

# **Лабораторные работы по курсу Базы данных**

## **Лабораторная работа 1 «Знакомство с PostgreSQL»**

**Москва, 2024**

## Оглавление

1. Теоретическая часть .....	3
1.1. Основные определения курса.....	3
1.2. Основы реляционной теории баз данных .....	3
1.3. Общее устройство PostgreSQL .....	4
1.4. Работа с командной строкой Postgres .....	5
1.5. Создание бэкапа базы данных .....	6
1.6. Работа с <i>pgAdmin</i> .....	6
1.7. Язык программирования SQL .....	8
1.8. Создание новой роли.....	9
1.9. Создание базы данных .....	10
1.10. Работа с программой ОРИОКС .....	12
2. Практическая часть .....	13
2.1. Задание 1.....	13
2.2. Задание 2.....	13
2.3. Задание 3.....	13
Контрольные вопросы.....	14
Список использованной литературы .....	14

## 1. Теоретическая часть

### 1.1. Основные определения курса

Прежде чем приступить к изучению курса баз данных, необходимо дать все основные определения.

**Модель данных** – абстракция, описывающая структуру (организацию) данных и методы их обработки

**База данных** – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами моделирования данных.

**Система управления базами данных (СУБД)** – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Каждый день мы сталкиваемся с огромным количеством информации. Это могут быть формулы по математике, оценки за контрольные, даты по истории, цены на выпечку в буфете. Чтобы легче ориентироваться во всем этом разнообразии данных, каждый из нас стремится их упорядочить, разложить на простые составляющие. Для этого создаются различные схемы, графики и таблицы. Например, гораздо проще анализировать рост цен на булочки с корицей, построив график зависимости их цены от текущей даты. Или запоминать даты по истории, выписав их в столбик, в виде таблицы.

Такой подход не только делает наглядным работу с информацией, но и структурирует её. Данные, хранящиеся в таблице *Microsoft Excel* гораздо проще обрабатывать, чем аналогичные данные, сохраненные через запятую в блокноте. Однако при большом объеме данных, использование обычных таблиц (например, в программе *Microsoft Excel*) становится неэффективным. Для решения подобных задач возможно разработать по определенным правилам базу данных, позволяющую решать задачи хранения и обработки данных более эффективно.

В данном курсе будет рассматриваться объектно-реляционная СУБД **PostgreSQL**. PostgreSQL является одной из наиболее популярных СУБД в настоящее время. Одной из основных её особенностей является открытый исходный код. [1]

### 1.2. Основы реляционной теории баз данных

Базы данных строятся на основе некоторой модели данных – абстракции, описывающей структуру данных и методы их обработки. Существуют различные виды моделей данных. Наиболее известными являются иерархическая, сетевая и реляционные модели. Наибольшую распространенность в настоящее время получила реляционная модель данных. Реляционная модель данных основана на понятии отношения (*relation*). В теории баз данных оно соответствует таблице, состоящей из столбцов (атрибутов) и строк (кортежей).

Среди атрибутов возможно выделить подмножество, обладающее свойствами уникальности и неизбыточности. Такое подмножество называется **Потенциальным ключом**. Из него возможно выбрать один ключ, который будет идентифицировать конкретную запись в таблице. Такой ключ называется **первичным (PK, от Primary Key)**. Если в качестве ключа выбирается дополнительный атрибут, содержащий порядковый номер записи или некоторое независимое от содержимого значения, то такое ключ называется **суррогатным**.

В реляционной базе данных таблицы могут быть связаны между собой. Они будут соотноситься как главные и подчиненные. Одной записи главной таблицы может соответствовать множество записей из подчиненной. Связь происходит посредством первичного и внешнего ключей. **Внешний ключ** – поле подчиненной таблицы, соответствующее первичному ключу главной таблицы.

Рассмотрим простой пример. Предположим, база данных состоит из двух таблиц – Группа, содержащая информацию об учебной группе вуза и Студент – содержащая информацию о студенте, обучающемся в вузе. В качестве первичного ключа выберем Номер группы в таблице Группа – он будет уникальным во всем вузе и Номер студенческого билета, который является уникальным для каждого из студентов. Каждый студент вуза обязательно состоит в одной конкретной группе. В одной группе может быть много студентов. Таким образом, возможно связать между собой эти две таблицы. Таблица Группа будет главной, а Студент – подчиненной. Для связи в таблице Студент создадим поле «Номер группы» и назначим его внешним ключом. Данное поле обязательно должно содержать одно из значений «Номера группы» из таблицы группа – студент не может быть прикреплен к несуществующей группе в вузе.

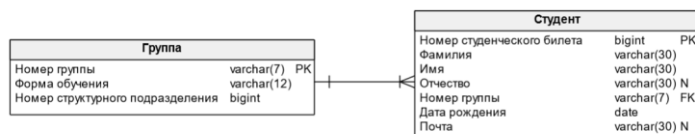


Рисунок 1 Связь Группа – Студент

Более подробно о реляционной теории можно прочитать в источнике [2].

### 1.3.Общее устройство PostgreSQL

СУБД PostgreSQL представляет собой клиент-серверную архитектуру. Рабочий сеанс PostgreSQL включает следующие взаимодействующие процессы:

- Главный серверный процесс, управляющий файлами баз данных, принимающий подключения клиентских приложений и выполняющий различные запросы клиентов к базам данных.
- Клиентское приложение пользователя, с помощью которого выполняются операции в базе данных.

Как и в других типичных клиент-серверных приложениях, клиент и сервер могут располагаться на разных компьютерах. В этом случае они взаимодействуют по сети TCP/IP. Важно не забывать это и понимать, что файлы, доступные на клиентском компьютере, могут быть недоступны (или доступны только под другим именем) на компьютере-сервере. [2]

Взаимодействие между клиентским приложением и сервером происходит посредством запросов. [3]

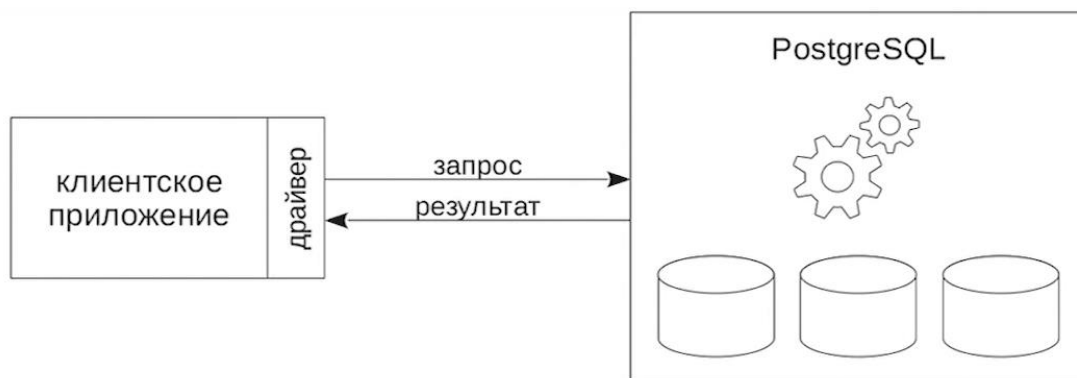


Рисунок 2 Взаимодействие клиента и сервера

Работу с PostgreSQL возможно осуществить из терминала или с помощью графического приложения.

## 1.4. Работа с командной строкой Postgres

PostgreSQL использует метод аутентификации *ident*.

Аутентификация – **процедура проверки подлинности**, например, проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.

Это означает, что выполняется привязка ролей PostgreSQL с соответствующей системной учетной записью ОС *Linux*. Таким образом, чтобы подключиться к базе данных PostgreSQL из терминала по сети TCP/IP необходимо это выполнять от учетной записи, чье имя совпадает с именем роли в PostgreSQL.

В ходе установки была создана учетную запись пользователя postgres, которая связана с используемой по умолчанию ролью Postgres. Чтобы использовать Postgres, вы можете войти в эту учетную запись.

Для этого в командной строке выполните следующую команду:

```
sudo -i -u postgres
```

Система предложит ввести пароль для текущего пользователя. После этого произойдет переключение на учетную запись postgres. Далее возможно подключение к командной строке PostgreSQL:

```
psql
```

После ввода данной команды, произведено соединение с сервером PostgreSQL. По умолчанию, произойдет подключение к базе данных, название которой совпадает с именем роли – postgres. Таким образом, в системе определено **три разные** сущности с одинаковым именем postgres – имена учетной записи *Linux*, роли в *PostgreSQL* и базы данных.



Рисунок 3 Подключение к серверу PostgreSQL

Для работы с командной строкой *PostgreSQL* необходимо использовать специализированные команды. Некоторые из них приведены в таблице ниже.

Таблица 1 Список команд *psql*

Команда	Описание работы
<code>\connect db_name</code>	подключение к базе данных <i>db_name</i>
<code>\dt</code>	вывести все таблицы
<code>\dt+</code>	вывести все таблицы с описанием
<code>\l</code>	вывести список баз данных
<code>\l+</code>	вывести список баз данных с описанием
<code>\dS</code>	вывести системные таблицы
<code>\dv</code>	вывести представления
<code>\dn</code>	вывести все схемы
<code>\du</code>	вывести всех пользователей
<code>\d имя_таблицы</code>	вывести информацию о таблице
<code>\o</code>	пересылка результатов запроса в файл
<code>\di</code>	вывести все индексы
<code>\help</code>	вывести справочник SQL

<code>\i</code>	запуск команды из внешнего файла, например <code>\i /my/directory/my.sql</code>
<code>\?</code>	вывести справочник <code>psql</code>
<code>\q</code>	выход из терминала <code>psql</code>

Если выводимые на экран данные будут превышать допустимые размеры терминала, то произойдёт открытие текстового редактора, в котором будет выведена информация. Для выхода из него нажмите клавишу *q*. Из командной строки возможно выполнять запросы на языке SQL.

Под управлением ОС Windows для работы с PostgreSQL через командную строку можно использовать **SQL Shell (psql)**.

Однако, более удобным и привычным для обучения способом является использование клиентских приложений. Ниже будет рассмотрена работа с клиентским приложением *pgAdmin*.

### 1.5. Создание бэкапа базы данных

Одной из задач администратора баз данных является периодическое создание резервной копии базы данных. Существует несколько способов решить поставленную задачу. Рассмотрим простейший из них. Для создания бэкапа будем использовать встроенную утилиту *pg\_dump*.

Для создания резервной копии в командной строке из-под пользователя postgres выполните следующий скрипт:

```
pg_dump название_БД > название_выходного_файла
```

Например,

```
pg_dump postgres > template_dump.sql
```

Для восстановления резервной копии воспользуйтесь утилитой *psql*

```
psql название_БД < название_выходного_файла
```

Например,

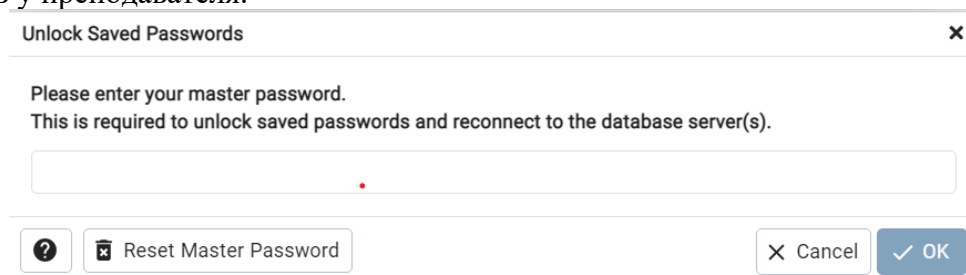
```
psql postgres < template_dump.sql
```

Аналогичные действия в **SQL Shell (psql)** можно выполнить с помощью:

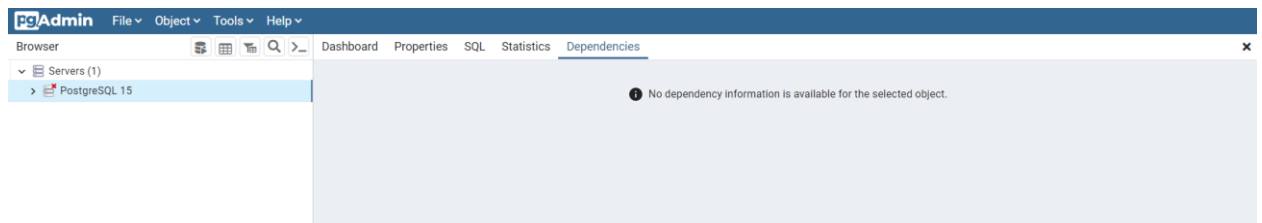
```
\connect db_name;
\i 'E:\script-file.sql';
```

### 1.6. Работа с *pgAdmin*

После запуска программы *pgAdmin* вам будет предложено ввести пароль, указанный при установке программы. Если программа была установлена администратором – спросите пароль у преподавателя.



Основное окно программы выглядит следующим образом:



Для подключения к серверу дважды щелкните на название сервера в окошке слева

▼ Servers (1)

> PostgreSQL 15

При удачном подсоединении появляются три новые вкладки

▼ Servers (1)

▼ PostgreSQL 15

> Databases (1)

> Login/Group Roles

> Tablespaces

Первая вкладка – Database содержит всю информацию о хранимых базах данных. На текущий момент база данных всего одна – **postgres**

▼ Servers (1)

▼ PostgreSQL 15

▼ Databases (1)

> postgres

> Login/Group Roles (13)

> Tablespaces (2)

Вторая вкладка – Login/Group Roles. В ней содержатся все созданные роли и группы, в которые данные роли могут входить. Это предназначено для разделения прав пользователей базы данных, например, между администратором и программистом. По умолчанию создана одна роль – *postgres*.

▼ Servers (1)

▼ PostgreSQL 15

> Databases (1)

▼ Login/Group Roles (13)

pg\_checkpoint

pg\_database\_owner

pg\_execute\_server\_program

pg\_monitor

pg\_read\_all\_data

pg\_read\_all\_settings

pg\_read\_all\_stats

pg\_read\_server\_files

pg\_signal\_backend

pg\_stat\_scan\_tables

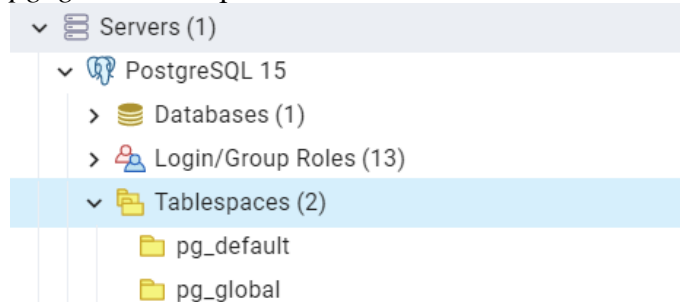
pg\_write\_all\_data

pg\_write\_server\_files

postgres

> Tablespaces (2)

Третья вкладка – Tablespaces. В ней располагаются *табличные пространства*, которые определяют физическое расположение данных. Например, табличные пространства возможно использовать, чтобы расположить архивные данные на медленных носителях, а данные, с которыми идет постоянная работа, на быстрых. При инициализации создается два табличных пространства - *pg\_default*, для хранения данных по умолчанию и *pg\_global* для хранения общих объектов.



## 1.7. Язык программирования SQL

Работа с базами данных будет осуществляться с помощью языка программирования SQL. В отличие от знакомых вам императивных языков программирования C, C++, Python, Pascal и т.п. SQL является декларативным языком. [4]

**Декларативное программирование** (от *declare* - описание) — парадигма программирования, в которой задаётся спецификация решения задачи, то есть описывается ожидаемый результат, а не способ его получения.

Сравним между собой два подхода. Предположим, что нам необходимо найти в некоторой базе данных, содержащей информацию о студентах вуза, всех молодых людей по имени Александр. Напишем на псевдо-языке программирования решение данной задачи.

Императивный подход	Декларативный подход
Для всех строчек таблицы Студент	<b>Выбери</b> всю информацию
Если (имя студента = Александр)	<b>Из</b> таблицы Студент
То выведи информацию о нем на экран	<b>Где</b> имя студента = Александр

Как можно увидеть из таблицы, в первом случае мы задаем последовательность действий, которые приведут к желаемому результату. Во втором случае – описываем результат того, что хотим получить.

Приведем более простой пример. Предположим, мы хотим приготовить на обед салат овощей. Императивный подход к решению задачи выглядит следующим образом:

*Купить огурцы, помидоры, лук, редис, оливковое масло;*

*Порезать огурцы, помидоры, лук, редис;*

*Полить оливковым маслом.*


При декларативном подходе описание будет звучать так: *хочу на обед салат из свежих овощей, заправленный оливковым маслом.*

Язык SQL включает в себя операторы, инструкции, вычисляемые функции.

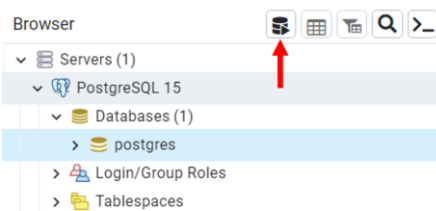
Операторы SQL делятся на:

- операторы определения данных (Data Definition Language, DDL)
- операторы манипуляции данными (Data Manipulation Language, DML)
- операторы определения доступа к данным (Data Control Language, DCL)
- операторы управления транзакциями (Transaction Control Language, TCL)

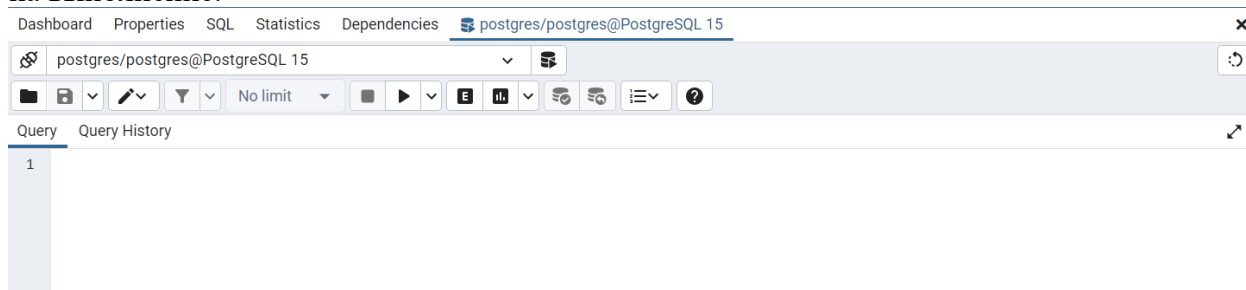
Более подробно данные операторы будут рассмотрены в дальнейшем.

Для того, чтобы создать запрос на языке SQL в программе *pgAdmin* необходимо воспользоваться утилитой Query tool. Для этого перейдите во вкладку Databases – postgres и нажмите на символ .





Перед вами откроется командное окно, в которое возможно вводить запросы и запускать их на выполнение.



Обратим внимание на строку с подключением. Она записана в формате «база данных/роль@сервер». Для данного примера база данных называется *postgres*, пользователь – *postgres*, сервер – PostgreSQL 15

### 1.8. Создание новой роли.

При работе с базами данных важно разделять права доступа между различными пользователями. В первую очередь это необходимо в целях безопасности. Например, рядовой сотрудник не должен иметь возможности удалить или испортить базу данных. С подобными разграничениями мы сталкивались при работе с электронной системой ОРИОКС. При подключении в качестве студента имеется возможность лишь просматривать оценки, но при авторизации в качестве преподавателя – их выставять и редактировать.

**Авторизация** — процесс предоставления пользователю или группе пользователей определенных разрешений, прав доступа и привилегий в компьютерной системе.

В СУБД PostgreSQL разграничение доступа реализуется с помощью понятий роли и привилегий.<sup>1</sup>

Каждому пользователю в СУБД назначается роль, обладающая определенными привилегиями. Например, определенной роли возможно выделить привилегию только на чтение данных из таблиц.

Для создания роли используется оператор **CREATE ROLE**

**CREATE ROLE** *имя* [ [ WITH ] *параметр* [ ... ] ]

Здесь *параметр*:

```

SUPERUSER | NOSUPERUSER
| CREATEDB | NOCREATEDB
| CREATEROLE | NOCREATEROLE
| INHERIT | NOINHERIT
| LOGIN | NOLOGIN
| REPLICATION | NOREPLICATION
| BYPASSRLS | NOBYPASSRLS
| CONNECTION LIMIT предел_подключений
| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'пароль'

```

<sup>1</sup> В теории баз данных существует разделение между понятиями пользователь (*user*) и роль (*role*). Пользователь – физическое лицо, которому могут быть выделены особые привилегии – роли. Однако, в последних версиях *PostgreSQL* данные определения имеют одинаковый смысл.

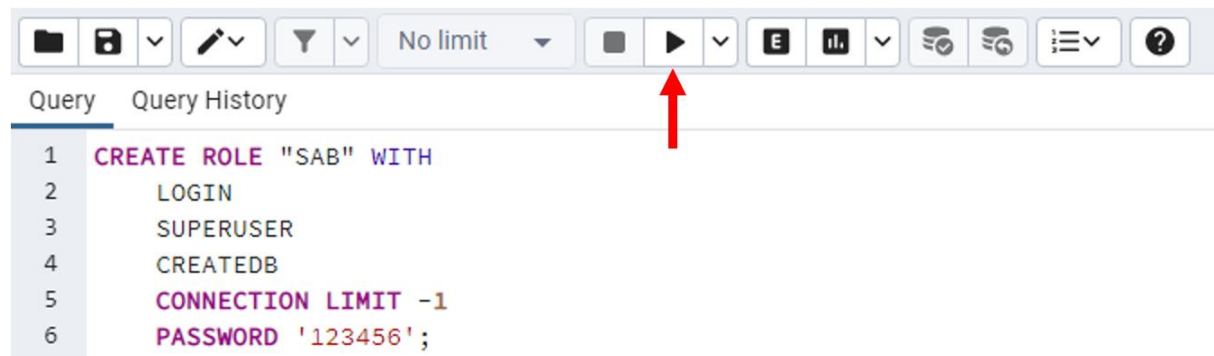
```

| VALID UNTIL 'дата_время'
| IN ROLE имя_роли [, ...]
| IN GROUP имя_роли [, ...]
| ROLE имя_роли [, ...]
| ADMIN имя_роли [, ...]
| USER имя_роли [, ...]
| SYSID uid

```

Подробно о каждом из параметров возможно прочитать в приложении к документации PostgreSQL [2]

Создадим роль, название которой будет содержать ваши инициалы и наделим её правами администратора. Для этого скопируем следующий скрипт в рабочую область и запустим его с помощью символа ▶. Имя пользователя и пароль должны быть выбраны вами.



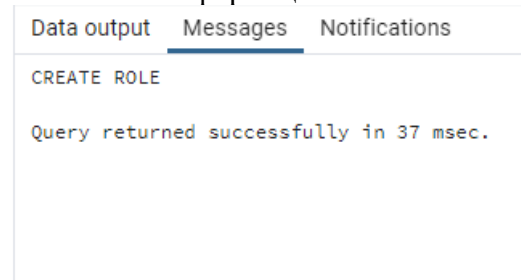
```

CREATE ROLE "SAB" WITH
  LOGIN
  SUPERUSER
  CREATEDB
  CONNECTION LIMIT -1
  PASSWORD '123456';

```

Данный скрипт создает роль SAB, наделяет её привилегиями на вход, создание базы данных и делает её суперпользователем. Созданный пользователь может подключаться неограниченное число раз и имеет пароль для входа 123456.

Обратите внимание, что после успешного выполнения запроса в поле Messages появилась информация об этом.



## 1.9. Создание базы данных

Перейдем непосредственно к работе с базой данных. Описание предметной области учебной базы данных расположено в приложении.

Первым делом необходимо создать базу данных. Для этого существует команда CREATE DATABASE. Её синтаксис представлен ниже.

```

CREATE DATABASE имя
[ [ WITH ] [ OWNER [=] имя_пользователя ]
  [ TEMPLATE [=] шаблон ]
  [ ENCODING [=] кодировка ]
  [ LC_COLLATE [=] категория_сортировки ]

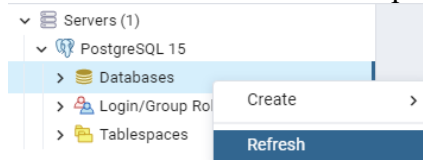
```

```
[ LC_CTYPE [=] категория_типов_символов ]
[ TABLESPACE [=] табл_пространство ]
[ ALLOW_CONNECTIONS [=] разр_подключения ]
[ CONNECTION LIMIT [=] предел_подключений ]
[ IS_TEMPLATE [=] это_шаблон ] ]
```

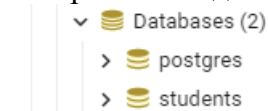
Создадим учебную базу данных с информацией о студентах вуза. Для этого выполним запрос:

**CREATE DATABASE students;**

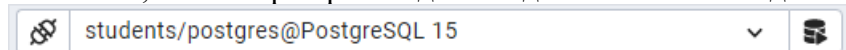
Обновим информацию о базах данных на сервере. Для этого щелкнем по строке Databases в левой колонке правой кнопкой мыши и выберем пункт *Refresh*.



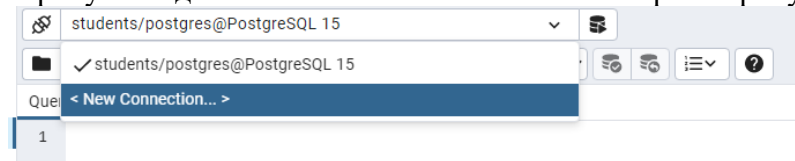
Раскроем выпадающий список и убедимся, что появилась новая база данных.



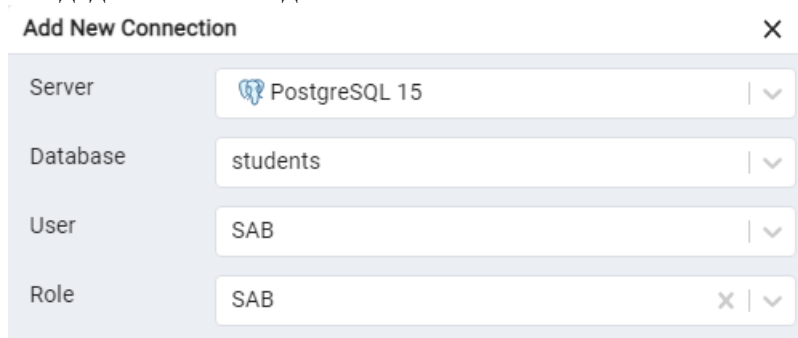
Нажмем на созданную базу данных и создадим новый экземпляр Query tool. Обратите внимание, что теперь произведено подключение к базе данных *students*.



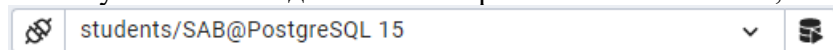
Подключимся к базе данных от имени созданной нами выше роли. Для этого нажмем на строку с соединением и в выпавшем окне выберем строку <New Connection>




Создадим новое соединение



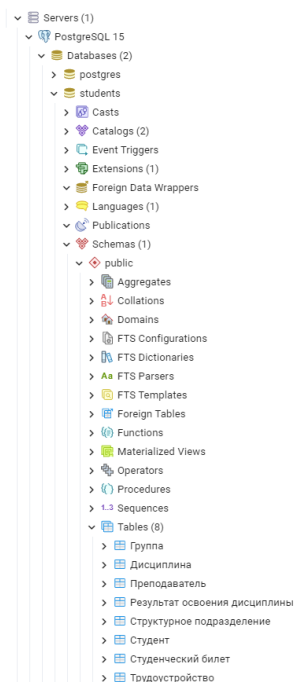
После успешного подключения обратим внимание на то, что имя пользователя изменилось.



Создадим таблицы в базе данных. Для этого откроем скрипт «*create\_database.sql*», нажав

на символ  и запустим его. Если все выполнилось верно, то в поле сообщений появится строчка: «CREATE TABLE Query returned successfully»


Аналогично откроем файл «*insert\_students.sql*» и заполним базу данными.



Данные скрипты будут более подробно рассмотрено в следующих лабораторных работах. Убедимся в том, что все таблицы были созданы. Для этого в выпадающем списке найдем созданные таблицы по пути: «Servers – PostgreSQL 15 – students – Schemas – public – Tables». Нажав правой кнопкой мыши на название таблицы, мы можем выбрать некоторые действия. Рассмотрим некоторые из них.

- Count Rows возвращает число строк в таблице.
- View/Edit Data позволяет вывести на экран содержимое таблицы.

Дважды щелкнув по любой ячейке таблицы, возможно изменить её значение.

После внесенных изменений необходимо зафиксировать их, нажав на символ  или клавишу F6.

### 1.10. Работа с программой ОРИОКС

В качестве демонстрации работы приложения, взаимодействующего с учебной базой данных на курсе предложена программа ORIOKS Simulator. Данная программа написана на языке C++ в среде *QT Creator*. Более подробно о программе вы можете прочитать в приложении.

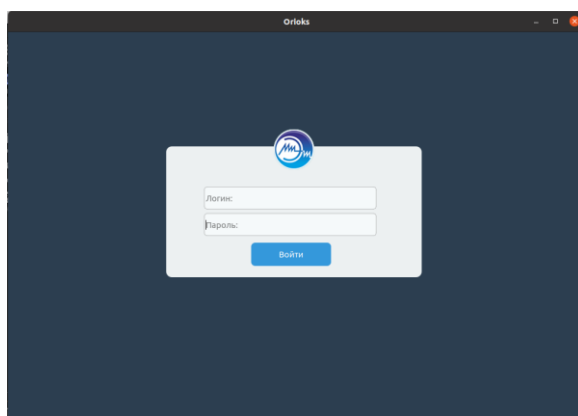


Рисунок 4 Главное окно

Практика	Обучение	Зачеты	Михеев Александр Алексеевич
Контроль и диагностика		5	
Операционные системы		4	
Интерфейсы вычислительных систем		5	
Базы данных		4	
Аналоговая техника		5	
Администрирование локальных вычислительных сетей		5	
Электроника и импульсная техника		5	
Электротехника		5	
Программируемые логические интегральные схемы		4	
Основы цифровой схемотехники		4	
Алгебра и геометрия		5	

Рисунок 5 Окно с интерфейсом студента

## 2. Практическая часть

Вариант к практической части выбирается по формуле:  $V = (N \% 10) + 1$ , где  $N$  – номер в списке группы,  $\%$  - остаток от деления.

### 2.1. Задание 1.

*Создание базы данных*

- 2.1.1. Создайте учебную базу данных *Students*. Подробно о создании базы данных описано в п. 1.9.
- 2.1.2. Подключитесь к созданной базе данных из-под командной строки. Заполните базу данных, используя файл резервной копии.
- 2.1.3. Используя программу *pgAdmin*, ознакомьтесь со схемой данных, содержимым таблиц БД. Определите число строк в каждой из таблиц.
- 2.1.4. Определите, какие таблицы в базе данных *Students* являются главными, а какие для них подчиненными.

### 2.2. Задание 2.

*Администрирование СУБД*

- 2.2.1. Подключитесь к созданной базе данных из-под командной строки. Определите, какой размер на диске занимает таблица *student*?
- 2.2.2. Создайте новую роль «Ваши инициалы junior». Выделите ей привилегии на просмотр данных. Подключитесь от её имени к базе данных *person* и попробуйте удалить её с помощью запроса:

**DROP DATABASE students;**

Удалось ли вам это сделать?

### 2.3. Задание 3.

*Редактирование содержимого базы данных*

- 2.3.1. Выполните в соответствии с вариантом задание (см. таблицу ниже) на изменение содержимого базы данных.
- 2.3.2. После внесенных изменений, создайте новую резервную копию базы данных *Students*.

№ варианта	Задание
1	Студентка группы ИТД-33 Коровина Мария Георгиевна пересдала экзамен по Колористике на 5. Исправьте любым из возможных способов её оценку.

2	В связи с ошибкой при заполнении документов, студенту Егорову Артему из группы ИТД-33 назначили неверный номер студенческого билета. Исправьте его первую цифру на 8.
3	Дарья Кондрашова вышла замуж за своего одногруппника отличника и сменила свою фамилию на его. Выполните соответствующее изменение в базе данных.
4	Из-за конфликтов с одногруппниками, Артем Зайцев из группы ИВТ-41 решил перевестись в группу ИВТ-42. Выполните данное изменение.
5	Преподаватель, чье имя совпадает с именем беллетриста из пьесы Чехова повысили ставку на 20% и перевели в институт МПСУ. Исправьте значение в базе данных.
6	Добавьте кафедре маркетинга сокращенное название – МИУП.
7	Измените ЗЕТ дисциплины, которую читает Александр Докучаев на 5.
8	Почта студента, родившегося 20 сентября, изменилась на secondcosmonaut@miet.ru. Произведите данное изменение.
9	После провала на второй пересдаче студент Андрей Алехин был отчислен. Удалите его из базы данных
10	У студента Ивана Белякова был обнаружен пропавший после его рождения отец. Добавьте студенту отчество Алексеевич.

### Контрольные вопросы

1. Что такое база данных?
2. Что такое СУБД?
3. Чем потенциальный ключ отличается от первичного ключа?
4. Опишите работу метода аутентификации *ident*
5. Для чего предназначено создание ролей в СУБД?

### Список использованной литературы

- [1] «Исходный код СУБД postgres,» [В Интернете]. Available: <https://github.com/postgres/postgres>. [Дата обращения: 30 01 2023].
- [2] Документация к PostgreSQL 15.1, 2022.
- [3] Е. Рогов, PostgreSQL изнутри, 1-е ред., Москва: ДМК Пресс, 2023, р. 662.
- [4] Б. А. Новиков, Е. А. Горшкова и Н. Г. Графеева, Основы технологии баз данных, 2-е ред., Москва: ДМК пресс, 2020, р. 582.
- [5] Е. П. Моргунов, PostgreSQL. Основы языка SQL, 1-е ред., Санкт-Петербург: БХВ-Петербург, 2018, р. 336.