МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

«Вятский государственный университет»

(ФГБОУ ВПО «ВятГУ»)

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Методические указания

к самостоятельным и лабораторным работам

по дисциплине «Базы данных»

«Основы DDL-запросов в PostgreSQL»

Киров 2019

Основы DDL-запросов в PostgreSQL

Цели лабораторной работы:

- Познакомится со схемами, пользователями и ролями в PostgreSQL;
- Познакомиться с типами данных в PostgreSQL;
- Освоить основные варианты DDl-запросов в PostgreSQL;
- Закрепить знания по проектированию структуры реляционной БД;
- Создать рабочий материал для следующих лабораторных работ.

Задание на лабораторную работу:

Нужно выполнить следующие шаги:

1. Придумать структуру базы данных на любую выбранную тему. Структура должна отвечать следующим условиям (это важно для следующих лабораторных работ):
   - должно быть не меньше пяти таблиц;
   - хотя бы одна таблица должна содержать колонку с числовыми данными;
   - структура БД должна быть в третьей нормальной форме.
2. Создать нового пользователя и пустую БД. Подключиться к созданной БД.
3. Написать и выполнить SQL-скрипт, создающий таблицы согласно разработанной структуре БД. Созданный в п.2 пользователь должен иметь все права на созданные объекты. В этом же скрипте должны создаваться нужные ограничения и индексы:
   - обязательно должны быть созданы внешние ключи для поддержания ссылочной целостности;
   - желательно должны быть проставлены ограничения и уникальные индексы для поддержания консистентности данных (например, если в таблице есть столбец с температурой воздуха за окном, стоит ограничить возможный промежуток значений);
   - желательно должны быть проставлены индексы для производительности там, где они могут помочь (из расчета, что созданная БД будет хранить достаточно большое количество данных).

Отчет по лабораторной работе должен содержать:

- ER-диаграмму для разработанной структуры БД;
- описание выбранной темы и краткое описание таблиц;
- SQL-скрипт из п.3 задания с комментариями.

Выполнение лабораторной работы:

I. Придумать структуру базы данных на любую выбранную тему.

Создадим базу данных по теме «Студенты». База данных будет содержать основные данные о студенте и его успеваемости по различным предметам.

Каждый студент в рамках базы данных имеет ФИО, адрес проживания, номер курса и учебную группу.

Также для каждого студента может иметься информация о его успеваемости по тому или иному предмету, включающая название предмета, оценку и длительность предмета в академических часах.

База данных также должна содержать информацию о преподавателях, которая включает в себя: ФИО, должность, номер телефона.

Помимо этого, база данных должна описывать какой преподаватель проводил занятия и какой тип занятия он вел (лекция, практика, лабораторная работа или семинар). При этом один преподаватель может вести различные предметы и их типы. Различные типы занятий по одному предмету также могут проводить различные преподаватели.

Для реализации данной базы данных необходимо 6 таблиц: «Студент», «Группа», «Успеваемость», «Предмет», «Преподаватель», и промежуточная таблица «Тип проводимого занятия».

Таблица «Студент» содержит столбцы id, имя, id группы, адрес, курс.

Таблица «Группа» содержит столбцы id, название, количество студентов в группе.

Таблица «Успеваемость» содержит столбцы id студента, id предмета, оценку.

Таблица «Предмет» содержит столбцы id, название предмета, количество часов.

Таблица «Преподаватель» содержит столбцы id, имя, должность, номер телефона.

Таблица «Тип проводимого занятия» содержит столбы id предмета, id преподавателя, тип занятия. Данная таблица введена для того, чтобы избежать связи «многие ко многим» между таблицами «Предмет» и «Преподаватель» и привести базу данных в 3 нормальную форму. Подробнее о нормальных формах можно прочитать здесь: (https://habr.com/ru/post/254773/).

II. Создать нового пользователя и пустую БД. Подключиться к созданной БД.

Создание БД

Любой сервер PostgreSQL может обслуживать несколько БД сразу. По умолчанию после установки он содержит только одну базу данных – postgres, которая содержит системную информацию.

Подробнее о базовых командах создания и управления настройками БД в целом можно прочитать в документации (стр. 1319; https://postgrespro.ru/docs/postgresql/9.6/sql-createdatabase.html).

Ниже показаны примеры команд для создания и удаления БД.

```
1.-- Создать БД lab
2.create database lab;
3.-- Две команды ниже эквиваленты и создают БД lab, назначая владельцем
   пользователя postgres (если postgres должен быть владельцем, а команда
   выполняется из-под другого пользователя)
4.create database lab owner postgres;
5.create database lab with owner postgres;
6.-- Удалить БД lab и все её данные
7.drop database lab;
```

Для создания БД необходимо открыть pgAdmin3, выбрать базу данных postgres, а затем выбрать пункт PSQL Console, как это сделано на рисунке 2.

Затем в консоли необходимо написать команду для создания БД, примеры которых описаны ранее, как это сделано на рисунке 3.

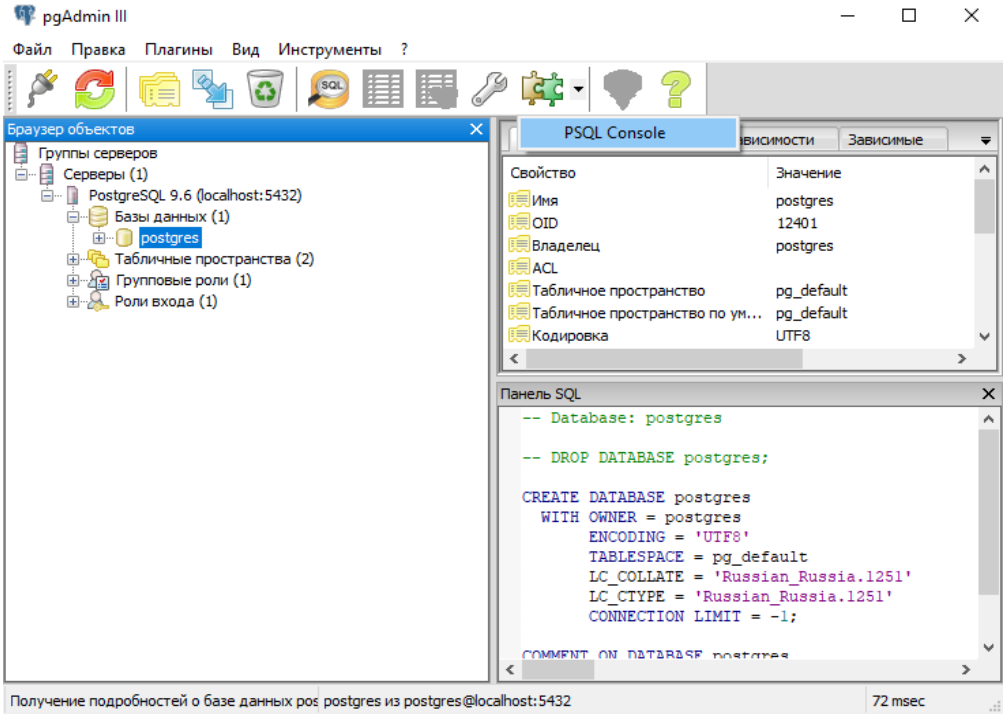

Рисунок 1 - Выбор PSQL Console

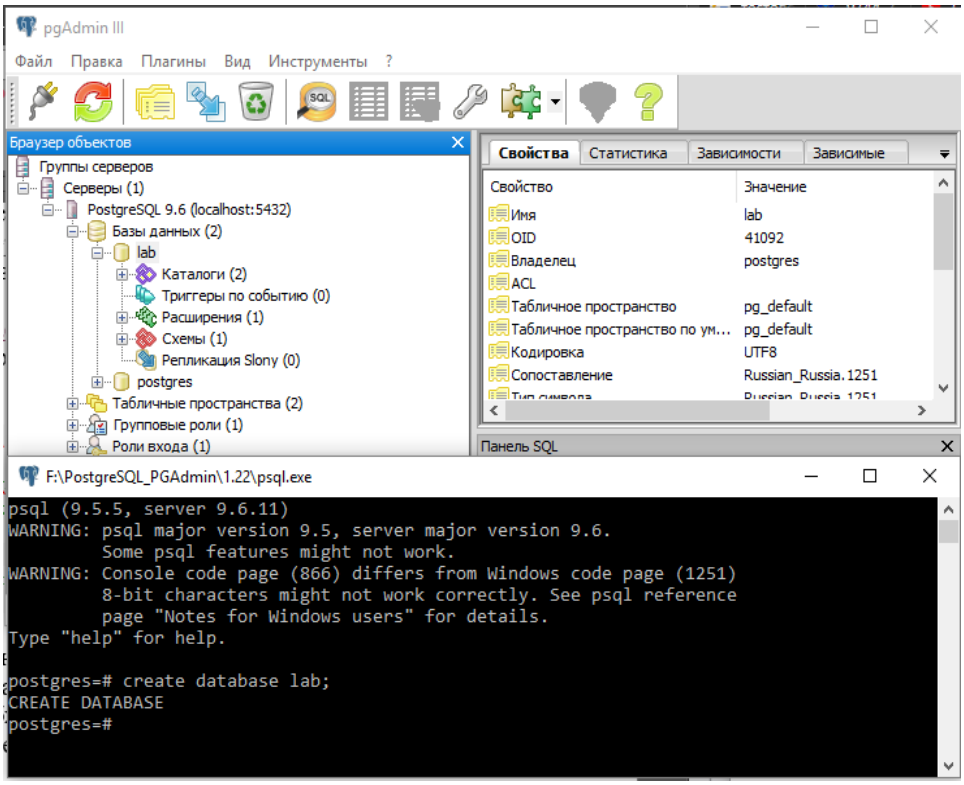

Рисунок 2 – Создание БД

Также базу данных можно создать при помощи встроенных возможностей pgAdmin3. Для этого необходимо подключится к серверу, перейти внутри сервера к базам данных и выбрать пункт «Новая база данных…», как это показано на рисунке 4.

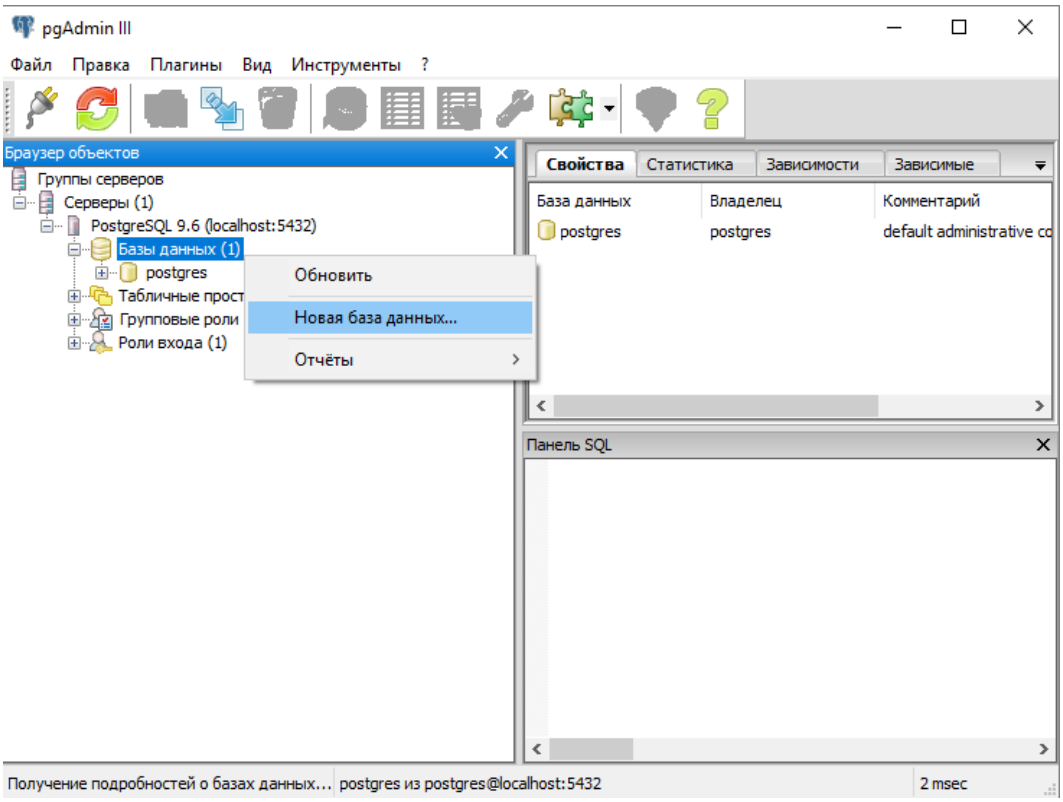

Рисунок 3 – Создание базы данных

Далее необходимо указать владельца базы данных и ее имя, как это показано на рисунке 5.

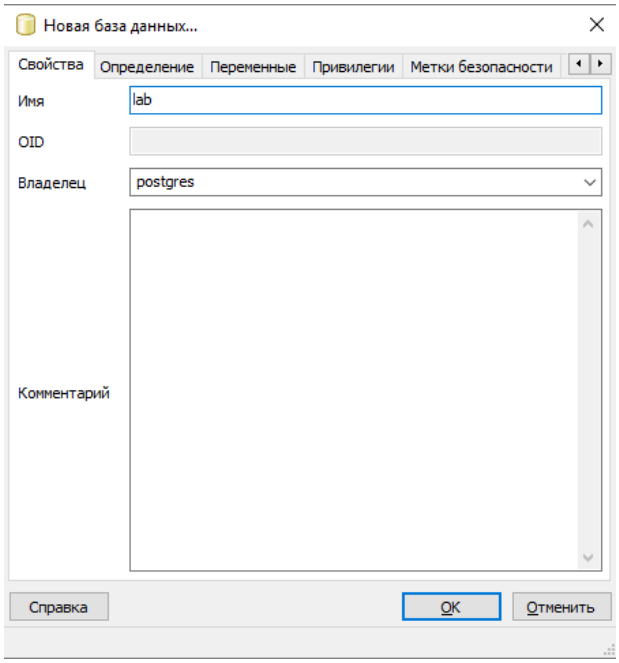

Рисунок 4 – Создание базы данных

Схемы

В PostgreSQL база данных состоит из именованных схем. Схемы, в свою очередь, содержат таблицы, функции, типы данных и другие объекты.

Чтобы обращаться к объектам внутри схемы, необходимо указывать их полное имя, состоящее из имени схемы и имени объекта, разделённых точкой:

```
1.stud.table1 -- Схема stud, таблица table1
```

По умолчанию каждая БД содержит только одну схему – public. Если не указывать имя схемы перед именем объекта, автоматически будет подставляться public.

```
1.-- Следующие два запроса эквивалентны
2.select * from public.table1;
3.select * from table1;
```

После создания схемы все права на неё имеет только владелец. Всем остальным пользователям необходимо выдать соответствующие права, если им нужен доступ к объектам внутри схемы.

По умолчанию все пользователи имеют полные права на схему public.

Подробнее о схемах можно прочитать в документации (гл. 5.8, стр. 62; https://postgrespro.ru/docs/postgresql/9.6/ddl-schemas).

Ниже показаны примеры различных команд, связанных с созданием схем в PostgreSQL.

```
-- Создать схему stud
create schema stud;
-- Создать схему stud и назначить пользователя postgres владельцем (если postgres должен быть владельцем, а команда выполняется из-под другого пользователя)
create schema stud authorization postgres;
-- Удалить пустую схему stud (если в ней есть какие-либо объекты, поднимется ошибка)
drop schema stud;
-- Удалить схему stud и все объекты внутри неё
drop schema stud cascade;
```

Создание схем производится аналогично созданию базы данных, описанному выше.

Создание пользователя

Роль можно рассматривать как пользователя базы данных или как группу пользователей, в зависимости от того, как роль настроена. Роли могут владеть объектами базы данных (например, таблицами и функциями) и выдавать другим ролям разрешения на доступ к этим объектам.

Подробнее о ролях можно прочитать в документации (гл. 21, стр. 549; https://postgrespro.ru/docs/postgresql/9.6/user-manag).

Ниже показаны примеры различных команд, связанных с созданием и удалением ролей.

```
-- Две команды ниже аналогичны
-- В результате любой из них будет создан пользователь first_user с паролем first_user
create user first_user password 'first_user';
create user first_user password 'first_user' login;
-- Две команды ниже аналогичны
-- Удалить созданную роль (пользователя)
drop user first_user;
drop role first_user;
-- Две команды ниже аналогичны
-- Сделать роль (пользователя) суперюзером (имеет все права; фактически, проверка прав суперпользователя выполняется только во время логина)
alter role first_user superuser;
alter user first_user superuser;
-- Две команды ниже аналогичны
-- Убрать у пользователя lab_test роль суперюзера
alter role first_user nosuperuser;
alter user first_user nosuperuser;
```

На рисунке 6 показан пример создания пользователя при помощи PSQL Console и результат выполнения команды.

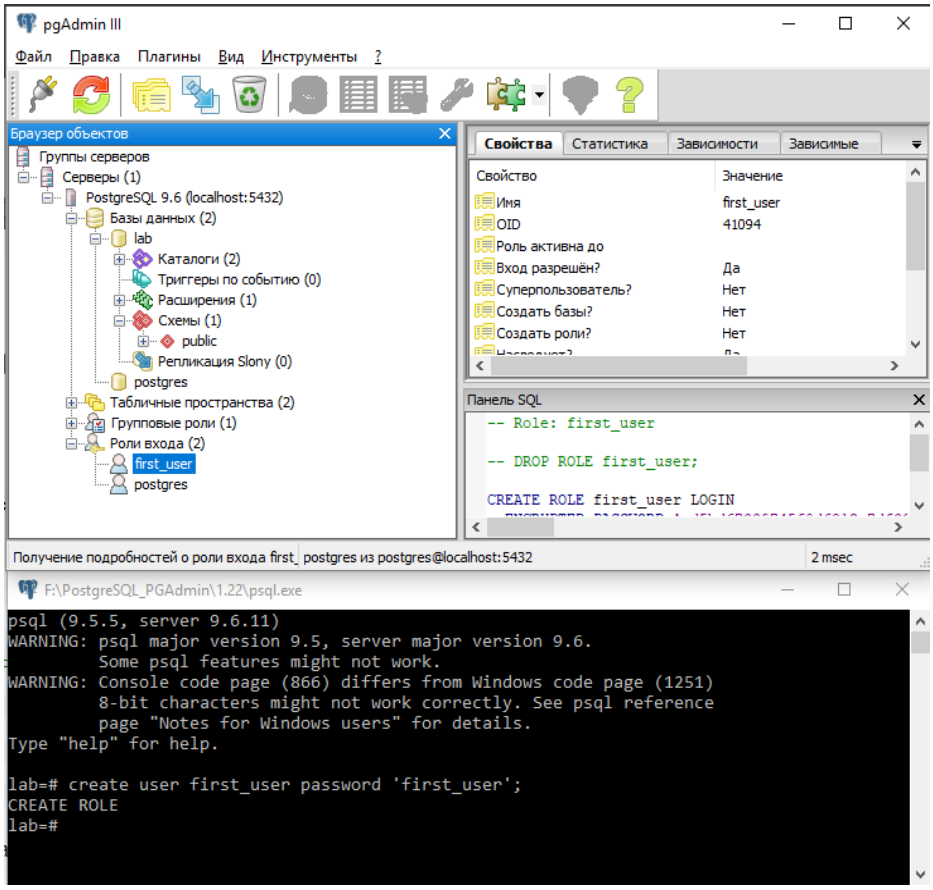

Рисунок 5 – Создание нового пользователя

III. Написать и выполнить SQL-скрипт, создающий таблицы согласно разработанной структуре БД. Созданный в п.2 пользователь должен иметь все права на созданные объекты. В этом же скрипте должны создаваться нужные ограничения и индексы.

Основные типы данных

В таблице 1 перечислены числовые типы данных общего назначения.

Таблица 1 – Числовые типы данных общего назначения

Имя	Псевдонимы	Описание
bigint	int8	знаковое целое из 8 байт
bigserial	serial8	восьмибайтное целое с автоувеличением
double precision	float8	число двойной точности с плавающей точкой (8 байт)

integer	int, int4	знаковое четырёхбайтное целое
numeric [(p, s)]	decimal [(p, s)]	вещественное число заданной точности
real	float4	число одинарной точности с плавающей точкой (4 байта)
smallint	int2	знаковое двухбайтное целое
smallserial	serial2	двухбайтное целое с автоувеличением
serial	serial4	четырёхбайтное целое с автоувеличением

В таблице 2 показаны символьные типы данных общего назначения.

Таблица 2.2 – Символьные типы данных общего назначения

Имя	Псевдонимы	Описание
character varying(n)	varchar(n)	строка ограниченной переменной длины
character(n)	char(n)	строка фиксированной длины, дополненная пробелами
text		строка неограниченной переменной длины

В таблице 3 показаны остальные типы данных общего назначения.

Таблица 3 – Остальные типы данных общего назначения

Имя	Псевдонимы	Описание
boolean	bool	логическое значение (true/false)
bytea		двоичные данные («массив байт»)

Для создаваемой в данном примере БД необходимы перечисления. Типы перечислений (enum) определяют статический упорядоченный набор значений, так же, как и типы enum, существующие в ряде языков программирования.

В нашем случае они необходимы для описания типа занятия в таблице «Тип проводимого занятия» и для описания должности в таблице «Преподаватель». Ниже показан пример создания данных перечислений.

```
--Создание типа перечисления для обозначения типа проводимого занятия
create type public.occupation_type_enum as enum (
  'lecture', 'laboratory work', 'practical lesson', 'seminar');
--Создание типа перечисления для обозначения должности преподавателя
create type public.position_type_enum as enum (
  'docent', 'professors', 'senior lecturer', 'instructor');
```

Подробнее о всех существующих типах данных можно прочитать в документации (гл.8, стр. 105; https://postgrespro.ru/docs/postgresql/9.6/datatype.html).

Создание перечислений выполняется аналогично созданию таблиц (см. пункт «Создание таблиц»).

Создание таблиц

В качестве примера создадим таблицы «Группа», «Студент» и «Преподаватель».

Ниже приведен пример создания таблицы «Группа».

Таблицы создаются командой create table. Подробнее о полном синтаксисе команды create table можно прочитать в документации (стр. 1379; https://postgrespro.ru/docs/postgresql/9.6/sql-createtable.html).

```
--Создание таблицы Группа
create table public.group(
  id bigserial primary key,
  name varchar(30) not null unique,
  number_of_students int not null check (number_of_students >= 1)
);
```

В качестве первичного ключа используется столбец id. Он имеет тип bigserial, что означает, что каждая новая строка будет получать новое уникальное значение id с помощью автоинкремента предыдущего выданного. Указание primary key после описания столбца показывает, что:

- столбец является первичным ключом;
- столбец не может содержать null;
- значение в столбце уникально.

Если в описании столбца указано not null, то он не может содержать null.

Для описания столбца name также используется ограничение уникальности unique. Ограничения уникальности гарантируют, что данные

в определённом столбце или группе столбцов уникальны среди всех строк таблицы. При создании такого ограничения автоматически создаётся индекс. Подробнее про индексы можно прочитать в документации (гл. 11, стр. 239; https://postgrespro.ru/docs/postgresql/9.6/indexes.html).

Для описания столбца number_of_students используется ограничение-проверка check. В его определении указывается, что значение столбца должно удовлетворять определённому условию. Например, количество студентов в группе должно быть более или равно одному, как это сделано в примере выше.

Подробнее про ограничения можно прочитать в документации (гл. 5.3, стр. 47; https://postgrespro.ru/docs/postgresql/9.6/ddl-constraints).

Далее приведен пример создания таблицы «Студент».

```
--Создание таблицы Студент
create table public.student(
  id bigserial primary key,
  first_name varchar(30) not null,
  second_name varchar(30) not null,
  middle_name varchar(30),
  id_group bigint not null references public.group(id),
  address varchar(100),
  cource int check (cource >= 1 and cource <=5)
);
```

Здесь используются описанные выше ключевые слова, но появляется новая команда references. Это ключевое слово означает, что столбец id_group является внешним ключом и ссылается на таблицу «Группа».

Ограничение внешнего ключа указывает, что значения столбца (или группы столбцов) должны соответствовать значениям в некоторой строке другой таблицы. Это называется ссылочной целостностью связанных таблиц.

Далее приведен пример создания таблицы «Преподаватель».

```
--Создание таблицы Преподаватель
create table public.teacher(
  id bigserial primary key,
  first_name varchar(30) not null,
  second_name varchar(30) not null,
  middle_name varchar(30),
  position public.position_type_enum not null default 'instructor',
  phone_number varchar(30)
);
```

В данном примере стоит обратить внимание на столбец position, обозначающий должность преподавателя. Этот столбец имеет тип перечисления, которое создано ранее и ключевое слово default. Это означает, что столбец содержит значение по умолчанию. Position из примера в новой строке будет заполнен значением ‘instructor’, если в запросе на добавление данных не будет указано иное.

Все прочие таблицы создаются по аналогии с приведенными примерами.

Для того, чтобы создать описанные выше таблицы необходимо выбрать в pgAdmin3 нужную БД, а затем выбрать пункт «Выполнить пользовательские SQL-запросы», как это показано на рисунке 7.

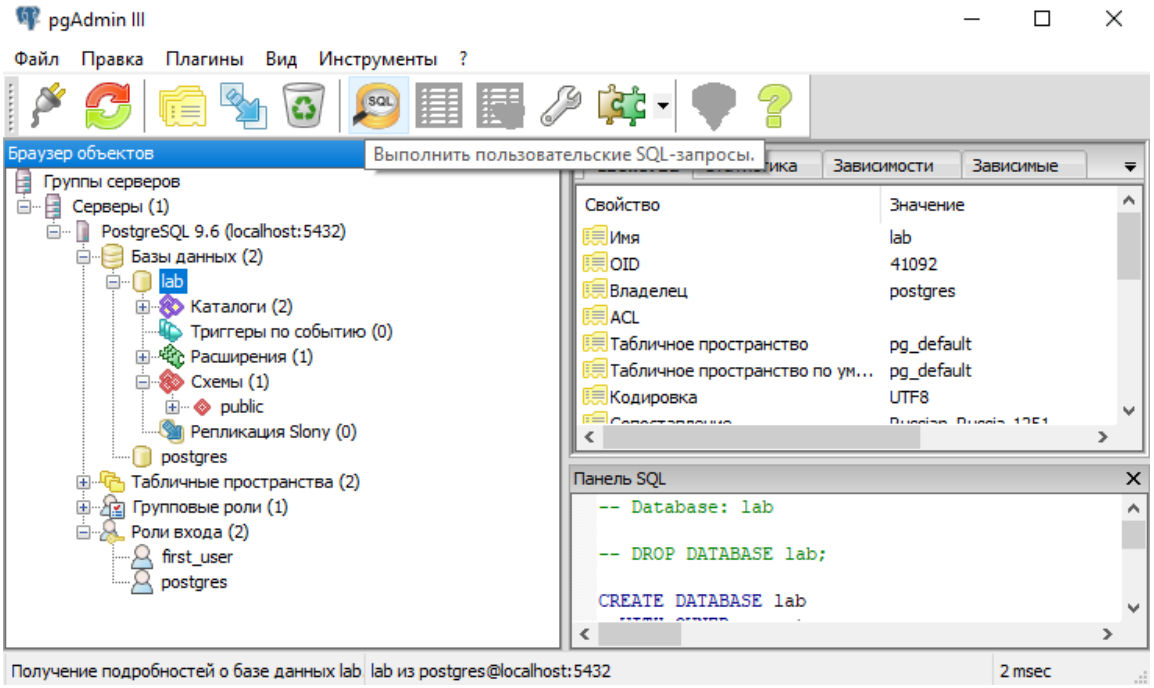

Рисунок 6 – Выполнение пользовательских SQL-запросов

Затем необходимо в открывшемся окне написать составленные запросы в поле «Редактор SQL» и нажать клавишу «Выполнить запрос». Как видно из рисунка 8 в БД были созданы вышеописанные таблицы.

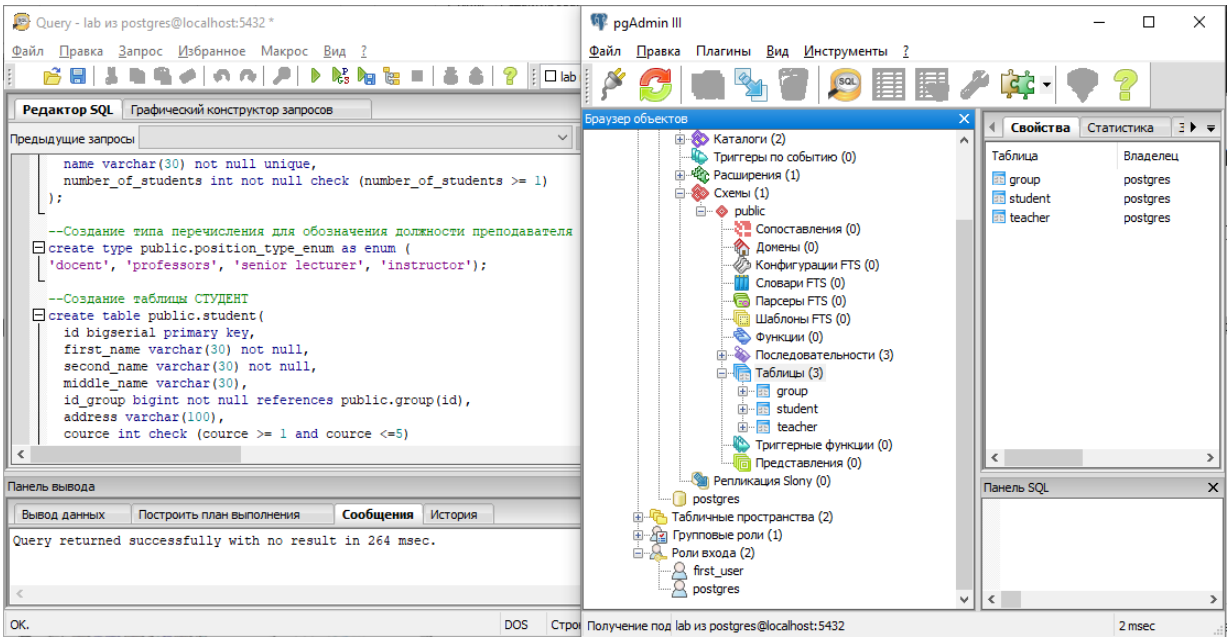

Рисунок 7 – Созданные таблицы

Следует отметить, что созданные таблицы можно изменять и удалять. Для изменения определения существующей таблицы используется команда alter table. Она позволяет добавлять и удалять колонки, переименовывать их, редактировать ограничения, триггеры, индексы и т.д.

Подробнее о полном синтаксисе команды alter table можно прочитать в документации (стр. 1253; https://postgrespro.ru/docs/postgresql/9.6/sql-altertable.html).

Удалить созданную таблицу можно с помощью команды drop.

Выдача прав пользователю

Для того, чтобы управлять привилегиями роли, используются команды grant и revoke. С их помощью можно разрешать или запрещать:

- подключение к БД;
- выполнение чтения, записи, изменения, удаления данных;
- создание различных объектов в БД (таблиц, триггеров, внешних ключей, функций на определенном языке и т.д.);
- выполнение определенных функций (включая агрегатные).

Ниже приведен пример, демонстрирующий выдачу созданному ранее пользователю first_user всех прав на созданные объекты.

```
1.--Выдача прав пользователю FIRST_USER
2.grant all on schema public to first_user;
```

Подробнее о привилегиях, управление которыми доступно, можно прочитать в документации (стр. 1485; https://postgrespro.ru/docs/postgresql/9.6/sql-grant).

Примечания:

- Все созданные запросы можно выполнять в сторонних программах, например, в DataGrip.
- ER-диаграмму можно сгенерировать автоматически, используя, например, средства DataGrip или DbVisualizer.
- Важно помнить, что максимальная длина имени объекта в PostgreSQL (без имени схемы и т.д.) – 63 символа;
- Важно помнить, что имена регистронезависимые;
- Самый лучший тип для столбца с первичным ключом – bigserial; индекс по числовому столбцу работает быстрее всего, автоинкремент помогает поддерживать уникальность, а на размер столбца под первичный ключ лучше не скупиться – если значения smallserial или serial закончатся, это вызовет проблемы;
- Соответственно, тип для столбцов для внешних ключей – bigint;