МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

“Санкт-Петербургский национальный исследовательский университет информационных технологий механики и оптики”

Мегафакультет: трансляционных информационных технологий

Факультет: информационных технологий и программирования

Лабораторная работа №5

По дисциплине: “Проектирование баз данных”

Тема: “Партицирование в PostgreSQL”

Выполнила студент группы №M3216:

Шевцов Роман Сергеевич

САНКТ-ПЕРЕРБУРГ

2025

Задача:

Сделать партицирование одной таблицы через наследование.

Требования:

1. Таблица должна быть разбита на не менее, чем 3 партиции.
2. В каждой партиции должно быть не менее 5 записей.

Порядок выполнения работы:

1. Выберите таблицу, которую можно разбить на партиции. Определите условия разбиения таблицы на партиции.
2. Создайте таблицы-партиции.
3. Создайте функцию, обеспечивающую партицирование.
4. Подключите функцию к мастер-таблице.
5. Перенесите данные из мастер-таблицы в партиции.
6. Очистите мастер-таблицу и добавьте в нее новые данные.

На защите лабораторной необходимо будет продемонстрировать мастер-таблицу, таблицы-партиции, функцию, обеспечивающую партицирование, результаты выполнения запросов для добавления новых записей и чтения таблиц.

Решение:

Создаем новую мастер-таблицу и вставляем в нее значения:

Query | Query History

```sql
CREATE TABLE device (
    deviceid SERIAL PRIMARY KEY,
    userid INTEGER,
    devicemodel TEXT,
    purchasedate DATE,
    price NUMERIC
);

INSERT INTO device (userid, devicemodel, purchasedate, price)
VALUES
  (101, 'Xiaomi 12T', '2023-03-10', 649.00),
  (102, 'Realme GT', '2023-09-01', 300.00),
  (103, 'Nokia 3310', '2023-05-01', 50.00),
  (104, 'Samsung A14', '2023-04-01', 250.00),
  (105, 'Motorola G7', '2023-07-01', 200.00),
  (106, 'Google Pixel 7', '2023-01-25', 799.99),
  (107, 'OnePlus 11', '2023-04-05', 699.99),
  (108, 'Sony Xperia', '2023-07-01', 850.00),
  (109, 'Huawei P50', '2023-08-01', 950.00),
  (110, 'iPhone 14 Pro', '2023-02-15', 1399.99);

SELECT * FROM device;
```

Проверяем вновь созданную таблицу:

Data Output | Messages | Notifications

	deviceid [PK] integer	userid integer	devicemodel text	purchasedate date	price numeric
1	1	101	Xiaomi 12T	2023-03-10	649.00
2	2	102	Realme GT	2023-09-01	300.00
3	3	103	Nokia 3310	2023-05-01	50.00
4	4	104	Samsung A14	2023-04-01	250.00
5	5	105	Motorola G7	2023-07-01	200.00
6	6	106	Google Pixel 7	2023-01-25	799.99
7	7	107	OnePlus 11	2023-04-05	699.99
8	8	108	Sony Xperia	2023-07-01	850.00
9	9	109	Huawei P50	2023-08-01	950.00
10	10	110	iPhone 14 Pro	2023-02-15	1399.99

Создаем партиции по диапазону цен:

Query | Query History

```
1  CREATE TABLE device_low_price (
2      CHECK (price < 700)
3  ) INHERITS (device);
4
5  CREATE TABLE device_medium_price (
6      CHECK (price >= 700 AND price < 1000)
7  ) INHERITS (device);
8
9  CREATE TABLE device_high_price (
10     CHECK (price >= 1000)
11 ) INHERITS (device);
```

Data Output | Messages | Notifications

```
CREATE TABLE

Query returned successfully in 149 msec.
```

Создаем функцию для автоматического распределения данных:

Query | Query History

```
CREATE OR REPLACE FUNCTION device_insert_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.price < 700) THEN
        INSERT INTO device_low_price VALUES (NEW.*);
    ELSIF (NEW.price >= 700 AND NEW.price < 1000) THEN
        INSERT INTO device_medium_price VALUES (NEW.*);
    ELSIF (NEW.price >= 1000) THEN
        INSERT INTO device_high_price VALUES (NEW.*);
    ELSE
        RAISE EXCEPTION 'Invalid price value: %', NEW.price;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Data Output | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 68 msec.
```

Создаем и подключаем триггер к мастер-таблице:

Query | Query History

```
CREATE TRIGGER device_insert_trigger
BEFORE INSERT ON device
FOR EACH ROW EXECUTE FUNCTION device_insert_trigger();
```

Data Output | Messages | Notifications

```
CREATE TRIGGER

Query returned successfully in 59 msec.
```

Переносим данные из мастер-таблицы в партиции и очищаем мастер-таблицу:

Query | Query History

```
INSERT INTO device_low_price
SELECT * FROM device WHERE price < 700;

INSERT INTO device_medium_price
SELECT * FROM device WHERE price >= 700 AND price < 1000;

INSERT INTO device_high_price
SELECT * FROM device WHERE price >= 1000;

TRUNCATE ONLY device;
```

Data Output | Messages | Notifications

```
TRUNCATE TABLE
```

Проверим партицию device_low_price:

Query | Query History

```
1  SELECT * FROM device_low_price
```

Data Output | Messages | Notifications

	deviceid integer	userid integer	devicemodel text	purchasedate date	price numeric
1	1	101	Xiaomi 12T	2023-03-10	649.00
2	2	102	Realme GT	2023-09-01	300.00
3	3	103	Nokia 3310	2023-05-01	50.00
4	4	104	Samsung A14	2023-04-01	250.00
5	5	105	Motorola G7	2023-07-01	200.00
6	7	107	OnePlus 11	2023-04-05	699.99

Вставим новые данные в мастер-таблицу и проверим что они автоматически добавляются в партиции:

Query | Query History

```
1  INSERT INTO device (userid, devicemodel, purchasedate, price) VALUES
2    (201, 'iPhone 15 Pro', '2023-09-15', 1299.00),
3    (202, 'Samsung Galaxy S23', '2023-02-01', 899.99),
4    (203, 'Xiaomi Redmi Note 12', '2023-05-20', 299.00),
5    (204, 'Google Pixel 8 Pro', '2023-10-05', 1099.00),
6    (205, 'OnePlus 11', '2023-01-15', 749.99);
7
8  SELECT 'device_low_price' AS partition, COUNT(*) FROM device_low_price
9  UNION ALL
10 SELECT 'device_medium_price' AS partition, COUNT(*) FROM device_medium_price
11 UNION ALL
12 SELECT 'device_high_price' AS partition, COUNT(*) FROM device_high_price;
```

Data Output | Messages | Notifications

	partition text	count bigint
1	device_low_price	7
2	device_medium_price	5
3	device_high_price	3

Вновь проверим партицию device_low_price и остальные, убедимся что данные обновляются:

Query | Query History

```
1  SELECT * FROM device_low_price
```

Data Output | Messages | Notifications

	deviceid integer	userid integer	devicemodel text	purchasedate date	price numeric
1	1	101	Xiaomi 12T	2023-03-10	649.00
2	2	102	Realme GT	2023-09-01	300.00
3	3	103	Nokia 3310	2023-05-01	50.00
4	4	104	Samsung A14	2023-04-01	250.00
5	5	105	Motorola G7	2023-07-01	200.00
6	7	107	OnePlus 11	2023-04-05	699.99
7	13	203	Xiaomi Redmi Note 12	2023-05-20	299.00

Query | Query History

```
1  SELECT * FROM device_high_price
```

Data Output | Messages | Notifications

	deviceid integer	userid integer	devicemodel text	purchasedate date	price numeric
1	10	110	iPhone 14 Pro	2023-02-15	1399.99
2	11	201	iPhone 15 Pro	2023-09-15	1299.00
3	14	204	Google Pixel 8 Pro	2023-10-05	1099.00

Query | Query History

```
1  SELECT * FROM device_medium_price
```

Data Output | Messages | Notifications

	deviceid integer	userid integer	devicemodel text	purchasedate date	price numeric
1	6	106	Google Pixel 7	2023-01-25	799.99
2	8	108	Sony Xperia	2023-07-01	850.00
3	9	109	Huawei P50	2023-08-01	950.00
4	12	202	Samsung Galaxy S23	2023-02-01	899.99
5	15	205	OnePlus 11	2023-01-15	749.99

Поскольку в условии лабораторной просят не менее 5 записей в каждой партиции, добавим пару записей в device_high_price:

Query Query History

```
1 INSERT INTO device (userid, devicemodel, purchasedate, price) VALUES
2   (206, 'Samsung Galaxy Z Fold5', '2023-08-10', 1799.00),
3   (207, 'iPhone 15 Pro Max', '2023-09-22', 1599.00),
4   (208, 'Google Pixel Fold', '2023-06-27', 1799.00),
5   (209, 'Huawei Mate X3', '2023-04-15', 1999.00);
```

Data Output Messages Notifications

```
INSERT 0 0

Query returned successfully in 174 msec.
```

Проверим количество записей в каждой партиции:

Query Query History

```
1 SELECT 'low_price' AS partition, COUNT(*) FROM device_low_price
2 UNION ALL
3 SELECT 'medium_price', COUNT(*) FROM device_medium_price
4 UNION ALL
5 SELECT 'high_price', COUNT(*) FROM device_high_price;
```

Data Output Messages Notifications

SQL

	partition text	count bigint
1	low_price	7
2	medium_price	5
3	high_price	7