МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

РУТ (МИИТ)

ИНСТИТУТ УПРАВЛЕНИЯ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

Кафедра «Цифровые технологии управления транспортными процессами»

Технологии хранения данных

Лабораторная работа 4 и 5

Выполнил студент:

Солнцев К.В.

Группа – УВПв-321,3 курс

Преподаватель:

Новиков А.И.

Москва

2023

Содержание

[Задание 3](#_Toc151469600)

[Выполнение 3](#_Toc151469601)

[Создание БД 3](#_Toc151469602)

[Создание реляционных таблиц 4](#_Toc151469603)

[Импорт данных из Excel в БД 9](#_Toc151469604)

[Создание резервной копии БД 11](#_Toc151469605)

[SQL запросы к БД 13](#_Toc151469606)

[SQL запрос на изменение данных 14](#_Toc151469607)

[SQL запрос на удаление данных 15](#_Toc151469608)

[SQL запрос на поиск данных 16](#_Toc151469609)

[SQL выборка из нескольких таблиц 17](#_Toc151469610)

[Агрегации в SQL 17](#_Toc151469611)

[Вывод 19](#_Toc151469612)

# Задание

Разработать реляционную модель базы данных в соответствии с ER-моделью базы данных, разработанной ранее. Разработку вести в программе PostgreSQL в среде pgAdmin4. Создавать таблицы и прописывать связи рекомендуется на языке SQL, а не через интерфейс (кнопки создания и т.д.) pgAdmin4.

Загрузить данные в таблицы базы данных с помощью INSERT-запросов на языке SQL или с помощью IMPORT/EXPORT команды pgAdmin4. Вывести пример успешно загруженной таблицы.

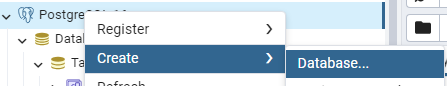
Написать SQL запросы к нашей БД на:

* Редактирование данных
* Удаление данных
* Поиск информации
  + С подзапросами
  + Без соединения таблиц
  + С соединением таблиц
* Использование функций агрегирования

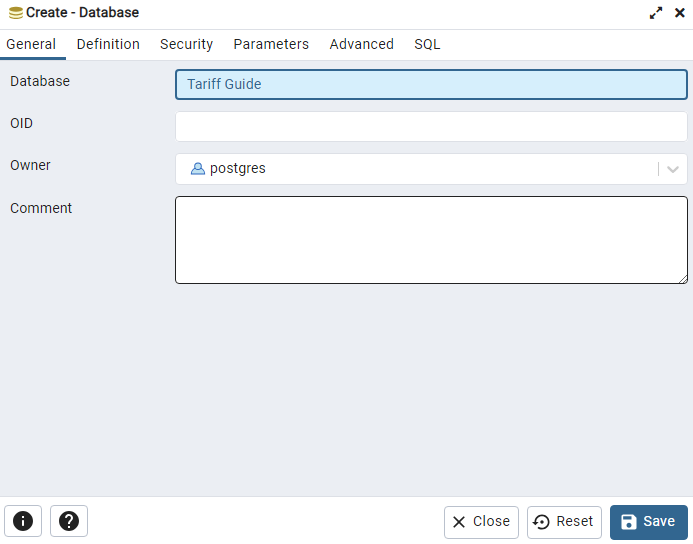
# Выполнение

## Создание БД

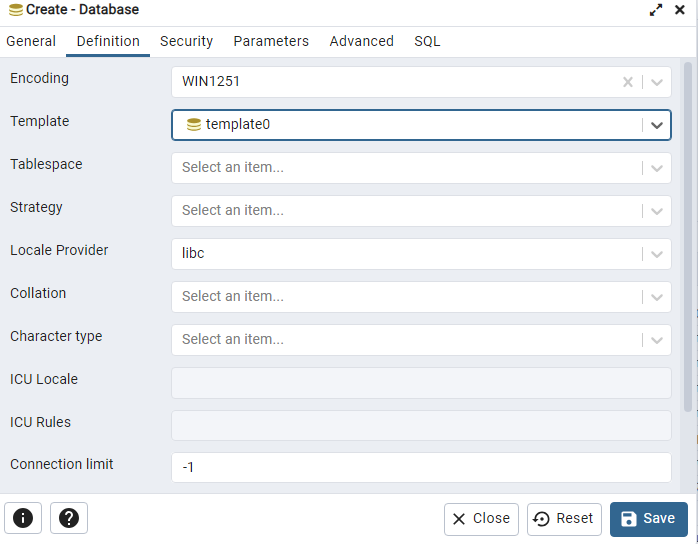
Создадим новую базу данных на сервере:



Пропишем для неё название и владельца (администратора):



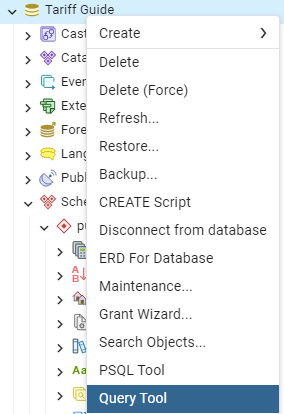
Изменим кодировку таблицы так, чтобы можно было работать с данными, названиями таблиц и названиями столбцов на кириллице:



Жмём сохранить и начинаем работать с созданной БД.

## Создание реляционных таблиц

Теперь перейдем непосредственно к созданию таблиц и связей между ними. Открываем Query Tool для написания SQL запросов к БД:



Вводим туда следующий код, в котором прописаны все таблицы, параметры их полей, основные и внешние ключи, параметры обновления данных и т.д. (можно создавать таблицы как одновременно, так и поодиночке, но при создании таблиц по одной необходимо помнить, что сначала добавляются материнские таблицы, а потом уже их дочерние, иначе возникнет ошибка и таблица не создастся):

CREATE TABLE "Дорога" (

"Код дороги" INTEGER NOT NULL,

"Название дороги" VARCHAR (50) NOT NULL UNIQUE,

"Адрес" VARCHAR (100) NOT NULL,

CONSTRAINT дорога\_pk PRIMARY KEY ("Код дороги")

);

CREATE TABLE "Участок" (

"id участка" INTEGER NOT NULL CHECK("id участка" >= 0),

"Название участка" VARCHAR (50) NOT NULL UNIQUE,

"Название дороги" VARCHAR (50) NOT NULL,

CONSTRAINT участок\_pk PRIMARY KEY ("id участка"),

CONSTRAINT дорога\_участок\_fk FOREIGN KEY ("Название дороги") REFERENCES "Дорога" ("Название дороги") ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE "ТранзитныйПункт" (

"№ ТП" INTEGER NOT NULL CHECK("№ ТП" >= 0),

"Название ТП" VARCHAR (50) NOT NULL UNIQUE,

"Название дороги" VARCHAR (50) NOT NULL,

CONSTRAINT транзитныйпункт\_pk PRIMARY KEY ("№ ТП"),

CONSTRAINT дорога\_транзитныйпункт\_fk FOREIGN KEY ("Название дороги") REFERENCES "Дорога" ("Название дороги") ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE "Станция" (

"Код станции" INTEGER NOT NULL CHECK("Код станции" >= 0),

"Название станции" VARCHAR (50) NOT NULL UNIQUE,

"Название дороги" VARCHAR (50) NOT NULL,

"Параграф" VARCHAR (30) NOT NULL,

"Страница Тр4т1" VARCHAR (20) NOT NULL,

"Строка Тр4т1" VARCHAR (20) NOT NULL,

CONSTRAINT станция\_pk PRIMARY KEY ("Код станции"),

CONSTRAINT дорога\_станция\_fk FOREIGN KEY ("Название дороги") REFERENCES "Дорога" ("Название дороги") ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE "СтанцияУчасток" (

"id станции-участка" INTEGER NOT NULL CHECK("id станции-участка" >= 0),

"Код станции" INTEGER NOT NULL,

"Название станции" VARCHAR (50) NOT NULL,

"id участка" INTEGER NOT NULL,

"Название участка" VARCHAR (50) NOT NULL,

CONSTRAINT станцияучасток\_pk PRIMARY KEY ("id станции-участка"),

CONSTRAINT участок\_станцияучасток\_fk FOREIGN KEY ("id участка") REFERENCES "Участок" ("id участка") ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT станция\_станцияучасток\_fk FOREIGN KEY ("Код станции") REFERENCES "Станция" ("Код станции") ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT участок\_станцияучасток\_fk\_1 FOREIGN KEY ("Название участка") REFERENCES "Участок" ("Название участка") ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT станция\_станцияучасток\_fk\_1 FOREIGN KEY ("Название станции") REFERENCES "Станция" ("Название станции") ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE "РасстояниеТпТп" (

"id ТП-ТП" INTEGER NOT NULL,

"№ ТП от" INTEGER NOT NULL,

"№ ТП до" INTEGER NOT NULL,

"Расстояние" INTEGER NOT NULL,

CONSTRAINT расстояниетптп\_pk PRIMARY KEY ("id ТП-ТП"),

CONSTRAINT транзитный\_расстояние\_fk FOREIGN KEY ("№ ТП от") REFERENCES "ТранзитныйПункт" ("№ ТП") ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT транзитный\_расстояние\_fk\_1 FOREIGN KEY ("№ ТП до") REFERENCES "ТранзитныйПункт" ("№ ТП") ON UPDATE CASCADE ON DELETE CASCADE

);

CREATE TABLE "РасстояниеСтТп" (

"id СТ-ТП" INTEGER NOT NULL,

"Код станции" INTEGER NOT NULL,

"№ ТП" INTEGER NOT NULL,

"Расстояние" INTEGER NOT NULL,

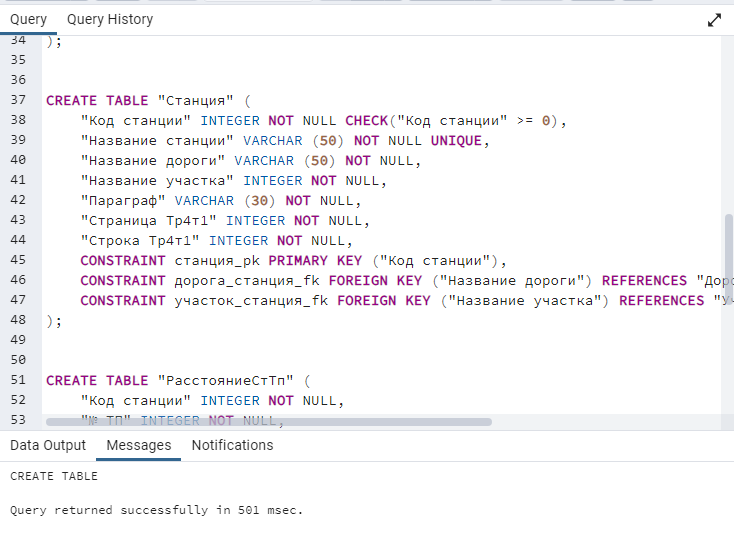
CONSTRAINT расстояниесттп\_pk PRIMARY KEY ("id СТ-ТП"),

CONSTRAINT станция\_расстояниесттп\_fk FOREIGN KEY ("Код станции") REFERENCES "Станция" ("Код станции") ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT транзитный\_расстояние\_fk\_2 FOREIGN KEY ("№ ТП") REFERENCES "ТранзитныйПункт" ("№ ТП") ON UPDATE CASCADE ON DELETE CASCADE

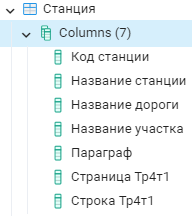
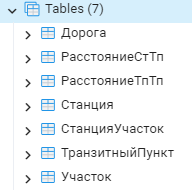
);

После запуска кода в программе это будет выглядеть примерно так:

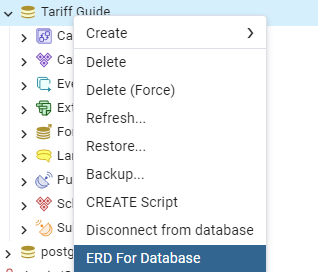


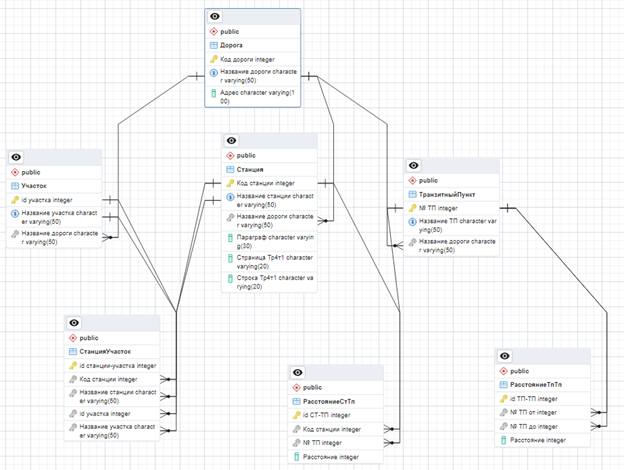
Уведомление об успешном завершении говорит, что все связи и таблицы соединены верно.

Далее откроем в левом меню Tariff Guide/Schemas/Tables и убедимся, что все необходимые таблицы и колонки в них создались правильно:



Выведем итоговую схему нашей БД со всеми отношениями между таблицами. Для этого воспользуемся опцией pgAdmin4 ERD For Database:



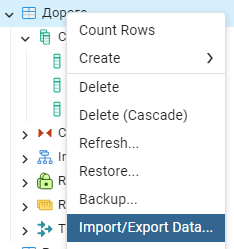


Исходник схемы в полном размере приложим в папку с отчетом и загрузим в github.

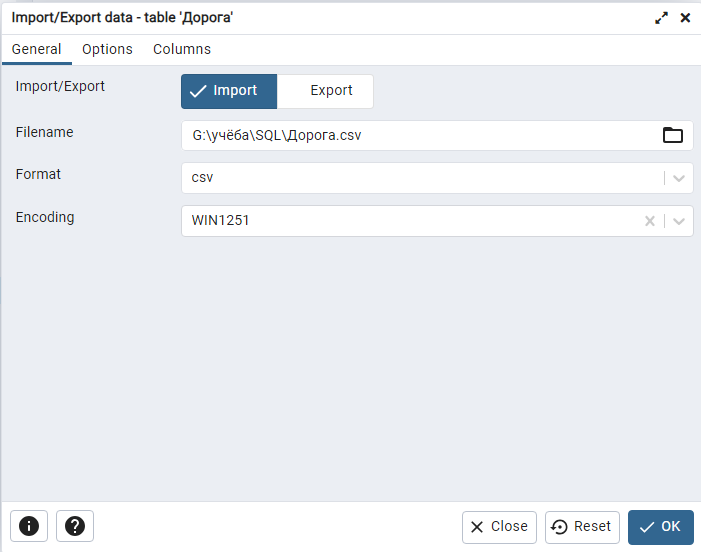
Теперь перейдем к заполнению таблиц данными.

## Импорт данных из Excel в БД

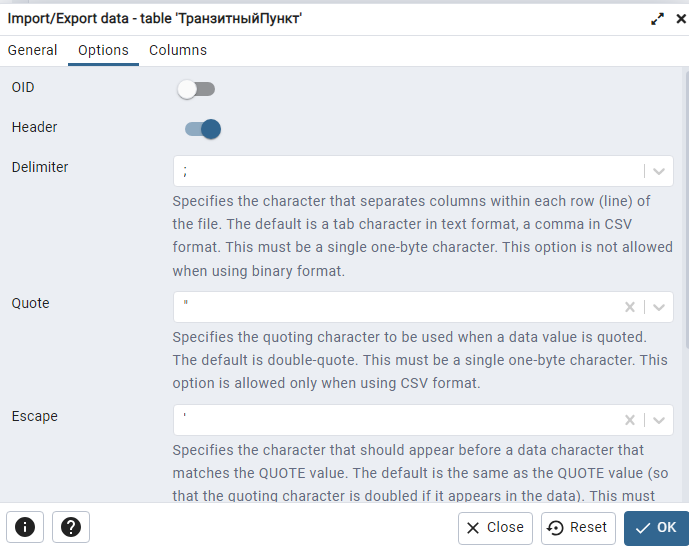
Для того, чтобы перенести данные из Excel в БД – сохраним таблицы в формате .csv (разделители запятые) и воспользуемся встроенной функцией Import/Export Data pgAdmin4, которая аналогична SQL запросам Import и Export:



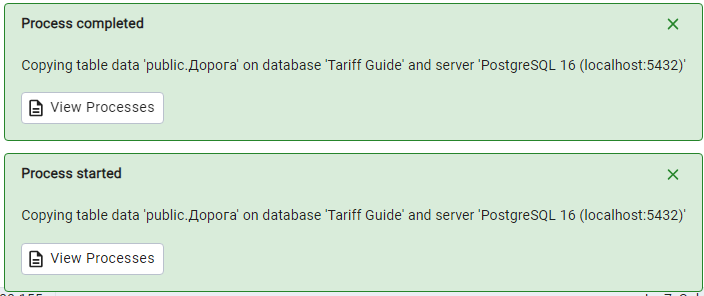
Укажем путь к .csv файлу и необходимую кодировку:



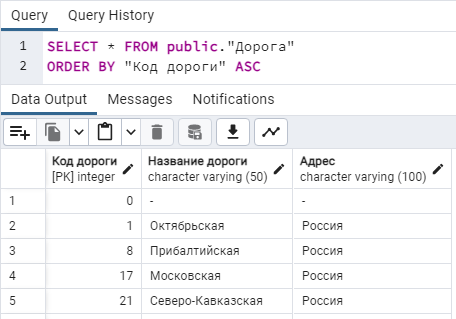
Добавим наличие заголовка, изменим разделитель на “;”, т.к. в русской локализации используется этот разделитель, и начинаем импорт:



Если не допущено ошибок и таблица в БД и .csv одинаковы, то справа увидим не только всплывающее уведомление о начале операции, но и о её успешном завершении:



После можем вывести таблицу, чтобы убедиться, что все данные подгружены верно:

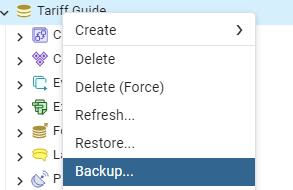


Проделываем подобные операции со всеми имеющимися таблицами. Не забываем, что сначала загружаются материнские таблицы и только потом их дочерние, иначе будет ошибка загрузки данных.

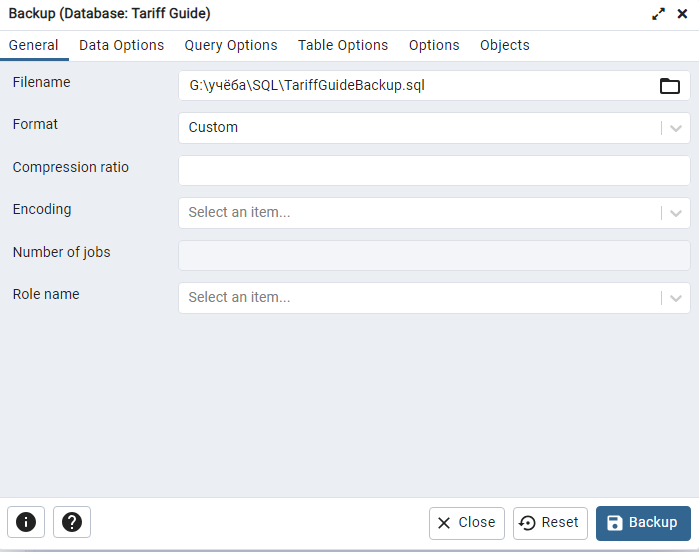
Теперь приступим к созданию резервной копии нашей БД.

## Создание резервной копии БД

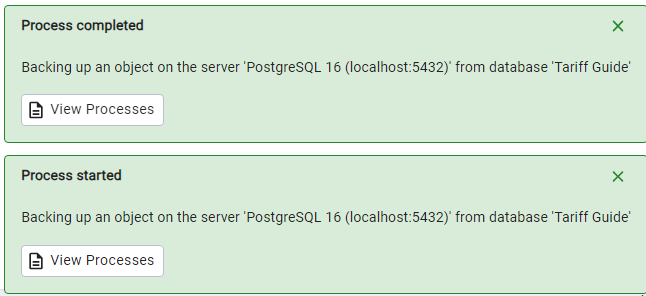
Для создания резервной копии БД воспользуемся опцией Backup в pgAdmin4:



Укажем название и путь, в котором будем хранить резервную копию:



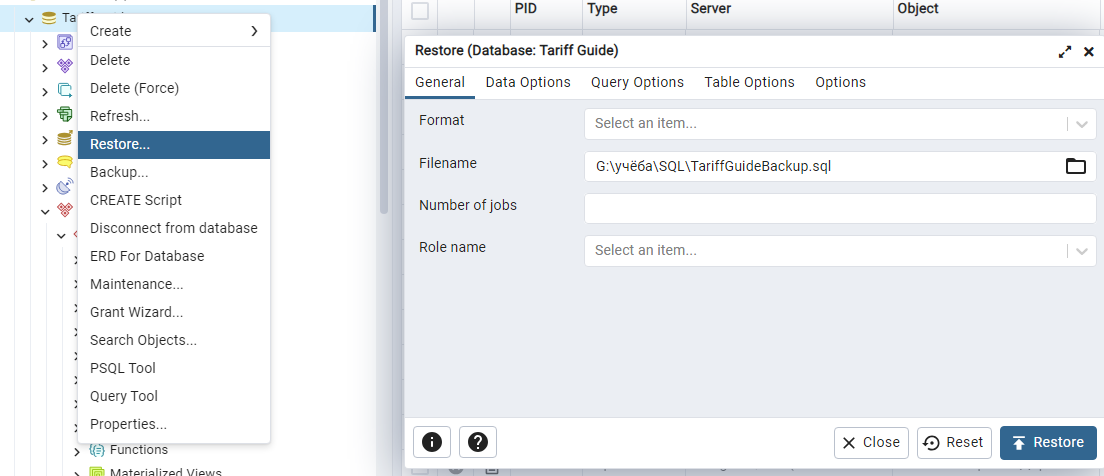
Получаем уведомления об успешном начале и завершении бэкапа:

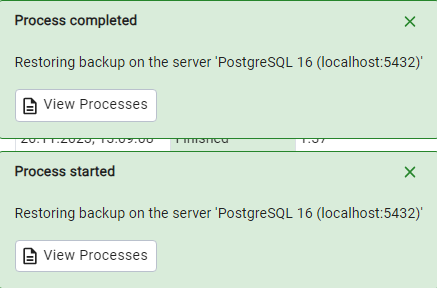


И проверяем, что файл появился в папке:



Теперь при необходимости БД может быть восстановлена опцией Restore:

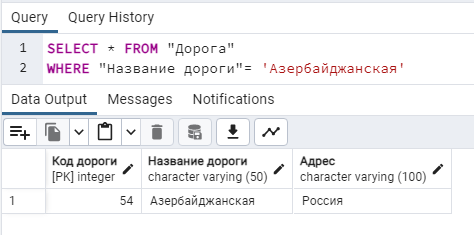




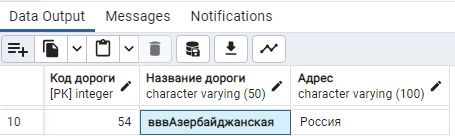
Данные загружены, резервная копия создана, можно переходить к запросам.

## SQL запросы к БД

Перед началом создания запросов проверим, работает ли каскадное обновление данных. Для этого выберем ЖД «Азербайджанская» (убедившись, что она есть в таблице «Дорога») и изменим её, добавив несколько букв:

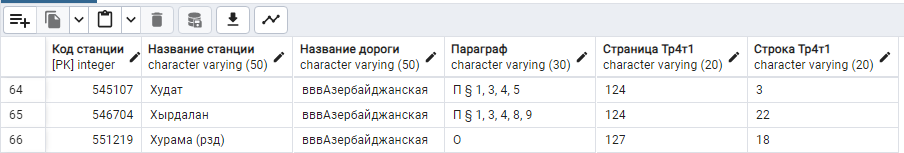


Дорога есть, вносим изменения и видим, сто они успешны:





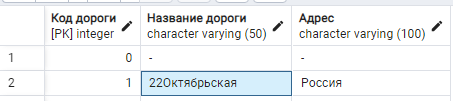
Проверяем данные в дочерней таблице «Станция» и убеждаемся, что каскадное обновление работает исправно.



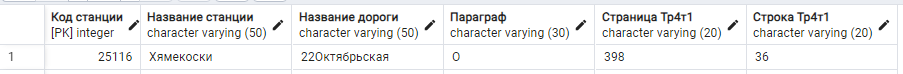
Теперь откатим изменения и приступим к запросам.

### SQL запрос на изменение данных

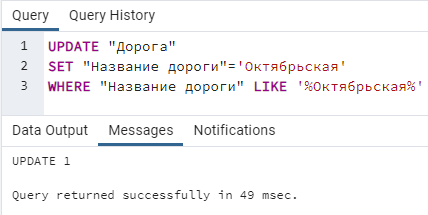
Испортим одно название дороги, чтобы потом изменить его на правильное:



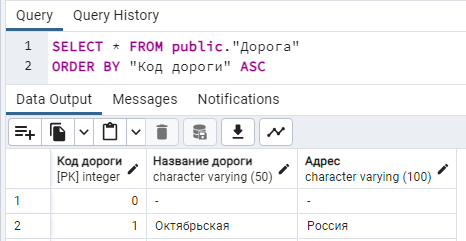
Убеждаемся, что в дочерней таблице значение тоже неверное:

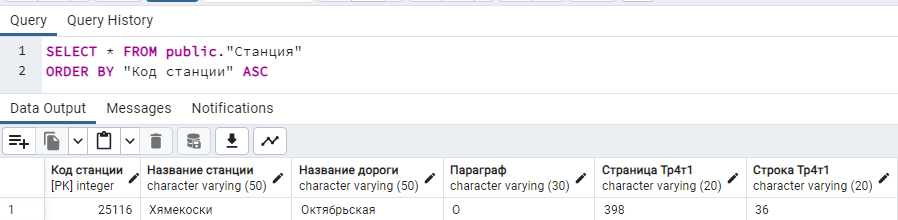


Теперь создаем запрос, который в таблице «Дорога» пройдет по всем строкам и изменит на правильные те, у которых в ячейке «Название дороги» встречается вхождение слова «Октябрьская». Видим, что изменение успешно и исправлено одно значение:



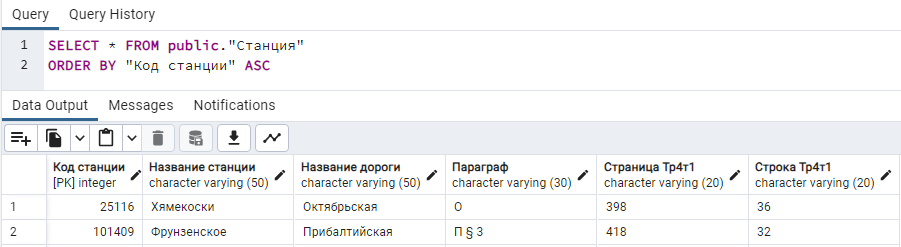
Теперь выведем материнскую и дочернюю таблицы, чтобы убедиться, что исправление успешно:

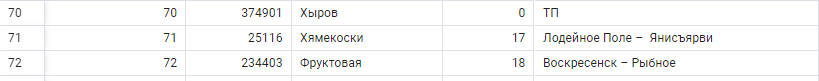




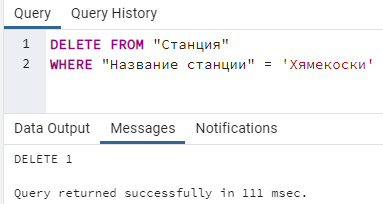
### SQL запрос на удаление данных

Выберем в таблицах строку, которою хотим удалить (Хямекоски):

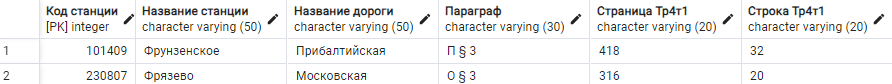


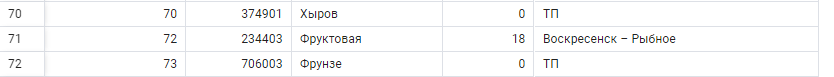


Создаем запрос на удаление и видим уведомление об успешном удалении данных:



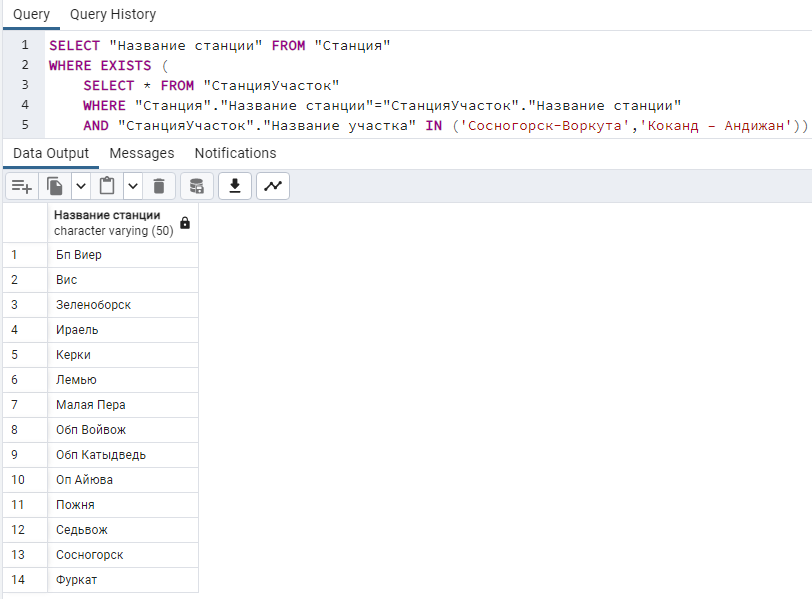
Убеждаемся, что данные удалились и из материнской и из дочерних таблц:



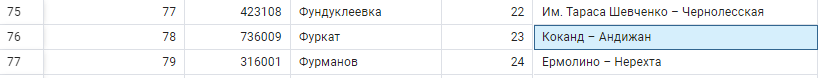
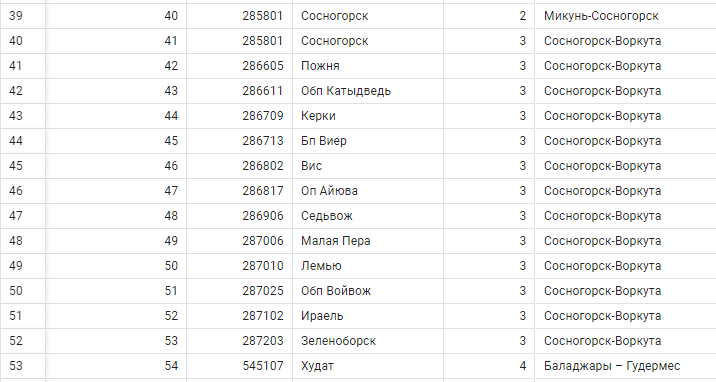


### SQL запрос на поиск данных

Для удобства сразу сделаем поиск с вложенными запросами и логическим условием. Выведем все станции, относящиеся к участкам «Сосногорск-Воркута» или «Коканд – Андижан»:

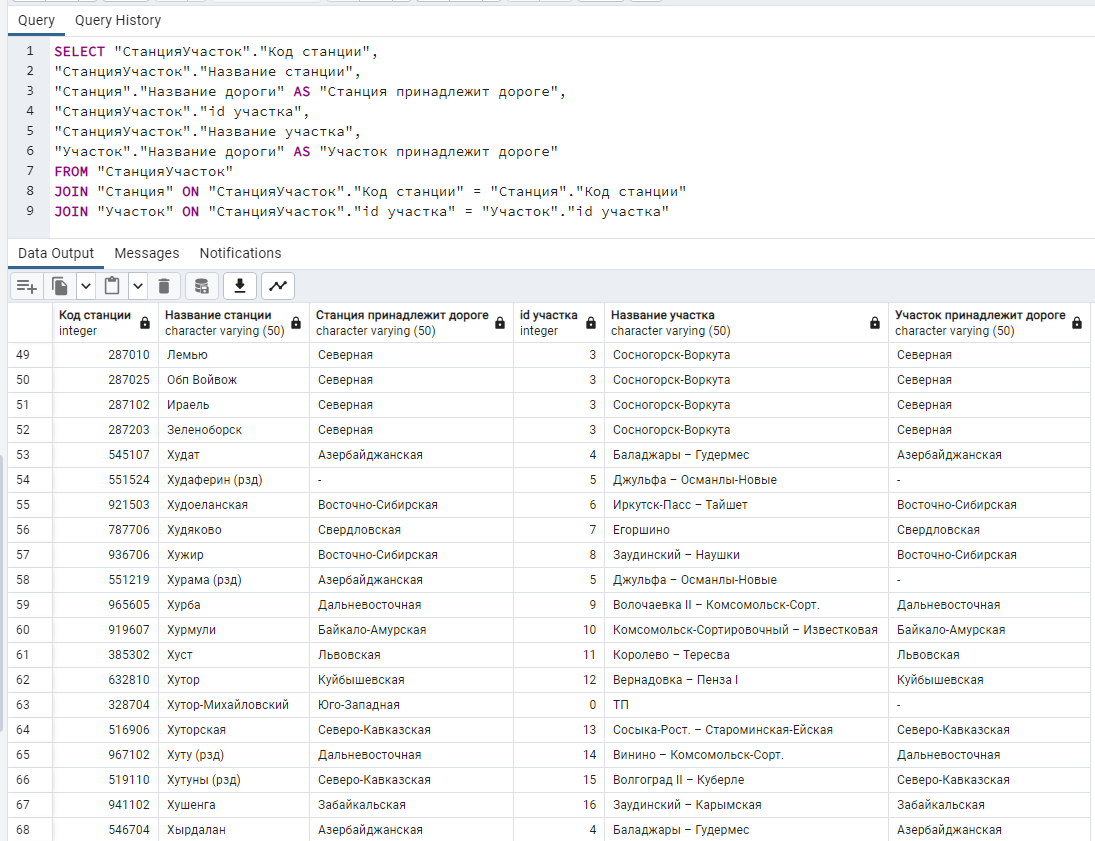


Сверяемся с таблицей и видим, что запрос отработал верно:



### SQL выборка из нескольких таблиц

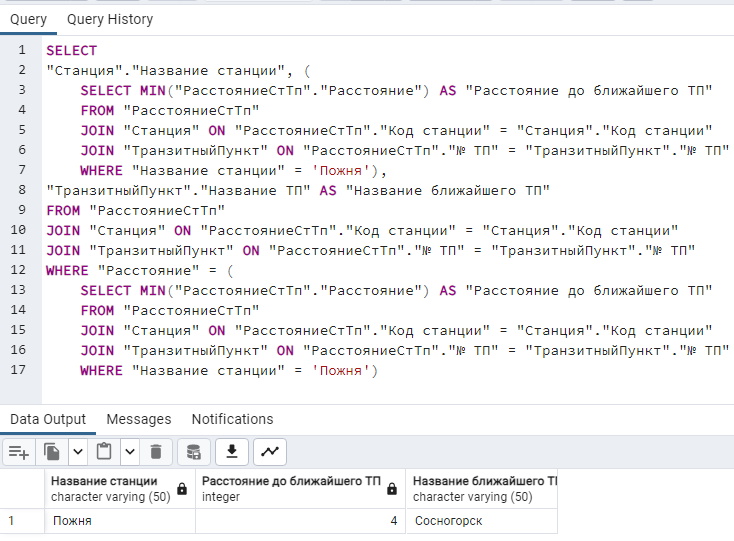
Три таблицы свяжем между собой с помощь. JOIN. В итоге данный запрос дополнит таблицу «СтанцияУчасток» названиями дорог, полученными из таблиц «Станция» и «Участок». Данный запрос поможет нам убедиться в том, что связи таблиц «Дорога», «Станция», «Участок» и «СтанцияУчасток» работают верно, ведь в данной выборке не встретилось несовпадающих названий дорог (за исключением прочерков, т.е. незаполненных в книге данных):



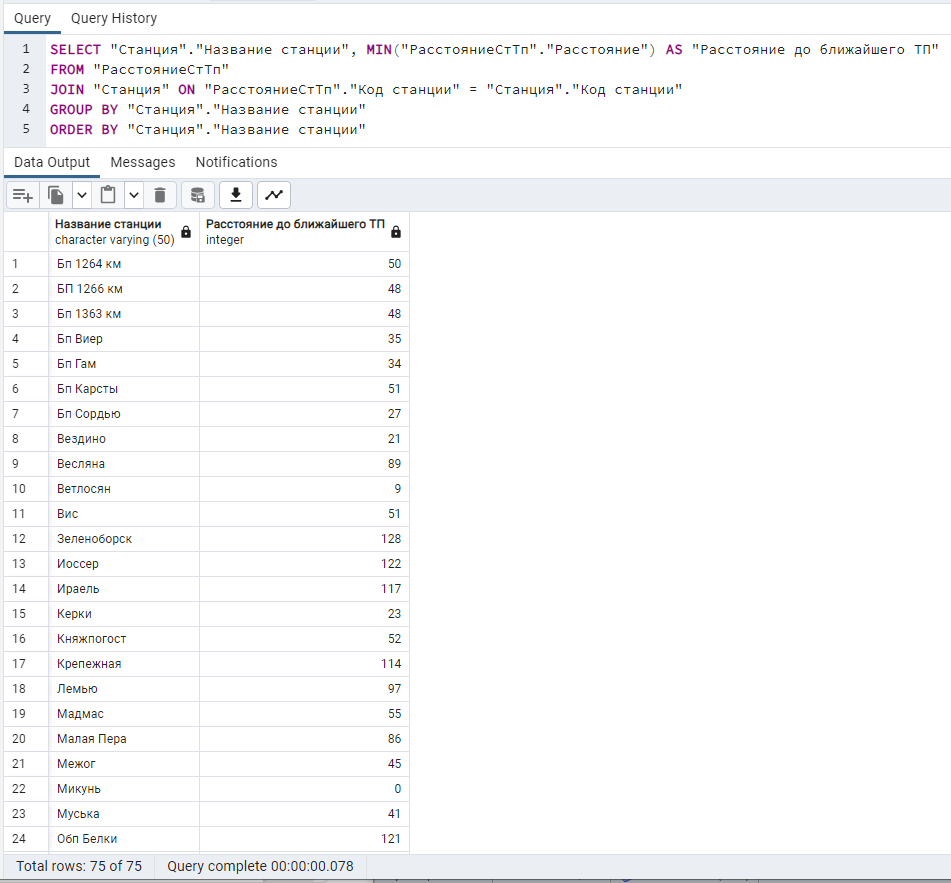
### Агрегации в SQL

Выполним несколько с агрегацией.

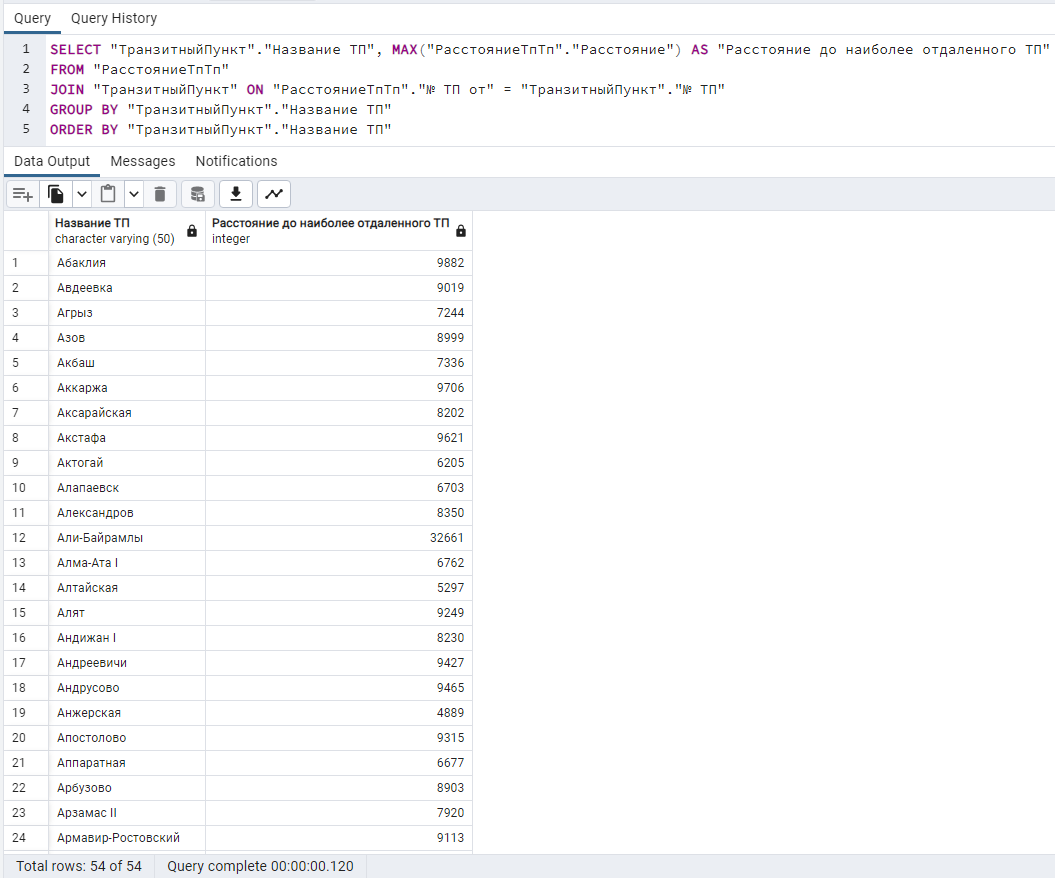
В первом запросе, связав между собой таблицы «Станция», «РасстояниеСтТп», «ТранзитныйПункт» выведем на экран ближайший к станции Пожня транзитный пункт и расстояние до него:



Во втором запросе, связав между собой таблицы «Станция» и «РасстояниеСтТп» выведем на экран расстояние до ближайшего транзитного пункта для каждой станции (сгруппировав их по названию станции):



В третьем запросе, связав между собой таблицы «ТранзитныйПункт» и «РасстояниеТпТп», выведем на экран для каждого транзитного пункта расстояние до наиболее отдаленного от него другого транзитного пункта:



# Вывод

В ходе данных лабораторных работ мы освоили программу PostgreSQL и ее графический интерфейс pgAdmin4 и научились следующему:

* создавать базу данных;
* создавать реляционные таблицы в базе данных;
* устанавливать связи между реляционными таблицами;
* выводить erd модель базы данных в программе;
* импортировать данные в базу с помощью .csv файлов;
* создавать резервную копию базы данных;
* восстанавливать базу данных из резервной копии;
* осуществлять различные sql запросы к базе данных.