**Лабораторные работы по курсу**

**Базы данных**

**Лабораторная работа 1**

**«Знакомство с PostgreSQL»**

**Москва, 2024**

Оглавление

[1. Теоретическая часть 3](#_Toc165224525)

[1.1. Основные понятия теории реляционных баз данных 3](#_Toc165224526)

[1.2. Общее устройство PostgreSQL 4](#_Toc165224527)

[1.3. Язык программирования SQL 5](#_Toc165224528)

[1.4. Аутентификация в PostgreSQL. 6](#_Toc165224529)

[1.5. Подключение к учетной записи. 6](#_Toc165224530)

[1.6. Создание резервной копии базы данных 6](#_Toc165224531)

[1.7. Работа с командной строкой PostgreSQL 7](#_Toc165224532)

[1.8. Работа с *pgAdmin* 8](#_Toc165224533)

[1.9. Создание новой роли. 10](#_Toc165224534)

[1.10. Работа с базами данных 11](#_Toc165224535)

[2. Практическая часть 14](#_Toc165224536)

[2.1. Задание 1. 14](#_Toc165224537)

[2.2. Задание 2. 14](#_Toc165224538)

[2.3. Задание 3. 14](#_Toc165224539)

[Контрольные вопросы 15](#_Toc165224540)

[Список использованной литературы 15](#_Toc165224541)

1. Теоретическая часть
   1. Основные понятия теории реляционных баз данных

Каждый день мы сталкиваемся с огромным количеством информации. Это могут быть формулы по математике, оценки за контрольные, даты исторических событий, цены на выпечку в буфете. Чтобы легче ориентироваться во всем этом разнообразии данных, каждый из нас стремиться 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 делает работу с информацией более наглядной. Кроме того, структурированные данные, хранящиеся, например, в таблице *Microsoft* *Excel* проще обрабатывать, чем данные, записанные через запятую в блокноте. Однако при большом объеме информации использование обычных таблиц (*Excel)* становится неэффективным. В этом случае для хранения структурированной информации используют базы данных, а для ее обработки - одну из существующих СУБД.

**База данных (БД)** – совокупность данных, организованных по определённым правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ. Эти данные относятся к определённой предметной области и организованы таким образом, что могут быть использованы для решения многих задач многими пользователями.

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

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]

Базы данных строятся на основе некоторой модели данных.

**Модель данных** – абстракция, описывающая структуру (организацию) 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**).

Приведем пример.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер в списке | Номер студенческого билета | Фамилия | Имя | Отчество | Группа | Дата рождения | Номер паспорта |
| 1 | 825644 | Семенов | Павел | Романович | ИВТ-31 | 23.10.04 | 4356 678958 |
| 2 | 845227 | Семенов | Иван | Иванович | ИВТ-32 | 21.07.05 | 4456 458326 |
| 3 | 861247 | Орлова | Екатерина | Леонидовна | ИВТ-34 | 02.10.03 | 4244 523417 |

В нашей таблице, являющейся физической реализацией отношения, каждая из трех строк с информацией о студенте - кортеж, а столбцы «Номер», «Группа», «Фамилия», «Номер зачетной книжки», «Дата рождения», «Номер паспорта» – атрибуты.

Среди атрибутов или совокупности атрибутов можно выделить такие, которые обладают свойством уникальности и неизбыточности. Они называются **потенциальными ключами.** Из них можно выделить один, который будет **первичным** ключом, остальные – **альтернативными**. Первичный ключ должен однозначно идентифицировать каждую запись в таблице. Если среди атрибутов потенциальных ключей нет, или их неудобно использовать, то возможно создать дополнительный атрибут (столбец). Дополнительный атрибут должен быть также уникален и неизбыточен для каждой записи. Содержательного значения он не несет. Такой дополнительный атрибут называется **суррогатным** ключом.

В примере «Номер студенческого билета», «Номер паспорта», «Фамилия+Дата рождения» - потенциальные ключи. Для однозначной идентификации достаточно одного, выберем «Номер зачетной книжки», тогда он будет первичным ключом, а «Номер паспорта», «Фамилия+Дата рождения» альтернативными. В случае, если нам неудобно использовать потенциальные ключи, можно ввести дополнительный атрибут (например, «Номер в списке»), он будет суррогатным ключом.

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 в таблице *Группа* выберем «Номер группы» (он будет уникальным во всем вузе). 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 возьмём поле «Номер группы» и назначим его внешним ключом. Данное поля обязательно должно содержать одно из значений «Номера группы» из таблицы *Группа* – студент не может быть прикреплен к несуществующей в вузе группе.

Изображение выглядит как текст, снимок экрана, чек, Шрифт

Автоматически созданное описание

Рисунок 1 Связь Группа – Студент

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

* 1. 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, которому требуется выполнять различные операции над базами данных. Это может быть текстовая утилита, графическое приложение, веб-сервер, использующий базу данных для отображения веб-страниц.

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]

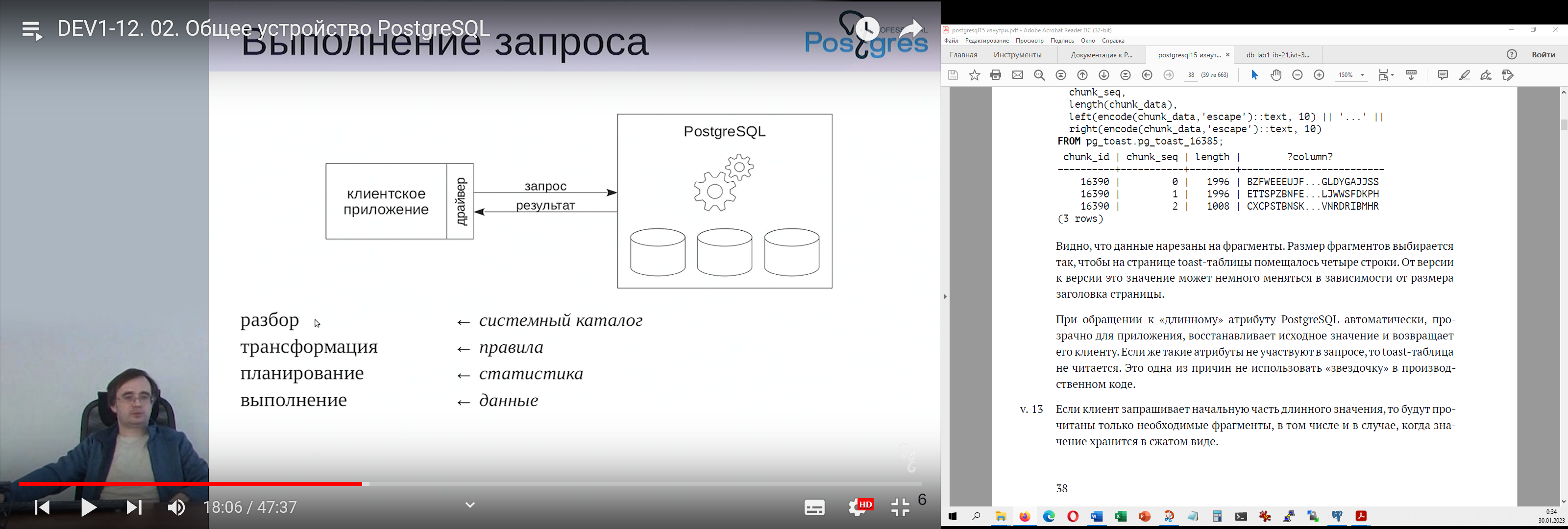


Рисунок 2 Взаимодействие клиента и сервера

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

* 1. Язык программирования SQL

Работа с базами данных будет осуществляться с помощью языка программирования SQL. В отличии от знакомых вам императивных языков программирования С, С++, 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  **Если** (имя студента = Александр)  **То** выведи информацию о нем на экран | **Выбери** всю информацию  **Из** таблицы Студент  **Где** 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.

* 1. Аутентификация в PostgreSQL.

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 выставлять и редактировать.

PostgreSQL использует концепцию ролей (*roles*) для управления разрешениями на

доступ к базе данных. Роль можно рассматривать как пользователя базы данных или как группу пользователей. Для каждой роли существуют свои права, от возможности любого изменения (администратор), до простого просмотра (простой пользователь). Роли назначаются администратором базы данных.

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.

При подключении к серверу базы данных клиентское приложение указывает имя пользователя PostgreSQL, так же, как и при обычном входе пользователя на компьютер с ОС Unix. При работе в среде SQL по имени пользователя определяется, какие у него есть права доступа к объектам базы данных. Для соотнесения имени пользователя и его роли в PostgreSQL служит аутентификация.

**Существуют понятия, которые необходимо различать:**

**Аутентификация** — это процесс подтверждения права на доступ с помощью ввода пароля, пин-кода, использования биометрических данных и других способов.

**Идентификация** используется для определения, существует ли конкретный пользователь в системе. Проводится, например, по номеру телефона или логину.

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

PostgreSQL предлагает несколько различных методов аутентификации клиентов. Метод аутентификации конкретного клиентского соединения может основываться на адресе компьютера клиента, имени базы данных, имени пользователя. После установки Postgres настроена на использование аутентификации *ident*, что значит, что выполняется привязка ролей *postgres* с соответствующей системной учетной записи Unix/Linux.

* 1. Подключение к учетной записи.

В ходе установки была создана учетная запись пользователя postgres, которая связана с используемой по умолчанию ролью *postgres*. Чтобы использовать postgres, вы должны войти в эту учетную запись. Сделаем это с помощью утилиты sudo операционной системы Linux.

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.

* 1. Создание резервной копии 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

Аналогичное действие возможно выполнить с помощью утилиты *pg\_restore*. Для этого файл резервной копии сохраняется утилитой *pg\_dump* в определённом формате с помощью ключей -Fc и далее восстанавливается.

pg\_dump -Fc название\_БД > название\_выходного\_файла

pg\_restore -d название\_БД название\_входного\_файла

* 1. Работа с командной строкой PostgreSQL

Подключение к командной строке PostgreSQLосуществляется следующей командой:

****psql****

После ввода данной команды будет установлено 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.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

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 |
| *\connect db\_name* | подключение к базе данных *db\_name* |
| *\dt* | вывести все таблицы |
| *\dt+* | вывести все таблицы с описанием |
| *\l* | вывести список баз данных |
| *\l+* | вывести список баз данных с описанием |
| *\dS* | вывести системные таблицы |
| *\dv* | вывести представления |
| *\dn* | вывести все схемы |
| *\du* | вывести всех пользователей |
| *\d имя\_таблицы* | вывести информацию о таблице |
| *\o* | пересылка результатов запроса в файл |
| *\di* | вывести все индексы |
| *\help* | вывести справочник SQL |
| *\i* | запуск команды из внешнего файла, например *\i /my/directory/my.sql* |
| *\?* | вывести справочник *psql* |
| *\q* | выход из терминала *psql* |

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*.

Запросы на языке SQL возможно выполнять из командной строки. **Каждая команда SQL должна заканчиваться символом «;»**. Однако, более удобным и привычным для обучения способом является использование клиентских приложений. Ниже будет рассмотрена работа с клиентским приложением *pgAdmin*.

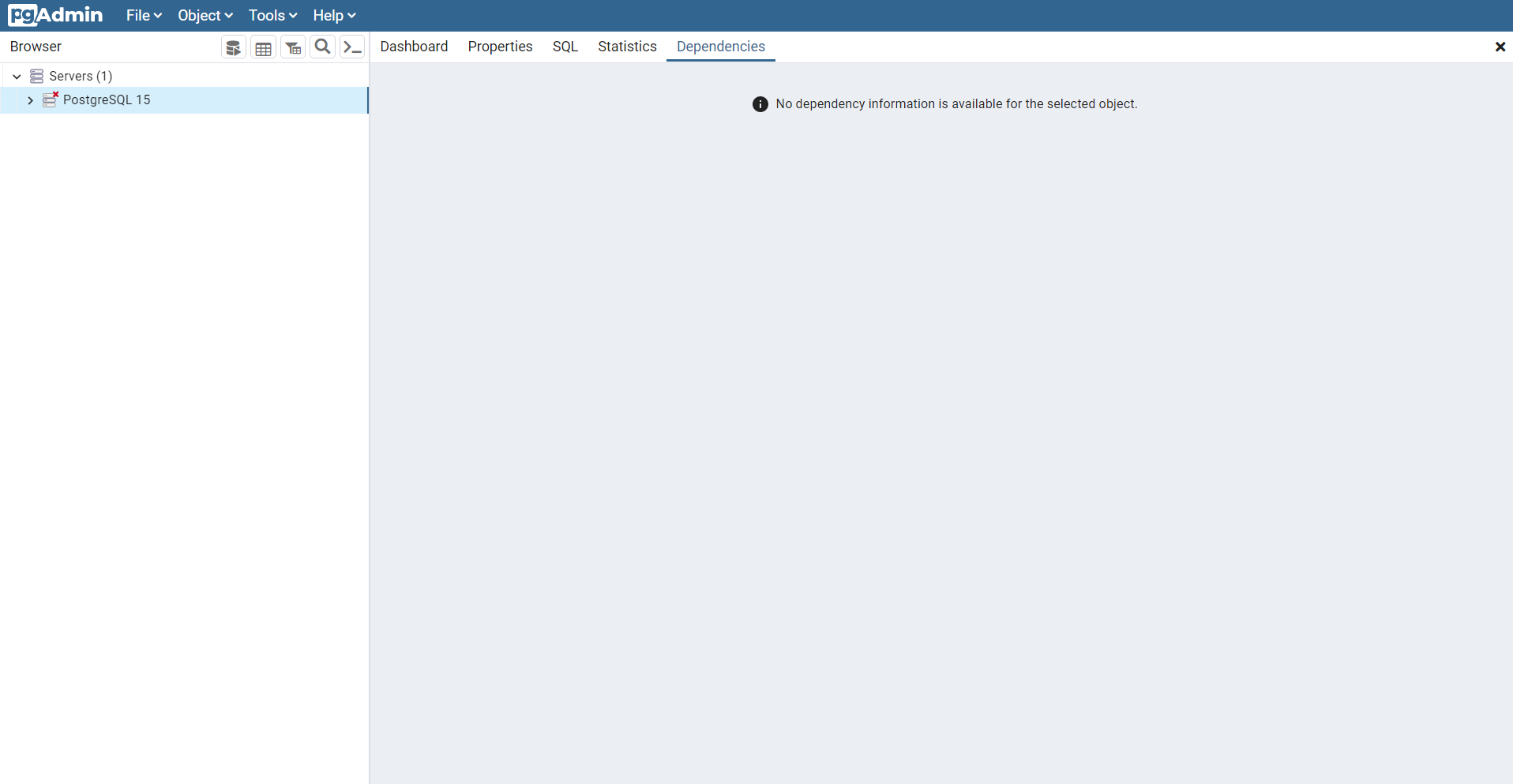
* 1. Работа с *pgAdmin*

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

Изображение выглядит как текст

Автоматически созданное описание

Основное окно программы выглядит следующим образом:



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

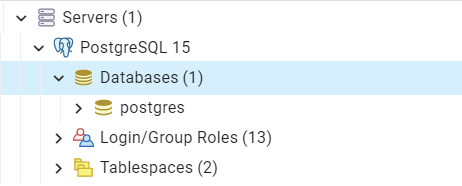


При удачном подсоединении появляются три новые вкладки

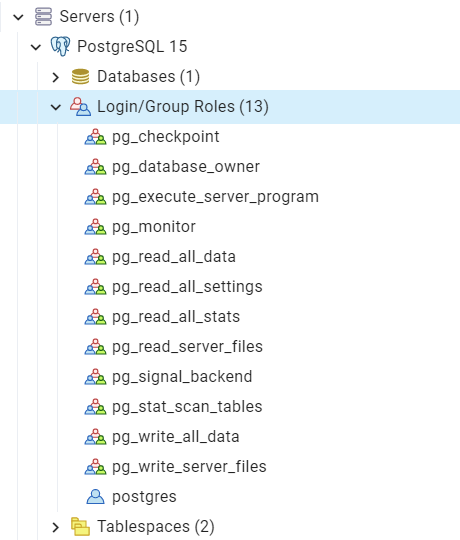
Изображение выглядит как стол

Автоматически созданное описание

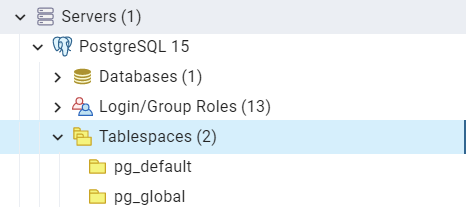
Первая вкладка – Database содержит всю информацию о хранимых базах данных. На текущий момент база данных всего одна – **postgres**



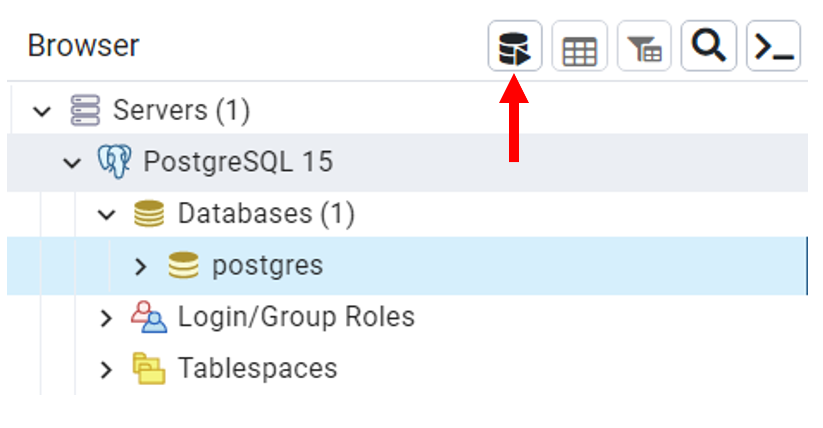
Вторая вкладка – Login/Group Roles. В ней содержаться все созданные роли и группы, в которые данные роли могут входить. Это предназначено для разделения прав пользователей базы данных, например, между администратором и программистом. По умолчанию создана одна роль – *postgres*.



Третья вкладка – Tablespaces. В ней располагаются *табличные пространства,* которыеопределяют 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.



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. Утилита доступна только если курсор установлен на имени БД (в нашем случае**postgres***).*

Изображение выглядит как текст

Автоматически созданное описание

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. Создание новой роли.

В СУБД PostgreSQL разграничение доступа реализуется с помощью понятий роли и привилегий.

Каждому пользователю в СУБД назначается роль, обладающая определенными привилегиями. Данный процесс называется авторизацией. 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*'

| VALID UNTIL '*дата\_время*'

| IN ROLE *имя\_роли* [, ...]

| IN GROUP *имя\_роли* [, ...]

| ROLE *имя\_роли* [, ...]

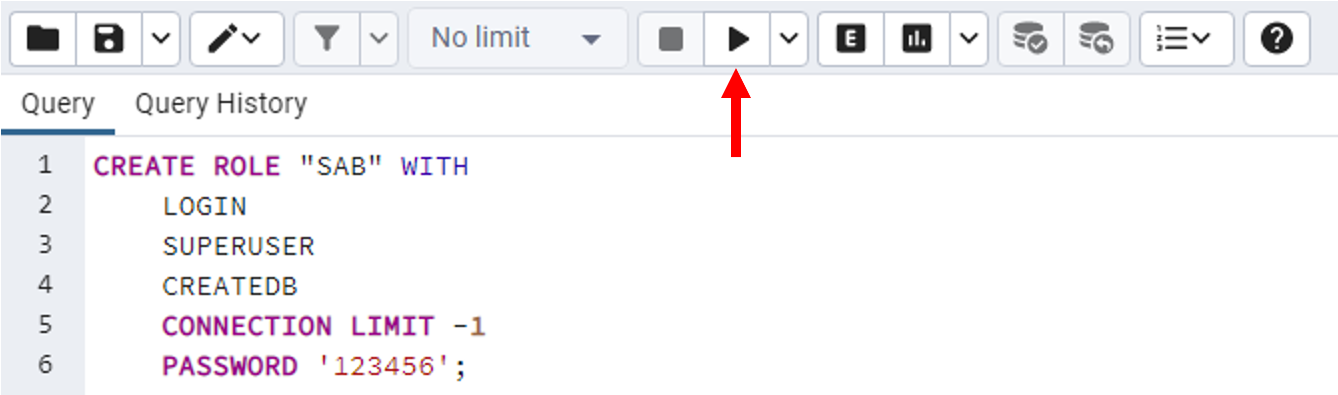
| ADMIN *имя\_роли* [, ...]

| 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 (параметр LOGIN), создание базы данных (CREATEDB) и делает её суперпользователем (SUPERUSER). Созданный пользователь может подключаться неограниченное число раз (CONNECTION LIMIT -1) и имеет пароль для входа 123456.

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

Изображение выглядит как текст

Автоматически созданное описание

* 1. Работа с базами данных

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

* + 1. *Создание базы данных*

Для создания БД служит 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* ]

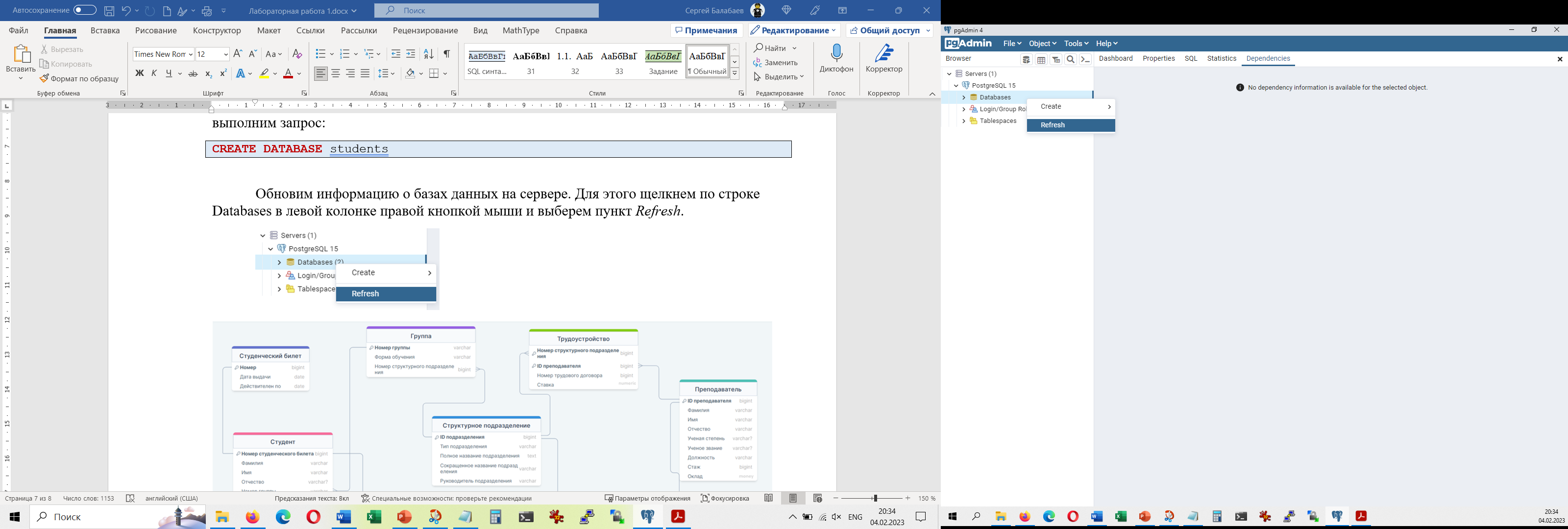
[ IS\_TEMPLATE [=] *это\_шаблон* ] ]

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

CREATE DATABASE students;

Напомним, этот запрос возможно выполнять как из командной строки PostgreSQL, так и с помощью программы *pgAdmin.*

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.

Изображение выглядит как текст

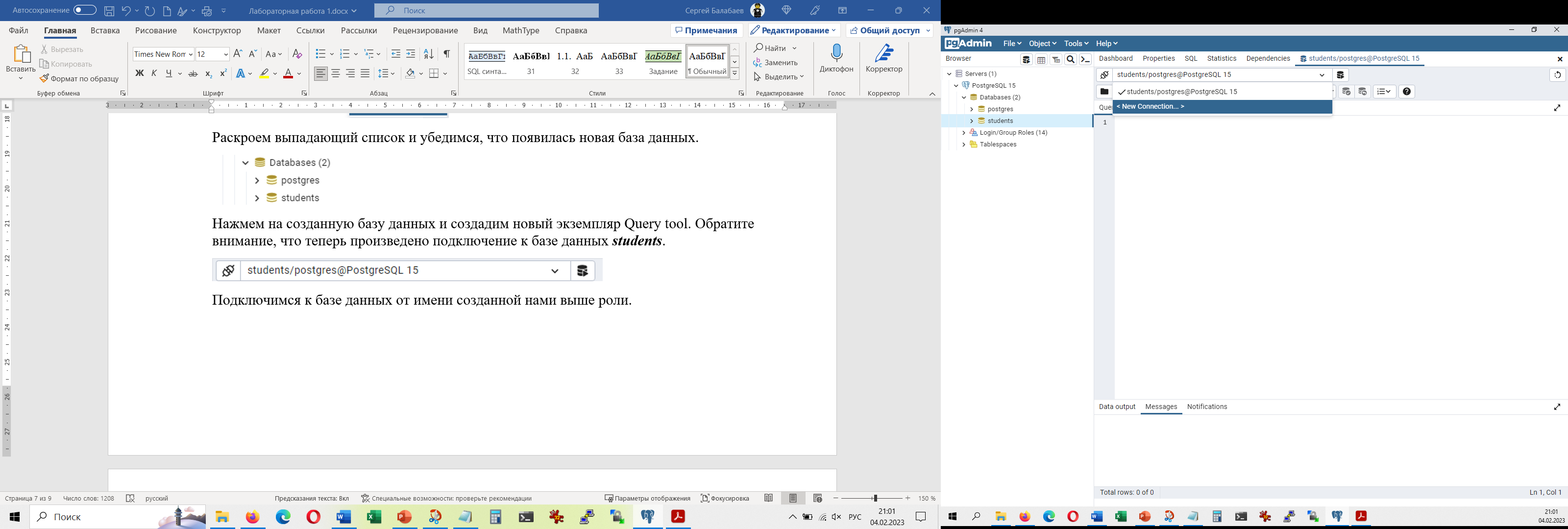
Автоматически созданное описание

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



* + 1. *Смена роли пользователя*

Для подключения к базе данных от имени другой роли откройте новое окно Query tool, нажмите на строку с соединением и в выпавшем окне выберете строку <New Connection>



Создадим новое соединение

Изображение выглядит как текст

Автоматически созданное описание

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



Обратите внимание, что от имени выбранного пользователя будут выполняться только команды, запущенные в открытом окне. В других окнах по умолчанию будет использоваться роль, под которой было осуществлено подключение к серверу баз данных. В нашем случае это роль *postgres*.

* + 1. *Работа с таблицами в базе данных*

Рассмотрим простейшие действия с таблицами. В выпадающем списке найдем созданные таблицы по пути: «Servers – PostgreSQL 15 – students – Schemas – public – Tables».

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

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.
  + All rows – выводит все строки таблицы
  + Filtered rows – выводит стрки, удовлетворяющие определенному условию. Например, для поиска всех студентов, чье имя – Иван – введите следующее условие: first\_name='Иван'

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

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

* + 1. *Работа с ERD моделью базы данных*

Для графического представления схемы базы данных используется диаграмма «Сущность-связь» - ER модель. Чтобы её открыть в программе pgAdmin необходимо в левом столбце правой кнопкой мыши нажать на название базы данных и выбрать пункт ERD for Database.

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

1. Практическая часть

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

* 1. Задание 1.

Создание базы данных

* + 1. Создайте учебную базу данных *Students.* Для этого необходимо войти в учетную запись *postgres* и подключиться к программе *psql*.
    2. Выйдите из программы *psql* и заполните базу данных, используя файл резервной копии.
    3. Используя программу *pgAdmin*, ознакомьтесь со схемой данных, содержимым таблиц БД. Определите число строк в каждой из таблиц.
    4. Определите**,** какие таблицы в базе данных *Students* являются главными, а какие для них подчиненными.
  1. Задание 2.

Администрирование СУБД

* + 1. Подключитесь к созданной базе данных *Students* из-под командной строки. Определите, какой размер на диске занимает таблица *student*?
    2. Создайте новую роль «Ваши инициалы junior». Выделите ей привилегии на вход и установите пароль «654321». Подключитесь от её имени к базе данных *students* и попробуйте удалить её с помощью запроса:

DROP DATABASE students;

Удалось ли вам это сделать?

* 1. Задание 3.

Редактирование содержимого базы данных

* + 1. Выполните в соответствии с вариантом задание (см. таблицу ниже) на изменение содержимого базы данных.
    2. После внесенных изменений, создайте новую резервную копию базы данных *Students*.

|  |  |
| --- | --- |
| № варианта | Задание |
| 1 | Студентка группы ИВТ-32 Барашкова Настасья Филлиповна пересдала экзамен по Основы теории информации и кодирования на 4. Исправьте её оценку. |
| 2 | В связи с ошибкой при заполнении документов, студенту Безухову Пьеру Кирилловичу из группы ИТД-12 указали неверно дату рождения. Исправьте её на 12 августа 2004 года. |
| 3 | Студентка группы ИТД-32 Ольга Ильинская вышла замуж за своего одногруппника, родившегося 6 февраля, и сменила свою фамилию на его. Выполните соответствующее изменение в базе данных. |
| 4 | Из-за конфликтов с одногруппниками, Роман Хлудов 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 Михаил Астров на 3. |
| 8 | Почта студента, родившегося в 1999 году, изменилась на oldeststudent@miet.ru. Произведите данное изменение. |
| 9 | После провала на второй пересдаче по философии студент Андрей Алехин был отчислен. Удалите его из базы данных |
| 10 | У студентки Анны Незвановой был обнаружен пропавший после её рождения отец. Добавьте ей отчество Александровна. |

Контрольные вопросы

1. Что такое база данных?
2. Что такое СУБД?
3. Чем потенциальный ключ отличается от первичного ключа? Когда используются внешний и суррогатный ключи.
4. Опишите работу метода аутентификации *ident.* В чем различие между аутентификацией и авторизацией?
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, p. 662. |
| [4] | Б. А. Новиков, Е. А. Горшкова и Н. Г. Графеева, Основы технологии баз данных, 2-е ред., Москва: ДМК пресс, 2020, p. 582. |
| [5] | Е. П. Моргунов, PostgreSQL. Основы языка SQL, 1-е ред., Санкт-Петербург: БХВ-Петербург, 2018, p. 336. |