МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**Кафедра ИИТ**

ОТЧЁТ

По лабораторной работе №9, 10, 11

«Реализация операций управления структурами баз данных»

Выполнил:  
Студент группы ИИ-24  
Крейдич А.А.   
Проверил:  
Козик И. Д.

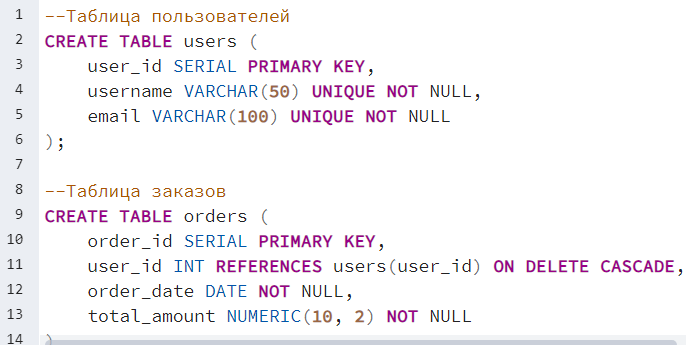
**Цель работы:** сформулировать запросы для создания, удаления и модификации структуры базы данных.

**Ход работы:**

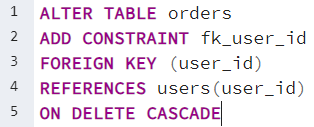
1. Создадим БД:

CREATE DATABASE kurashii24

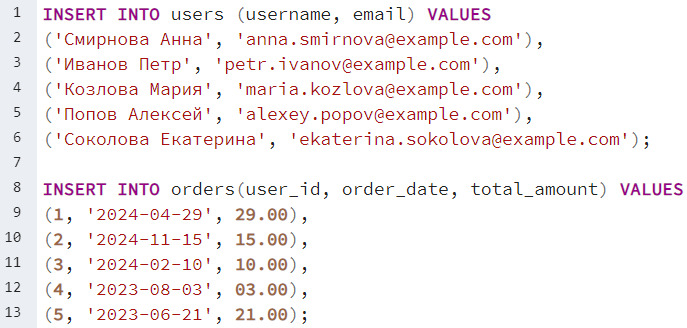
1. Создадим таблицы:



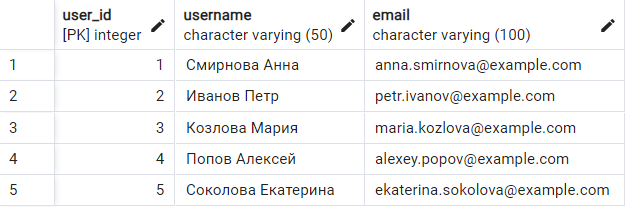
1. Установим возможность каскадного удаления данных.

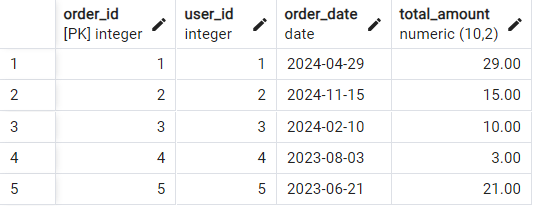


1. Заполним таблицы:

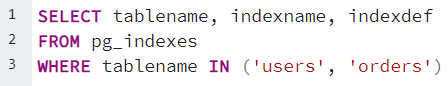


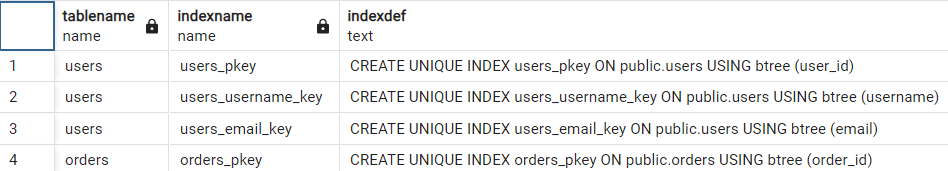
Данные в таблицах:





1. Для столбцов первичных ключей автоматически создаются кластеризованные индексы.





1. Создадим некластеризованные индексы



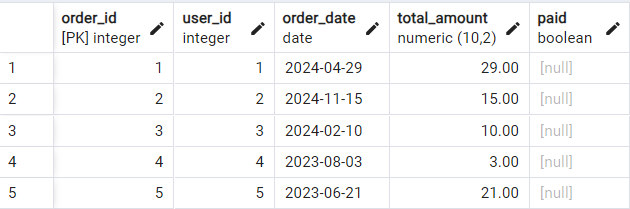
1. Реализуем простые действия с помощью оператора ALTER TABLE по изменению структуры таблицы (не менее трех) и свойств столбцов.

*Изменение типа столбца:*



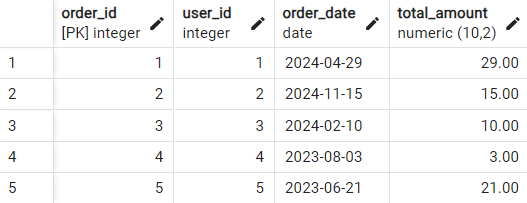
*Добавление нового столбца:*





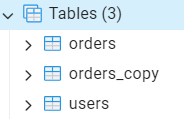
*Удаление столбца:*





1. Создадим копию таблицы заказов:



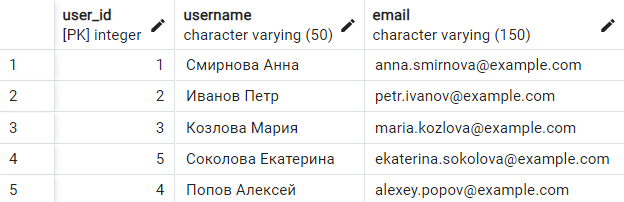


Удалим таблицу:



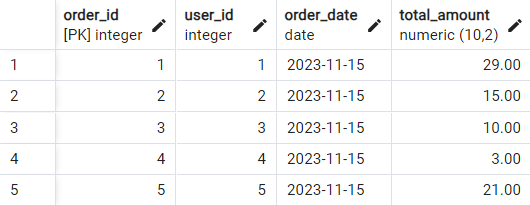
1. Создадим запрос на изменение с применением WHERE:





Без применения WHERE:





Код программы:

import sqlite3

import random

# Создаём/подключаемся к базе данных в памяти (можно заменить ':memory:' на имя файла)

conn = sqlite3.connect(':memory:')

cur = conn.cursor()

# ---------------------------------------

# Часть 1. Создаём таблицу users

# ---------------------------------------

cur.execute("""

DROP TABLE IF EXISTS users

""")

cur.execute("""

CREATE TABLE users (

user\_id INTEGER PRIMARY KEY,

username TEXT,

email TEXT

)

""")

# Вставляем данные (user\_id задаём вручную, чтобы соответствовало примеру)

users\_data = [

(1, "Смирнова Анна", "anna.smirnova@example.com"),

(2, "Иванов Петр", "petr.ivanov@example.com"),

(3, "Козлова Мария", "maria.kozlova@example.com"),

(4, "Соколова Екатерина", "ekaterina.sokolova@example.com"),

(5, "Попов Алексей", "alexey.popov@example.com")

]

cur.executemany("""

INSERT INTO users (user\_id, username, email)

VALUES (?, ?, ?)

""", users\_data)

conn.commit()

# Добавляем столбец «Примечание» (REAL) и заполняем случайными дробными значениями [50..300]

cur.execute("ALTER TABLE users ADD COLUMN Примечание REAL")

conn.commit()

cur.execute("SELECT user\_id FROM users")

rows = cur.fetchall()

for (uid,) in rows:

note\_value = round(random.uniform(50, 300), 2)

cur.execute("""

UPDATE users

SET Примечание = ?

WHERE user\_id = ?

""", (note\_value, uid))

conn.commit()

# Создаём представления (views) для Задания 1

# 1) Агрегатные функции (MIN, MAX, AVG, SUM) с двумя знаками после запятой

cur.execute("""

CREATE VIEW view\_aggregates AS

SELECT

printf('%.2f', MIN(Примечание)) AS min\_note,

printf('%.2f', MAX(Примечание)) AS max\_note,

printf('%.2f', AVG(Примечание)) AS avg\_note,

printf('%.2f', SUM(Примечание)) AS sum\_note

FROM users

""")

# 2) Подсчёт количества строк, где «Примечание» <= MIN(Примечание)+50

cur.execute("""

CREATE VIEW view\_count AS

SELECT

COUNT(\*) AS count\_rows

FROM users

WHERE Примечание <= (

SELECT MIN(Примечание) FROM users

) + 50

""")

# 3) Выбор строк, где user\_id <= (AVG(Примечание)/100)

cur.execute("""

CREATE VIEW view\_filtered AS

SELECT \*

FROM users

WHERE user\_id <= (

SELECT AVG(Примечание) FROM users

) / 100

""")

# Проверим вывод из созданных представлений

print("=== Задание 1: Представление view\_aggregates ===")

for row in cur.execute("SELECT \* FROM view\_aggregates"):

print(row)

print("\n=== Задание 1: Представление view\_count ===")

for row in cur.execute("SELECT \* FROM view\_count"):

print(row)

print("\n=== Задание 1: Представление view\_filtered ===")

for row in cur.execute("SELECT \* FROM view\_filtered"):

print(row)

# ---------------------------------------

# Часть 2. Цепочка из четырёх таблиц

# ---------------------------------------

# Удаляем, если уже есть

cur.execute("DROP TABLE IF EXISTS tableD")

cur.execute("DROP TABLE IF EXISTS tableC")

cur.execute("DROP TABLE IF EXISTS tableB")

cur.execute("DROP TABLE IF EXISTS tableA")

# Создаём таблицы: tableA -> tableB -> tableC -> tableD

cur.execute("""

CREATE TABLE tableA (

a\_id INTEGER PRIMARY KEY AUTOINCREMENT,

dataA TEXT

)

""")

cur.execute("""

CREATE TABLE tableB (

b\_id INTEGER PRIMARY KEY AUTOINCREMENT,

a\_id INTEGER,

dataB TEXT,

FOREIGN KEY(a\_id) REFERENCES tableA(a\_id)

)

""")

cur.execute("""

CREATE TABLE tableC (

c\_id INTEGER PRIMARY KEY AUTOINCREMENT,

b\_id INTEGER,

dataC TEXT,

Примечание REAL,

FOREIGN KEY(b\_id) REFERENCES tableB(b\_id)

)

""")

cur.execute("""

CREATE TABLE tableD (

d\_id INTEGER PRIMARY KEY AUTOINCREMENT,

c\_id INTEGER,

dataD TEXT,

FOREIGN KEY(c\_id) REFERENCES tableC(c\_id)

)

""")

# Заполняем таблицы тестовыми данными

cur.execute("INSERT INTO tableA (dataA) VALUES ('A\_item1')")

cur.execute("INSERT INTO tableA (dataA) VALUES ('A\_item2')")

cur.execute("INSERT INTO tableA (dataA) VALUES ('A\_item3')")

cur.execute("INSERT INTO tableB (a\_id, dataB) VALUES (1, 'B\_item1')")

cur.execute("INSERT INTO tableB (a\_id, dataB) VALUES (2, 'B\_item2')")

cur.execute("INSERT INTO tableB (a\_id, dataB) VALUES (3, 'B\_item3')")

# В tableC вставляем случайные Примечание в диапазоне [50..300]

for i in range(1, 4):

note\_val = round(random.uniform(50, 300), 2)

dataC\_val = f"C\_item{i}"

cur.execute("""

INSERT INTO tableC (b\_id, dataC, Примечание)

VALUES (?, ?, ?)

""", (i, dataC\_val, note\_val))

cur.execute("INSERT INTO tableD (c\_id, dataD) VALUES (1, 'D\_item1')")

cur.execute("INSERT INTO tableD (c\_id, dataD) VALUES (2, 'D\_item2')")

cur.execute("INSERT INTO tableD (c\_id, dataD) VALUES (3, 'D\_item3')")

conn.commit()

# Вычислим среднее Примечание в tableC (для наглядности)

cur.execute("SELECT AVG(Примечание) FROM tableC")

avg\_note = cur.fetchone()[0]

print(f"\nСреднее значение Примечание в tableC: {avg\_note:.2f}\n")

# Задание 2.1: Запрос с оператором JOIN

join\_query = """

SELECT

A.dataA AS a\_data,

D.dataD AS d\_data,

C.Примечание AS c\_note

FROM tableA A

JOIN tableB B ON A.a\_id = B.a\_id

JOIN tableC C ON B.b\_id = C.b\_id

JOIN tableD D ON C.c\_id = D.c\_id

WHERE C.Примечание >= (SELECT AVG(Примечание) FROM tableC)

"""

print("=== Задание 2 (JOIN) ===")

for row in cur.execute(join\_query):

print(row)

# Задание 2.2: Запрос с подзапросами (без JOIN)

subquery = """

SELECT

A.dataA AS a\_data,

D.dataD AS d\_data,

(SELECT Примечание

FROM tableC

WHERE tableC.c\_id = D.c\_id

) AS c\_note

FROM tableA A, tableD D

WHERE A.a\_id IN (

SELECT b.a\_id

FROM tableB b

WHERE b.b\_id IN (

SELECT c.b\_id

FROM tableC c

WHERE c."Примечание" >= (SELECT AVG(Примечание) FROM tableC)

AND c.c\_id IN (SELECT d.c\_id FROM tableD d)

)

)

AND D.c\_id IN (

SELECT c.c\_id

FROM tableC c

WHERE c."Примечание" >= (SELECT AVG(Примечание) FROM tableC)

)

"""

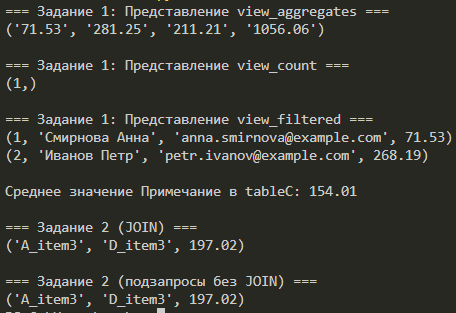
print("\n=== Задание 2 (подзапросы без JOIN) ===")

for row in cur.execute(subquery):

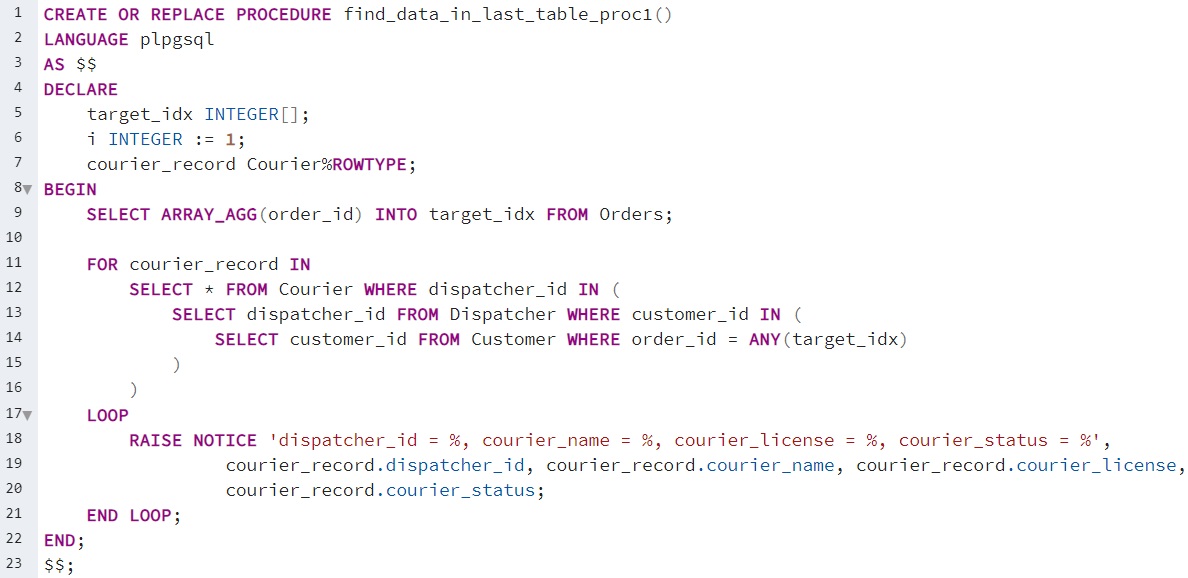
print(row)

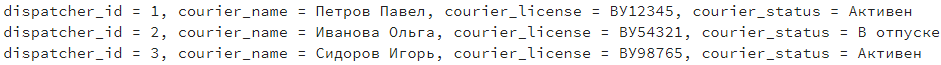
conn.close()

Console:

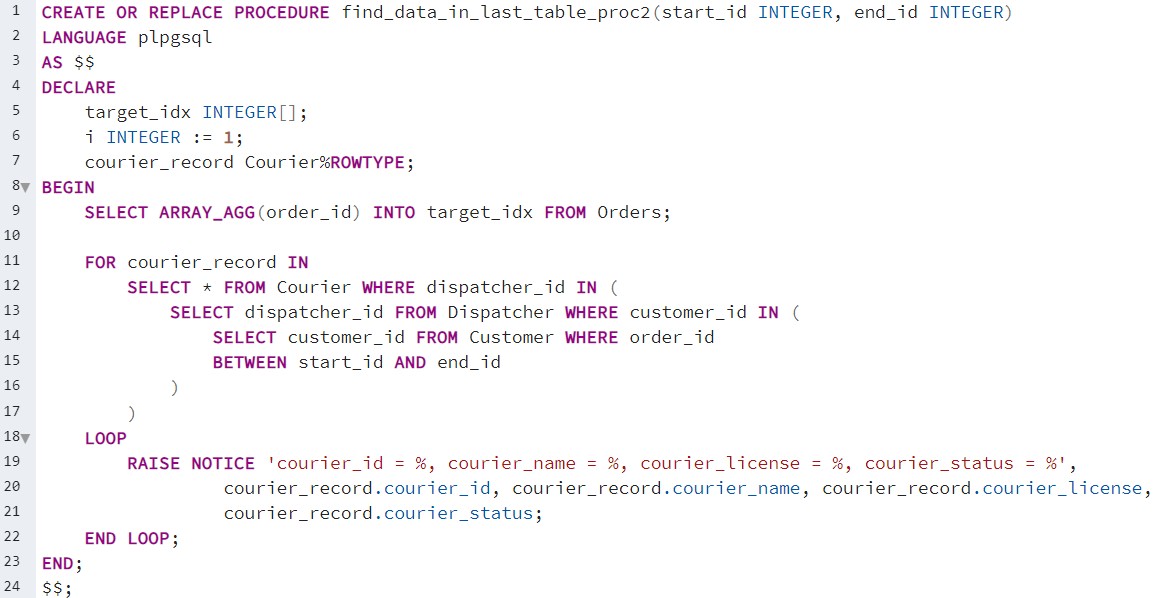


Процедура 1 – без входных параметров:



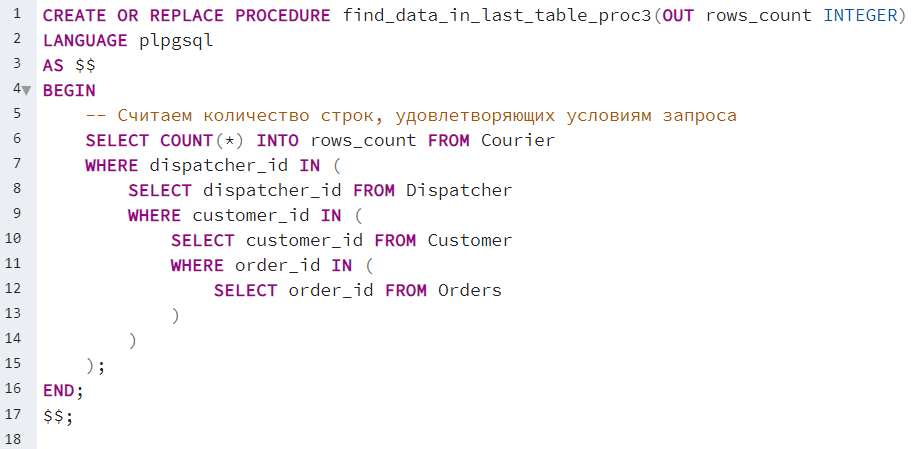


Процедура 2 – С входными параметрами



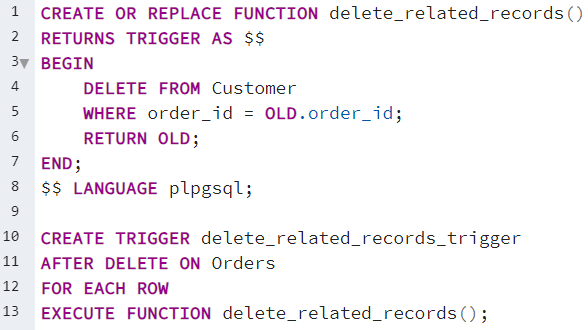


Процедура 3 – С выходным параметром.

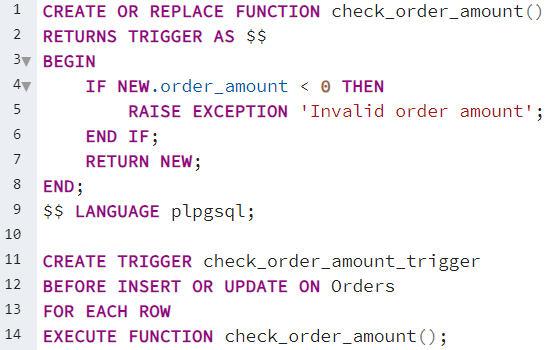




Триггер на удаление:



Триггер на изменение и вставку:





**Вывод:** сформулировал запросы для создания, удаления и модификации структуры базы данных.