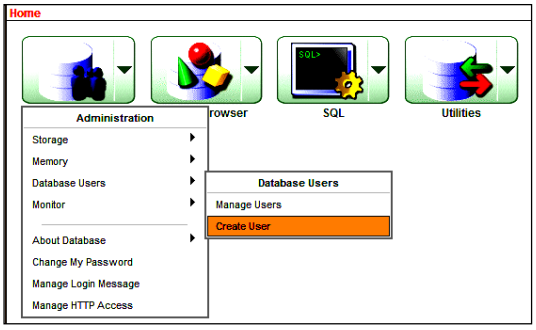
Лабораторная работа № 1. Работа в Oracle Database Express Edition. Создание объектов базы данных в Oracle

Введение. Пакет Oracle Database Express Edition (Oracle Database XE) является свободно распространяемой версией СУБД Oracle. Работа с СУБД выполняется с помощью интуитивно понятного WEB-интерфейса браузера. С помощью этого интерфейса можно выполнять все основные операции по созданию таблиц баз данных, установлению связей между таблицами, вводу данных, созданию запросов, отчетов, администрированию пользователей.
Рассмотрим основные правила работы с этим пакетом.

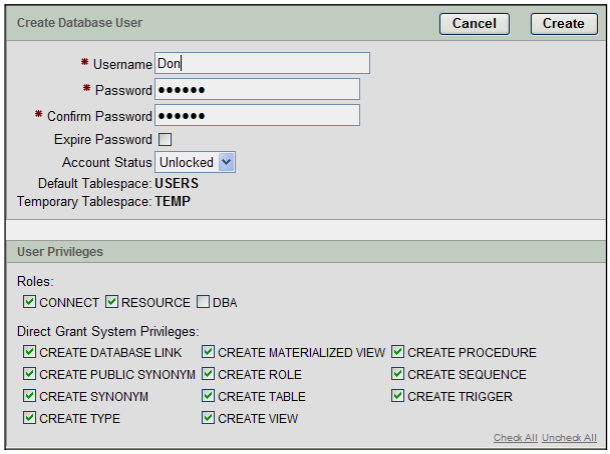
1. Запускаем дистрибутив пакета - файл **OracleXE.exe**. Указываем место для установки и соглашаемся с остальными опциями. На одной из страниц установки потребуется указать пароль, с которым будем осуществляться доступ к серверу СУБД. Укажем в качестве пароля слово **oracle**.
2. После установки в меню **Пуск\Программы** появится раздел **Oracle Database Express Edition**, содержащий подразделы
   Get Help - помощь
   Backup Database – резервирование БД
   Get Started – вызов справки по Oracle Express
   Go To Database Home Page – домашняя страница
   Run SQL Command Line – работа с БД с помощью команд SQL
   Start Database – запуск сервера
   Stop Database – остановка сервера
3. Идем к домашней странице **Go To Database Home Page**. При этом открывается окно,



4. Первоначально на сервере создан один пользователь **system** с паролем, указанный при установке (**oracle**), поэтому первоначально надо зайти с этими логином и паролем. Для проверки Имени пользователя кликните по ссылке: «**Click here to learm how to get started**». В пункте 2 будет указан **Username,** который надо ввести**.**
5. Далее нажмем на кнопку **Administration**, выберем раздел **Database Users → Create Users:**



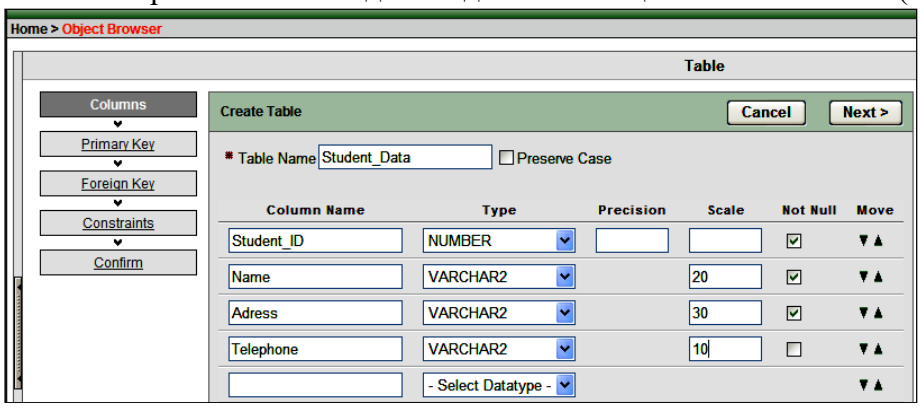
6. Создадим нового пользователя с вашим именем и паролем, например:



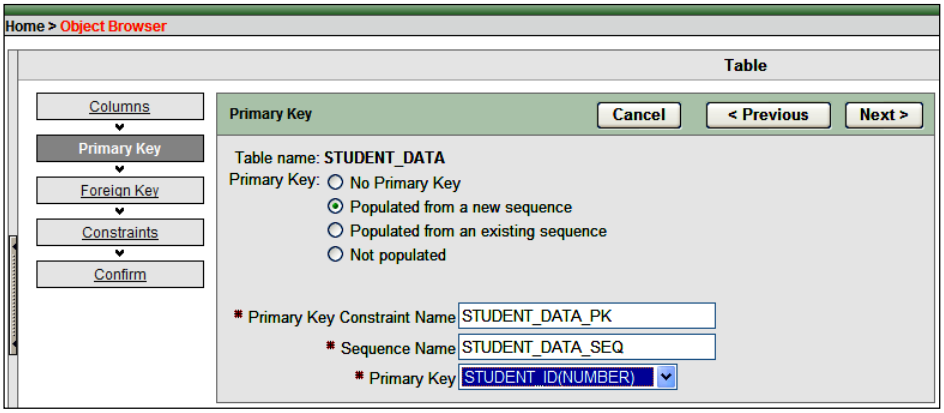
7. После заполнения формы нажимаем на кнопку **Create** (Создать). Далее заканчиваем сеанс, нажав ссылку **Logout** в правом верхнем углу экрана.

8. Выбираем 

. Далее нажимаем **Create → Table.**
9. Открывается меню для создания таблицы. Заполним поля (**Columns**) таблицы:



10. Далее нажимаем **Next**. Открывается форма для создания Ключа (**Primary Key**): Выбираем **Populated from a new sequence**, задаем ключевое поле (в нашем случае **Student_ID(Number**):



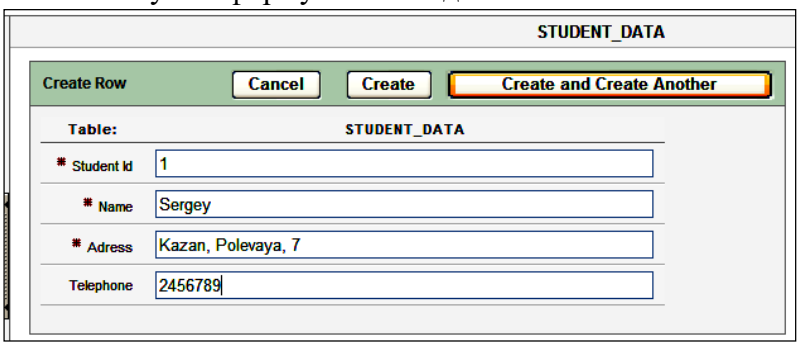
11. Нажимаем кнопку **Next.** Открывается форма для задания внешнего ключа **(Foreing Key).** Если внешний ключ не задается, нажимаем **Далее**.
12. Открывается форма для создания Ограничений (**Constraints)**. При отсутствии ограничений нажимаем **Finish.**
13. Следующая форма сообщает о том, что пользователем создана таблица. Нажимаем кнопку **Create.**
14. Открывается диалоговое окно построителя таблицы**: STUDENT_DATA.** В левой части приводится список всех созданных таблиц. В правой части: мастер создания и изменения таблицы:

- Добавить столбец,
- Изменить столбец,
- Переименовать столбец,

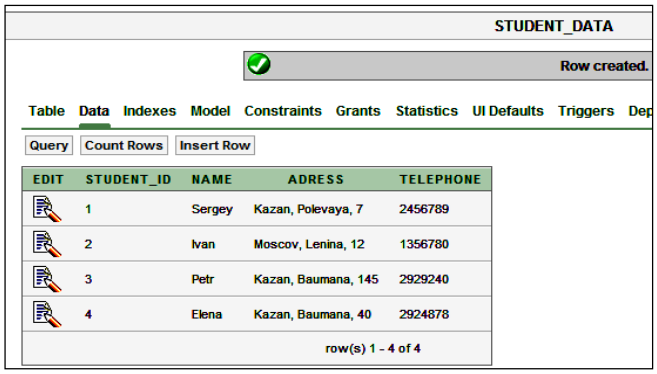
- Удалить столбец,
- Переименовать таблицу,
- Копировать таблицу,
- Удалить таблицу.

Чтобы заполнить таблицу, выбираем вкладку **Data,** кнопку **Insert Row.**

В появившуюся форму заносим данные:



Далее нажимаем **Create**, затем кнопку **Insert Row.** Заполняем данные на следующего студента. В результате заполнения полей таблицы появляется список всех студентов:



15. Создать таблицу **SUBJECT:**

Column Name	Data Type	Nullable	Default	Primary Key
SUBJECT_ID	NUMBER	No	-	-
SUBJECT_NAME	VARCHAR2(25)	Yes	-	-
				1 - 2

Внести в таблицу следующие данные:

EDIT	SUBJECT_ID	SUBJECT_NAME
	1	BD
	2	Math
	3	Inform
		row(s) 1 - 3 of 3

16. Создать таблицу **EXAMS:**

Column Name	Data Type	Nullable	Default	Primary Key
EXAM_ID	NUMBER	No	-	1
STUD_NAME	VARCHAR2(20)	No	-	-
EXAM_NAME	VARCHAR2(20)	No	-	-
MARK	NUMBER	Yes	-	-
				1 - 4

Занести в таблицу следующие данные:

EDIT	EXAM_ID	STUD_NAME	EXAM_NAME	MARK
	1	Sergey	BD	5
	2	Sergey	Math	3
	3	Sergey	Inform	-
	4	Ivan	BD	3
	5	Ivan	Math	4
			row(s) 1 - 5 of 5	

Лабораторная работа № 2. Создание пользовательских приложений. Управление правами доступа и разрешениями.

Пользовательские приложения в Oracle представляют собой отчеты и формы, выполненные как Web-страницы и позволяющие получать информацию из Базы данных.

Рассмотрим пример создания приложения формы отчета по таблицам, входящим в учебную базу данных пользователя с идентификатором HR.

1. Разблокировка пробной учетной записи пользователя

Чтобы создать свое приложение, Вы должны войти как пользователь базы данных. Oracle Database XE поставляется с экспериментальным пользователем базы данных, именуемым **HR**. Этот пользователь владеет несколькими таблицами базы данных в пробной схеме, которая может быть использована при создании приложений для вымышленного подразделения Human Resources. Однако, из соображений безопасности, учетная запись этого пользователя заблокирована. Вы должны разблокировать эту учетную запись, прежде чем сможете создать свое пробное приложение.

Чтобы разблокировать пробную учетную запись пользователя:

1. Убедитесь, что Вы все еще подключены как администратор базы данных, как это описано в предыдущем разделе.
2. Щелкните на иконке **Administration**, а затем щелкните **Database Users**.
3. Щелкните на иконку схемы **HR**, чтобы отобразить пользовательскую информацию для **HR.**

4. В Manage Database User введите следующие настройки:
   - **Password** и **Confirm Password**: Введите **hr** в качестве пароля.
   - **Account Status**: Выберите **Unlocked**.
   - **Roles**: Убедитесь, что активированы как **CONNECT**, так и **RESOURCE**.
5. Щелкните **Alter User**.

Теперь все готово для создания приложения.

2. Подключение к пробной учетной записи

Для подключения к пробной учетной записи:

1. Закончите работу с учетной записью администратора базы данных, нажав **Logout** в верхнем правом углу домашней страницы базы данных.
2. Нажмите **Login**.
3. В окне подключения введите **hr** в качестве имени пользователя и пароля.
4. Нажмите **Login**.
   Появится домашняя страница базы данных.

3. Создание простого приложения

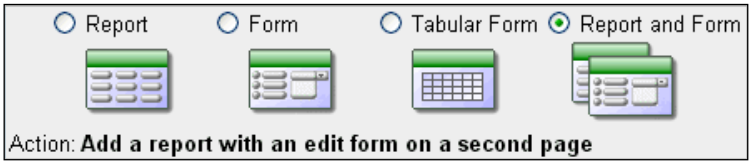
Создание приложения это самый простой способ просматривать и редактировать данные в Вашей базе данных. Вы создадите это приложение на основе таблицы **EMPLOYEES**, являющейся частью схемы **HR**.

Чтобы создать приложение, основанное на таблице **EMPLOYEES**:

1. На домашней странице базы данных щелкните иконку **Application Builder**.
2. Нажмите кнопку **Create**.
3. На открывшейся странице выберите **Create Application** и нажмите **Next**.

4. В поле Create Application введите следующие параметры:
   a. В поле Name введите **MyApp.**
   b. Остальные поля оставьте без изменений.
   c. Нажмите **Next**.

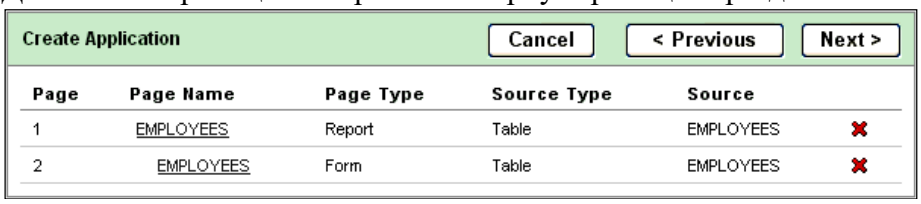
   Далее, добавьте страницы к вашему приложению.
5. Под разделом Add Page:
   a. В опции Select Page Type выберите **Report and Form**.



Обратите внимание, что в поле **Action** отображается тип страницы, которую вы добавляете.

   b. В поле Table or View выберите **EMPLOYEES**.
   c. Нажмите кнопку **Add Page**.

   Две новые страницы отобразятся вверху страницы в разделе Create Application.



Page	Page Name	Page Type	Source Type	Source	
1	EMPLOYEES	Report	Table	EMPLOYEES	✖
2	EMPLOYEES	Form	Table	EMPLOYEES	✖

   d. Click **Next**.
6. Опцию Tabs оставьте без изменений (**One Level of Tabs**) и нажмите **Next**.
7. Опцию Shared Components оставьте без изменений и нажмите **Next**.

   Эта опция позволит вам импортировать общие компоненты из других приложений. Общие компоненты - это стандартные элементы, которые могут быть отображены или применены на любой странице приложения.
8. Параметры полей Authentication Scheme, Language и User Language Preference Derived From оставьте без изменений и нажмите **Next**.
9. В опциях **User** Interface выберите **Theme 2** и нажмите **Next**.

   Темы это наборы шаблонов, которые можно использовать для задания расположения элементов и определения внешнего вида всего приложения.
10. Подтвердите сделанные изменения. Чтобы вернуться на предыдущую страницу мастера, нажмите **Previous**. Чтобы принять изменения, нажмите **Create**.

    После того как вы нажмёте **Create**, вверху страницы появится следующее сообщение:
    Application created successfully.

4. Запуск нового приложения

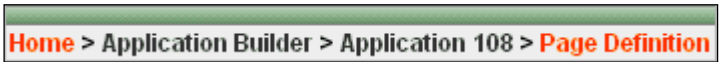
Чтобы запустить Ваше приложение:

1. Щелкните иконку **Run Application**.

2. На странице авторизации, введите **hr** как в поле **User Name**, так и в поле **Password**.

   Появится Ваше приложение, показывая таблицу **EMPLOYEES**.
3. Исследуйте Ваше приложение. При желании, Вы можете делать запросы к таблице **EMPLOYEES**. Для управления приложением, используйте инструментальное меню разработчика в нижней части страницы.

   Инструментальное меню разработчика дает возможность оперативно отредактировать текущую страницу, создать новую страницу, элемент управления или компонент, посмотреть состояние сессии, а также включить/отключить режим отладки или ссылки редактирования.
4. Для выхода из приложения и возврата в Application Builder, щелкните **Edit Page 1** в инструментальном меню разработчика.
5. Для возврата на домашнюю страницу базы данных выберите пункт **Home** в верхней части страницы.



Поздравляем! Вы только что создали Ваше первое приложение, используя Oracle Database XE.

Лабораторная работа № 3. Освоение программирования с помощью встроенного языка pl/sql в Oracle. Утилита SQLplus.

Утилита SQLplus является клиентским приложением, которое может осуществить доступ к базе данных Oracle Express через локальную или глобальную сеть. Ее можно установить на компьютер клиента без установки сервера Oracle Express. При установке сервера она устанавливается автоматически.
Для работы с этой утилитой, добавим нового пользователя с именем **Stud**. Для этого выполним команды:

1. Запустите домашнюю страницу **Oracle Express**, выполняя команду **Пуск\Все программы\ Oracle Database Express Edition\Go To Database Home Page.**
2. В появившемся приглашении введите логин system и пароль oracle. В первой вкладке администрирования выберите раздел DateBase Users\Create Users и добавьте нового пользователя с именем **test**, пароль – **test**.
3. Выполните подсоединение к базе данных пользователя **test** с помощью **SQLplus**, для этого:
4. Откройте меню **Пуск** системы **Windows** и нажмите **«Выполнить».** В появившемся окне введите команду **cmd** для запуска командного интерпретатора **DOS**.
5. Наберите команду **SQLplus**, клавиша <Enter>.
6. Введите имя пользователя **test,** затем пароль **test**. Должно появиться приглашение **SQL>**
7. Создайте таблицу:
   - **Orders (Заказы)** с полями: **№ заказа, ФИО покупателя, Дата**.
     Create table Orders (
     ID Int not null primary key,
     FIO varchar2 (20) not null,
     Data date not null); клавиша <Enter>.

При работе в командной строке DOS предыдущие команды можно выбрать с помощью кнопок стрелочек вверх-вниз.

   - Добавьте новый столбец **Название товара**: **(Name_Tovar)**
     Alter table Orders add (Name_Tovar varchar2 (40) not null); клавиша <Enter>.
   - Добавьте строку данных.

     Insert into Orders values (
     1, ‘Ivanov’,’25.10.2008’,’Bred’);
   - Добавьте еще несколько строк данных.
   - Выполните просмотр данных из таблицы **Orders**
     SELECT * FROM Orders; клавиша <Enter>.
   - Выполните просмотр количества записей:
     SELECT count(*) FROM Orders; клавиша <Enter>.
8. Выполните команду отсоединения:
   Disconnect клавиша <Enter>.

Снова откройте домашнюю страницу, войдя под именем **test**, пароль – **test** и посмотрите созданную таблицу через раздел **Object Brower**.

Лабораторная работа № 4. Создание хранимых процедур и функций.

Подпрограммы (процедуры и функции) в PL/SQL бывают хранимые и локальные. Хранимые подпрограммы во многом отличаются от локальных.Если подпрограмма должна будет вызываться из многих блоков, то ее делают хранимой, она хранится в базе данных. Короткие процедуры и функции рекомендуется объявлять локально в блоке, если они вызываются только из одного конкретного фрагмента программы. Хранимые подпрограммы (процедуры и функции) хранятся в базе данных в откомпилированном виде и могут вызываться из любого блока.
В этом разделе будем рассматривать хранимые подпрограммы. *Хранимые процедуры и функции* — это объекты базы данных и, следовательно, они создаются командой **CREATE** и уничтожаются командой

DROP. При создании процедуры и функции должны быть определены: имя объекта, перечень и тип параметров и логика работы процедуры или функции, описанные на языке PL/SQL
Чтобы создать процедуру или функцию, необходимо иметь системные привилегии CREATE PROCEDURE. Для создания процедуры, функции или пакета в схеме, отличной от текущей схемы пользователя, требуются системные привилегии CREATE ANY PROCEDURE. После определения имени новой процедуры или функции необходимо задать ее имена, типы и виды параметров. Для каждого параметра должен быть указан один из видов параметра —
IN, OUT или IN OUT. Вид параметра IN предполагает, что значение параметра должно быть определено при обращении к процедуре и не изменяется процедурой.
Попытка изменить в теле процедуры параметр вида **IN** приведет к сообщению об ошибке. Вид параметра **OUT** предполагает изменение значения параметра в процессе работы процедуры, то есть параметр вида **OUT** — это возвращаемый параметр. Параметр **IN OUT** — это параметр, которому при вызове должно быть присвоено значение, которое может быть изменено в теле процедуры. Параметры процедур или функций имеют виды, присвоенные по умолчанию
Дополнительно к определениям, необходимым для процедуры, в определении функции должен быть указан тип данных единственного возвращаемого функцией значения. Возврат значения функции выполняется командой PL/SQL **RETURN**. Оператор определения процедуры ORACLE использует следующий синтаксис:

CREATE [OR REPLACE] PROCEDURE [схема.]имя_процедуры
[(имя_параметра[{Ш | **OUT** | **IN OUT**}] тип_данных
[,имя_параметра[{Ш | **OUT** | **IN OUT**}] тип_данных...])]
{IS | AS}

Ключевое слово **OR REPLACE** указывает на безусловное замещение старого текста процедуры. Если ключевое слово
OR REPLACE не указано, и процедура определена, то замещения старого значения кода процедуры не происходит, и возвращается сообщение об ошибке.
В описании переменных процедуры не используется ключевое слово DECLARE. Блок определения данных начинается сразу после ключевого слова IS (или AS, по выбору пользователя).
Рассмотрим пример создания процедуры, которая заносит в таблицу значения, зависящие от числового параметра. Пусть таблица сформирована предложением:

```
CREATE TABLE SUBJLECT(LECT_ID NUMBER, SUBJ_ID NUMBER);
```

Пример.

```
SQL>CREATE OR REPLACE PROCEDURE InsRec ( Argl IN NUMBER, Arg2 IN NUMBER ) IS
Coeff CONSTANT NUMBER:=0.5;
BEGIN
INSERT INTO SUBJLECT VALUES(Coeff*Argl, Coeff*Arg2);
END;
/
```

Исполнение созданной процедуры INSREC может быть выполнено оператором EXEC языка PL/SQL. Последующая выборка из таблицы SUBJLECT иллюстрирует изменения в базе данных, осуществленные вызовом процедуры InsRec.

```
SQL> EXEC InsRec (240,120);
```

Чтобы уточнить выявленные в процессе синтаксического анализа ошибки, можно воспользоваться командой PL/SQL SHOW ERRORS. Эта команда показывает ошибки, обнаруженные в процессе выполнения CREATE PROCEDURE, CREATE FUNCTION, CREATE PACKAGE, CREATE PACKAGE BODY, CREATE TRIGGER. Если команда SHOW ERRORS используется без параметров, то возвращаются ошибки последней компилированной процедуры, функции, пакета, тела пакета или триггера.
Рассмотрим пример обнаружения и исправления ошибки. В процедуре, представленной выше, добавим ошибочный оператор, изменяющий значение параметра вида IN, который не должен изменяться.

Пример.

```
SQL>CREATE OR REPLACE PROCEDURE InsRec (Argl IN NUMBER, Arg2 IN NUMBER) IS
2       Coeff CONSTANT NUMBER:=0.5;
3       BEGIN
```

```
4       Argl:=Argl+200;       -- ОШИБКА
5       NSERT INTO SUBJLECT VALUES(Coeff*Argl, Coeff*Arg2);
6       END;
```

Протокол выполнения:

```
SQL>SHOW ERRORS
Errors for PROCEDURE InsRec:
LINE/COL ERROR
4/1 PLS-00363: expression 'ArgL cannot be used as an assignment target 4/1 PL/SQL: Statement ignored
```

Напомним, что функции PL/SQL отличаются от процедур тем, что возвращают в вызывающую среду результат. Оператор определения функции ORACLE использует следующий синтаксис:

```
CREATE [OR REPLACE] FUNCTION [схема.]имя_функции [ (имя_параметра[{Ш | OUT | IN OUT}]
тип_данных [, имя_параметра[{Ш | OUT | IN OUT}] тип_данных...])]
RETURN тип_данных {IS | AS}
```

Описание типа данных для возвращаемого функцией значения требуется обязательно. При описании переменных функции так же, как и при описании переменных процедуры, не используется ключевое слово DECLARE. Блок определения данных начинается сразу после ключевого слова IS (или AS, по выбору пользователя).

Рассмотрим пример создания функции, которая вычисляет сумму значений атрибутов, таких, что один из атрибутов попадает в заданный параметрами функции интервал. Пусть таблица сформирована предложением как, в примере 2.3.25.

Пример.

```
SQL>CREATE OR REPLACE FUNCTION SumRecInt (Argl IN NUMBER, Arg2 IN NUMBER)
2       RETURN NUMBER
3       IS
4       Sum_Var NUMBER:=0.0;
5       BEGIN
7       SELECT Sum(LECT_ID) INTO Sum_Var FROM SUBJLECT
WHERE SUBJ_ID BETWEEN Argl AND Arg2;
8       RETURN Sum_Var;
9       END;
/
FUNCTION CREATED
```

Если характер использования приложений изменился, то для освобождения ресурсов базы данных может потребоваться уничтожить процедуру или функцию. В собственной схеме пользователю не требуются дополнительные привилегии для уничтожения процедуры или функции. Для уничтожения процедуры или функции в схеме другого пользователя необходимо наличие привилегии DROP ANY PROCEDURE.

Для уничтожения {процедуры | функции} ORACLE использует следующий синтаксис:

DROP {**PROCEDURE** | **FUNCTION**} [схема.]имя_процедуры

Например

```
SQL>DROP FUNCTION SumRecInt
Function dropped;
```

Задания

1. Описать процедуру, которая заносит данные в таблицу STUDENT в зависимости от параметров процедуры, вызвать эту процедуру.
2. Описать процедуру, которая заносит данные в таблицы STUDENT и EXAM_MARKS с соблюдением ссылочной целостности, в зависимости от параметров процедуры, вызвать эту процедуру.
3. Описать процедуру, которая заносит данные в таблицы STUDENT и SUBJECT с соблюдением ссылочной целостности, в зависимости от параметров процедуры, вызвать эту процедуру.
4. Описать процедуру, которая заносит данные в таблицы STUDENT и UNIVERSITY с соблюдением ссылочной целостности, в зависимости от параметров процедуры, вызвать эту процедуру.
5. Описать процедуру, которая заносит данные в таблицу LECTURE в зависимости от параметров процедуры, вызвать эту процедуру.

6. Описать и вызвать функцию, которая вычисляет сумму баллов для студентов, номера которых находятся в заданном интервале в таблице EXAM_MARKS.
7. Описать и вызвать функцию, которая вычисляет для студентов, номера которых находятся в данном диапазоне, максимальный балл по заданному предмету (с некоторым номером) в таблице EXAM_MARKS.
8. Описать и вызвать функцию, которая определяет для студентов с заданной фамилией сумму баллов по заданному предмету в таблице EXAM_MARKS.
9. Описать и вызвать функцию, которая вычисляет для студентов, номера которых находятся в данном диапазоне, максимальный балл по заданному названию предмета.
10. Описать и вызвать функцию, которая вычисляет для студентов, номера которых находятся в данном диапазоне, средний балл по заданному названию предмета.

Лабораторная работа № 5. Создание триггеров.

Триггер базы данных — это процедура PL/SQL, которая автоматически запускается при возникновении определенных событий, связанных с выполнением операций вставки, удаления или модификации данных таблицы. Событие, управляющее запуском триггера, описывается в виде логических условий. Когда возникает событие, соответствующее условиям триггера, сервер ORACLE автоматически запускает триггер, то есть интерпретирует код программы триггера, записанный на языке PL/SQL.

Обычно триггеры используют для выполнения сложных проверок ограничений целостности, многоаспектных проверок выполнения правил разграничения доступа и т.п.

Триггер запускается при выполнении одной из трех операций изменения содержимого таблицы: INSERT, DELETE или UPDATE.
Триггер может запускаться и несколькими операторами, но хотя бы один оператор из тройки должен быть обязательно указан в условии запуска триггера. Если перечень операторов, запускающих триггер, включает оператор UPDATE, то для условий срабатывания могут быть указаны конкретные изменяемые столбцы.

Код триггера может выполняться либо до, либо после тех операторов, которые инициировали запуск триггера. Например, если триггер запускается для проверки полномочий пользователя на право выполнения операции, то, конечно, нужно использовать триггер с запуском до выполнения операции (с ключевым словом BEFORE).
Если триггер применяется для формирования данных для аудиторской записи, то разумно использовать триггер с запуском после выполнения операции (с ключевым словом AFTER).

Код триггера может быть ассоциирован либо с операцией над таблицей в целом, либо с каждой строкой, над которой выполняется операция. В зависимости от этого триггеры подразделяют на операторные триггеры и строчные триггеры. Операторные триггеры обычно используют для проверки правил разграничения доступа, оперирующих таблицей в целом, а строчные триггеры часто используют для проверки ограничений целостности при вставке строк. Условие запуска строчного триггера может быть уточнено дополнительным логическим условием.
Чтобы создать триггер, необходимо иметь системные привилегии CREATE TRIGGER. Для создания триггера в схеме, отличной от текущей схемы пользователя, требуются системные привилегии CREATE ANY TRIGGER.

Синтаксис оператора определения триггера:
CREATE [**OR REPLACE**] **TRIGGER** [схема.]имя_триггера {**BEFORE | AFTER**}
{**INSERT** | **DELETE** | **UPDATE** [**OF** имя_столбца[,имя_столбца...]]}
[**OR** {**INSERT** | **DELETE** | **UPDATE** [**OF** имя_столбца[,имя_столбца...]]}...]
ON [схема.]{имя_таблицы | имя_представления}
[**FOR EACH ROW**] [**WHEN** условие] спецификации_пакета_на_PL/SQL

Ключевое слово OR REPLACE указывает на безусловное замещение старого текста триггера. Если ключевое слово OR REPLACE не указано, и триггер определен в системе, то замещения старого значения триггера не происходит, и озвращается сообщение об ошибке.

Ключевые слова BEFORE или AFTER указывают на выполнение кода триггера либо до, либо, соответственно, после операторов манипулирования данными, инициировавших запуск триггера.

Ключевые слова INSERT, DELETE или UPDATE определяют конкретный оператор, запускающий триггер. Для оператора UPDATE могут быть указаны столбцы, изменение которых запускает триггер.
Если конкретные столбцы не указаны, то триггер запускает изменение любого столбца. Необязательное ключевое слово OR присоединяет дополнительный оператор, запускающий триггер.

Ключевое слово ON задает имя таблицы или представления, ассоциированного с триггером.

Необязательное ключевое слово FOR EACH ROW определяет триггер как строчный.

Необязательное ключевое слово WHEN задает дополнительное логическое условие, сужающее область действия триггера.

Прежде чем перейти к примеру, построения триггера, приведем некоторые дополнительные сведения об обработке исключительных ситуаций в ORACLE. Процедура RAISE_APPLICATION_ERROR применяется для подключения к механизму обработки ошибок пользовательских точек входа. С ее помощью можно обработать до 1000 определяемых пользователем ошибок с номерами в диапазоне от20000 до -20999. Вызов процедуры RAISE_APPLICATION_ERROR приводит к генерации исключительной ситуации и завершению выполнения вызвавшей процедуру программы (сравните с рассмотренным выше оператором PL/SQL RAISE). При этом в среду, вызвавшую программу, возвращается номер и текстовое сообщение о типе ошибки.

Рассмотрим пример триггера, который выполняется, если значение вводимого атрибута "уклоняется по модулю" от среднего значения для текущего состояния таблицы больше, чем на 30. Пусть таблица Tab1 сформирована предложениями:

CREATE TABLE Tabl (Atl NUMBER);

INSERT INTO Tabl VALUES(10);

INSERT INTO Tabl VALUES(30);

INSERT INTO Tabl VALUES(50);

Протокол создания триггера представлен ниже. При срабатывании триггера предусмотрена генерация стандартной обработки ошибки, которой присваивается номер -20002 с соответствующим диагностирующим сообщением. Обратите внимание на предопределенную переменную :new.Atl, содержащую (по ее смыслу) вводимое значение атрибута Atl.

Пример.

SQL>**CREATE OR REPLACE TRIGGER** TRIG_TBI **BEFORE INSERT ON** Tabl **FOR EACH ROW**

DECLARE StatAvg NUMBER;

StatN NUMBER;

BEGIN

SELECT COUNT(Atl),SUM(Atl) INTO StatN, StatAvg

FROM Tabl;

IF (ABS(StatAvg-StatN* (:new.Atl))>3 0 * StatN) THEN RAISE_AP-

PLICATION_ERROR(-20002, 'Слишком большое уклонение');

END IF;

END;

/

Trigger created.

Работу механизма триггера проиллюстрируем на примере. При вводе значения, достаточно близкого к среднему (в данном случае 40), триггер не запускается, и "ничего не происходит". При вводе значения атрибута, равного 90, соответствующая статистика указывает на большое уклонение, происходит срабатывание триггера, и новая строка не включается. Операция выборки подтверждает ожидаемое изменение в таблице.

SQL>insert into tabl values(40);
1 row created.

SQL>insert into tabl values (90); insert into tabl values(90)

ERROR at line 1:

ORA-20002: Слишком большое уклонение
ORA-06512: at "SYSTEM. TRIG_TBI", line 9
ORA-04088: error during execution of trigger ‘SYSTEM .TRIG_TBI'

SQL>SELECT * FROM tabl;

Следующий пример иллюстрирует возможность обработки исключительной ситуации средствами пользовательской исключительной ситуации, В данном случае создается триггер, который при превышении заданного порога уклонения вводимоxx) значения атрибута выводит диагностическое сообщение. При этом данные в таблицу вводятся.

Пример.

```
SQL>CREATE OR REPLACE TRIGGER TRIG_TB2
BEFORE INSERT ON Tabl
FOR EACH ROW
DECLARE
StatAvg NUMBER;
StatN NUMBER;
Special_case EXCEPTION;  —Пользовательская исключительная ситуация
BEGIN
SELECT COUNT(Atl),SUM(Atl) INTO StatN, StatAvg FROM Tabl;
IF (ABS(StatAvg - StatN*(:new.Atl))>30)
THEN RAISE Special_case;
END IF;
EXCEPTION
WHEN Special_case THEN
DBMS  QUTPUT.PUT_LINE('Слишком большое уклонение');
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE(‘Не диагностированная ошибка триггера ') ;
END;
/
Trigger created.
```

При вводе значения атрибута, равного 90, происходит срабатывание триггера TRIG_TB2. Выводится диагностическое сообщение, и вводится новая строка. Представленная операция выборки подтверждает ожидаемое изменение в таблице.

```
SQL> insert into tabl values (90);
Слишком большое уклонение
```

В отличие от процедур, функций и пакетов сервер ORACLE не хранит код триггера в виде скомпилированного блока PL/SQL. При первом запуске триггера его код считывается из словаря данных, компилируется, и скомпилированная версия сохраняется в области SGA. Поэтому для часто используемых триггеров целесообразно код, отвечающий за процедурную часть триггера, включать в хранимую процедуру, а в теле триггера оставлять только запись условий запуска и вызовы соответствующих процедур и функций.

На предложения языка SQL, включенные в код триггера ORACLE, наложены следующие ограничения. Тело триггера не может включать в себя явное использование управляющих операторов COMMIT, ROLLBACK и SAVEPOINT, операторов языка определения данных CREATE, ALTER и DROP, операторов, управляющих разграничением доступа GRANT и REVOKE, а также неявное выполнение перечисленных операторов через вызовы процедур и функций.

Строчный триггер срабатывает один раз для каждой строки. Внутри триггера можно обращаться к строке, обрабатываемой в данный момент. Для этого служат две псевдозаписи — :OLD и :NEW, синтаксически они рассматриваются как записи, хотя записями не являются, поэтому их называют псевдозаписями. Тип обеих псевдозаписей определяется как

Активирующая_таблица**&ROWTYPE** ;

Псевдозапись :OLD не определена для операторов INSERT, а псевдозапись :NEW — для оператора DELETE, при этом ошибка генерироваться не будет, но значения полей обеих записей будут NULL значениями. Двоеточие перед :NEW и :OLD обязательно, это двоеточие используется для ограничения переменных привязки. Операции, которые выполняются над записями, не могут быть выполнены над псевдозаписями.

:OLD и :NEW нельзя передавать процедурам и функциям, принимающим аргументы типа Активирующая_таблица**&ROWTYPE** ;

Задания

1. Создать триггер, который считает среднюю стипендию и выдает диагностическое сообщение при ревышении заданного порога уклонения вводимого значения атрибута в зависимости от средней стипендии, при этом происходит заполнение некоторой таблицы.

2. Создать триггер, который считает средний балл в заданный день и выдает диагностическое сообщение при превышении заданного порога уклонения вводимого значения атрибута в зависимости от среднего балла, при этом происходит заполнение некоторой таблицы.
3. Создать триггер, который определяет границы изменения номеров предметов и выдает диагностическое сообщение при превышении заданного порога уклонения вводимого значения атрибута, при этом происходит заполнение таблицы.
4. Создать триггер, который определяет границы изменения номеров преподавателей и выдает диагностическое сообщение при превышении заданного порога уклонения вводимого значения атрибута, при этом происходит заполнение некоторой таблицы.
5. Создать триггер, который вычисляет средний рейтинг университетов и выдает диагностическое сообщение при превышении заданного порога уклонения вводимого значения атрибута в зависимости от величины среднего рейтинга, при этом происходит заполнение некоторой таблицы.
6. Создать триггер, который определяет границы изменения номеров лекторов в зависимости от номеров читаемых курсов и выдает диагностическое сообщение при превышении заданного порога уклонения вводимого значения атрибута, при этом происходит заполнение некоторой таблицы.

Лабораторная работа № 6. Разработка web-приложения

Цель работы – получение навыков создания интернет-приложений.
Задачи:

1) Создать простейшее приложение для отчета по таблице DEPARTMENTS.
2) Дополнить приложение отчетом и формой для таблицы EMPLOYEES.

Очевидно, что непосредственно работать с таблицами для просмотра данных возможно, но затруднительно, особенно для конечного пользователя. Для организации эргономичного интерфейса с целью удобного отображения информации из БД используют *отчеты*, которые, наряду с другими элементами интерфейса (они будут рассматриваться далее), создаются в рамках *приложения* пользователя (можно сказать, что это аналог понятия БД в СУБД Access). Такие приложения включают, в общем случае, разное количество форм представления требуемой пользователю информации.

1. Создание простейшего приложения

1.1. Разработка приложения

Для создания приложения активизируют **Application Builder.** Затем**:**

1. Щелкните по кнопке **Create>**. Появится мастер создания приложения (**Create an Application**).
2. Выберите **Desktop** и нажмите **Next>.**
3. На шаге **Name**:
   а. **Schema** – выберите схему БД, содержащую объекты, с которыми будет работать приложение.
   б. **Name** – введите имя торговой корпорации, соответствующее Вашей фамилии в транслитерации (в рассмотренном примере имя приложения - КГТУ).
   в. **Application** – оставьте автоматически сгенерированное значение (это уникальный идентификатор приложения, с которым работает система и который помогает пользователю сориентироваться).
   г. Выберите тему **Theme – 21 (Scarlet)** и щелкните **Next>**.

Далее добавьте страницы в приложение. Для этого:

1. На шаге **Pages**:

- Удалите страницу **Home,** нажав на **Удаление** ×**.**
- Установите следующее в секции **Add Page**:

а. **Select Page Type** – оставьте значение по умолчанию – **Blank**.
б. **Page Name** – введите Главная**.** Созданная страница является логической «верхушкой» всех создаваемых далее элементов интерфейса.
в. Щелкните по кнопке **Add Page**. Созданная страница появилась в списке страниц приложения в верхней секции.

2. Затем добавьте страницу с отчетом о подразделениях, основанном на таблице DEPARTMENTS. Для этого в секции **Add Page** установите следующее:
   а. **Select Page Type** – выберите **Report**.
   б. **Parent Page**– выберите Главная (это необходимо для определения иерархии страниц в приложении).
   в. **Page Source** – оставьте значение **Table**.
   г. **Table Name** – выберите DEPARTMENTS. В списке отображаются все таблицы и представления схемы, с которой ассоциировано приложение.
   д. **Report Type** – выберите **Classic.**
   е. Щелкните **Add Page**. Страницы, перечисленные в секции **Create an Application,** отображают иерархию страниц в приложении.

Далее измените имя страницы, установленное по умолчанию – Departments**,** – на Подразделения:

а. Щелкните по значку у страницы Departments.
б. В секции **New Page Definition** измените **Page Name** на Подразделения.
в. Щелкните **Apply Changes**.
г. В секции **Pages** щелкните **Next>**.

После добавления страницы необходимо определить некоторые параметры приложения:

1. На шаге **Shared Components** в опции **Copy Shared Components from Another Application**: оставьте **No** и щелкните **Next>**.
2. На шаге **Attributes:**
   а. **Authentication Scheme** оставьте **Application Express Accounts**.
   б. **Language** – выберите **Russian (ru)**.
   в. **Date Format** – выберите маску DD-MON-YYYY, щелкните **Next>**.
3. На шаге **Confirm** проверьте введенные данные и щелкните **Create Application**.

Две созданные страницы – Главная и Подразделения – по умолчанию отображаются в виде иконок на странице приложения в Application Builder. Обратите внимание, что Oracle APEX автоматически добавил в приложение страницу Login Page.

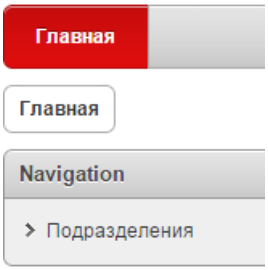
Таким образом, создано приложение с именем КГТУ, содержащее служебную страницу Главная и страницу с данными из таблицы Departments в виде отчета на странице Подразделения, т.е. первая задача решена.

1.2. Разработка навигации по приложению

Для доступа к странице Подразделения отобразим соответствующую ссылку на странице приложения:

1. На странице приложения нажмите **Shared Components:**
   а. В разделе **Navigation** выберите **Lists**.
   б. Нажмите **Create>**.
   в. На шаге **Source** в **Create List** выберите значение From Scratch. Нажмите **Next>.**
   г. На шаге **Name and Type** в опции **Name** напишите **–** Navigation**,** остальные параметры оставьте без изменения, нажмите **Next>.**
   д. На шаге **Query or Static Values** в опции **List Entry Label** напишите –Подразделения, а в опции **Target Page ID or custom URL** – нажмите на значок и выберите страницу Подразделения. Нажмите **Next>**.
   е. На шаге **Confirm** в опции **Create List Regions?** – выберите значение **Create List Region on current page**. Появятся дополнительные поля. В поле **Region Position** выберите - **Page Template Body**, в опции **Region Template** выберите **– Report List**. Нажмите **Create List.**

Чтобы просмотреть приложение, необходимо запустить его. Тогда APEX на основе данных, сохраненных в БД, динамически визуализирует приложение в виде HTML страниц. Для запуска приложения можно щелкнуть по иконке **Run** на домашней странице приложения (вкладка **Application Builder**). После авторизации появится главная страница приложения вида:



Также обратите внимание на панель инструментов разработчика внизу страницы. Данные ссылки появляются, когда приложение запускается в среде разработки.

Перейдите по ссылке `Подразделения`. Появится страница `Подразделения`.

Видно, что полученный отчет отображает **все** данные исходной таблицы. Ему соответствует запрос в виде скрипта:

```
select
"DEPARTMENT_ID",
"DEPARTMENT_NAME",
"MANAGER_ID",
"LOCATION_ID"
from "DEPARTMENTS"
where
(
instr(upper("DEPARTMENT_NAME"),upper(nvl(:P2_REPORT_SEARCH,"DEPART-
MENT_NAME"))) > 0
)
```

Чтобы «добраться» до этого текста, надо сделать следующее:

1. Откройте определение страницы `Подразделения` в списке страниц на вкладке, соответствующей идентификатору Вашего приложения. Для выполнения последующих действий при необходимости следует поменять вид интерфейса, нажав на иконку **Component view**  наверху страницы.
2. В разделе **Page Rendering** найдите секцию **Regions** и щелкните по ссылке `Подразделения`. Откроется окно **Identification.** Найдите секцию **Source** – в ней находится представленный текст запроса.

Обратите внимание, что приложение содержит элементы и свойства, которые позволяют быстро выполнять определенные задачи:

- Строка навигации отображает иерархию и путь страницы в виде ссылок, по которым можно перемещаться.
- Сортировка – чтобы отсортировать данные по столбцу, нужно щелкнуть по заголовку этого столбца.
- **Search** – для поиска записей, содержащих определенные данные, необходимо ввести эти данные (строку поиска) в поле поиска и нажать кнопку **Go**. Поиск не чувствителен к регистру.
- **Display** – поле со списком позволяет выбрать число записей, которое будет отображаться на странице. Для обновления необходимо нажать кнопку **Go**.
- **Spread Sheet** – щелчок по этой ссылке вызывает диалог сохранения данных в CSV файле.
- **Next** и **Previous** – эти ссылки позволяют перемещаться между подмножествами данных (если все записи целиком не помещаются на одной странице, они разбиваются в соответствии со значением параметра **Display**).
- Список подмножеств записей – позволяет выбрать, какое из подмножеств отобразить на странице.

Можно запускать также и отдельные страницы приложения. Для этого нужно нажать на `View Report` (иконка ) и в списке страниц выбрать опцию **Run** напротив нужной страницы.

В полученном отчете для обозначения названий столбцов используется латиница, что является его недостатком. Технология русификации интерфейса рассматривается далее.

Созданный отчет лишь отображает данные из БД и применим для той информации, которая носит условно-постоянный характер. На практике часто возникает необходимость вносить изменения в отчет и, следовательно, в исходные таблицы. Для этого используется особый вид отчетов – отчеты с формой.

2. Создание отчета и формы для таблицы `EMPLOYEES`

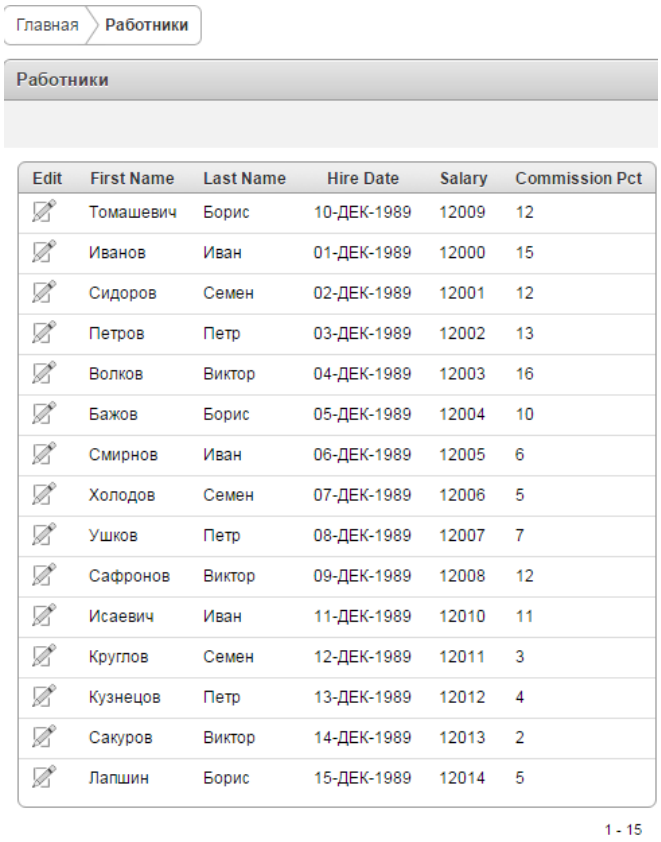
2.1. Разработка отчета и формы

Для создания отчета о работниках и формы для его редактирования:

1. Перейдите на домашнюю страницу приложения в **Application Builder**.
2. Щелкните по кнопке **Create Page>**.
3. На шаге **Create a Page**:
   a. Выберите **Form**, затем **Form on a Table with Report.** Эта опция создает две страницы: отчет и форму, основанные на одной таблице.
4. На шаге **Report Page:**
   a. **Implementation** – выберите `Classic`.
   б. **Breadcrumb** – выберите `Breadcrumb`. Появятся дополнительные свойства **Parent Entry** и **Entry Name .**
   в. В **Parent Entry** выберите ссылку на страницу `Главная`.
   г. В полях **Entry Name**, **Page Name** и **Region Title** измените значение на `Работники`, щелкните **Next>**.
5. На шаге **Data Source**:
   a. В поле **Table/View Owner** оставьте значение по умолчанию.
   б. В поле **Table/View Name** выберите `EMPLOYEES(table)`и щелкните **Next >**.
6. На шаге **Tabs** в **Tab Options** оставьте опцию **Do not use tabs** и щелкните **Next >**.
7. На шаге **Report Columns** необходимо сформировать список столбцов, которые появятся на странице отчета (напомним, что в предыдущем случае отчет содержал все столбцы исходной таблицы)**:**
   a. В списке **Select Column(s)** показаны все имеющиеся данные, в правом списке – те, которые следует отображать. С помощью стрелок «>»и «<» сформируйте требуемый список:
      - `EMPLOYEE_ID`
      - `FIRST_NAME`
      - `LAST_NAME`
      - `HIRE_DATE`
      - `SALARY`
      - `COMMISSION_PCT`

      Выбранные столбцы появятся на странице отчета. Щелкните **Next>**.
   б. В **Edit Link Image** оставьте иконку, выбранную по умолчанию, и щелкните **Next>**.
8. На шаге **Form Page:**
   a. В полях **Page Name**, **Region Title** введите `Создать/Редактировать Работника`. Щелкните **Next>**.
   б. В поле **Primary Key Type** выберите **Select Primary Key Column(s)**, а затем в поле **Primary Key Column 1** выберите `EMPLOYEE_ID` и щелкните **Next >**.
   в. В **Source for Primary Key Column 1** выберите **Existing sequence**, а в появившемся внизу поле **Sequence** выберите `EMPLOYEES_SEQ` (`EMPLOYEES_SEQ` – это объект БД «сиквенс» (последовательность), предназначенный для генерации уникальных числовых значений, которые используются в качестве значений суррогатных идентификаторов). Щелкните **Next>**.
   г. В списке **Select Column(s)** выберите ***все*** столбцы и щелкните **Next>**. Эти столбцы появятся в форме `Создать/Редактировать Работника`.
   д. В следующем блоке оставьте все как есть (значения `Yes` для операций **Insert**, **Update** и **Delete**) и щелкните **Next>**. Это позволит пользователям добавлять, изменять и удалять записи о работниках.
9. На шаге **Confirm** проверьте выбранные атрибуты страниц формы и отчета и щелкните по кнопке **Create**.

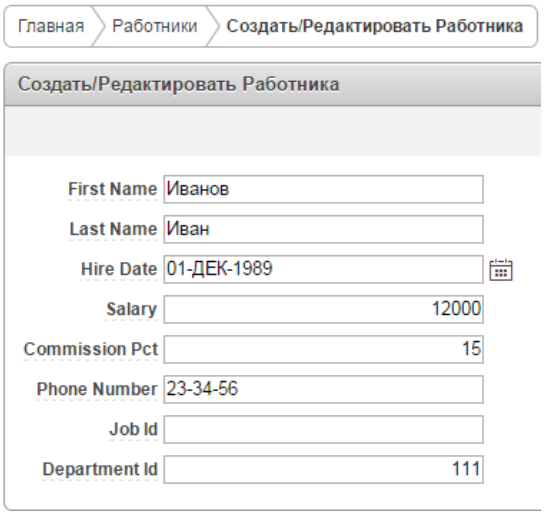
10. Запустите созданную страницу. Отобразится отчет `Работники`:



Обратите внимание на следующие моменты:

- Выбранная иконка редактирования появляется в каждой записи. Иконка представляет собой ссылку, щелчок по которой приводит к переходу на страницу формы `Создать/Редактировать Работника`, в которой можно изменить запись о работнике.
- В правом верхнем углу мастером была создана кнопка **Create** (не видна на рисунке), щелчок по которой также приводит к переходу на страницу формы `Создать/Редактировать Работника`, на которой можно создать запись о новом работнике в таблице `EMPLOYEES`.
- Отчет `Работники` включает в себя выбранные столбцы. Oracle APEX на основе этого выбора создал соответствующий SQL-запрос, который возвращает эти данные.

11. Для того чтобы просмотреть форму для редактирования, которая была создана вместе с отчетом, щелкните по иконке редактирования в какой-либо записи отчета `Работники`. Появится форма `Создать/Редактировать Работника`:



Обратите внимание на следующие моменты:

- Форма содержит кнопки **Cancel**, **Delete** и **Apply Changes**.

- Обязательные столбцы (на которые в БД наложено ограничение целостности **Not Null**) выделены цветом.
- Рядом с полем **Hire Date** отображается иконка календаря  , т.к. тип данных соответствующего столбца – DATE. Щелчок по этой иконке приводит к появлению окна с календарем.
- Названия полей формы отображаются на латинице. Технология их русификации рассмотрена в следующей работе.

2.2. Русификация заголовков и форматирование полей в отчете

1. Установите для каждого столбца страницы Работники в поле **Heading** русскоязычные названия:

	Alias	Link	Edit	Heading	Column Width	Column Alignment	Heading Alignment	Show	Sum	Sort	Sort Sequence	
✎	EMPLOYEE_ID	✓		Изменить		left		☑	☐	☐	-	^ v
✎	FIRST_NAME			Фамилия		left		☑	☐	☐	-	^ v
✎	LAST_NAME			Имя		left		☑	☐	☐	-	^ v
✎	HIRE_DATE			Дата найма		left		☑	☐	☐	-	^ v
✎	SALARY			Оклад/Месяц		left		☑	☐	☐	-	^ v
✎	COMMISSION_PCT			Процент		left		☑	☐	☐	-	^ v

2. В **Column Alignment** установите **right** для столбцов SALARY, COMMISSION_PCT и значение **center** для **всех** столбцов в **Heading Alignment.**
3. Для того чтобы отредактировать формат столбца SALARY:
   а. Щелкните по иконке  слева от него.
   б. На странице редактирования атрибута-столбца SALARY найдите секцию **Column Attributes** и в поле **Number/Date Format** выберите из списка 5,234.10. Соответствующая форматная маска будет подставлена в данное поле.
   в. Щелкните по кнопке **Apply Changes**.
4. Примените изменения.
5. Запустите страницу отчета Работники (показан фрагмент отчета):

Главная > Работники

Работники

Изменить	Фамилия	Имя	Дата найма	Оклад/Месяц	Процент
✎	Петрушенко	Петр	01-МАЙ-2014	10 000,00	10
✎	Прибылев	Павел	14-НОЯ-2014	12 345,00	10
✎	Соколов	Семен	18-ИЮН-2014	12 000,00	15

Обратите внимание на следующие моменты:

- Все столбцы отчета имеют русскоязычные заголовки.
- Столбец Оклад/Месяц отображается в денежном формате.

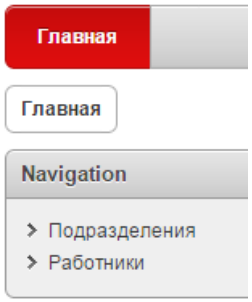
Если сейчас запустить приложение, то можно обнаружить, что нет возможности просмотреть отчет о работниках, т.к. на Главной странице нет никакой ссылки на страницу Работники. На самом деле перейти к любой странице можно, введя ее полный адрес в адресной строке браузера, например, адрес Главной страницы может выглядеть следующим образом: https://apex.oracle.com/pls/apex/f?p=58430:1:10680797661828. В этой ссылке особый интерес представляет строка f?p=58430:1:10680797661828, в которой 58430 – это уникальный идентификатор приложения в APEX, 1 – это номер страницы в приложении, а 10680797661828 – идентификатор сессии, автоматически генерируемый APEX. Заменив номер страницы в ссылке, можно перейти к соответствующей странице.

6. По описанной выше технологии русифицируйте заголовки в отчете Подразделения.

1.3. Добавление на Главной странице ссылки на отчет Работники

Поскольку навигационные средства для нашего приложения уже начали разрабатываться в разделе 1.2, технология добавления ссылки на отчет Работники отличается от той, которая рассматривалась в упомянутом разделе:

1. На странице приложения нажмите **Shared Components**:
   a. В разделе **Navigation** щелкните по ссылке **Lists.** Откроется страница с элементами списка **Lists.** Перейдите на страницу **List Details**. В окне **List** выберите значение Navigation. Появится только одна ссылка – на страницу Подразделение.
   б. На странице **List Details** щелкните по кнопке **Create List Entry>**. Отобразится страница создания/редактирования элемента списка (**Create/Edit**).
   в. На странице **Create/Edit** введите следующее:
      - **Sequence** – 20.
      - **List Entry Label** – Работники.
      - **Page** – выберите страницу Работники.
   г. Щелкните **Create and Create Another.**
2. Запустите Главную страницу:



Обратите внимание, что теперь на Главной странице есть ссылка на страницу Работники.

3. Протестируйте ссылки на Главной странице.

Задание к работе:

1. В своей рабочей области создайте приложение с именем, соответствующим своей фамилии в транслитерации.
2. Для таблицы с условно-постоянной информацией создайте отчет.
3. Для таблицы с переменной информацией создайте отчет с формой для редактирования.
4. Включите в стартовую страницу своего приложения ссылки на оба отчета.
5. Русифицируйте названия полей в обоих отчетах.