

**Кафедра ИУ-4  
«Проектирование и технология производства ЭС»**

# **Журнал практических работ**

**по курсу: «Конструкторско-технологические  
базы данных»**

**Для студентов приборостроительных специальностей**

**20\_\_ / \_\_ учебный год**

Студент \_\_\_\_\_ Группа \_\_\_\_\_  
(фамилия, и. о.)

Преподаватель \_\_\_\_\_ Допуск к экзамену (зачету) \_\_\_\_\_ Подпись \_\_\_\_\_  
(фамилия, и. о.) (число)

**Москва  
2025**

**Программа**  
к учебному плану направления подготовки - 11.03.03  
«Конструирование и технология электронных средств»,

Виды учебных работ	Объем работ, час.	
	Всего	8 семестр
<b><u>Выделено на дисциплину</u></b>	<b>108</b>	<b>108</b>
<b><u>Аудиторная работа:</u></b>	<b>52</b>	<b>52</b>
лекции	<b>40</b>	<b>40</b>
семинары	<b>12</b>	<b>12</b>
<b><u>лабораторные работы</u></b>	<b>-</b>	<b>-</b>
Самостоятельная работа:	<b>56</b>	<b>56</b>
домашнее задание	<b>30</b>	<b>30</b>
изучение разделов	<b>16</b>	<b>16</b>
подготовка к контрольным мероприятиям	<b>10</b>	<b>10</b>
Контроль знаний	Недели контроля	
Срок сдачи домашнего задания		<b>6, 11</b>
<b>Зачет</b>		<b>-</b>
<b>Экзамен</b>		<b>экзамен</b>

Комплексное задание:

Разработать модуль автоматизированной системы радиотехнического предприятия согласно варианту комплексного задания. Система должна позволять просматривать статистическую информацию по процессу производства изделий электронной техники, обрабатывать ее, составлять сводные отчеты и т.п. Система должна позволять генерировать конструкторско-технологические документы операционные, маршрутные карты, спецификации (и т.п. документы, предусмотренные заданием) в формате PDF, используя введенные в нее данные. Система должна быть реализована с использованием лингвистического обеспечения PHP и развернута на основе СУБД Oracle.

По сумме рейтинговых баллов модулей (М1, М2: 6 практических занятий по 5 баллов, ДЗ – 30 баллов, РК – 10 баллов) в семестре (0-70) и на экзамене (М3:0-30) выставляется итоговая оценка исходя из следующих рейтинговых баллов: 60-75 – удовлетворительно, 76-90 – хорошо, 90 и выше – отлично.

График выполнения контрольных мероприятий*																				
Февраль				Март				Апрель					Май					Июнь		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17				
6	13	20	27	6	13	20	27	3	10	17	24	31	1	8	15	22	29	5	12	19
					M1					M2		M3								

\* - все контрольные мероприятия должны быть сданы не менее чем на минимальную пороговую рейтинговую оценку

<p align="center"><b>Отчет по лабораторной работе № 1</b>  <b>«Разработка интерфейсных модулей АИС на РНР»</b></p>			
дата	Оценка (max 5)	Бонус за сложность	подпись

ПОДПИСЬ

Повторение и обобщение принципов разработки пользовательского интерфейса на РНР с функциями доступа к базе данных созданной под управлением СУБД Oracle.

- знакомство с работой интерпретатора PHP и вебсервера Apache
- разработка примеров простейших программ на PHP

-разработка интерфейса модуля АИС с расширенными функциями (поиск и т.п.)

## Основные понятия РНР

---

---

---

---

---

---

---

---

## Простейшие примеры работы с PHP

### Пример 1: Варианты объявления PHP скрипта:

<center><b>Пример 1: Варианты объявления PHP скрипта</b></center>

```
<? echo "1. простейший способ, но возможен конфликт при использовании XML"; ?> <br>
<?php echo("2. Наиболее распространенный способ"); ?> <br>
<% echo("3. Начиная с PHP 3.0.4 можно факультативно применять ASP-теги"); %> <br>
<script language="php">
    echo("4. используется для лучшей совместимости с HTML редакторами");
</script><br>
```

Вид отображения на экране

### Пример 2: Объявления переменных:

```
<?php
// объявление переменных
$a = 1;
$b = 2;

/* тело скрипта*/
$c = $a + $b ;
echo ( "результат сложения a=1 и b=2 равен" );
echo $c;
?>
```

Вид отображения на экране

### Пример 3: Передача значения из формы:

```
<form action="02.php" method="post">
    Name: <input type="text" name="name"><br>
    <input type="submit">
</form>

<?php
echo ("name=");
echo $_POST["name"]; //используется глобальный массив $_POST["name"]
?>
```

Вид отображения на экране

#### Пример 4: Передача нескольких значений из формы:

```
<?php
    echo ("<P>");
    echo ("name= "); echo $_GET["name"];
    echo ("<P>");
    echo ("e-mail= "); echo $_GET["e-mail"];
?>

<P>

<form action=03.php method="get">
    Name:   <input type="text" name="name"><br>
    E-mail: <input type="text" name="e-mail"><br>
           <input type="submit">
</form>
```

Вид отображения на экране

#### Пример 5: Обработка переключателей и радиокнопок

```
<?
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    echo ('Мой любимый предмет: <i>' . $_POST['kurs'] . '</i><br>');

    $favorite_times = count($times);

    if( $favorite_times <= 1) {
        $times_message = 'не ботан';
    } elseif ($favorite_times > 1 && $favorite_times < 4) {
        $times_message = 'ботаю иногда';
    } else {
        $times_message = 'ботан';
    }

    echo ('Я <i>' . $times_message . '</i><br>');
}
?>

<FORM TARGET="04.php" METHOD="POST">
Любимый предмет:<br>
    <INPUT TYPE=RADIO NAME="kurs" VALUE="КТБД">конструкторско-технологические базы данных<br>
    <INPUT TYPE=RADIO NAME="kurs" VALUE="СФМ">Системы функционального моделирования<br>
    <INPUT TYPE=RADIO NAME="kurs" VALUE="СИИ">Системы искусственного интеллекта<br>

Когда вы предпочитаете его изучать:<br>
    <INPUT TYPE=CHECKBOX NAME="times[]" VALUE="м">за завтраком
    <INPUT TYPE=CHECKBOX NAME="times[]" VALUE="н">В обед
    <INPUT TYPE=CHECKBOX NAME="times[]" VALUE="д">за ужином
    <INPUT TYPE=CHECKBOX NAME="times[]" VALUE="л">Поздно ночью

<P>
<INPUT TYPE=HIDDEN NAME="stage" VALUE="results">
<INPUT TYPE=SUBMIT VALUE="Всегда!">
</FORM>
```

Вид отображения на экране

**Задание 1:** Разработать PHP модуль, обрабатывающий значения, передаваемые через опции SELECT, “LISTBOX” (выпадающих меню) форм

[illegible]

Формы представляют собой простейший способ организации внутри HTML-документа обратной связи между пользователем и сервером. Упрощенно формы можно понимать как набор кнопок, флажков, полей ввода, передаваемых сценарию, в качестве входной информации для обработки. Обработка, принятой сервером информации, ничего общего с HTML не имеет и может выполняться самыми разными средствами.

Спецификация дескриптора задания форм		
Дескриптор	Назначение	
<form> </form>	Формуляр	
	Атрибут	Значение
	Action	URL-адрес для отправки заполненного формуляра
	Enctype	кодирование передаваемых данных
	Method	способ передачи формуляра

Типы внутренних конструкций в формах:

- Поля ввода объектов (типы объектов определяются значением атрибута type).
- Поля ввода многострочных текстов.
- Выпадающие меню.
- Поля списков.

## Поля ввода.

Спецификация дескриптора задания форм		
Дескриптор	Назначение	
<input> </input>	Поле для ввода строки	
	Атрибут	Значение
	Action=URL	URL адрес для отправки заполненной формы
	Align= <div> <div>Bottom</div> <div>Left</div> <div>Middle</div> <div>Right</div> <div>top</div> </div>	<div>выравнивает нижний край кнопки по базовой линии строки</div> <div>выравнивает кнопку-иллюстрацию по левому краю текста</div> <div>центрирует кнопку-иллюстрацию в текстовой строке</div> <div>выравнивает кнопку-иллюстрацию по правому краю текста</div> <div>выравнивает верхний край кнопки-иллюстрации по верхней линии строки текста</div>
	Checked	Установленный флажок или выбранное положение переключателя
	Maxlength	Максимальная длина вводимых текстов
	Name	Название элемента ввода/управления
	Src	Источник графического файла картинки кнопки
	Type = <div> <div>checkbox</div> <div>file</div> <div>image</div> <div>hidden</div> <div>password</div> <div>radio</div> <div>reset</div> <div>submit</div> <div>text</div> </div>	<div>Тип элемента управления /ввода</div> <div>флажки, независимы друг от друга, их можно установить/сбросить в любой комбинации</div> <div>поле ввода для имени файла,</div> <div>рядом отображается кнопка, "Пролистать/Browse"</div> <div>открывающая стандартное диалоговое окно выбора файла</div> <div>пользовательская командная кнопка вместо стандартных,</div> <div>получаемых с помощью type=submit или type=reset</div> <div>параметры передаваемые на сервер, которые не могут быть изменены пользователем</div> <div>текстовое поле, вводимые данные отображаются "звездочками"</div> <div>селекторные кнопки (переключатели), из группы можно выбрать только одну</div> <div>командная кнопка, возвращает формуляр к исходному состоянию; данные не пересылаются командная кнопка,</div> <div>отправляет на сервер всё внесенное в формуляр</div> <div>однострочное текстовое поле</div>
	Value	Установленное по умолчанию значение

## Поля ввода многострочных текстов

Спецификация дескриптора ввода многострочного текстового поля		
Дескриптор	Назначение	
<textarea> </textarea>	Многострочное поле для ввода текста в форме	
	Атрибут	Значение
	Cols=N	Количество символов /столбцов в поле ввода
	Name=char	Имя поля ввода
	Rows=N	Количество строк поля ввода
	Wrap=	<div>off</div> <div>physical</div> <div>virtual</div> <div>верстка не выполняется (сервер получает текст одним куском)</div> <div>автоматическая верстка, с переносом строк по мере достижения правого края (сервер получает текст с разрывами строк)</div> <div>автоматическая верстка, с переносом строк по мере достижения правого края (сервер получает текст одной строкой - без разрывов)</div>

## Формирование выпадающих меню и полей списков

Спецификация дескриптора меню или поля списка в форме		
Дескриптор	Назначение	
<select> </select>	Меню или поле списка в форме	
	Атрибут	Значение
	Multiple	Возможность выбора нескольких опций
	Name=char	Имя элемента
	Size=N	Количество одновременно отображаемых элементов

Спецификация дескриптора задания элемента списка/меню в форме		
Дескриптор	Назначение	
<option> </option>	Элемент списка/меню в форме.	
	Атрибут	Значение
	Selection	Выбран по умолчанию
	Value	Параметры элемента



### Пример 6: Подключение к базе данных

```
<?php
    if ($c=OCILogon("scott", "tiger", "//localhost/orcl")) {
        echo "successfully connected to oracle.\n";
        OCILogoff($c);
    }else {
        $err = OCIError();
        echo "Oracle Connect Error " . $err[text];
    }
?>
```

Вид отображения на экране

### Пример 7: Формирование простейшего отчета из базы данных

```
<?php
$c=OCILogon("scott", "tiger", "//localhost/orcl");
if ( ! $c ) {
    echo "Невозможно подключится к базе: " . var_dump( OCIError() );
    die();
}

// производим выборку из базы данных
$s = OCIParse($c, "SELECT object_name, object_type FROM user_objects");
OCIExecute($s, OCI_DEFAULT);

echo ("<center><table border=1>
    <tr><td><center>OBJECT_NAME</center></td>
    <td><center>OBJECT_TYPE</center></td>
    </tr>");
while (OCIFetch($s)) {
    echo ("<tr><td>" . ociresult($s, "OBJECT_NAME") . "</td><td>" .
        ociresult($s, "OBJECT_TYPE") . "</td></tr>");
}

echo ("</table></center>" );

// выполняем commit;
OCICommit($c);

// отключаемся от базы данных
OCILogoff($c);
?>
```

Вид отображения на экране

### Пример 8: Вычислительные возможности Oracle

```
<?php
$c=OCILogon("scott", "tiger", "///localhost/orcl");
if ( ! $c ) {
    echo "Невозможно подключиться к базе: " . var_dump( OCIError() );
    die();
}

// Производим выборку из базы данных
$s = OCIParse($c, "SELECT sin(3.14) FROM dual");
OCIExecute($s, OCI_DEFAULT);
while (OCIFetch($s)) {
    echo "sin ( 3.14 ) = " . ociresult($s) . "\n";
}

// Выполняем commit;
OCICommit($c);

// Отключаемся от базы данных
OCILogoff($c);
?>
```

**Задание 2:** Модифицируйте исходный код так, чтобы выводился в итоговую web страницу результат с точностью до 5 знака, после запятой, реализуйте вычисление других числовых функций.

[illegible]

**Задание 3:** Реализуйте шаблон интерфейсной формы АСУ ТП на РНР, модель которой была разработана вами в 8 ЛР 7 семестра. Разместите шаблон в вашем персональном разделе на сервере [http:// host.iu4.bmstu.ru](http://host.iu4.bmstu.ru).

[illegible]

**Задание 4:** Для тестовой схемы HR выполните задания по формированию отчетов, представленных в задании для самоконтроля, результаты отобразите посредством РНР в своем личном каталоге на сервере.

**Задания для самоконтроля:**

№	Текст задания	Число возвращенных строк
1	Задание: Выбор служащих по диапазону окладов. Выбрать имена, фамилии и оклады служащих, чья зарплата попадает в диапазон от 3000 до 4000 включительно. Упорядочить по окладу в убывающем порядке	
	SQL запрос:	
2	Задание: Выбор подчиненных менеджеров Выберите имена, фамилии и код менеджера служащих, у которых код менеджера <105>, <149> или <205>	
	SQL запрос:	
3	Задание: Выбор сотрудников по буквам Email Выберите, фамилии, адреса почты и телефоны служащих, у которых в e-mail второй стоит буква "H"	
	SQL запрос:	
4	Задание: Задача на выборку Выберите фамилии, зарплаты и комиссионные служащих, не имеющих комиссионных. Отсортируйте результат по окладу.	
	SQL запрос:	
5	Задание: Выбор высокооплачиваемых сотрудников отдела Выберите имена, фамилии, оклады и телефоны служащих, работающих в отделе <60> у которых оклад больше 3000. Упорядочите по окладу в убывающем порядке.	
	SQL запрос:	

6	Задание: Выбор коммиссионных и IT сотрудников Выберите имена, фамилии, оклады, должности и коммиссионные служащих, у которых должность начинается с символов <IT> или у которых есть коммиссионные.	
	SQL запрос:	
7	Задание: Выбор сотрудников не подчиняющихся менеджерам. Выберите имена, фамилии и код менеджера служащих, у которых код менеджера отличается от <105>, <149> или <205>. Упорядочите по фамилии.	
	SQL запрос:	
8	Задание: Дата поступления на работу Выберите имена, фамилии и даты поступления на работу служащих и отсортируйте по возрастанию даты поступления на работу.	
	SQL запрос:	
9	Задание: Телефоны служащих Выберите коды отделов, фамилии и телефоны служащих. Отсортируйте по отделам, а в рамках отдела по фамилиям служащих.	
	SQL запрос:	
10	Задание: Взаимосвязь подразделений Выберите коды подразделений и определите их подчиненность.	
	SQL запрос:	

## **Контрольные вопросы**

1. Основные понятия РНР?
2. Переменные в РНР?
3. Библиотеки работы с СУБД?
4. Авторизация средствами РНР?
5. Установка и настройка РНР?
6. Сессии и управление ими?

## **СПИСОК ЛИТЕРАТУРЫ**

1. Норенков И.П. Системы автоматизированного проектирования. - М.: Изд-во МГТУ им.Н.Э.Баумана. 2001.
2. Иванова Г.С. Технология программирования: Учебник для Вузов. – 2-ое издание, стереотипное. М.: Изд-во МГТУ им.Н.Э.Баумана, 2003. – 320 с.; ил. (Сер. Информатика в техническом университете).
3. Власов А.И., Лыткин С.Л., Яковлев В.Л. Краткое практическое руководство по языку PL/SQL - М.: Машиностроение. 2000. 64 с.
4. Сервер Oracle. Справочное руководство по языку SQL / Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1994.
5. Сервер ORACLE. Основные концепции/ Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1996.
6. ГОСТ Р ИСО 10303-1-99. Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Ч.1. Общие представления и основополагающие принципы. - Москва: ИПК Издательство стандартов, 2000.
7. ГОСТ Р ИСО 10303-1-99. Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Ч.21. Методы реализации. Кодирование открытым текстом структуры обмена. - Москва: ИПК Издательство стандартов, 2000.
8. Проектирование и эксплуатация конструкторско-технологических баз данных на основе СУБД Oracle/ Конспект лекций - М.: Москва, МГТУ им. Н.Э. Баумана, 2001, 120 с.
9. Спецификация UML OMG ISO/IEC 19501.

Отчет по лабораторной работе № 2 «Основные компоненты СУБД Oracle и создание БД»			
дата	Оценка (max 5)	Бонус за сложность	подпись

### Цели работы:

Изучение основных компонентов и инструментария СУБД Oracle, создание тестовой базы данных и ее основных элементов

### Задачи работы:

- Установка СУБД и изучение базового инструментария для работы с СУБД Oracle
- Создание базы данных
- Создание основных элементов БД (таблиц, ограничений и т.п.)

### Задание повышенной сложности (бонус за сложность – 10 баллов):

- создание тестовой базы данных на СУБД Oracle под LINUX

### Краткий конспект теоретической части (ответы на контрольные вопросы)

Основной инструментарий для работы с СУБД Oracle \_\_\_\_\_

---

---

---

---

---

---

---

---

---

---

DDL и DML \_\_\_\_\_

---

---

---

---

---

---

---

---

---

---

Создание пользователей \_\_\_\_\_

---

---

---

---

---

---

---

---

---

---

Управление привилегиями и ролями \_\_\_\_\_

---

---

---

---

---

---

---

---

---

---



Перед тем, как приступить к разработке базы данных для ИИС сформулируем основные общие требования к принципам разработки баз данных.

1. База должна состоять из модулей (совокупности объектов), каждый из которых по возможности должен являться независимой единицей и должен быть способен работать максимально автономно от других модулей.
2. Взаимодействие между модулями должно осуществляться с помощью хранимых процедур и представлений, что имеет целью сделать возможным изменение структуры базы модуля без переделки других модулей, которые с ним взаимодействуют. Запрещается прямое обращение к таблицам другого модуля. При таком подходе при изменении структуры базы (таблиц) в модуле для сохранения работоспособности других модулей будет достаточно переделать соответствующие представления и процедуры.
3. Взаимодействие клиентской части и серверной рекомендуется осуществляться с помощью хранимых процедур и представлений.
4. База данных должна создаваться под единым **ORACLE-пользователем** (владельцем объектов). Конечные пользователи должны обращаться к процедурам, функциям, представлениям владельца объектов через общие синонимы (**PUBLIC SYNONYM**).
5. Конечные пользователи системы не имеют привилегий на действия с объектами БД.
6. Все ORACLE-права должны раздаваться через роли, которые создаются администратором по согласованию с разработчиком. Разработчики должны включать привилегии на создаваемые объекты в ранее созданные роли.
7. С целью обеспечения уникальности и мнемоничности имен объектов базы необходимо придерживаться

**СОГЛАШЕНИЯ (Префиксное наименование таблиц) о наименовании объектов:**

- Имена таблиц и хранимых процедур модуля должны начинаться с какого-либо определенного префикса (2-4 буквы), за которым следует знак подчеркивания «  ». Префикс должен быть одним и тем же для всех таблиц и процедур одного модуля.
- Имена создаваемых общих синонимов должны совпадать с именами самих объектов.
- Имена последовательностей (SEQUENCE) должны начинаться с символов «S  », за которыми следует префикс модуля с последующим подчеркиванием, после чего идет смысловое имя индекса. При создании последовательности для первичного ключа таблицы рекомендуется такое имя последовательности: **S ИМЯ ТАБЛИЦЫ**.
- Имена представлений (View) должны начинаться с символов «V  », далее префикс модуля и смысловая часть (например: имя основной таблицы).
- Имена индексов (INDEX) должны начинаться с символов «I  », далее имя таблицы (или его сокращение), после чего следует смысловая часть. Например: для первичных ключей рекомендуется применять аббревиатуру «PK», для FOREIGN KEY - имена столбцов или сокращений, на которые осуществляется ссылка.
- Имена ограничений на таблицы (CONSTRAINT) должны начинаться с символов «C  » (ограничения типа PRIMARY KEY должны начинаться с символов «I  », т.к. в соответствии с этим именем создается индекс), далее аналогично именам последовательностей.
- Имена триггеров (TRIGGER) должны начинаться с символов «T  », далее аналогично именам последовательностей.
- Имена связей (DB LINK) должны начинаться с символов «L  », далее аналогично именам последовательностей.

**Требования к именам объектов базы данных**

- должны иметь длину от 1 до 30 байт, за исключением имен баз данных, длина которых ограничена 8 байтами;
- не могут содержать кавычек;
- не могут совпадать с именами других объектов.

Имена, которые всегда заключены в двойные кавычки, могут нарушать, приведенные ниже правила. В противном случае, имена

- должны начинаться с букв A-Z;
- могут содержать только символы A-Z, 0-9,   , \$ и #;
- не могут дублировать зарезервированные слова SQL.

Различие между прописными и строчными буквами учитывается только в именах, заключенных в двойные кавычки.

**Рекомендации:**

1. Рекомендуется, чтобы поля таблиц имели префиксы (для каждой таблицы свой префикс).

2. Все создаваемые ограничения (CONSTRAINT) должны быть поименованы. Например:

```
CREATE TABLE DAMN_TABLE  
    (A CHAR CONSTRAINT I DAMN_TABLE PK PRIMARY KEY  
    USING INDEX TABLESPACE  
                                    &index_tablespace);
```

а не:

```
CREATE TABLE DAMN_TABLE  
    (A CHAR PRIMARY KEY);
```

т.к. во втором варианте название индекса генерируется автоматически и не связано с именем таблицы.

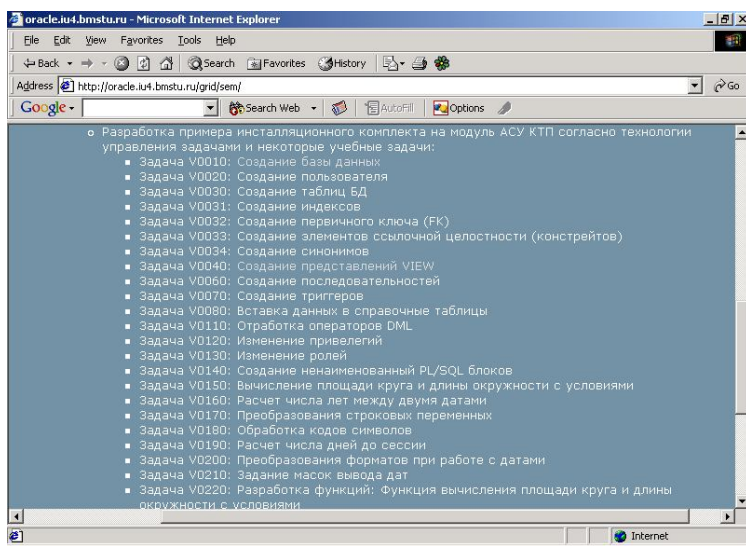
3. Создание общих синонимов должно прописываться в тех же SQL-программах, что и создание объектов.

4. Требуется писать комментарии в тексте хранимых процедур и комментировать таблицы, представления и их поля с помощью SQL команды COMMENT.

5. Необходимо использовать существующую систему разграничения доступа или, по крайней мере, предусмотреть в коде представлений и хранимых процедур места, куда в будущем можно было бы вставить процедуры системы разграничения доступа. Эти процедуры отвечают на вопрос «имеет ли текущий пользователь право сделать данное действие над данным объектом», а представления показывают совокупность разрешенных действий над объектами для текущего пользователя.

- 6 Хранимые процедуры не должны находиться в состоянии ожидания освобождения заблокированных другими пользователями записей длительное время. Для этого перед попыткой захвата данных необходимо убедиться в отсутствии чужих блокировок (SELECT ... FOR UPDATE ...NOWAIT), и в случае наличия блокировок пользователю должно быть выдано соответствующее сообщение.
- 7 При написании хранимых процедур нужно считать, что на Delphi-клиенте выбран режим работы SHARED NOAUTOCOMMIT.
- 8 Ошибки хранимых процедур, как логические так и подключения к ORACLE, должны обрабатываться в самих процедурах с выполнением ROLLBACK, по крайней мере, до точки вызова процедуры. При этом к клиенту не должен возвращаться текст сообщения ORACLE SERVER об ошибке. Вместо этого клиенту должен возвращаться код ошибки и подробное описание ошибки, созданное разработчиком.
- 9 В начале процедуры должны быть комментарии, отражающие следующие сведения:
- наименование подсистемы;
  - наименование модуля;
  - версия и дата последнего обновления;
  - фамилия разработчика;
  - краткое назначение;
  - комментарий к каждому входному и выходному параметру;
  - производит ли процедура фиксацию транзакций во время своей работы;
  - до какой точки производится откат транзакций в случае ошибки;
  - оставляет ли процедура заблокированные записи в таблицах после выполнения своей работы.

### ТРЕБОВАНИЕ К ОФОРМЛЕНИЮ SQL-ПРОГРАММ ПРИ ПРОВЕДЕНИИ ИЗМЕНЕНИЙ В БАЗЕ ДАННЫХ



Разрабатываемые скрипты для создания объектов базы данных и внесения изменений рекомендуется оформлять в виде отдельных sql программ (скриптов), каждый из которых (либо группа скриптов) размещается в отдельном каталоге на диске с именем, например, V0001 – где цифра обозначает номер текущей задачи для рассматриваемого набора скриптов. В каталоге находится файл index.htm (с описанием сути задачи), файл start.sql (в котором прописана очередность запуска скриптов) и файлы самих скриптов. Все файлы index.htm связаны в единый перечень задач корневым файлом inex.htm, образуя тем самым гипертекстовую справочную систему по всему созданному в проекте инсталляционному комплекту (рис)

Необходимо предусмотреть возможность запуска sql-файлов в назначенной последовательности несколько раз.

При выполнении sql-программы должен включаться вывод результатов в файл. Для этого необходимо добавить:

- в начало файла:

**SPOOL xxxxxxxx.lst**

где xxxxxxxx - имя файла sql-программы;

- в конец файла:

**SPOOL off.**

Необходимо, чтобы в sql-программе при выполнении действий с объектом базы данных отражалось имя этого объекта, например:

**PROMPT создается TABLE1**

**CREATE TABLE1 ...**

Следует включить в команду CREATE INDEX опцию TABLESPACE и предусмотреть возможность интерактивной подстановки имени табличного пространства администратору базы данных в момент выполнения sql-программы:

**CREATE INDEX i\_table1 ON table1 ...TABLESPACE &&tablespace;**

Перед командой создания синонимов должна ставиться команда удаления синонимов с тем же именем:

**DROP PUBLIC SYNONYM xxx. ...**

**CREATE PUBLIC SYNONYM xxx ...**

## 1 Изучение инструментария для работы с СУБД Oracle

### Инструментарий:

- проектирования и моделирования - Oracle Designer;
- разработки экранных форм - Oracle Forms Developer;
- разработки отчетных форм - Oracle Reports Developer;
- быстрой разработки приложений на языке Java - Oracle JDeveloper;
- разработки и внедрения корпоративных хранилищ данных и интеллектуальных приложений электронного бизнеса - Oracle Warehouse Builder;
- выполнения DDL- и DML-запросов к базе данных - SQL\*Plus.

Для расширения информационной системы предприятия за счет дополнительных аналитических средств поиска, анализа и моделирования предназначены семейства продуктов Oracle Discoverer (ROLAP) и Oracle Express (MOLAP).

Для обеспечения функций интерактивного управления Oracle Enterprise Grids — сетями распределенных вычислений предприятия предназначен специальный модуль - Oracle Enterprise Manager 10g. Этот модуль значительно упрощает управление бизнес-приложениями при одновременном сокращении затрат на их поддержку и обновление. Oracle Enterprise Manager 10g представляет собой управляющее ПО на основе тонкого клиента, которое дает полную картину вычислительной инфраструктуры предприятия. Системные администраторы получают возможность вносить изменения в информационные политики, уровни сервисного обслуживания, а также в распределение существующих вычислительных ресурсов и приложений в зависимости от изменяющихся потребностей бизнеса, обеспечивая высочайшее качество обслуживания.

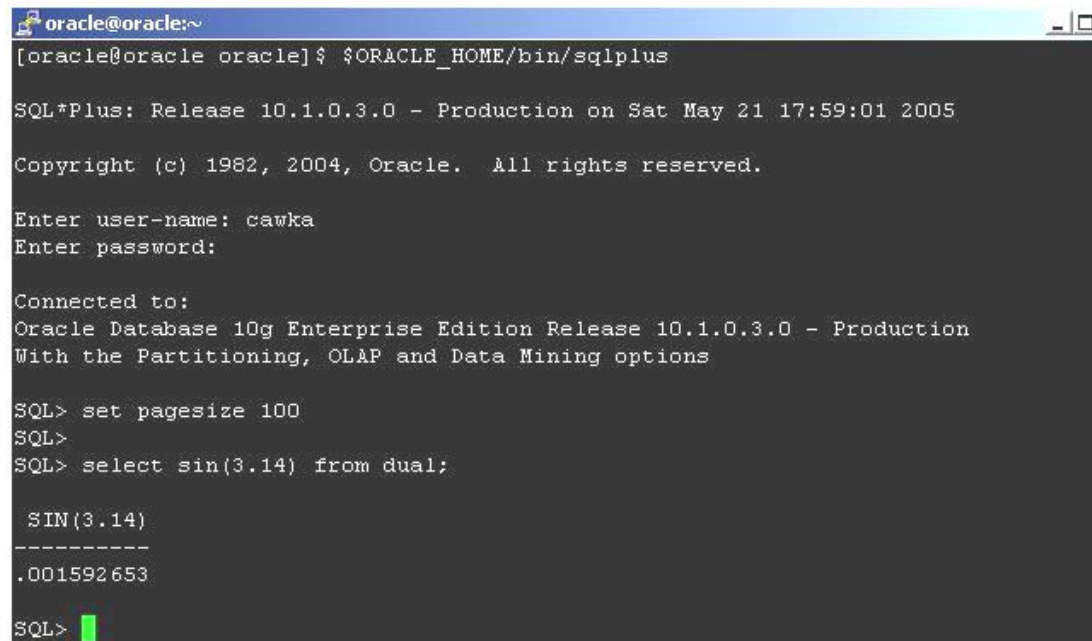
### 1.1. Использование SQL\*Plus

Одним из основных инструментов по работе с СУБД Oracle является **SQL\*Plus**, позволяющий задавать текст команд (скриптов) на языке SQL и выполнять их в интерактивном режиме. Приложение принимает от пользователя инструкции для доступа к базе данных и направляет их серверу Oracle, а результаты отображает на экране монитора. Кроме того среда SQL\*Plus имеет набор специальных команд, используемых для:

- форматирования результатов, возвращаемых сервером
  - установки опций настройки среды SQL\*Plus
  - редактирования и хранения SQL-команд.
- Приложение SQL\*Plus можно запустить разными способами.
- **Как консольную программу** Программа выполняется из оболочки или командной строки (окружения, которое иногда называют консолью).
  - **В качестве программы с псевдографическим пользовательским интерфейсом.** Эта разновидность SQL\*Plus доступна только в Microsoft Windows. Ее называют псевдографическим интерфейсом, поскольку она очень похожа на консольную программу, отличаясь от таковой, кроме прочего, наличием растровых шрифтов.
  - **Через iSQL\*Plus (в Oracle9: или более поздней версии).** Программа выполняется из web-браузера машины, на которой работает HTTP-сервер Oracle и сервер iSQL\*Plus.
  - **Через SQL\*Plus Worksheet.** Графический пользовательский интерфейс Java с консольной версией SQL\*Plus. Хотя он поддерживает небольшой журнал инструкций, ничего примечательного в этой версии среды разработки нет.

## Запуск SQL\*Plus в консольном режиме:

Консольный режим работы обеспечивает доступ к серверу, на котором установлена СУБД в консольном режиме (например, по telnet). На рисунке 1.1. показано обращение к серверу oracle.iu4.bmstu.ru по протоколу telnet, подключение к серверу и вызов sql+ из командной строки.



```
oracle@oracle:~  
[oracle@oracle oracle]$ $ORACLE_HOME/bin/sqlplus  
  
SQL*Plus: Release 10.1.0.3.0 - Production on Sat May 21 17:59:01 2005  
  
Copyright (c) 1982, 2004, Oracle. All rights reserved.  
  
Enter user-name: cawka  
Enter password:  
  
Connected to:  
Oracle Database 10g Enterprise Edition Release 10.1.0.3.0 - Production  
With the Partitioning, OLAP and Data Mining options  
  
SQL> set pagesize 100  
SQL>  
SQL> select sin(3.14) from dual;  
  
      SIN(3.14)  
-----  
      .001592653  
  
SQL>
```

Рис.1.1. Работа с SQL+ в командной строке.

Для запуска консольной версии SQL\*Plus, нужно ввести команду *sqlplus* в ответ на приглашение операционной системы. Далее следуют текст, а также строки для ввода имени пользователя и пароля (например, scott, tiger – для тестовой схемы):

```
D:\>sqlplus  
  
SQL*Plus: Release 9.0.1.0.1 - Production on Sun Jan 23 14:23:03 2005  
  
<C> Copyright 2001 Oracle Corporation. All rights reserved.  
  
Enter user-name: scott  
Enter password:  
  
Connected to:  
Oracle9i Enterprise Edition Release 9.0.1.1.1 - Production  
With the Partitioning option  
JServer Release 9.0.1.1.1 - Production  
  
SQL> _
```

Можно также загрузить SQL\*Plus, сразу задав имя пользователя и пароль в командной строке:

```
D:\>sqlplus scott/tiger
```

С помощью ключа */NOLOG* можно запустить SQL\*Plus без подключения к базе данных, указав имя пользователя и пароль посредством команды *CONNECT*:

```
D:\>sqlplus /nolog  
  
SQL*Plus: Release 9.0.1.0.1 - Production on Sun Jan 23 14:36:08 2005  
  
<C> Copyright 2001 Oracle Corporation. All rights reserved.  
  
SQL> CONNECT scott/tiger  
Connected.  
SQL> _
```

Для подключения к удаленным базам данных (то есть серверам баз данных, работающих на других компьютерах) наряду с именем пользователя и паролем нужно знать *идентификатор подключения* Oracle Net (SID), называемый также *именем сервиса*. Идентификатор вводится после имени пользователя и пароля, отделяясь от них символом «@»:

```
SQL> CONNECT scott/tiger@ORADB
Connected.
SQL> _
```

Исполняемый файл для запуска SQL+ находится в подкаталоге bin домашней директории Oracle и называется sqlplus (sqlplus.exe для Windows 1).

### i SQL \* Plus:

iSQL\*Plus представляет собой модную ныне Web-реализацию SQL\*Plus. Продукт работает в трехзвенной архитектуре: Web браузер - iSQL\*Plus - База Данных. Пользовательский интерфейс выполнен при помощи HTML+JScript. Впервые продукт появился вместе с Oracle 9.0.1 на платформе Windows. С версией 9.2 он доступен и на \*nix системах. Между ним и консольным SQL\*Plus есть существенные различия. Например, он не поддерживает следующие команды: EXIT, GET, HOST, SPOOL, WHENEVER... и это далеко не полный перечень.

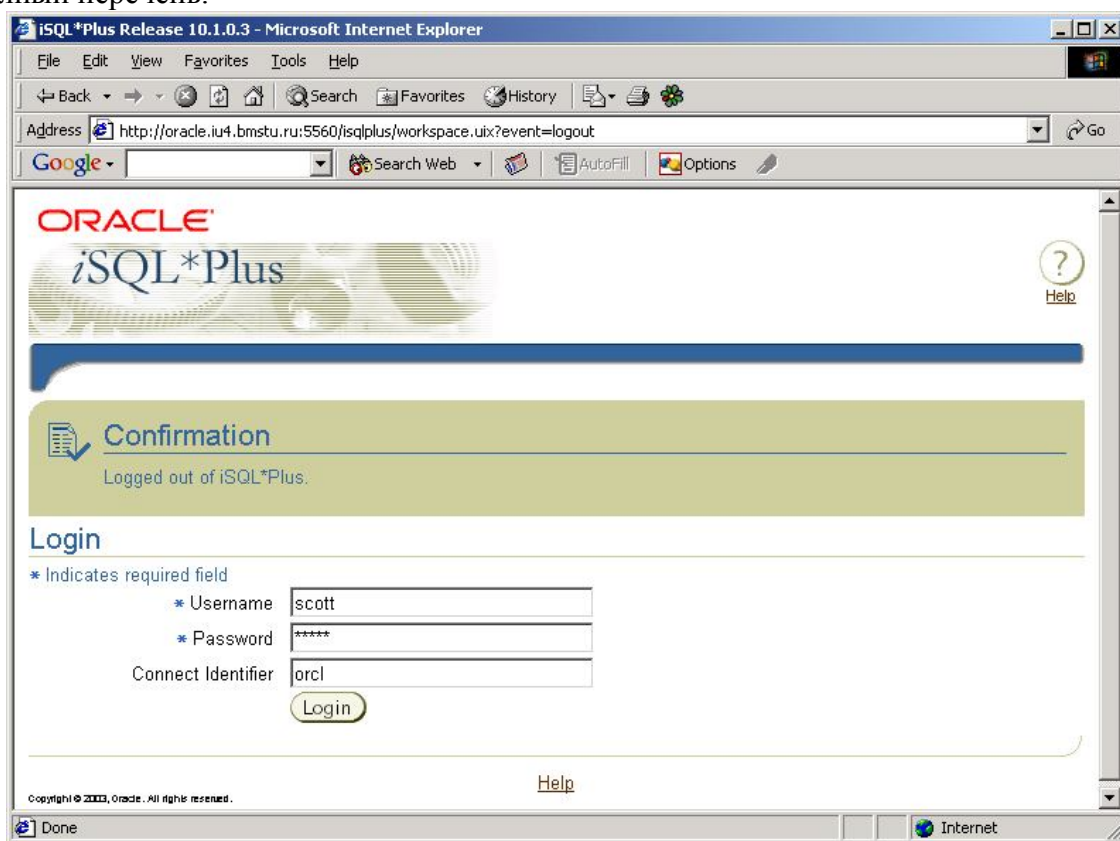


Рис.1.2. Подключение к iSQL\*Plus.



**Задание:** Подключиться к iSQL\*Plus на тестовом сервере и выполнить простейший скрипт:

Адрес тестового сервера в интернете: <http://oracle.iu4.bmstu.ru:5560/isqlplus>;  
LOGIN: scott (доступна схема SCOTT, а также схемы, созданные Вами на занятиях);

PASSWORD: tiger (пароль пользователя SCOTT, либо Ваш); SID: ORCL

Пример SQL команды: Вычисление значения синуса

```
SELECT sin(3.14) FROM DUAL;
```

Посредством SELECT запроса обращаемся к встроенным тригонометрическим функциям Oracle, результат выводится в выходной поток. Специальная таблица DUAL используется для сохранения синтаксиса SELECT запроса.

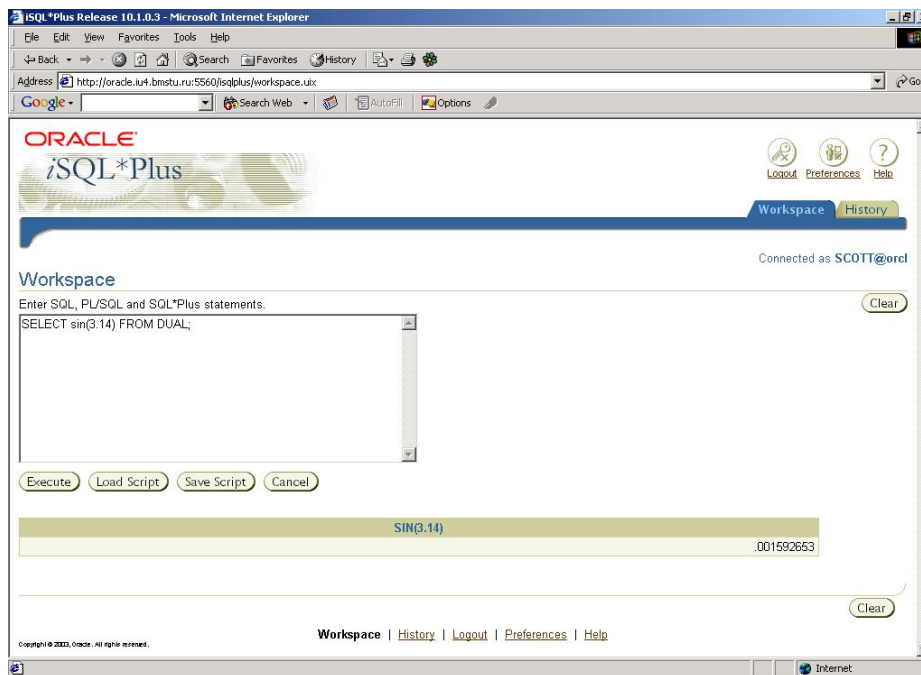


Рис. 1.3. Работа с iSQL\*Plus.

Для запуска iSQL\*Plus необходимо выполнить под юзером oracle \$ORACLE\_HOME/bin/isqlplus start, после чего вы сможете обращаться к нему через WEB интерфейс <http://localhost:5560/isqlplus>. Кроме iSQL\*Plus существует несколько различных его реализаций.

### Графический SQL\*Plus (Win):

Существует только на платформе Windows. Представляет собой такое же консольное приложение, но в графическом окне. По своим возможностям он практически идентичен консольному варианту. Исполняемый файл для запуска находится в подкаталоге bin домашней директории Oracle и называется sqlplusw (plus80w.exe для восьмой версии и plus73w.exe для 7.3).

Среди преимуществ этой реализации можно отметить ещё более простое использование выделения и вставки текста. Выделите мышью нужный фрагмент текста и, не отпуская левой кнопки мыши, нажмите на правую кнопку. Выделенный фрагмент будет вставлен в позицию курсора. Причем нужно отметить, что выделенный фрагмент не помещается в стандартный буфер Windows, который доступен через Ctrl-C/Ctrl-V, как и в любом графическом Windows приложении.

Ещё одним приятным и не слишком известным фактом, является возможность изменить название и размер используемого шрифта.

Для тех, кто использует эту реализацию SQL\*Plus, не самой лучшей новостью станет известие о том, что в будущих версиях графического SQL\*Plus больше не будет. Ему на смену идет iSQL\*Plus (эту информацию можно найти в файле \$ORACLE\_HOME92/sqlplus/doc/README.htm, поставляемым к SQL\*Plus версии 9.2). Хотя никто и не запрещает поставить его и использовать совместно с Oracle 10g.

### SQL\*Plus Worksheet

SQL\*Plus Worksheet является компонентом Enterprise Manager (до Oracle 9.x), хотя может вызываться и автономно. Как и у Enterprise Manager, пользовательский интерфейс SQL\*Plus Worksheet написан на Java. Но это только пользовательский интерфейс. Для выполнения команд и скриптов используется обычный sqlplus в фоновом режиме.



Вообще говоря, это очень странный продукт. Единственное чем он выгодно отличается от рассмотренных ранее реализаций, так это наличием истории команд. Но при этом Worksheet не поддерживает: START, EXIT, GET, HOST, SPOOL, WHENEVER... и это далеко не полный перечень. Вероятно, нет смысла использовать сильно урезанную версию продукта, если доступна полнофункциональная версия (в наши планы не входит описание различных “hacks”, при помощи которых можно добиться от Worksheet полной функциональности SQL\*Plus).

Забегая не много вперед, отметим, что вся информация по всем объектам Oracle хранится в таблицах системного словаря, обращаясь к которым мы можем получить ее. При этом каждому пользователю соответствует своя **схема** (имя которой совпадает с именем пользователя), в которой хранятся все объекты принадлежащие пользователю. Так, например, пользователю SCOTT соответствует схема SCOTT, а информация по всем объектам в этой схеме можно получить обратившись к таблице системного словаря USER\_OBJECTS с правами пользователя SCOTT, но для начала надо узнать структуру этой таблицы, а для этого есть команда DESCRIBE (DESC).



**Задание:** Подключиться к iSQL\*Plus на тестовом сервере и выполнить простейший скрипт:

Адрес тестового сервера в интернете: <http://oracle.iu4.bmstu.ru:5560/isqlplus>;

LOGIN: scott (доступна схема SCOTT, а также схемы, созданные Вами на занятиях);

PASSWORD: tiger (пароль пользователя SCOTT, либо Ваш);

SID: ORCL

Пример SQL команды: Получение информации о атрибутах таблицы.

DESC USER\_OBJECTS;

Пример SQL команды: Вывести все объекты, принадлежащие схеме SCOTT

```
SELECT OBJECT_NAME
       ,OBJECT_TYPE
       ,CREATED
FROM USER_OBJECTS;
```

Name	Null?	Type
OBJECT_NAME		VARCHAR2(128)
SUBOBJECT_NAME		VARCHAR2(30)
OBJECT_ID		NUMBER
DATA_OBJECT_ID		NUMBER
OBJECT_TYPE		VARCHAR2(19)
CREATED		DATE
LAST_DDL_TIME		DATE
TIMESTAMP		VARCHAR2(19)
STATUS		VARCHAR2(7)

Рис. 1.4. Результат выполнения команды DESC.

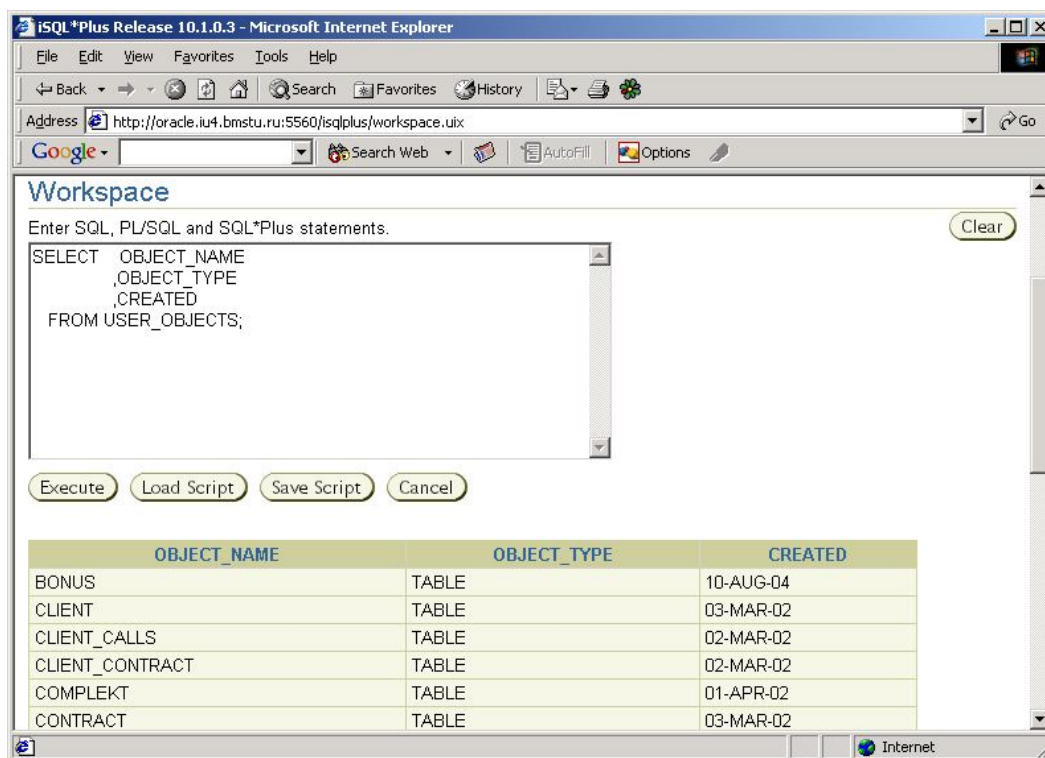


Рис. 1.5. Получение списка объектов принадлежащих схеме пользователя SCOTT.

Используя этот не хитрый набор команд можно в любой момент получить информации по всем объектам конкретной схемы.

## 1.2. Использование Oracle Enterprise Manager

Oracle Enterprise Manager 10g — модуль, предназначенный для управления Oracle Enterprise Grids — сетями распределенных вычислений предприятия. Этот модуль значительно упрощает управление бизнес-приложениями при одновременном сокращении затрат на их поддержку и обновление. Oracle Enterprise Manager 10g представляет собой управляющее ПО на основе тонкого клиента, которое дает полную картину вычислительной инфраструктуры предприятия. Системные администраторы получают возможность вносить изменения в информационные политики, уровни сервисного обслуживания, а также в распределение существующих вычислительных ресурсов и приложений в зависимости от изменяющихся потребностей бизнеса, обеспечивая высочайшее качество обслуживания. Открытость и расширяемость.

Oracle Enterprise Manager 10g создан на базе открытой архитектуры, основанной на стандартах Common Information Model (CIM) и Web-based Enterprise Management (WBEM), разработанные рабочей группой по управлению распределенными средами (Distributed Management Task Force, DMTF). Управление. Управление сетями распределенных вычислений является технологической дисциплиной, охватывающей вопросы, связанные с вычислительной архитектурой, персоналом, процессами, а также с накопленным опытом. Oracle Enterprise Manager 10g отлично справляется с этой задачей.

Oracle Enterprise Manager 10g — это компонент управления, который создан на базе накопленного корпорацией Oracle опыта в области управления центрами хранения и обработки данных; он помогает администраторам повысить производительность вычислительных систем и выйти на качественно новый уровень автоматизации и анализа. Oracle Enterprise Manager 10g — инструмент, который увеличит отдачу от использования решений Oracle для сетей распределенных вычислений.





**Задание:** Подключиться к ЕМ на тестовом сервере:

Адрес тестового сервера в интернете: <http://oracle.iu4.bmstu.ru:5500/em>;

LOGIN: SYSTEM (при первом обращении после установки, либо логин с ролью dba, сообщенный Вам на занятиях);

PASSWORD: MANAGER (пароль пользователя SYSTEM); SID: ORCL

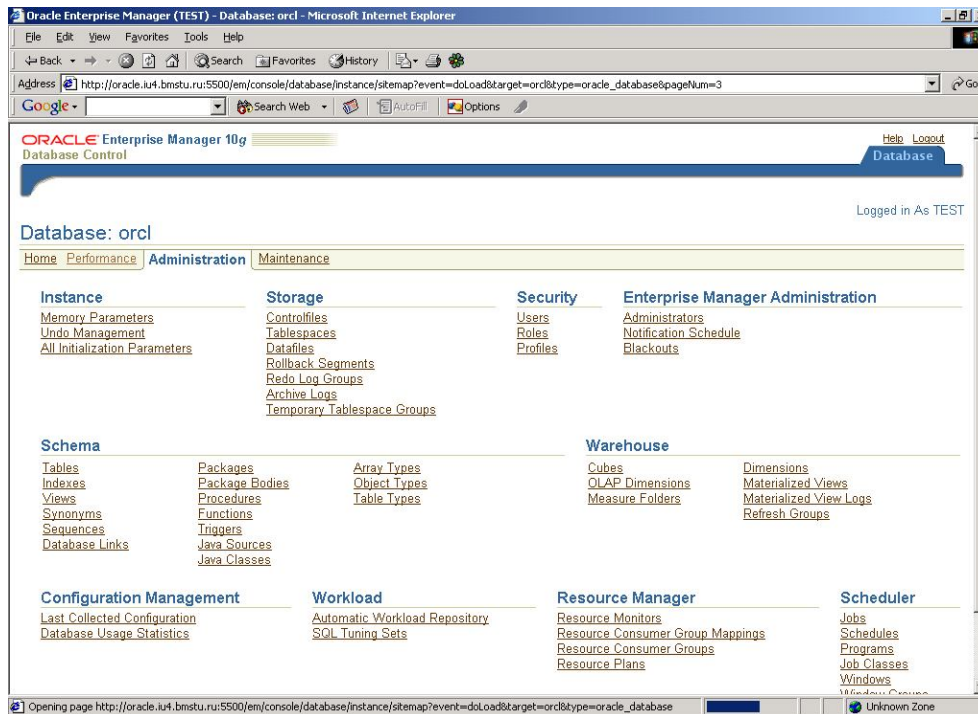


Рис.1.6. Вид WEB интерфейса

В случае перезагрузки системы, если инициализация сервисов не настроена автоматически ЕМ инициализируется следующей последовательностью команд (ОС LINUX):

```
#заходим в систему под специальным пользователем oracle
su - oracle -c "$ORACLE_HOME/bin/emctl start dbconsole"
#определяем переменные
ORACLE_HOME=/home/oracle/product/10.1.0/Db_1/
export ORACLE_HOME
ORACLE_SID=orcl
export ORACLE_SID
#Запускаем СУБД Oracle
#Стартуем базу
su - oracle -c "$ORACLE_HOME/bin/dbstart"
#Запускаем isqlplus
su - oracle -c "$ORACLE_HOME/bin/isqlplusctl start"
#Запускаем EM
su - oracle -c "$ORACLE_HOME/bin/emctl start dbconsole"
#Запускаем TNS Listener
su - oracle -c "$ORACLE_HOME/bin/lsnrctl start"
```

После этого мы можем обращаться к ЕМ по адресу <http://host:5500/em>;

Участники программы Oracle Technology Network (OTN) смогут бесплатно загрузить новое управляющее ПО, посетив сайт <http://otn.oracle.com>.

**Вклейте отчет с конфигурацией созданного вами тестового сервера**

## 2. ИЗУЧЕНИЕ ОПЕРАТОРОВ СОЗДАНИЯ И ИЗМЕНЕНИЯ ОБЪЕКТОВ БД

Перед тем как работать с данными БД ее надо создать, для этих целей используется специальная группа операторов, предназначенных для создания объектов базы данных, все операторы данной группы начинаются с ключевого слова **CREATE**:

- **CREATE DATABASE** - СОЗДАНИЕ БАЗЫ ДАННЫХ
- **CREATE USER** - СОЗДАНИЕ ПОЛЬЗОВАТЕЛЯ
- **CREATE TABLE** - СОЗДАНИЕ ТАБЛИЦ,
- **CREATE SYNONYM** - СОЗДАНИЕ СИНОНИМОВ,
- **CREATE INDEX** - СОЗДАНИЕ ИНДЕКСОВ,
- **CREATE VIEW** - СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ.
- **CREATE SEQUENCE** - СОЗДАНИЕ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

и т.д.

Для внесения изменений в уже созданные объекты используются операторы группы **ALTER**, которые начинаются с ключевого слова **ALTER**. Будем рассматривать операторы групп **CREATE** и **ALTER** попарно, применительно к каждому из рассматриваемых объектов БД.

**Синтаксическая диаграмма** – это графическое представление правил построения конструкций языка в наглядной форме. При этом принято, символы алфавита изображать блоками в овальных рамках, названия конструкций – в прямоугольных, а правила построения конструкций в виде линий со стрелками (часто овальные и прямоугольные рамки для упрощения опускают). Также выделяют осевую линию синтаксической конструкции, на которой размещаются все обязательные символы алфавита и конструктивные элементы. Все необязательные элементы и символы размещаются ниже или выше осевой линии конструкции. Если конструкция имеет значительную длину, то осевая линия переносится соответственно на следующую строку. Существуют и альтернативные формы отображения синтаксиса языка, например форма Бэкуса-Наура (БНФ) и т.п. [2].

Рассмотрим суть использования синтаксической диаграммы для представления синтаксиса языка HTML на простейшем примере. Так, например, стандарт определяет, что документ должен начинаться тэгом **<html>** и заканчиваться тэгом **</html>**. Сам документ, с всевозможными включениями (например, JavaScript или VBScript), располагается внутри этих тэгов. Синтаксическая диаграмма, описывающая эту конструкцию представлена на рис.2.1.



Рис.2.1 Синтаксическая диаграмма конструкции HTML документа.

В дальнейшем обозначение символов алфавита в овальных рамках будем опускать.

## CREATE/ALTER DATABASE - СОЗДАНИЕ БАЗЫ ДАННЫХ

Задаёт и определяет максимальное число экземпляров файлов данных и журнальных файлов, устанавливает режим архивирования.

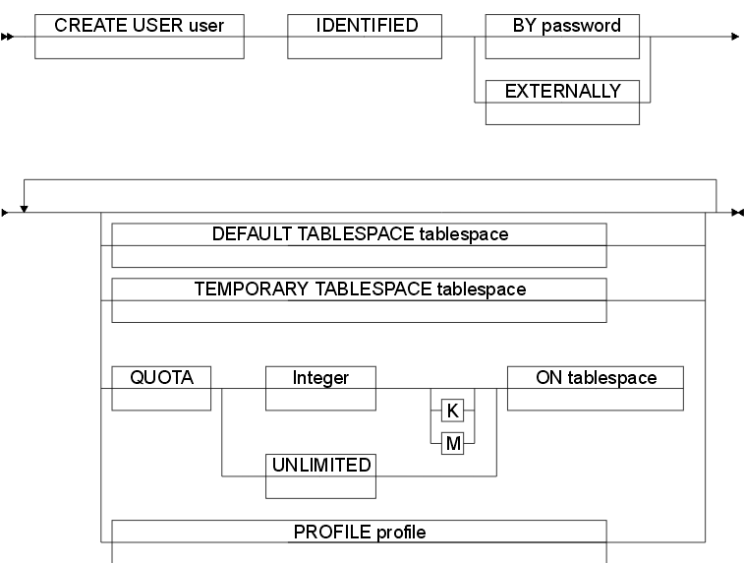

Синтаксическая диаграмма	Тело скрипта
	<p>Пример:</p> <pre>PROMPT Создаем БД CLINICS SPOOL clinic.log CONNECT internal  STARTUP                                NOMOUNT pfile=/oracle/dbs/initclinic.ora  CREATE DATABASE "clinics"   MAXINSTANCES 1   MAXLOGFILES 10   CHARACTER SET "RU8PC866"   NATIONAL CHARACTER SET   "RU8PC866"   DATAFILE   '/oracle/db/system01.dbf'             size   100M   LOGFILE   '/oracle/db/lo  g01.dbf' size 1M,   '/oracle/db/log02.dbf' size 1M;  DISCONNECT SPOOL OFF</pre>
	<p><b>Вклеить текст скрипта создания своей БД</b></p>

Вклеить HTML отчет с конфигурацией созданной тестовой базы данных

### 3 УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ. ПРИВИЛЕГИИ И РОЛИ

#### CREATE/ALTER USER - СОЗДАНИЕ И ИЗМЕНЕНИЯ ПОЛЬЗОВАТЕЛЯ

Задания по управлению профилями пользователей выполняются на тестовом сервере с права DBA.

Синтаксическая диаграмма	Тело скрипта
	<p>Пример: PROMPT Скрипт создания пользователя test</p> <pre>CREATE USER "test"   PROFILE "DEFAULT"   IDENTIFIED BY "123456"   DEFAULT TABLESPACE "USERS"   QUOTA UNLIMITED     ON CWM Lite   QUOTA UNLIMITED     ON DRSYS   QUOTA UNLIMITED     ON EXAMPLE   QUOTA UNLIMITED     ON INDX   QUOTA UNLIMITED     ON SYSTEM   QUOTA UNLIMITED     ON TEMP   QUOTA UNLIMITED     ON TOOLS   QUOTA UNLIMITED     ON USERS ACCOUNT UNLOCK; GRANT "CONNECT" TO "test"; GRANT "DBA" TO "test";</pre>
	Вклеить текст собственного скрипта

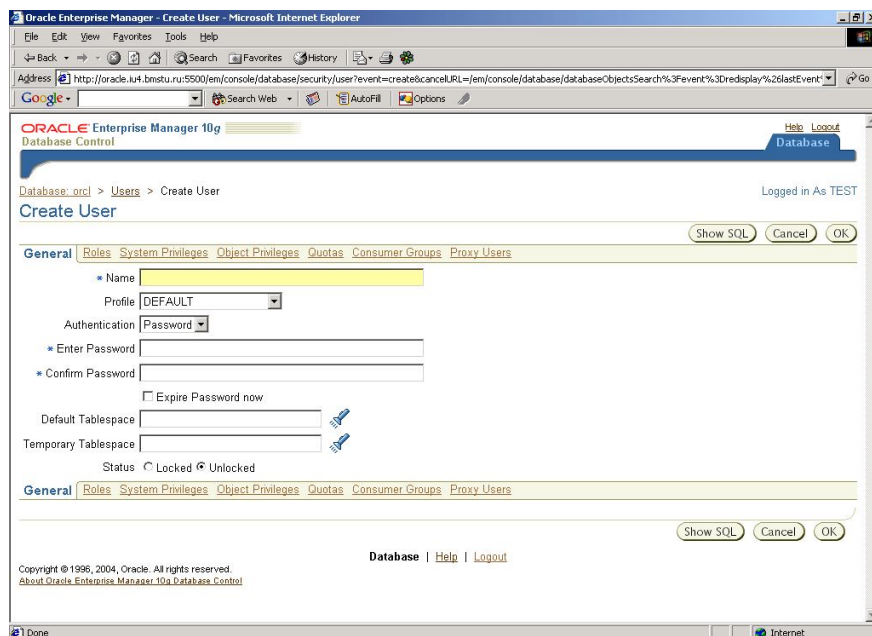


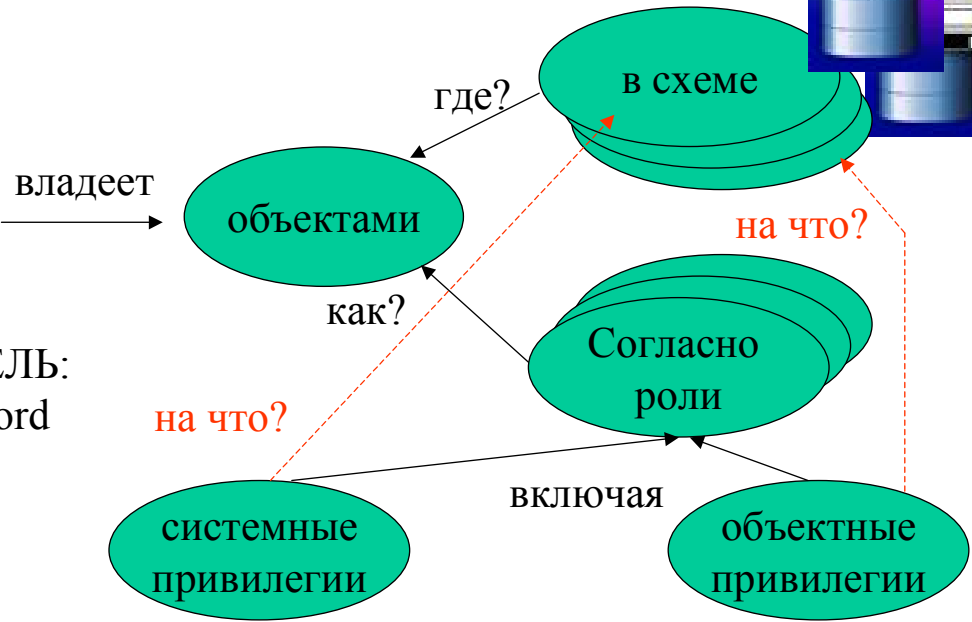
Рис.2.2. Создание пользователя через интерфейс EM.

! При первом подключении к установленной СУБД рекомендуется также сменить все пароли пользователей SYS и SYSTEM, если вы не сделали это раньше.

# СИСТЕМНЫЕ ПРИВЕЛЕГИИ И РОЛИ



ПОЛЬЗОВАТЕЛЬ:  
login & password



Вместо того, чтобы назначать привилегии непосредственно пользователям, необходимо назначать привилегии ролям, которые затем назначаются пользователям.



**Задание 1:**  
Вывести, используя SQL запрос, отчет по ролям и привилегиям пользователя “SCOTT”, пользователя “HR” и своей учетной записи.

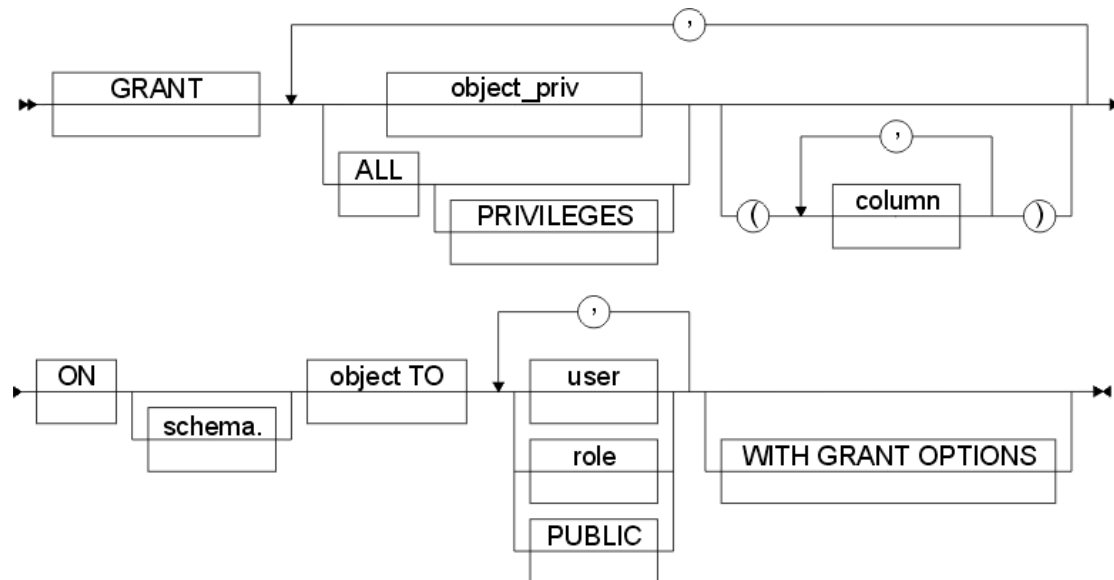
```
SQL> SELECT
2      USERNAME
3      , GRANTED_ROLE
4  FROM    USER_ROLE_PRIVS; (USER_SYS_PRIVS)
```

Для пользователя SCOTT	Для пользователя HR	Для своей учетной записи

## GRANT (привилегии доступа к объектам)

Предоставляет привилегии доступа к определенным объектам (таблицам, представлениям, синонимам, пакетам, процедурам и т.д.) пользователям и ролям.

### Установка объектных привилегий



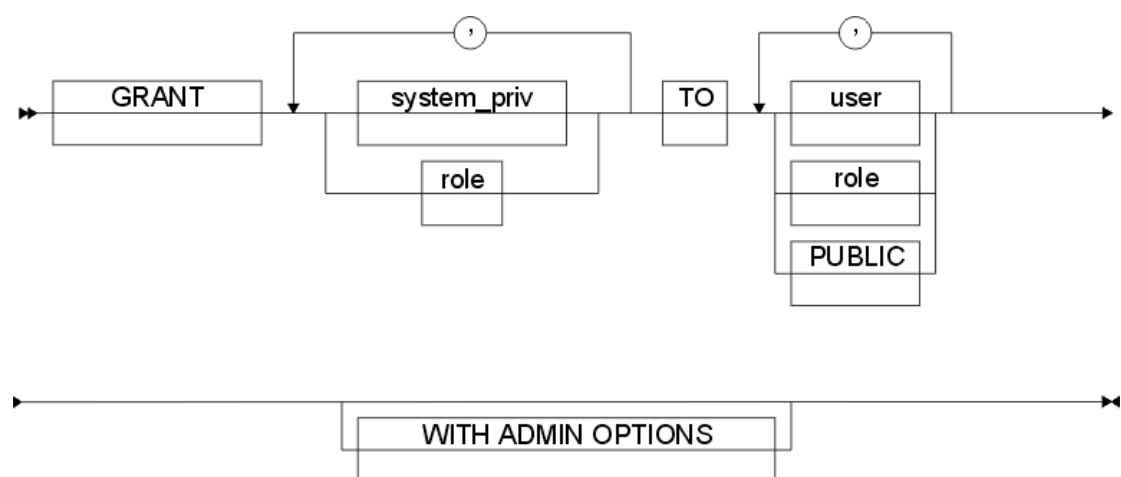
#### Задание 2:

Предоставить пользователю SCOTT объектные привилегии на объекты вашей схемы (объекты и виды привилегий выбрать самостоятельно)

Результат обращения к объекту Вашей схемы пользователем SCOTT до предоставления привилегии

Результат обращения к объекту Вашей схемы пользователем SCOTT после предоставления привилегии

### Установка системных привилегий

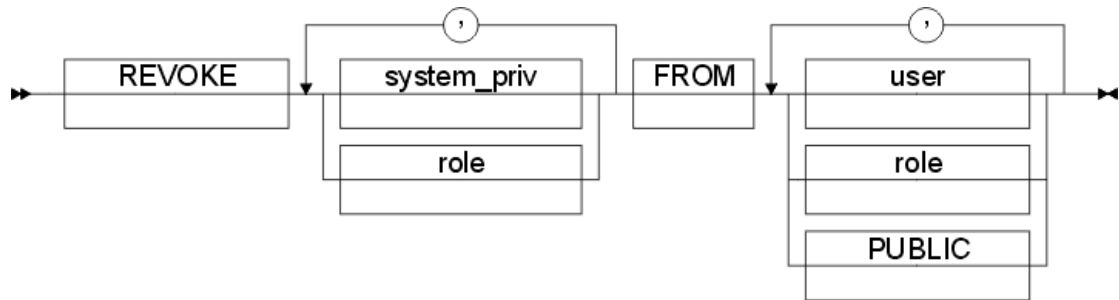


#### Задание 3:

Предоставить вашей учетной записи системные привилегии на работу с триггерами, функциями и процедурами.

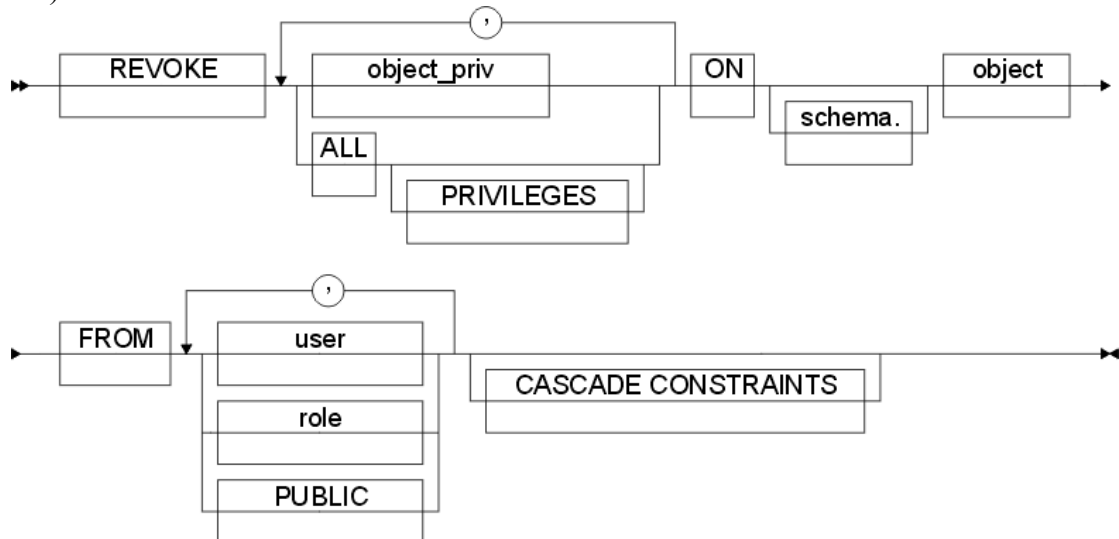
## REVOKE (системные привилегии и роли)

Отменяет системные привилегии и роли, ранее предоставленные пользователям и ролям. Действие, обратное команде GRANT (системные привилегии и роли).



## REVOKE (привилегии доступа к объектам)

Отменяет привилегии доступа к определенному объекту, ранее предоставленные пользователям и ролям. Действие, обратное команде GRANT (привилегии доступа к объектам).

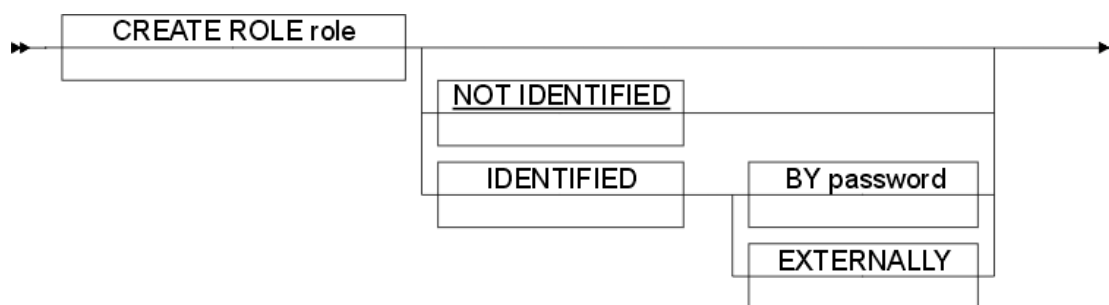


### Задание 4:

Отменить действия, реализованные в примере 2 и 3.

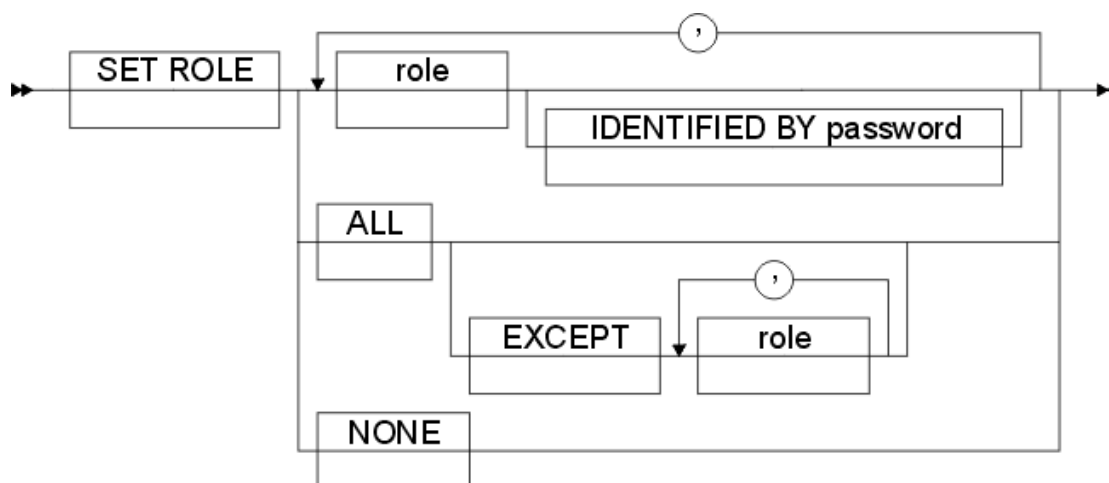


### CREATE ROLE - Создает роль



### SET ROLE - управление сеансом

Разрешает заданную роль в текущем сеансе и запрещает все другие роли пользователя. Должна выполняться в начале транзакции вместе с оператором SET TRANSACTION.



#### Задание 5:

Создать и назначить роль для своей учетной записи (набор объектных и системных привилегий выбрать самостоятельно).

### **Контрольные вопросы**

1. Основной инструментарий для работы с СУБД Oracle?
2. DDL и DML?
3. Создание базы данных?
4. Создание таблиц?
5. Элементы управления безопасностью и разграничения прав?

### **СПИСОК ЛИТЕРАТУРЫ**

1. Власов А.И., Лыткин С.Л., Яковлев В.Л. Краткое практическое руководство по языку PL/SQL - М.: Машиностроение. 2000. 64 с.
2. Сервер Oracle. Справочное руководство по языку SQL / Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДTeX, 1994.
3. Сервер ORACLE. Основные концепции/ Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДTeX, 1996.
4. Проектирование и эксплуатация конструкторско-технологических баз данных на основе СУБД Oracle/ Конспект лекций - М.: Москва, МГТУ им. Н.Э. Баумана, 2001, 120 с.

<b>Отчет по лабораторной работе № 3</b> <b>«Создание и модификация компонентов базы данных»</b>			
дата	Оценка (max 5)	Бонус за сложность	подпись

**Цели работы:**

Изучение основных компонентов и инструментария СУБД Oracle, создание и модификация основных элементов тестовой базы данных

**Задачи работы:**

- Создание и модификация основных компонентов БД (таблиц, ограничений и т.п.)

**Задание повышенной сложности (бонус за сложность – 10 баллов):**

Верификация схемы HR

**Краткий конспект теоретической части (ответы на контрольные вопросы)**

Виды ограничений \_\_\_\_\_

---

---

---

---

---

---

---

---

Создание индексов \_\_\_\_\_

---

---

---

---

---

---

---

---

Создание синонимов \_\_\_\_\_

---

---

---

---

---

---

---

---

Создание представлений \_\_\_\_\_

---

---

---

---

---

---

---

---

## CREATE/ALTER TABLE - СОЗДАНИЕ И ИЗМЕНЕНИЯ ТАБЛИЦ

В своей схеме создайте (а потом удалите) тестовую таблицу.

Синтаксическая диаграмма	Тело скрипта
<pre> graph LR     A[CREATE TABLE] --&gt; B[ ]     B --&gt; C[schema.]     B --&gt; D[table (]     D --&gt; E[column_element]     D --&gt; F[table_constraint]     E --&gt; G[)]     F --&gt; G     B --&gt; H[extent_specs]     H --&gt; I[TABLESPACE tablespace]     B --&gt; J[CLUSTER cluster (]     J --&gt; K[column]     K --&gt; L[)]     B --&gt; M[AS subquery]     B --&gt; N[ENABLE enable_clause]     B --&gt; O[DISABLE disable_clause]     </pre>	<p>Пример:</p> <pre> PROMPT Создание таблицы DOCTORS CREATE TABLE      doctors(     dc_nnn           NUMBER(12,0)   , dc_dc_nnn       NUMBER(12,0)   , dc_name          VARCHAR2(255)   , dc_cs_nnn        NUMBER(12,0)   , dc_diploma_number NUMBER(12,0)   , dc_specialty_nnn NUMBER(12,0)   , dc_shat_nnn      NUMBER(12,0)   , dc_calendar      NUMBER(12,0) ) TABLESPACE users;     </pre>
	<p><b>Вклеить текст собственного скрипта</b></p>

*\* Опция **DEFAULT** - Для столбца может быть задано значение по умолчанию. Если значение поля не указано при создании строки, то значение по умолчанию гарантирует нас от появления в поле пустого значения (или от ошибки, если на это поле задана опция **NOT NULL**). Значением по умолчанию может быть литерал или выражение, не содержащее имен других столбцов. Допускаются системные функции, такие как **SYSDATE** и **USER**.*

Вклеить скрипты создания собственных таблиц в своей схеме для модуля АСУ фирмы (информационная модель разработана в лабораторной работе 7-8 за 7 семестр)

## СОЗДАНИЕ ОГРАНИЧЕНИЙ

Oracle позволяет накладывать ограничения на таблицы и столбцы для обеспечения соблюдения некоторых правил как внутри таблицы, так и во взаимосвязи с другими таблицами БД. Допускаются ограничения на двух уровнях: Ограничения на таблицы могут быть определены на один или более столбцов таблицы и у задаются отдельно от описания столбцов таблицы. Ограничения на таблицу могут быть добавлены и после ее создания, а также временно отключены (см. команду ALTER TABLE в следующем разделе),

Все ограничения фиксируются в словаре данных. Каждому ограничению присваивается имя; удобнее давать ограничениям свои имена при их создании, это облегчит ссылку на ограничения в последующем. Если имя не задано, то система сама дает ограничению имя, генерируемое в форме: SYS\_Cn, где n -уникальный номер. Ключевое слово **CONSTRAINT** позволяет вам давать ограничениям свои имена.

### Типы ограничений

Вы можете задавать следующие типы ограничений:

- NULL/NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY (внешняя ссылка)
- CHECK

<b>Ограничение NOT NULL</b> Этот ограничитель запрещает указанным полям иметь пустые значения. Столбцы без ограничителя NOT NULL могут иметь пустые значения. NOT NULL - это одно из ограничений целостности, которое может быть задано в описании таблицы.	Привести пример задания данного типа ограничения в своей схеме
<b>Ограничитель UNIQUE</b> Этот квалификатор объявляет столбец или комбинацию столбцов уникальным ключом. Две разных строки таблицы не могут иметь одинаковое значение ключа. При этом допускаются пустые значения ключевого поля, если ключ задан на один столбец.	Привести пример задания данного типа ограничения в своей схеме  <pre>CREATE TABLE DEPT   (DEPTNO NUMBER,    DNAME VARCHAR2 (9 ) ,    LOC VARCHAR2 (10) ,    CONSTRAINT UNQ_DEPT_LOC UNIQUE (DNAME,    LOC) );</pre>
<b>Ограничитель PRIMARY_KEY</b> Как и уникальные ключи, первичный ключ (PRIMARY KEY) обеспечивает уникальность значений столбца или комбинации полей столбцов, на которые он задан. Также создается уникальный индекс для поддержания ограничения. Отличие от квалификатора <b>UNIQUE</b> заключается в том, что, во-первых, первичный ключ может быть в таблице только один; во-вторых, в нем запрещены пустые значения (NULL). Первичный ключ, таким образом, может служить для однозначной идентификации строк таблицы. Синтаксис задания на уровне таблицы:	Привести пример задания данного типа ограничения в своей схеме

<p>[<b>CONSTRAINT</b> имя ограничения]  <b>PRIMARY KEY</b> (столбец, столбец)  Синтаксис задания на уровне отдельного столбца:</p> <p>[<b>CONSTRAINT</b> имя ограничения]  <b>PRIMARY KEY</b>  Заметим, что одна и та же комбинация столбцов не может быть задана одновременно в первичном и уникальном ключах.</p>	
---	--

<p><b>Ограничитель FOREIGN KEY</b>  Внешние ключи (<b>FOREIGN KEY</b>) обеспечивают поддержание целостности данных как внутри одной таблицы, так и на уровне реляционных взаимосвязей различных таблиц. Внешний ключ используется в совокупности с уникальным или первичным ключом (в том смысле, что столбцы, на которые делаются ссылки, должны иметь ограничение <b>UNIQUE</b> или <b>PRIMARY KEY</b>).</p> <p>Синтаксис задания: На уровне таблицы:  [<b>CONSTRAINT</b> имя ограничения]  <b>FOREIGN KEY</b> (столбец, столбец)  <b>REFERENCES</b> таблица (столбец, столбец,...)</p> <p>На уровне отдельного столбца:  [<b>CONSTRAINT</b> имя ограничения]  <b>REFERENCES</b> таблица (столбец)</p> <p>Заметим, что слова "FOREIGN KEY" не указываются в ограничениях на уровне столбцов.</p>	<p>Привести пример задания данного типа ограничения в своей схеме</p>
--	---

*\* В результате ограничения строка не может быть удалена, их основной таблицы до тех пор пока из дочерней таблице не будут удалены связанные записи. Ограничение можно также задать таким образом, что при удалении информации из основной таблицы, будут удалены и соответствующие записи из дочерней таблицы. Это достигается заданием опции **ON DELETE CASCADE** в спецификации ограничения:*

<p><b>Ограничитель CHECK</b>  Ограничение <b>CHECK</b> задает условие, которому должно удовлетворять значение столбца в каждой строке таблицы. Условия, которые могут при этом задаваться, аналогичны заданию критерия поиска в предложении <b>WHERE</b> команды <b>SELECT</b>, но со следующими ограничениями:</p> <ul style="list-style-type: none"> <li>• подзапросы не допускаются;</li> <li>• ссылки на псевдостолбцы, такие <b>SYSDATE</b>, также запрещены.</li> </ul> <p>Синтаксис задания:  [<b>CONSTRAINT</b> имя ограничения] <b>CHECK</b> (условие)</p>	<p>Привести пример задания данного типа ограничения в своей схеме</p>
---	---

<p><b>Создание таблицы на базе запроса к другим таблицам</b></p> <p>Существует другая форма команды CREATE TABLE, в которой таблица создается на базе команды SELECT - запроса к существующим в БД таблицам:</p> <pre>CREATE TABLE DEPT [(имя столбца, ...)] AS SELECT текст-команды SELECT;</pre>	Привести пример задания данного типа ограничения в своей схеме
--	--

## УПРАВЛЕНИЕ ТАБЛИЦАМИ

<p><b>Изменение таблицы (ALTER TABLE)</b></p> <p>Для внесения изменений в описание таблицы используйте команду ALTER TABLE.</p> <p><b>ADD</b></p> <p>Для добавления к таблице нового столбца или навешивания на таблицу нового ограничения, задайте ключевое слово ADD:</p> <p><b>ПРИМЕР:</b> ALTER TABLE EMP ADD (FIRST_NAME CHAR(32));</p> <p><b>Предложение MODIFY</b></p> <p>В случае когда необходимо модифицировать описание столбца таблицы применяют ключевое слово MODIFY.</p> <p><b>ПРИМЕР:</b> ALTER TABLE имя таблицы MODIFY (столбец тип [NULL]);</p>	Привести пример задания данного типа ограничения в своей схеме
--	--

Существует ряд правил, которые необходимо соблюдать при добавлении и модификации описания столбцов:

- Вы не можете добавлять к столбцу опцию NOT NULL, если в нем есть пустые значения
- Вы не можете добавить новый столбец с опцией NOT NULL. Сначала создайте его без этой опции, заполните его значения во всех строках и затем добавьте опцию NOT NULL.
- Вы не можете уменьшить размер столбца или изменить его тип, если столбец содержит какие-то данные.
- Вы не можете исправлять с помощью опции MODIFY ограничения на таблицы, за исключением опции NULL/NOT NULL. Чтобы исправить другие ограничения, вы должны сначала удалить их, затем создать снова.

### Предложение DROP

Задавайте предложение DROP для удаления ограничения из описания таблицы.

Синтаксис:

ALTER TABLE имя таблицы		
DROP	CONSTRAINT имя ограничения	[CASCADE]
	PRIMARY KEY	
	UNIQUE (столбец, столбец,...)	

Опция CASCADE ставится в предложении DROP для удаления ограничений, связанных с тем, которое удаляется.

### Предложение ENABLE/DISABLE

Это предложение команды ALTER TABLE позволяет временно включать/выключать действие заданных ограничений, не удаляя их из описания таблицы. Синтаксис:

DISABLE	UNIQUE (столбец, столбец, ...)	[CASCADE]
ENABLE	PRIMARY KEY	
	CONSTRAINT имя ограничения	

Как и в предложении DROP, опция CASCADE означает, что все связанные ограничения также на время отключаются.

**ПРИМЕР:** ALTER TABLE DEPT DISABLE CONSTRAINT DEPT PRIM CASCADE;

## DROP TABLE - УДАЛЕНИЕ ТАБЛИЦ

<p><b>Синтаксис:</b> <b>DROP TABLE</b> имя_таблицы [<b>CASCADE CONSTRAINTS</b>]</p> <p>Удаление таблицы уничтожает все данные в ней и используемые ею индексы. Опция <b>CASCADE CONSTRAINTS</b> удаляет все внешние ссылки на столбцы таблицы. Без этой опции, при наличии внешних ссылок, таблица не будет удалена.</p> <p>Замечания:</p> <ul style="list-style-type: none"><li>• Все данные в таблице удаляются вместе с ней.</li><li>• Все синонимы (<b>SYNONYM</b>) и представления (<b>VIEW</b>) на таблицу остаются в словаре данных, но обращаться к ним при этом нельзя.</li><li>• Все связанные с таблицей незавершенные транзакции закрываются. Если в таблице есть блокированные строки, то удаления таблицы не происходит.</li><li>• Только создатель таблицы и системный администратор могут удалить ее.</li></ul>	<p>Привести пример удаления таблицы в своей схеме</p>
---	---

## COMMENT – ЗАДАНИЕ КОММЕНТАРИЕВ

<p><b>Синтаксис:</b> Выполняйте команду <b>COMMENT</b> для занесения в словарь данных комментариев о таблицах и их столбцах. Комментарий - строка, содержащая, до 255 символов.</p> <p><b>COMMENT ON TABLE EMP IS 'Employee Information';</b></p> <p>Чтобы завести комментарий на столбец таблицы <b>COMMENT ON COLUMN EMP.EMPNO IS 'Unique Employee Number';</b></p> <p>Чтобы уничтожить комментарий на столбец таблицы, необходимо выполнить команду <b>COMMENT ON COLUMN EMP.EMPNO IS</b></p> <p>Чтобы просмотреть комментарии на таблицы и их столбцы, следует задать запрос к соответствующим представлениям словаря данных: <b>USER_TAB_COMMENTS</b> или <b>ALL_TAB_COMMENTS</b> - комментарии на таблицы; <b>ALL_COL_COMMENTS</b> или <b>USER_COL_COMMENTS</b> - на столбцы таблиц</p>	<p>Привести пример задания комментариев в своей схеме</p>
---	---



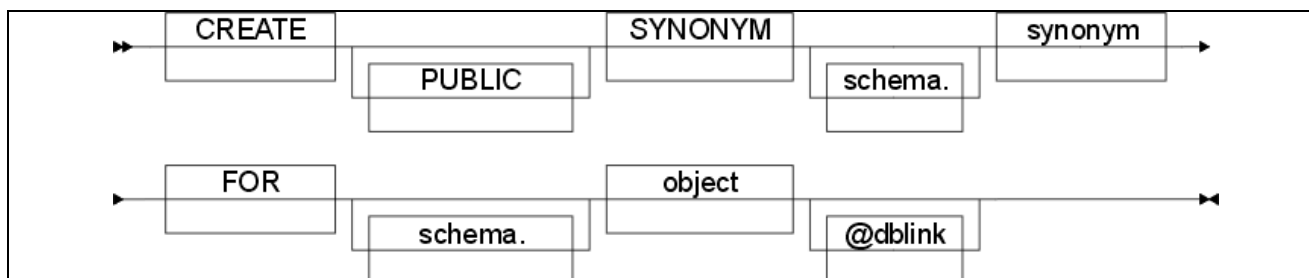
## RENAME ИЗМЕНЕНИЯ ИМЕН ОБЪЕКТОВ БД

RENAME старое_имя TO новое_имя;  <b>ВНИМАНИЕ:</b> все предложения/программы/отчеты, ссылающиеся на переименовываемую таблицу по имени, должны при этом быть откорректированы.	Привести пример в своей схеме
---	-------------------------------

## TRUNCATE TABLE - УДАЛИТЬ ВСЕ СТРОКИ ИЗ ТАБЛИЦЫ

То же самое может быть сделано командой DELETE, но очистка всей таблицы командой TRUNCATE более эффективна, поскольку производится не через механизм транзакций, а непосредственно, без возможности впоследствии восстановить данные путем отката транзакций. Синтаксис: <b>TRUNCATE TABLE</b> имя_таблицы [REUSE STORAGE] Опция <b>REUSE STORAGE</b> освобождает также память, зарезервированную за таблицей. По умолчанию очищенная область памяти остается за таблицей.	Привести пример в своей схеме
---	-------------------------------

## CREATE SYNONYM - СОЗДАЕТ СИНОНИМ



**Пример:** Сначала удаляем публичный синоним для таблицы DOCTORS, а потом его заново создаем.

PROMPT Создает синоним для таблицы, представления – вклеить свой  
PROMPT последовательности, хранимой процедуры или  
PROMPT функ., пакетной процедуры или др. синонима.

```
DROP PUBLIC SYNONYM doctors;
```

```
CREATE PUBLIC SYNONYM doctors FOR doctors;
```

```
CREATE PUBLIC SYNONYM имя FOR [хозяин.] имя_объекта;
```

Синоним может быть удален командой DROP:

```
DROP [PUBLIC] SYNONYM имя;
```

Для обращения к таблице другого пользователя вам необходимо к имени таблицы добавить слева в качестве префикса имя ее хозяина, отделенное точкой. Как альтернатива, вы можете создать синоним (второе имя) на таблицу или представление и обращаться к ней по этому имени. Только администратор БД (точнее, пользователь, обладающий привилегией **CREATE PUBLIC SYNONYM**) может создать синоним, доступный всем пользователям.

**CREATE SEQUENCE - СОЗДАНИЕ ГЕНЕРАТОРА ПОСЛЕДОВАТЕЛЬНОСТИ**

Создает новую последовательность для генерации первичных ключей.

```
CREATE SEQUENCE schema.sequence
```

- INCREMENT BY integer
- START WITH integer
- MAXVALUE integer
- NOMAXVALUE
- MINVALUE integer
- NOMINVALUE
- CYCLE
- NOCYCLE
- CACHE integer
- NOCACHE
- ORDER
- NOORDER

Привести пример в своей схеме

```
CREATE SEQUENCE s_dept  
INCREMENT BY 10 START WITH  
10 MAXVALUE 10000;
```

PROMPT создаем сиквенс и устанавливаем его начальное значение в единицу

```
CREATE SEQUENCE s_prj_nnn  
START WITH 1;  
*****
```

PROMPT получаем номер текущей задачи

```
SELECT s_prj_nnn.NEXTVAL FROM dual;
```

Сервер возвращает номер текущей задачи «N», поле чего создаем каталог V000N

**CREATE SEQUENCE** [пользователь.]имя последовательности [INCRENENT BY n] [START WITH n] [MAXVALUE n I NOMAXVALUE] [MINVALUE n | NOMINVALUE];

Все стоящие в прямоугольных скобках опции не обязательны для задания и разъясняются ниже.

пользователь	Хозяин последовательности. По умолчанию ставится имя пользователя, выполняющего команду CREATE SEQUENCE.
имя последовательности	Должно удовлетворять стандартным требованиям на имя объекта в языке SQL.
INCREMENT BY	Определяет интервал между соседними элементами последовательности. Если значение шага положительное, то последовательность возрастающая; если отрицательное - то последовательность убывающая. Можно задавать любое не равное нулю целое число. По умолчанию ставится значение 1.
START WITH	Задает первое значение последовательности при ее создании. По умолчанию задается 1 для возрастающих последовательностей и MAXVALUE для убывающих.

MINVALUE   NOMINVALUE	Минимальное значение последовательности, которое может быть сгенерировано (нижняя граница). По умолчанию задается 1 для возрастающих последовательностей и 10e27-1 для убывающих.
MAXVALUE   NOMAXVALUE	Максимальное значение, которое может быть сгенерирована. Задает верхнюю границу последовательности. По умолчанию определяется 1 для убывающих последовательностей и 10e27-1 для возрастающих. Любая попытка сгенерировать число, превышающее верхнюю границу последовательности, вызовет ошибку. (Если вы не используете последовательность для кодирования столбца с уникальными значениями, то при задании опции CYCLE генерация элементов последовательности может быть продолжена со стартового элемента).

После того как последовательность создана, она может использоваться для генерации уникальных числовых кодов.

### ***Генерация последовательных чисел с помощью псевдостолбца NEXTVAL***

Псевдостолбец **NEXTVAL** используется для генерации и выбора элементов заданной числовой последовательности. Когда задан псевдостолбец **NEXTVAL**, формируется очередное значение последовательности. Когда генерируется очередной элемент последовательности, ее значение возрастает на шаг независимо от того, произойдет ли закрытие или откат транзакции. Если два пользователя одновременно обращаются к одной последовательности, то один из них получит номер больший, другой - меньший, поскольку каждое обращение генерирует новый номер. Два пользователя никогда не сгенерируют одинаковых номеров, обращаясь к одной последовательности. Некоторые номера последовательности могут оказаться пропущенными, если пользователь не закрывает транзакцию или транзакция прерывается ненормально.

### ***Доступ к номерам последовательности с помощью псевдостолбца CURRVAL***

Для выбора номера последовательности, который уже был сгенерирован (текущий номер последовательности), используют псевдостолбец **CURRVAL**. Он возвращает последнее сгенерированное пользователем значение.

### ***Ограничения на псевдостолбцы NEXTVAL и CURRVAL***

Псевдостолбцы **NEXTVAL** и **CURRVAL** не могут быть заданы:

- в предложении **SELECT** на представления
- с ключевым словом **DISTINCT**
- одновременно с предложениями **ORDER BY**, **GROUP BY**, **CONNECT BY** или **HAVING** команды **SELECT**
- с операторами **UNION**, **INTERSECT**, **MINUS**
- в подзапросах.

Обращение к последовательностям аналогично обращению к таблицам, их параметры могут корректироваться командой **ALTER**; они могут быть удалены командой **DROP**.

Владелец последовательности может давать привилегии на доступ к ней другим пользователям.

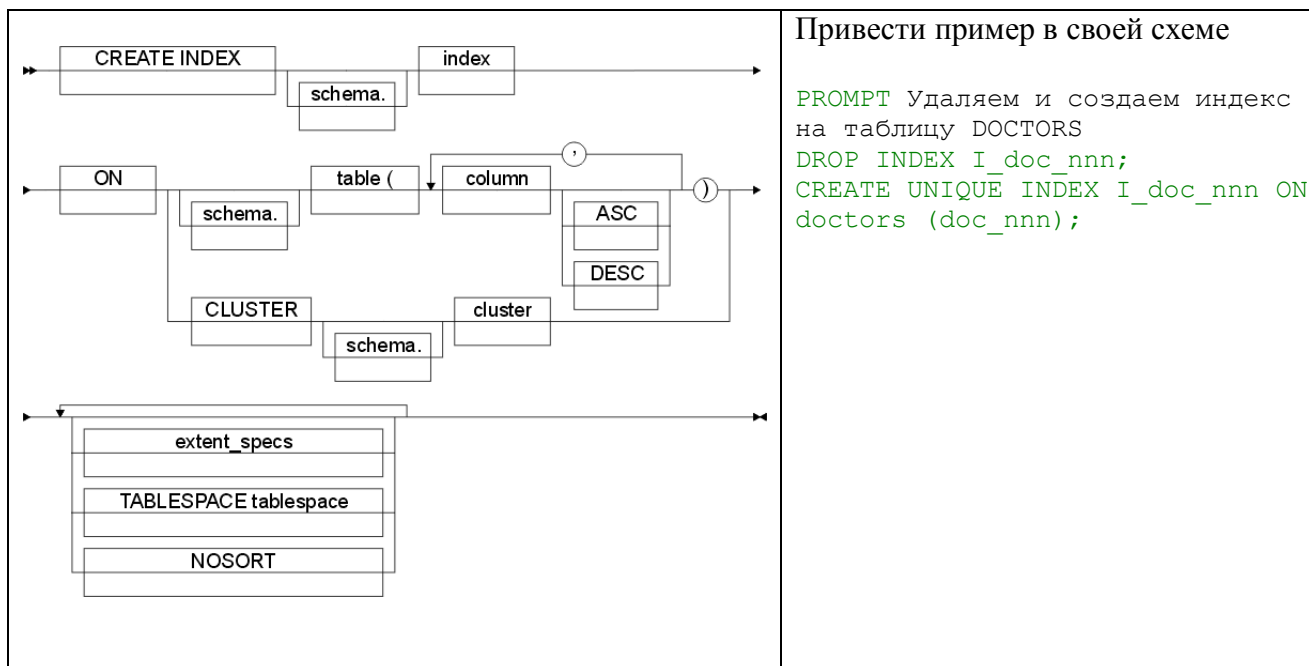
### ***Удаление последовательности***

Для удаления описания последовательности из словаря данных используйте команду **DROP SEQUENCE**. Синтаксис команды:

**DROP SEQUENCE** [пользователь.] имя\_последовательности;

Чтобы удалить последовательность, вы должны быть ее владельцем или обладать привилегией администратора.

## CREATE INDEX – СОЗДАНИЕ ИНДЕКСА



Индексы Oracle имеют два основных назначения:

1. Ускорить выбор данных по запросам к определенным столбцам.
2. Обеспечить уникальность значений столбца или группы столбцов, используемых обычно в качестве первичного ключа таблицы.

Для повышения производительности системы задание индексов настоятельно рекомендуется, и обычно одним из первых индексов создается индекс на первичный ключ.

Oracle8 автоматически создает индексы на столбцы, которые объявлены как PRIMARY KEY или UNIQUE.

Владелец таблицы может создавать на нее индексы. Любой пользователь Oracle, имеющий на таблицу привилегию INDEX, может также индексировать ее столбцы.

После того как индекс создан, Oracle использует его везде, где можно, для ускорения доступа к данным. Использование индекса производится системой автоматически и не требует от пользователя дополнительных действий; он даже не обязан знать, что индекс существует.

### Структура индекса Oracle

Oracle использует сбалансированные двоичные деревья для построения индексов. Это эффективный метод обеспечения примерно одинакового времени доступа к любой строке таблицы независимо от того, находится ли она в начале, середине или конце таблицы. Время доступа также практически не зависит от объема индексируемых данных.

Каждый построенный в БД Oracle индекс состоит из набора страниц памяти, организованных в виде бинарного дерева; каждая страничка содержит набор ключевых значений и указателей на нижестоящие в дереве страницы, и так до тех пор, пока ключевое значение не указывает на само значение поля. Oracle поддерживает эту структуру непосредственно в момент занесения и удаления строк. Пустые значения не хранятся в индексе и не занимают памяти.

### Типы индекса

Тип	Термин	Пояснение
Уникальный	UNIQUE	Обеспечивает уникальность значений заданного столбца или комбинации столбцов.
Не уникальный	ON UNIQUE	Обеспечивает самый быстрый доступ к данным при выполнении запроса (по умолчанию)
Простой	INGLE COLUMN	Индекс только на один столбец таблицы
Конкатенированный	ONCATEDENATED	Индекс на комбинацию столбцов (до 16) для обеспечения уникальности комбинации значений ил повышения эффективности доступа по сцепленному ключу.

<b>Создание индекса</b> Индексы могут быть созданы с помощью команды CREATE INDEX. Синтаксис: CREATE [UNIQUE] INDEX имя индекса ON таблица (столбец1,столбец2,...)	<b>Удаление индекса</b> Для удаления индекса выполните команду DROP INDEX. Синтаксис: DROP INDEX имя-индекса;
Привести пример в своей схеме	Привести пример в своей схеме

### Когда используется индекс?

Использование системой индексов частично зависит от того, какой в данный момент применяется оптимизатор. Oracle7 допускает два вида оптимизации выполнения SQL-запроса: на основе правил (**rule-based**) и на основе стоимостной оценки (**cost-based**).

### Использование индексов на основе правил

Oracle определяет, когда следует использовать индекс. Oracle имеет информацию о том, какие столбцы проиндексированы и какие типы индексов заданы. На основе этой информации система принимает решение согласно определенным правилам:

- Индексированный столбец должен быть указан в предложении WHERE.
- Индекс не будет использован, если ссылка на столбец в предложении WHERE является частью функции или выражения. В следующем примере индекс не будет использован, так как ссылка на столбец задана в функции: `select * from EMP where UPPER(ENAME)='JONES'`
- индекс на столбец не должна стоять в выражении.

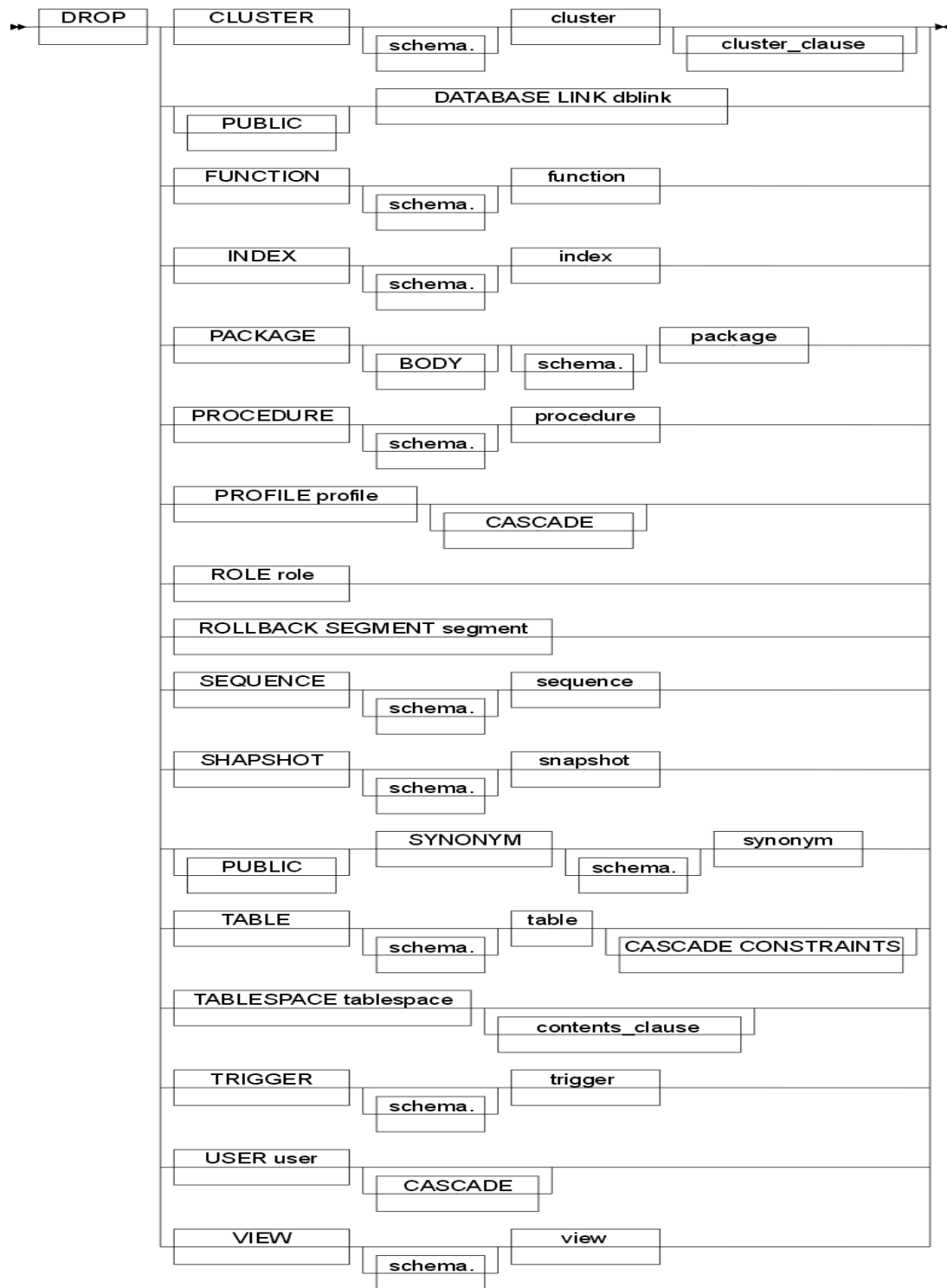
### Использование индексов на основе стоимостной оценки

Стоимостной оптимизатор строит план выполнения SQL-запроса путем вычисления стоимости альтернативных путей, используя хранящуюся в базе данных статистику, где это возможно. Обычно стоимостной оптимизатор дает наилучшие результаты оценки порядка задействования индексов и предпочтителен для применения в новых приложениях Oracle7. Тем не менее, этот оптимизатор может использовать индексы только при условии соблюдения приведенных выше правил.

Оптимизатор допускает включение в SQL-команды подсказок пользователя о том, следует ли задействовать определенный индекс.

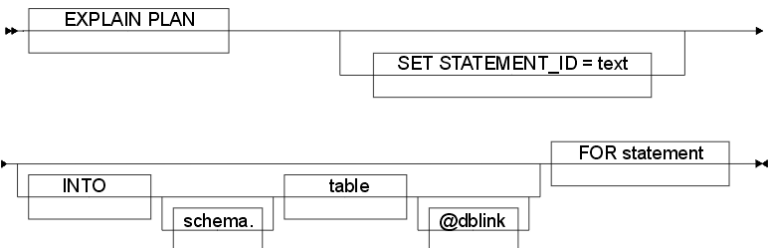
## DROP - УДАЛЕНИЕ ОБЪЕКТОВ И ОГРАНИЧЕНИЙ ИЗ БАЗЫ ДАННЫХ

<p>Эта команда. Для этого действия требуются соответствующие привилегии. Например, для удаления общего канала связи базы данных требуется привилегия</p> <p>Чтобы удалить таблицу из БД, выполните команду DROP TABLE.</p> <p>Синтаксис:  DROP TABLE имя_таблицы [CASCADE CONSTRAINTS]</p> <p>Удаление таблицы уничтожает все данные в ней и используемые ею индексы. Опция CASCADE CONSTRAINTS удаляет все внешние ссылки на столбцы таблицы. Без этой опции, при наличии внешних ссылок, таблица не будет удалена.</p> <p>Замечания:</p> <ul style="list-style-type: none"> <li>• Все данные в таблице удаляются вместе с ней.</li> <li>• Все синонимы (SYNONYM) и представления (VIEW) на таблицу остаются в словаре данных, но обращаться к ним при этом нельзя.</li> <li>• Все связанные с таблицей незавершенные транзакции закрываются. Если в таблице есть заблокированные строки, то удаления таблицы не происходит.</li> <li>• Только создатель таблицы и системный администратор могут удалить ее.</li> </ul>	Привести пример в своей схеме
--	-------------------------------



Приведите примеры удаления разных объектов вашей схеме:

**EXPLAIN PLAN** - Описывает каждый шаг плана выполнения оператора SQL и помещает (если задано) это описание в указанную таблицу

 <pre> EXPLAIN PLAN   SET STATEMENT_ID = text   INTO schema. table @dblink   FOR statement </pre>	Привести пример в своей схеме
--	-------------------------------

### **ROLLBACK (управление транзакцией)**

Отменяет все изменения, сделанные до контрольной точки. Отменяет все изменения, произведенные в текущей транзакции, если контрольная точка не задана.

 <pre> ROLLBACK   WORK   TO   SAVEPOINT savepoint   FORCE 'text' </pre>	Привести пример в своей схеме
---	-------------------------------

### **Контрольные вопросы**

1. Виды ограничений?
2. Создание индексов?
3. Создание синонимов?
4. Удаление объектов БД?
5. Формирование плана выполнения SQL оператора?
6. Управление транзакциями?

### **СПИСОК ЛИТЕРАТУРЫ**

1. Власов А.И., Лыткин С.Л., Яковлев В.Л. Краткое практическое руководство по языку PL/SQL - М.: Машиностроение. 2000. 64 с.
2. Сервер Oracle. Справочное руководство по языку SQL / Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1994.
3. Сервер ORACLE. Основные концепции/ Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1996.
4. Проектирование и эксплуатация конструкторско-технологических баз данных на основе СУБД Oracle/ Конспект лекций - М.: Москва, МГТУ им. Н.Э. Баумана, 2001, 120 с.

Отчет по лабораторной работе № 4 «Операторы DML»			
дата	Оценка (max 5)	Бонус за сложность	подпись

### Цели работы:

## Изучение операторов DML

### Задачи работы:

-разработка примеров простейших программ с использованием DML для схемы HR

**Задание повышенной сложности (бонус за сложность – 10 баллов):**

- выполнение задание 5.8.1., 5.8.2, 5.8.3.

## Краткий конспект теоретической части (ответы на контрольные вопросы)

SELECT

---

---

---

---

INSERT \_\_\_\_\_

---

---

---

---

---

UPDATE

UPDATE \_\_\_\_\_

---

---

---

---

---

---

DELETE

---

---

---

---

---



## Язык DML

Язык обработки данных DML позволяет добавлять, изменять, удалять информацию в базе данных. В полном соответствии со своим названием, язык DML предназначен для работы с информационным содержанием базы данных. Операторы языка DML представляют собой SQL-команды, позволяющие изменять и дополнять хранящуюся в базе данных информацию. Наиболее используемыми DML-операторами являются INSERT, UPDATE, DELETE и SELECT

## Стандартизация SQL

Процедура разработки международных стандартов в соответствии с регламентом деятельности ISO предусматривает несколько стадий, каждая из которых заключается в подготовке очередной версии документа или комплекта документов, содержащих спецификации проекта стандарта. После подготовки некоторой версии проекта по ней в течение установленного срока проводится голосование (рассмотрение) среди определенного круга экспертов, высказывающих свои замечания. Для принятия решений по этим замечаниям и внесения необходимых поправок в текст проекта созываются редакционные заседания подразделения ISO, ответственного за разработку стандарта.

**Серия документов, рождающихся в процессе разработки спецификаций стандарта, включает** (в порядке их создания):

Рабочий проект (Working Draft, WD)

Проект комитета (Committee Draft, CD)

Заключительный проект комитета (Final Committee Draft, FCD)

Проект международного стандарта (Draft International Standard, DIS)

Заключительный проект международного стандарта (Final Draft International Standard, FDIS)

Международный стандарт (International Standard, IS).

Часть стандарта	Сроки принятия
Часть 10, SQL/OLB	Середина 2000 (Связывание для объектных языков)
Поправка 1, SQL/OLAP	Конец 2000 (Интерактивная аналитическая обработка)
Часть 9, SQL/MED	2001 (Управление внешними данными)
Часть 7, SQL/Temporal	2003?
SQL:200x	2003?

Часть 10, SQL/OLB – Голосование по **Заключительному проекту комитета** завершилось в начале 1999 г. с рядом существенных замечаний. В январе 2000 года в Санта-Фе (штат Нью-Мексико, США) были успешно разрешены вопросы, касающиеся последних замечаний, и участники рекомендовали перейти к стадии голосования по **Заключительному проекту международного стандарта** (Final Draft International Standard).

В результате опубликована новая часть стандарта SQL:1999, идентифицируемая как ISO/IEC 9075-10:2000.

### Новые возможности SQL:1999:

- поддержка позиционируемых (scrollable) и удерживаемых (holdable) курсоров;
- поддержка типов, определяемых пользователем;
- поддержка обновляемых результирующих наборов и пакетных обновлений;
- расширенные средства подключения (connection) и транзакций.

Часть 9, SQL/MED- В конце 1998 года в ISO было инициировано голосование по **проекту комитета**. Голосования по **Заключительному проекту международного стандарта**-середина 2001 года.

SQL/MED опубликован как международный стандарт ISO/IEC 9075 9:2001 в 2001 г.

### **Новые возможности SQL:1999:**

обеспечивает связь между SQL-сервером (т.е. некоторой «локальной» SQL системой управления базами данных) и другим объектом, называемым адаптером внешних данных (foreign-data wrapper). Локальный SQL-сервер и этот адаптер обмениваются информацией, которая позволяет локальному SQL-серверу осуществлять выборку и, возможно, вставку, обновление и удаление данных, которые фактически управляются некоторым внешним сервером (foreign server).

Поправка 1, SQL/OLAP- голосование по предложенному **Заключительному проекту поправки** (Final Proposed Draft Amendment, FPDAM) к SQL/OLAP было инициировано в начале 2000 г. Результат: ISO/IEC 9075-1/AMD1:2000. Этот документ вносит поправки в несколько частей SQL:1999, и поэтому было бы вполне уместно рассматривать его как поправку к части 1, SQL/Framework.

### **Новые возможности SQL:1999:**

- SQL/OLAP содержит ряд новых числовых функций, например:
- LN ;
- EXP;
- POWER (возводит один аргумент в степень, указываемую другим аргументом);
- SQRT (извлекает квадратный корень из аргумента); FLOOR и CEILING (возвращает наибольшее целое, не превышающее значения аргумента, и наименьшее целое, большее или равное значению аргумента);
- функция ранжирования (возвращает ранжирование аргумента на мультимножестве значений);
- функция процентиля (возвращает ранжирование аргумента как значения его процентиля на мультимножестве значений).

Часть 7, SQL/Temporal -Работа над SQL/Temporal приостановилась уже три года назад по трем причинам. Во-первых, не проявляли большого энтузиазма поставщики продуктов SQL – или, в действительности, их заказчики – в вопросе о включении средств поддержки временных данных в механизмы систем управления базами данных. Во-вторых, специалисты, занимающиеся стандартами SQL были вынуждены сконцентрироваться на завершении проекта SQL3. Наконец, имеется фундаментальное расхождение между двумя лагерями относительно того, какие в точности возможности должен предоставлять SQL/Temporal и каким образом эти возможности следует обеспечивать (т.е., и синтаксис и семантика).

Когда разработка SQL3 завершена, участники работы выяснили, что рынок проявляет все более возрастающую информированность относительно преимуществ поддержки временных данных и, по крайней мере, некоторые из поставщиков продуктов SQL стали интересоваться этой проблемой. Поэтому работы над SQL/Temporal недавно начали возрождаться, и можно надеяться увидеть публикацию этой дополнительной части стандарта SQL, вероятно, в течение 2002-2003 гг.



Рис.1.

Итак: **SQL:**

- язык доступа к реляционным базам данных, в том числе и к РБД Oracle;
- может быть использован в любом средстве создания приложений Oracle, где требуется доступ к БД.

**PL/SQL:**

- процедурный язык для написания подпрограмм и манипулирования данными вне БД;
- может включать ряд команд языка SQL при необходимости обращения к БД;
- не имеет самостоятельных операторов ввода/вывода, но доступен в любом продукте CDE и прекомпиляторах Pro\*Oracle для написания подпрограмм. Также доступен на самом сервере Oracle (при наличии процедурной опции). "SQL-PLUS
- продукт Oracle, в котором могут использоваться языки SQL и PL/SQL. Представляет собой среду для задания и выполнения команд языка SQL, а также процедур на языке PL/SQL;
- имеет свой собственный набор команд для настройки программной среды и форматирования выходных результатов.

Таким образом, SQL и PL/SQL - это языки, используемые в продуктах Oracle.

Прежде чем перейти к рассмотрению основных синтаксических конструкций языка (т.е. описание алфавита и правил построения различных конструкций языка из символов алфавита и более простых конструкций языка) введем понятие синтаксической диаграммы.

**Синтаксическая диаграмма** – это графическое представление правил построения конструкций языка в наглядной форме. При этом принято, символы алфавита изображать блоками в овальных рамках, названия конструкций – в прямоугольных, а правила построения конструкций в виде линий со стрелками (часто овальные и прямоугольные рамки для упрощения опускают). Также выделяют осевую линию синтаксической конструкции, на которой размещаются все обязательные символы алфавита и конструктивные элементы. Все необязательные элементы и символы размещаются ниже или выше осевой линии конструкции. Если конструкция имеет значительную длину, то осевая линия переносится соответственно на следующую строку. Существуют и альтернативные формы отображения синтаксиса языка, например форма Бэкуса-Наура (БНФ) и т.п. [2].

## Задание SQL-команд

При написании SQL-команд важно помнить ряд простых правил, позволяющих конструировать корректные, легко читаемые и редактируемые предложения:

- SQL-команды могут быть на одной или нескольких строках.
- Отдельные компоненты команды (предложения) обычно задаются на разных строках.
- Допускается использование табуляторов.
- Отдельные слова в команде не могут быть разделены на несколько строк с переносом.
- SQL-команды могут задаваться как прописными, так и строчными буквами, если не задано специальных ограничений.
- SQL-команда вводится в ответ на системную подсказку (по умолчанию это: SQL) ), и строки команды нумеруются.


Каждая из нижеследующих команд верна, но читаемость их различна:

SELECT * FROM emp;	SELECT * FROM EMP;	SELECT * FROM emp;
--------------------	--------------------------	-----------------------

**Типы данных обрабатываемых СУБД Oracle представлены в таблице.**

*Таблица. Типы обрабатываемых данных.*

<i>Тип данных</i>	<i>Описание</i>
CHAR(size)	Символьная строка фиксированной длины, имеющая максимальную длину <i>size</i> символов. Длина по умолчанию 1, максимальная -255.
CHARACTER (size)	То же, что и CHAR.
DATE	Правильные даты в интервале от 1 января 4712 года до н.э. до 31 декабря 4712 года.
LONG	Символьные данные переменной длины до 2 Гигабайт.
LONG RAW	Двоичные данные переменной длины вплоть до 2 Гигабайт.
CLOB	Character Long
BLOB	Binary Long
MLSLABEL	Используется в Trusted ORACLE.
NUMBER(p,s)	Число, имеющее <i>p</i> значащих цифр и масштаб <i>s</i> . <i>p</i> может быть от 1 до 38. <i>s</i> может принимать значения от -84 до 127.
RAW(size)	Двоичные данные длиной <i>size</i> байт. Максимальное значение для <i>size</i> - 2000 байт. Параметр <i>me</i> для RAW обязателен.
RAW MLSLABEL	Используется в Trusted ORACLE.
ROWID	Значения псевдостолбца ROWID.
VARCHAR2(size)	Символьная строка переменной длины, имеющая максимальную длину <i>size</i> символов. Длина по умолчанию 1, максимальная - 2000.
VARCHAR(size)	То же что и VARCHAR2.


	<b>Задание 5.1.:</b> Выведите структуру таблиц схемы HR в SQL+ с указанием типов данных атрибутов и наложенных ограничений	DESC <имя таблицы>

## Работа с псевдосто́лбцами

Извлекать данные можно также и из псевдосто́лбцов, которые похожи на столбцы таблиц, но их значения нельзя изменять при помощи операторов DML.

*Таблица. Псевдосто́лбцы.*

Название столбца	Возвращаемое значение
Sequence.CURRVAL	Текущее значение sequence в данном сеансе (sequence.NEXTVAL должен быть выбран).
Sequence.NEXTVAL	Следующее значение sequence в текущем сеансе.
[table.]LEVEL	1 - для корня дерева, 2 - для узлов второго уровня и так далее. Используется в операторе SELECT в иерархических запросах.
[table.]ROWID	Значение, которое идентифицируют строку в таблице table уникальным образом. Значения псевдосто́лбца ROWID имеют тип данных ROWID, а не NUMBER и не CHAR.
ROWNUM	Порядковый номер строки среди других строк, выбираемых запросом. ORACLE выбирает строки в произвольном порядке и приписывает значения ROWNUM, прежде чем строки будут отсортированы предложением ORDER BY.

	<b>Задание 5.2.:</b> Выполните обращение к псевдосто́лбцам таблиц в схеме HR	результат
SELECT имя сиквенса.CURRVAL FROM dual		
SELECT имя сиквенса.NEXTVAL FROM dual		
SELECT [table.]LEVEL FROM [table]		
SELECT [table.] ROWID FROM [table]		
SELECT ROWNUM FROM [table]		

## Операции и их приоритеты

Арифметические операции	Символьные операции	Логические операции	Операции сравнения
<b>+ - (один операнд)</b>		<b>NOT</b>	=
<b>* /</b>		<b>AND</b>	!= ^= ~= <>
<b>+ - (два операнда)</b>		<b>OR</b>	> >= < <=
			<b>IN</b>
			<b>NOT IN</b>
			<b>ANY, SOME</b>
			<b>ALL</b>

**Приоритеты операций.** При вычислении выражения, содержащего несколько операций, ORACLE сначала выполняет операции с более высоким приоритетом. Операции, приведенные на одной и той же строке, имеют одинаковые приоритеты.

Замечание: В выражениях можно использовать круглые скобки, чтобы изменять последовательность выполнения операций, предписываемую приоритетом. Выражения, заключенные в скобки, ORACLE вычисляет в первую очередь. Без скобок операции с одинаковым приоритетом ORACLE выполняет слева направо.



### Задание 5.3:

Выбрать сведения о сотрудниках, имеющих оклады, начиная с 2-ого ранга по сумме оклада и кончая 5-ым. (Ранг 1 имеет сотрудник или сотрудники с наибольшим окладом, то есть находящиеся на первом месте по величине оклада) В задании используется таблица HR.EMPLOYEES

Выводимые столбцы:

1. Имя (First\_name)
2. Фамилия (Last\_name)
3. Название отдела
4. Должность (Job\_id)
5. Оклад (Salary)
6. Позиция по окладу (ранг)

Строки в выходной таблице должны быть упорядочены по позиции (рангу) оклада и по названию отдела и должности в возрастающем порядке.

#### Дополнительные требования к выполнению:

Не использовать аналитические функции. Для ранжирования использовать псевдостолбец ROWNUM.

```
SELECT
    EMP.First_name "Имя"
  , EMP.Last_name "Фамилия"
  , DEPT.Department_name "Отдел"
  , EMP.Salary    "Оклад"
  , EMP.Job_id
  , D_S.Position  "Позиция"
FROM
    EMPLOYEES EMP
  , DEPARTMENTS DEPT
  , (select
        Rownum Position
      , Sal
    from (
        select
            Distinct Salary Sal
        from EMPLOYEES
        order by Sal desc
    )
    ) D_S
WHERE
    D_S.Sal = EMP.Salary
and DEPT.Department_id = EMP.Department_id
and D_S.Position between 2 and 5
order by
    D_S.Position
  , DEPT.Department_name
  , Job_id;
/
```


Результат:

строк

## Список, зарезервированных слов SQL

Язык SQL включает зарезервированные слова, имеющие определенное значение в операторах SQL. Эти слова нельзя использовать в качестве имен объектов базы данных.

ACCESS*	DEFAULT*	INTEGER	OPTION*	START*
ADD*	DELETE*	INTERSECT*	OR*	SUCCESSFUL
ALL*	DESC*	INTO*	ORDER*	SYNONYM
ALTER*	DISTINCT*	IS*	PCTFREE*	SYSDATE
AND*	DROP*	LEVEL*	PRIOR*	TABLE*
ANY*	ELSE*	LIKE*	PRIVILEGES	THEN*
AS*	EXCLUSIVE	LOCK	PUBLIC*	TO*
ASC*	EXISTS*	LONG	RAW	TRIGGER
AUDIT	FILE	MAXEXTENTS	RENAME*	UID
BETWEEN*	FLOAT	MINUS*	RESOURCE*	UNION*
BY*	FOR*	MODE	REVOKE	UNIQUE*
CHAR*	FROM*	MODIFY	ROW	UPDATE*
CHECK*	GRANT*	NOAUDIT	ROWID	USER
0CLUSTER*	GROUP*	NOCOMPRESS*	ROWLABEL	VALIDATE
COLUMN	HAVING*	NOT*	ROWNUM*	VALUES*
COMMENT	IDENTIFIED*	NOWAIT	ROWS	VARCHAR*
COMPRESS*	IMMEDIATE	NULL*	SELECT*	VARCHAR2*
CONNECT*	IN*	NUMBER*	SESSION	VIEW*
CREATE*	INCREMENT	OF*	SET*	WHENEVER
CURRENT*	INDEX*	OFFLINE	SHARE	WHERE*
DATE*	INITIAL	ON*	SIZE*	WITH*
DECIMAL*	INSERT*	ONLINE	SMALLINT	

	<b>Задание 5.4.:</b> Попробуйте создать публичный синоним (или иной объект БД) с именем соответствующим зарезервированному слову	Результат и его анализ

### Комментарии

Комментарии, заданные ограничителями '/' и '/\*', могут стоять в любом месте оператора SQL:

ALTER USER petrov /\* Это комментарий \*/ IDENTIFIED BY petr;

Можно использовать стандартные комментарии ANSI. Все символы после двух дефисов до конца строки игнорируются.

ALTER USER petrov /\* Это комментарий продолжен до конца строки IDENTIFIED BY petr;

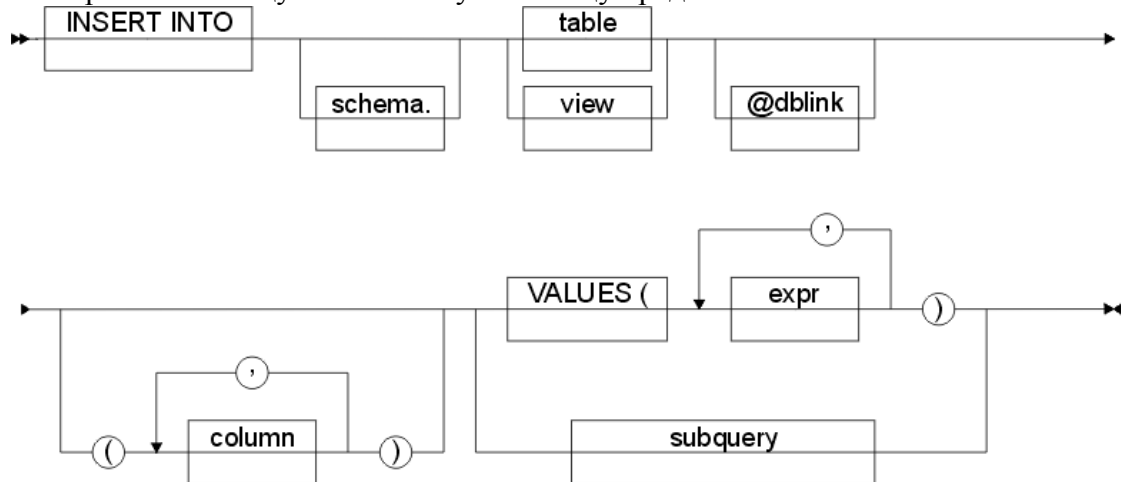
## Оператор INSERT - Занесение/изменение данных в таблицах/

Команда INSERT используется для ввода новых строк в таблицу. Синтаксис команды INSERT:

**INSERT INTO имя таблицы (столбец, столбец, ...) VALUES (значение, значение, ... );**

Допускается пропуск списка столбцов после имени таблицы, если список значений соответствует количеству и порядку столбцов в описании таблицы. Рекомендуется всегда указывать список столбцов. Если список не задан, то при изменении структуры таблицы необходимо будет исправлять и программное обеспечение. Символьные значения и даты должны быть взяты в одинарные кавычки.

Вставить строки в таблицу или в базовую таблицу представления.



Условные обозначения на синтаксических диаграммах:

<b>INSERT INTO</b>	Наименование команды
<b>HEMA</b>	Используемая схема
<b>TABLE (VIEW)</b>	Таблица или представление
<b>COLUMNS</b>	Наименование столбца
<b>VALUES</b>	Значение ячейки
<b>alias</b>	Псевдоним (базы данных, таблицы, столбца)
<b>Column_element</b>	Определение столбца таблицы
<b>Conditions</b>	Условное выражение, которое после вычисления может принимать значение TRUE, FALSE или UNKNOWN
<b>Constraint</b>	Имя, идентифицирующее правило целостности и хранящееся в словаре данных
<b>@dblink</b>	Канал связи с БД. (Полный путь к базе данных)
<b>expr</b>	Любое выражение или описание
<b>subquery</b>	Запрос в каком либо другом операторе

### Ввод строк из других таблиц

Эта форма команды INSERT позволяет заносить в таблицу набор строк, найденных по запросу команды SELECT. Список столбцов в команде SELECT должен соответствовать списку столбцов предложения INTO.



**ПРИМЕР:** Копирования сведений о служащих 10-го отдела в таблицу D10 HISTORY.

```
insert into D10_HISTORY (EMPNO, ENAME, SAL, JOB, HIREDATE)
select EMPNO, ENAME, SAL, JOB, HIREDATE
from EMP where DEPTNO = 10;
```

Обратите внимание на то, что ключевое слово 'VALUES' здесь не задано.



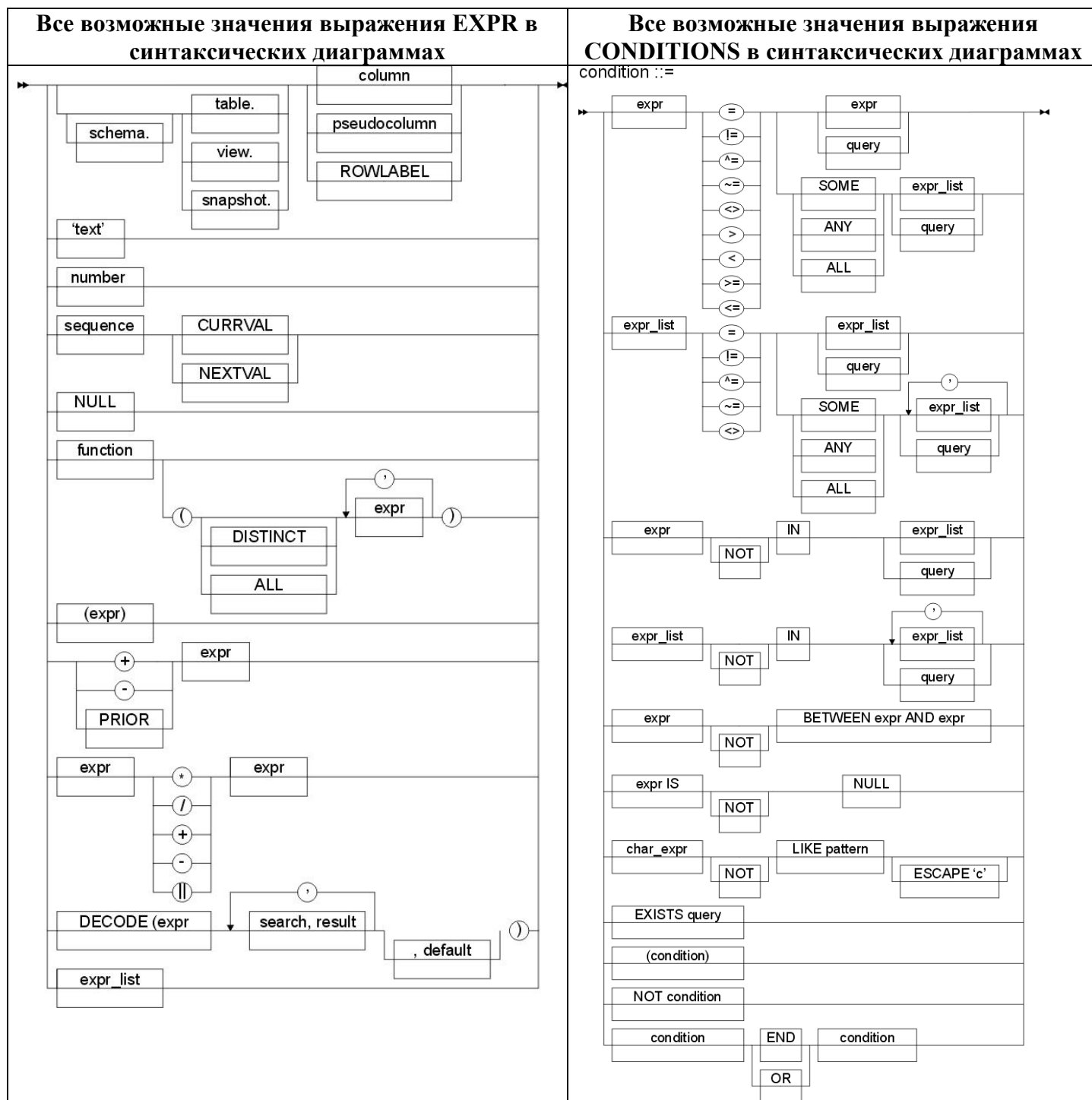
**Задание 5.5.:**

Приведите пример sql-скрипта вставки данных в таблицы вашего модуля, созданные в ЛР №4



**Задание:**

Выполните проверку правильности вставки данных



**Условные обозначения:**

(+) - Указывает, что предшествующий столбец является столбцов внешнего соединения.

\* - Указатель на выбор всех столбцов из таблицы или представления.

PRIOR - Используется в иерархическом запросе для определения зависимости между родительскими и дочерними строками.

ALL - Обеспечивает выбор всех строк в запросе (в том числе и повторяющихся).

DISTINCT - Удаляет повторяющиеся строки из результата запроса.

### Условные обозначения:

NOT	Отрицание логического выражения
AND	Объединение нескольких логических выражений: TRUE когда все условия истинны.
OR	Объединение нескольких логических выражений, TRUE когда хотя бы одно из выражений истинно.
END	Символ конца списка элементов.
ALL	TRUE, если операция сравнения выполняется для всех элементов списка выражений или множества значений, возвращаемых запросом.
NOT BETWEEN x AND y	[Не]Больше или равно X и меньше или равно Y
IN	Равно любому элементу множества или подзапроса.
NOT IN	Не равно любому элементу множества или подзапроса.
IS [NOT] NULL	Проверка на непустое значение
ANY, SOME	Результат TRUE, если указанная операция сравнения выполняется хотя бы для одного элемента списка выражений или множества значений, возвращаемых запросом.
[NOT] EXISTS	TRUE, если подзапрос [не] возвращает хотя бы одну строку.
[NOT] LIKE p	TRUE, если не совпадает с заданным шаблоном: % - любая последовательность символов, - любой символ.

#### Пример 1:

```
INSERT INTO doctors (dc_name) VALUES('ИВАНОВ')
```

#### Пример 2:

```
INSERT INTO doctors (dc_name,  
                    dc_shtat_nnn)  
VALUES('ИВАНОВ',  
      'психотерапевт')
```

#### Пример 3:

```
INSERT INTO doctors (dc_name,  
                    dc_shtat_nnn)  
VALUES('ИВАНОВ',  
      SELECT shtat_nnn FROM shtat  
      WHERE UPPER(shtat_name) = 'ПСИХОТЕРАПЕВТ'))
```

#### Пример 4: В качестве примера рассмотрим вставку данных в таблицу "Праздничные дни" (PRAZDNIKI)

```
insert into prazdniki values ('понедельник');  
insert into prazdniki values ('вторник');  
insert into prazdniki values ('среда');  
insert into prazdniki values ('четверг');  
insert into prazdniki values ('пятница');  
insert into prazdniki values ('суббота');  
insert into prazdniki values ('воскресенье');
```

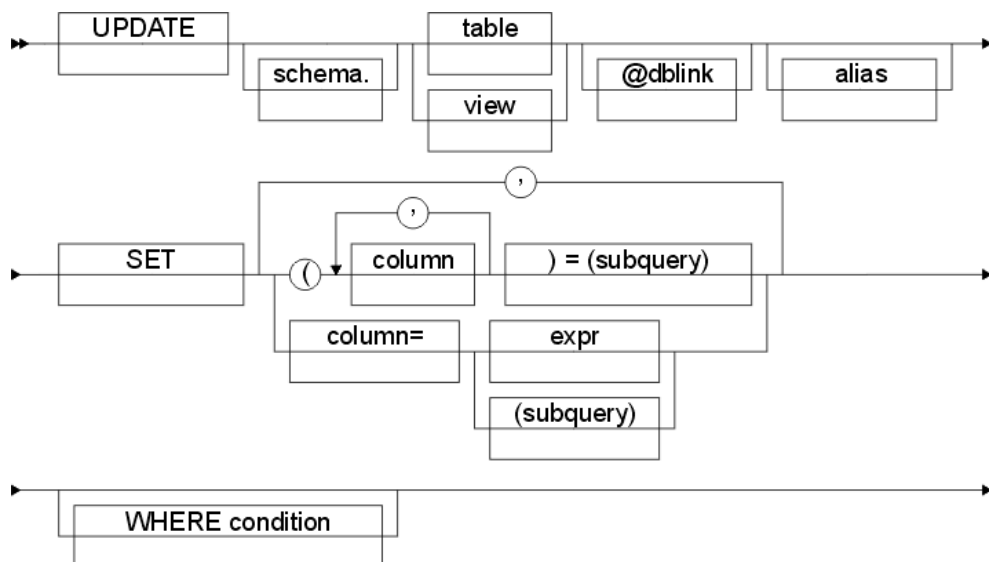
## Изменение строк (UPDATE)

Команда UPDATE позволяет исправлять содержимое строк в таблице. Синтаксис:

UPDATE таблица [заменитель] SET столбец [, столбец .... ] = (выражение, подзапрос)  
[WHERE условие];

### UPDATE

Изменяет существующие значения в таблице или в представлении (View).



### Пример:

UPDATE doctors set dc\_shtat\_nnn = 11 WHERE dc\_name = 'ИВАНОВ';

Результат 'ИВАНОВ' переводится в отдел с номером 11, но таких ивановых может быть несколько, следовательно такое обновление должно осуществляться не по имени пользователя, а по его NNN.



#### Задание 5.6.:

Приведите пример sql-скрипта для изменения значений ряда атрибутов в таблицах вашего модуля, созданных в ЛР №4



#### Задание :

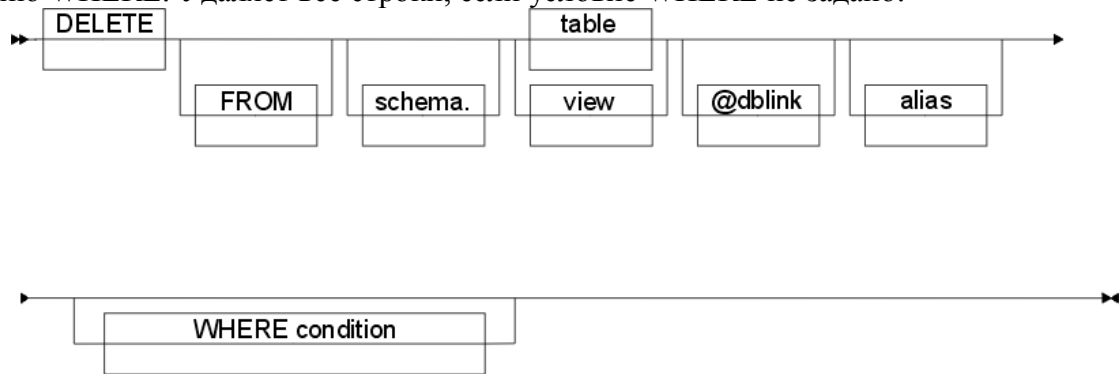
Выполните проверку правильности изменения данных

## Удаление строк из таблицы (DELETE)

Команда DELETE позволяет удалять одну или более строк из таблицы. Синтаксис:  
DELETE FROM таблица [WHERE условие];  
При отсутствии предложения WHERE будут удалены все строки таблицы.

### DELETE

Удаляет строки из таблицы или из базовой таблицы представления, удовлетворяющие условию WHERE. Удаляет все строки, если условие WHERE не задано.



### Пример:

Удаляем все записи из таблицы Праздничных дней. delete from prazdniki;



#### Задание 5.7.:

Приведите пример sql-скрипта для удаления строк в таблицах вашего модуля, созданных в ЛР №4 по условию

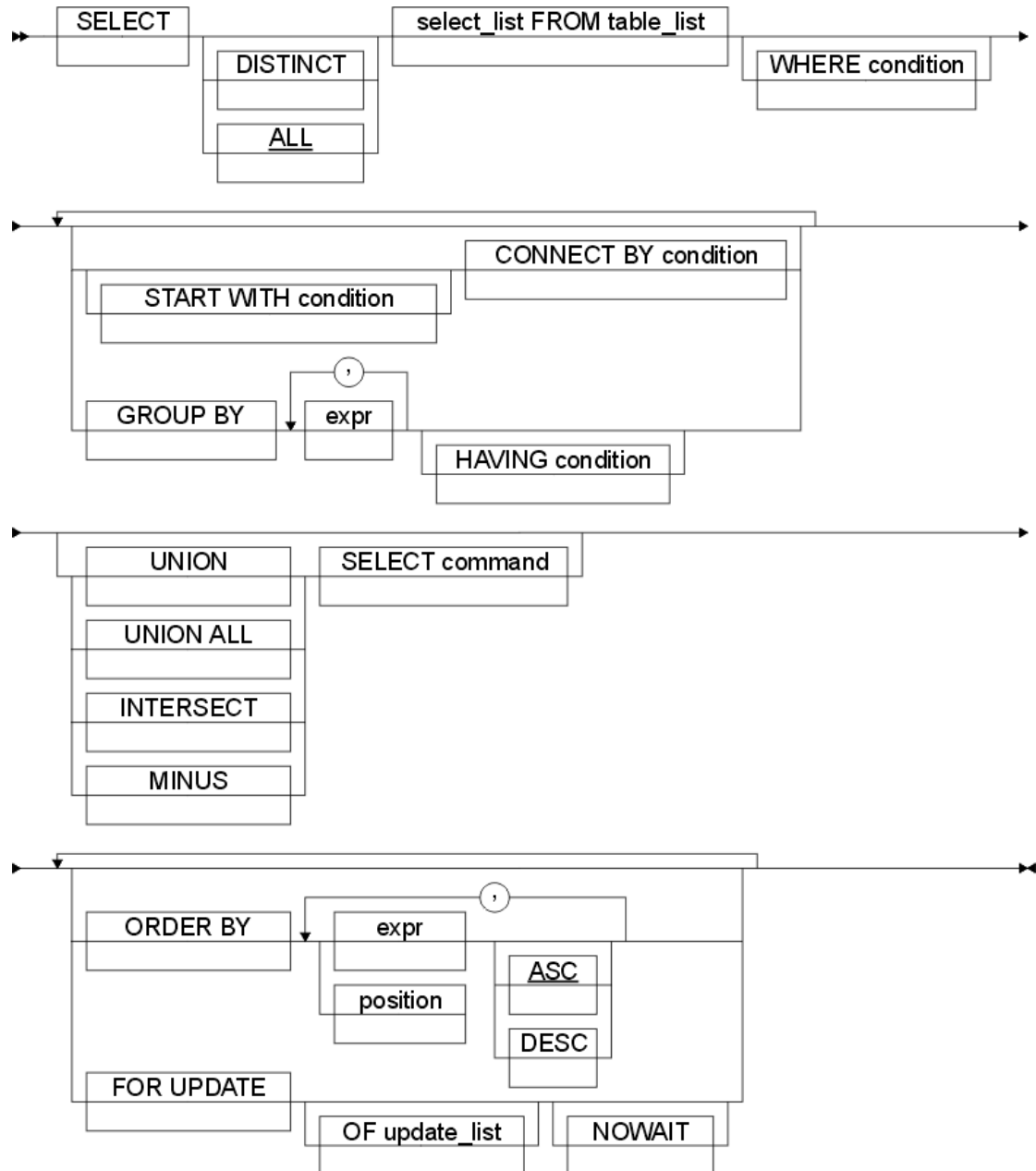


#### Задание:

Выполните проверку правильности изменения данных

## Оператор SELECT

Выбирает данные из одной или нескольких таблиц или представлений. Может использоваться как оператор или как подзапрос в другом операторе.



**Пример 1:** Лучшим примером, иллюстрирующим работу оператора SELECT, является юмористический пример "Как программист SQL охотится на слонов". Дано: Слон живет в Африке. Задача: Что надо сделать, чтобы найти слона? Метод решения: Программист SQL делает SELECT.

**SELECT "СЛОН" FROM AFRICA;**

Итог: Все африканские слоны найдены ☺.

Проиллюстрируем использование оператора SELECT на нескольких примерах.

## Арифметические выражения

Выражение - это комбинация одного или более числовых значений, операторов и функций, вырабатывающая числовое значение. Арифметические выражения могут содержать имена полей, числовые константы и арифметические операторы:

## Оператор конкатенации

Оператор конкатенации (||) позволяет соединять столбцы с другими столбцами, арифметическими выражениями или константами для формирования символьных строк. Столбцы по обе стороны оператора сливаются воедино.

## Обработка пустых значений

Если в поле нет значения, то оно считается пустым (NULL). Пустое значение - это значение, которое недоступно, не присвоено, неизвестно или не определено. Пустое значение - это не нулевое значение. Ноль - это число. Пустые значения занимают один байт памяти. Пустые значения поддерживаются языком SQL. Если хотя бы одно поле в выражении имеет пустое значение, то и результат не определен.

## Сортировка результатов

При выдаче таблицы результатов порядок строк в ней не определен. Предложение ORDER BY используется для сортировки строк по значениям полей в столбцах. Предложение ORDER BY всегда стоит в конце команды SELECT.

Чтобы изменить порядок сортировки, т.е. отсортировать строки по убыванию значений полей столбца (Descending order), задайте квалификатор DESC в предложении ORDER BY.

Допускается сортировка по нескольким столбцам, вплоть до всех столбцов таблицы результатов. Необходимо задавать столбцы сортировки в предложении ORDER BY через запятую. При этом первичный ключ сортировки стоит в предложении ORDER BY первым, вторичный - вторым и т.д.

## Правила соединения таблиц

Чтобы соединить три таблицы, необходимо задать два условия соединения. Чтобы соединить четыре таблицы, нужны как минимум три соединяющих условия.

**Общее простое правило: количество соединяемых таблиц минус один = минимальное число условий соединения.**

Это правило может не работать, если в вашей таблице есть конкатенированный (состоящий из нескольких полей) первичный ключ (PRIMARY KEY), однозначно идентифицирующий каждую запись. Когда условие соединения задано некорректно или вообще отсутствует, результатом соединения оказывается произведение двух таблиц.

## Использование сокращенных имен таблиц

Довольно утомительно несколько раз набирать имена таблиц в одной команде - имена могут быть длинными и повторяться много раз в предложениях SELECT, WHERE, ORDER BY. Чтобы избежать этого, можно использовать временные синонимы (или заменители) имен таблиц, задаваемые в предложении FROM и действующие только в данной команде. Заменители имен таблиц (называемые также табличными метками) следует также задавать в предложении SELECT. Это повышает эффективность выполнения запроса, упрощая его компиляцию

**ПРИМЕР:** select E.ENAME, D.DEPTNO, D.DNAME from EMP E, DEPT.D  
where E.DEPTNO = D.DEPTNO  
order by D.DEPTNO

## Внешние соединения

Оператор внешнего соединения позволяет выбрать те строки из одной таблицы, которым нет пар в другой. Он может быть задан только с одной стороны соединяющего условия - с той, где отсутствует информация. Для получения не имеющих пары строк в обеих таблицах сразу, необходимо воспользоваться оператором UNION (см. ниже).

Другие ограничения на внешние соединения:

- Вы не можете сделать внешнее соединение одной таблицы с двумя другими в одной команде SELECT (т.е. поставить (+) напротив одной и той же таблицы в двух операторах сравнения).
- Условие внешнего соединения не может включать оператор IN или быть связано с другим условием оператором OR.

**ПРИМЕР:** select E.ENAME, D.DEPTNO, D.DNAME  
from EMP E, DEPT D  
where E.DEPTNO(+) = D.DEPTNO  
and D.DEPTNO IN (30,40);

## Рекурсивные соединения

С помощью табличных меток можно подсоединить таблицу к самой себе, как будто это две различные таблицы. Это дает возможность соединить строки таблицы со строками этой же таблицы.

**ПРИМЕР:** select E.ENAME "EMP NAME", E.SAL "EMP SAL",  
M.ENAME "MGR NAME", M.SAL "MNR SAL"  
from EMP E, EMP M  
where E.MGR = M.EMPNO  
and E.SAL < M.SAL;

## Вложенные подзапросы

Подзапросы могут быть вложенными (т.е. в одних подзапросах могут задаваться другие). Допускается до 255 уровней вложенности подзапросов.

## Основные правила при задании подзапросов

Внутренний запрос берется в круглые скобки и должен стоять в правой части оператора сравнения внешнего запроса.

- Подзапрос не может содержать предложения ORDER BY.
- Предложение ORDER BY ставится последним в основном запросе.
- Имена столбцов в предложении SELECT внутреннего запроса должны стоять в той же последовательности, что и имена столбцов в левой части оператора сравнения внешнего запроса. Типы столбцов также должны попарно соответствовать.
- Подзапросы всегда выполняются от внутренних к внешним, пока не коррелируют друг с другом (обсуждается позже). Выполнение сравнения во внешнем запросе базируется на результатах выполнения внутреннего запроса.
- При задании критерия поиска могут использоваться логические операторы, SQL-операторы, а также операторы ANY и ALL.
- Подзапросы могут:
  - возвращать одну или более строк;



- возвращать один или более столбцов;
- использовать группы или групповые функции;
- задаваться в сложных критериях поиска внешних запросов с использованием предикатов AND и OR;
- соединять таблицы;
- обращаться к таблице, отличной от той, к которой обращается внешний запрос;
- стоять в командах SELECT, UPDATE, DELETE, INSERT, CREATE TABLE.
- коррелировать с внешним запросом;
- использовать операторы над множествами.

### Коррелирующие подзапросы

Коррелирующие подзапросы - это вложенные подзапросы, выполняющиеся для каждой "строки-кандидата" из главного запроса, и для выполнения которых требуется информация из строк главного запроса. Такая корреляционная зависимость подразумевает иной алгоритм выполнения запроса, чем в запросах с обычными вложенными подзапросами. Коррелирующий подзапрос задается постановкой во внутренний подзапрос имени столбца из внешнего запроса. В обычном вложенном подзапросе внутренний запрос выполняется первым и один раз, возвращая значения, используемые в главном запросе. Коррелирующий подзапрос выполняется по разу для каждой строки (кандидата), выбранной внешним запросом.

**ПРИМЕР:** Найти поиска всех сотрудников, получающих больше, чем средняя зарплата в их отделе:

```
select EMPNO, ENAME, SAL, DEPTNO
from EMP E
where SAL > (SELECT AVG(SAL) FROM EMP
              where DEPTNO = E.DEPTNO)
order by DEPTNO;
```



#### Задание 5.8.:

Выведите сведения о сотрудниках, которые подчиняются такому же менеджеру, что и сотрудники с номерами (EMPLOYEE\_ID) 141 или 174, и работают в одном отделе, что и сотрудники с номерами 141 или 174, не включая самих сотрудников в список.

```
SELECT employee_id, last_name, manager_id, department_id
FROM employees
WHERE manager_id IN
      (SELECT manager_id
       FROM employees
       WHERE employee_id IN (174,141))
AND department_id IN
      (SELECT department_id
       FROM employees
       WHERE employee_id IN (174,141))
AND employee_id NOT IN(174,141);
```

Результат: строк

## Операции над множествами

Операторы **UNION**, **INTERSECT** и **MINUS** удобны при составлении запросов к разным таблицам. Они комбинируют результаты по нескольким запросам в одну результирующую таблицу. Таким образом, запрос может состоять из нескольких SQL-команд, связанных одним или более операторами над множествами. Операторы над множествами часто называются вертикальным соединением, поскольку соединение в данном случае происходит не по строкам, а по столбцам.

Операция	Выполняемые функции
UNION	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные хотя бы одним из запросов.
UNION ALL	Комбинирует два запроса; возвращает все строки, извлеченные хотя бы одним из запросов, включая повторяющиеся.
INTERSECT	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные каждым из запросов.
MINUS	Комбинирует два запроса; возвращает все неповторяющиеся строки, извлеченные первым запросом, но не извлеченные вторым.

Операция	Выполняемые функции
(+)	Указывает, что предшествующий столбец является столбцом внешнего соединения.
*	Используется вместо имен столбцов при выборке всех столбцов из таблицы или представления.
PRIOR	Используется в иерархическом древовидном запросе для определения зависимости между родительскими и дочерними строками. Смотрите оператор SELECT.
ALL	Оставляет повторяющиеся строки в результате запроса (установлен по умолчанию ALL, но не DISTINCT).
DISTINCT	Удаляет повторяющиеся строки из результата запроса.

Правила, которые необходимо соблюдать при использовании операторов над множествами.

- Команды SELECT должны выбирать одинаковое количество столбцов.
- Соответствующие столбцы должны иметь одинаковый тип.
- Одинаковые строки автоматически исключаются (за исключением UNION ALL). Предикат DISTINCT не допускается.
- В результирующей таблице столбцы именуются по первой SQL-команде.

## UNION

Служит для получения всех неодинаковых строк, выбираемых по нескольким командам (объединение результирующих таблиц):

**ПРИМЕР:**   select JOB from EMP where DEPTNO=10  
                  UNION  
                  select JOB from EMP where DEPTNO=30;

## INTERSECT

Служит для получения всех неодинаковых строк, которые выбраны каждой из команд (пересечение результирующих таблиц)

**ПРИМЕР:**    select JOB from EMP where DEPTNO=10  
                 INTERSECT  
                 select JOB from EMP where DEPTNO=30;

## MINUS

Служит для того, чтобы выбрать все строки, найденные по одному запросу, но не найденные по другому (разность результирующих таблиц).

### Операторы SOME|ANY и ALL

Операторы ANY и ALL могут применяться в подзапросах, возвращающих более одной строки. Они задаются в предложениях WHERE или HAVING вместе с логическими операторами (=, <>, <, >, >=, <=). ANY (или его синоним SOME) сравнивает значение левой части оператора сравнения с каждым значением, возвращаемым подзапросом. Результат сравнения положителен, если хотя бы одно значение из найденных по подзапросу удовлетворяет оператору сравнения.

**ПРИМЕР:**    Все сотрудники, зарабатывающие больше минимальной зарплаты в отделе 30;  
                 select ENAME, SAL, JOB, DEPTNO  
                 from EMP  
                 where SAL > SOME (select DISTINCT SAL  
   from EMP  
   where DEPTNO=30)  
                 order by SAL DESC;

**ПРИМЕР:** Все служащие, которые зарабатывают больше, чем любой из сотрудников 30-го отдела;  
                 select ENAME, SAL, JOB, DEPTNO  
                 from EMP  
                 where SAL > ALL (select DISTINCT SAL  
                                 from EMP where DEPTNO = 30);



#### **Задание 5.8.1:**

Выполните задание 5.8. с использованием операций над множествами

## Функции группировки

Статистическая информация (такая, как среднее или суммарное значение) может быть получена для группы строк таблицы с помощью так называемых групповых функций. Групповые функции оперируют с набором строк. Они формируют результаты, основанные на обработке группы строк и для этой группы, в отличие от простых функций, работающих с отдельными строками. По умолчанию все строки таблицы результатов трактуются как одна группа. Чтобы разбить строки на группы, используется предложение **GROUP BY**, специфицируемое в команде SELECT.

Ниже приведен список групповых функций.

Функция	Возвращаемое значение
AVG ([DISTINCT/ALL]n)	Среднее значение группы полей в столбце, не включая пустых значений
COUNT ([DISTINC/ALL] выражение)	Количество строк в группе, в которых имеет непустое значение
MAX ([DISTINCT/ALL] выражение)	Максимальное значение выражения в группе
MIN ([DISTINCT/ALL] выражение)	Минимальное значение выражения в группе
STDDEV ([DISTINCT/ALL] n)	Функция, определяющая математическое ожидание в группе
SUM ([DISTINCT/ALL] n)	Сумма значений, игнорируя пустые значения
VARIANCE ([DISTINCT/ALL] n)	Дисперсия в группе.

В приведенной таблице под числом понимается имя поля, константа или арифметическое выражение числового типа. Выражение может иметь любой тип. Чаще всего в качестве чисел и выражений используются имена столбцов таблиц. Перечисленные функции оперируют с группами строк выходных таблиц (например, со всей таблицей, как с одной группой) и в связи с этим называются групповыми или агрегированными функциями. Квалификатор DISTINCT дает команду групповой функции оперировать только с неодинаковыми строками в группах, т.е. дублированные строки при работе функции отбрасываются. Квалификатор ALL оставляет дублированные строки для обработки групповыми функциями. По умолчанию все функции используют квалификатор ALL. Тип данных выражения может быть CHAR, NUMBER или DATE (для функций, использующих в качестве аргумента выражение). Все групповые функции, кроме COUNT(\*), игнорируют пустые значения. Для обработки пустых значений используйте функцию NVL.

**ПРИМЕР:** Вычислить среднее значение зарплаты для каждой должности  
select JOB, AVG(SAL) from EMP group by JOB



#### Задание 5.9:

Составьте список всех сотрудников, зарабатывающих больше среднего оклада по отделу, в котором они работают.

```
SELECT last_name, salary, department_id
FROM employees outer
WHERE salary > (SELECT AVG(salary)
                FROM employees
                WHERE department_id = outer.department_id);
```

Результат: строк



#### Задание 5.10:

Выведите Название стран и среднюю зарплату (с округлением до целого значения) сотрудников работающих в этих странах, отсортировав результат в порядке убывания среднего оклада.

```
select country_name, round(avg(salary)) as n from
countries c,locations l,departments d,employees e
where c.country_id = l.country_id and l.location_id =
d.location_id and e.department_id = d.department_id
group by country_name
order by n desc
```

Результат: строк

## HAVING

Предложение **HAVING** задается, если необходимо выбрать не все формируемые предложением GROUP BY группы, а лишь часть их по определенному критерию (на групповые характеристики).

**ПРИМЕР:** Определить среднюю зарплату в отделах, имеющих более трех служащих:  
select DEPTNO, AVG(SAL) from EMP group by DEPTNO having COUNT (\*) > 3;



### Задание 5.8.2:

**ЗАДАНИЕ:** Для схемы SCOTT определить среднюю зарплату по должностям для каждого из отделов, величина которой превышает 3000;

**ЗАДАНИЕ:** Для схемы SCOTT определить в каком году в компанию было зачислено наибольшее количество человек? Выдайте этот год и количество зачисленных служащих.

## Оператор EXISTS

Оператор EXISTS часто задается с коррелирующими подзапросами. Он проверяет, найдена ли по подзапросу хотя бы одна строка. Если да, то условие выполнено и оператор возвращает значение TRUE; если нет - то FALSE (оператор NOT EXISTS возвращает TRUE, если, наоборот, ни одной строки по подзапросу не найдено).

**ПРИМЕР:** Найти всех служащих, у которых имеется хотя бы один подчиненный  
select EMPNO, ENAME, JOB, DEPTNO  
from EMP E  
where EXISTS (select EMPNO from EMP  
where EMP.MGR=E.EMPNO);



### Задание 5.8.3:

**ЗАДАНИЕ:** Найти всех служащих, которые никем не руководят. В чем ошибка след. Запроса:

select ENAME, JOB FROM EMP  
where EMPNO NOT IN (select MGR from EMP);

## Правила задания комментариев в тексте SQL программ

Введение комментариев в исходные тексты программ:

- значительно упрощает разработку и отладку программ, особенно при ведении совместных разработок.
- использование комментариев при разметке сложной программы на языке PL/SQL, позволяет отметить отдельные разделы, а затем вернуться к каждому разделу и заполнять его операторами.
- комментарий в заголовке модуля, содержащий основную информацию о модуле, поможет получить общую информацию и значительно сокращает время на анализ сценария;

После компиляции сценария, сохраненного с такими многострочными комментариями, компилятор PL/SQL выводит их на экран в виде блока описания программ DOC:

Это удобно в случае компиляции полного набора модулей (при этом все результаты выводятся в файл). Рекомендуется ограничивать количество строк в комментарии заголовка модуля, так как он может быть выведен на экран в процессе компиляции и сбить с толку оператора. Если первый ограничитель многострочного комментария не размещен в строке один, то первая строка комментария не выводится на экран.



**Задание 5.11:** Вывести из таблицы HR.EMPLOYEES информацию о сотрудниках отделов с номерами 10,30,50,90. Вывод должен быть оформлен в таблицу, содержащую столбцы:

1. Сквозной порядковый номер сотрудника.
2. Порядковый номер сотрудника внутри отдела.
3. Номер отдела для данного сотрудника (Department\_id).
4. Должность сотрудника (Job\_id).
5. Фамилия сотрудника (Last\_name).
6. Оклад (Salary).
7. Ранг зарплаты сотрудника в отделе (1-самый высокооплачиваемый).

Строки в выводимой таблице должны удовлетворять следующим условиям:

1. Строки, представляющие сотрудников одного отдела должны располагаться друг за другом.
2. Строки, представляющие сотрудников одного отдела должны располагаться в порядке убывания окладов.

**Дополнительные требования к выполнению:**

Обратите внимание, что ранг в отделе зависит от нумерации в отделе.

```
select
    row_number() OVER(order by Department_id) "Сквозной №"
  , row_number() OVER(PARTITION BY Department_id Order by
    Salary desc) "№ внутри отдела"
  , Department_id "# отд."
  , Job_id        "Должность"
  , Last_name     "Фамилия"
  , Salary        "Зарплата"
  , rank() OVER(PARTITION BY Department_id Order by salary
    desc) "Ранг в отделе"
from EMPLOYEES
where Department_id in (10,30,50,90);
```

Результат: строк

### Контрольные вопросы

1. Синтаксическая диаграмма SELECT, пример.
2. Синтаксическая диаграмма INSERT, пример.
3. Синтаксическая диаграмма UPDATE, пример.
4. Синтаксическая диаграмма DELETE, пример.

### СПИСОК ЛИТЕРАТУРЫ

1. Власов А.И., Лыткин С.Л., Яковлев В.Л. Краткое практическое руководство по языку PL/SQL - М.: Машиностроение. 2000. 64 с.
2. Сервер Oracle. Справочное руководство по языку SQL / Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1994.
3. Сервер ORACLE. Основные концепции/ Под ред. А.В.Емельяненко, Н.В. Емельяненко - Протвино, АО РДТех, 1996.
4. Проектирование и эксплуатация конструкторско-технологических баз данных на основе СУБД Oracle/ Конспект лекций - М.: Москва, МГТУ им. Н.Э. Баумана, 2001, 120 с.

Отчет по лабораторной работе № 5 «Основы PL/SQL»			
дата	Оценка (max 5)	Бонус за сложность	подпись

**Цели работы:**

Изучение операторов PL/SQL

**Задачи работы:**

- создание ненаименованных блоков
- создание функции
- создание процедур
- создание пакетов
- создание триггеров

**Задание повышенной сложности (бонус за сложность – 5 баллов):**

- выполнение триггеров – фильтров входных данных строкового типа

**Краткий конспект теоретической части (ответы на контрольные вопросы)**

Определение «функции» \_\_\_\_\_

---



---



---



---

Определение «процедуры» \_\_\_\_\_

---



---



---



---

Определение «пакета» \_\_\_\_\_

---



---



---



---

Определение «триггера» \_\_\_\_\_

---



---



---



---

Структура ненаименованного PL/SQL блока \_\_\_\_\_

---



---



---



---

**PL/SQL** – язык Oracle четвертого поколения, объединяющий структурированные элементы процедурного языка программирования с языком SQL, разработанный специально для организации вычислений в среде клиент/сервер.

#### **Свойства:**

- Он позволяет передать на сервер программный блок PL/SQL, содержащий логику приложения, как оператор SQL, одним запросом.
- Используя PL/SQL, можно значительно уменьшить объем обработки в клиентской части приложения и нагрузку на сеть. Например, может понадобиться выполнить различные наборы операторов SQL в зависимости от результата некоторого запроса. Запрос, последующие операторы SQL и операторы условного управления могут быть включены в один блок PL/SQL и пересланы серверу за одно обращение к сети.
- Вся логика приложений делится на клиентскую и серверную части. Серверная часть может быть реализована в виде функций, хранимых процедур и пакетов.

#### **Базовые элементы:**

##### ***Ненаименованные блоки***

Законченный логический блок, реализованный на PL/SQL, выполняемый в терминальном режиме, без возможности обращения к нему из других модулей.

##### ***Наименованные блоки:***

**Функции.** Часть логики приложения ориентированной на выполнение конкретного комплекса операций на сервере, результат которых возвращается в виде значения функции. Откомпилированные функции и их исходные тексты содержатся в базе данных.

**Хранимые процедуры.** Часть логики приложения, особенно нуждающаяся в доступе к базе данных, может храниться там, где она обрабатывается (на сервере). Хранимые процедуры не возвращают значения результата, обеспечивают удобный и эффективный механизм безопасности. Откомпилированные хранимые процедуры и их исходные тексты содержатся в базе данных.

**Пакеты.** Часть логики приложений: функций и пакетов, предназначенных для решения задач в рамках одного модуля (подсистемы) АИС.

**Триггеры базы данных.** Можно использовать триггеры, чтобы организовать сложный контроль целостности, выполнять протоколирование (аудит) и другие функции безопасности, реализовать в приложениях выдачу предупреждений и мониторинг.

**Декларативная целостность.** Ограничения активизируются сервером всякий раз, когда записи вставляются, обновляются или удаляются. В дополнение к ограничениям ссылочной целостности, которые проверяют соответствие первичного и внешнего ключей, можно также накладывать ограничения на значения, содержащиеся в столбцах таблицы.



## Принципы разработки ненаименованных PL/SQL блоков

Программы на PL/SQL имеют блочную структуру:

DECLARE

-- объявления переменных, констант, типов данных, курсоров, функций и процедур.

BEGIN

-- выполняемый код

EXCEPTION

-- обработка исключений

END;



### Пример 1:

Вывести в выходной поток SQL+ строковую переменную «HELLO WORLD»

```
SET SERVEROUTPUT ON  -- определяет вывод SQL*Plus всю информацию
                      -- возвращаемую сервером.

BEGIN
    DBMS_OUTPUT.enable;          -- включение механизма вывода
    DBMS_OUTPUT.put_line('HELLO WORLD'); -- печать строки
END;
/ -- указание к выполнению блока PL/SQL
```

Вклейте результат работы программы в SQL+



### Пример 2:

Пример: Разработать ненаименованный PL/SQL блок, вычисляющий площадь круга и длину окружности.

```
DECLARE
PI CONSTANT REAL := 3.141519265359;
LOKR REAL;
SKRG REAL;
RADIUS REAL := &RADIUS; -- указывает на необходимость ввода перем.
BEGIN
    LOKR := PI * RADIUS * 2.0;
    SKRG := PI *RADIUS ** 2;
    DBMS_OUTPUT.put_line('Радиус = ' || to_char(RADIUS)|| ', ДЛИНА
ОКРУЖНОСТИ =' || to_char(LOKR) || ', ПЛОЩАДЬ КРУГА =' || to_char(SKRG));
END;
/
```

Вклейте результат работы программы в SQL+

## Управляющие структуры PL/SQL


Для управления работой PL/SQL блоков используются **условные операторы и циклы**.

### Условные операторы: IF-THEN-ELSE


Синтаксис:


```
IF логическое выражение 1 THEN
    операторы 1;
    [ELSIF логическое выражение 2 THEN
        операторы 1;]
.....
    ELSE операторы N;
END IF;
```


### IF-THEN-ELSE

	<b>Пример 3:</b> Модифицировать пример, чтобы учитывать проверку на равенство NULL значения v_num1
<pre>DECLARE     v_num1 NUMBER;     v_num2 NUMBER;     v_result VARCHAR(7); BEGIN     IF v_num1 &gt;= v_num2 THEN         v_result := 'No';     ELSE         v_result := 'Yes';     END IF; END;</pre>	

### Работа с циклами

		<b>Пример 4:</b> Привести примеры работы с циклами
<b>Простые циклы:</b>  LOOP операторы; END LOOP;	<b>Условные циклы:</b>  WHILE условие LOOP операторы; END LOOP;	<b>Числовые циклы:</b>  FOR счетчик IN min..max LOOP операторы END LOOP;


	<b>Пример 5:</b> Модифицировать пример, используя различные варианты задания циклов
<pre> DECLARE dt date; radius real := &amp;radius; BEGIN   dt:=sysdate;   for i in 1..5 loop dbms_output.put_line(to_char( rai_lokrug(radius+i))  'время'    to_char(sysdate-dt));   end loop; END; / </pre>	

	<b>Пример 6:</b> Приведите собственный пример обработки исключительных ситуаций
<p>Prompt Обработка исключительных ситуаций</p> <pre> DECLARE x real := &amp;x; n integer := 0; BEGIN   dbms_output.enable;   dbms_output.put_line('Значения функции');   dbms_output.put_line('abs (x)='  to_char( ABS(x) ) );   n:=n+1;   dbms_output.put_line('ceil (x)='  to_char( ceil(x) ) );   n:=n+1; exception   when others then begin     dbms_output.put_line('ERRORS');     if n=0 then       dbms_output.put_line('Ошибка в функции ABS (x) ');     end;   end; END; / </pre>	

## Библиотечные функции в Oracle

### Числовые функции

Функция	Возвращаемое значение
<b>ABS(n)</b>	Абсолютное значение величины <i>n</i> .
<b>CEIL(n)</b>	Наименьшее целое, большее или равное <i>n</i> .
<b>COS(n)</b>	Косинус <i>n</i> (угла, выраженного в радианах).
<b>COSH(n)</b>	Гиперболический косинус <i>n</i> .
<b>EXP(n)</b>	<i>e</i> в степени <i>n</i> .
<b>FLOOR(n)</b>	Наибольшее целое, меньшее или равное <i>n</i> .
<b>LN(n)</b>	Натуральный логарифм <i>n</i> , где $n > 0$ .
<b>LOG(m,n)</b>	Логарифм <i>n</i> по основанию <i>m</i> .
<b>MOD(m,n)</b>	Остаток от деления <i>m</i> на <i>n</i> .
<b>POWER(m,n)</b>	<i>m</i> в степени <i>n</i> .
<b>ROUND(n[,m])</b>	<i>n</i> , округленное до <i>m</i> позиций после десятичной точки. По умолчанию <i>m</i> равно нулю.
<b>SIGN(n)</b>	Если $n < 0$ , -1; если $n = 0$ , 0; если $n > 0$ , 1.
<b>SIN(n)</b>	Синус <i>n</i> (угла, выраженного в радианах).
<b>SINH(n)</b>	Гиперболический синус.
<b>SQRT(n)</b>	Квадратный корень от <i>n</i> , если $n < 0$ , возвращает значение NULL.
<b>TAN(n)</b>	Тангенс <i>n</i> (угла, выраженного в радианах).
<b>TANH(n)</b>	Гиперболический тангенс <i>n</i> .
<b>TRUNC(n[,m])</b>	<i>n</i> , усеченное до <i>m</i> позиций после от десятичной точки. По умолчанию <i>m</i> равно нулю.

	<b>Пример 7:</b> Приведите пример использования библиотечной функции данного типа

## Символьные функции, возвращающие символьные значения:

Функция 1	Возвращаемое значение
CHR(n)	Символ с кодом <i>n</i> .
CONCAT(char1, char2)	Конкатенация символьных строк <i>char1</i> и <i>char2</i> .
INITCAP(char)	Символьная строка <i>char</i> , первые буквы всех слов в которой преобразованы в прописные.
LOWER(char)	Символьная строка <i>char</i> , все буквы которой преобразованы в строчные.
LPAD(char1.n [,char2])	Символьная строка <i>char1</i> , которая дополняется слева последовательностью символов из <i>char2</i> так, чтобы общая длина строки стала равна <i>n</i> . Значение <i>char2</i> по умолчанию - " (один пробел). Если часть многобайтового символа не помещается в добавляемой строке, то конец строки заполняется пробелами.
LTRIM(char[,set])	Символьная строка <i>char</i> , в которой удалены все символы от начала вплоть до первого символа, которого нет в строке <i>set</i> . Значение <i>set</i> по умолчанию - " (один пробел).
NLS_INITCAP(char[,nls_sort])	Символьная строка <i>char</i> , в которой первые буквы всех слов преобразованы в прописные. Параметр <i>nls_sort</i> определяет последовательность сортировки.
NLS_LOWER(char[,nls_sort])	Символьная строка <i>char</i> , все буквы которой преобразованы в строчные. Параметр <i>tils-sort</i> определяет последовательность сортировки.
NLS_UPPER(char[,nls_sort])	Символьная строка <i>char</i> , все буквы которой преобразованы в прописные. Параметр <i>nts_sort</i> определяет последовательность сортировки.
REPLACE(char, search_string [,replacement_string])	Символьная строка <i>char</i> , в которой все фрагменты <i>search_string</i> заменены на <i>replacement_string</i> . Если параметр <i>replacement_string</i> не определен, все фрагменты <i>search-string</i> удаляются.
RPAD(char1 [,char2])	Символьная строка <i>char1</i> , которая дополнена справа последовательностью символов из <i>char2</i> так, что общая длина строки равна <i>n</i> . Если часть многобайтового символа не помещается в добавляемой строке, то конец строки заполняется пробелами.
RTRIM(char[,set])	Символьная строка <i>char</i> , в которой удалены все символы справа вплоть до первого символа, которого нет в строке <i>set</i> . Значение параметра <i>set</i> по умолчанию - " (один пробел).
SOUNDEX(char)	Символьная строка, содержащая фонетическое представление для <i>char</i> , на английском языке.
SUBSTR(char, m[,n])	Фрагмент символьной строки <i>char</i> , начинающийся с символа <i>m</i> , длиной <i>n</i> символов (до конца строки, если параметр <i>n</i> не указан).
SUBSTRB(char, m[,n])	Фрагмент символьной строки <i>char</i> , начинающийся с символа <i>m</i> , длиной <i>n</i> байтов (до конца строки, если параметр <i>n</i> не указан).
TRANSLATE(char, from, to)	Символьная строка <i>char</i> , в которой все символы, встречающиеся в строке <i>from</i> , заменены на соответствующие символы из <i>to</i> .
UPPER(char)	Символьная строка <i>char</i> , в которой все буквы преобразованы в прописные.




### Пример 8:

Приведите пример использования библиотечной функции данного типа

## Символьные функции, возвращающие числовые значения

Функция	Возвращаемое значение
<b>ASCII(char)</b>	Возвращает десятичный код первого символа строки <i>char</i> в кодировке, принятой в базе данных. (Код ASCII в системах, использующих кодировку ASCII). Возвращает значение первого байта многобайтового символа.
<b>INSTR(char1.char2[,n[,m]])</b>	Позиция первого символа <i>m</i> -ого фрагмента строки <i>char1</i> , совпадающего со строкой <i>char2</i> , начиная с <i>n</i> -ого символа. По умолчанию <i>n</i> и <i>m</i> равны 1. Номер символа отсчитывается от первого символа строки <i>char1</i> , даже когда <i>n</i> > 1.
<b>INSTRB(char1.char2[,n[,m]])</b>	Позиция первого символа <i>m</i> -ого фрагмента строки <i>char1</i> , совпадающего со строкой <i>char2</i> , начиная с <i>m</i> -ого байта. По умолчанию <i>n</i> и <i>m</i> равны 1. Номер байта отсчитывается от первого символа строки <i>char1</i> , даже когда <i>n</i> > 1.
<b>LENGTH(char)</b>	Длина строки <i>char</i> в символах.
<b>LENGTHB(char)</b>	Длина строки <i>char</i> в байтах.
<b>NLSSORT(char1,char2[,n[,m]])</b>	Зависящее от национального языка значение, используемое при сортировке строки <i>char</i> .

	<b>Пример 9:</b> Приведите пример использования библиотечной функции данного типа

## Групповые функции


Функция	Возвращаемое значение
<b>AVG([DISTINCT ALL]n)</b>	Среднее значение от <i>n</i> , нулевые значения опускаются.
<b>COUNT([ALL]*)</b>	Число строк, извлекаемых в запросе или подзапросе.
<b>COUNT(DISTINCT ALL] <i>expr</i>)</b>	Число строк, для которых <i>expr</i> принимает не пустое значение.
<b>MAX([DISTINCT ALL] <i>expr</i>)</b>	Максимальное значение выражения <i>expr</i> .
<b>MIN([DISTINCT ALL] <i>expr</i>)</b>	Минимальное значение выражения <i>expr</i> .
<b>STDDEV([DISTINCT ALL] <i>n</i>)</b>	Стандартное отклонение величины <i>n</i> , нулевые значения опускаются.
<b>SUM([DISTINCT ALL] <i>n</i>)</b>	Сумма значений <i>n</i>
<b>VARIANCE([DISTINCT ALL] <i>n</i>)</b>	Дисперсия величины <i>n</i> , нулевые значения опускаются.

## Функции работы с датами

Функция	Возвращаемое значение
<b>ADD-MONTHS (d,n)</b>	Дата d плюс n месяцев.
<b>LAST-DAY (d)</b>	Последнее число месяца, указанного в d
<b>MONTHS-BETWEEN (d1, d2)</b>	Число месяцев между датами d1 и d2.
<b>NEW-TIME (d, a, b)</b>	Дата и время в часовом поясе a, соответствующие дате и времени в часовом поясе b, при этом d,a и b значения типа CHAR, определяющие часовые пояса.
<b>NEW-DAY (d, char)</b>	Дата первого после даты (/дня недели, название которого записано в <i>спис.</i>
<b>SYSDATE</b>	Текущая дата и время.

## Усечение и округление дат

Функция	Возвращаемое значение
<b>ROUND(d [,fmt])</b>	Дата d, округленная до единиц, указанных в форматной маске.
<b>TRUNC(d [,fmt])</b>	Дата d, усеченная по форматной маске fmt.

	<b>Пример 10:</b> Приведите пример использования библиотечной функции данного типа
<pre>Select sysdate from dual;</pre>	

## Форматные маски дат для функций ROUND и TRUNC.


В таблице перечислены форматные маски, которые можно использовать в функциях ROUND и TRUNC. По умолчанию используется форматная маска “DD”.

Форматная маска	Возвращаемое значение
<b>CC или SCC</b>	Первый день столетия
<b>SYYYYY или YYYYY или YYY или YY или Y или YEAR или SYEAR</b>	Первый день года ( округляется до 1 июля)
<b>Q</b>	Первый день квартала (округляется до 16 числа второго месяца квартала)
<b>MONTH или MON или MM или RM</b>	Первый день месяца (округляется до 16 числа)
<b>WW или IW</b>	Тот же день недели, что и первый день текущего года
<b>W</b>	Тот же день недели, что и первый день текущего месяца
<b>DDD или DDD или J</b>	День
<b>DAY или DY или D</b>	Первый день недели
<b>HH HH12 HH24</b>	Час
<b>MI</b>	Минута

### Форматные маски дат в TO\_CHAR и TO\_DATE.

Элементы форматной маски даты перечислены в приведенной ниже таблице. Любую комбинацию этих элементов можно использовать как аргумент fmt функций TO\_CHAR или TO\_DATE. По умолчанию fmt равен 'DD-MON-YY'.

Элемент формата	Возвращаемое значение
<b>SCC или CC</b>	Столетие; если указано 'S' то перед датами до нашей эры ставится '-'. YYY или YY или Y] Последние 3, 2, или 1 цифра года.
<b>YYYY или SYYYY</b>	Год; если указано 'S' то перед датами до нашей эры ставится '-'. YYY или YY или Y] Последние 3, 2, или 1 цифра года.
<b>IYYY</b>	4 цифры года по стандарту ISO. IYY или IY или I] Последние 3, 2, или 1 цифра года по стандарту ISO.
<b>Y,YYY</b>	Год с запятой в указанной позиции.
<b>SYEAR или YEAR</b>	Год, записанный словами, а не цифрами; если указано 'S' то перед датами до нашей эры ставится '-'. YYY или YY или Y] Последние 3, 2, или 1 цифра года.
<b>RR</b>	Последние 2 цифры года; для указания года в других столетиях.
<b>BC или AD</b>	BC- до нашей эры(до н.э.); AD – нашей эры
<b>B.C. или A.D.</b>	B.C.- до нашей эры(до н.э.); A.D. – нашей эры
<b>Q</b>	Квартал (1, 2, 3, 4; JAN-MAR=1).
<b>MM</b>	Месяц(01-12; JAN=1).
<b>RM</b>	Нумерация месяцев римскими цифрами(I-XII; JAN=I).
<b>MONTH</b>	Название месяца, дополненное пробелами до 9-ти символов.
<b>MON</b>	Сокращенное название месяца.
<b>WW или W</b>	Неделя года (1-52) или месяца (1-5).
<b>IW</b>	Неделя года (1-52 или 1-53) по стандарту ISO.
<b>DDD или DD или D</b>	День года (1-366) или месяца (1-31) или недели (1-7).
<b>DAY</b>	Название дня, дополненное пробелами до 9-ти символов.
<b>DY</b>	Сокращенное название дня.
<b>J</b>	Дата юлианского календаря; число дней, считая с первого января 4712 года до н.э.
<b>AM или PM</b>	AM –до полудня PM- после полудня
<b>A.M. или P.M.</b>	A.M. –до полудня P.M.- после полудня
<b>HH или HH12</b>	Час дня (1-12).
<b>HH24</b>	Час дня (0-23).
<b>MI</b>	Минута (0-59)
<b>SS или SSSS</b>	Секунда (0-59) или количество секунд после полуночи (0-86399).
<b>-,,::</b>	Знаки пунктуации.
<b>“...текст...”</b>	Текст воспроизводится в возвращенном значении.

	<b>Пример 11:</b> Приведите пример использования форматных масок
<pre>Select sysdate from dual;</pre>	



## Префиксы и суффиксы элементов формата даты

К элементам формата даты можно добавлять следующие префиксы:

FM	“Режим заполнения” подавляет заполнение пробелами, когда стоит перед MONTH или DAY
FX	“Точный формат”. Этот модификатор задает точное соответствие символического аргумента и форматной маски даты в функции TO_DATE.

К элементам формата даты можно добавлять следующие суффиксы:


TH	Порядковый номер (“DDTH” для “4TH”).
SP	Номер, записанный словами (“DDSP” для “FOUR”).
SPTH и THSP	Порядковый номер, записанный словами (“DDSPTH” для “FOURTH”).

Прописные и строчные буквы в элементах формата даты.

Следующие строки задают вывод прописными буквами, вывод прописными буквами только начальных букв слов, или вывод строчными буквами.

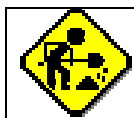
Прописные	Прописная начальная	Строчные
DAY	Day	.day
DY	Dy	.dy
MONTH	Month	.month
MON	Mon	.mon
YEAR	Year	.year
AM	Am	.am
PM	Pm	.pm
A.M.	A.m.	a.m.
P.M.	P.m.	p.m.

Если к элементу формата даты добавляется префикс или суффикс, то регистр (прописные, строчные буквы) определяется элементом формата, а не префиксом или суффиксом. Например, ‘ddTH’ задает “04th” а не “04TH”.

	<b>Пример 12:</b> Приведите пример использования префиксов и суффиксов
<pre>Select sysdate   from dual;</pre>	

### Функции преобразования

Функция	Возвращаемое значение
<b>CHARTOROWID(char)</b>	Char преобразуется из типа данных CHAR в тип данных ROWID
<b>CONVERT( char, dest_char_set [,source_char_set])</b>	Преобразует символьную строку из набора символов source_char_set в набор символов dest_char_set
<b>HEXTORAW ( char)</b>	Преобразует значение char, содержащее шестнадцатеричные цифры, в значение типа RAW
<b>RAWTOHEX ( raw)</b>	Преобразует raw в символьное значение, содержащее его шестнадцатеричный эквивалент
<b>ROWIDTOCHAR (rowid)</b>	Преобразует значение типа ROWID в значение типа CHAR
<b>TO_CHAR ( expr [,fmt [,’nls_num_fmt’]])</b>	Преобразует значение expr типа DATE или NUMBER в значение типа CHAR по формату форматной маски fmt. Если fmt отсутствует, значения типа DATE преобразуются по формату, заданному по умолчанию, и значения типа NUMBER- в значение типа CHAR с шириной, достаточной для того, чтобы вместить все значащие цифры. Значение ‘nls_num_fmt’ определяет связанные с языком форматные маски. В Trusted ORACLE преобразует значения MLS или MLS_LABEL в значение типа VARCHAR2
<b>TO_DATE ( char[,fmt [,’nls_lang’]])</b>	Преобразует char в значение типа DATE с помощью форматной маски fmt. Если fmt опускается, используется форматная маска для даты, принятая по умолчанию ‘nls_lang’ задает язык, используемый в названиях месяцев и дней
<b>TO_MULTI_BYTE ( char)</b>	Преобразует однобайтовые символы, имеющие многобайтовые эквиваленты, в соответствующие многобайтовые символы
<b>TO_NUMBER (char [,fmt [,’nls_lang’]])</b>	Преобразует char, содержащее число в формате, указанном параметром fmt, в значение типа NUMBER. ‘nls_lang’ задает язык, определяющий символы валют и числовые разделители
<b>TO_SINGLE_BYTE ( char)</b>	Преобразует многобайтовые символы, имеющие однобайтовые эквиваленты, в соответствующие однобайтовые символы



#### Пример 12:

Приведите пример использования функций преобразования

--	--

## Элементы формата числа для TO\_CHAR

В следующей таблице перечислены элементы формата числа. Комбинацию этих элементов можно использовать как аргумент *fmt* функции TO\_CHAR.

Элемент формата	Пример	Описание
9	'999'	Количество девяток указывает число возвращаемых значащих цифр.
0	'0999'	Добавляет нули перед числом.
\$	'\$9999'	Добавляет знак доллара перед числом.
B	'B9999'	Заменяет нулевые значения пробелами.
MI	'99999MI'	Возвращает знак '-' после отрицательных значений.
S	S9999	Возвращает знак '+' для положительных значений и знак '-' для отрицательных значений в указанную позицию.
PR	'9999PR'	Возвращает отрицательные значения в <угловых скобках>.
D	99D99	Возвращает символ, представляющий десятичную точку, в указанную позицию.
C	9G999	Возвращает символ разделения цифр на группы в указанную позицию.
C	C999	Возвращает международной знак валюты в указанную позицию.
L	L999	Возвращает знак местной валюты в указанную позицию.
,	'9,999'	Возвращает запятую в указанную позицию.
.	'99.99'	Возвращает точку в указанную позицию.
V	'999V99'	Умножает значение на $10^n$ , где <i>n</i> количество девяток после 'V'.
EEEE	'9.999EEEE'	Возвращает значение в нормализованной форме. В <i>fmt</i> должно быть ровно четыре буквы 'E'.
RN или rn	RN	Возвращает римские цифры прописными или строчными буквами (целое число в диапазоне от 1 до 3999).
DATE	'DATE'	Возвращает значение, преобразованное из даты юлианского календаря в формат 'MM/DD/YY'.



### Пример 13:

Приведите пример использования форматных масок для to\_char()

## Другие функции

Функция	Возвращаемое значение
<b>DECODE</b> (expr, search1, return1, [search2, return2, ]...[default])	Если expr равно search, возвращается соответствующий результат return. Если совпадающей пары не найдено, возвращается default.
<b>DUMP</b> (expr[, return_format [, art_position[, length]]])	Expr во внутреннем формате Oracle
<b>GREATEST</b> (expr[, expr]...)	Наибольшее значение expr
<b>LEAST</b> (expr[, expr]...)	Наименьшее значение expr
<b>NVL</b> (expr1, expr2)	Возвращает expr2, если expr1 имеет пустое значение, в противном случае возвращает expr1.
<b>UID</b>	Целое число, которое уникально идентифицирует текущего пользователя.
<b>USER</b>	Имя текущего пользователя ORACLE.
<b>USERENV</b> (option)	Возвращает информацию о текущем сеансе. Аргументы помещаются в одиночных кавычках. Аргументы: ENTRYID, SESSIONSID, TERMINAL, LANGUAGE или LABEL.
<b>VSIZE</b> (expr)	Длина в байтах внутреннего представления для expr.



### Пример 14:

Приведите пример использования дополнительных функций

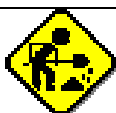
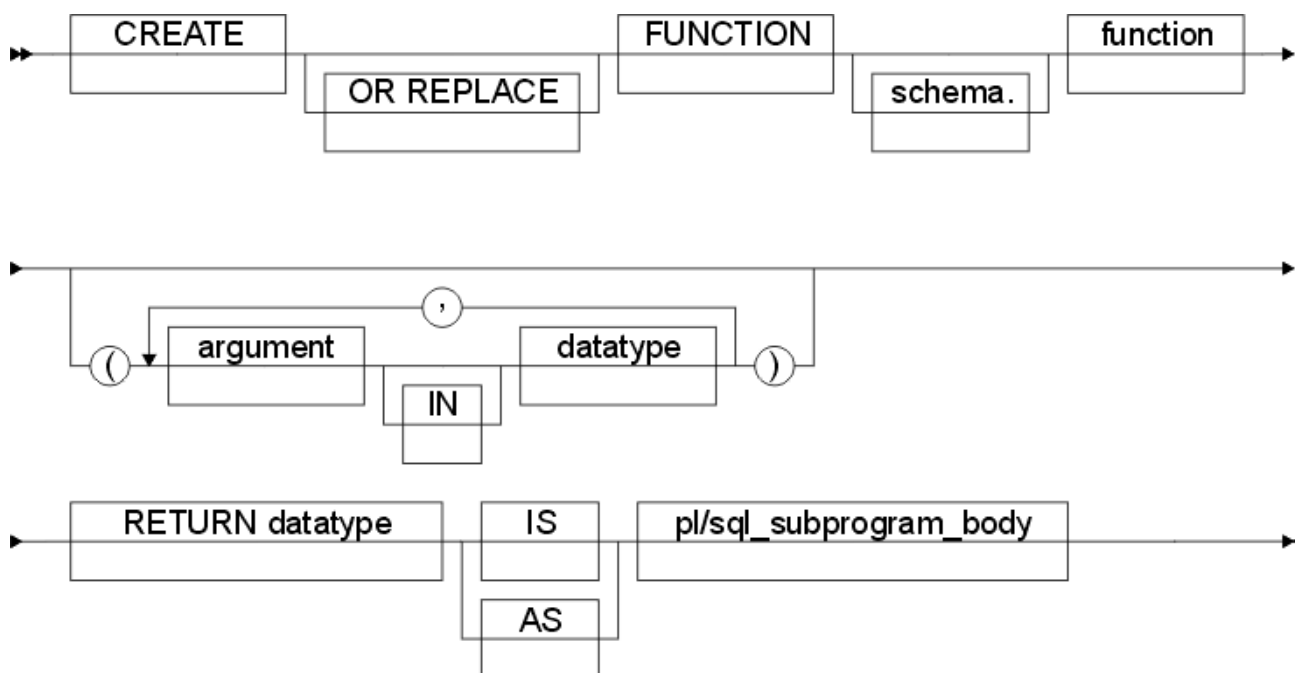


### Задание:

Разработать ненаименованный PL/SQL блок выполняющий расчет времени (в разных форматах) до сессии. Диапазоны дат задаются с клавиатуры.

## НАИМЕНОВАННЫЕ PL/SQL БЛОКИ

Создание функции:



### Пример 15:

Разработать функцию, вычисляющую площадь круга

```

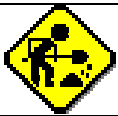
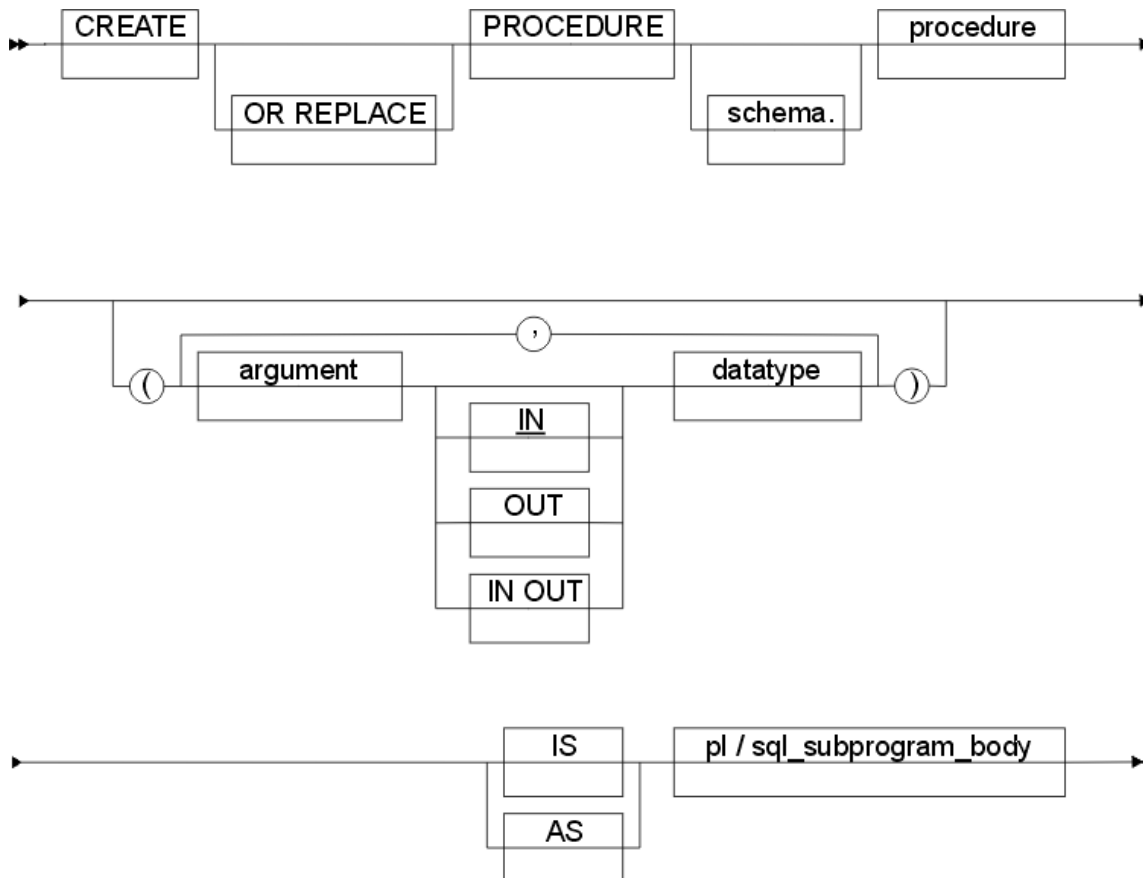
create or replace
function lokrug(radius
real) return real is
    l real;
begin
    l:=3.14*radius*2;
    return(l);
end;
/

create or replace
function lokrug1(radius
real) return real is
begin

return(3.14*radius*2);

end;
    
```

## Создание процедуры



### Пример 16:

Разработать процедуру, вычисляющую площадь круга и длину окружности

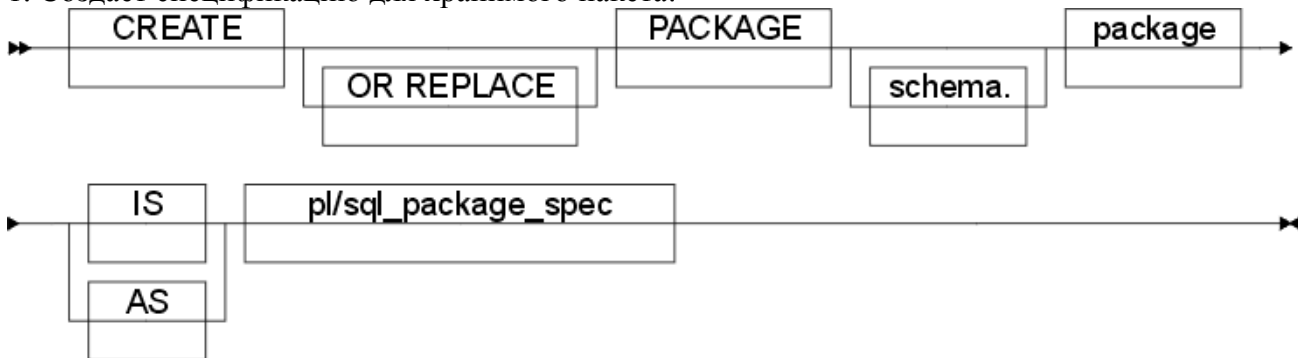
Prompt формируем отчет по всем объектам созд. за промежуток времени

```

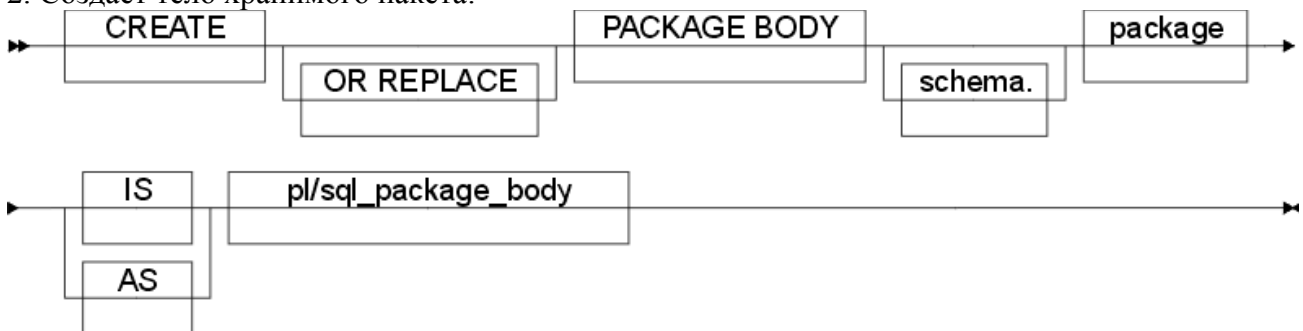
create or replace procedure
prot_user is
db date:=&db;
de date:=&de;
a char(20); b char(20); c
date; e date;
begin
  select
    substr(object_name,1,20)
  name,
    substr(object_type,1,20)
    type_obj,
    created, last_ddl_time
  into a,b,c,e
  from user_objects
  where last_ddl_time>=db
  and
    last_ddl_time<=de;
end;/
  
```

## Создание пакета

1. Создает спецификацию для хранимого пакета:



2. Создает тело хранимого пакета:

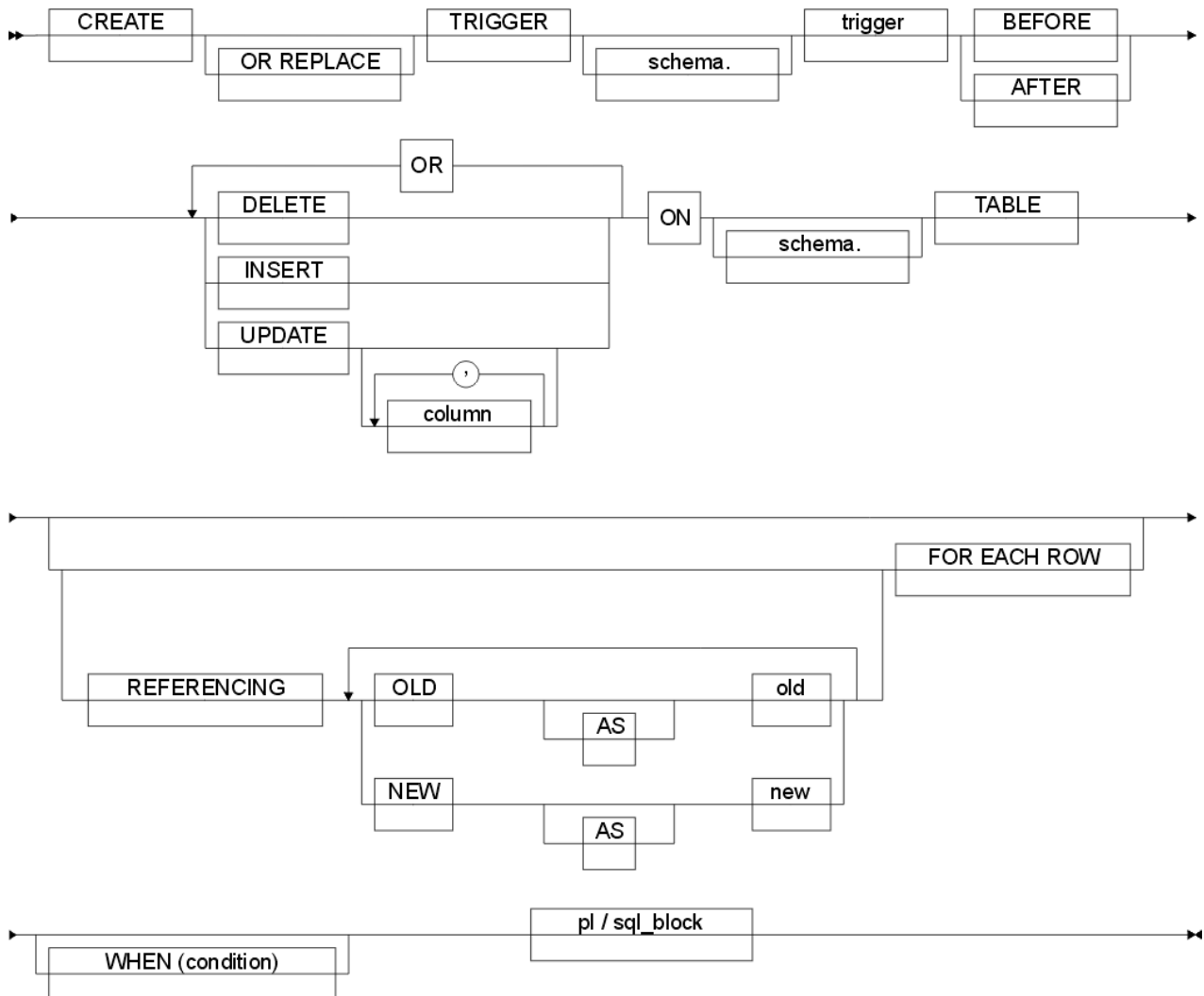


### Пример 17:

Разработать пакет – геометрический калькулятор, который по введенному значению радиуса вычисляет площадь круга и длину окружности (используя функции и процедуры)

Вызов функций и процедур пакета:

## Создание триггера



### Пример 18:



Разработать триггеры для всех таблиц вашей схемы, обеспечивающих автоматическую вставку уникального значения поля ID.

```

create or replace
trigger tr_pay_telefon
before insert
on pay_telefon
for each row
begin
    select
        s_pt_id.nextval
        into :new.pt_id
        from dual;
end;
/
  
```



## Создание констрейтов

	<b>Пример 19:</b> Разработать констрейты для всех связей таблиц вашей инфологической модели (см сем 1), обеспечивающих ссылочную целостность данных.
<pre>ALTER TABLE abonents   ADD CONSTRAINT     C_abonents_ab_kateg   FOREIGN KEY (ab_kateg) REFERENCES   list_kategs(lk_id);  ALTER TABLE telefons   ADD CONSTRAINT     C_telefons_tel_ab_num   FOREIGN KEY (tel_ab_num) REFERENCES   abonents(ab_num);</pre>	
	<b>Пример 20:</b> Проверить корректную работу созданных констрейтов
<pre>prompt Неправильная вставка данных при констрейтах  insert into telefons (tel_num,tel_ab_num) values('12345',1);  prompt Правильная вставка данных при констрейтах  insert into telefons (tel_num,tel_ab_num) values('12345',null); insert into telefons (tel_num,tel_ab_num) values('20012',10005); commit;</pre>	

## Контрольные вопросы

1. Принципы разработки ненаименованных PL/SQL блоков
2. Принципы разработки и использования наименованных PL/SQL блоков.

Отчет по лабораторной работе № 6 «Разработка подсистемы ввода/вывода»			
дата	Оценка (max 5)	Бонус за сложность	подпись

### Цели работы:

Отработка технологий ввода/вывода информации в информационных системах. Итоговая лабораторная работа систематизирует и обобщает все методы применения PL/SQL и PHP для реализации модулей информационных систем с использованием СУБД Oracle.

### Задачи работы:

**Подсистема ввода:** Ввод данных скриптами, ввод данных в формы пользовательского интерфейса, ввод засекреченных данных, ввод данных из файлов, элементы ввода данных с внешних устройств (сканеров) и т.п.

**Подсистема вывода:** Форматированный вывод отчетов с помощью процедур в SQL+, форматированный вывод отчетов средствами пользовательского интерфейса, вывод отчетов в текстовые файлы, вывод форматированных отчетов в файлы разных форматов (\*.pdf, \*.xls и т.п.) и т.п.

**Итог:** Результат модуля АСУ отдела фирмы согласно варианту задания

### Краткий конспект теоретической части (ответы на контрольные вопросы)

Принципы построения подсистемы ввода информации

---

---

---

---

---

---

---

---

---

---

Принципы построения подсистемы вывода информации

---

---

---

---

---

---

---

---

---

---

Описание функционала модуля АСУ фирмы по варианту задания

---

---

---

---

---

---

---

---

---

---

## 1. Подсистема ввода данных

### 1.1. Ввод данных посредством SQL скриптов

Для модуля АСУ согласно варианту (отделу фирмы) приведите пример ввода данных в связанные таблицы SQL скриптами (пример заполнения справочников)

Скрипт:

[illegible]

---

Проверка:

## 1.2. Ввод данных посредством интерактивных интерфейсных форм на РНР

Для модуля АСУ согласно варианту (отделу фирмы) приведите пример ввода данных в связанные таблицы через интерактивные РНР формы

Скрипт:

This image shows a single sheet of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page. There are approximately 20 lines visible. The paper has a slight shadow on the right side, suggesting it's resting on a surface.

Скриншот интерактивной формы:

<p><b>1.3. Ввод засекреченных данных</b></p> <p>Для модуля АСУ согласно варианту (отделу фирмы) приведите пример ввода засекреченных данных (модуль авторизации)</p>
--

[illegible]

Скриншот:

**1.4. Ввод данных из файла**  
Для модуля АСУ согласно варианту (отделу фирмы) приведите пример ввода данных из текстового файла в БД.

**1.4. Ввод данных из файла**  
Для модуля АСУ согласно варианту (отделу фирмы) приведите пример ввода данных из текстового файла в БД.

[illegible]

Проверка:

**1.5 Ввод данных посредством внешних устройств**  
Для модуля АСУ согласно варианту (отделу фирмы) приведите пример ввода данных при помощи сканера штрих кодов (<http://oracle.iu4.bmstu.ru/grid/sem/sem8/index.php>).

**1.5 Ввод данных посредством внешних устройств**  
Для модуля АСУ согласно варианту (отделу фирмы) приведите пример ввода данных при помощи сканера штрих кодов (<http://oracle.iu4.bmstu.ru/grid/sem/sem8/index.php>).

[illegible]

Проверка:

## 2. Подсистема вывода данных

Реализовать различными способами вывода информации получение отчетов по следующим задачам для схемы HR:

### Задание 2.1: Решить задачу и организовать форматированный вывод отчетов с помощью процедур в SQL+:

Отобразить всех сотрудников, находящихся в подчинении вице-президента по фамилии Kochhar. В выходной таблице должно быть 2 столбца:

1. Фамилия сотрудника. Первая строка должна содержать Kochhar без отступа, вторая - сотрудника, непосредственно подчиняющегося Kochhar с отступом в 2, далее - сотрудники на третьем уровне подчинения с отступом в 4 и т.д. Для идентификации отступа использовать символ . (точка)
2. Второй столбец должен содержать фамилии сотрудников, отражающих иерархию подчинения, разделённых знаком /

#### Дополнительные требования к выполнению:

Пример фрагмента вывода:

Фамилия	Подчинение
Kochhar	/Kochhar
Baer	/Kochhar/Baer
Greenberg	/Kochhar/Greenberg
Chen	/Kochhar/Greenberg/Chen
. . . . .	
Urman	/Kochhar/Greenberg/Urman
Higgins	/Kochhar/Higgins
Gietz	/Kochhar/Higgins/Gietz
Mavris	/Kochhar/Mavris
Whalen	/Kochhar/Whalen

Фамилии подчинённых должны быть упорядочены в алфавитном порядке на КАЖДОМ уровне иерархии

Подсказки:

1. Для реализации отступов использовать функцию SQL lpad и псевдостолбец level
2. Для перечисления фамилий в иерархии во втором столбце использовать функцию Oracle 9i sys\_connect\_by\_path.
3. Для сортировки на каждом уровне иерархии использовать конструкцию order siblings by.

Скрипт:

---

---

---

---

---

---

---

---

---

---

Вид отчета:

**Задание 2.2: Решить задачу и организовать форматированный вывод отчетов средствами пользовательского интерфейса на РНР:**

Для каждого данного сотрудника, имеющего оклад выше среднего в его отделе, отобразить всех сотрудников его отдела, имеющих оклад, больше чем у данного сотрудника.

Вывести данные для отделов с номерами 60 и 80.

Выходная таблица должна содержать следующие столбцы:

1. Фамилия данного сотрудника
2. Оклад данного сотрудника
3. Фамилия сотрудника, с большим окладом
4. Оклад сотрудника, с большим окладом
5. Отдел
6. Средний оклад по отделу (округлить до 2-х знаков после запятой)

Данные в выходной таблице должны быть упорядочены по номеру отдела, окладу данного сотрудника, фамилии данного сотрудника, окладу и фамилии сотрудника с большим чем у данного окладом.

Скрипт:

---

---

---

---

---

---

---

---

---

---

Вид отчета:



### Задание 2.3: Решить задачу и организовать вывод отчетов в текстовые файлы

Имеется: таблица с тремя столбцами: именем, фамилией и коэффициентом размножения, созданная и загруженная следующим образом:

```
drop table EMP_SELECTED;
create table EMP_SELECTED (
    First_name varchar2(20) not null
    , Last_name  varchar2(20) not null
    , N integer not null);
insert into EMP_SELECTED values('Ellen', 'ABEL', 3);
insert into EMP_SELECTED values('Matthew', 'WEISS', 5);
commit;
```

Требуется написать запрос, выводящий на печать таблицу, содержащую строки с именами и фамилиями двух сотрудников. Число строк для каждого сотрудника должно определяться коэффициентом размножения (столбец N исходной таблице). То есть, должны быть 3 строки для сотрудника Ellen ABEL и 5 строк для Matthew WEISS. Строки должны быть объединены в группы и отсортированы по фамилии и имени. Кроме того, должны быть пронумерованы элементы внутри группы и присутствовать сквозная нумерация. Этот select должен работать для произвольного количества строк в исходной таблице EMP\_SELECTED. В выходной таблице должны присутствовать следующие столбцы:

- сквозной номер строки по порядку;
- номер сотрудника в группе;
- имя (First\_name);
- фамилия (Last\_name).

Пример выходного отчёта:

Сквозной №	№ в группе	Имя	Фамилия
1	1	Ellen	ABEL
2	2	Ellen	ABEL
3	3	Ellen	ABEL
4	1	Matthew	WEISS
5	2	Matthew	WEISS
6	3	Matthew	WEISS
7	4	Matthew	WEISS
8	5	Matthew	WEISS

8 rows selected.

Подсказки:

1. Для оформления нумераций воспользоваться аналитической функцией row\_number() OVER.
2. Для получения таблицы натуральных чисел воспользоваться конструкцией connect by Level, появившейся в Oracle 10g, например:

```
select Level from DUAL connect by Level <= 10;
```

выдаёт 10 натуральных чисел, начиная с 1.

(Здесь DUAL стандартная таблица Oracle с одним столбцом и одной строкой)

```
select
    row_number() OVER(order by Last_name, First_name) "Сквозной №"
    , row_number() OVER(PARTITION BY Last_name, First_name Order by
                        Last_name, First_name) "№ в группе"
    , First_name "Имя"
    , Last_name  "Фамилия"
from EMP_SELECTED
    , (select /* Этот select выдаёт таблицу с MAX(N) строками и со значениями в
    столбце 1,2,..max(N) */
        level Lev
    from DUAL
        connect by Level <= (select max(N) from EMP_SELECTED))
where Lev <= N /* Хитрый join */
order by Last name, First name;
```

Сквозной №	№ в группе	Имя	Фамилия
	1	1 Ellen	ABEL
	2	2 Ellen	ABEL
	3	3 Ellen	ABEL
	4	1 Matthew	WEISS
	5	2 Matthew	WEISS
	6	3 Matthew	WEISS
	7	4 Matthew	WEISS
	8	5 Matthew	WEISS

8 rows selected.

Скрипт:

---



---



---



---



---



---



---



---



---



---

#### Задание 2.4: Решить задачу и организовать вывод форматированных отчетов в файлы разных форматов (\*.pdf, \*.xls и т.п.)

Из таблицы EMPLOYEES надо выбрать не более 19-ти различных зарплат (столбец SALARY) среди самых новых сотрудников (столбец HIRE\_DATE), анализируя не более 50 сотрудников. Желательно осуществить выборку одним SQL-запросом

```

Select Salary from
(Select Salary,Max(Hire_Date) Hire_Date from
(Select Hire_Date,Salary from
(Select Hire_Date,Salary from Employees order by Hire_Date Desc,Salary desc)
where RowNum<=50)
Group by Salary
order by 2 desc)
Where RowNum<20

```

Результат:

**Задание 2.4а: Решить задачу 2.4. и организовать форматированный вывод отчетов посредством формирования различных штрихкодов:**

Скрипт:

Вид отчета (комбинации штрихкодов):

**ИТОГО: Вклеить основные интерфейсные формы ввода/вывода разработанного модуля АСУ согласно варианту задания**

## Задания для самоконтроля

### Задание 2.5: Решить задачу и организовать форматированный вывод отчетов с помощью процедур в SQL+:

Оператор отдела кадров при регистрации сотрудников с номерами 194 и 195 присвоила им данные друг друга. Необходимо исправить эту ошибку . Ситуация осложняется тем что у вас нет доступа к другим полям таблицы за исключением поля employee\_id, являющегося первичным ключом. Необходимо решить задачу одним оператором SQL.

Проверка 1:

Вид проверки:

Корректность решения

Эталонный запрос:

```
Update employees set employee_id=Decode(employee_id,194,195,195,194) where employee_id in (194,-195)
```

Дополнительная информация:

Суть проверки – выполнение запроса

```
Select RowID,N from T2 where N in (5,-2)
```

До и после прогона представленного решения

Скрипт:

---

---

---

---

---

---

---

---

---

---

Вид отчета:

**Задание 2.6: Решить задачу и организовать форматированный вывод отчетов с помощью процедур в SQL+:**

Как известно, неделя у разных народов начинается с разных дней. Надо с помощью стандартных функций Oracle создать выражение, вычисляющее номер дня недели (начиная с понедельника), независимо от текущей версии Oracle, NLS-установок и кодировок.

Дополнительные требования к выполнению:

По возможности надо обойтись только стандартными функциями Oracle

Проверка 1:

Каноническое решение для текущей даты:

```
Select (InStr('MONTUEWEDTHUFRISATSUN', To_Char(SysDate, 'DY',  
'NLS_DATE_LANGUAGE = AMERICAN')) + 2) / 3 from dual
```

Скрипт:

---

---

---

---

---

---

---

---

---

Вид отчета:

### Задание 2.7: Решить задачу и организовать форматированный вывод отчетов с помощью процедур в SQL+:

Использование конструкции SQL rollup (задание № 1) Написать запрос, выдающий отчёт о суммарных выплатах сотрудникам, непосредственно подчиняющихся заданному руководителю по идентификаторам должностей (поле Job\_id). Непосредственное подчинение предполагает подчинение на первом уровне. Иными словами, записи о сотрудниках, непосредственно подчиняющихся сотруднику с Employee\_id, равным 101, содержат 101 в поле Manager\_id.

Отчёт должен содержать группы строк. Каждая группа относится к данному руководителю и состоит из регулярных строк, отображающих суммарные выплаты и количество сотрудников на данной должности, непосредственно подчиняющихся этому руководителю. Группу должна завершать строка с итоговыми значениями суммарных выплат и количества сотрудников, для сотрудников, непосредственно подчиняющихся данному руководителю. Итоговая строка не должна содержать значение в поле идентификатора должности.

Кроме того, отчёт должен быть завершён строкой, представляющей общий итог и содержащей сумму выплат и количество сотрудников по всем упомянутым руководителям. В этой строке поля идентификаторов руководителя и должности должны быть пустыми.

Столбцы отчёта:

1. Идентификатор руководителя. Для строки, представляющей общий итог это поле должно быть пустым. Для остальных строк в этом поле представлен соответствующий идентификатор (Manager\_id).

2. Идентификатор должности.

Для регулярных строк здесь должен присутствовать соответствующий идентификатор(Job\_id).

Для итоговых строк и для строки общего итога – пустое значение.

3. Количество сотрудников.

Для регулярных строк – количество сотрудников на данной должности у данного руководителя.

Для итоговых строк – количество сотрудников, находящихся в непосредственном подчинении данного руководителя.

Для строки общего итога – общее количество сотрудников, находящихся в непосредственном подчинении у всех руководителей, представленных в отчёте.

Суммарные выплаты.

Для регулярных строк -суммарные выплаты для сотрудников, находящихся на данной должности у данного руководителя.

Для итоговых строк – суммарные выплаты всем сотрудникам, находящимся в непосредственном подчинении данного руководителя. Для строки общего итога – сумма выплат сотрудникам, находящимся в непосредственном подчинении у всех руководителей, представленных в отчёте.

Месячная суммарная выплата каждому сотруднику представляет собой оклад (столбец Salary) плюс комиссионные (столбец Commission\_pct), представляющие указанную часть оклада (положительное число < 1).

Проверка 1:

Вид проверки: Название проверяемой правильности. Например: «проверка на данных имеющих неопределённые значения» БД: Код базы данных на котором проводится проверка. Для одной проверки используем HR. Код всех дополнительных баз, имеющих данные, отличные от HR согласуем впоследствии дополнительно и впишем сюда.

Скрипт:

---

---

---

---

---

---

---

---

---

---

Вид отчета:

### Задание 2.8: Решить задачу и организовать форматированный вывод отчетов с помощью процедур в SQL+:

Написать запрос, выдающий отчёт о суммарных выплатах сотрудникам, непосредственно подчиняющихся руководителю (задаётся полное имя) по названиям должностей (поле JOBS.Job\_Title).

Отчёт должен содержать группы строк. Каждая группа относится к данному руководителю и состоит из регулярных строк, отображающих суммарные выплаты и количество сотрудников на данной должности, непосредственно подчиняющихся этому руководителю. Группу должна завершать строка с итоговыми значениями суммарных выплат и количества сотрудников, для сотрудников, непосредственно подчиняющихся данному руководителю.

Кроме того, отчёт должен быть завершён строкой, представляющей общий итог и содержащей количество сотрудников и сумму выплат и по всем упомянутым руководителям.

Столбцы отчёта:

1. Полное имя руководителя.

Итоговые строки в этом поле должны содержать полное имя руководителя (First\_name, пробел, Last\_name из таблицы EMPLOYEES) с отступом (несколько точек) за которым следует текстовая константа “итоги:”.

Для строки, представляющей общий итог, это поле должно содержать текстовую константу “О Б Щ И Й” с отступом, представленным несколькими точками.

Для остальных (регулярных) строк в этом поле должно быть полное имя руководителя (First\_name, пробел, Last\_name) без отступа.

2. Название должности.

Для итоговых строк это поле должно содержать название должности руководителя (поле Job\_Title из таблицы JOBS).

и для строки общего итога – пустое значение. а в поле названия должностей – текстовая константа “И Т О Г”.

Для регулярных строк здесь должно быть название должности, которую занимают сотрудники с представленными в следующих полях количеством и суммарными выплатами.

3. Количество сотрудников.

Для строки общего итога – общее количество сотрудников, находящихся в непосредственном подчинении у всех руководителей, представленных в отчёте.

Для итоговых строк – количество сотрудников, находящихся данного руководителя.

Для регулярных строк – количество сотрудников на данной должности в непосредственном подчинении данного руководителя.

Суммарные выплаты.

Для строки общего итога – сумма выплат сотрудникам, находящимся в непосредственном подчинении у всех руководителей, представленных в отчёте.

Для итоговых строк – суммарные выплаты всем сотрудникам, находящимся в непосредственном подчинении данного руководителя. Для регулярных строк -суммарные месячные выплаты для сотрудников, находящихся на данной должности у данного руководителя.

Скрипт:

---

---

---

---

---

---

---

---

---

---

Вид отчета: