**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

**Проектирование баз данных**

**Лабораторная работа №5.**

Партицирование в PostgresSQL

Вариант №6 (Библиотека)

Выполнил студент группы №М34061

**Величко Максим**

Преподаватель:

**Боркунов Владимир Юрьевич**

САНКТ-ПЕТЕРБУРГ

2024

**Задача:**

Создание партиционированной таблицы с использованием наследования в PostgreSQL. Таблица должна быть разбита на не менее чем три партиции, каждая из которых должна содержать как минимум 5 записей.

Моя бд выглядит вот так:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

**1. Выбор таблицы для разбиения**

Для партиционирования была выбрана таблица transactions, которая хранит информацию о транзакциях. В таблице содержатся данные, такие как:

* id — уникальный идентификатор транзакции,
* transaction\_kind — тип транзакции (например, покупка, возврат),
* date — дата транзакции,
* cost — стоимость транзакции,
* user\_id, book\_id, subscription\_id — идентификаторы пользователя, книги и подписки.

Выбор этой таблицы был обусловлен тем, что транзакции часто имеют дату и могут быть эффективно разделены по временным диапазонам (например, по годам).

**2. Условия разбиения таблицы на партиции**

Таблица transactions\_master была разбита на три партиции с использованием **диапазона дат**:

* Партиция для 2022 года (transactions\_2022),
* Партиция для 2023 года (transactions\_2023),
* Партиция для 2024 года (transactions\_2024).

Каждая партиция будет хранить данные только за один календарный год, что позволяет эффективно управлять данными и улучшать производительность запросов, особенно при фильтрации по дате.

**3. Создание таблицы-партиции**

Вначале была создана основная таблица transactions\_master, которая является мастер-таблицей для хранения всех транзакций. Эта таблица использует механизм **партиционирования по диапазону дат**:

CREATE TABLE transactions\_master (

id uuid NOT NULL,

transaction\_kind INT NOT NULL CHECK (transaction\_kind IN (1, 2, 3)),

date TIMESTAMP NOT NULL,

cost DECIMAL(10, 2) DEFAULT 0 CHECK (cost >= 0),

user\_id uuid NOT NULL,

book\_id uuid,

subscription\_id uuid

) PARTITION BY RANGE (date);

Затем были созданы три партиции для каждого года:

CREATE TABLE transactions\_2022 PARTITION OF transactions\_master

FOR VALUES FROM ('2022-01-01') TO ('2023-01-01');

CREATE TABLE transactions\_2023 PARTITION OF transactions\_master

FOR VALUES FROM ('2023-01-01') TO ('2024-01-01');

CREATE TABLE transactions\_2024 PARTITION OF transactions\_master

FOR VALUES FROM ('2024-01-01') TO ('2025-01-01');

**4. Создание функции для партиционирования**

Для автоматического распределения данных по партициям была создана функция insert\_into\_transactions\_master(), которая проверяет дату каждой вставляемой записи и направляет ее в соответствующую партицию. Функция работает следующим образом:

Если дата транзакции находится в пределах 2022 года, данные вставляются в партицию transactions\_2022,

Если дата транзакции находится в пределах 2023 года, данные вставляются в партицию transactions\_2023,

Если дата транзакции находится в пределах 2024 года, данные вставляются в партицию transactions\_2024.

CREATE OR REPLACE FUNCTION insert\_into\_transactions\_master()

RETURNS TRIGGER AS $$

BEGIN

IF (NEW.date >= '2022-01-01' AND NEW.date < '2023-01-01') THEN

INSERT INTO transactions\_2022 (id, transaction\_kind, date, cost, user\_id, book\_id, subscription\_id)

VALUES (NEW.id, NEW.transaction\_kind, NEW.date, NEW.cost, NEW.user\_id, NEW.book\_id, NEW.subscription\_id);

ELSIF (NEW.date >= '2023-01-01' AND NEW.date < '2024-01-01') THEN

INSERT INTO transactions\_2023 (id, transaction\_kind, date, cost, user\_id, book\_id, subscription\_id)

VALUES (NEW.id, NEW.transaction\_kind, NEW.date, NEW.cost, NEW.user\_id, NEW.book\_id, NEW.subscription\_id);

ELSIF (NEW.date >= '2024-01-01' AND NEW.date < '2025-01-01') THEN

INSERT INTO transactions\_2024 (id, transaction\_kind, date, cost, user\_id, book\_id, subscription\_id)

VALUES (NEW.id, NEW.transaction\_kind, NEW.date, NEW.cost, NEW.user\_id, NEW.book\_id, NEW.subscription\_id);

ELSE

RAISE EXCEPTION 'Date out of range: %, %', NEW.id, NEW.date;

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

**5.** **Создание триггера для автоматического распределения данных**

Для того чтобы автоматизировать процесс вставки данных в соответствующие партиции, был создан триггер transactions\_insert\_trigger, который вызывает функцию insert\_into\_transactions\_master() перед каждой вставкой записи в мастер-таблицу:

CREATE TRIGGER transactions\_insert\_trigger BEFORE INSERT ON transactions\_master FOR EACH ROW EXECUTE FUNCTION insert\_into\_transactions\_master();

**6. Перенос данных из исходной таблицы**

Затем были перенесены данные из исходной таблицы transactions в новую мастер-таблицу transactions\_master. Для этого был выполнен запрос INSERT INTO:

INSERT INTO transactions\_master (id, transaction\_kind, date, cost, user\_id, book\_id, subscription\_id)

SELECT id, transaction\_kind, date, cost, user\_id, book\_id, subscription\_id

FROM transactions;

После этого данные из исходной таблицы были удалены с помощью запроса DELETE:

DELETE FROM transactions;

**7. Добавление уникальных ограничений для партиций**

Каждая партиция была дополнена уникальными ограничениями, чтобы гарантировать уникальность записей по комбинации полей id и date. Это помогает предотвратить дублирование данных внутри каждой партиции:

ALTER TABLE transactions\_2022 ADD CONSTRAINT transactions\_2022\_pkey PRIMARY KEY (id, date);

ALTER TABLE transactions\_2023 ADD CONSTRAINT transactions\_2023\_pkey PRIMARY KEY (id, date);

ALTER TABLE transactions\_2024 ADD CONSTRAINT transactions\_2024\_pkey PRIMARY KEY (id, date);

**8. Отключение и включение триггера**

Для отладки триггер был временно отключен:

ALTER TABLE transactions\_master DISABLE TRIGGER transactions\_insert\_trigger;

Когда отладка была завершена, триггер можно было включить снова:

ALTER TABLE transactions\_master ENABLE TRIGGER transactions\_insert\_trigger;

**Результаты выполнения запросов:**

**Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание**