

***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 № 16**

**«Индексы и транзакции 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**

**Индексы и транзакции в PostgreSQL.**

**А. Индексы в PostgreSQL.**

**Цели лабораторной работы:** научиться создавать индексы для повышения эффективности запросов.

**Часть 1. Краткий теоретический материал по индексам в PostrgeSQL.**

***Создание индексов.***

Индексы нужны для того, чтобы ускорить доступ к данным. Для каждой таблицы доступен только один самый быстрый индекс - первичный ключ. Как правило, это **id** в каждой таблице. Запросы, в условиях которых фильтрация происходит только по **id,** работают быстро, таблица при этом не сканируется целиком.

В таблицу также можно добавить еще индексы для ускорения выполнения запросов. Они могут относиться как к одному, так и к группе атрибутов. Индексы выбирают исходя из информационной потребности пользователей: если в запросах часто встречается какая-либо комбинация условий, то индексы строят именно на эти условия.

***Типы индексов***.

Наиболее популярный тип индекса: ***B-дерево***. Эта структура хорошо подходит для атрибутов, которые можно сравнивать между собой, выполняя операции: <, <=, =, >=, >. Также индексы этого типа поддерживают быструю проверку IS (NOT) NULL, поэтому они хорошо подходят для атрибутов с большим количеством пропусков. В-деревья ускоряют любые операции, связанные со сравнением значений, в том числе сортировку и группировку.

***Синтаксис создания индекса типа В-дерево:***

CREATE INDEX index\_name ON table\_name (column\_name);

Будет создан индекс типа В-дерево ***по умолчанию***.

Второй по популярности тип индекса - ***HASH***, который позволяет быстро проводить сравнения на равенство. Этот тип также подходит для условий типа WHERE column\_name IN ( ... ).

***Синтаксис создания индекса типа HASH:***

CREATE INDEX index\_name ON table\_name **USING HASH** (column\_name);

***Функции в индексах.***

Удобно для строк создавать также индексы, независимые от регистра:

CREATE INDEX index\_name ON table\_name (lower(column\_name));

В этом случае, выбирая WHERE lower(title) = 'the matrix' приведение значений колонки не нужно, и запрос выполнится быстро.

***Индексы на несколько колонок*.**

Такие индексы удобны для часто встречающихся комбинаций условий.

***Синтаксис:***

CREATE INDEX index\_name ON table\_name (column1\_name, column2\_name);

***Уникальный индекс.***

Этот индекс также добавляет проверку целостности при изменении данных.

***Синтаксис:***

CREATE UNIQUE INDEX index\_name on table\_name (column\_name);

***Условные индексы.***

Можно создать специализированный индекс на конкретные условия, например:

CREATE INDEX index\_name on title (kind\_id, production\_year) where production\_year>2000 and kind\_id=1);

Этот индекс позволит выбирать фильмы, выпущенные после 2000 года, при этом для записей, не удовлетворяющих условию, все останется по-прежнему (как скорость выборки, так и скорость изменения данных).

***Удаление индекса.***

Для удаления индексов из таблицы используется команда DROP INDEX. ***Синтаксис команды:***

DROP INDEX: DROP INDEX индекс [. ...]

В качестве параметра команде передается имя удаляемого индекса. Допускается одновременное удаление нескольких индексов, перечисленных через запятую.

**Часть 2. Выполнение лабораторной работы.**

1. Предметная область соответствует варианту задания **лабораторной работы № 2** (стр. 10-21).

CREATE TABLE Группы (  
Номер\_группы INT PRIMARY KEY,  
Специальность VARCHAR(50),  
Отделение VARCHAR(50),  
Количество\_студентов INT  
);  
  
CREATE TABLE Преподаватели (  
Код\_преподавателя INT PRIMARY KEY,  
Фамилия VARCHAR(50),  
Имя VARCHAR(50),  
Отчество VARCHAR(50),  
Телефон VARCHAR(15),  
Стаж INT  
);  
  
CREATE TABLE Нагрузка (  
Код\_преподавателя INT,  
Номер\_группы INT,  
Количество\_часов INT,  
Предмет VARCHAR(50),  
Тип\_занятия VARCHAR(50),  
Оплата DECIMAL(10,2),  
PRIMARY KEY (Код\_преподавателя, Номер\_группы),  
FOREIGN KEY (Код\_преподавателя) REFERENCES Преподаватели (Код\_преподавателя),  
FOREIGN KEY (Номер\_группы) REFERENCES Группы (Номер\_группы)  
);

1. В каждом варианте представлены три таблицы: два справочника и одна таблица дочерняя (сводная). Например, в варианте №30 «**Интернет-магазин**» справочниками являются таблицы **Товары** и **Клиенты**, а дочерней (сводной) таблицей является **Продажи**.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Online C++ Compiler.

Code, Compile, Run and Debug C++ program online.

Write your code in this editor and press "Run" button to compile and execute it.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

int main()

{

//INSERT INTO Группы (Номер\_группы, Специальность, Отделение, Количество\_студентов)

cout << "INSERT INTO Преподаватели (Код\_преподавателя, Фамилия, Имя, Отчество, Телефон, Стаж) VALUES" << endl;

for(int i = 0; i<1000; i++){

string str = "";

for(int k = 0; k < 20; k++){

str = str + (char)(65+rand()%40);

}

cout << "("<<i+1 <<",'" <<str<<"','" <<str<<"','" <<str<<"'," <<rand()<<"," <<rand()%70<< "),"<<endl;

}

cout << ";"<< endl;

cout << "INSERT INTO Группы (Номер\_группы, Специальность, Отделение, Количество\_студентов) VALUES" << endl;

for(int i = 0; i<1000; i++){

string str = "";

for(int k = 0; k < 20; k++){

str = str + (char)(65+rand()%40);

}

cout << "("<<i+1 <<",'" <<str<<"','" <<str<<"'," <<rand()%70<< "),"<<endl;

}

cout << ";"<< endl;

cout << "INSERT INTO Нагрузка (Код\_преподавателя, Номер\_группы, Количество\_часов, Предмет, Тип\_занятия, Оплата) VALUES" << endl;

for(int it = 0; it<10; it++){

for(int i = 0; i<1000; i++){

string str = "";

for(int k = 0; k < 20; k++){

str = str + (char)(65+rand()%40);

}

cout << "("<<i+1 <<"," <<i+1<<"," <<rand()%10<<",'" <<str<<"','" <<str<<"'," <<rand()%70<< "),"<<endl;

}

}

cout << ";"<< endl;

return 0;

}

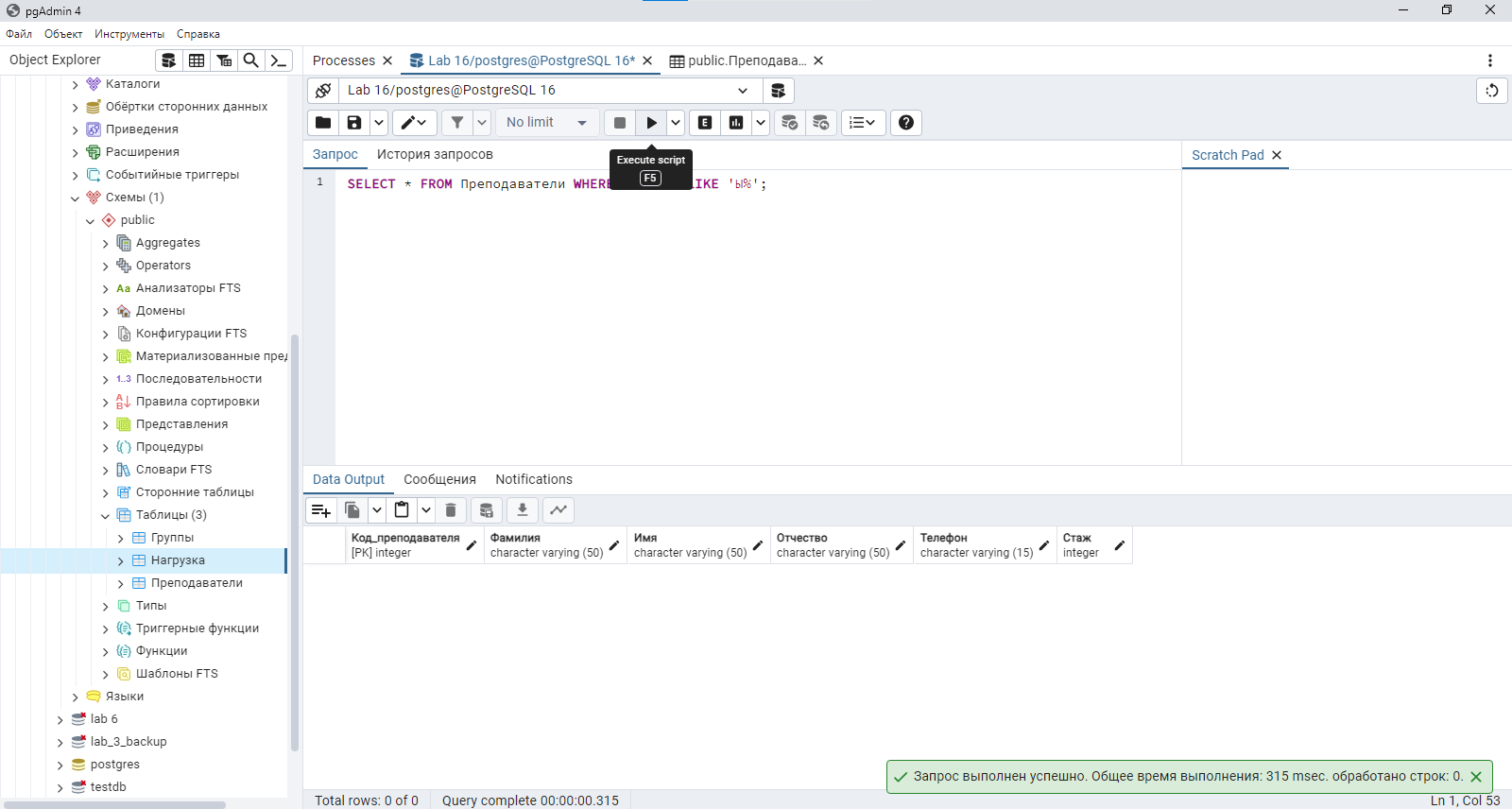
1. Работы индексов лучше анализировать на ***большом объеме данных***. Для этого с помощью пользовательских функций вставить по **1000** строк в таблицы-справочники, ***сгенерировав случайным образом*** значения полей. Причем текстовые поля – случайные буквы русского алфавита (заглавные буквы) длиной **20** символов.

В дочернюю (сводную) таблицу вставить **10 000** строк со случайными данными (***будьте внимательны к генерации внешних ключей!!!***)

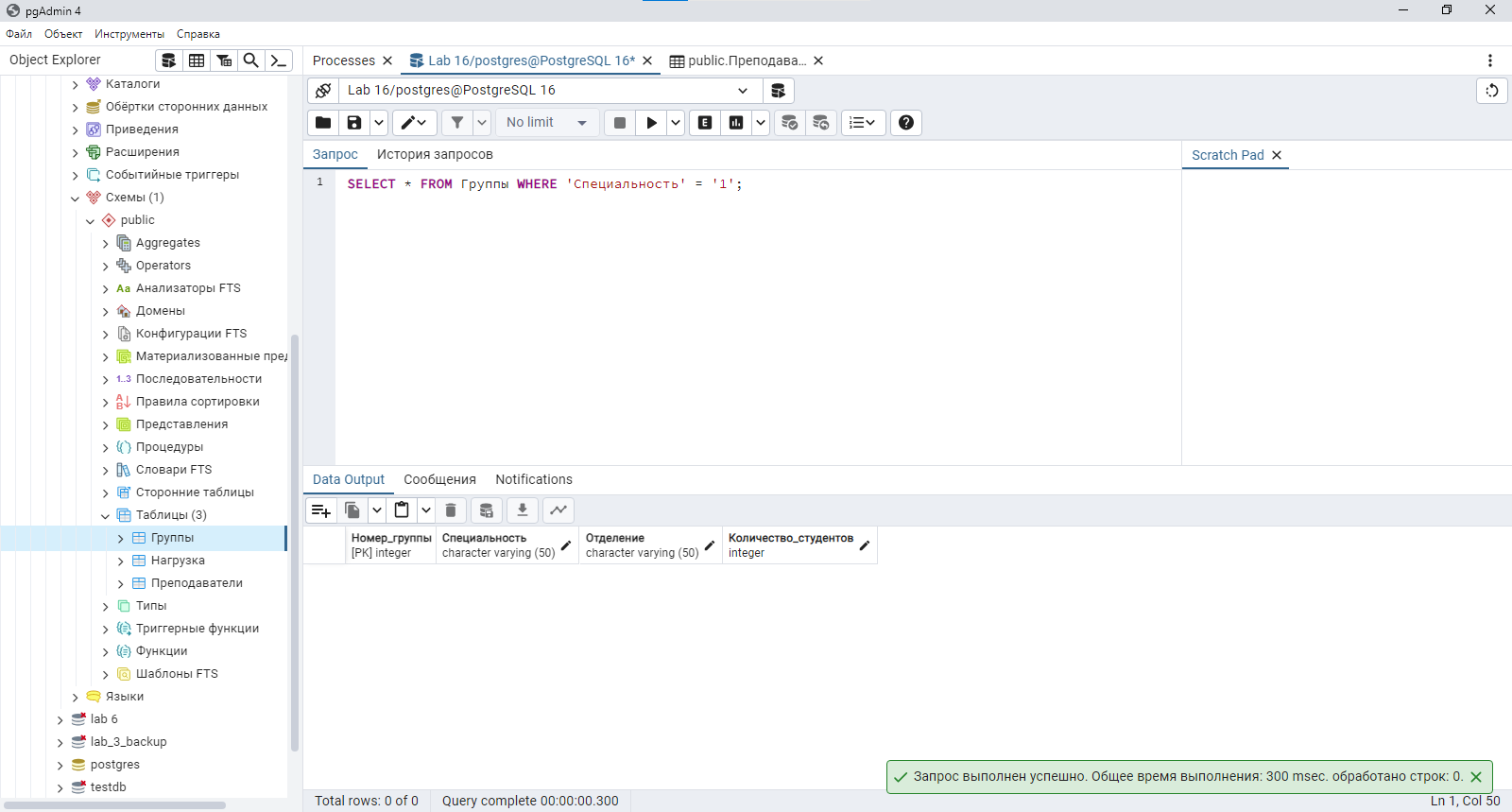
4. Выполнить две команды поиска данных в таблицах-справочниках по текстовым полям, и одну команду поиска по трем таблицам.

***Пример***:

*Запрос 1:* SELECT \* FROM “Преподаватели” WHERE “Фамилия” LIKE ‘Ы%’;



*Запрос 2:* SELECT \* FROM “Группы” WHERE “Специальность” = ‘1’;

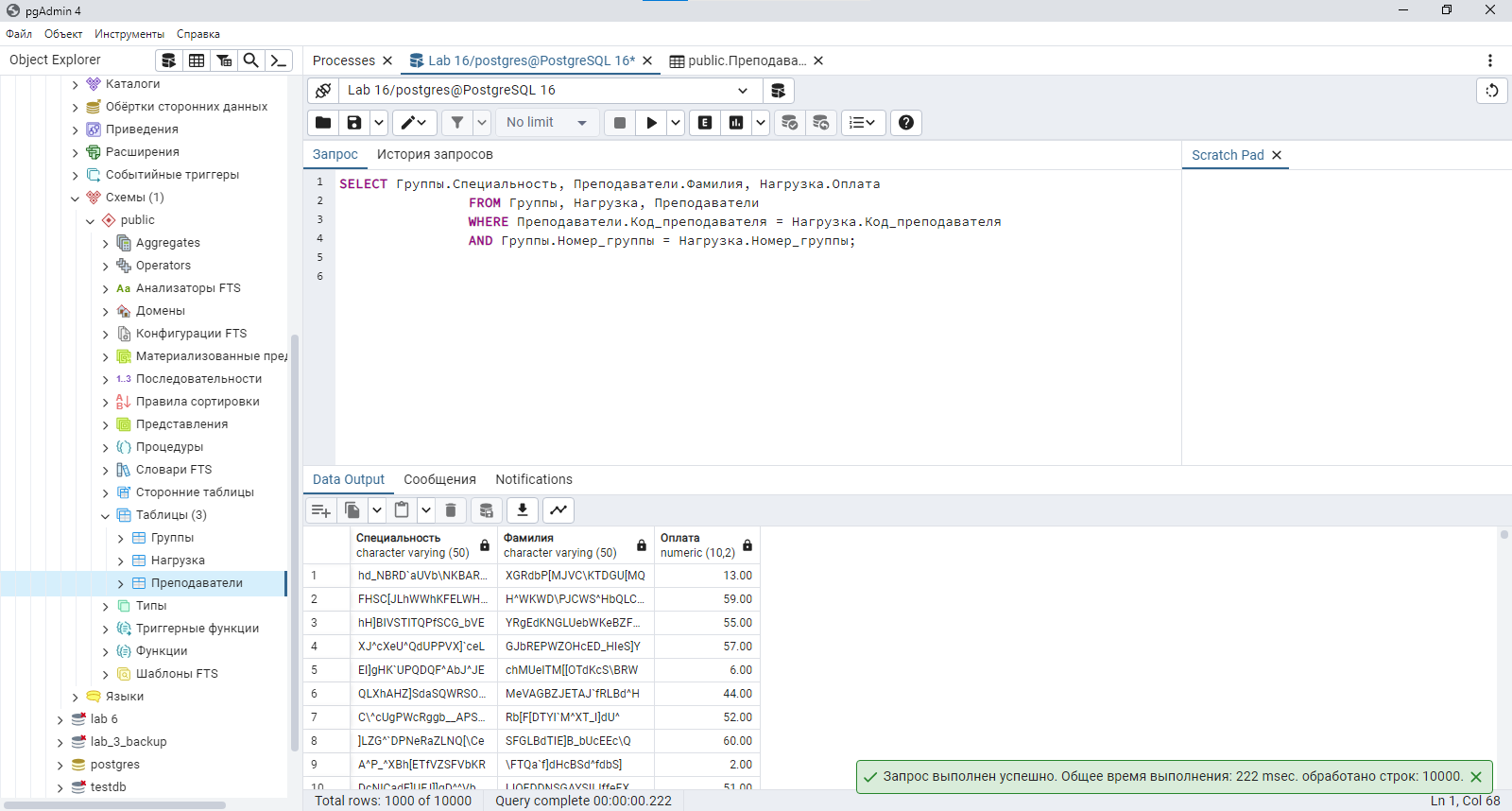


*Запрос 3:* SELECT Группы.Специальность, Преподаватели.Фамилия, Нагрузка.Оплата

FROM Группы, Нагрузка, Преподаватели

WHERE Преподаватели.Код\_преподавателя = Нагрузка.Код\_преподавателя

AND Группы.Номер\_группы = Нагрузка.Номер\_группы;



1. Получить время выполнения каждого запроса и занести в таблицу.
2. Создать индексы типа ***В-дерево*** на текстовые поля таблиц-справочников.

Например, создать индекс на поле ***Название*** в таблице ***Товар***, на поле ***Фамилия*** в таблице ***Клиент***.

CREATE INDEX Индекс\_Специальности

ON Группы (Специальность);

CREATE INDEX Индекс\_Отделения

ON Группы (Отделение);

CREATE INDEX Индекс\_Фамилии

ON Преподаватели (Фамилия);

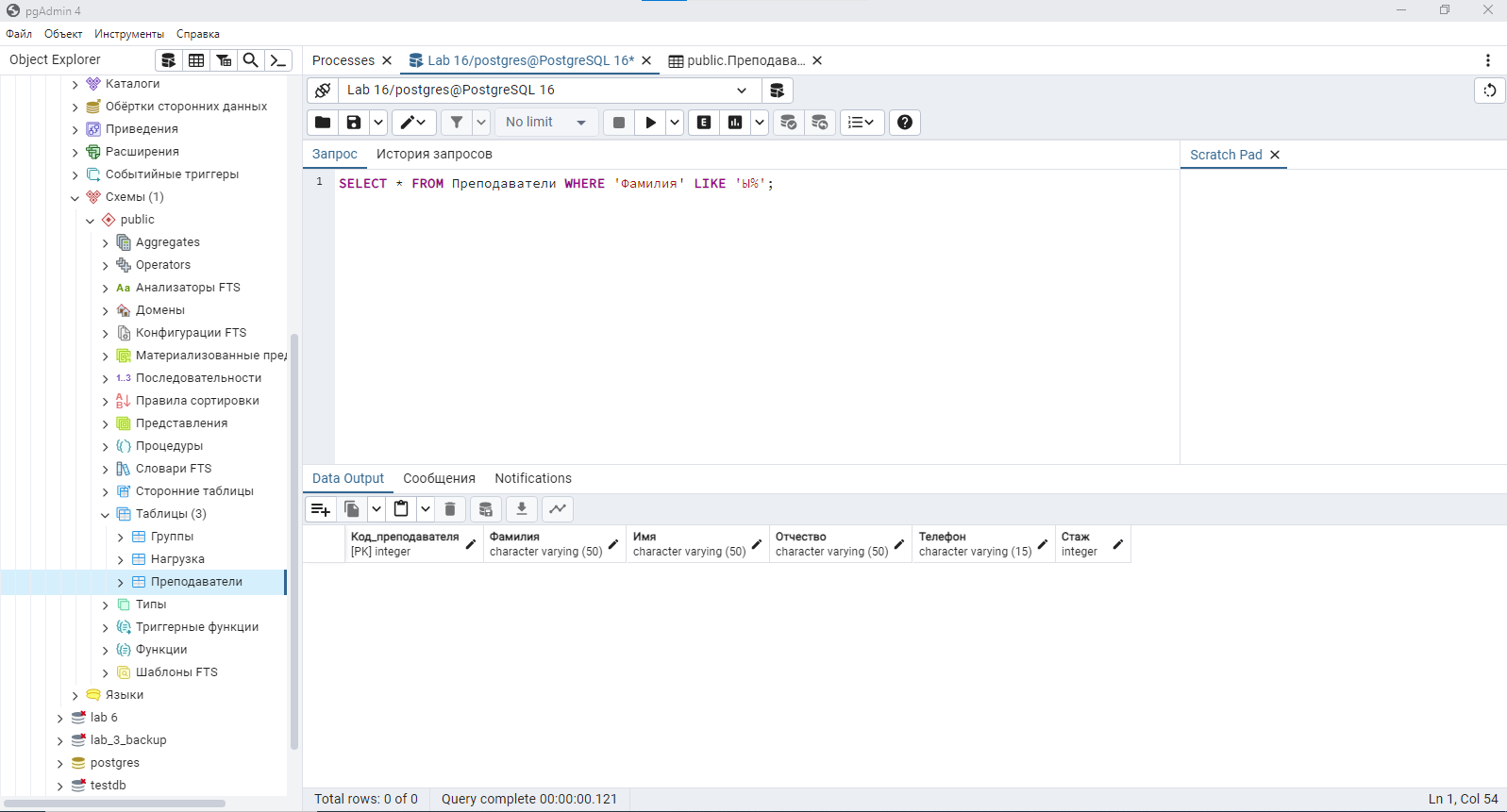
CREATE INDEX Индекс\_Имени

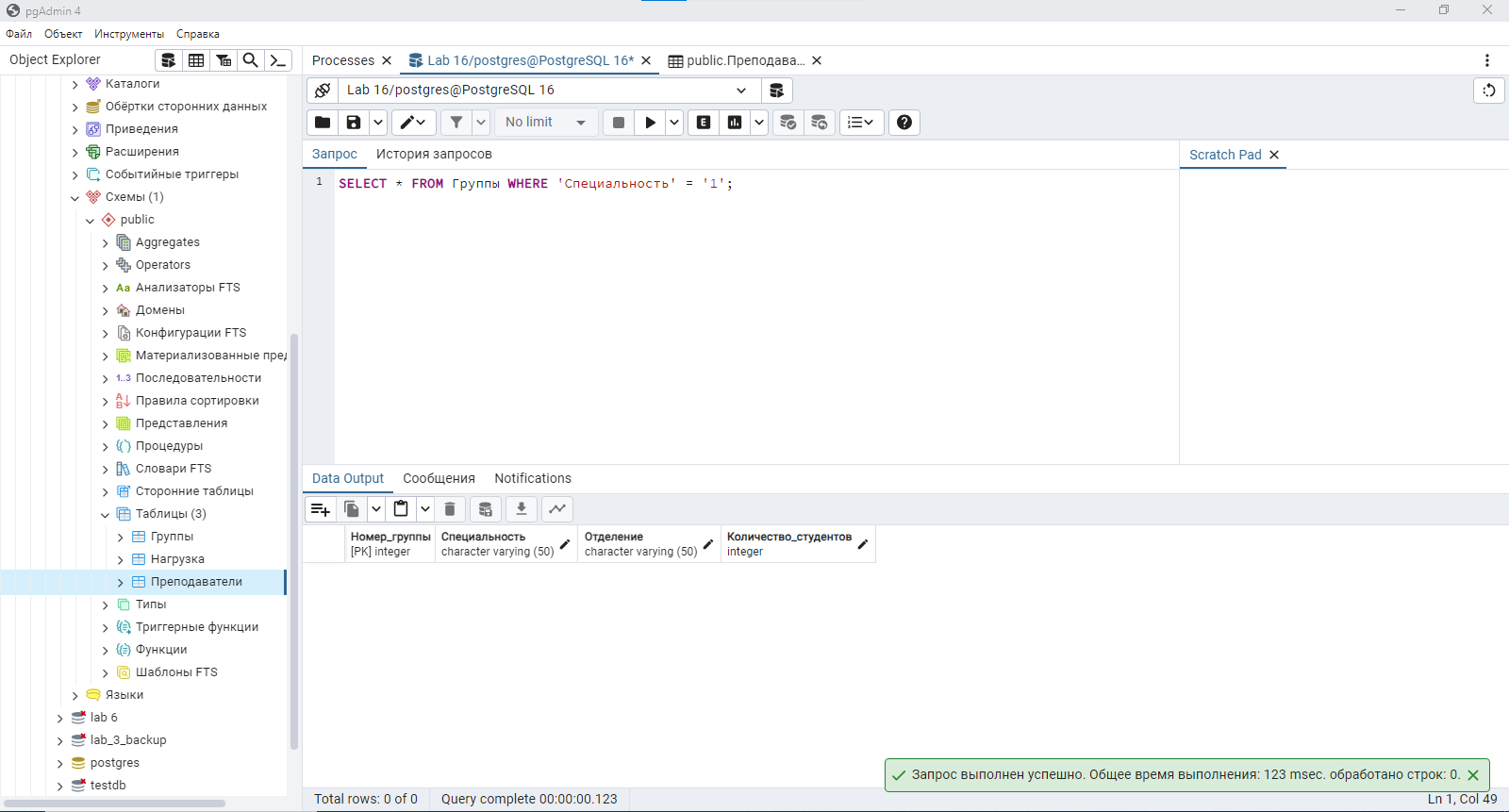
ON Преподаватели (Имя);

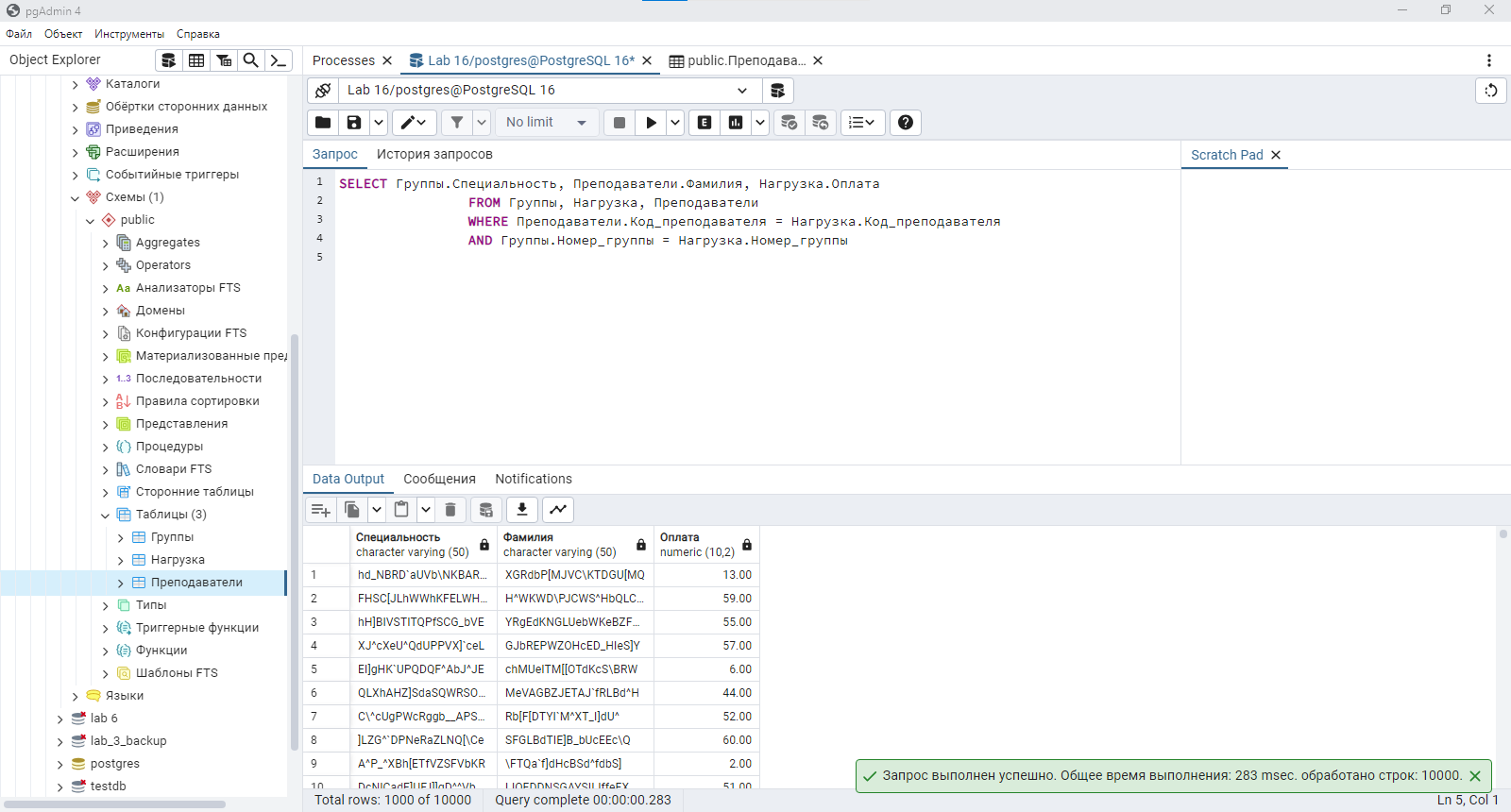
CREATE INDEX Индекс\_Отчества

ON Преподаватели (Отчество);

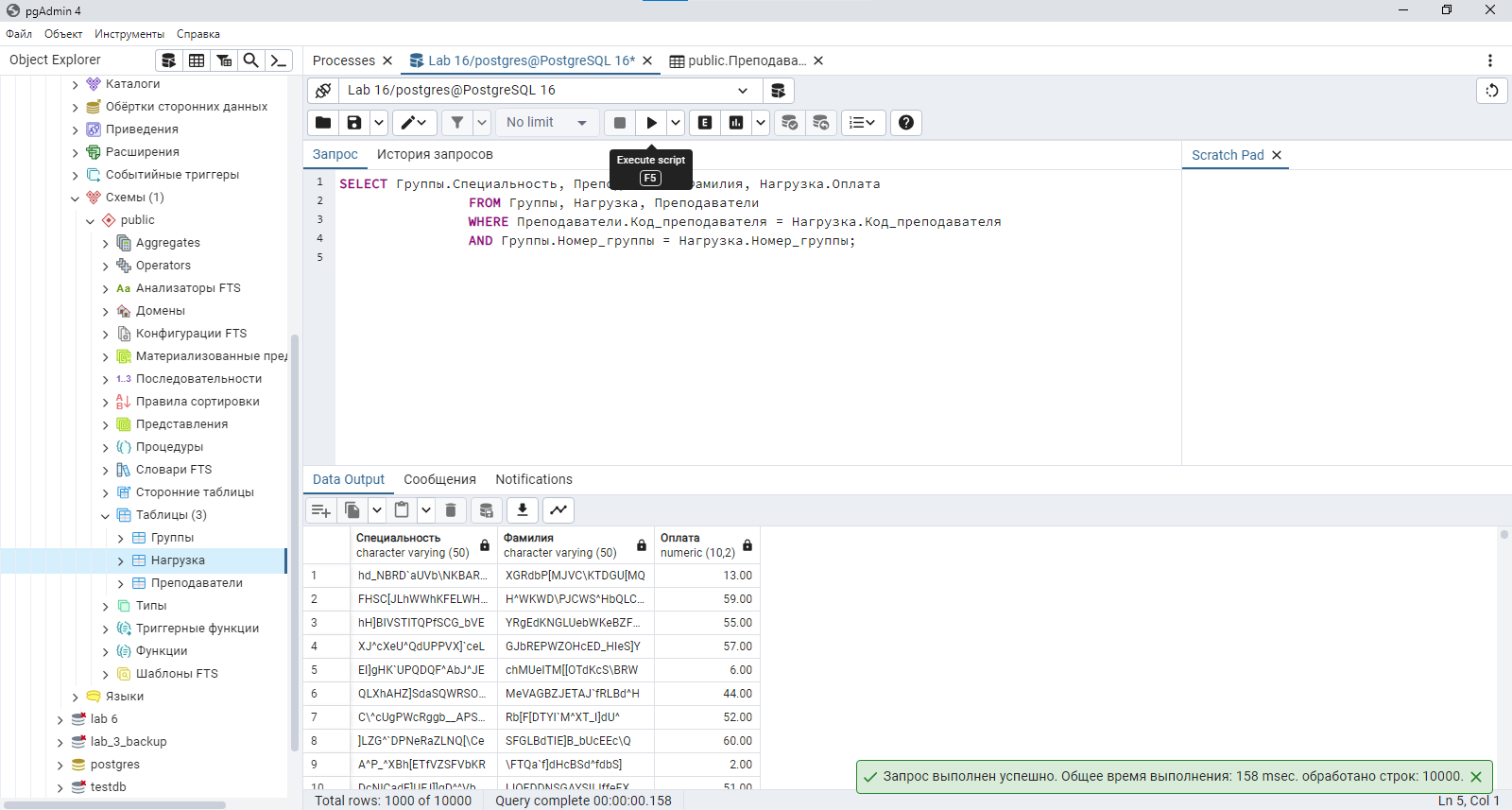
1. Выполнить три запроса (см. п. 4).
2. Получить время выполнения каждого запроса и занести в таблицу.







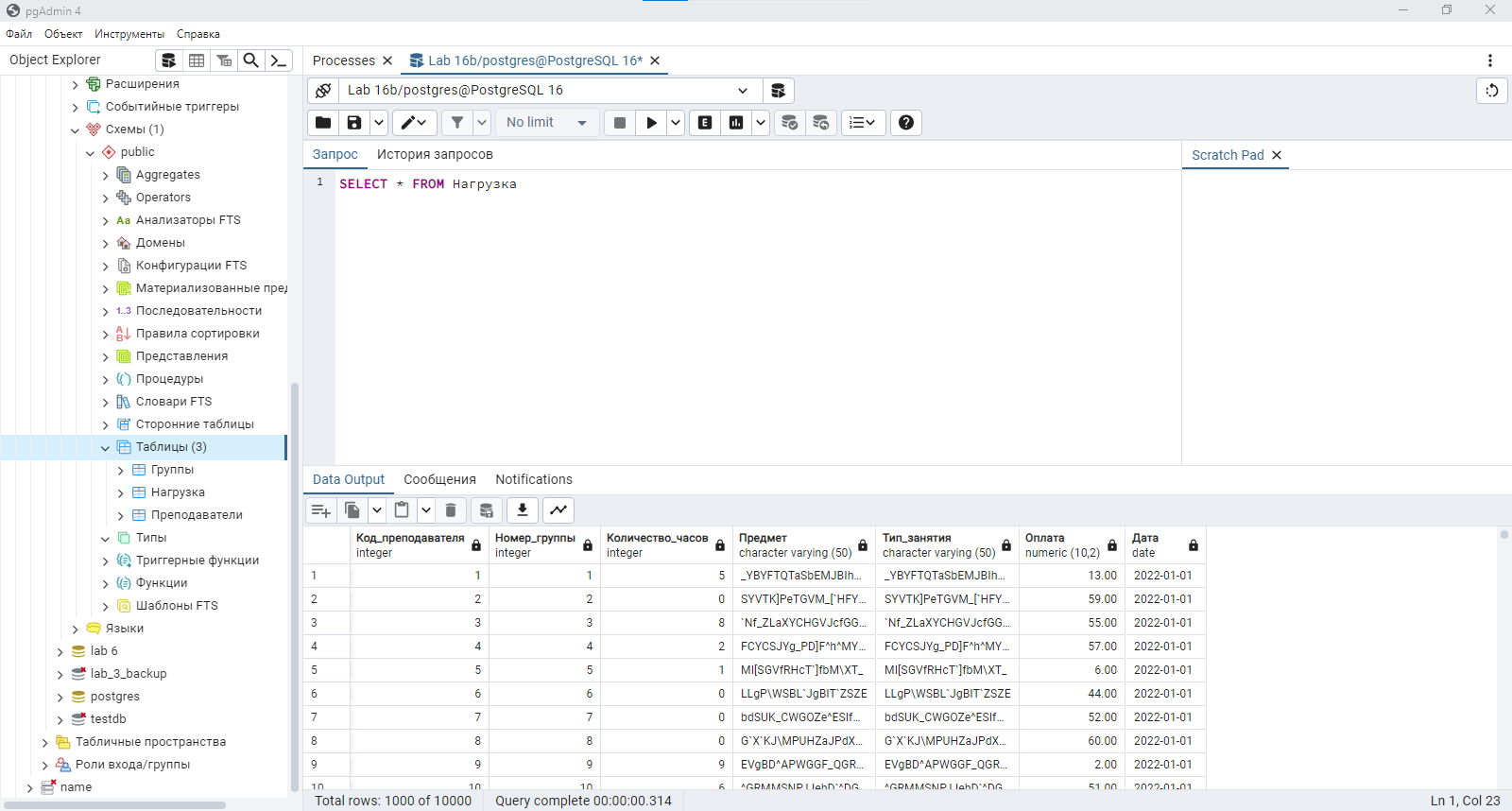
1. Создать индекс типа **HASH** на поле даты, например, поле ***Дата\_продажи***.

****

1. Выполнить запрос 3 (см. п. 4).
2. Сравнить время выполнения запроса без использования индекса и с использованием индекса.
3. Сформулировать выводы.
4. Представить отчет о выполнении лабораторной работы.

**Б. Транзакции в PostgreSQL**

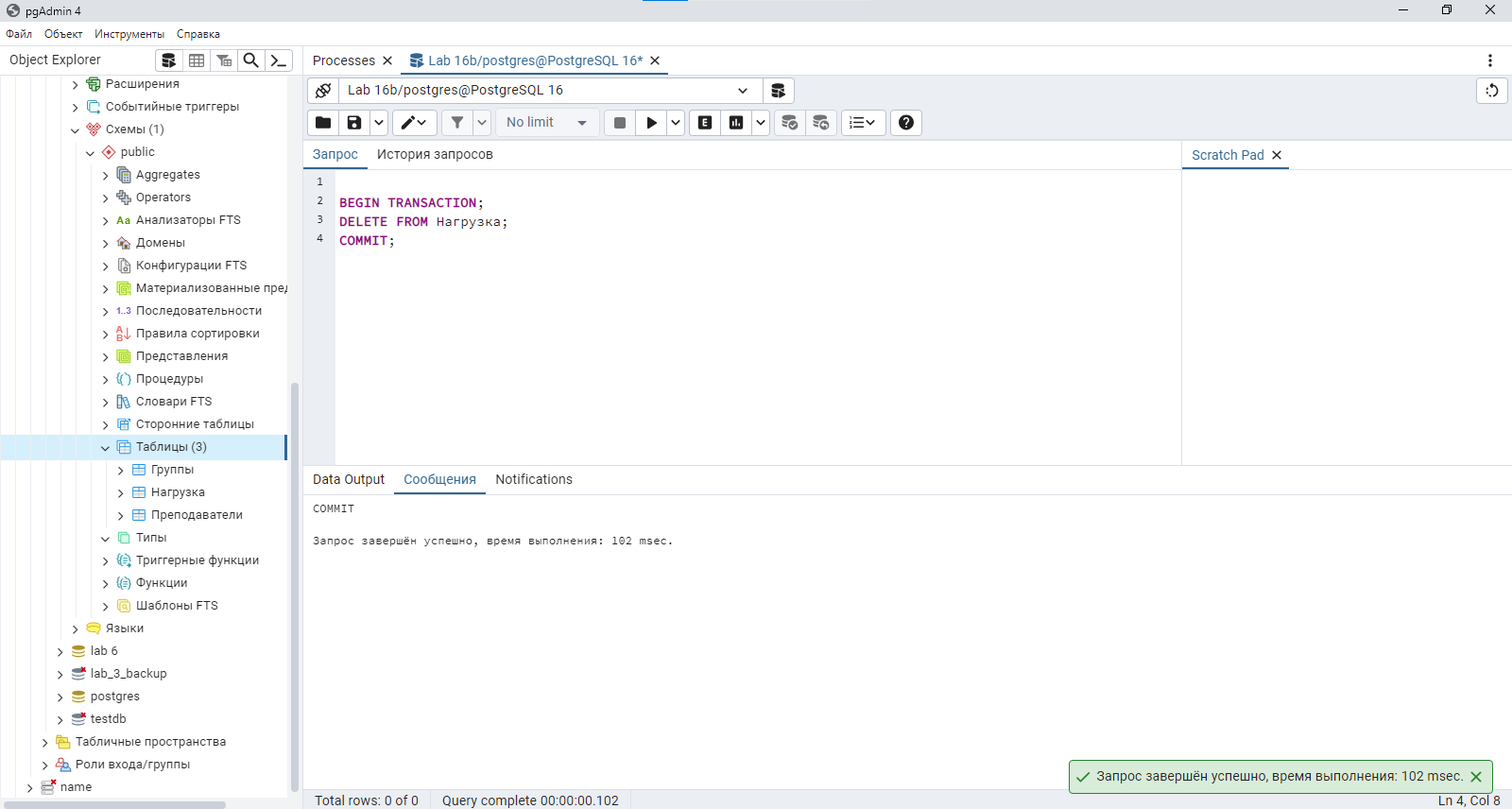
1. Создайте резервную копию базы данных (согласно своему варианту задания № 2).
2. Выведите командой SELECT все записи из дочерней таблицы.



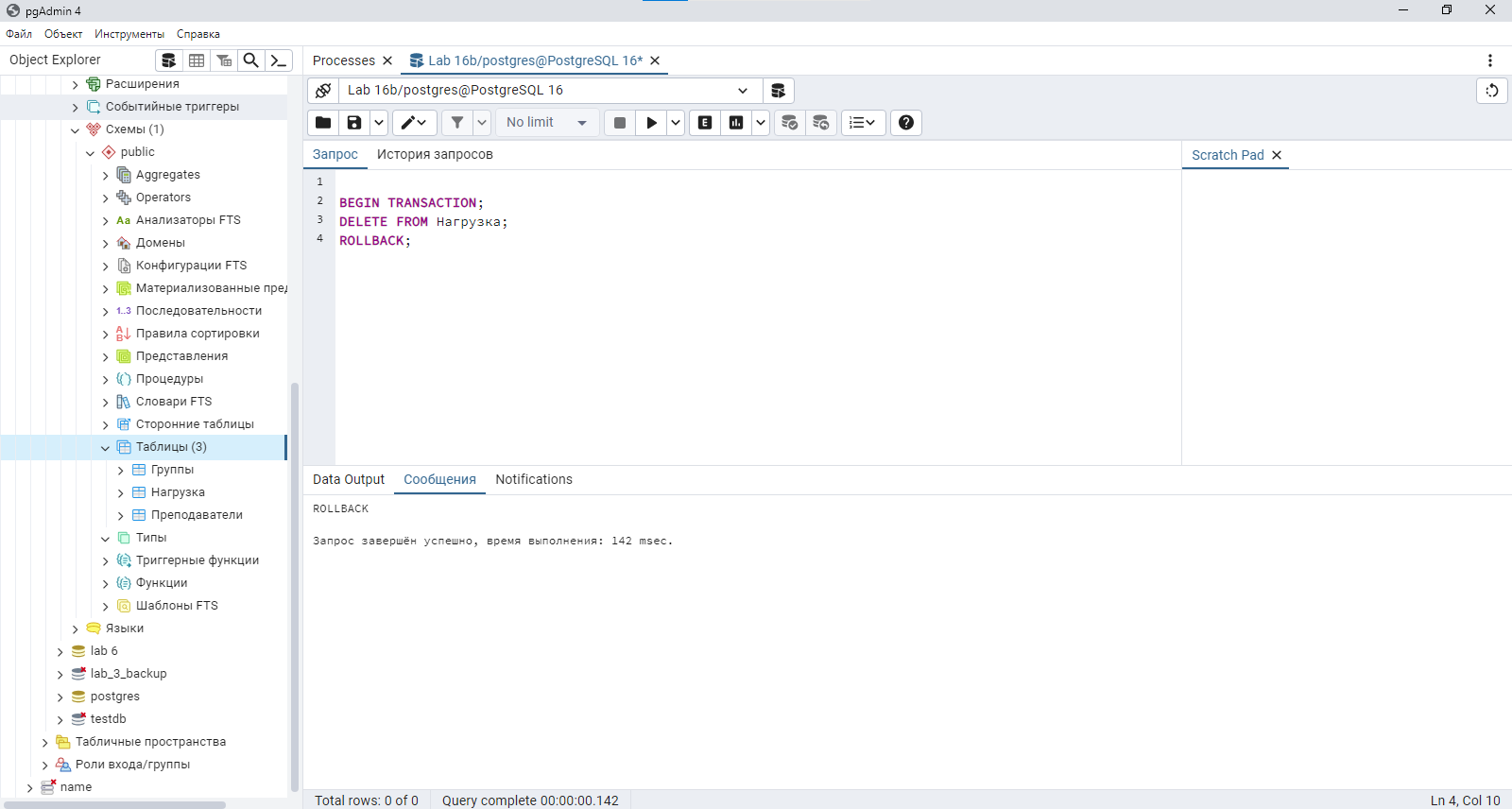
1. Создайте транзакцию, в которой удаляются все строки из дочерней таблицы.

BEGIN TRANSACTION;

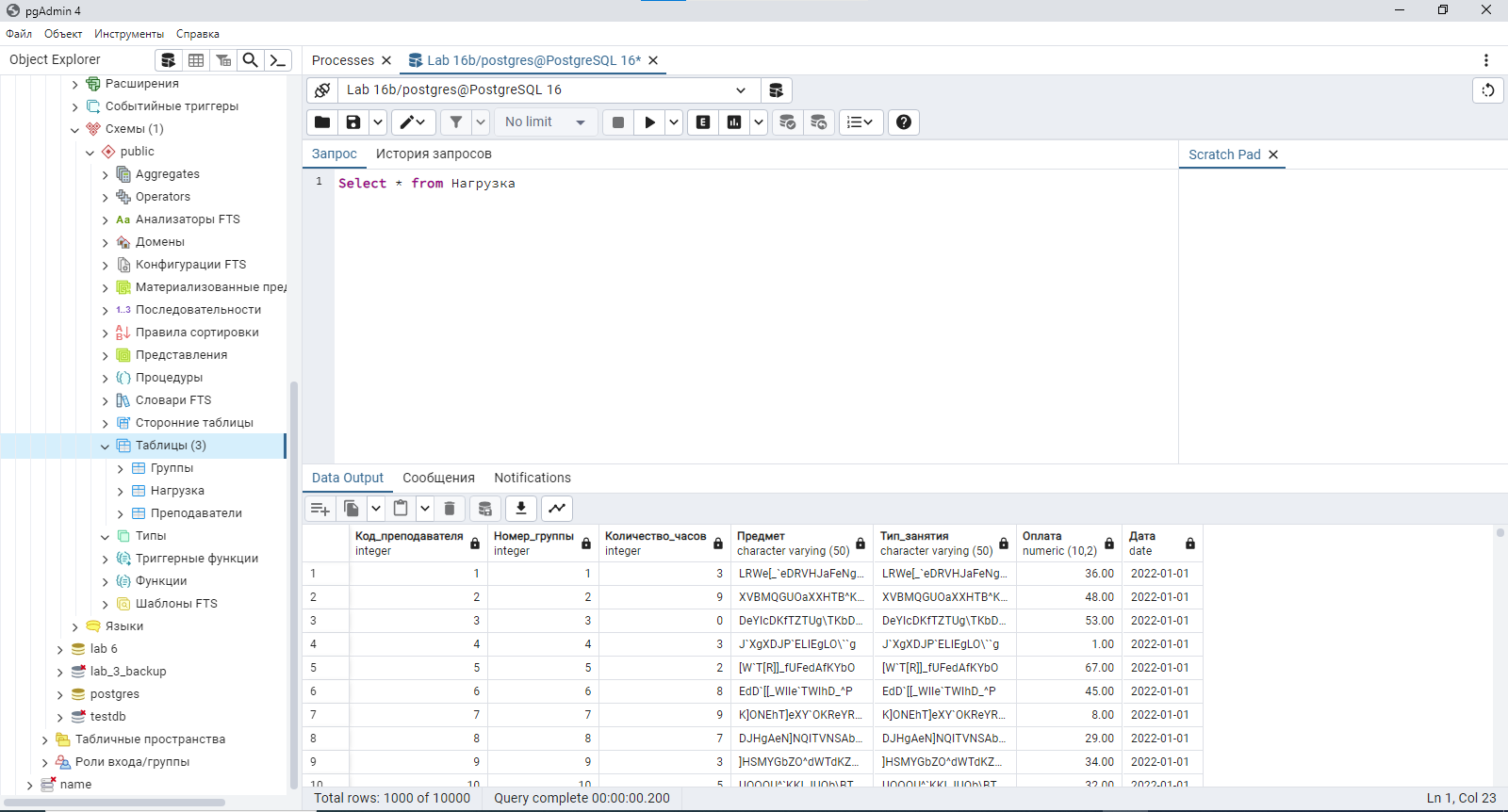
DELETE FROM Нагрузка;



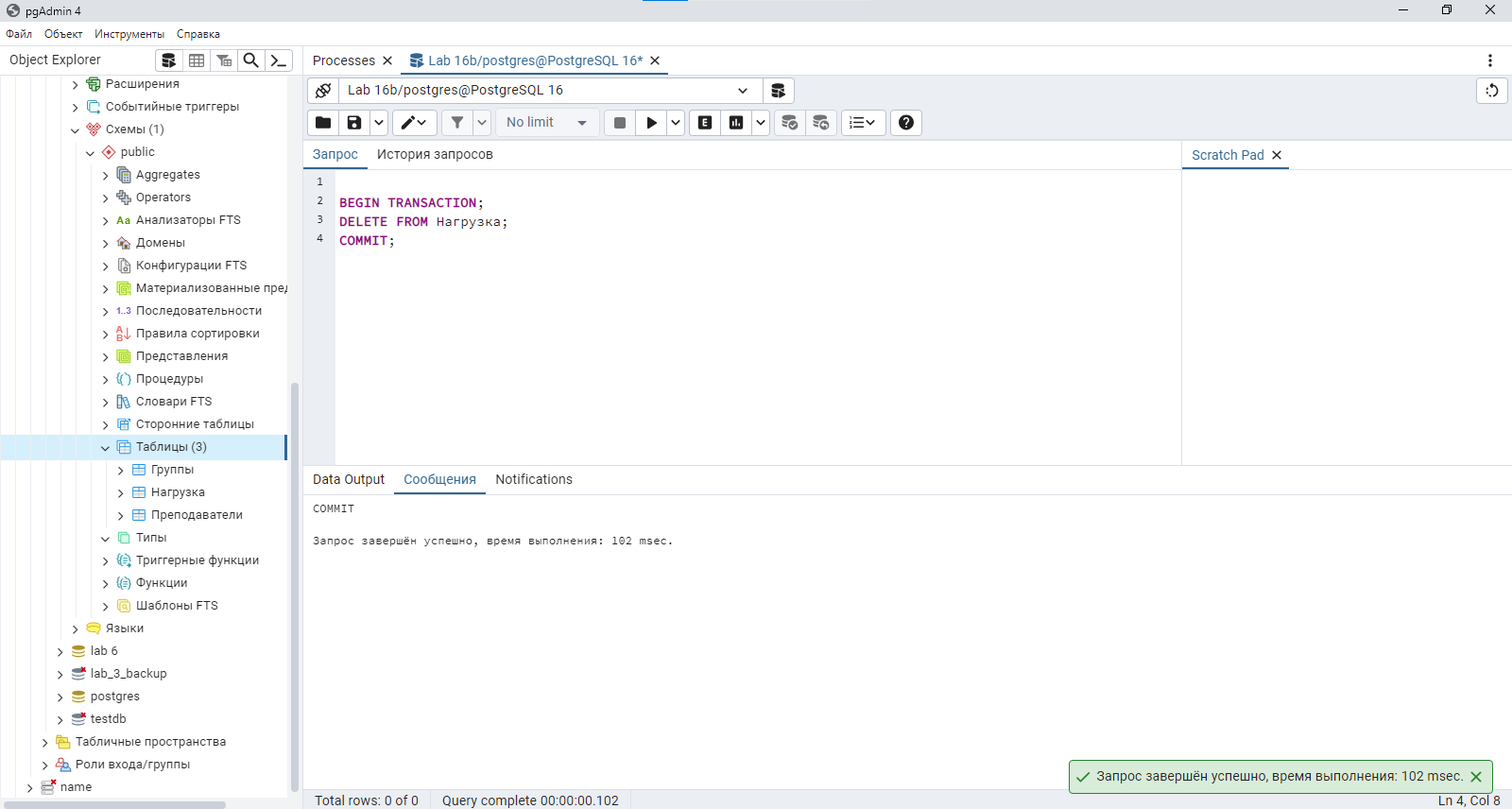
1. Убедитесь в том, что все записи удалены.
2. Выполните откат изменений командой ROLLBACK.



1. Убедитесь в том, что все записи таблицы восстановлены.



1. Зафиксируйте изменения в базе данных командой COMMIT.



1. Создайте тестовую таблицу с тремя целочисленными полями типа bigint.
2. Вставьте в тестовую таблицу 5 строк.
3. Выведите все записи из тестовой таблицы.

-- Создание тестовой таблицы

CREATE TABLE test\_table (

id SERIAL PRIMARY KEY,

number1 BIGINT,

number2 BIGINT,

number3 BIGINT

);

-- Вставка 5 строк в тестовую таблицу

INSERT INTO test\_table (number1, number2, number3) VALUES (10, 20, 30);

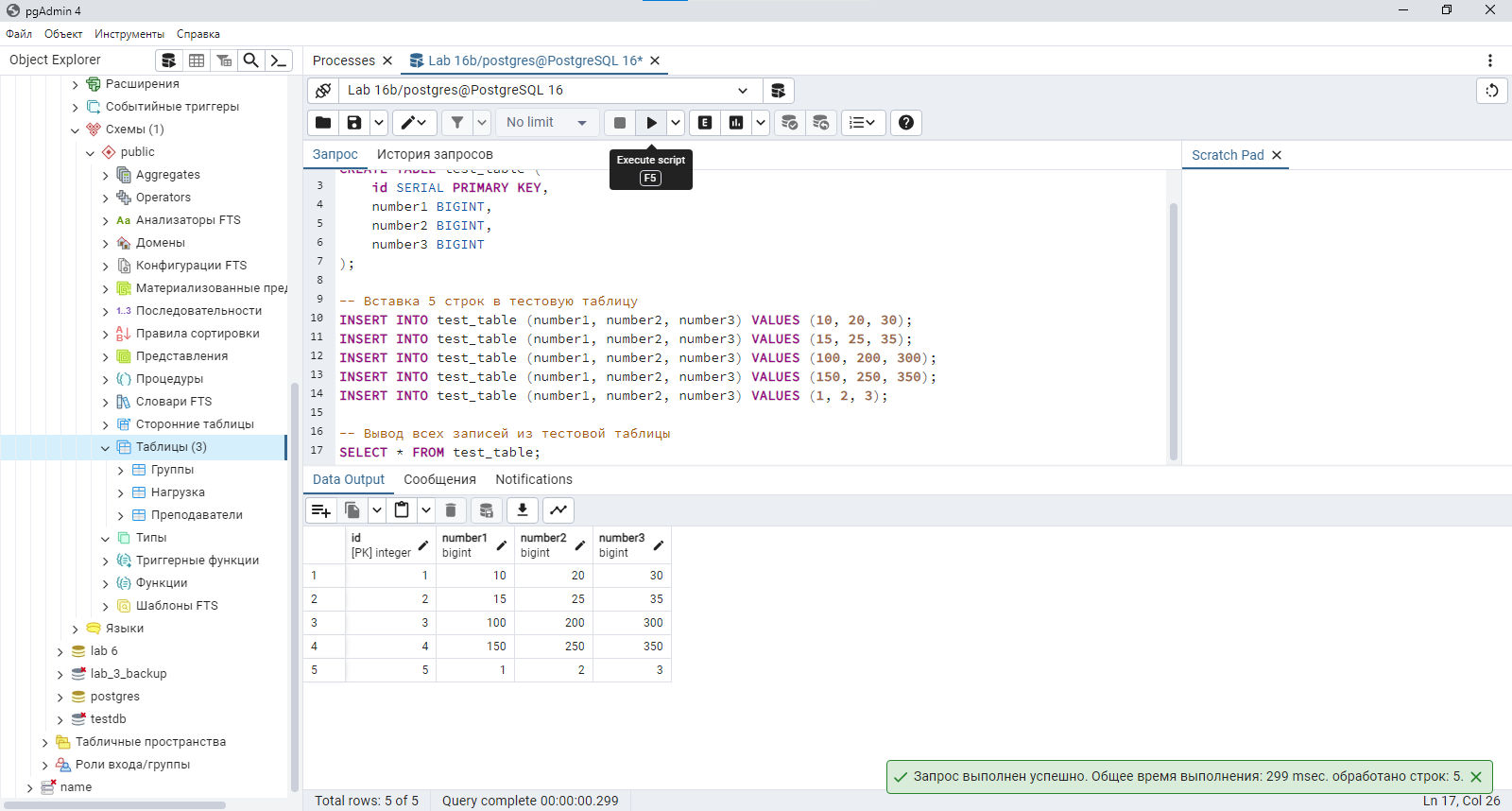
INSERT INTO test\_table (number1, number2, number3) VALUES (15, 25, 35);

INSERT INTO test\_table (number1, number2, number3) VALUES (100, 200, 300);

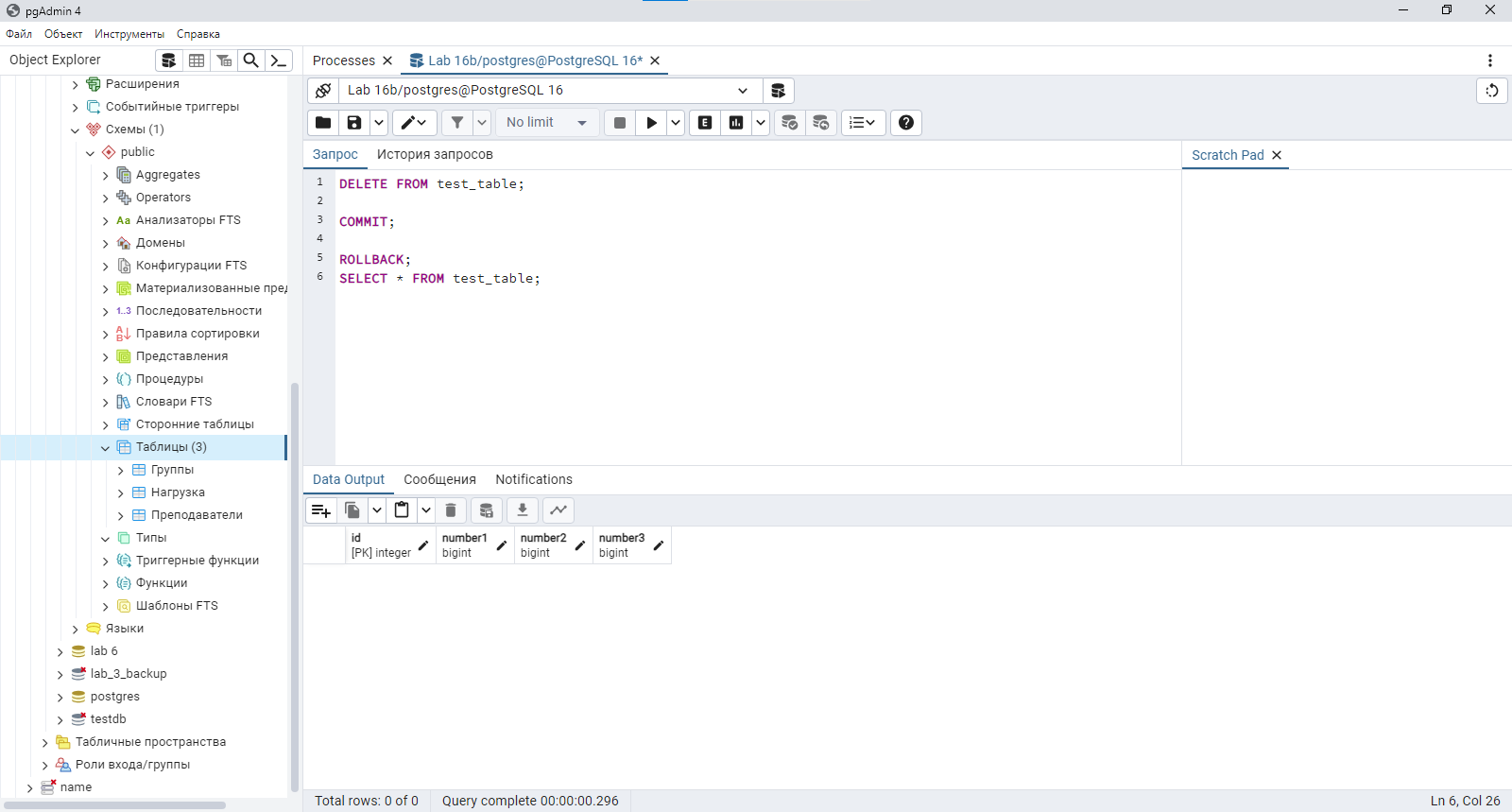
INSERT INTO test\_table (number1, number2, number3) VALUES (150, 250, 350);

INSERT INTO test\_table (number1, number2, number3) VALUES (1, 2, 3);

-- Вывод всех записей из тестовой таблицы

SELECT \* FROM test\_table;  
  


1. Удалите все записи из тестовой таблицы командой DELETE.
2. Зафиксируйте изменения командой COMMIT.
3. Попробуйте отметить последние изменения командой ROLLBACK.
4. Проанализируйте результат выполнения последней команды.



1. Вставьте в тестовую таблицу одну строку со значениями (1, 1, 1) *или (1,1,1,1).*

INSERT INTO test\_table (number1, number2, number3) VALUES (1, 1, 1);

1. Создайте функцию Count, которая в цикле увеличивает (или уменьшает) значение всех полей тестовой таблицы на 1 (в зависимости от значения передаваемого параметра)

CREATE OR REPLACE FUNCTION update\_values(increment\_value INT)

RETURNS VOID AS $$

DECLARE

row\_record RECORD;

BEGIN

FOR row\_record IN SELECT \* FROM test\_table

LOOP

UPDATE test\_table

SET

number1 = number1 + increment\_value,

number2 = number2 + increment\_value,

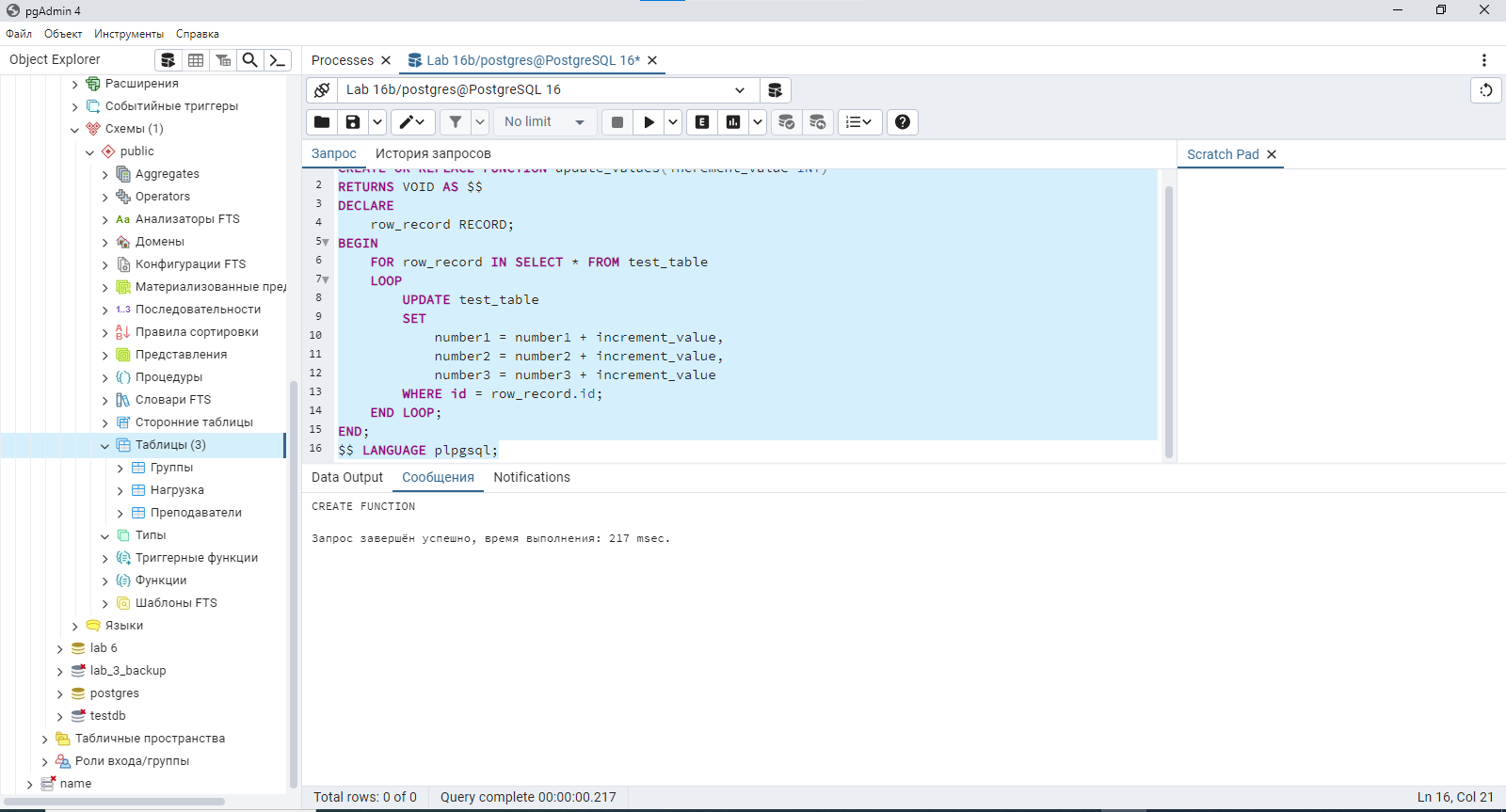
number3 = number3 + increment\_value

WHERE id = row\_record.id;

END LOOP;

END;

$$ LANGUAGE plpgsql;



1. Создайте две транзакционные функции: Count1 (эта функция вызывает функцию Count с параметром для инкремента) и Count2 (эта функция вызывает функцию Count с параметром для декремента).

Вызовите на выполнение одновременно двух функций подряд (сначала следует вызов Count1, затем Count2).

CREATE OR REPLACE FUNCTION Count1(increment\_value INT)

RETURNS VOID AS $$

BEGIN

PERFORM update\_values(increment\_value);

END;

$$ LANGUAGE plpgsql;

-- Создаем функцию Count2, которая вызывает Count1 с параметром для декремента

CREATE OR REPLACE FUNCTION Count2(decrement\_value INT)

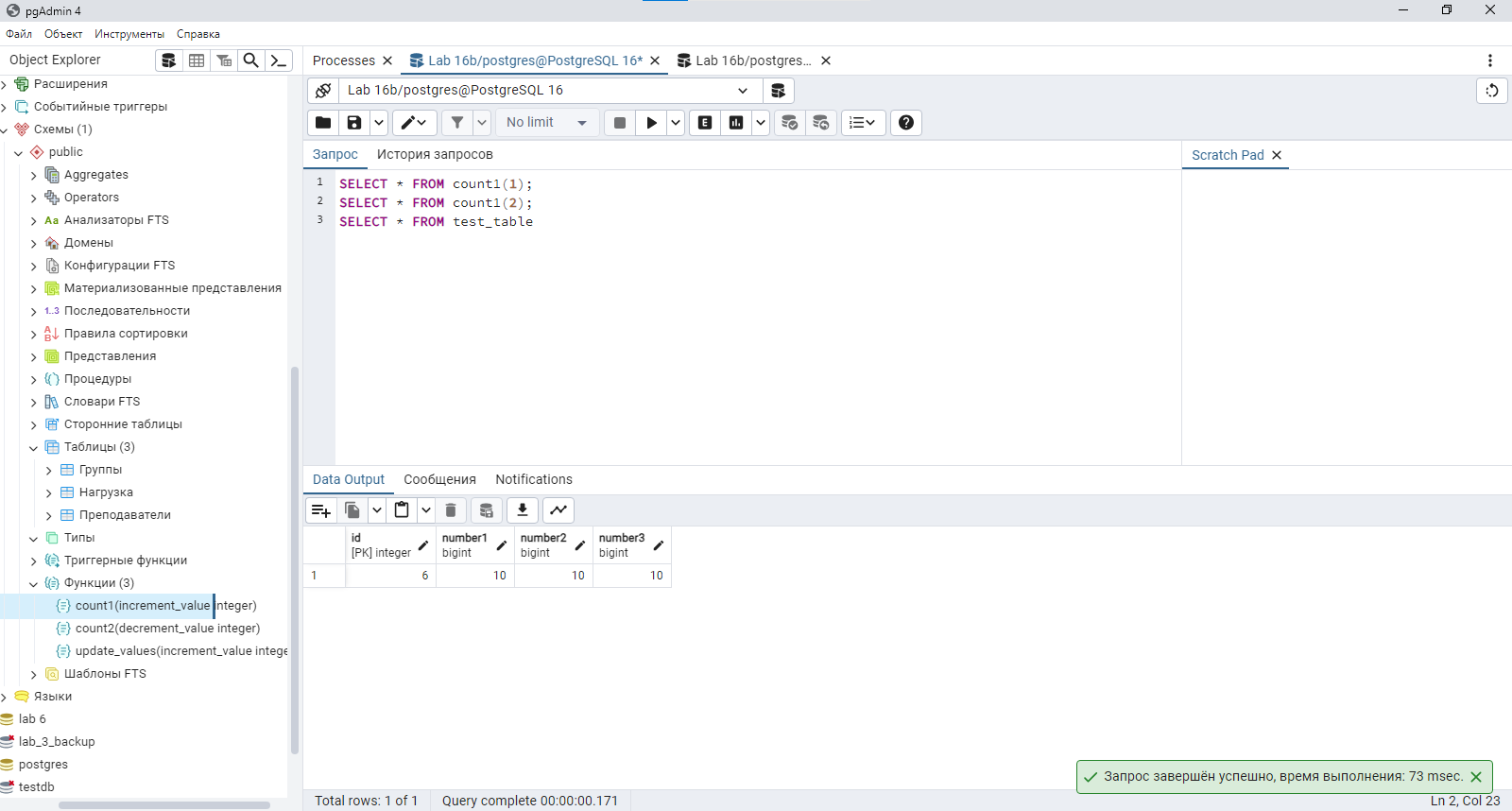
RETURNS VOID AS $$

BEGIN

PERFORM Count1(-decrement\_value);

END;

$$ LANGUAGE plpgsql;



1. Проанализируйте результаты работы функций над тестовой таблицей.
2. Удалите все строки из тестовой таблицы.
3. Вставьте в тестовую таблицу одну строку со значениями (1, 1, 1).
4. Вызовите на выполнение одновременно двух функций подряд (но теперь сначала следует вызов Count2, затем Count1). P.S. Выводит одно и то же)
5. Проанализируйте результаты работы функций над тестовой таблицей.
6. **Дополнительное необязательное задание** (***для повышения оценки по дисциплине)***:

Придумайте примеры (напишите код запросов), демонстрирующие побочные эффекты параллелизма («грязные» операции чтения, неповторяющееся чтение, фантомные операции чтения) **на своей базы данных из предметной области задания (лабораторная работа № 2).**

Выводы: в ходе лабораторной работы я научился создавать индексы для повышения эффективности запросов.