

***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 № 9**

**«PostgreSQL. Основы языка определения данных.**

**Значения по умолчанию и ограничения целостности. Часть 1.**»

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**

**Цель работы**: Получить практические навыки в использовании команд языка определения данных DDL в СУБД PostgreSQL.

**Литература**: 1. Лекция № 3. Раздел «Значения по умолчанию и ограничения

целостности».

2. Моргунов Е.П. PostgreSQL. Основы языка SQL: учеб. пособие. – СПб.:

БХВ-Петербург, 2018.- 336 с.: ил.

**Задание 1.**

При использовании значений по умолчанию с ключевым словом **DEFAULT** возможны ситуации, когда типичным будет не конкретное значение данных, а способ его получения.

Например, если необходимо фиксировать в каждой строке таблицы «**Студенты**» имя пользователя базы данных, добавившего эту строку в таблице, то следует в определение таблицы добавить еще один столбец. Этот столбец по умолчанию будет получать значение, возвращаемое функцией **current\_user**.

CREATE TABLE students

( record\_book numeric(8) NOT NULL,

name text NOT NULL,

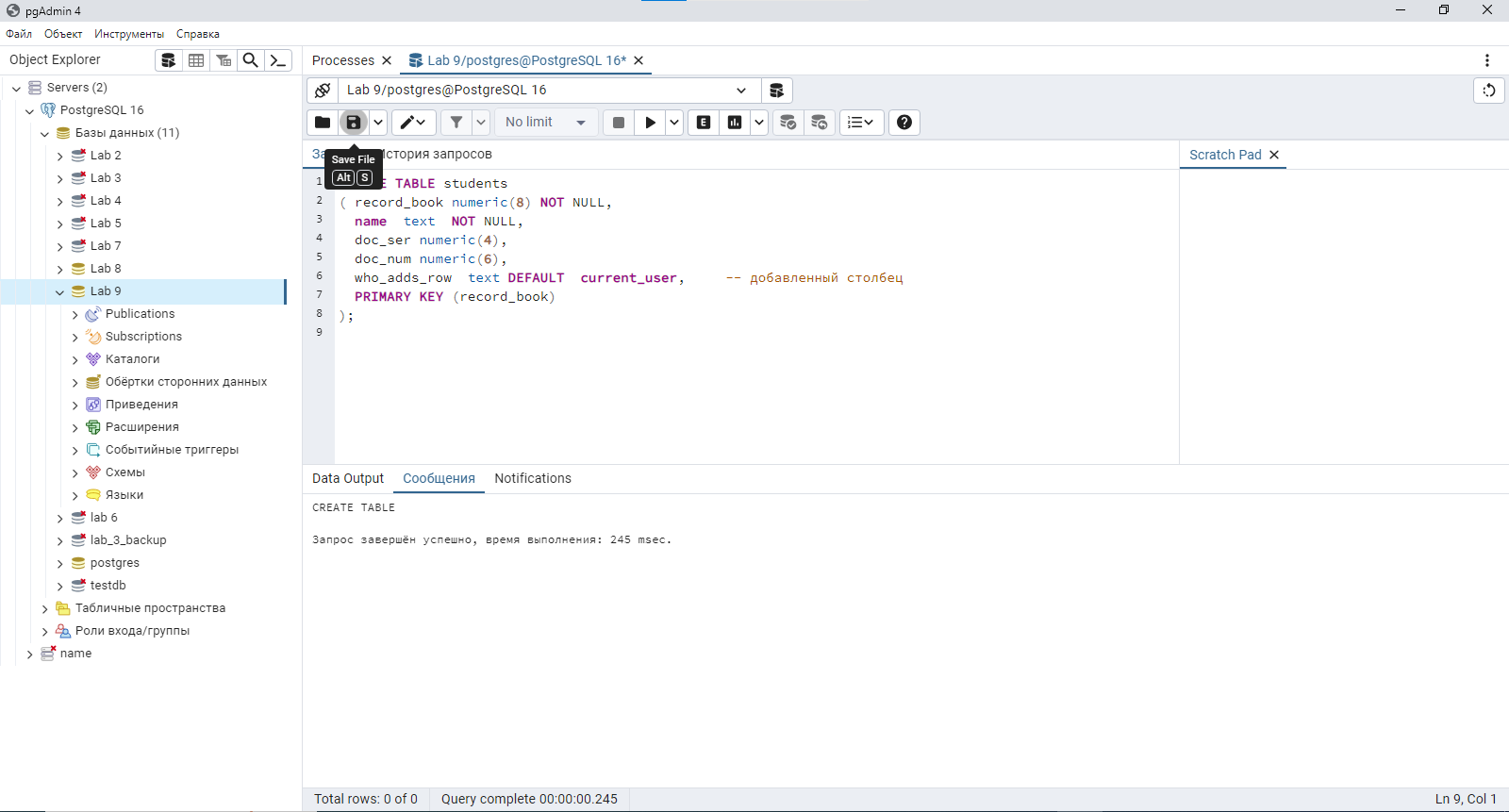
doc\_ser numeric(4),

doc\_num numeric(6),

who\_adds\_row text DEFAULT current\_user, -- добавленный столбец

PRIMARY KEY (record\_book)

);



Функция current\_user будет вызываться не при создании таблицы, а при вставке каждой строки. При этом в команде **INSERT** не требуется указывать значение для столбца **who\_adds\_row**, поскольку функция current\_user будет вызываться самой СУБД PostgreSQL.

Вставьте несколько строк в таблицу «**Студенты**» по примеру:

INSERT INTO students (record\_book, name, doc\_ser, doc\_num)

VALUES (16123456, ‘Иванов Иван Иванович’, 0402, 543281);

Выполните выборку строк из таблицы «Студенты» командой SELECT.

**INSERT INTO students (record\_book, name, doc\_ser, doc\_num)**

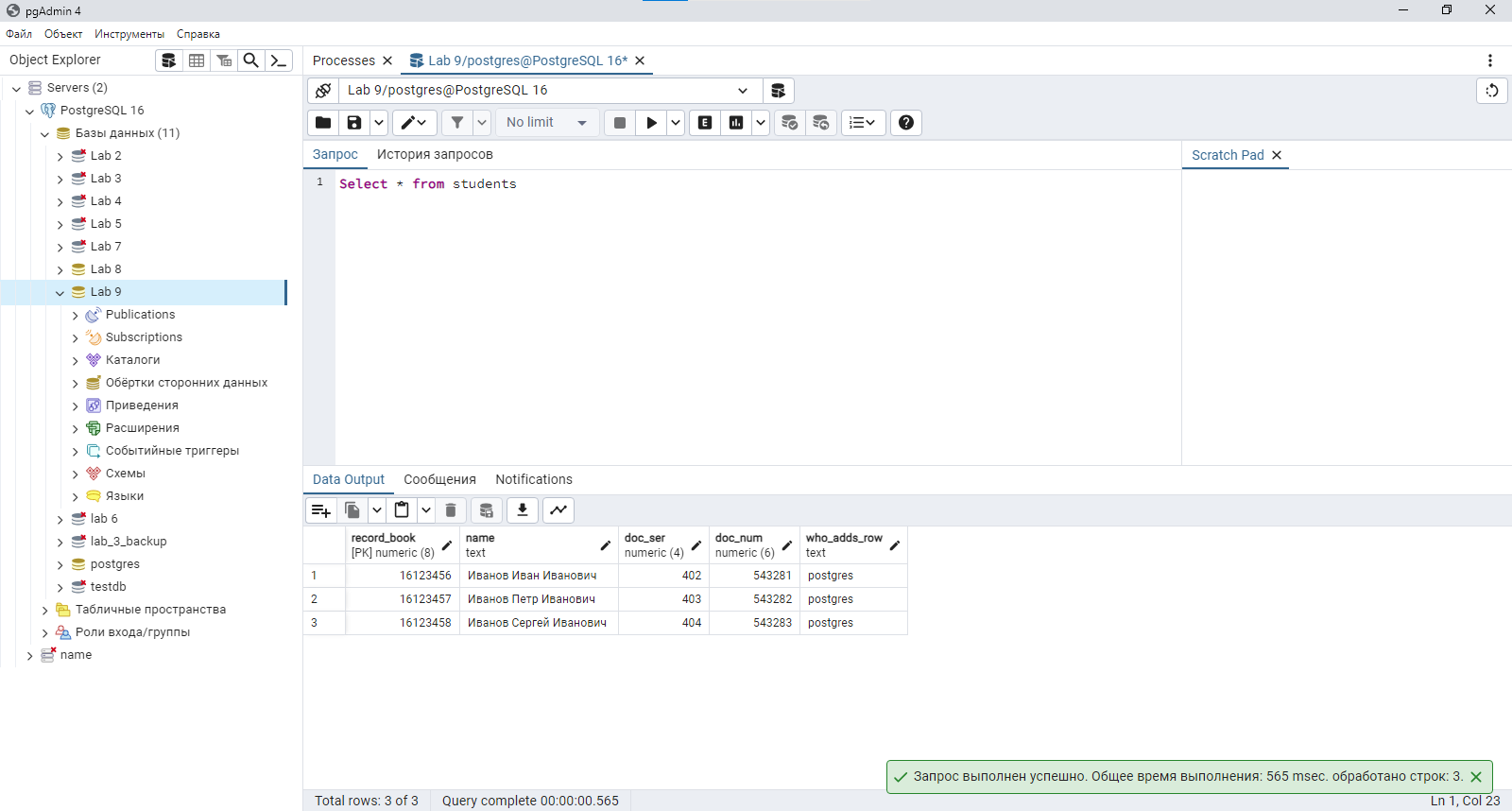
**VALUES (16123456, 'Иванов Иван Иванович', 0402, 543281);**

**INSERT INTO students (record\_book, name, doc\_ser, doc\_num)**

**VALUES (16123457, 'Иванов Петр Иванович', 0403, 543282);**

**INSERT INTO students (record\_book, name, doc\_ser, doc\_num)**

**VALUES (16123458, 'Иванов Сергей Иванович', 0404, 543283);**

****

**Задание 2.**

Пусть необходимо фиксировать не только имя пользователя базы данных, добавившего строку в таблицу, но также и момент времени, когда это было сделано.

Самостоятельно внесите модификацию в определение таблицы **students** для решения этой задачи, а затем выполните несколько команд **INSERT** для проверки полученного решения.

***Подсказка***: если Вы не знакомы с командой **ALTER TABLE**, то вместо модификации определения таблицы сначала удалите ее, а затем создайте заново:

DROP TABLE students;

CREATE TABLE students …

**CREATE TABLE students**

**( record\_book numeric(8) NOT NULL,**

**name text NOT NULL,**

**doc\_ser numeric(4),**

**doc\_num numeric(6),**

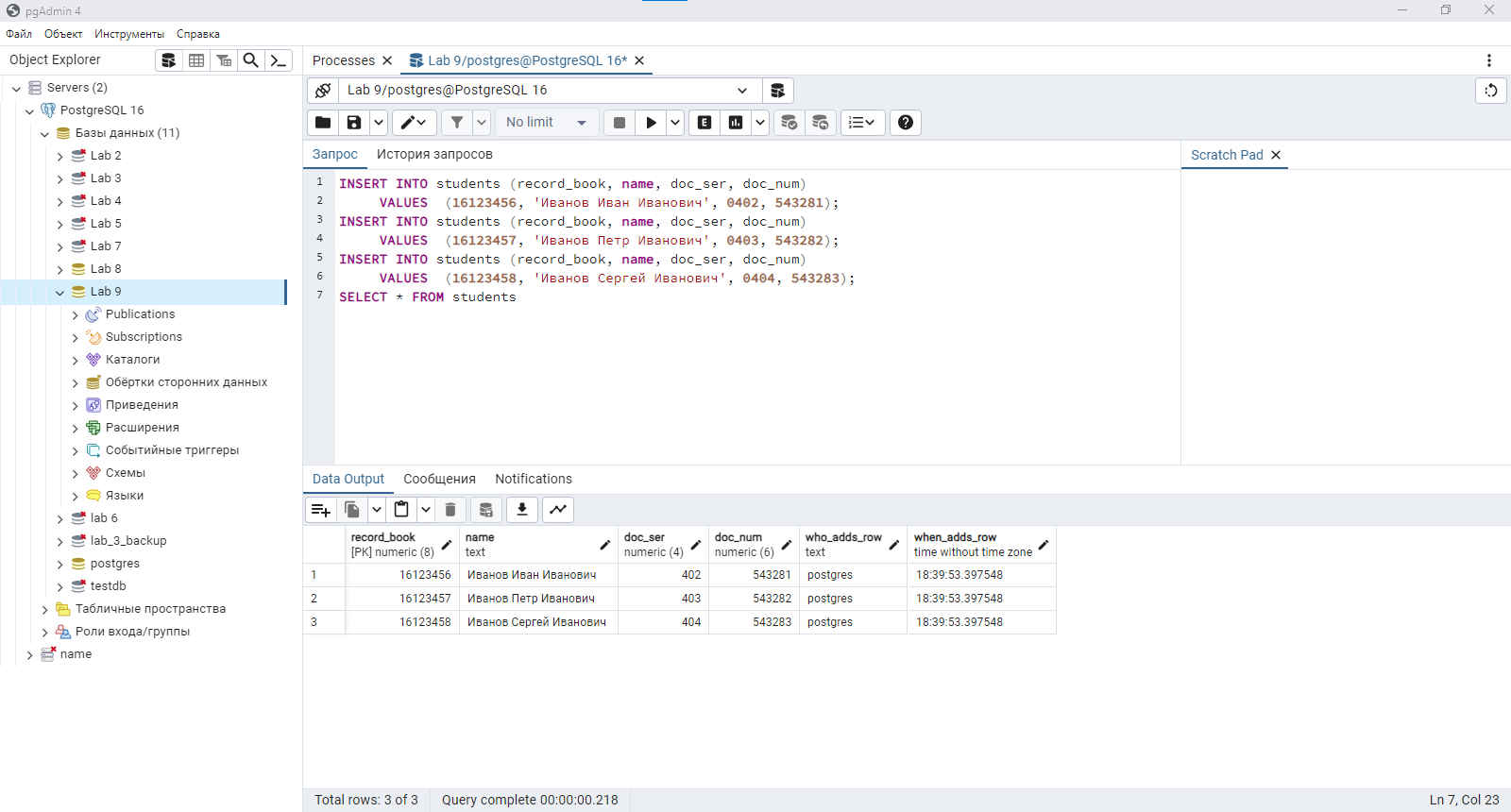
**who\_adds\_row text DEFAULT current\_user,**

**when\_adds\_row time DEFAULT current\_time,**

**-- добавленный столбец**

**PRIMARY KEY (record\_book)**

**);**

****

**Задание 3**.

Создайте таблицу «»Успеваемость»:

CREATE TABLE progress

( record\_book numeric(8) NOT NULL,

subject text NOT NULL,

acad\_year text NOT NULL,

term numeric(1) NOT NULL CHECK(term = 1 OR term = 2),

mark numeric(1) NOT NULL CHECK(mark >= 3 AND mark <= 5) DEFAULT 5,

FOREIGN KEY(record\_book)

REFERENCES students(record\_book)

ON DELETE CASCADE

ON UPDATE CASCADE

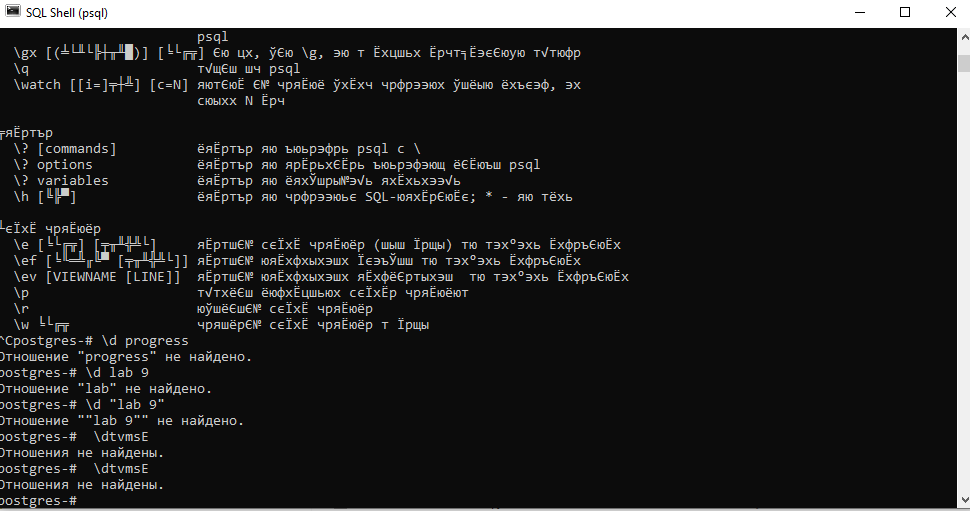
);

Посмотрите, какие ограничения уже наложены на атрибуты таблицы **progress**. Для этого воспользуйтесь командой **\d** утилиты **psql**. Представьте в отчете полученные данные.

Предложите для этой таблицы ограничения уровня таблицы.

***Подсказка***: надо сначала удалить таблицу students, а затем создать заново.

Посмотрите, какие ограничения наложены на атрибуты таблицы progress, созданной заново. Для этого воспользуйтесь командой **\d** утилиты **psq**l. Представьте в отчете полученные данные.



**Задание 4.**

Добавьте в таблицу **progress** еще один атрибут – «**Форма проверки знаний**» (**test\_form**)¸ который может принимать только два значения: «экзамен» или «зачет». Тогда набор допустимых значений атрибута «Оценка» **(mark**) будет зависеть от того, экзамен это или зачет предусмотрены по данной дисциплине. Если предусмотрен экзамен, то допускаются значения 3, 4, 5; если зачет – то 0 (не зачтено) или 1 (зачтено). Не забудьте, что значения **NULL** для атрибутов test\_form и mark не допускаются.

Новое ограничение может быть таким:

ALTER TABLE progress

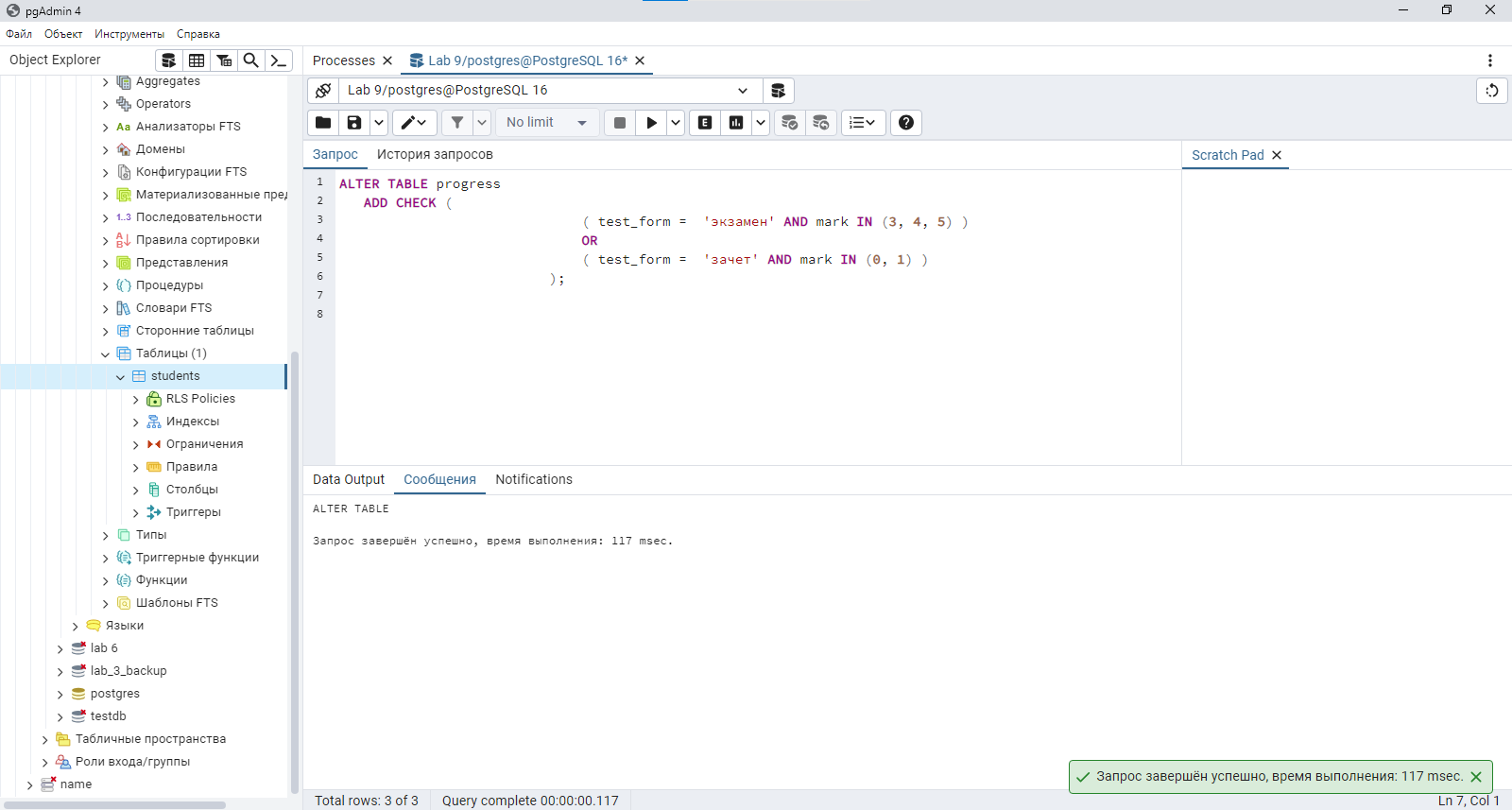
ADD CHECK (

( test\_form = 'экзамен' AND mark IN (3, 4, 5) )

OR

( test\_form = 'зачет' AND mark IN (0, 1) )

);



Проверьте, как будет работать новое ограничение в модифицированной таблице progress.

Для этого выполните несколько команд **INSERT**, как удовлетворяющих ограничению, так и нарушающих его.

В таблице **progress** уже было ограничение на допустимые значения атрибута mark.

Как Вы думаете, не будет ли оно конфликтовать с новыми ограничениями?

Проверьте эту гипотезу.

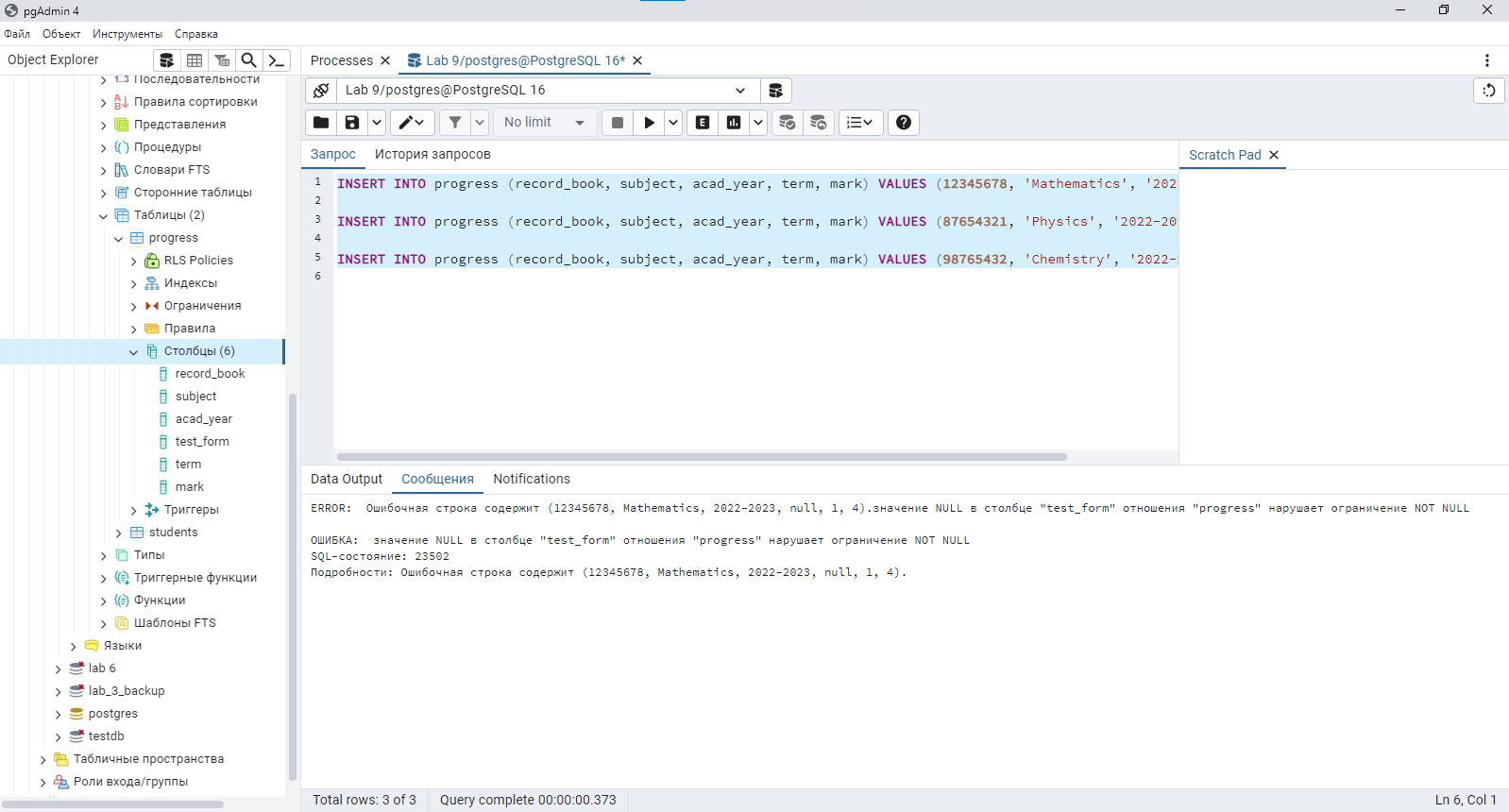
Если ограничения конфликтуют, тогда удалите старое ограничение и снова попробуйте добавить строки в таблицу.

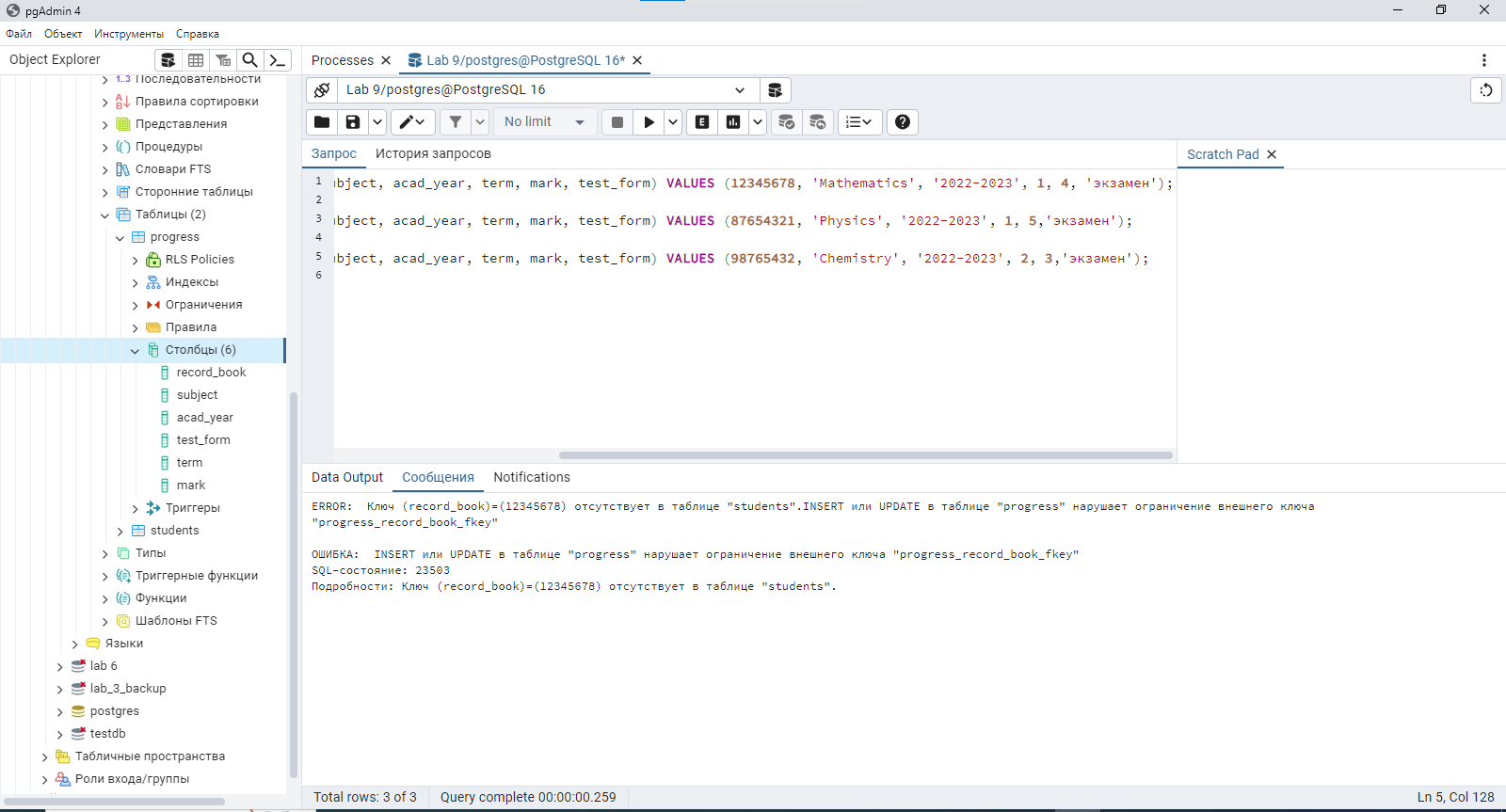
Подумайте, какое еще ограничение уровня таблицы можно предложить для этой таблицы? Ведь Вы прекрасно знаете эту предметную область!!!

INSERT INTO progress (record\_book, subject, acad\_year, term, mark) VALUES (12345678, 'Mathematics', '2022-2023', 1, 4);

INSERT INTO progress (record\_book, subject, acad\_year, term, mark) VALUES (87654321, 'Physics', '2022-2023', 1, 5);

INSERT INTO progress (record\_book, subject, acad\_year, term, mark) VALUES (98765432, 'Chemistry', '2022-2023', 2, 3);





**Задание 5.**

В определении таблицы «**Успеваемость**» (**progress**) на атрибуты **term** и **mark** наложены как ограничения **CHEСK**, так и ограничения **NULL**.

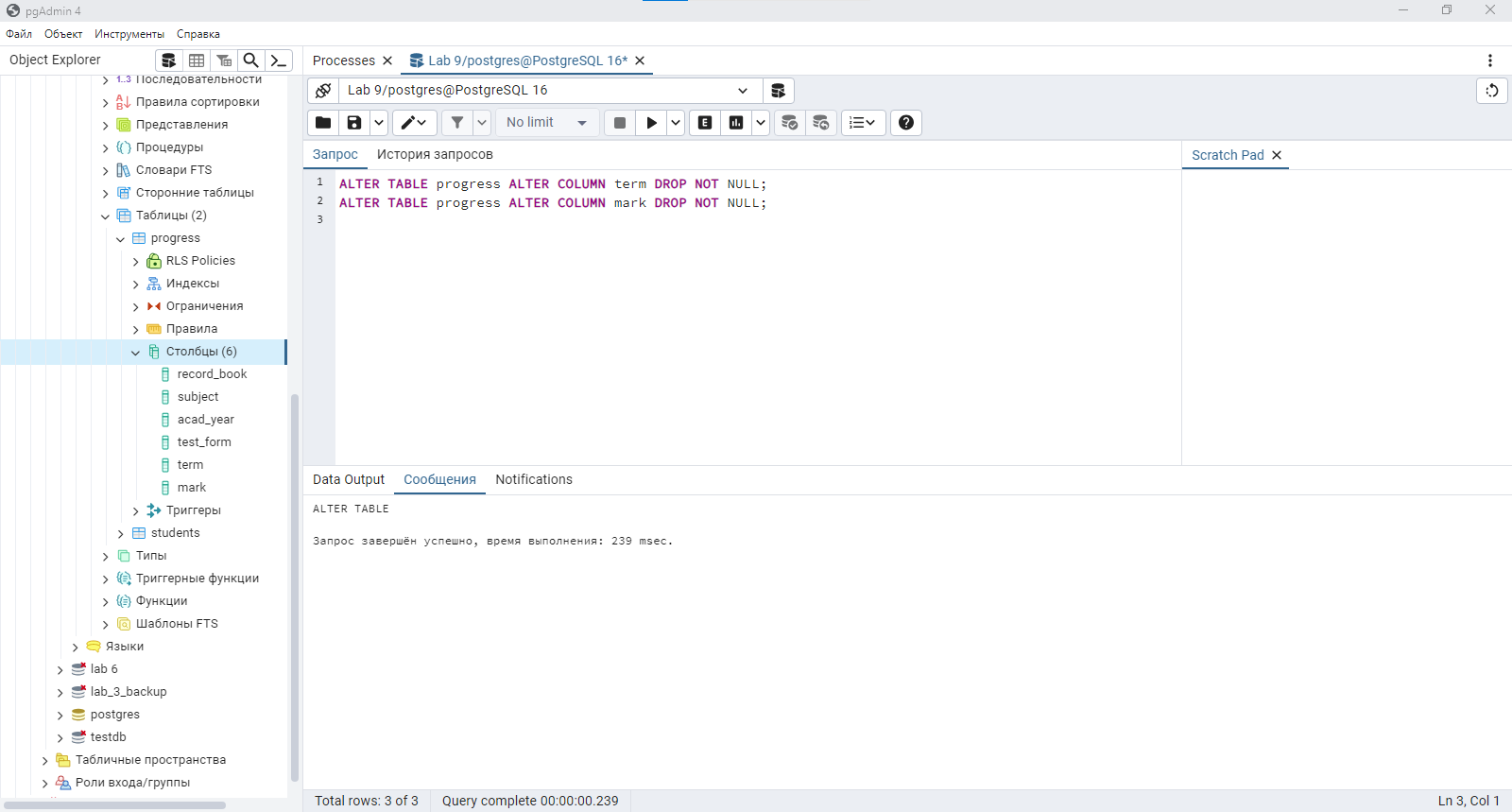
Возникает вопрос: не является ли ограничение **NOT NULL** избыточным? Ведь в ограничении **CHECK** явно указаны допустимые значения.

Проверьте гипотезу об избыточности ограничения **NOT NULL** в данном случае.

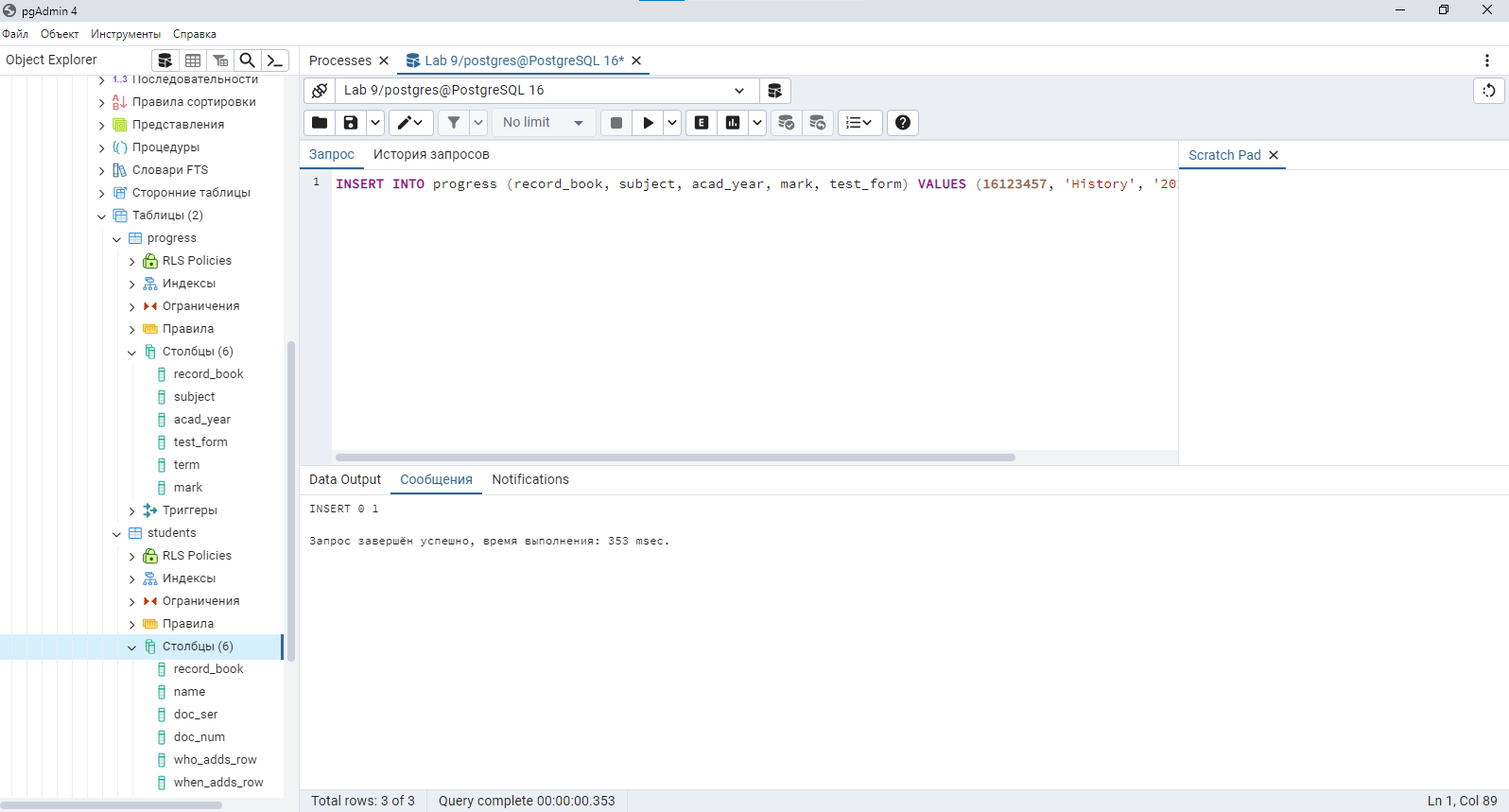
Для этого модифицируйте таблицу, убрав ограничение **NOT NUL**, и попробуйте добавить в нее строку с отсутствующим значением атрибута **term** (или **mark**).

Для того чтобы изменить таблицу и убрать ограничение NOT NULL, можно использовать следующий SQL запрос:  
ALTER TABLE progress ALTER COLUMN term DROP NOT NULL;

ALTER TABLE progress ALTER COLUMN mark DROP NOT NULL;

  
После выполнения этих запросов, можно попробовать добавить строку с отсутствующим значением атрибута term (или mark). Например:

INSERT INTO progress (record\_book, subject, acad\_year, mark) VALUES (12345678, 'History', '2022-2023', 4);

  
Этот запрос добавит строку с отсутствующим значением атрибута term в таблицу.

**Задание 6.**

В определении таблицы «**Успеваемость**» (**progress**) для атрибута **mark** не только задано ограничение **CHECK**, но и установлено значение по умолчанию с помощью ключевого слова **DEFAULT**:

…

mark numeric(1) NOT NULL CHECK(mark >= 3 AND mark <= 5) DEFAULT 5,

…

Как Вы думаете, что будет, если в ограничении **DEFAULT** случайно допустить ошибку, написав **DEFAULT 6**?

Если в команде **INSERT** не указать значение для атрибута **mark**, то на каком этапе эта ошибка будет выявлена: уже на этапе создания таблицы или только при вставке строки в нее?

Если в ограничении DEFAULT случайно допустить ошибку, например, установив DEFAULT 6 вместо правильного значения 5, то при попытке вставить строку без указания значения для атрибута mark (то есть при использовании значения по умолчанию), будет сгенерировано исключение.  
  
Ошибку с неверным значением по умолчанию DEFAULT 6 можно обнаружить на этапе создания таблицы, так как при выполнении команды создания таблицы будет проведена проверка правильности всех указанных значений по умолчанию. Если значение по умолчанию не соответствует типу данных или ограничениям CHECK, то создание таблицы завершится с ошибкой.  
  
Таким образом, ошибка с неверным значением по умолчанию будет выявлена на этапе создания таблицы, а не только при вставке строки в нее.

Ниже приведена полезная для проверки гипотезы команды, поскольку в ней отсутствует передаваемое значение для атрибута **mark**:

INSERT INTO progress (record\_book, subject, acad\_year, term)

VALUS (16000001), ‘Физика’, '2016-2017', 1);

**Задание 7.**

В стандарте **SQL** сказано, что при наличии ограничения уникальности, включающего один или более столбцов, все же возможны повторяющиеся значения этих столбцов в разных строках, но лишь в том случае, если это значения **NULL**.

PostgreSQL придерживается такого же подхода.

Модифицируйте определение таблицы «**Студенты**» (**students**), добавив ограничение уникальности по двум столбцам: **doc\_ser** и **doc\_num**. А затем проверьте вышеприведенное утверждение, добавив в таблицу не только строки, содержащие конкретные значения этих двух столбцов, но также и по две строки, имеющие следующие свойства:

- одинаковые значения столбца **doc\_ser** и **NULL**-значения столбца **doc\_num**;

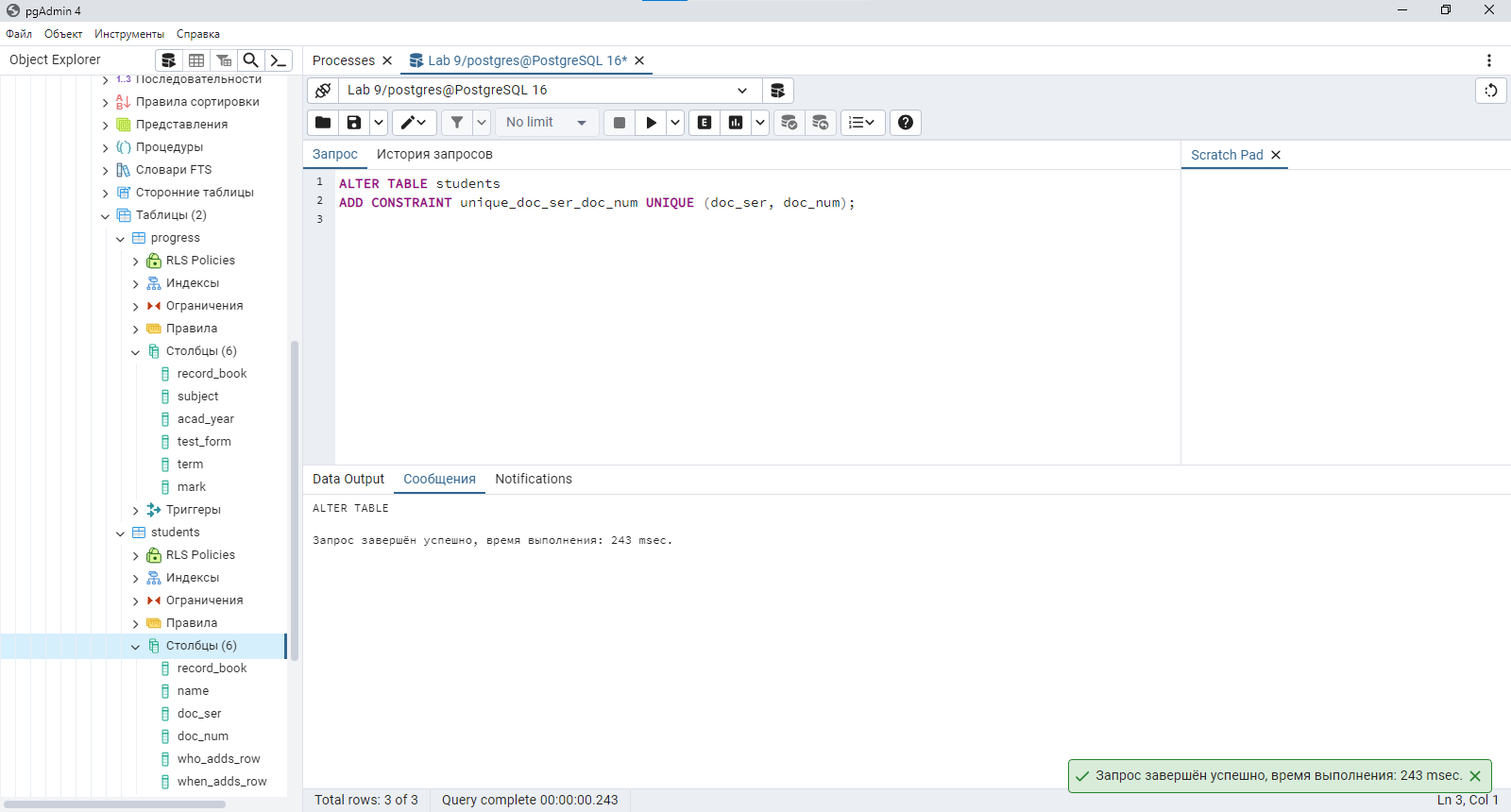
- **NULL**-значения столбца **doc\_ser** и столбца **doc\_num**.

*Подобные вещи возможны, так как NULL-значения не считаются совпадающими!*

Проверьте с помощью команды:

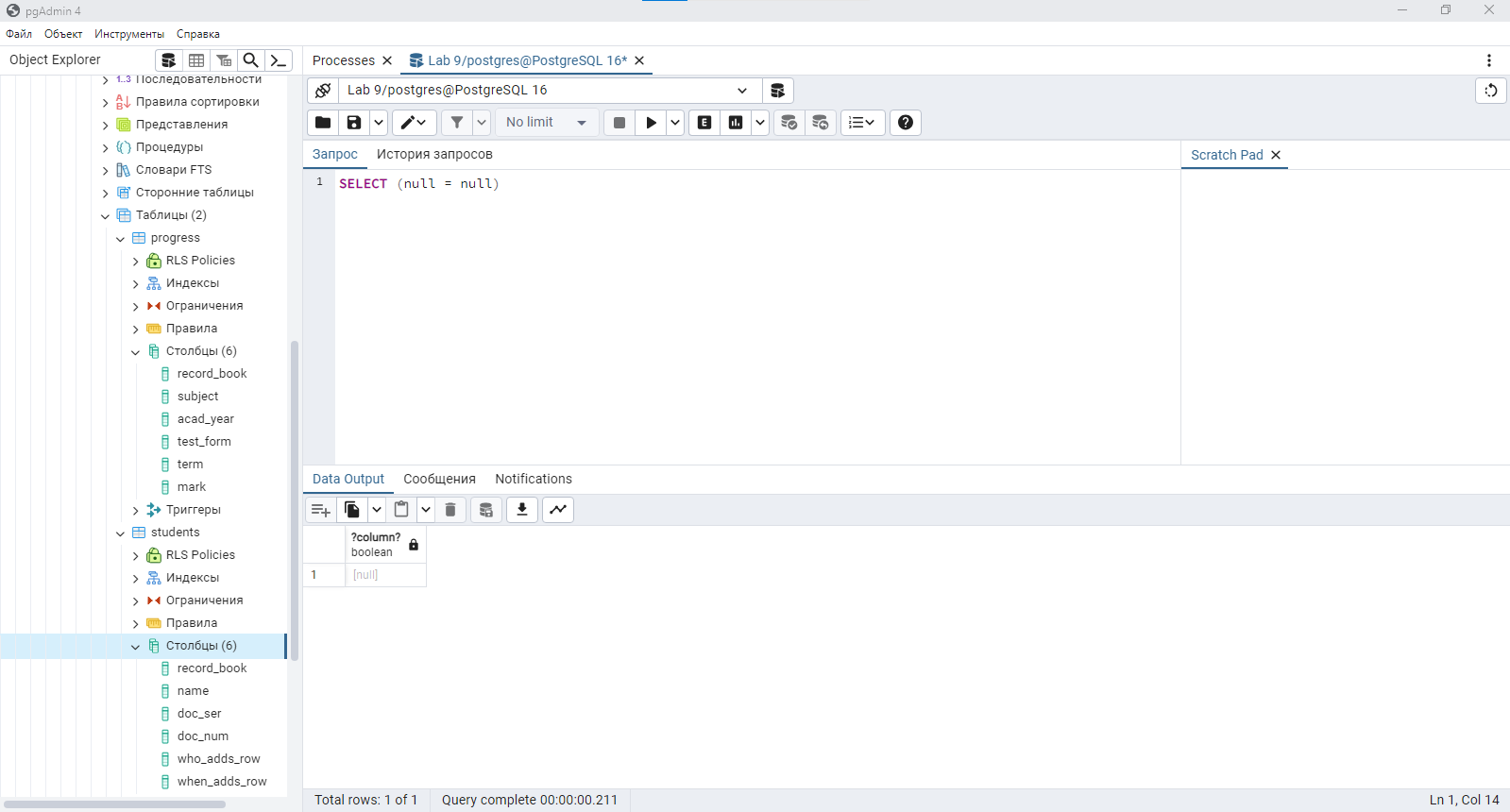
SELECT (null = null);

Эта команда даст пустой результат (т.е. **NULL**).



ALTER TABLE students

ADD CONSTRAINT unique\_doc\_ser\_doc\_num UNIQUE (doc\_ser, doc\_num);



**Задание 8.**

Модифицируйте определения таблиц «**Студенты**» (**students**) и «**Успеваемость**» (**progress**).

В таблице **students** в качестве первичного ключа назначьте комбинацию атрибутов **doc\_ser** и **doc\_num**, а в таблице **progress** соответствующим образом измените определение внешнего ключа.

CREATE TABLE students

( record\_book numeric(8) NOT NULL UNIQUE,

name text NOT NULL,

doc\_ser numeric(4),

doc\_num numeric(6),

PRIMARY KEY (doc\_ser, doc\_num)

);

Обратите внимание, что для атрибутов **doc\_ser** и **doc\_num** можно не указывать ограничение **NOT NULL**: они входят в состав первичного ключа, а в нем **NULL**-значения не допускаются, поэтому ограничение **NOT NULL** фактически подразумевается при включении атрибута в состав первичного ключа.

CREATE TABLE progress

( doc\_ser numeric(4),

doc\_num numeric(6),

subject text NOT NULL,

acad\_year text NOT NULL,

term numeric(1) NOT NULL CHECK(term = 1 OR term = 2),

mark numeric(1) NOT NULL CHECK(mark >= 3 AND mark <= 5) DEFAULT 5,

FOREIGN KEY(doc\_ser, doc\_num)

REFERENCES students(doc\_ser, doc\_num)

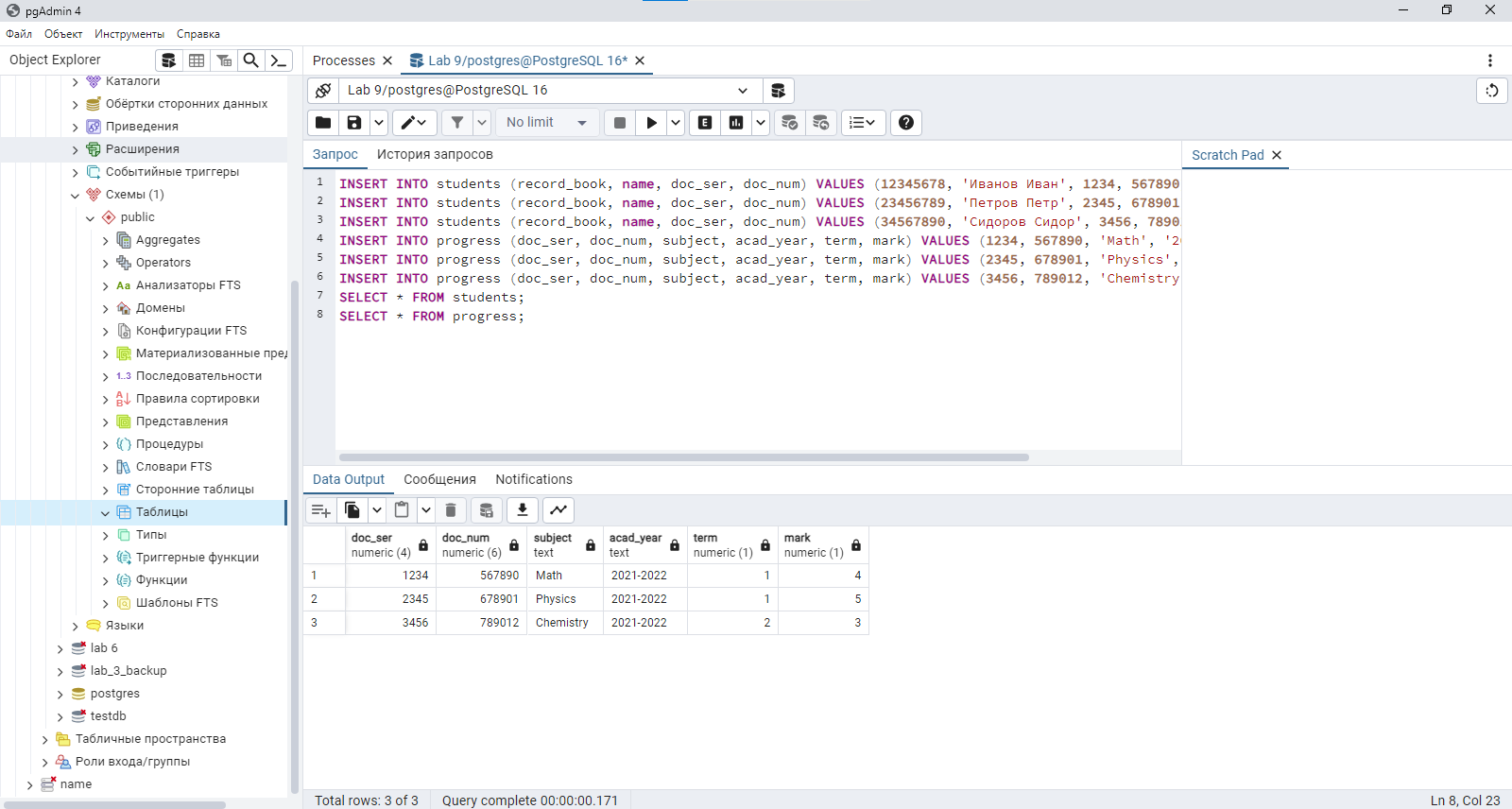
ON DELETE CASCADE

ON UPDATE CASCADE

);

Теперь и первичный , и внешний ключи – составные!

Проверьте их действие, добавив несколько строк с каждую таблицу.

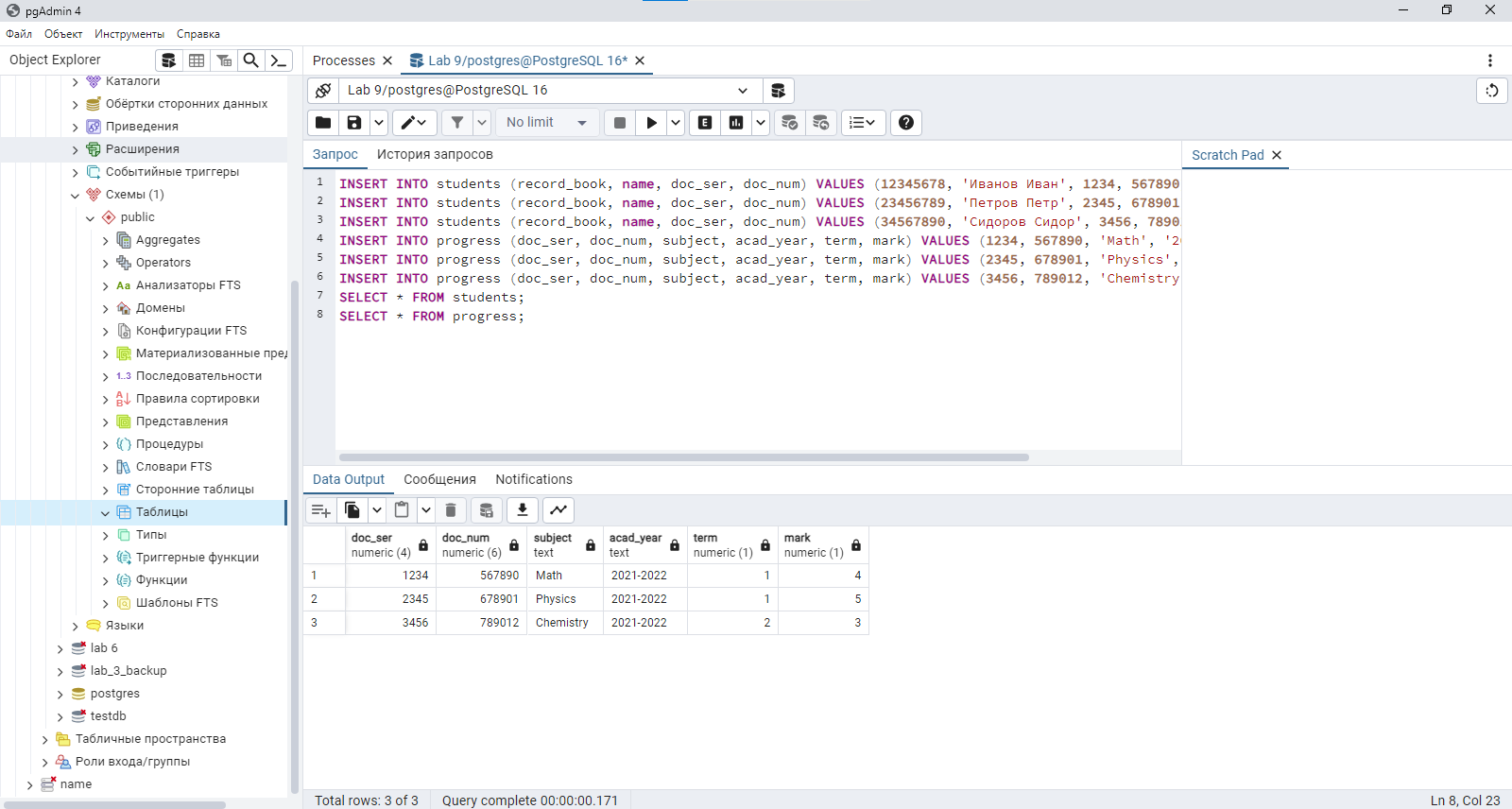
******

***Задание 9***.

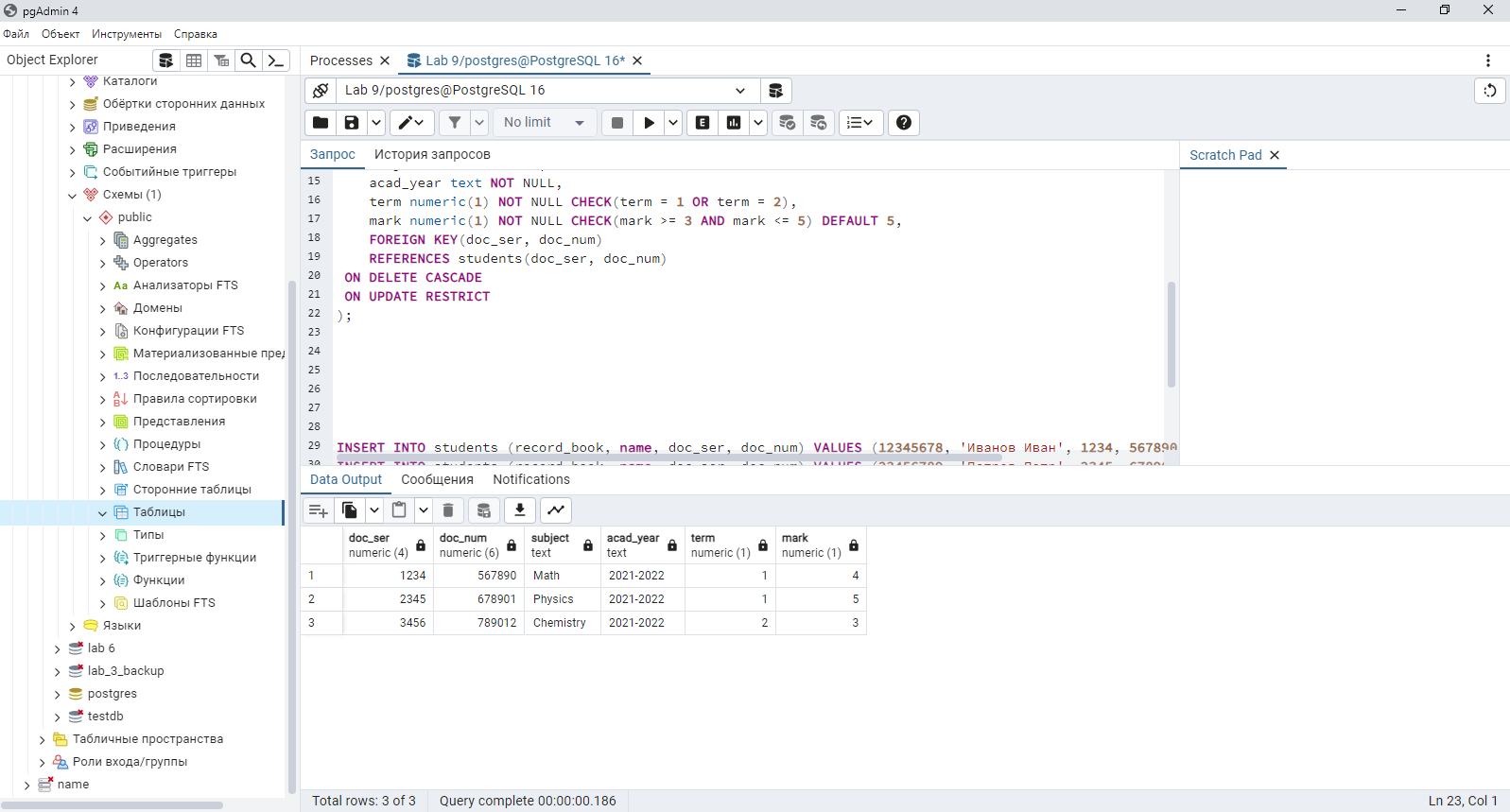
Модифицируйте определение таблицы «**Успеваемость**» (**progress**), а если потребуется, то и определение таблицы «**Студенты**» (**students**), чтобы изучить все варианты реагирования СУБД на обновление строк в ссылочной таблице, в данном случае - **students**.

Последовательно изменяйте определение внешнего ключа таблицы **progress**, испробовав варианты:

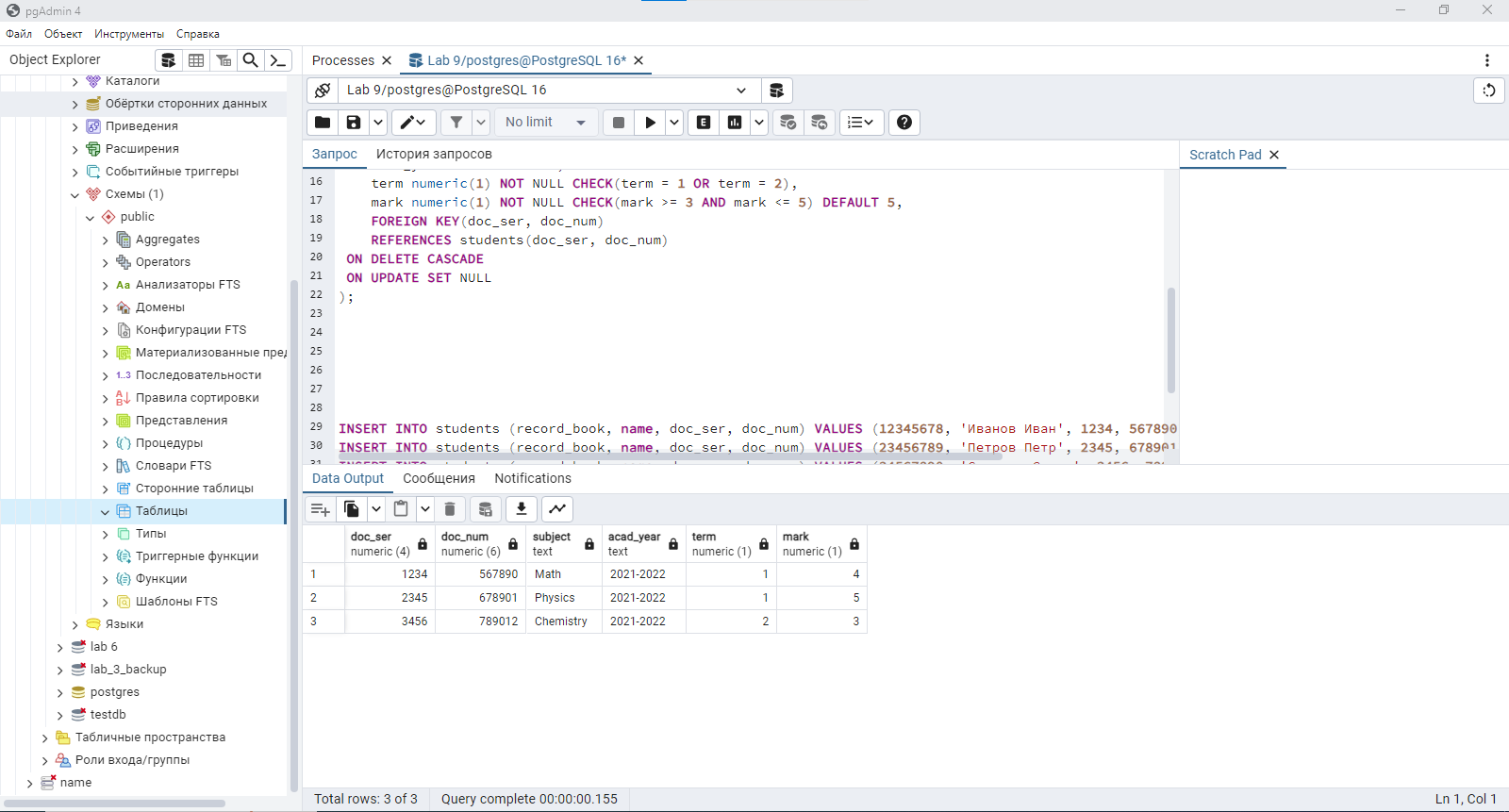
- ON UPDATE CASCADE

******

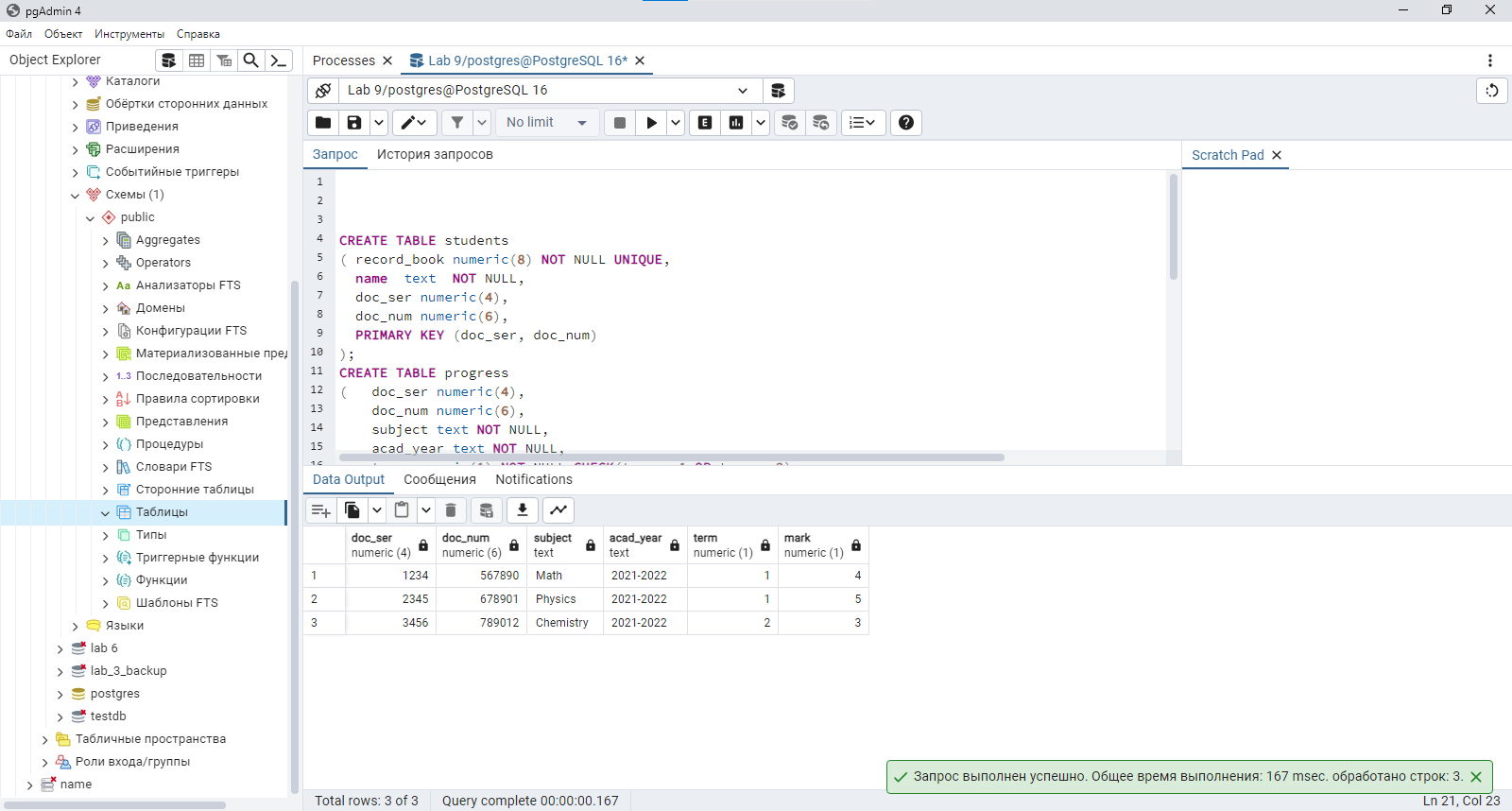
- ON UPDATE RESTRICT



- ON UPDATE SET NULL



- ON UPDATE SET DEFAULT



Для получения информативной картинки введите несколько строк в обе таблицы, а затем выполняйте операцию **UPDATE**, подбирая значения ключевых атрибутов таким образом, чтобы вызвать ожидаемую реакцию СУБД.

Учтите, что при использовании фразы **ON UPDATE SET DEFAULT** необходимо, чтобы:

- во-первых, с помощью ключевого слова **DEFAULT** было установлено значение по умолчанию для атрибута внешнего ключа в ссылающейся таблице (**progress**).

- во-вторых, это значение это значение по умолчанию все равно должно присутствовать в одной из строк ссылочной таблицы.

Как Вы считаете, с учетом сказанного, возможно ли использование **ON UPDATE SET DEFAULT** в нашем случае?

Попробуйте обосновать или, наоборот, опровергнуть целесообразность использования каждой из этих политик – CASCADE, RESTRICT, SET NULL и SET DEFAULT – при выполнении операции **UPDATE** в реальной информационной системе, предназначенной для учета успеваемости студентов.

CREATE TABLE students

( record\_book numeric(8) NOT NULL UNIQUE,

name text NOT NULL,

doc\_ser numeric(4),

doc\_num numeric(6),

PRIMARY KEY (doc\_ser, doc\_num)

);

CREATE TABLE progress

( doc\_ser numeric(4),

doc\_num numeric(6),

subject text NOT NULL,

acad\_year text NOT NULL,

term numeric(1) NOT NULL CHECK(term = 1 OR term = 2),

mark numeric(1) NOT NULL CHECK(mark >= 3 AND mark <= 5) DEFAULT 5,

FOREIGN KEY(doc\_ser, doc\_num)

REFERENCES students(doc\_ser, doc\_num)

ON DELETE CASCADE

ON UPDATE SET DEFAULT

);

INSERT INTO students (record\_book, name, doc\_ser, doc\_num) VALUES (12345678, 'Иванов Иван', 1234, 567890);

INSERT INTO students (record\_book, name, doc\_ser, doc\_num) VALUES (23456789, 'Петров Петр', 2345, 678901);

INSERT INTO students (record\_book, name, doc\_ser, doc\_num) VALUES (34567890, 'Сидоров Сидор', 3456, 789012);

INSERT INTO progress (doc\_ser, doc\_num, subject, acad\_year, term, mark) VALUES (1234, 567890, 'Math', '2021-2022', 1, 4);

INSERT INTO progress (doc\_ser, doc\_num, subject, acad\_year, term, mark) VALUES (2345, 678901, 'Physics', '2021-2022', 1, 5);

INSERT INTO progress (doc\_ser, doc\_num, subject, acad\_year, term, mark) VALUES (3456, 789012, 'Chemistry', '2021-2022', 2, 3);

SELECT \* FROM students;

SELECT \* FROM progress;

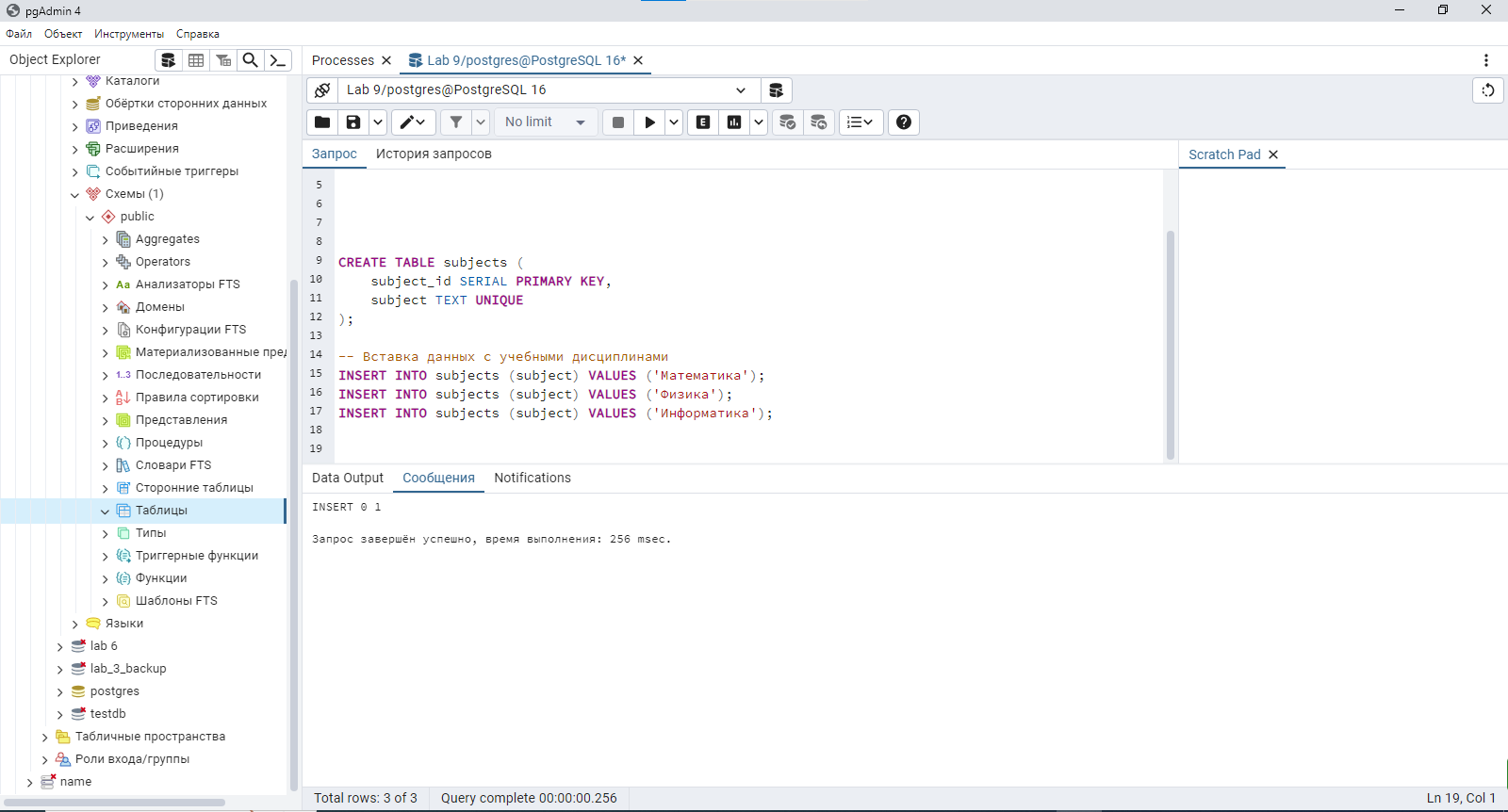
**Задание 10.**

В таблице «**Успеваемость**» (**progress**) есть атрибут «**Учебная дисциплина**» (**subject**). Это текстовый атрибут. Одинаковые наименования учебных дисциплин записываются в таблицу **progress** многократно.

Создайте еще одну таблицу – «**Учебные дисциплины**» (**subjects**), в которой будет два атрибута: «**Идентификатор учебной дисциплины**» (**subject\_id**) и «**Учебная дисциплина**» (**subject**). Тип данных первого из них будет **integer**, а второго – **text**. В

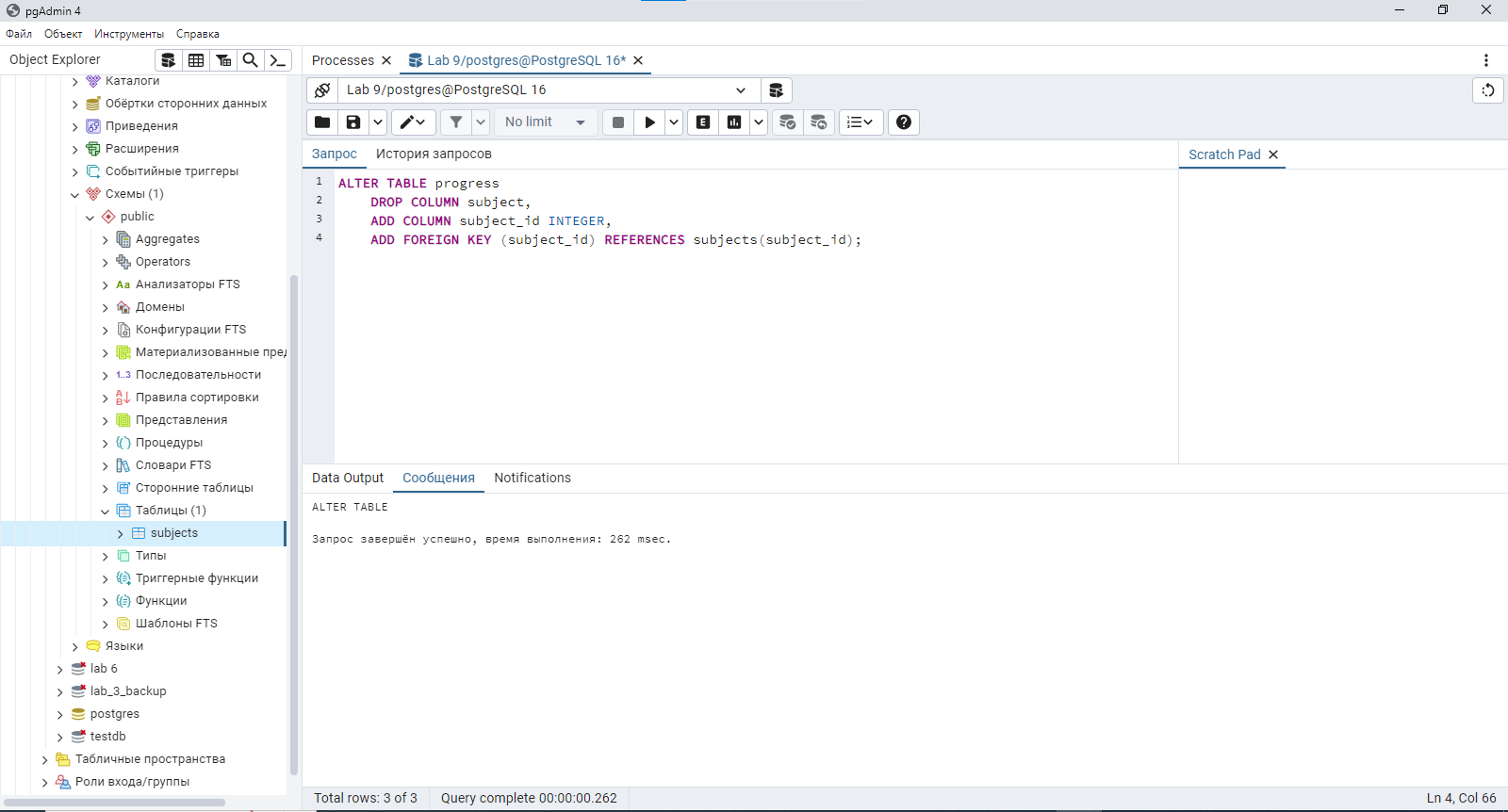
качестве первичного ключа будет служить subject\_id, а второй атрибут будет уникальным.

Введите в новую таблицу две-три строки для различных учебных дисциплин.



Модифицируйте таблицу **progress**, заменив атрибут **subject** на **subject\_id**. Тип данных нового атрибута будет integer. Поскольку тип данных изменится, то для замены первоначальных значений, хранящихся в этом столбце, на новые придется использовать конструкцию **USING** (*попробуйте изучить эту команду самостоятельно*), или внести изменения непосредственно в таблице.

Добавьте в определение таблицы **progress** еще один внешний ключ, который будет ссылаться на таблицу **subjects**. В составе этого внешнего ключа будет только один атрибут – **subject\_id**. Таким образом, таблица может иметь больше одного внешнего ключа!

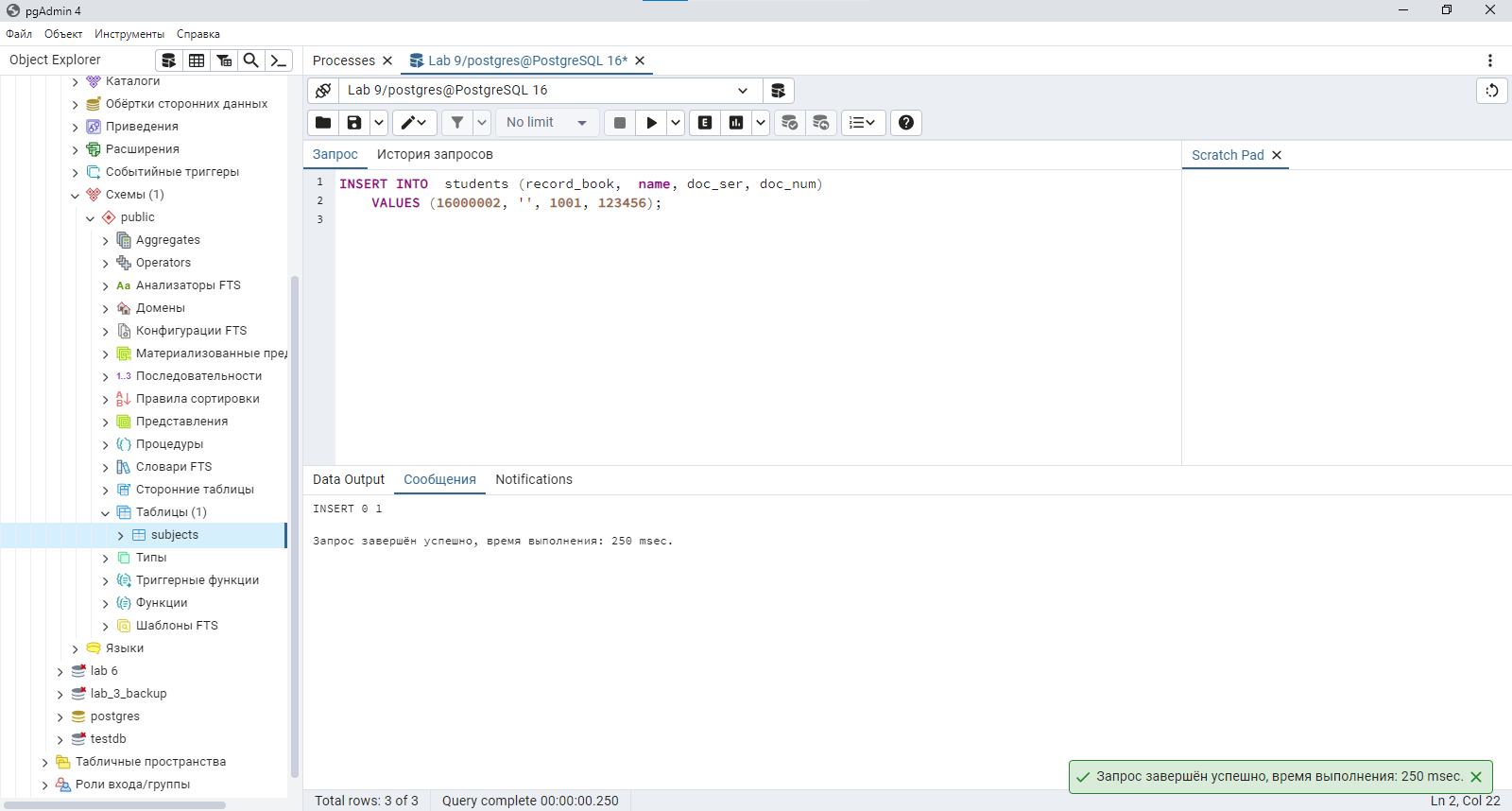
Структура связей в реальной базе данных может оказаться довольно сложной!

Теперь введите несколько строк и в таблицу **progress**, учитывая ее связь с новой таблицей **subjects**.

**Задание 11.**

В таблице «**Студенты**» (**students**) есть текстовый атрибут **name**, на который наложено ограничение NOT NULL.

Что будет, если при вводе новой строки в эту таблицу дать атрибуту **name** в качестве значения пустую строку?



Например:

INSERT INTO students (record\_book, name, doc\_ser, doc\_num)

VALUES (16000002, ‘’, 1001, 123456);

Как сделать так, чтобы не проходили пустые строки в качестве значения атрибута name в таблицу?

Следует добавить ограничение **CHECK** для столбца name.

Если Вам пока не знакома команда **ALTER TABLE**, то удалите таблицу **students** и создайте ее заново с учетом нового ограничения.

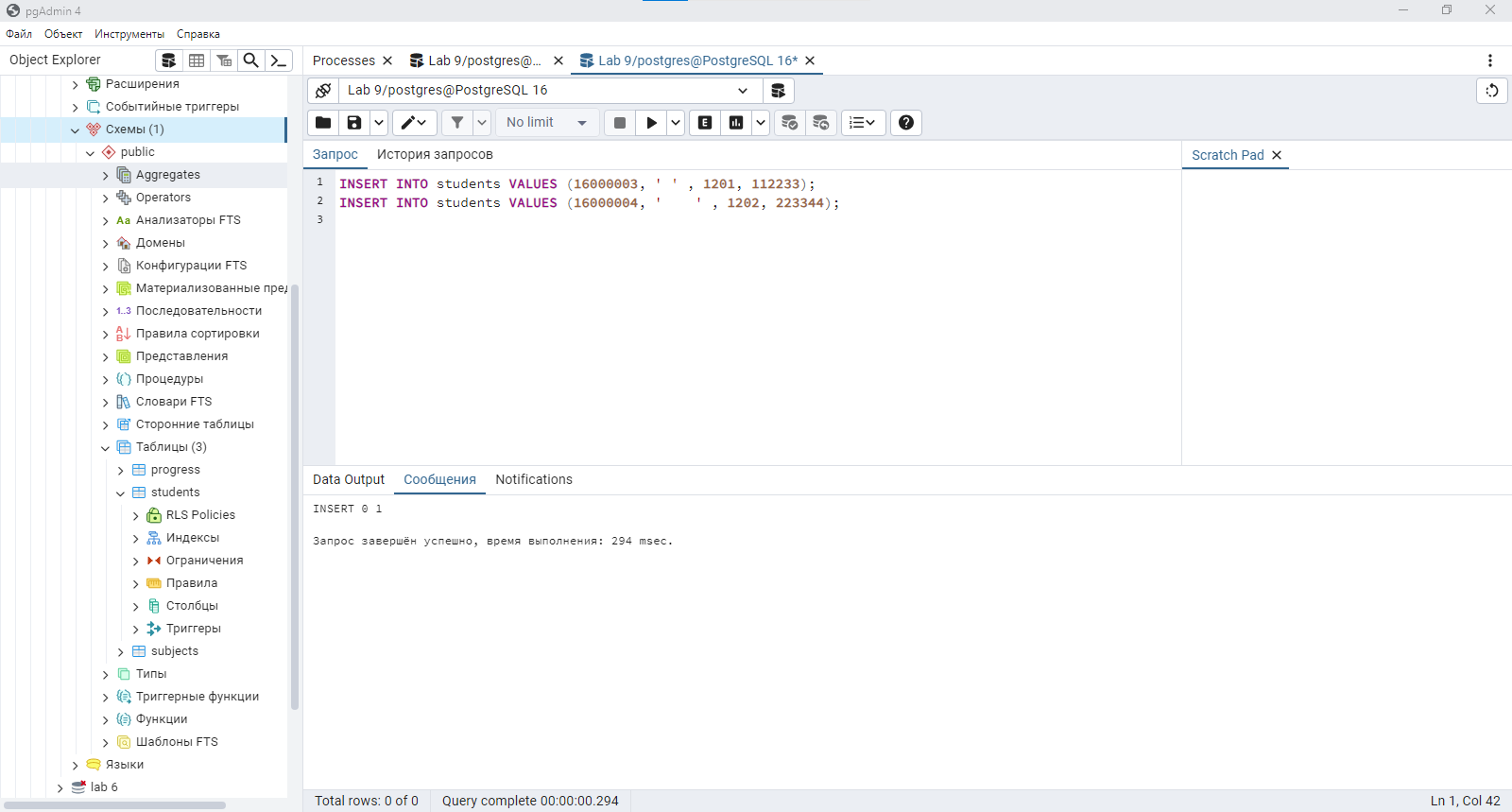
Если же Вы знакомы с командой **ALTER TABLE**, то сделайте так:

ALTER TABLE students ADD CHECK (name < > ‘’);

Добавив ограничение, попробуйте теперь вставить в таблицу значения атрибута **name** строку, содержащую сначала один пробел, а потом два пробела.

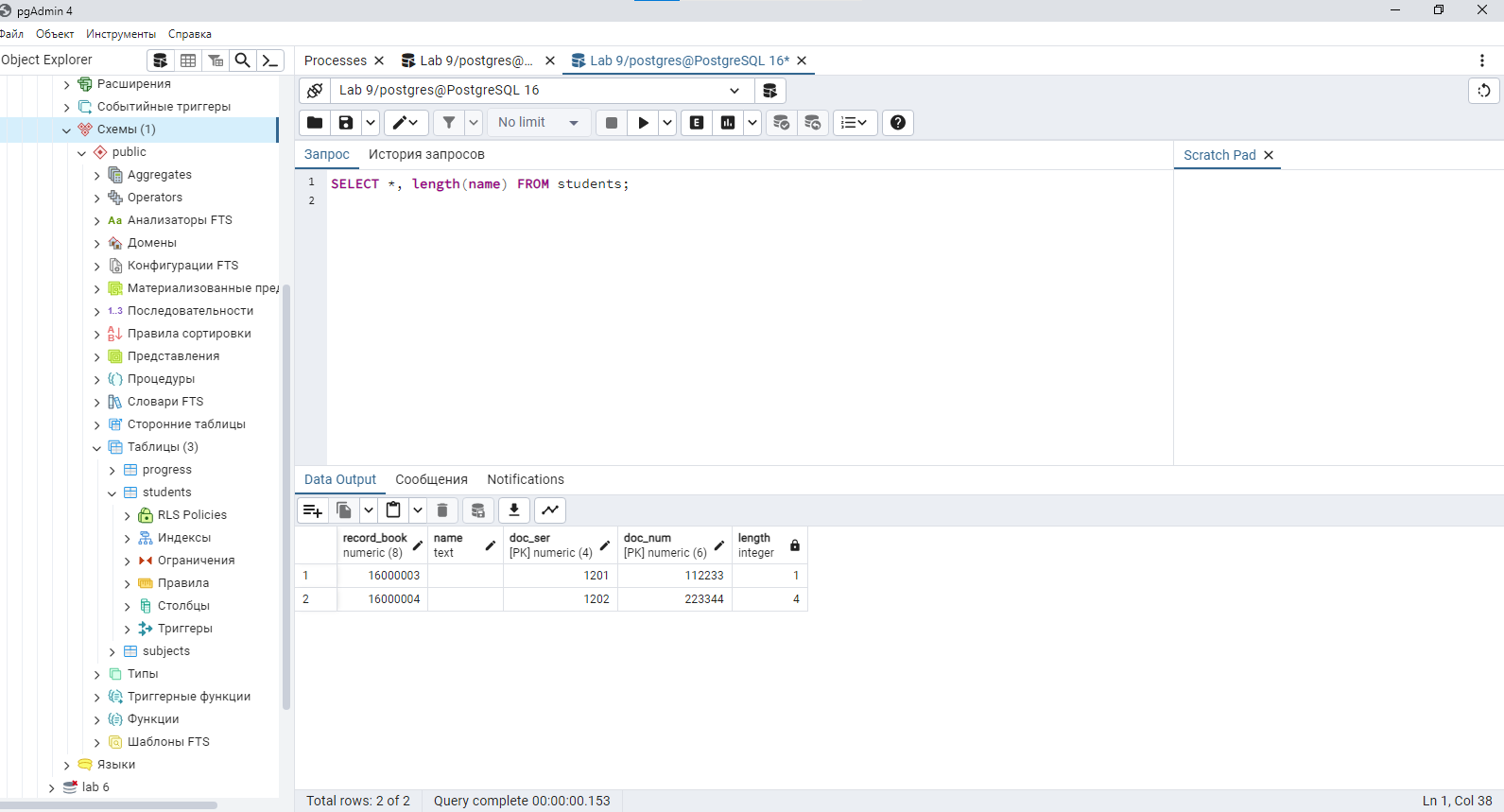
INSERT INTO students VALUES (16000003, ‘ ‘ , 1201, 112233);

INSERT INTO students VALUES (16000004, ‘ ‘ , 1202, 223344);



Для того, чтобы «увидеть» эти пробелы в выборке, сделайте так:

SELECT \*, length(name) FROM students;



Оказывается, эти невидимые значения имеют ненулевую длину!

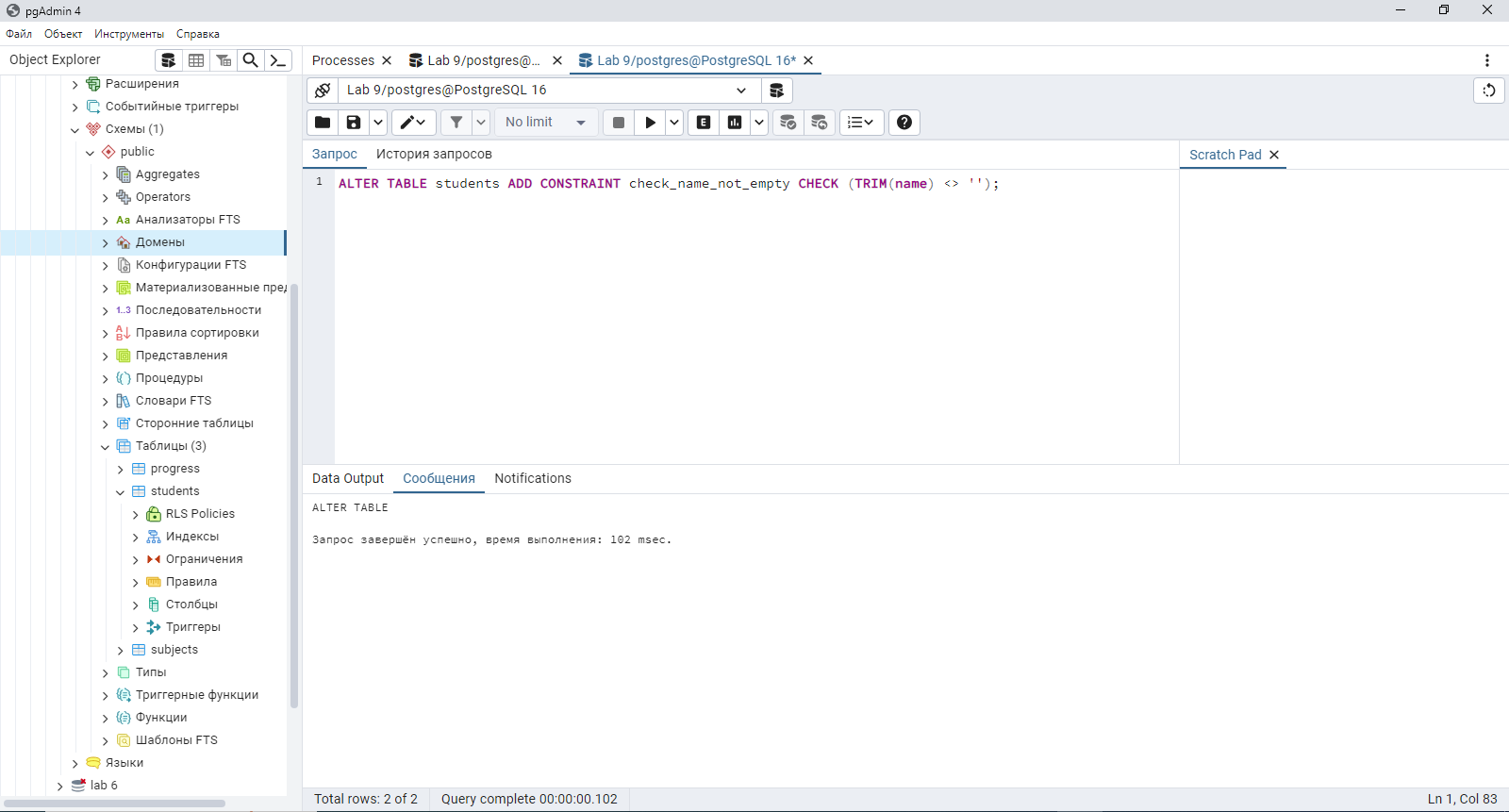
Что сделать, чтобы не допустить таких значений-невидимок?

Один из способов: возложить проверку таких ситуаций на прикладную программу, что не желательно.

Но это можно сделать на уровне определения таблицы **students**!

Есть функция **trim**, удаляющая начальные пробелы и конечные пробелы в текстовом поле.

Используйте команду **ALTER TABLE students ADD CHECK (…)** или удалите таблицу **students** и создайте ее заново с учетом нового ограничения.



**Задание 12.**

В таблице «**Студенты**» (**students**) атрибут “**Серия документа, удостоверяющего личность**» (**doc\_ser**) имеет числовой тип, однако в сериях таких документов могут встречаться лидирующие нули, которые в числовых столбцах не сохраняются. Например, при записи значения серии «0402» первый ноль не сохранится.

Модифицируйте таблицу **students**, заменив числовой тип данных на символьный, например, **character**.

ALTER TABLE students

ALTER COLUMN doc\_ser TYPE character varying;

