

***Федеральное агентство по рыболовству***

***Федеральное государственное бюджетное образовательное***

***учреждение высшего образования***

***«Астраханский государственный технический университет»***

**Система менеджмента качества в области образования, воспитания, науки и инноваций сертифицирована**

**ООО «ДКС РУС» по международному стандарту ISO 9001:2015**

|  |  |
| --- | --- |
| Институт | Информационных технологий и коммуникаций |
| Направление | 09.03.01 Информатика и вычислительная техника |
| Профиль | «Автоматизированные системы обработки информации и управления» |
| Кафедра | «Автоматизированные системы обработки информации и управления» |

**Лабораторная работа № 17**

**«Роли в PostgreSQL**»

по дисциплине «СУБД PostgreSQL»

|  |  |  |
| --- | --- | --- |
|  |  | Работа выполнена студентом группы ДИНРБ-31  Кузургалиев Р.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Фамилия И.О.) подпись |
|  |  | Проверил работу:  ст. преподаватель Мамлеева А.Р,\_\_\_\_\_\_\_\_\_\_\_\_  (ученая степень, ученое звание, Фамилия И.О.) |

Работа защищена

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**АСТРАХАНЬ – 2024**

**СОДЕРЖАНИЕ ОТЧЕТА**

**Роли в PostgreSQL**

**Цель:** изучить роли пользователей в PostgreSQL.

Для начала необходимо создать базу данных OPTOVIK. База данных будет состоять из трёх таблиц:

* Agent (Агент);
* Customer (Заказчик)
* Order (Заказ)

Структура таблицы Agent представлена в табл. 1.

Таблица 1 – Структура таблицы Agent

| **Имя колонки** | **Тип поля, ограничения** | **Длина** | **Примечание** |
| --- | --- | --- | --- |
| ID\_агента | serial, primary key | 4 | Номер агента |
| ФИО | varchar, not null | 40 | ФИО агента |
| Город | varchar, not null | 30 | Город проживания агента |
| Комиссионные | real, not null, default 0.0 | 4 | Комиссионные агента |

Структура таблицы Customer представлена в табл. 2.

Таблица 2 – Структура таблицы Customer (Заказчик)

| **Имя колонки** | **Тип поля, ограничения** | **Длина** | **Примечание** |
| --- | --- | --- | --- |
| ID\_заказчика | serial, primary key | 4 | Номер заказчика |
| Название | varchar, not null | 50 | Название заказчика |
| Город | varchar, not null | 30 | Город нахождения заказчика |
| Рейтинг | integer | 4 | Рейтинг (от 0 до 100) |
| ID\_агента | integer, foreign key | 4 | Номер агента, закрепленного за заказчиком |

Структура таблицы Order представлена в табл. 3

Таблица 3 – Структура таблицы Order (Заказ)

| **Имя колонки** | **Тип поля, ограничения** | **Длина** | **Примечание** |
| --- | --- | --- | --- |
| ID\_заказа | serial, primary key | 4 | Номер заказа |
| Дата | date, not null | 8 | Дата заказа |
| Сумма | real, not null | 4 | Сумма заказа |
| ID\_заказчика | integer, foreign key | 4 | Номер заказчика |
| ID\_агента | integer, foreign key | 4 | Номер агента |

--CREATE DATABASE "OPTOVIK";

CREATE TABLE "Agent" (

"ID\_Agent" serial PRIMARY KEY,

"Name" varchar (40) NOT NULL,

"City" varchar (30) NOT NULL,

"Commission" real DEFAULT 0.0

);

CREATE TABLE "Customer" (

"ID\_Customer" serial PRIMARY KEY,

"Name" varchar (50) NOT NULL,

"City" varchar (30) NOT NULL,

"Rating" integer,

"ID\_Agent" integer REFERENCES "Agent"

);

CREATE TABLE "Order" (

"ID\_Order" serial PRIMARY KEY,

"Sum" real NOT NULL,

"Date" date DEFAULT CURRENT\_DATE,

"ID\_Customer" integer REFERENCES "Customer",

"ID\_Agent" integer REFERENCES "Agent"

);

INSERT INTO "Agent" ("Name", "City", "Commission") VALUES

('Иванов И.И.', 'Астрахань', 10000.00),

('Петров П.П.', 'Астрахань', 22000.00),

('Сидоров С.С.', 'Волгоград', 15000.00),

('Горяев Р.А.', 'Элиста', 8000.00),

('Родионов Н.О.', 'Ростов-на-Дону', 30000.00),

('Медведев Л.Д.', 'Ростов-на-Дону', 18000.00),

('Арефьев В.А.', 'Ставрополь', 25000.00),

('Краснов С.В.', 'Краснодар', 32000.00),

('Мурзалиев М.М.','Махачкала', 12000.00),

('Дудкин В.И.', 'Владикавказ', 20000.00);

INSERT INTO "Customer" ("Name", "City", "Rating", "ID\_Agent") VALUES

('Три кота', 'Астрахань', 65, 1),

('ЭЛКО', 'Астрахань', 40, 2),

('ТЦ Восток', 'Астрахань', 25, 1),

('ЦУМ', 'Астрахань', 40, 2),

('Астрахань-Гарант-Сервис', 'Астрахань', 80, 1),

('Музей В.О.В.', 'Волгоград', 55, 3),

('Речной порт', 'Волгоград', 20, 3),

('Мясокомбинат', 'Ставрополь', 60, 7),

('Молокозавод', 'Ставрополь', 65, 7),

('Турфирма ОТДЫХ', 'Краснодар', 40, 8),

('Гостиничный комплекс МОСКВА', 'Краснодар', 70, 5),

('Кондитерская фабрика', 'Краснодар', 65, 8),

('Издательство КАВКАЗ', 'Краснодар', 80, 8),

('Трикотажная фабрика', 'Элиста', 75, 4),

('ООО Бензин Калмыкии', 'Элиста', 20, 4),

('Консервный завод', 'Махачкала', 50, 9),

('ООО Северная Осетия', 'Владикавказ', 35, 10),

('ООО ПРЕДГОРЬЕ', 'Грозный', 60, 10),

('Аэропорт РОСТОВ', 'Ростов-на-Дону', 75, 6),

('Конный завод им. С.М.Буденного', 'Ростов-на-Дону', 80, 5),

('Часовой завод ЛУЧ', 'Ростов-на-Дону', 40, 6),

('Морской порт', 'Таганрог', 60, 6);

INSERT INTO "Order" ("Sum", "Date", "ID\_Customer", "ID\_Agent") VALUES

(100000.00, '01-09-2014', 3, 1),

( 75000.00, '05-09-2014', 7, 3),

(120000.00, '08-09-2014', 2, 2),

( 50000.00, '11-09-2014', 4, 2),

(230000.00, '15-09-2014', 1, 1),

(150000.00, '18-09-2014', 5, 1),

(300000.00, '20-09-2014', 8, 7),

(170000.00, '20-09-2014', 6, 3),

(340000.00, '23-09-2014', 9, 7),

(500000.00, '25-09-2014', 11, 5),

(450000.00, '29-09-2014', 10, 8),

(350000.00, '30-09-2014', 13, 8),

(180000.00, '02-10-2014', 12, 8),

(370000.00, '04-10-2014', 15, 4),

( 90000.00, '06-10-2014', 17, 10),

(460000.00, '09-10-2014', 14, 4),

(290000.00, '10-10-2014', 18, 10),

(420000.00, '10-10-2014', 16, 9),

(310000.00, '16-10-2014', 20, 5),

(130000.00, '21-10-2014', 19, 6),

(490000.00, '26-10-2014', 22, 6),

(200000.00, '31-10-2014', 21, 6),

(400000.00, '31-10-2014', 21, 6);

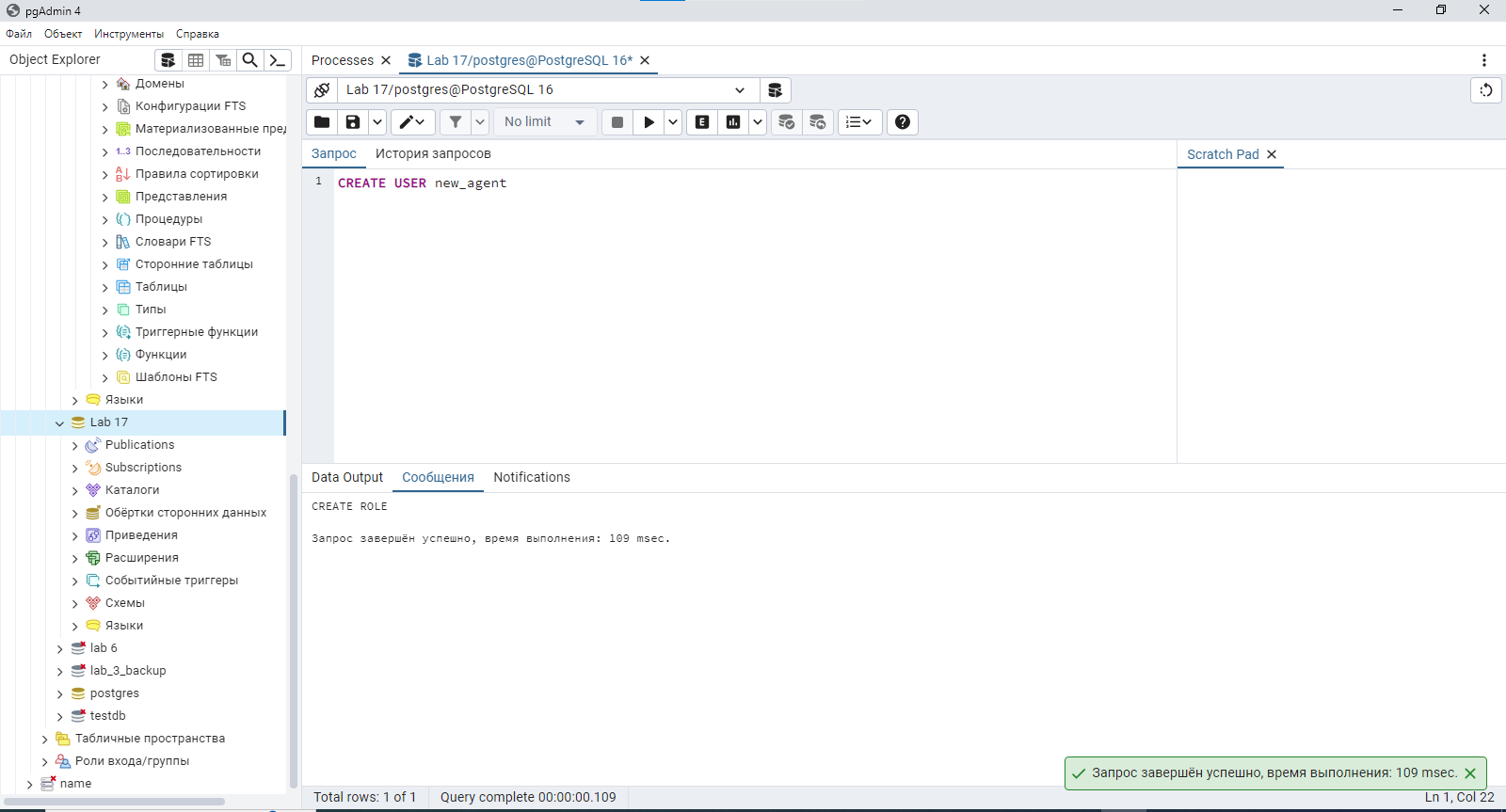
Заполните таблицу произвольными данными. Пример данных для заполнения таблиц можно взять из лабораторных работ по базам данных второго курса.

В [документации PostgreSQL](https://postgrespro.ru/docs/postgresql/9.6/sql-createrole.html) приведён подробный синтаксис команды CREATE ROLE – создание роли в базе данных. В лабораторной работе будет использоваться несколько устаревший, но более говорящий, псевдоним ROLE – USER. Например, команда CREATE USER эквивалентна CREATE ROLE LOGIN.

Если необходимо создать группу, которая по логике не является отдельным пользователем, а значит и не может подключаться к системе, то можно использовать не CREATE ROLE, а CREATE GROUP. Оба варианты эквиваленты, так как в обоих случаях подразумевается параметр NOLOGIN.

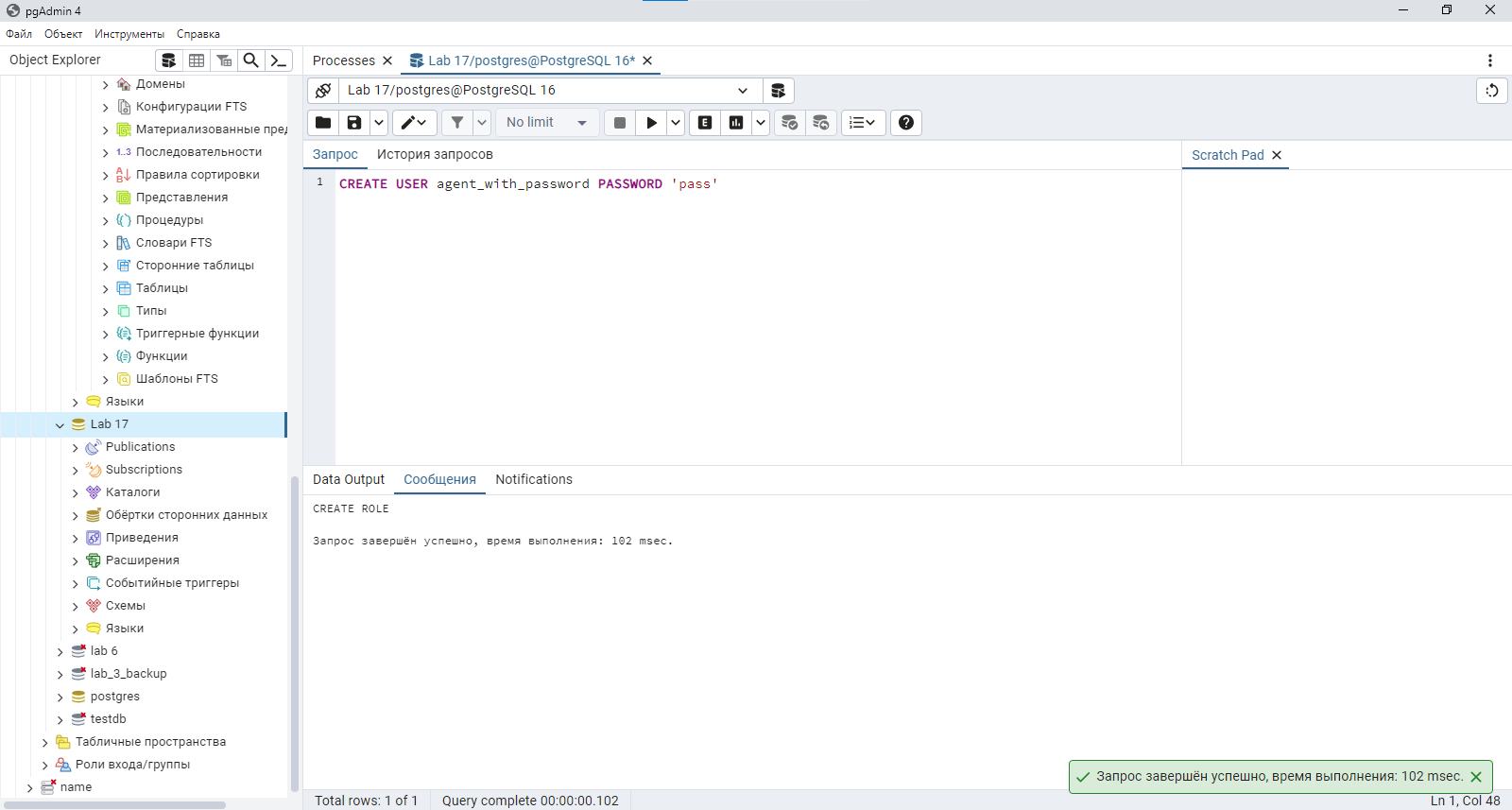
Так же, в лабораторной работе будут использоваться команды [GRANT](https://postgrespro.ru/docs/postgresql/9.6/sql-grant.html) (наделить правами доступа) и [REVOKE](https://postgrespro.ru/docs/postgresql/9.6/sql-revoke.html) (отозвать права доступа), синтаксис которых так же подробно описан в документации к PostgreSQL.Создайте нового пользователя **агента**.

CREATE USER new\_agent

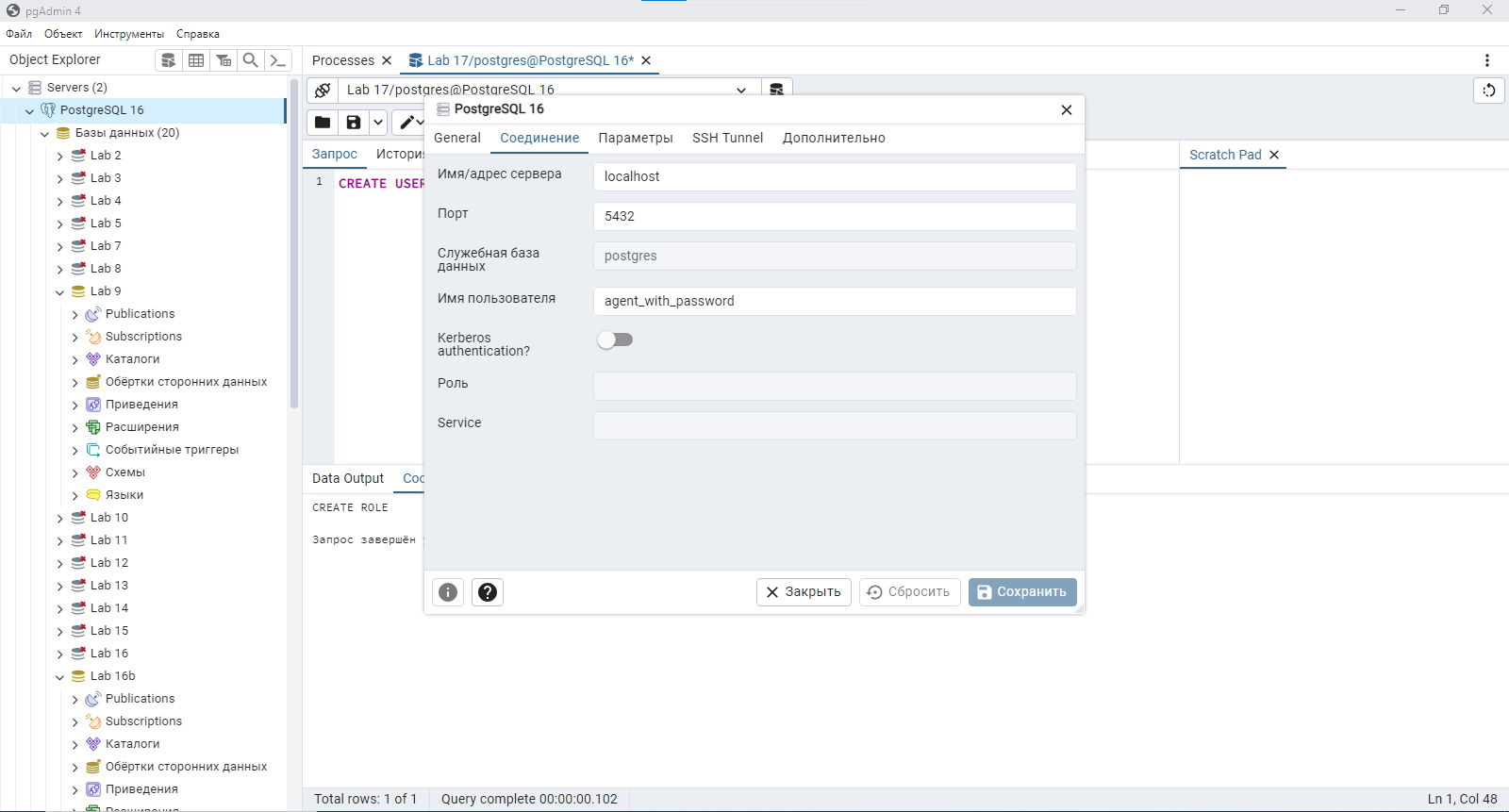


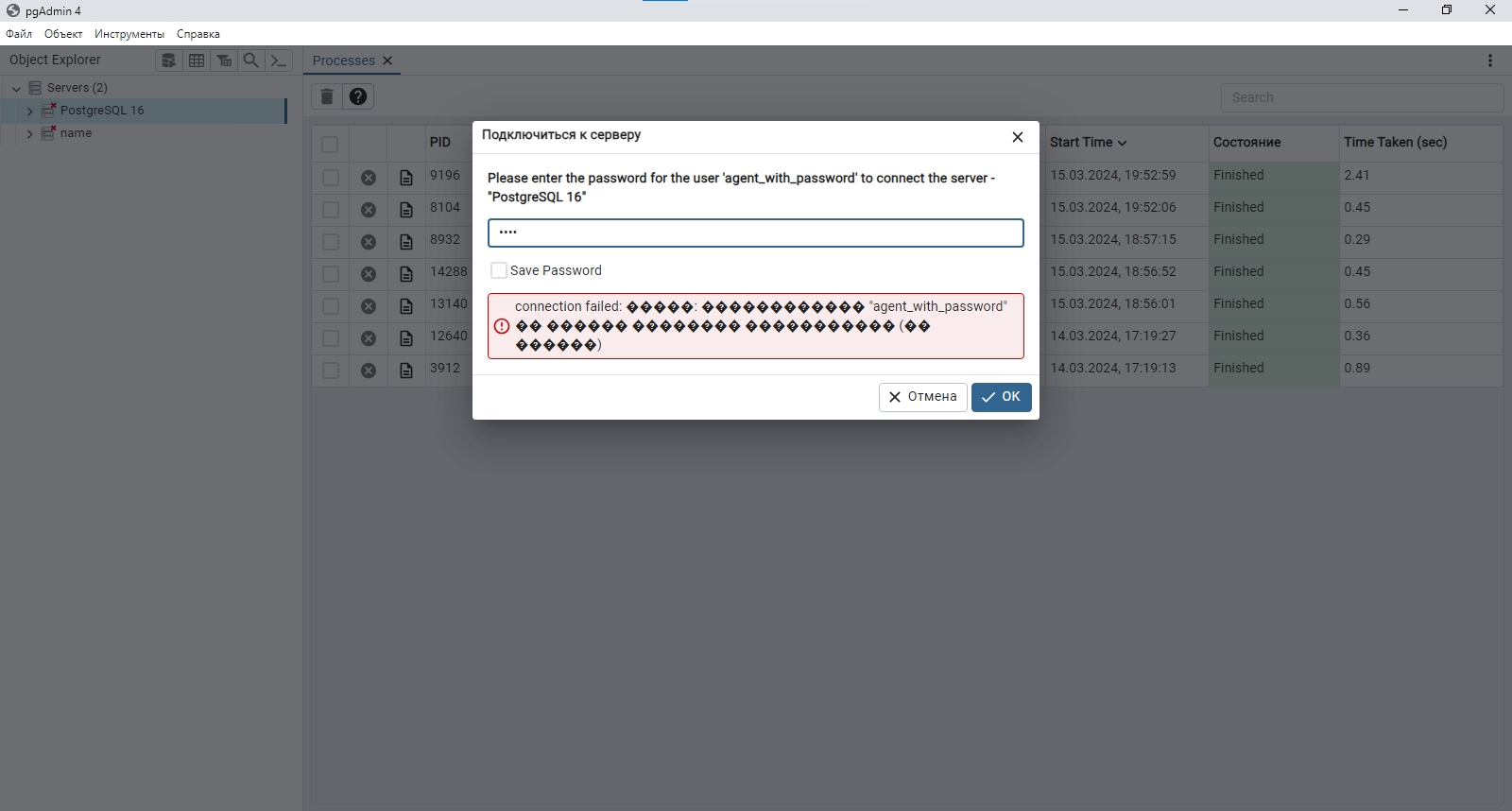
1. Создайте нового **агента** с паролем.

CREATE USER agent\_with\_password PASSWORD ‘pass’

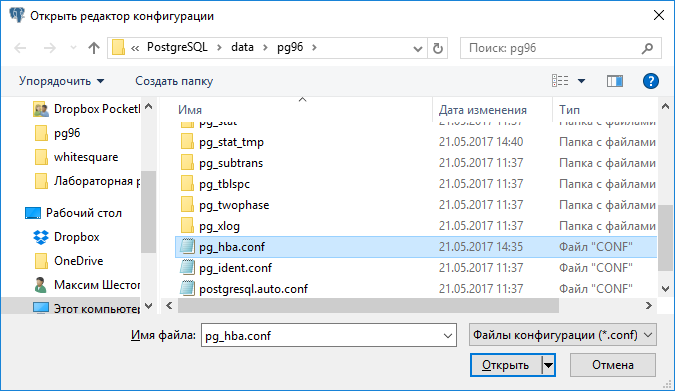


1. Попробуйте зайти в систему под каждой из созданных ролей: new\_agent, agent\_with\_password. Для того, чтобы в pgAdmin зайти на сервер под другим пользователем необходимо нажать на нужный сервер правой кнопкой мыши, выбрать из контекстного меню пункт «Свойства» и в поле «Имя пользователя» ввести имя пользователя под которым хотите зайти. Опишите результаты.





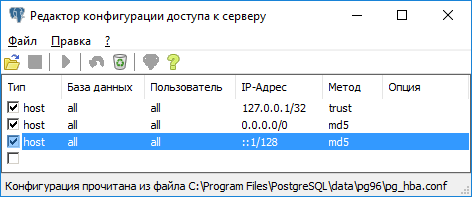
1. Убедитесь, что agent\_with\_password может подключаться без пароля. Это необходимо исправить. Для этого отсоединяемся от сервера. В pgAdmin во вкладке «Файл» выбираем пункт «Открыть pg\_hba.conf...». Появится окно выбора файла как на рис. 1.

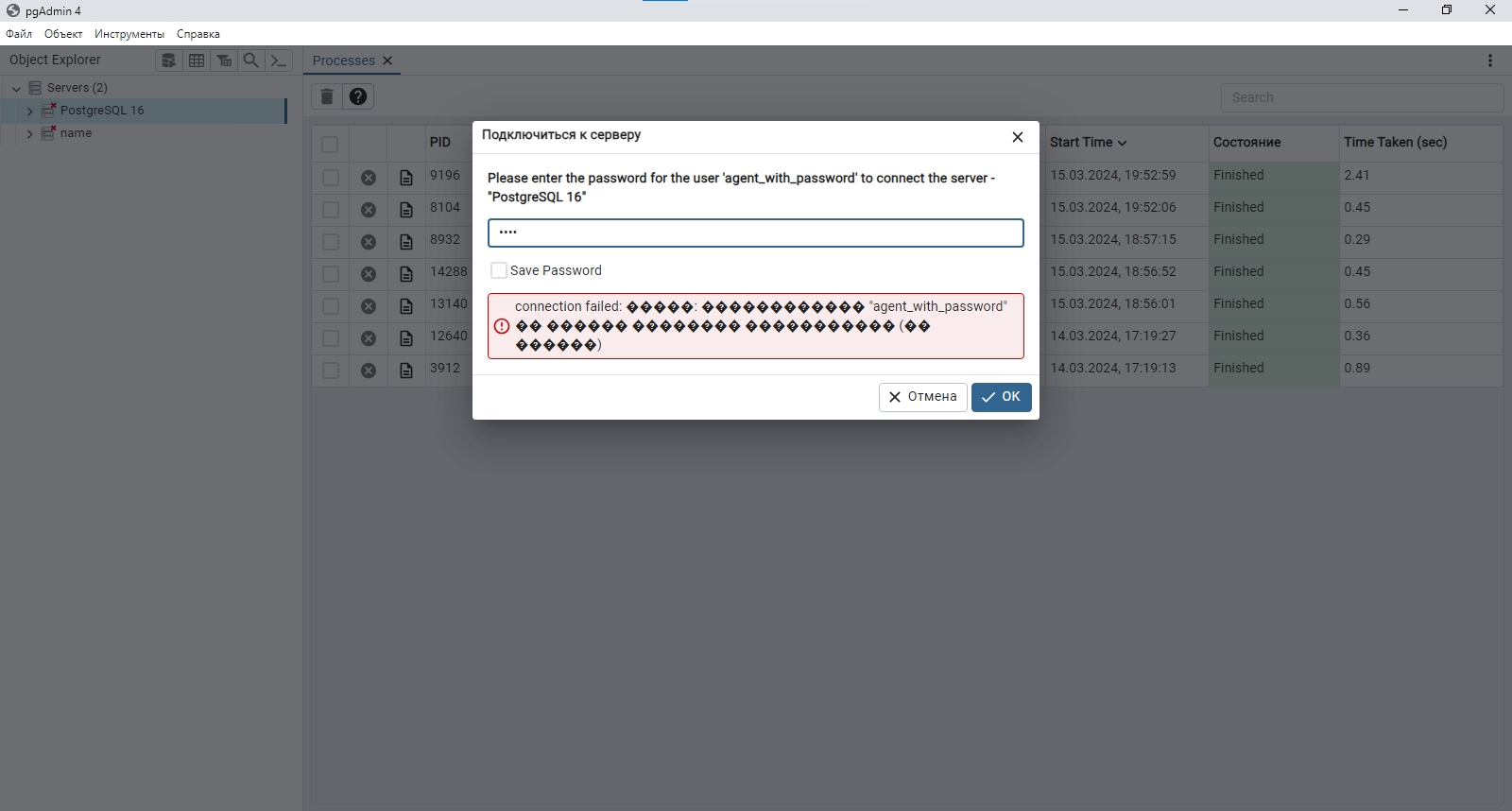
Рисунок 1 — Расположение файла pg\_hba.conf

Примерное расположение файла так же показано на этом же рисунке.

После выбора файла станет доступно следующее окно со списком (см. рис. 2).

В последнем пункте списка необходимо исправить «Метод» на «md5».

Рисунок 2 — Редактор конфигурации доступа к серверу

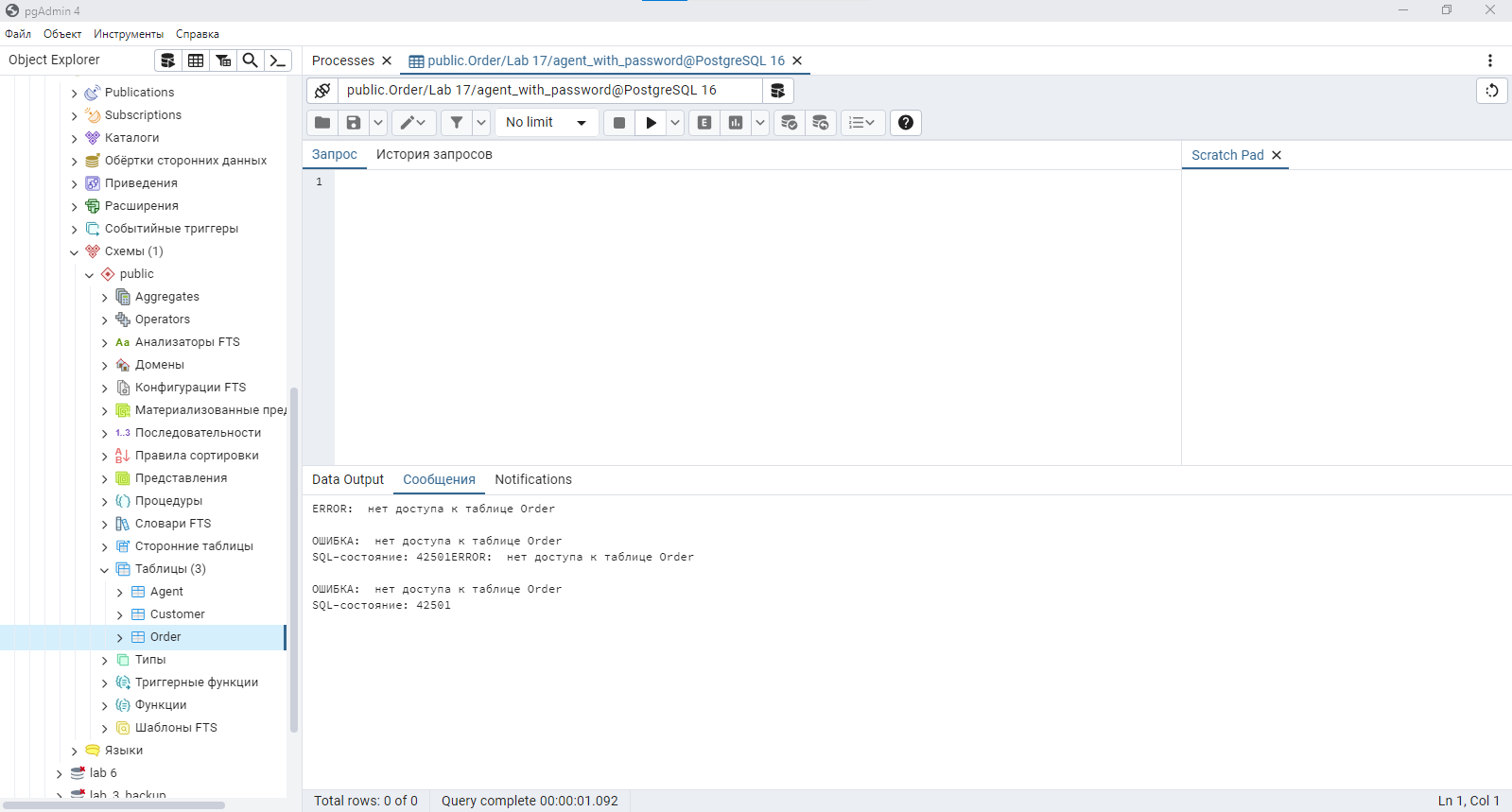
1. Сохраните измененния и попробуйте ещё раз подключится к серверу в качестве пользователя agent\_with\_password. Убедитесь, что пароль для входа теперь требуется.
2. 
3. Однако теперь пользователь new\_agent никак не может подключиться к серверу. Для того, чтобы исправить эту оплошность добавьте в файл pg\_hba.conf запись:

host all new\_agent ::1/128 trust

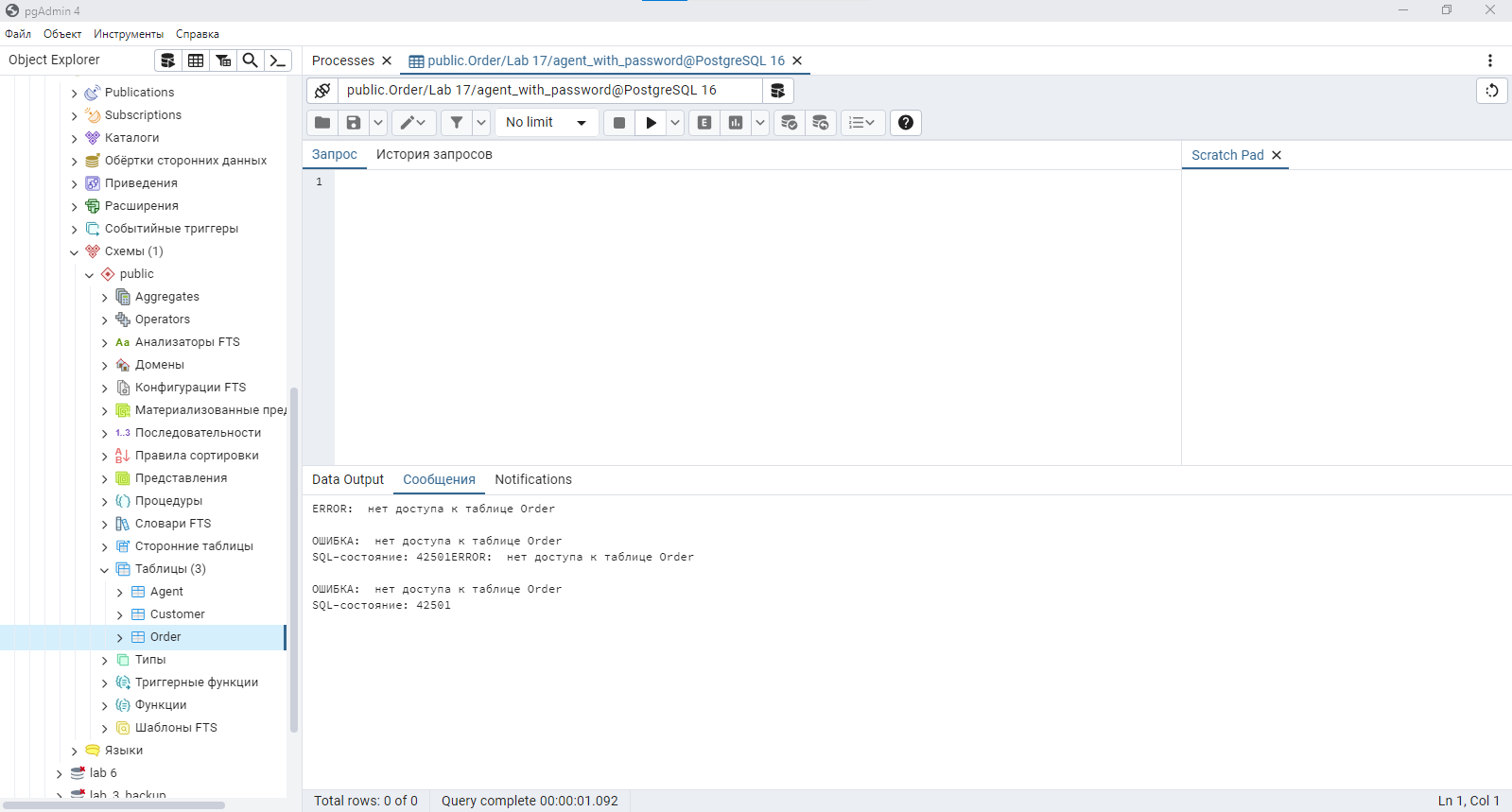
Убедитесь, что теперь new\_agent может подключаться к серверу.

1. Находясь в системе в качестве агента, попробуйте произвести различные операции над таблицами, например:

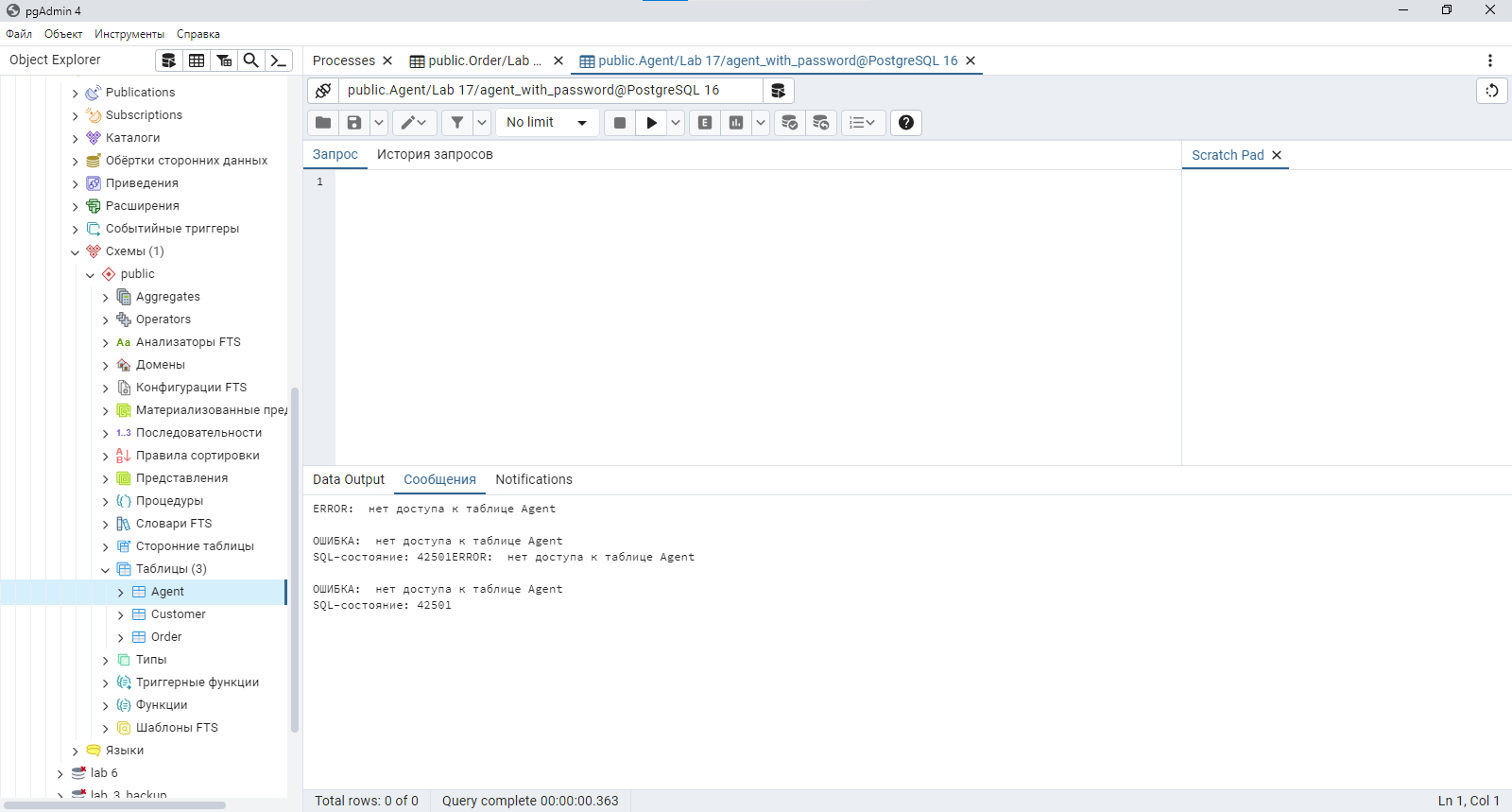
* Удалите любого (самого вредного) **заказчика**.



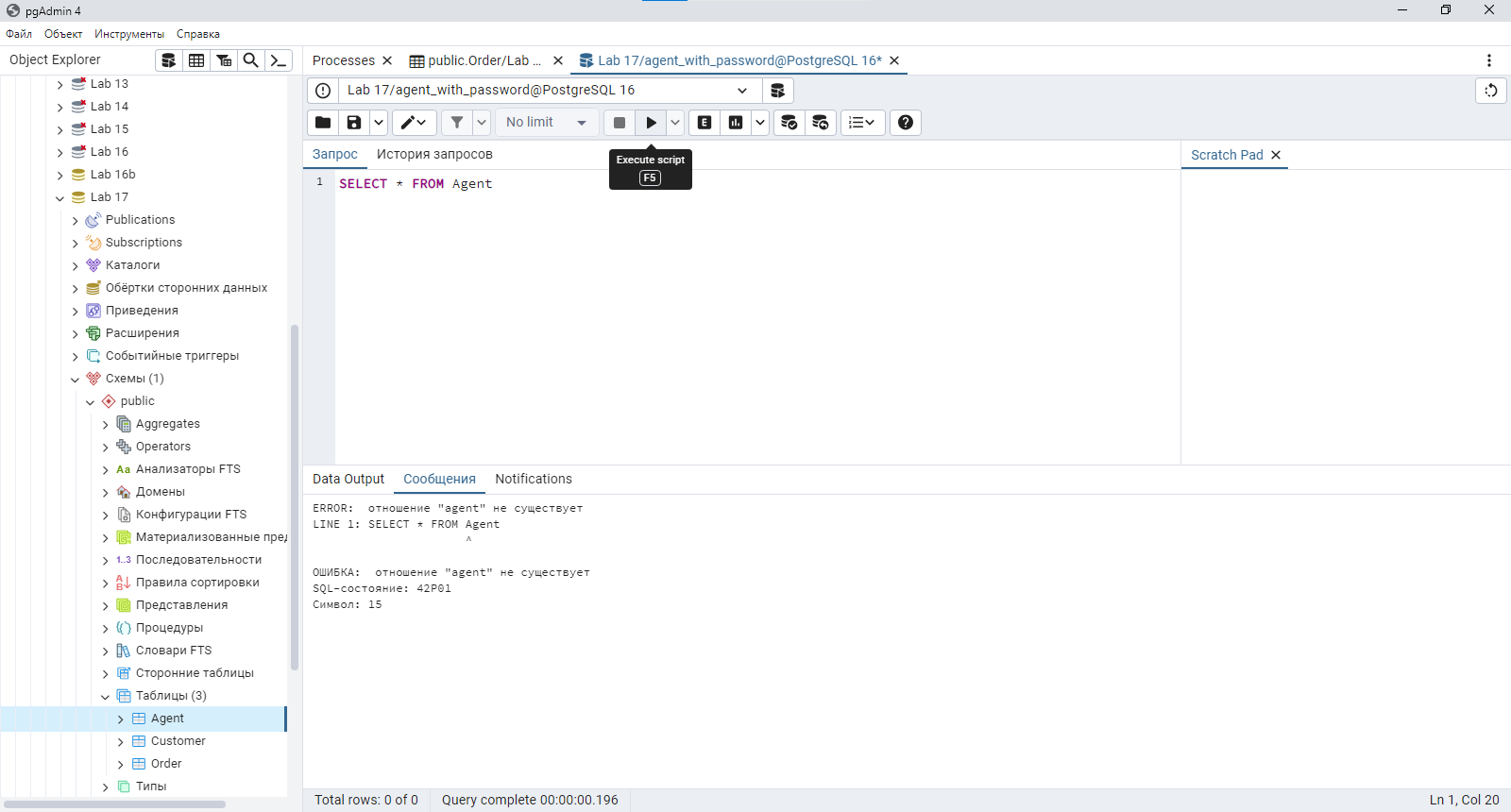
* Добавьте новый (дорогой) **заказ.**



* Увеличьте комиссионные всем **агентам**.



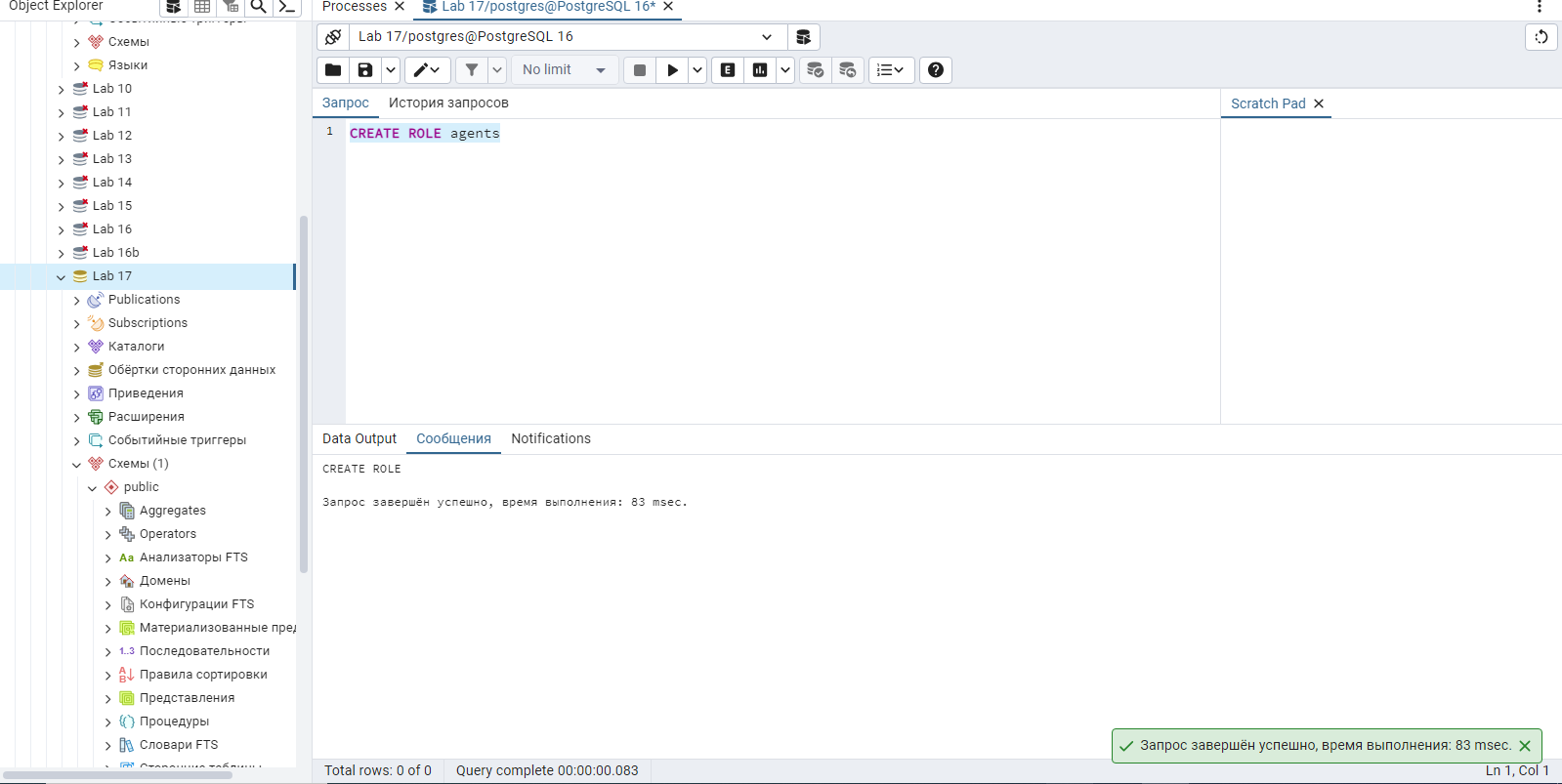
* Выведите всех агентов на экран.



Опишите результат каждой операции.

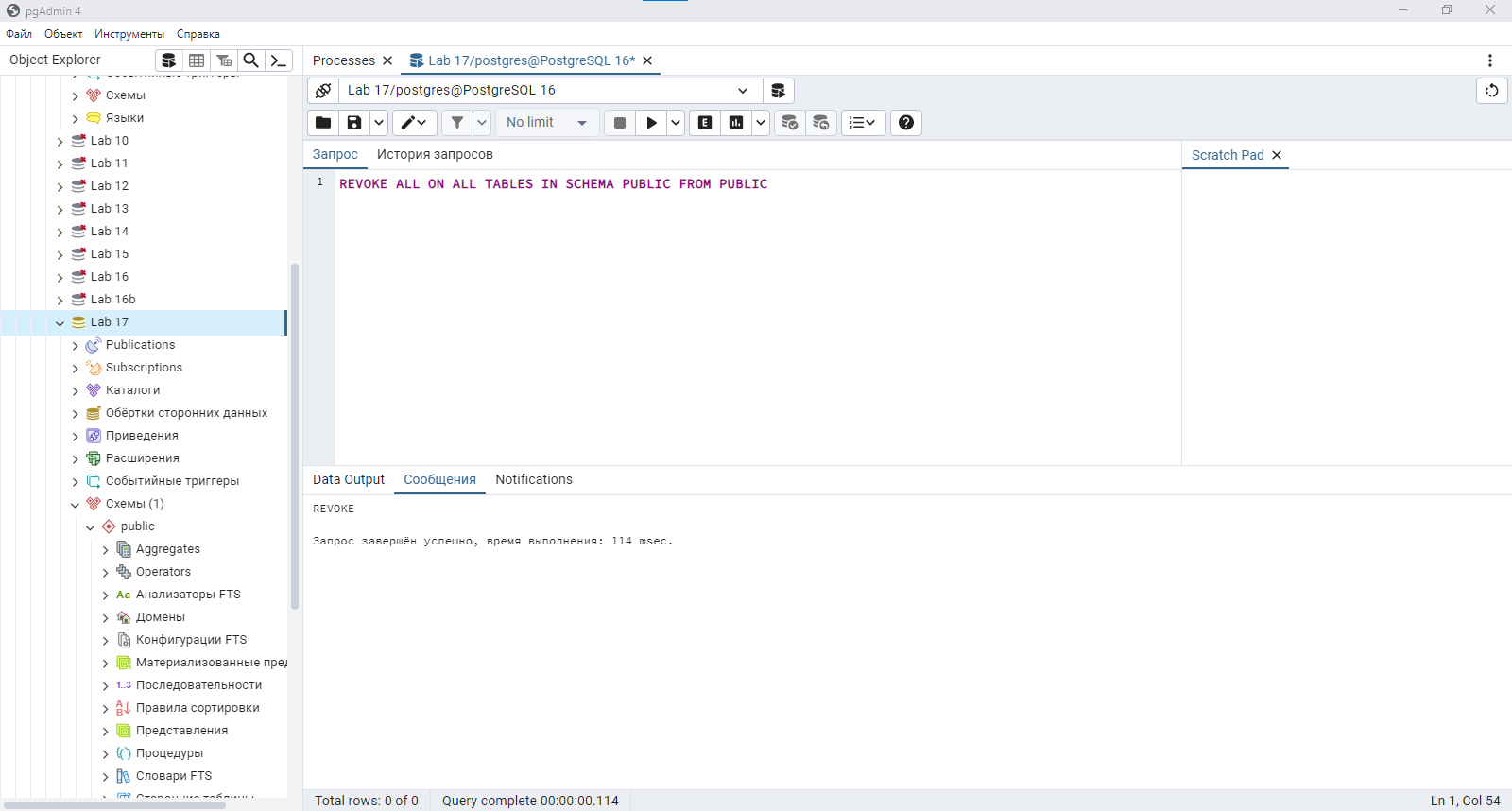
1. Очевидно, что агенты не должны иметь такую свободу действий в базе данных. Ограничим их. Для этого необходимо зайти под суперпользователем (postgres) и продолжить изучение ролей.
2. Создайте групповую роль для **агентов.**

CREATE ROLE (или GROUP) agents



1. Ограничьте **агентов** вдоступе ко всем таблицам: только чтение. Для полной уверенности, что у нас всё под контролем, закроем **всем** доступ ко **всем** таблицам.

REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC FROM PUBLIC

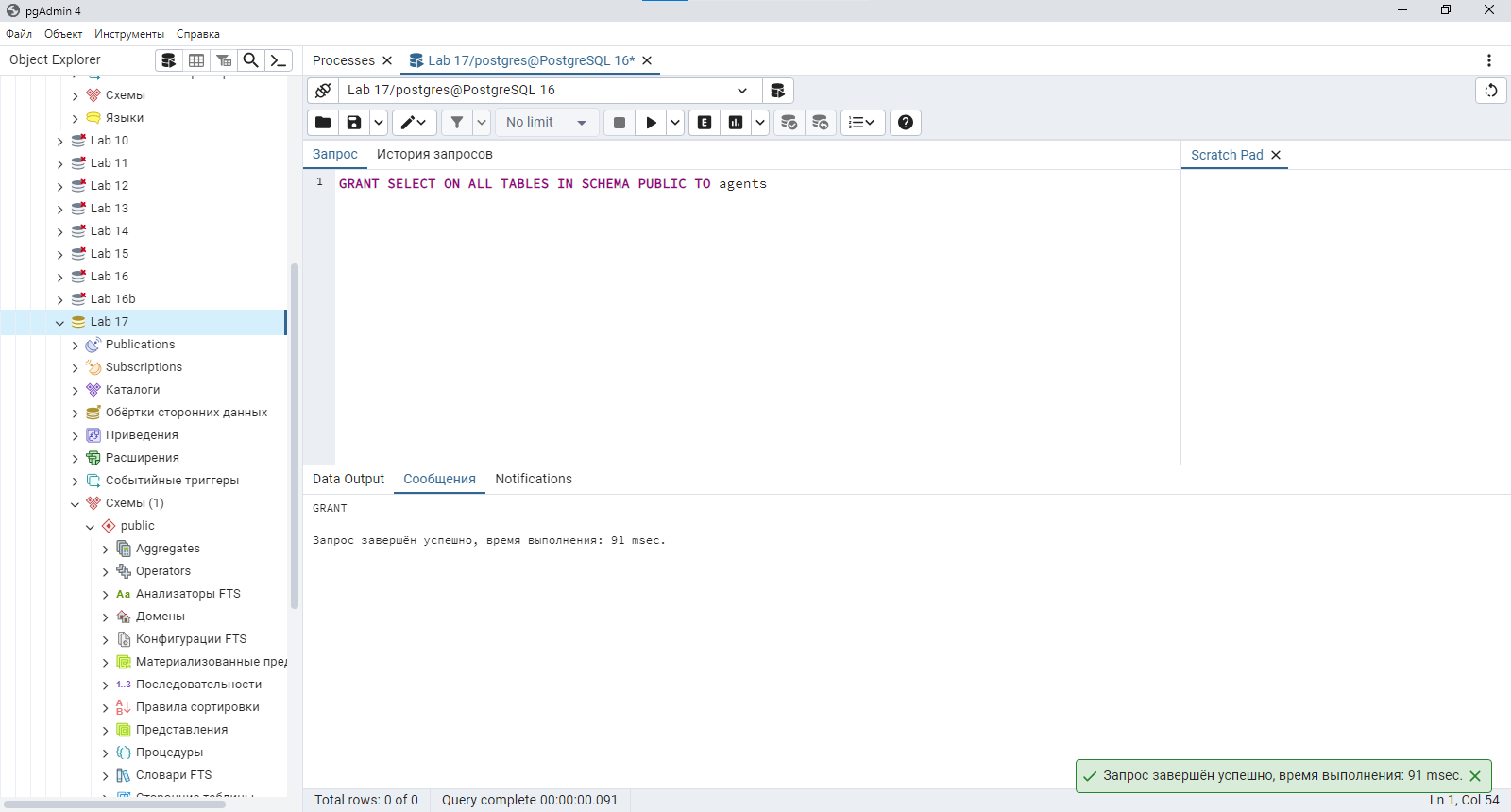


Теперь можем спокойно давать права **агентам**.

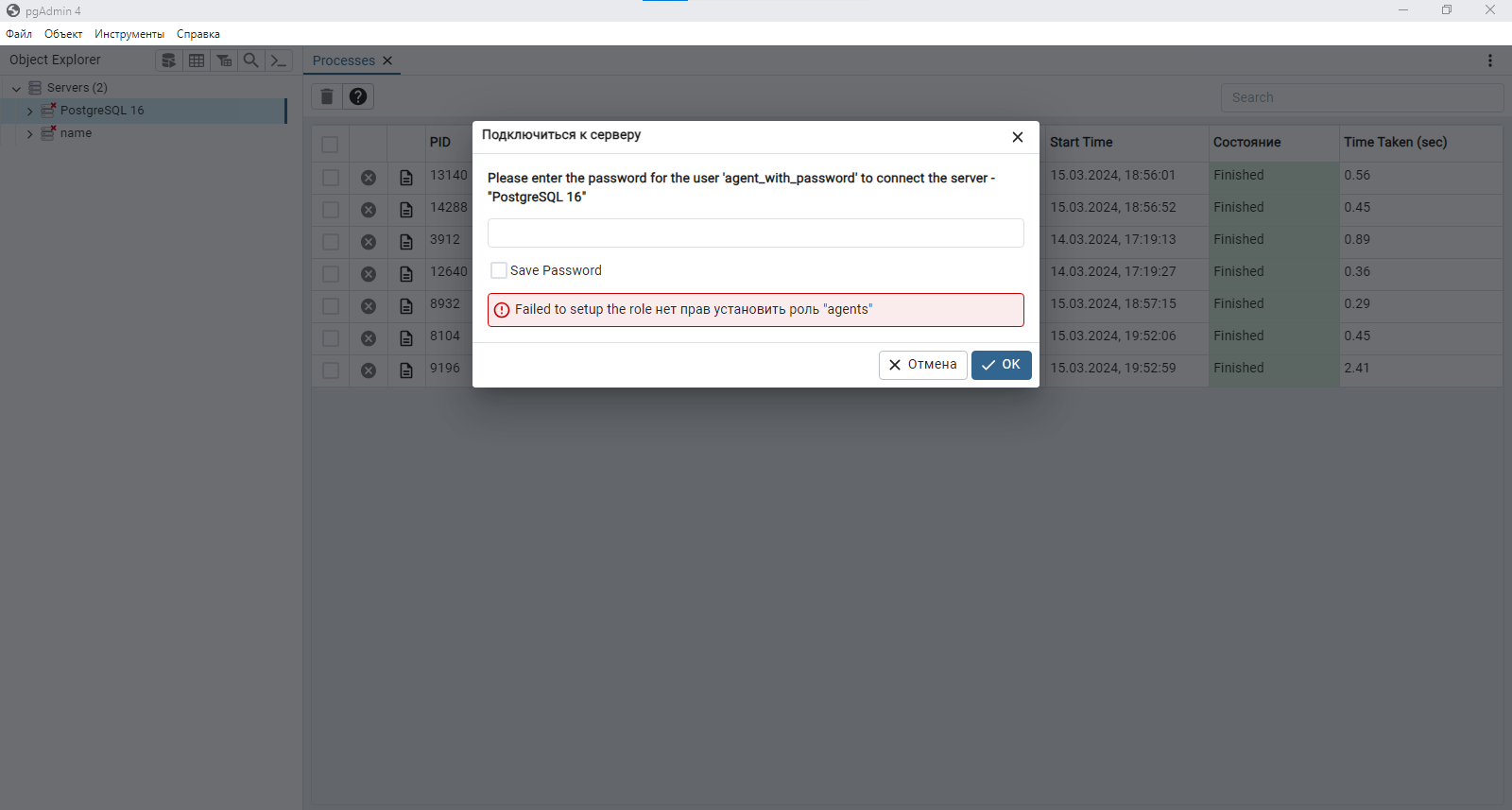
GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO agents.

1. Назначим созданным агентам роль агентов.

GRANT agents TO new\_agent, agent\_with\_password



1. Повторите действия из пункта 7 и снова опишите результат выполнения каждой операции.
2. Попробуйте на сервер зайти под ролью agents. Не забудьте предварительно внести роль agents в список доверенных (см. п. 4).
3. Убедитесь, что сервер не позволяет войти под групповой ролью.



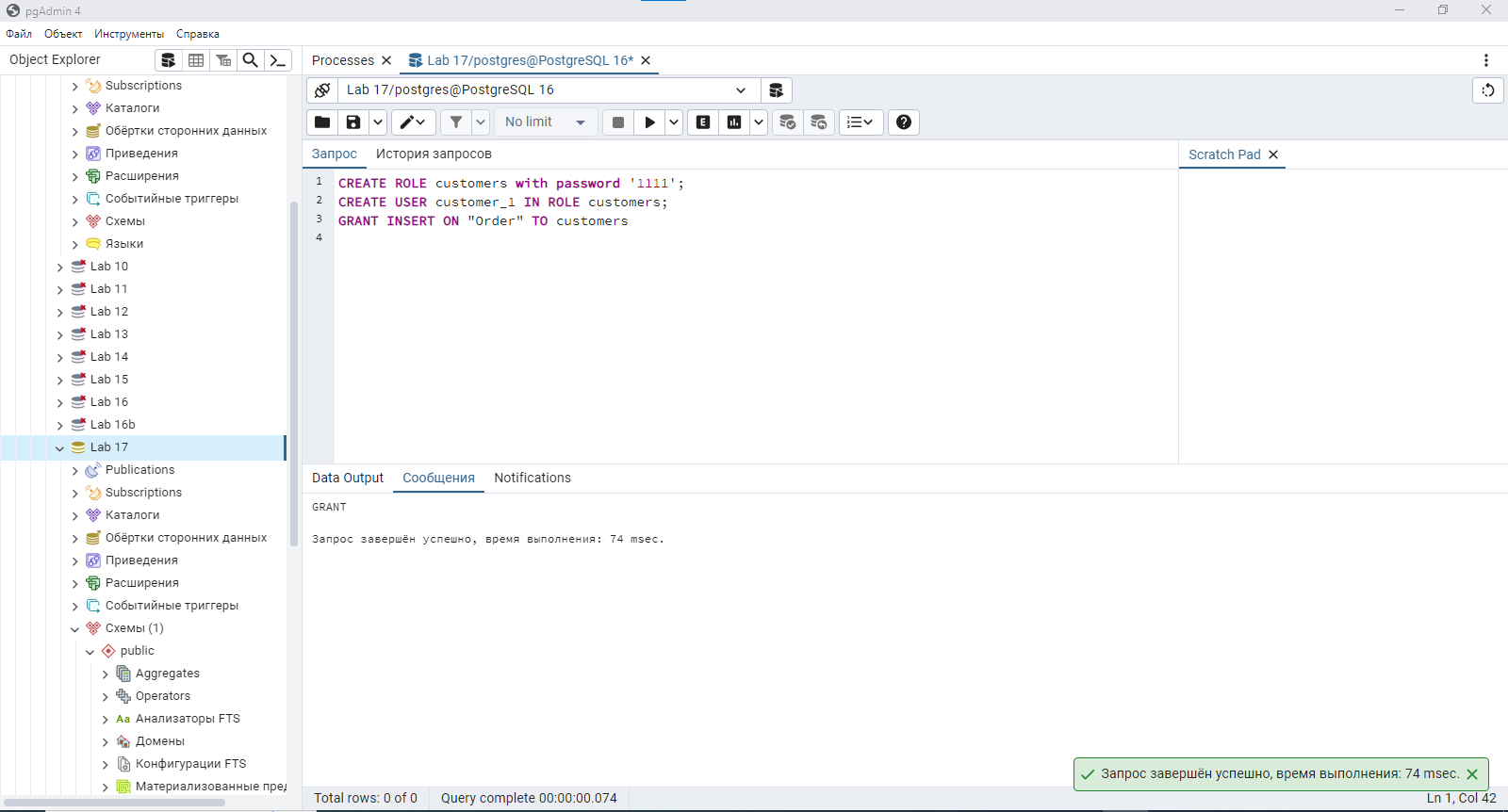
1. Создайте роль для заказчиков (см. п. 9).
2. Наделите заказчиков правами только добавлять новые заказы.

GRANT INSERT ON Order TO customers

1. Создайте новых пользователей **заказчиков**.

CREATE USER customer\_1 IN ROLE customers

В данной команде мы сразу прописали к какой роле будет относиться данный пользователь. И снова вместо команды ROLE можно использовать GROUP.

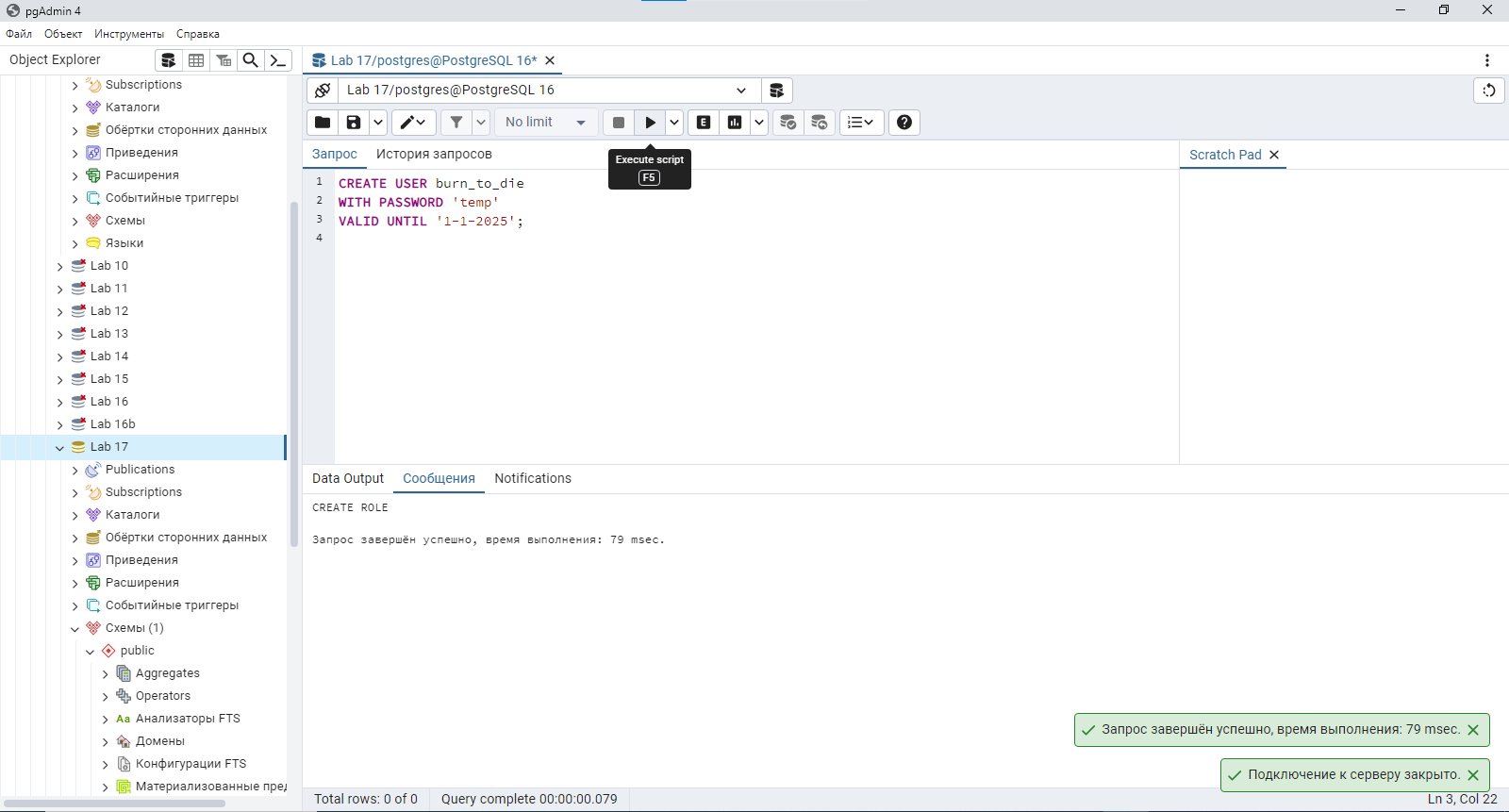


1. Попробуйте различные операции с таблицами (не менее трёх), зайдя в систему в качестве одного из заказчиков. Опишите полученные результаты.
2. Убедитесь, что заказчики могут только добавлять новые заказы.
3. Создайте пользователя с временным паролем.

CREATE USER burn\_to\_die

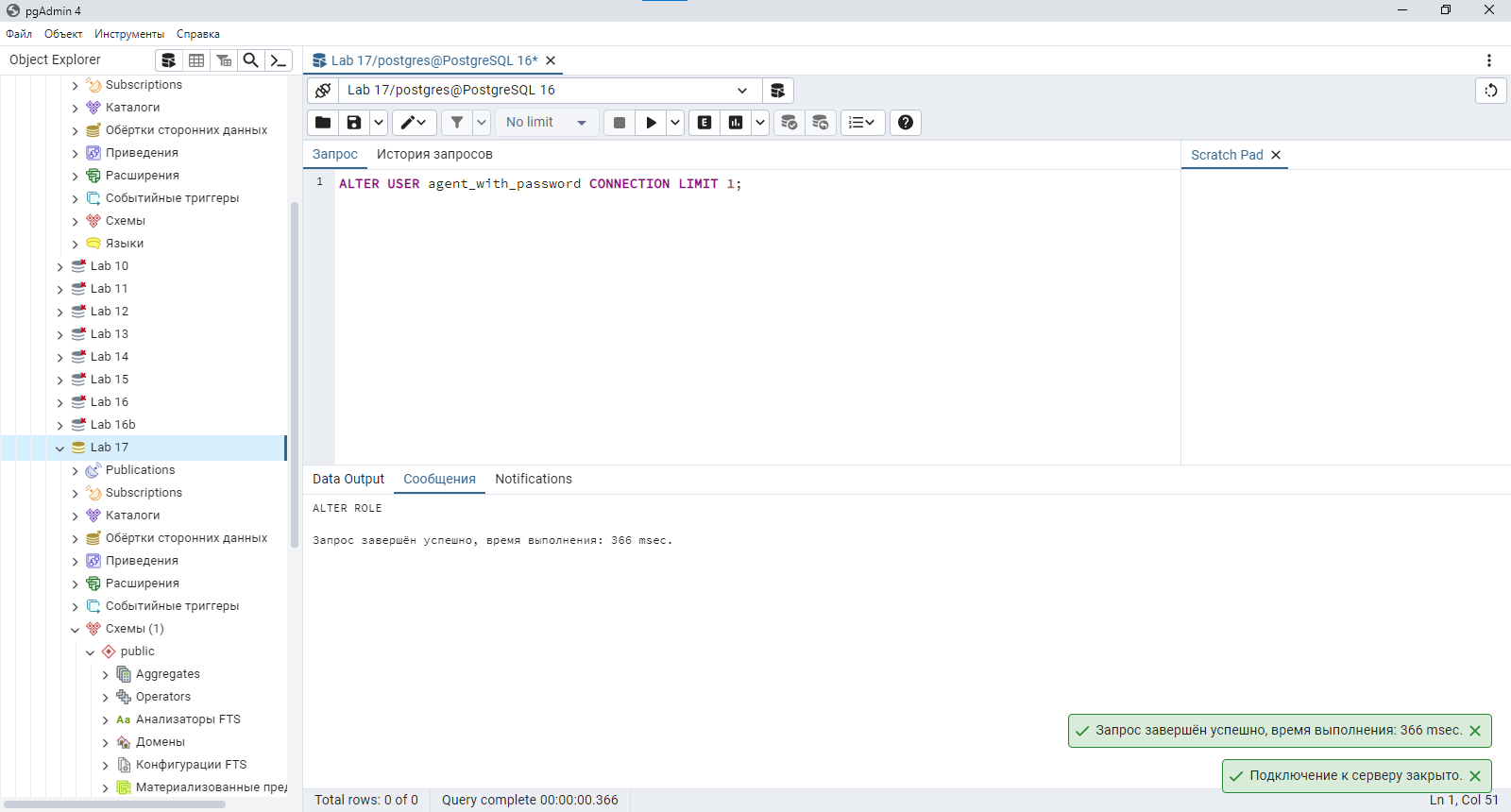
WITH PASSWORD ‘temp’

VALID UNTIL ‘дата время + столько, чтобы успеть перезайти на сервер’



1. Убедитесь, что пароль действительно был временным и после указанного вами времени сервер не позволяет пользователю войти.
2. По умолчанию, под одной ролью могут заходить неограниченное число пользователей (параметр равен -1). Предположим, что пользователь agent\_with\_password реальный и в целях безопасности ограничим количество одновременно подключенных agent\_with\_password до 1.

ALTER USER agent\_with\_password CONNECTION LIMIT 1



1. Попробуйте теперь подключиться к серверу через пользователя agent\_with\_password.
2. Убедитесь, что не получается открыть базу данных OPTOVIK.

Довольно необычный эффект, но вполне объяснимый.

Если выполнить запрос SELECT \* FROM pg\_stat\_activity можно увидеть, что agent\_with\_password уже подключен к стандартной базе данных postgres. В действительности, конечно, таких казусов быть не может, так как при соединении с сервером прописывается имя базы данных, к которой необходимо подключиться.

1. Попробуйте инициировать повторное подключение к **серверу** через пользователя agent\_with\_password. Например, это можно сделать запустив ещё один экземпляр pgAdmin.
2. Убедитесь, что сервер выдаёт ту же ошибку, что и в п. 24.
3. Создайте функцию, которая позволила бы агентам просматривать краткую информацию о заказах, на которые они назначены. Например, функция могла бы возвращать дату и сумму заказа, а так же название заказчика.

CREATE OR REPLACE FUNCTION agent\_orders(id integer)

RETURNS что-то

**SECURITY DEFINER**

AS

$$

BEGIN

Сбор информации с недоступных таблиц

END

$$ LANGUAGE plpgsql;

Ключевым моментом в данной функции является строка SECURITY DEFINER, которая временно наделяет вызывающего её пользователя правами суперпользователя.

CREATE OR REPLACE FUNCTION get\_agent\_orders(agent\_id integer)

RETURNS TABLE("ID\_Order" integer, "Sum" real, "Date" date, "Customer\_Name" varchar)

AS $$

BEGIN

RETURN QUERY

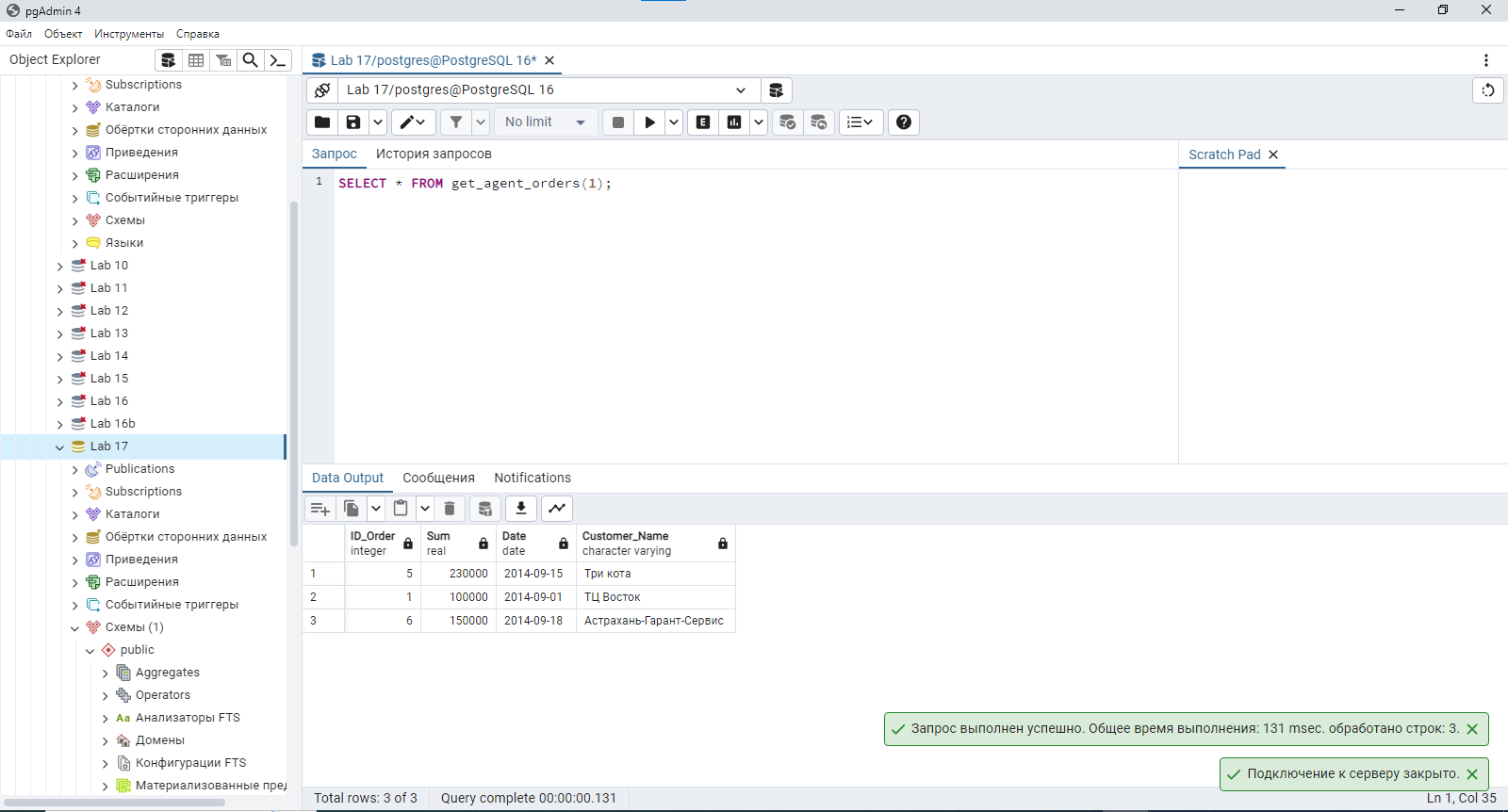
SELECT o."ID\_Order", o."Sum", o."Date", c."Name" as "Customer\_Name"

FROM "Order" o

JOIN "Customer" c ON o."ID\_Customer" = c."ID\_Customer"

WHERE o."ID\_Agent" = agent\_id;

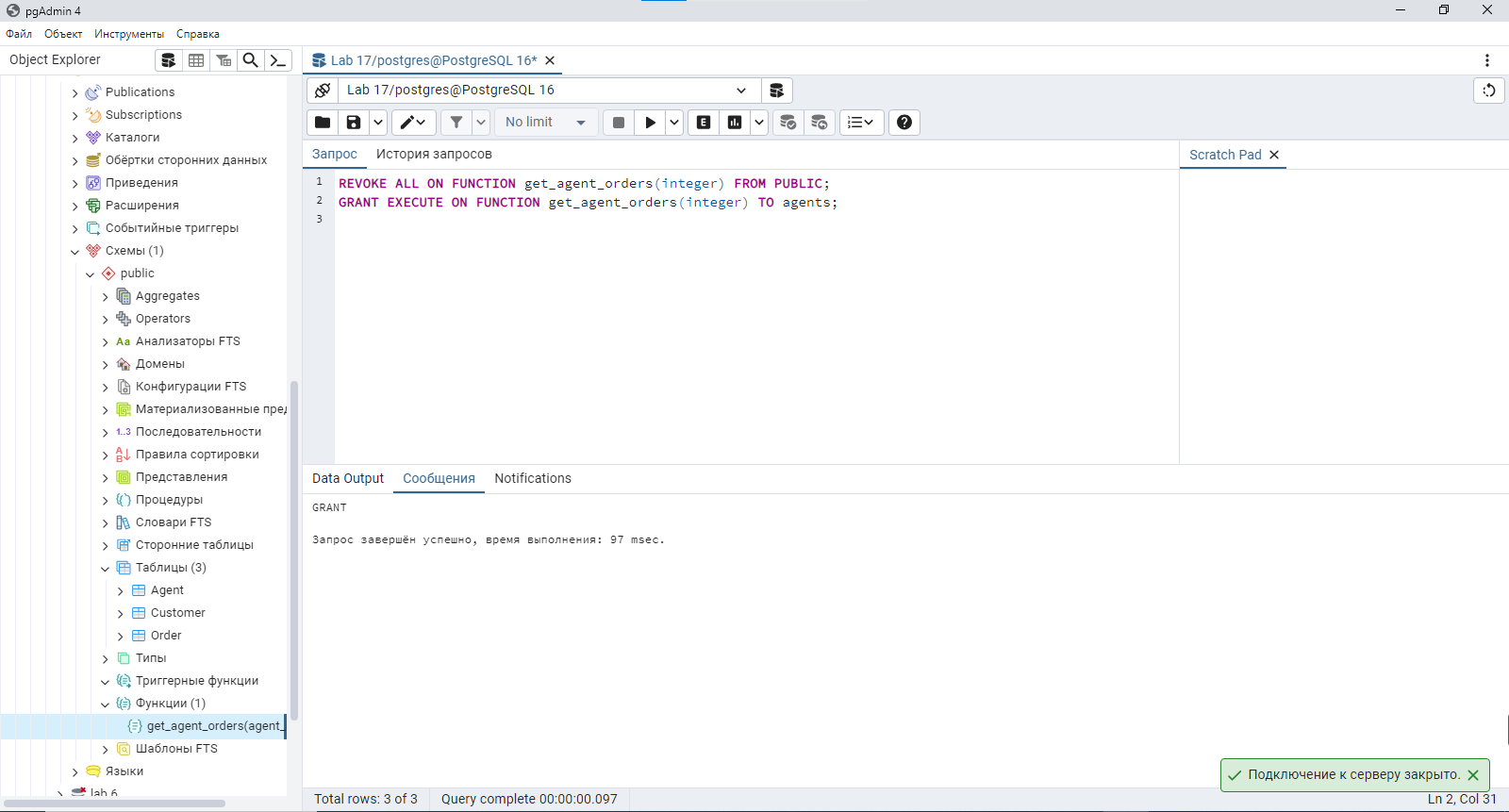
END;

$$ LANGUAGE plpgsql;  


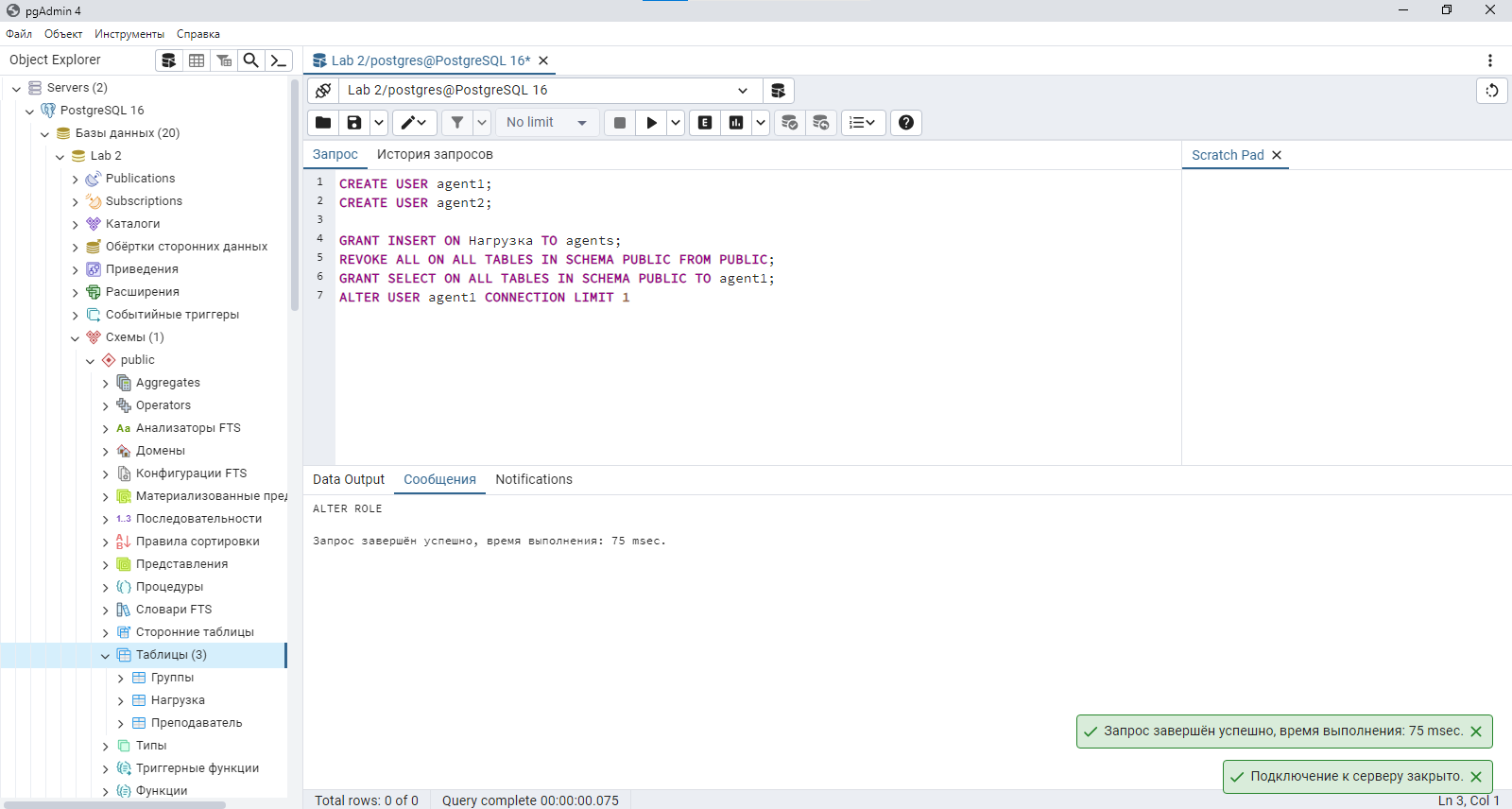
1. Присвойте **только агентам** право на использование функции agent\_orders(id).

REVOKE ALL ON FUNCTION agent\_orders(integer) FROM PUBLIC;

GRANT EXECUTE ON FUNCTION agent\_orders(integer) TO agents;



1. Убедитесь, что функция недоступна никому кроме агентов и, что функция отрабатывает правильно.
2. **Выполните домашнее индивидуальное задание согласно варианту из лабораторной работы №2. Придумайте 2-3 групповые роли для своей предметной области. Роли должны иметь разные права и ограничения на таблицы.**



Выводы: в ходе лабораторной работы я изучил роли пользователей в PostgreSQL.