МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра защиты информации

По расчетно-графическому заданию
по дисциплине: «Безопасность систем баз данных**»**
на тему «Работа с транзакциями**»**
Вариант: «Курьерская служба»

Выполнил:
студент гр. АБ-120, АВТФ
Плешков К.А.

«___»_____2023 г.

_____

(подпись)

Проверил:
ассистент кафедры ЗИ
Питько Я.А.

«___»_____20__г.

_____

(подпись)

Новосибирск
2023

Содержание

Цель работы: получить практические навыки использования транзакций в СУБД PostgreSQL.

Основные положения:

В рамках данного расчетно-графического задания предполагается получить практические навыки использования транзакций в СУБД PostgreSQL.

Основываясь на изменении и удалении полей (столбцов) и данных (записей), а также сконструированных по варианту БД, необходимо создать три транзакции, включающие взаимосвязанные между собой операции.

- Транзакция фиксируется;
- Изменения транзакции откатываются;
- Транзакция содержит точку сохранения в текущей транзакции и фиксирует только часть изменений.

Результаты транзакций до/после (для двух последних транзакций - до/во время выполнения транзакции/после отката транзакции) в данном расчетно-графическом задании необходимо прилагать к отчёту, составленному в формате *.pdf.

Задачи:

1. Создать ветку от master(main) с названием “rgz”. Все изменения в рамках данного расчетно-графического задания производить в этой ветке.
2. Дополнить скрипт инициализации транзакциями с взаимосвязанными командами по соответствующему варианту.
3. При необходимости, дополнить недостающими сущностями существующую структуру БД.
4. Составить отчет по расчетно-графическому заданию, который содержит следующие элементы: титульник с названием расчетно-графического задания и вариантом студента, содержание, включающее все предыдущие четыре лабораторные работы по названию и раздел работы с транзакциями, объединенные 1-4 лабораторные работы с описанием созданной структуры БД, включая обязательные таблицы по варианту,

таблицы-справочники, составную таблицу, SELECT-запросы, обернутые в функции, а также созданные в данной работе транзакции, вывод.

5. После окончания работы, создать merge request ветки с лабораторной работой в master(main), добавить преподавателя в reviewer. Не сливать без аппрува от преподавателя!

6. В Диспейсе прикрепить ссылку на репозиторий и отчёт в *.pdf формате.

Содержание:

Поднятие сервера PostgreSQL с помощью Docker:

Настройка gitlab:

- Создали профиль и проект на gitlab
- Настроили ssh ключи
- С клонировали пустой проект.

Создали docker-compose.yml:

```yaml
version: "3.9"
services:
  postgres:
    image: postgres:16
    environment:
      POSTGRES_DB: "Pleshkov"
      POSTGRES_USER: "root"
      POSTGRES_PASSWORD: "root"
    volumes:
      - .:/docker-entrypoint-initdb.d
    ports:
      - "8080:5432"
```

Рисунок 1 - docker-compose.yml.

После создаем .sql файла для инициализации БД. В нем создадим 3 таблицы описывающие сборки ПК для магазина:

```sql
CREATE TABLE Components (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    type VARCHAR(255),
    price DECIMAL(10, 2)
);

CREATE TABLE Computers (
    id SERIAL PRIMARY KEY,
    model VARCHAR(255),
    processor VARCHAR(255),
    memory INTEGER,
    component_id INTEGER REFERENCES Components(id)
);

CREATE TABLE Orders (
    id SERIAL PRIMARY KEY,
    date DATE,
    computer_id INTEGER REFERENCES Computers(id)
);
```

Рисунок 2 – создание таблиц.

С помощью команды docker-compose up поднимаем контейнер.

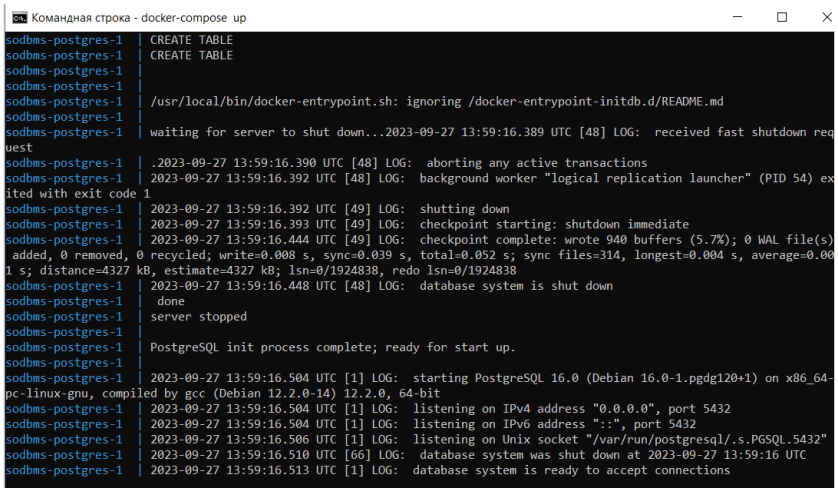

Рисунок 3 – выполнение команды docker-compose up.

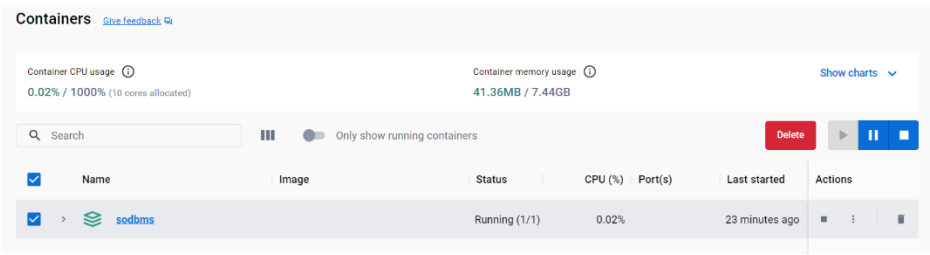

Рисунок 4 – поднятый контейнер в докере.

Проверим подключение с помощью pgAdmin.

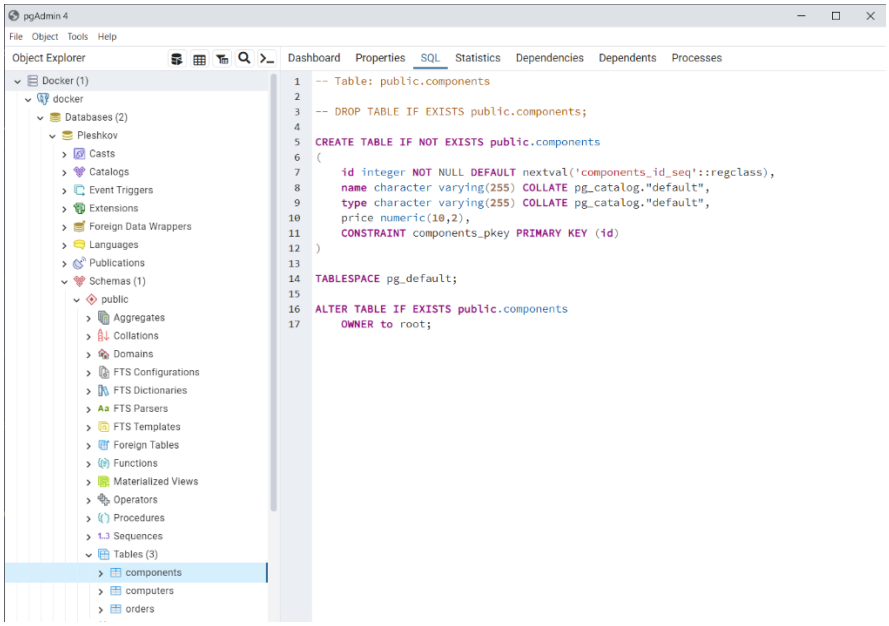

Рисунок 5 – pgAdmin.

Работа с данными в таблицах:

Создаем таблицы по указанному варианту, добавив необходимые таблицы «справочники».

```
CREATE TABLE TravelMethods (
  method_of_travel_id SERIAL PRIMARY KEY,
  method_of_travel VARCHAR(30) NOT NULL
);

CREATE TABLE Positions (
  position_id SERIAL PRIMARY KEY,
  position VARCHAR(60) NOT NULL
);

CREATE TABLE PhoneNumbers (
  phone_number_id SERIAL PRIMARY KEY,
  phone_number VARCHAR(11) NOT NULL
);

CREATE TABLE Couriers (
  courier_id SERIAL PRIMARY KEY,
  full_name VARCHAR(60) NOT NULL,
  birth_date DATE NOT NULL,
  method_of_travel_id INT NOT NULL,
  phone_number_id INT NOT NULL,
  FOREIGN KEY (method_of_travel_id) REFERENCES TravelMethods (method_of_travel_id),
  FOREIGN KEY (phone_number_id) REFERENCES PhoneNumbers (phone_number_id)
);

CREATE TABLE Employees (
  employee_id SERIAL PRIMARY KEY,
  full_name VARCHAR(60) NOT NULL,
  location VARCHAR(80) NOT NULL,
  position_id INT NOT NULL,
  FOREIGN KEY (position_id) REFERENCES Positions (position_id)
);
```

```sql
CREATE TABLE Orders (
  order_id SERIAL PRIMARY KEY,
  order_date DATE NOT NULL,
  distance_to_addressee DECIMAL(8,1) NOT NULL
);

CREATE TABLE Points (
  point_id SERIAL PRIMARY KEY,
  address VARCHAR(255) NOT NULL,
  is_oversized_cargo_service BOOLEAN NOT NULL,
  phone_number_id INT NOT NULL,
  effective_storage_area DECIMAL(8,1) NOT NULL,
  FOREIGN KEY (phone_number_id) REFERENCES PhoneNumbers (phone_number_id)
);
```

Рисунки 6-7 – создание таблиц.

После этого заполним полученные таблицы.

```sql
INSERT INTO TravelMethods (method_of_travel)
VALUES
  ('Автомобиль'),
  ('Самокат'),
  ('Пеший');

INSERT INTO Positions (position)
VALUES
  ('Менеджер'),
  ('Продавец'),
  ('Администратор');

INSERT INTO PhoneNumbers (phone_number)
VALUES
  ('12345678901'),
  ('23456789012'),
  ('34567890123'),
  ('45678901234'),
  ('56789012345'),
  ('67890123456'),
  ('78901234567'),
  ('89012345678'),
  ('90123456789'),
  ('01234567890');
```

```sql
INSERT INTO Couriers (full_name, birth_date, method_of_travel_id, phone_number_id)
VALUES
  ('Иван Иванов', '1990-01-01', 1, 1),
  ('Алексей Петров', '1992-05-15', 2, 2),
  ('Елена Смирнова', '1985-11-30', 1, 3),
  ('Мария Ковалева', '1993-07-20', 3, 4),
  ('Андрей Васильев', '1988-03-10', 2, 5),
  ('Ольга Николаева', '1991-09-25', 1, 6),
  ('Дмитрий Соколов', '1987-12-05', 3, 7),
  ('Сергей Морозов', '1994-04-18', 2, 8),
  ('Наталья Волкова', '1989-08-12', 1, 9),
  ('Павел Лебедев', '1995-02-28', 3, 10);

INSERT INTO Employees (full_name, location, position_id)
VALUES
  ('Анна Сидорова', 'Москва', 1),
  ('Александр Иванов', 'Санкт-Петербург', 2),
  ('Екатерина Козлова', 'Екатеринбург', 3),
  ('Игорь Павлов', 'Новосибирск', 1),
  ('Марина Смирнова', 'Казань', 2),
  ('Артем Петров', 'Владивосток', 3),
  ('Ольга Морозова', 'Ростов-на-Дону', 1),
  ('Денис Васильев', 'Самара', 2),
  ('Николай Ковалев', 'Омск', 3),
  ('Татьяна Николаева', 'Красноярск', 1);
```

```sql
INSERT INTO Orders (order_date, distance_to_addressee)
VALUES
  ('2023-01-01', 10.5),
  ('2023-02-15', 5.2),
  ('2023-03-30', 8.7),
  ('2023-04-10', 12.3),
  ('2023-05-20', 6.8),
  ('2023-06-05', 9.1),
  ('2023-07-18', 7.4),
  ('2023-08-25', 11.2),
  ('2023-09-10', 4.9),
  ('2023-10-28', 14.6);

INSERT INTO Points (address, is_oversized_cargo_service, phone_number_id, effective_storage_area)
VALUES
  ('Московская улица, 1', false, 1, 100.5),
  ('Ленинградский проспект, 2', true, 2, 150.2),
  ('Сибирский тракт, 3', false, 3, 80.7),
  ('Красная площадь, 4', true, 4, 200.1),
  ('Невский проспект, 5', false, 5, 120.8),
  ('Уральская улица, 6', true, 6, 180.3),
  ('Волгоградский проспект, 7', false, 7, 90.6),
  ('Казанская улица, 8', true, 8, 160.9),
  ('Омская улица, 9', false, 9, 110.4),
  ('Тихоокеанская улица, 10', true, 10, 140.0);
```

Рисунки 8-10 – заполнение базы данных.

После этого добавим необходимые команды.

```sql
-- Задание 1 добавляем столбец полный ли рабочий день у курьера
ALTER TABLE Couriers
ADD COLUMN is_full_time_working BOOLEAN;

-- Задание 2 меняем должность работников
UPDATE Employees
SET location = 'Казань'
WHERE location = 'Ленск';

-- Задание 3 удаляем столбец
ALTER TABLE Couriers
DROP COLUMN is_full_time_working;

-- Задание 4 удаляем пункт по адресу
DELETE FROM Points
WHERE address = 'Московская улица, 1';
```

Рисунок 11 – команды.

Также наложим ограничения на необходимые поля.

```
--Ограничения

--Ограничение на дату рождения
ALTER TABLE Couriers
ADD CONSTRAINT check_birth_date CHECK (birth_date >= '1900-01-01' AND birth_date <= CURRENT_DATE);

--Ограничение на дату заказа
ALTER TABLE Orders
ADD CONSTRAINT order_date CHECK (order_date <= CURRENT_DATE);

--Ограничение на расстояние до адресата
ALTER TABLE Orders
ADD CONSTRAINT distance_to_addressee CHECK (distance_to_addressee > 0);

--Ограничение на эффективную площадь хранения
ALTER TABLE Points
ADD CONSTRAINT effective_storage_area CHECK (effective_storage_area > 0);
```

Рисунок 12 – ограничения на поля.

Проверим работу ограничений. Попробуем добавить курьера с датой рождения позже текущей даты.

	courier_id [PK] integer	full_name character varying (60)	birth_date date	method_of_travel character varying (30)	phone_number character varying (11)
1+	[default]	Кирилл Плешков	2024-01-01	Автомобиль	89133248281

Рисунок 13 – попытка добавить запись.

Failing row contains (12, Кирилл Плешков, 2024-01-01, Автомобиль, 89133248281).new row for relation "couriers" violates check constraint "check_birth_date"

Рисунок 14 – полученная ошибка.

Попробуем создать заказ с расстоянием до адресата меньше нуля.

	order_id [PK] integer	order_date date	distance_to_addressee numeric (8,1)
1+	[default]	2023-10-11	-5

Рисунок 15 – попытка добавить запись.

Failing row contains (11, 2023-10-11, -5.0).new row for relation "orders" violates check constraint "distance_to_addressee"

Рисунок 16 – полученная ошибка.

Работа с выводом данных:

Перед созданием составной таблицы, нужно связать таблицы: Employees, Points и Orders. Свяжем их через таблицу Orders, добавив данные строки:

employee_id INT NOT NULL,

point_id INT NOT NULL,

FOREIGN KEY (employee_id) REFERENCES Employees (employee_id),

FOREIGN KEY (point_id) REFERENCES Points (point_id)

После этого мы можем создать составную таблицу. Для этого воспользуемся конструкцией CREATETABLE AS.

```
--Составная таблица
CREATE TABLE Orders_by_points_and_employees(order_id, employee_full_name, point_address) AS
    SELECT o.order_id, e.full_name, p.address FROM Orders o
    JOIN Employees e ON o.employee_id = e.employee_id
    JOIN Points p ON o.point_id = p.point_id;
```

Рисунок 17 – Создание составной таблицы.

Добавим ограничения на таблицу:

```
--Ограничения на составную таблицу
ALTER TABLE Orders_by_points_and_employees ADD PRIMARY KEY (order_id);
ALTER TABLE Orders_by_points_and_employees ADD FOREIGN KEY (order_id) references Orders on delete cascade;
```

Рисунок 18 – Ограничения таблицы.

Посмотрим в pgAdmin на полученную таблицу:

	order_id [PK] integer	employee_full_name character varying (60)	point_address character varying (255)
1	1	Анна Сидорова	Московская улица, 1
2	2	Анна Сидорова	Ленинградский проспект, 2
3	3	Александр Иванов	Сибирский тракт, 3
4	4	Анна Сидорова	Сибирский тракт, 3
5	5	Марина Смирнова	Невский проспект, 5
6	6	Екатерина Козлова	Сибирский тракт, 3
7	7	Игорь Павлов	Московская улица, 1
8	8	Александр Иванов	Московская улица, 1
9	9	Екатерина Козлова	Ленинградский проспект, 2
10	10	Марина Смирнова	Сибирский тракт, 3

Рисунок 19 – Составная таблица.

Составим SELECT-запросы и посмотрим на вывод:

1. Вывести все пункты, обслуживающие крупногабаритные грузы и с эффективной площадью хранения больше 400.

```
sodbms-postgres-1  |  point_id |       address       | is_oversized_cargo_service | phone_number | effective_storage_area
sodbms-postgres-1  | ----------+---------------------+----------------------------+--------------+-----------------------
sodbms-postgres-1  |         5 | Невский проспект, 5 | t                          | 56789012345  |                 1200.8
sodbms-postgres-1  |         8 | Казанская улица, 8  | t                          | 89012345678  |                 1600.9
sodbms-postgres-1  |         9 | Омская улица, 9     | t                          | 90123456789  |                 1100.4
sodbms-postgres-1  | (3 rows)
```

```
178
179  --Вывести все пункты, обслуживающие крупногабаритные грузы и с эффективной площадью хранения больше 400.
180  SELECT p.point_id, p.address, p.is_oversized_cargo_service, pn.phone_number, p.effective_storage_area
181      FROM Points p
182      JOIN PhoneNumbers pn ON p.phone_number_id = pn.phone_number_id
183      WHERE is_oversized_cargo_service = true AND effective_storage_area > 400;
```

Рисунок 20 – Первый SELECT-запрос.

2. Вывести всех работников определённого пункта и определённой должности.

```
sodbms-postgres-1  |  employee_id |     full_name     | location | position
sodbms-postgres-1  | -------------+-------------------+----------+----------
sodbms-postgres-1  |            1 | Анна Сидорова     | Москва   | Менеджер
sodbms-postgres-1  |           10 | Татьяна Николаева | Москва   | Менеджер
sodbms-postgres-1  | (2 rows)
```

```
--Вывести всех работников определённого пункта и определённой должности.
SELECT e.employee_id, e.full_name, e.location, p.position
    FROM Employees e
    JOIN Positions p ON e.position_id = p.position_id
    WHERE e.location = 'Москва' AND p.position = 'Менеджер';
```

Рисунок 21 – Второй SELECT-запрос.

3. Вывести всех пеших-курьеров старше 30 лет

```
sodbms-postgres-1  |  courier_id |    full_name    | birth_date | age | method_of_travel | phone_number
sodbms-postgres-1  | ------------+-----------------+------------+-----+------------------+-------------
sodbms-postgres-1  |           4 | Мария Ковалева  | 1992-07-20 |  31 | Пеший            | 45678901234
sodbms-postgres-1  |           7 | Дмитрий Соколов | 1987-12-05 |  35 | Пеший            | 78901234567
sodbms-postgres-1  | (2 rows)
```

```
--Вывести всех пеших-курьеров старше 30 лет.
SELECT c.courier_id, c.full_name, c.birth_date, date_part('year', AGE(c.birth_date)) AS age, t.method_of_travel, pn.phone_number
    FROM Couriers c
    JOIN TravelMethods t ON c.method_of_travel_id = t.method_of_travel_id
    JOIN PhoneNumbers pn ON c.phone_number_id = pn.phone_number_id
    WHERE t.method_of_travel = 'Пеший' AND date_part('year', AGE(c.birth_date)) > 30;
```

Рисунок 22 – Третий SELECT-запрос.

Работа с функциями:

Обернем запросы в функции на языке SQL.

1. Вывести все пункты, обслуживающие крупногабаритные грузы и с эффективной площадью хранения больше 400.

```
CREATE FUNCTION
 point_id |       address       | is_oversized_cargo_service | phone_number | effective_storage_area
----------+---------------------+----------------------------+--------------+-----------------------
        1 | Московская улица, 1 | f                          | 12345678901  |                  400.5
        3 | Сибирский тракт, 3  | f                          | 34567890123  |                  800.7
(2 rows)
```

```
--Функция для вывода пунктов по параметрам: is_oversized_cargo_service и effective_storage_area
CREATE OR REPLACE FUNCTION getPoints(is_oversized BOOLEAN, storage_area INTEGER)
RETURNS TABLE (point_id INT, address VARCHAR, is_oversized_cargo_service BOOLEAN, phone_number VARCHAR, effective_storage_area DECIMAL(8,1))
AS $$
    SELECT p.point_id, p.address, p.is_oversized_cargo_service, pn.phone_number, p.effective_storage_area
    FROM Points p
    JOIN PhoneNumbers pn ON p.phone_number_id = pn.phone_number_id
    WHERE p.is_oversized_cargo_service = is_oversized AND p.effective_storage_area > storage_area;
$$ LANGUAGE SQL;

SELECT * FROM getPoints(false, 400);
```

Рисунок 23 – Первая функция и ее вывод.

2. Вывести всех работников определённого пункта и определённой должности.

```
CREATE FUNCTION
 employee_id |     full_name     | location | positionn
-------------+-------------------+----------+-----------
           1 | Анна Сидорова     | Москва   | Менеджер
          10 | Татьяна Николаева | Москва   | Менеджер
(2 rows)
```

```
--Функция для вывода всех работников по параметрам: location и positionn
CREATE OR REPLACE FUNCTION getEmployees(loc VARCHAR, pos VARCHAR)
RETURNS TABLE (employee_id INT, full_name VARCHAR, location VARCHAR, positionn VARCHAR)
AS $$
    SELECT e.employee_id, e.full_name, e.location, p.position
    FROM Employees e
    JOIN Positions p ON e.position_id = p.position_id
    WHERE e.location = loc AND p.position = pos;
$$ LANGUAGE SQL;

SELECT * FROM getEmployees('Москва', 'Менеджер');
```

Рисунок 24 – Вторая функция и ее вывод.

3. Вывести всех пеших-курьеров старше 30 лет.

```
CREATE FUNCTION
 courier_id |    full_name    | birth_date | age | method_of_travel | phone_number
------------+-----------------+------------+-----+------------------+-------------
          4 | Мария Ковалева  | 1992-07-20 |  31 | Пеший            | 45678901234
          7 | Дмитрий Соколов | 1987-12-05 |  36 | Пеший            | 78901234567
         10 | Павел Лебедев   | 1995-02-28 |  28 | Пеший            | 01234567890
(3 rows)
```

```sql
--Функция для вывода всех курьеров по параметрам: method_of_travel и age
CREATE OR REPLACE FUNCTION getCouriers(method_of_travel_param VARCHAR, age_param INT)
RETURNS TABLE (courier_id INT, full_name VARCHAR, birth_date DATE, age INT, method_of_travel VARCHAR, phone_number VARCHAR)
AS $$
    SELECT c.courier_id, c.full_name, c.birth_date, date_part('year', AGE(c.birth_date)) AS age, t.method_of_travel, pn.phone_number
    FROM Couriers c
    JOIN TravelMethods t ON c.method_of_travel_id = t.method_of_travel_id
    JOIN PhoneNumbers pn ON c.phone_number_id = pn.phone_number_id
    WHERE t.method_of_travel = method_of_travel_param AND date_part('year', AGE(c.birth_date)) > age_param;
$$ LANGUAGE SQL;

SELECT * FROM getCouriers('Пеший', 20);
```

Рисунок 25 – Третья функция и ее вывод.

Работа с транзакциями:

Первая транзакция:

```sql
SELECT * FROM Points;

BEGIN;
    UPDATE Points SET effective_storage_area = effective_storage_area * 2 WHERE address = 'Московская улица, 1';
    UPDATE Points SET effective_storage_area = effective_storage_area + 20 WHERE address = 'Ленинградский проспект, 2';
COMMIT;

SELECT * FROM Points;
```

Рисунок 26 – код транзакции.

```
 point_id |          address          | is_oversized_cargo_service | phone_number_id | effective_storage_area
----------+---------------------------+----------------------------+-----------------+-----------------------
        1 | Московская улица, 1       | f                          |               1 |                 400.5
        2 | Ленинградский проспект, 2 | t                          |               2 |                 150.2
        3 | Сибирский тракт, 3        | f                          |               3 |                 800.7
        4 | Красная площадь, 4        | t                          |               4 |                 200.1
        5 | Невский проспект, 5       | t                          |               5 |                1200.8
        6 | Уральская улица, 6        | t                          |               6 |                 180.3
        7 | Волгоградский проспект, 7 | f                          |               7 |                  90.6
        8 | Казанская улица, 8        | t                          |               8 |                1600.9
        9 | Омская улица, 9           | t                          |               9 |                1100.4
(9 rows)
```

Рисунок 27 – данные до транзакции.

```
BEGIN
UPDATE 1
UPDATE 1
COMMIT
 point_id |          address          | is_oversized_cargo_service | phone_number_id | effective_storage_area
----------+---------------------------+----------------------------+-----------------+-----------------------
        3 | Сибирский тракт, 3        | f                          |               3 |                 800.7
        4 | Красная площадь, 4        | t                          |               4 |                 200.1
        5 | Невский проспект, 5       | t                          |               5 |                1200.8
        6 | Уральская улица, 6        | t                          |               6 |                 180.3
        7 | Волгоградский проспект, 7 | f                          |               7 |                  90.6
        8 | Казанская улица, 8        | t                          |               8 |                1600.9
        9 | Омская улица, 9           | t                          |               9 |                1100.4
        1 | Московская улица, 1       | f                          |               1 |                 801.0
        2 | Ленинградский проспект, 2 | t                          |               2 |                 170.2
(9 rows)
```

Рисунок 28 – данные после транзакции.

По скриншотам видим, что транзакция завершилась успешно и все ее изменения сохраняются в базе данных. Это означает, что все операции, выполненные в рамках транзакции, становятся постоянными и видимыми для других пользователей или процессов. Для этого использовалась команда COMMIT.

Вторая транзакция:

```
SELECT * FROM Points;

BEGIN;
    UPDATE Points SET effective_storage_area = effective_storage_area / 2 WHERE address = 'Московская улица, 1';
    UPDATE Points SET effective_storage_area = effective_storage_area - 20 WHERE address = 'Ленинградский проспект, 2';
    SELECT * FROM Points;
ROLLBACK;

SELECT * FROM Points;
```

Рисунок 29 – код транзакции.

```
 point_id |          address          | is_oversized_cargo_service | phone_number_id | effective_storage_area
----------+---------------------------+----------------------------+-----------------+------------------------
        3 | Сибирский тракт, 3        | f                          |               3 |                  800.7
        4 | Красная площадь, 4        | t                          |               4 |                  200.1
        5 | Невский проспект, 5       | t                          |               5 |                 1200.8
        6 | Уральская улица, 6        | t                          |               6 |                  180.3
        7 | Волгоградский проспект, 7 | f                          |               7 |                   90.6
        8 | Казанская улица, 8        | t                          |               8 |                 1600.9
        9 | Омская улица, 9           | t                          |               9 |                 1100.4
        1 | Московская улица, 1       | f                          |               1 |                  801.0
        2 | Ленинградский проспект, 2 | t                          |               2 |                  170.2
(9 rows)
```

Рисунок 30 – данные до транзакции.

```
BEGIN
UPDATE 1
UPDATE 1
 point_id |          address          | is_oversized_cargo_service | phone_number_id | effective_storage_area
----------+---------------------------+----------------------------+-----------------+------------------------
        3 | Сибирский тракт, 3        | f                          |               3 |                  800.7
        4 | Красная площадь, 4        | t                          |               4 |                  200.1
        5 | Невский проспект, 5       | t                          |               5 |                 1200.8
        6 | Уральская улица, 6        | t                          |               6 |                  180.3
        7 | Волгоградский проспект, 7 | f                          |               7 |                   90.6
        8 | Казанская улица, 8        | t                          |               8 |                 1600.9
        9 | Омская улица, 9           | t                          |               9 |                 1100.4
        1 | Московская улица, 1       | f                          |               1 |                  400.5
        2 | Ленинградский проспект, 2 | t                          |               2 |                  150.2
(9 rows)
```

Рисунок 31 – данные во время транзакции.

```
ROLLBACK
 point_id |          address          | is_oversized_cargo_service | phone_number_id | effective_storage_area
----------+---------------------------+----------------------------+-----------------+------------------------
        3 | Сибирский тракт, 3        | f                          |               3 |                  800.7
        4 | Красная площадь, 4        | t                          |               4 |                  200.1
        5 | Невский проспект, 5       | t                          |               5 |                 1200.8
        6 | Уральская улица, 6        | t                          |               6 |                  180.3
        7 | Волгоградский проспект, 7 | f                          |               7 |                   90.6
        8 | Казанская улица, 8        | t                          |               8 |                 1600.9
        9 | Омская улица, 9           | t                          |               9 |                 1100.4
        1 | Московская улица, 1       | f                          |               1 |                  801.0
        2 | Ленинградский проспект, 2 | t                          |               2 |                  170.2
(9 rows)
```

Рисунок 32 – данные после транзакции.

По скриншотам видим, что транзакция откатывается, это означает, что все изменения, сделанные в рамках этой транзакции, отменяются и база данных возвращается к состоянию, которое было до начала транзакции. Для этого используется команды ROLLBACK.

Третья транзакция:

```
SELECT * FROM Points;

BEGIN;
    UPDATE Points SET effective_storage_area = effective_storage_area / 2 WHERE address = 'Московская улица, 1';
    SAVEPOINT save;

    DELETE FROM Points WHERE address = 'Красная площадь, 4';
    SELECT * FROM Points;

    ROLLBACK TO save;
    UPDATE Points SET effective_storage_area = effective_storage_area - 20 WHERE address = 'Ленинградский проспект, 2';
COMMIT;

SELECT * FROM Points;
```

Рисунок 33 – код транзакции.

point_id	address	is_oversized_cargo_service	phone_number_id	effective_storage_area
3	Сибирский тракт, 3	f	3	800.7
4	Красная площадь, 4	t	4	200.1
5	Невский проспект, 5	t	5	1200.8
6	Уральская улица, 6	t	6	180.3
7	Волгоградский проспект, 7	f	7	90.6
8	Казанская улица, 8	t	8	1600.9
9	Омская улица, 9	t	9	1100.4
1	Московская улица, 1	f	1	801.0
2	Ленинградский проспект, 2	t	2	170.2

(9 rows)

Рисунок 32 – данные до транзакции.

```
BEGIN
UPDATE 1
SAVEPOINT
DELETE 1
```

point_id	address	is_oversized_cargo_service	phone_number_id	effective_storage_area
3	Сибирский тракт, 3	f	3	800.7
5	Невский проспект, 5	t	5	1200.8
6	Уральская улица, 6	t	6	180.3
7	Волгоградский проспект, 7	f	7	90.6
8	Казанская улица, 8	t	8	1600.9
9	Омская улица, 9	t	9	1100.4
2	Ленинградский проспект, 2	t	2	170.2
1	Московская улица, 1	f	1	400.5

(8 rows)

Рисунок 33 – данные во время транзакции.

```
ROLLBACK
UPDATE 1
COMMIT
```

point_id	address	is_oversized_cargo_service	phone_number_id	effective_storage_area
3	Сибирский тракт, 3	f	3	800.7
4	Красная площадь, 4	t	4	200.1
5	Невский проспект, 5	t	5	1200.8
6	Уральская улица, 6	t	6	180.3
7	Волгоградский проспект, 7	f	7	90.6
8	Казанская улица, 8	t	8	1600.9
9	Омская улица, 9	t	9	1100.4
1	Московская улица, 1	f	1	400.5
2	Ленинградский проспект, 2	t	2	150.2

(9 rows)

Рисунок 34 – данные после транзакции.

По скриншотам видим, что транзакция выполнилась частично. Точка сохранения позволила сохранить состояние базы данных на определенном этапе транзакции. Это полезно, когда вы хотите фиксировать только часть изменений в транзакции. Для этого использовались команды SAVEPOINT и ROLLBACK TO.

Вывод:

В рамках данного расчетно-графического задания мы получили практические навыки использования транзакций в СУБД PostgreSQL. Мы создали три транзакции, включающие операции по изменению и удалению данных в БД. Первая транзакция была успешно фиксирована, что означает, что все изменения внутри транзакции были сохранены в базе данных. Вторая транзакция была откатана, что означает, что все изменения, сделанные внутри транзакции, были отменены и база данных вернулась к состоянию до начала транзакции. Третья транзакция содержала точку сохранения, что позволило фиксировать только часть изменений внутри транзакции.

ПРИЛОЖЕНИЕ

Листинг Pleshkov.sql:

```
CREATE TABLE TravelMethods (
  method_of_travel_id SERIAL PRIMARY KEY,
  method_of_travel VARCHAR(30) NOT NULL
);

CREATE TABLE Positions (
  position_id SERIAL PRIMARY KEY,
  position VARCHAR(60) NOT NULL
);

CREATE TABLE PhoneNumbers (
  phone_number_id SERIAL PRIMARY KEY,
  phone_number VARCHAR(11) NOT NULL
);

CREATE TABLE Couriers (
  courier_id SERIAL PRIMARY KEY,
  full_name VARCHAR(60) NOT NULL,
  birth_date DATE NOT NULL,
  method_of_travel_id INT NOT NULL,
  phone_number_id INT NOT NULL,
  FOREIGN KEY (method_of_travel_id) REFERENCES TravelMethods (method_of_travel_id),
  FOREIGN KEY (phone_number_id) REFERENCES PhoneNumbers (phone_number_id)
);

CREATE TABLE Employees (
  employee_id SERIAL PRIMARY KEY,
  full_name VARCHAR(60) NOT NULL,
  location VARCHAR(80) NOT NULL,
  position_id INT NOT NULL,
  FOREIGN KEY (position_id) REFERENCES Positions (position_id)
);

CREATE TABLE Points (
  point_id SERIAL PRIMARY KEY,
  address VARCHAR(255) NOT NULL,
  is_oversized_cargo_service BOOLEAN NOT NULL,
  phone_number_id INT NOT NULL,
  effective_storage_area DECIMAL(8,1) NOT NULL,
  FOREIGN KEY (phone_number_id) REFERENCES PhoneNumbers (phone_number_id)
);

CREATE TABLE Orders (
  order_id SERIAL PRIMARY KEY,
  order_date DATE NOT NULL,
  distance_to_addressee DECIMAL(8,1) NOT NULL,
  employee_id INT NOT NULL,
  point_id INT NOT NULL,
  FOREIGN KEY (employee_id) REFERENCES Employees (employee_id),
  FOREIGN KEY (point_id) REFERENCES Points (point_id)
);


INSERT INTO TravelMethods (method_of_travel)
VALUES
  ('Автомобиль'),
  ('Самокат'),
  ('Пеший');

INSERT INTO Positions (position)
VALUES
  ('Менеджер'),
  ('Продавец'),
  ('Администратор');

INSERT INTO PhoneNumbers (phone_number)
```

```
VALUES
  ('12345678901'),
  ('23456789012'),
  ('34567890123'),
  ('45678901234'),
  ('56789012345'),
  ('67890123456'),
  ('78901234567'),
  ('89012345678'),
  ('90123456789'),
  ('01234567890');

INSERT INTO Couriers (full_name, birth_date, method_of_travel_id, phone_number_id)
VALUES
  ('Иван Иванов', '1990-01-01', 1, 1),
  ('Алексей Петров', '1992-05-15', 2, 2),
  ('Елена Смирнова', '1985-11-30', 1, 3),
  ('Мария Ковалева', '1992-07-20', 3, 4),
  ('Андрей Васильев', '1988-03-10', 2, 5),
  ('Ольга Николаева', '1991-09-25', 1, 6),
  ('Дмитрий Соколов', '1987-12-05', 3, 7),
  ('Сергей Морозов', '1994-04-18', 2, 8),
  ('Наталья Волкова', '1989-08-12', 1, 9),
  ('Павел Лебедев', '1995-02-28', 3, 10);

INSERT INTO Employees (full_name, location, position_id)
VALUES
  ('Анна Сидорова', 'Москва', 1),
  ('Александр Иванов', 'Санкт-Петербург', 2),
  ('Екатерина Козлова', 'Екатеринбург', 3),
  ('Игорь Павлов', 'Новосибирск', 1),
  ('Марина Смирнова', 'Казань', 2),
  ('Артем Петров', 'Владивосток', 3),
  ('Ольга Морозова', 'Ростов-на-Дону', 1),
  ('Денис Васильев', 'Самара', 2),
  ('Николай Ковалев', 'Омск', 3),
  ('Татьяна Николаева', 'Москва', 1);

INSERT INTO Points (address, is_oversized_cargo_service, phone_number_id,
effective_storage_area)
VALUES
  ('Московская улица, 1', false, 1, 400.5),
  ('Ленинградский проспект, 2', true, 2, 150.2),
  ('Сибирский тракт, 3', false, 3, 800.7),
  ('Красная площадь, 4', true, 4, 200.1),
  ('Невский проспект, 5', true, 5, 1200.8),
  ('Уральская улица, 6', true, 6, 180.3),
  ('Волгоградский проспект, 7', false, 7, 90.6),
  ('Казанская улица, 8', true, 8, 1600.9),
  ('Омская улица, 9', true, 9, 1100.4),
  ('Тихоокеанская улица, 10', true, 10, 140.0);

INSERT INTO Orders (order_date, distance_to_addressee, employee_id, point_id)
VALUES
  ('2023-01-01', 10.5, 1, 1),
  ('2023-02-15', 5.2, 1, 2),
  ('2023-03-30', 8.7, 2, 3),
  ('2023-04-10', 12.3, 1, 3),
  ('2023-05-20', 6.8, 5, 5),
  ('2023-06-05', 9.1, 3, 3),
  ('2023-07-18', 7.4, 4, 1),
  ('2023-08-25', 11.2, 2, 1),
  ('2023-09-10', 4.9, 3, 2),
  ('2023-10-28', 14.6, 5, 3);

-- Задание 1 добавляем столбец полный ли рабочий день у курьера
ALTER TABLE Couriers
ADD COLUMN is_full_time_working BOOLEAN;

-- Задание 2 меняем должность работников
UPDATE Employees
```

```
SET location = 'Казань'
WHERE location = 'Ленск';

-- Задание 3 удаляем столбец
ALTER TABLE Couriers
DROP COLUMN is_full_time_working;

-- Задание 4 удаляем пункт по адресу
DELETE FROM Points
WHERE address = 'Тихоокеанская улица, 10';

--Ограничения

--Ограничение на дату рождения
ALTER TABLE Couriers
ADD CONSTRAINT check_birth_date CHECK (birth_date >= '1900-01-01' AND birth_date <=
CURRENT_DATE);

--Ограничение на дату заказа
ALTER TABLE Orders
ADD CONSTRAINT order_date CHECK (order_date <= CURRENT_DATE);

--Ограничение на расстояние до адресата
ALTER TABLE Orders
ADD CONSTRAINT distance_to_addressee CHECK (distance_to_addressee > 0);

--Ограничение на эффективную площадь хранения
ALTER TABLE Points
ADD CONSTRAINT effective_storage_area CHECK (effective_storage_area > 0);


--Составная таблица
CREATE TABLE Orders_by_points_and_employees(order_id, employee_full_name,
point_address) AS
    SELECT o.order_id, e.full_name, p.address FROM Orders o
    JOIN Employees e ON o.employee_id = e.employee_id
    JOIN Points p ON o.point_id = p.point_id;

--Ограничения на составную таблицу
ALTER TABLE Orders_by_points_and_employees ADD PRIMARY KEY (order_id);
ALTER TABLE Orders_by_points_and_employees ADD FOREIGN KEY (order_id) references
Orders on delete cascade;



--Функция для вывода пунктов по параметрам: is_oversized_cargo_service и
effective_storage_area
CREATE OR REPLACE FUNCTION getPoints(is_oversized BOOLEAN, storage_area INTEGER)
RETURNS TABLE (point_id INT, address VARCHAR, is_oversized_cargo_service BOOLEAN,
phone_number VARCHAR, effective_storage_area DECIMAL(8,1))
AS $$
    SELECT p.point_id, p.address, p.is_oversized_cargo_service, pn.phone_number,
p.effective_storage_area
    FROM Points p
    JOIN PhoneNumbers pn ON p.phone_number_id = pn.phone_number_id
    WHERE p.is_oversized_cargo_service = is_oversized AND p.effective_storage_area >
storage_area;
$$ LANGUAGE SQL;

SELECT * FROM getPoints(false, 400);



--Функция для вывода всех работников по параметрам: location и positionn
CREATE OR REPLACE FUNCTION getEmployees(loc VARCHAR, pos VARCHAR)
RETURNS TABLE (employee_id INT, full_name VARCHAR, location VARCHAR, positionn
VARCHAR)
AS $$
    SELECT e.employee_id, e.full_name, e.location, p.position
    FROM Employees e
    JOIN Positions p ON e.position_id = p.position_id
```

```
    WHERE e.location = loc AND p.position = pos;
$$ LANGUAGE SQL;

SELECT * FROM getEmployees('Москва', 'Менеджер');



--Функция для вывода всех курьеров по параметрам: method_of_travel и age
CREATE OR REPLACE FUNCTION getCouriers(method_of_travel_param VARCHAR, age_param INT)
RETURNS TABLE (courier_id INT, full_name VARCHAR, birth_date DATE, age INT,
method_of_travel VARCHAR, phone_number VARCHAR)
AS $$
    SELECT c.courier_id, c.full_name, c.birth_date, date_part('year',
AGE(c.birth_date)) AS age, t.method_of_travel, pn.phone_number
    FROM Couriers c
    JOIN TravelMethods t ON c.method_of_travel_id = t.method_of_travel_id
    JOIN PhoneNumbers pn ON c.phone_number_id = pn.phone_number_id
    WHERE t.method_of_travel = method_of_travel_param AND date_part('year',
AGE(c.birth_date)) > age_param;
$$ LANGUAGE SQL;

SELECT * FROM getCouriers('Пеший', 20);

--Транзакции

SELECT * FROM Points;

BEGIN;
    UPDATE Points SET effective_storage_area = effective_storage_area * 2 WHERE
address = 'Московская улица, 1';
    UPDATE Points SET effective_storage_area = effective_storage_area + 20 WHERE
address = 'Ленинградский проспект, 2';
COMMIT;

SELECT * FROM Points;

BEGIN;
    UPDATE Points SET effective_storage_area = effective_storage_area / 2 WHERE
address = 'Московская улица, 1';
    UPDATE Points SET effective_storage_area = effective_storage_area - 20 WHERE
address = 'Ленинградский проспект, 2';
    SELECT * FROM Points;
ROLLBACK;

SELECT * FROM Points;

BEGIN;
    UPDATE Points SET effective_storage_area = effective_storage_area / 2 WHERE
address = 'Московская улица, 1';
    SAVEPOINT save;

    DELETE FROM Points WHERE address = 'Красная площадь, 4';
    SELECT * FROM Points;

    ROLLBACK TO save;
    UPDATE Points SET effective_storage_area = effective_storage_area - 20 WHERE
address = 'Ленинградский проспект, 2';
COMMIT;

SELECT * FROM Points;
```