Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №3**

|  |
| --- |
| Основы языка определения данных |

тема

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | А. Д. Вожжов |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ23-17/2б, 032320981 |  |  |  | Р. А. Троицкий |
|  | номер группы, зачетной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2025

Содержание

[1 Цели 3](#_Toc190097585)

[2 Задачи 3](#_Toc190097586)

[3 Ход выполнения 9](#_Toc190097587)

[4 Выводы 29](#_Toc190097588)

1. Цели

Ознакомиться с типами данных в SQL.

1. Задачи

Для выполнения практической работы необходимо выполнить следующие задачи.

1. При использовании значений по умолчанию с ключевым словом DEFAULT возможны и ситуации, когда типичным будет не конкретное значение данных, а способ его получения. Например, если мы захотим фиксировать в каждой строке таблицы «Студенты» имя пользователя базы данных, добавившего эту строку в таблицу, тогда необходимо в определение таблицы добавить еще один 133 Глава 5. Основы языка определения данных столбец. Этот столбец по умолчанию будет получать значение, возвращаемое функцией current\_user. CREATE TABLE students ( record\_book numeric( 5 ) 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 ) ); Эта 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: INSERT INTO students ( record\_book, name, doc\_ser, doc\_num ) VALUES ( 12300, 'Иванов Иван Иванович', 0402, 543281 ); Давайте пойдем дальше и пожелаем 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 команду INSERT для проверки полученного решения. Если до выполнения этого упражнения вы еще не ознакомились с командой ALTER TABLE, то вместо модифицирования определения таблицы сначала удалите ее, а затем создайте заново: DROP TABLE students; CREATE TABLE students ...

2. В определении таблицы «Успеваемость» (progress) на атрибуты term и mark наложены как ограничения CHECK, так и ограничение NOT 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 NULL, и попробуйте добавить в нее строку с отсутствующим значением атрибута term (или mark).

3. 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\_num и столбца doc\_ser. Подобные вещи возможны, так как NULL-значения не считаются совпадающими. Это можно проверить с помощью команды SELECT (null = null); Она даст такой результат (т. е. NULL): ?column? ---------- (1 строка)

4. 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 и ON UPDATE SET DEFAULT. Для получения информативной картины 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, 137 Глава 5. Основы языка определения данных а во-вторых, 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.

5. В таблице «Студенты» (students) есть текстовый атрибут name, на который наложено ограничение NOT NULL. Как вы думаете, 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 ( 12300, '', 0402, 543281 ); 138 Контрольные вопросы и задания Наверное, проектируя эту таблицу, мы хотели бы все же, чтобы пустые строки в качестве значения атрибута name не проходили в базу данных? Какое решение вы можете предложить? Видимо, нужно добавить ограничение CHECK для столбца name. Если вы еще не изучили команду ALTER TABLE, то удалите таблицу students и создайте ее заново с учетом нового ограничения, а если вы уже познакомились с командой ALTER TABLE, то сделайте так: ALTER TABLE students ADD CHECK ( name <> '' ); Добавив ограничение, попробуйте теперь вставить в таблицу students строку (row), в которой значение атрибута name было бы пустой строкой (string). Давайте продолжим эксперименты и предложим в качестве значения атрибута name строку, содержащую сначала один пробел, а потом — два пробела. INSERT INTO students VALUES ( 12346, ' ', 0406, 112233 ); INSERT INTO students VALUES ( 12347, ' ', 0407, 112234 ); Для того чтобы «увидеть» эти пробелы в выборке, сделаем так: SELECT \*, length( name ) FROM students; Оказывается, эти невидимые значения имеют ненулевую длину. Что делать, чтобы не допустить таких значений-невидимок? Один из способов: возложить проверку таких ситуаций на прикладную программу. А что можно сделать на уровне определения таблицы students? Какое ограничение нужно предложить? В разделе 9.4 документации «Строковые функции и операторы» есть функция trim. Попробуйте воспользоваться ею. Если вы еще не изучили команду ALTER TABLE, то удалите таблицу students и создайте ее заново с учетом нового ограничения, а если уже познакомились с ней, то сделайте так: ALTER TABLE students ADD CHECK (...); Есть ли подобные слабые места в таблице «Успеваемость» (progress)?

6. В таблице «Рейсы» (flights) есть ограничение, которое регулирует соотношения значений фактического времени вылета и фактического времени прилета. Как вы думаете, не является ли выражение actual\_arrival IS NOT NULL во второй части условного оператора OR избыточным? CREATE TABLE flights ( ... CHECK ( actual\_arrival IS NULL OR ( actual\_departure IS NOT NULL AND actual\_arrival IS NOT NULL AND actual\_arrival > actual\_departure ) ), ... Проверьте ваши предположения на практике. Для этого сначала удалите существующее ограничение с помощью команды ALTER TABLE flights DROP CONSTRAINT имя-ограничения; Как определить имя этого ограничения? С помощью команды \d flights получите описание таблицы flights, а в нем есть названия всех ограничений. Затем создайте это же ограничение, но в модифицированном виде: ALTER TABLE flights ADD CHECK ( actual\_arrival IS NULL OR ( actual\_departure IS NOT NULL AND actual\_arrival > actual\_departure ) ); Попробуйте добавить в таблицу flights две-три строки, подбирая такие значения атрибутов actual\_departure и actual\_arrival, чтобы проверить все возможные исходы этих проверок. Конечно, вместо добавления новых строк можно модифицировать одну и ту же строку с помощью команды UPDATE.

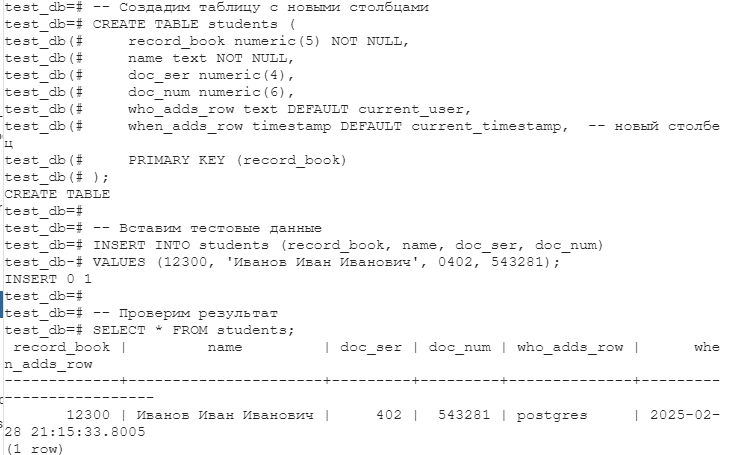
7. И представление «Рейсы» (flights\_v), и материализованное представление «Маршруты» (routes) построены на основе таблиц «Рейсы» (flights) и «Аэропорты» (airports). Логично предположить, что при каскадном удалении, например, таблицы «Аэропорты», представление «Рейсы» будет также удалено, поскольку при удалении базовой таблицы этому представлению просто неоткуда будет брать данные. А что вы можете предположить насчет материализованного представления «Маршруты»: будет ли оно также удалено или нет? Ведь оно уже содержит данные, в отличие от обычного представления. Так ли, условно говоря, сильна его связь с таблицами, на основе которых оно сконструировано? Проведите необходимые эксперименты, начав с команды DROP TABLE airports; Если вам потребуется восстановить все объекты базы данных, то вы всегда сможете воспользоваться файлом demo\_small.sql и просто повторить процедуру развертывания учебной базы данных, которая описана в главе 2. Поэтому смело экспериментируйте с таблицами и представлениями.

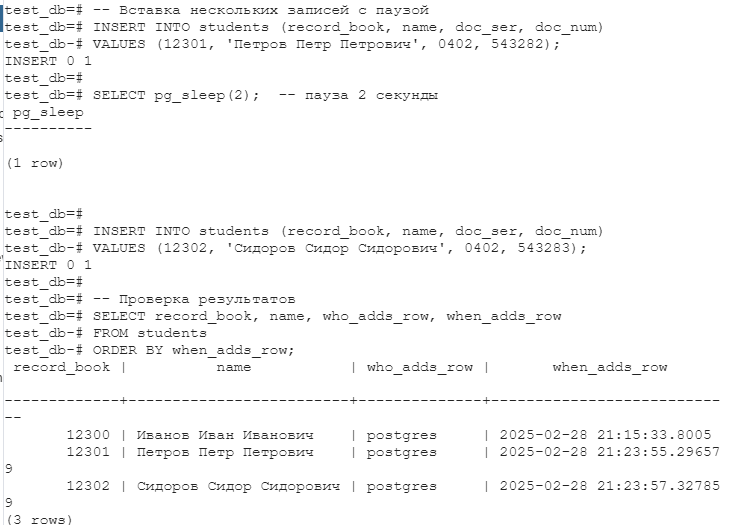
8. Определение таблицы можно изменить с помощью команды ALTER TABLE. Аналогичные команды существуют и для изменения представлений и материализованных представлений: ALTER VIEW и ALTER MATERIALIZED VIEW. Самостоятельно ознакомьтесь с их возможностями с помощью документации.

9. Представления могут быть, условно говоря, вертикальными и горизонтальными. При создании вертикального представления в список его столбцов включается лишь часть столбцов базовой таблицы (таблиц). Например: CREATE VIEW airports\_names AS SELECT airport\_code, airport\_name, city FROM airports; SELECT \* FROM airports\_names; В горизонтальное представление включаются не все строки базовой таблицы (таблиц), а производится их отбор с помощью фраз WHERE или HAVING. Например: CREATE VIEW siberian\_airports AS SELECT \* FROM airports WHERE city = 'Новосибирск' OR city = 'Кемерово'; SELECT \* FROM siberian\_airports; Конечно, вполне возможен и смешанный вариант, когда ограничивается как список столбцов, так и множество строк при создании представления. Подумайте, какие представления было бы целесообразно создать для нашей базы данных «Авиаперевозки». Необходимо учесть наличие различных групп пользователей, например: пилоты, диспетчеры, пассажиры, кассиры. Создайте представления и проверьте их в работе.

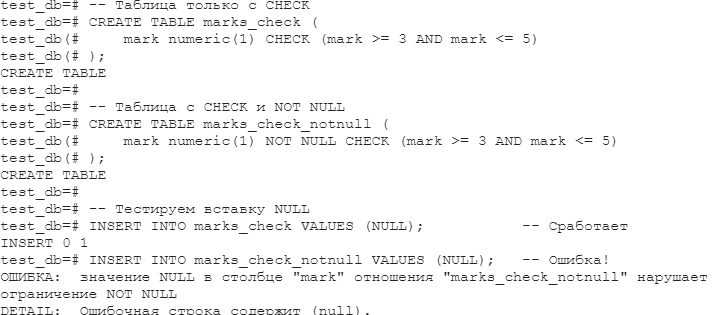
1. Ход выполнения

Выполнил пункт 1 задач.

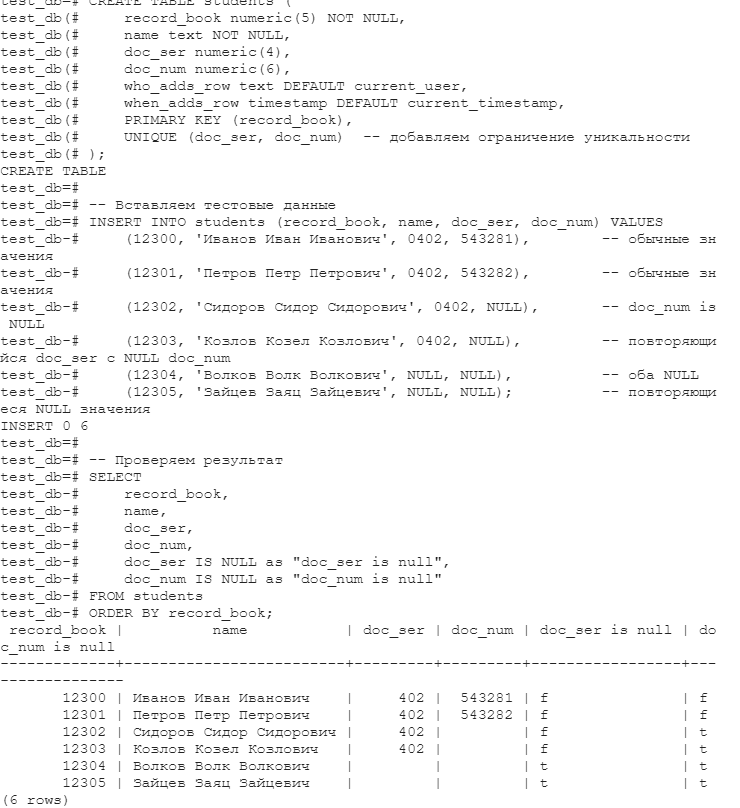


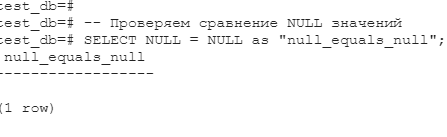


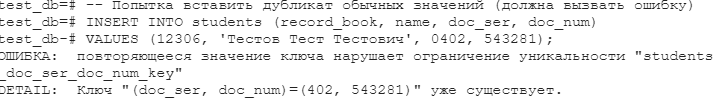
Выполнил пункт 2 задач. Нет, ограничение NOT NULL не является избыточным, даже при наличии ограничения CHECK.



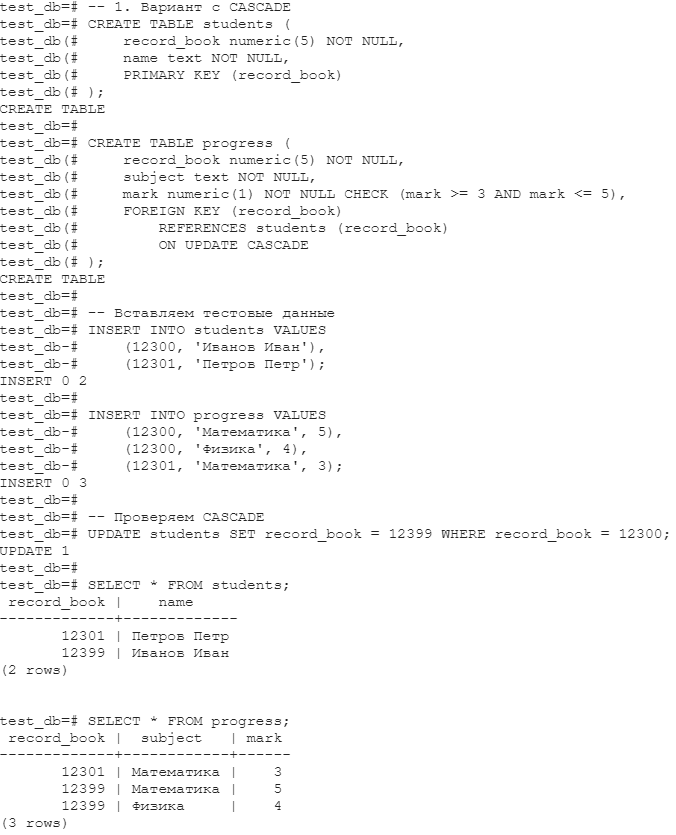
Выполнил пункт 3 задач

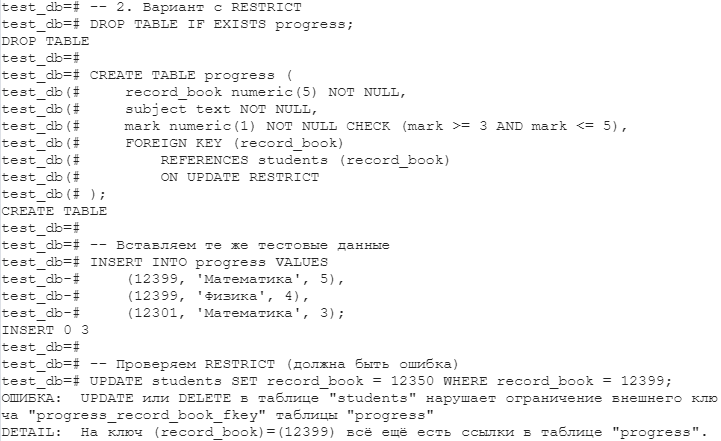


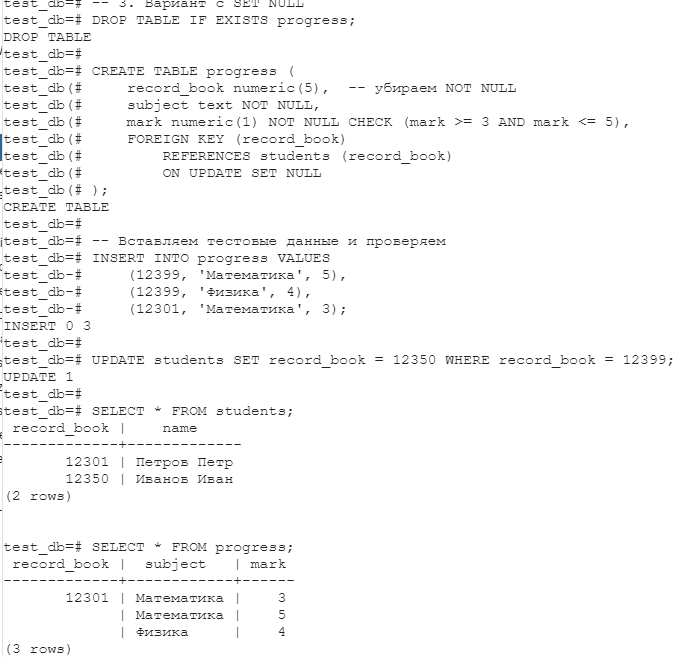




Выполнил пункт 4 задач

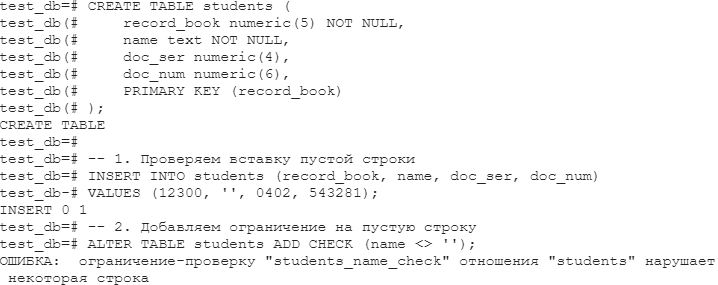


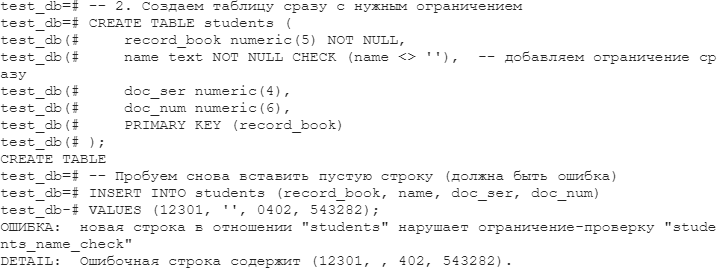


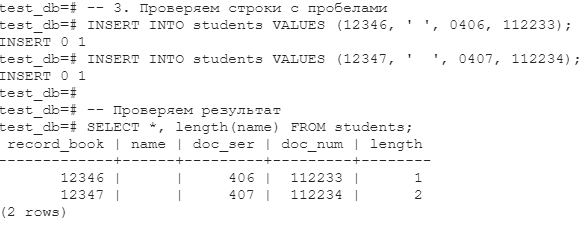


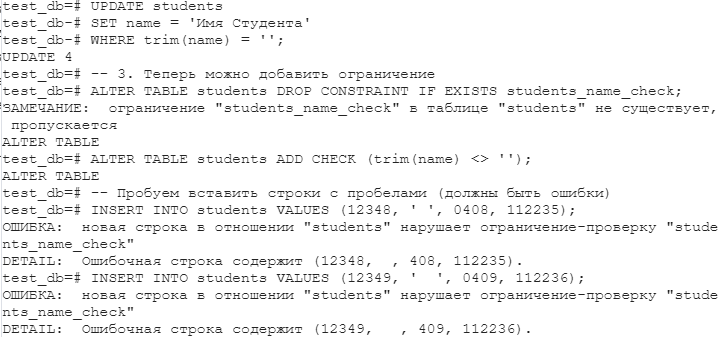
Вариант с default не подходит, так как нет логичного дефолтного значения для зачётной книжки, и в случае обновления таблиц данные 1 студента присвоятся другому

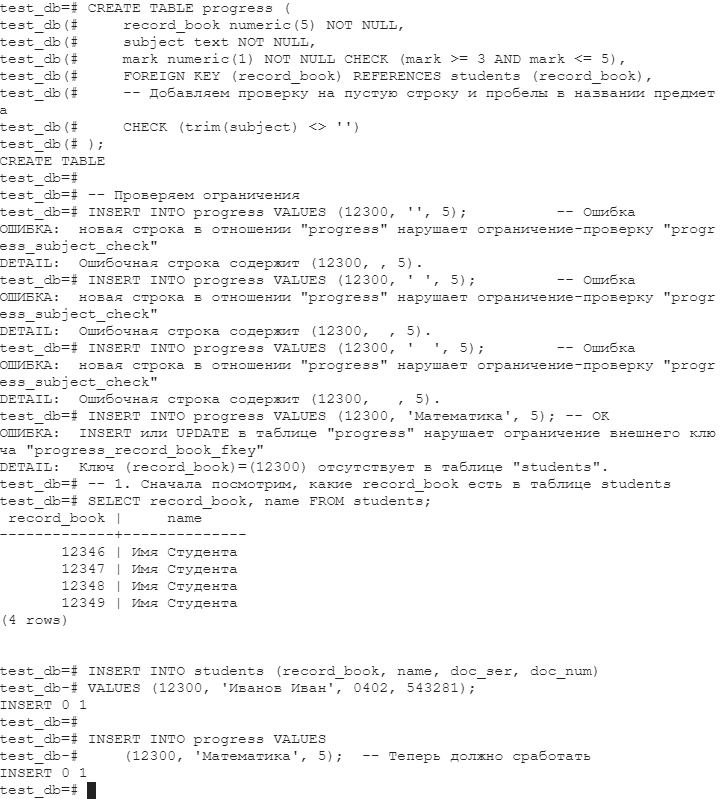
Выполнил пункт 5 задач



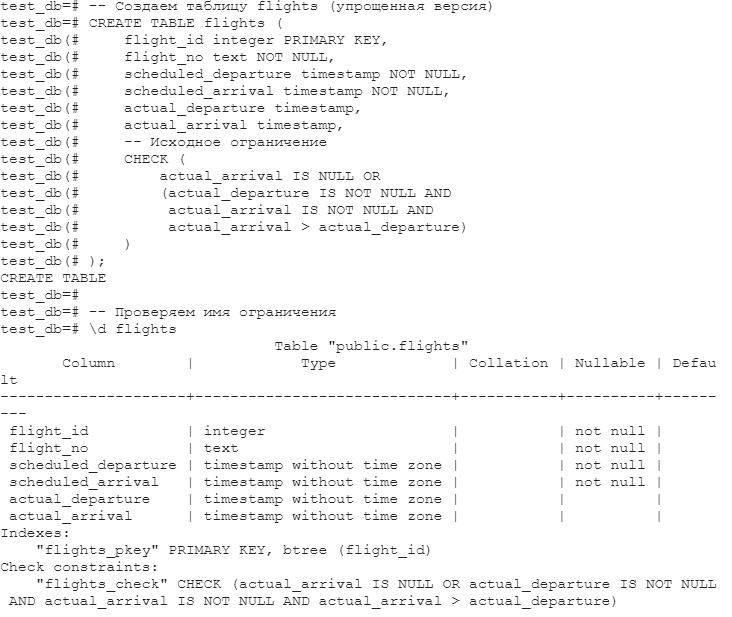


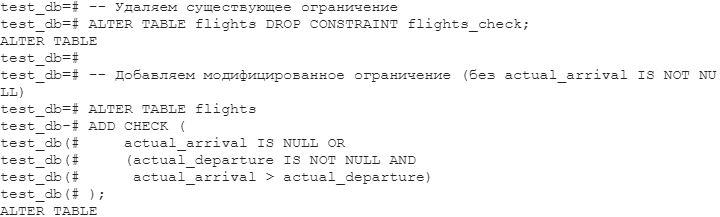


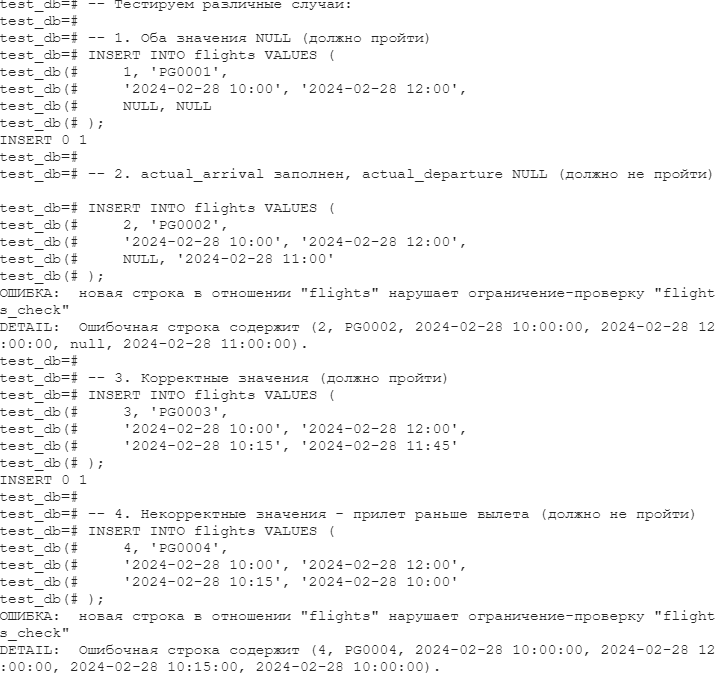




Выполнил пункт 6 задач

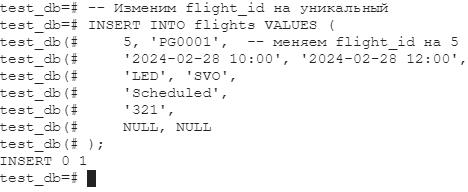






Выполнил пункт 7 задач

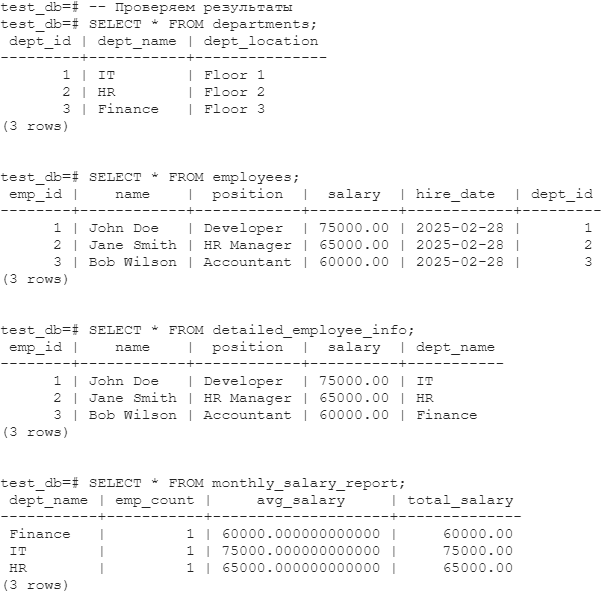




Выполнил пункт 8 задач

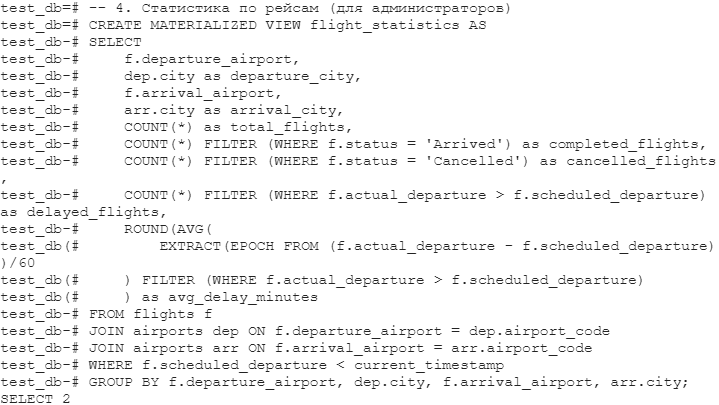


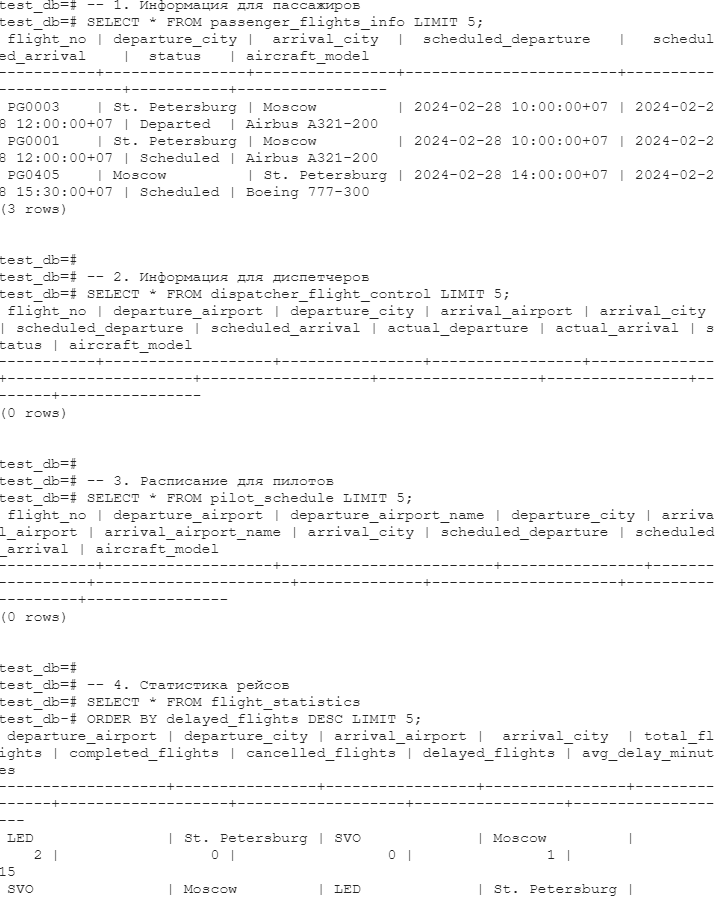




Выполнил пункт 9 задач







Особенности этих представлений:

* passenger\_flights\_info:
* Показывает только активные рейсы
* Использует понятные названия городов
* Включает модель самолета
* dispatcher\_flight\_control:
* Показывает все детали рейсов текущего дня
* Включает фактическое время вылета/прилета
* Содержит статус рейса
* pilot\_schedule:
* Показывает детальную информацию об аэропортах
* Включает только будущие рейсы
* Содержит информацию о модели самолета
* flight\_statistics:
* Материализованное представление для быстрого доступа
* Содержит статистику по задержкам и отменам
* Группирует данные по направлениям

Эти представления:

* Используют только существующие таблицы
* Предоставляют релевантную информацию для каждой группы пользователей
* Скрывают сложность базовых запросов
* Улучшают безопасность через ограничение доступа к данным

4 Выводы

Выполнив задание, были изучены основы языка определения данных.