Секционирование таблиц для больших объёмов данных

PostgreSQL для администраторов баз данных и разработчиков

Меня хорошо видно & слышно?

Защита проекта
Тема: Секционирование таблиц для больших объёмов данных

Вадим Ковалев

Разработчик back-end в компании
Цифровые Технологии и Платформы

План защиты

- Цель и задачи проекта
- Какие технологии использовались
- Что получилось
- Выводы
- Вопросы и рекомендации

Цель и задачи проекта

Цель проекта: из большой таблицы-монолита создать секционированную таблицу, сравнить быстродействие запросов

1. Проанализировать таблицу монолит на предмет оптимального типа секционирования (по диапазону, по списку или по хешу)
2. Создать секционированную таблицу с выбранным типом секционирования.
3. Перенести существующие данные из исходной таблицы в секционированную структуру.
4. Проверить правильность распределения данных по секциям.
5. Проверить влияние секционирования на производительность запросов.

Какие технологии использовались

1. PostgreSQL 18.0 (Debian 18.0-1.pgdg13+3) on x86_64-pc-linux-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit
2. Docker desktop 4.57.0 Engine 29.1.3
3. Windows 11 WSL 2
4. Для доступа к данным MS PostgreSQL for Visual Studio Code 1.14.0
5. Использовалось декларативное секционирование, доступное непосредственно в PostgreSQl с 10 версии
6. Для тестов использовалась демобаза bookings с ресурса https://postgrespro.ru/education/demodb

Примечание. К презентации прилагается файл всех скриптов, выполненных в процессе работы над проектом - otus-pg-sections-project.md

Что получилось

Работал с таблицей bookings на примерно пять миллионов строк(4905238):

```
 CREATE TABLE IF NOT EXISTS bookings.bookings
(
    book_ref character(6) COLLATE pg_catalog."default" NOT NULL,
    book_date timestamp with time zone NOT NULL,
    total_amount numeric(10,2) NOT NULL,
    CONSTRAINT bookings_pkey PRIMARY KEY (book_ref)
);
```

Поле book_ref уникально, поэтому секцинирование по списку значений отпало.
Теоретически можно было сделать по этому полю секционирование по хешу для расномерного распределния данных, но исходя из структуры таблицы видно, что скорее всего будут интересовать запросы по датам, поэтому остановился на секционировании по диапазонам дат.

Наиболее очевидный и интуитивно понятный способ разбиения - по диапазонам дат.
Проверил это предположение запросом к таблице-монолиту:

```
 SELECT
    EXTRACT(YEAR FROM book_date)::text || '-' || LPAD(EXTRACT(MONTH FROM
book_date)::text, 2, '0') AS "Section"
  , COUNT() AS "Rows"
  , MIN(book_date) AS "MinBookDate"
  , MAX(book_date) AS "MaxBookDate"
FROM bookings.bookings
GROUP BY EXTRACT(YEAR FROM book_date)::text || '-' || LPAD(EXTRACT(MONTH FROM
book_date)::text, 2, '0')
ORDER BY 1;
```

Section	Rows	MinBookDate	MaxBookDate
2025-09	448064	2025-09-01 00:00:06.265219+00	2025-09-30 23:59:58.026243+00
2025-10	434159	2025-10-01 00:00:08.899725+00	2025-10-31 23:59:53.263233+00
2025-11	410670	2025-11-01 00:00:01.790251+00	2025-11-30 23:59:28.616825+00
2025-12	410796	2025-12-01 00:00:00.372488+00	2025-12-31 23:59:40.191846+00
2026-01	412036	2026-01-01 00:00:01.314353+00	2026-01-31 23:59:54.499781+00
2026-02	370985	2026-02-01 00:00:19.902319+00	2026-02-28 23:59:58.993676+00
2026-03	408667	2026-03-01 00:00:08.455494+00	2026-03-31 23:59:58.526315+00
2026-04	383124	2026-04-01 00:00:12.949237+00	2026-04-30 23:59:31.125808+00
2026-05	399596	2026-05-01 00:00:23.516073+00	2026-05-31 23:59:48.496859+00
2026-06	403237	2026-06-01 00:00:10.490019+00	2026-06-30 23:59:59.951396+00
2026-07	411392	2026-07-01 00:00:06.299438+00	2026-07-31 23:59:54.689055+00
2026-08	412512	2026-08-01 00:00:08.873243+00	2026-08-31 23:59:58.283465+00

Предположение оказалось верным - данные внутри месячных диапазонов распределены равномерно, поэтому решил создавать секции по месячным диапазонам:
Создал схему для секционированной таблицы bookings_copy
Создал в ней секционированную таблицу по диапазону дат бронирования (book_date):

```
CREATE TABLE IF NOT EXISTS bookings_copy.bookings
(
    book_ref character(6) COLLATE pg_catalog."default" NOT NULL,
    book_date timestamp with time zone NOT NULL,
    total_amount numeric(10,2) NOT NULL,
    CONSTRAINT bookings_pkey PRIMARY KEY (book_ref, book_date)
)
PARTITION BY RANGE (book_date);
```

Создал секции из расчета одна секция на месяц:

```
CREATE TABLE bookings_copy.bookings_2025_09 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2025-09-01') TO ('2025-10-01');
CREATE TABLE bookings_copy.bookings_2025_10 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2025-10-01') TO ('2025-11-01');
CREATE TABLE bookings_copy.bookings_2025_11 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2025-11-01') TO ('2025-12-01');
CREATE TABLE bookings_copy.bookings_2025_12 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2025-12-01') TO ('2026-01-01');
CREATE TABLE bookings_copy.bookings_2026_01 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2026-01-01') TO ('2026-02-01');
CREATE TABLE bookings_copy.bookings_2026_02 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2026-02-01') TO ('2026-03-01');
CREATE TABLE bookings_copy.bookings_2026_03 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2026-03-01') TO ('2026-04-01');
CREATE TABLE bookings_copy.bookings_2026_04 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2026-04-01') TO ('2026-05-01');
CREATE TABLE bookings_copy.bookings_2026_05 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2026-05-01') TO ('2026-06-01');
CREATE TABLE bookings_copy.bookings_2026_06 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2026-06-01') TO ('2026-07-01');
CREATE TABLE bookings_copy.bookings_2026_07 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2026-07-01') TO ('2026-08-01');
CREATE TABLE bookings_copy.bookings_2026_08 PARTITION OF bookings_copy.bookings FOR VALUES FROM ('2026-08-01') TO ('2026-09-01');
```

Создал секцию по умолчанию для дат, не попадающих ни в одну секцию:

```
CREATE TABLE bookings_copy.bookings_other PARTITION OF bookings_copy.bookings DEFAULT;
```

Скопировал данные:

```
INSERT INTO bookings_copy.bookings
SELECT * FROM bookings.bookings;
```

Подробные скрипты проверки влияния секционирования на производительность запросов приведены в файле otus-pg-sections-project.md,
здесь в презентации показано сравнение планов наиболее показательного запроса с увеличением производительности на порядок.

Монолит:

```
VACUUM ANALYZE bookings.bookings;
EXPLAIN ANALYZE
SELECT \* FROM bookings.bookings WHERE book\_date BETWEEN '2025-10-01' AND '2025-10-30';

Gather  (cost=1000.00..103637.14 rows=406994 width=21) (actual time=30.944..314.811 rows=407769.00 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  Buffers: shared hit=3 read=31277
  ->  Parallel Seq Scan on bookings  (cost=0.00..61937.74 rows=169581 width=21) (actual time=15.520..277.306 rows=135923.00 loops=3)
        Filter: ((book\_date >= '2025-10-01 00:00:00+00'::timestamp with time zone) AND (book\_date <= '2025-10-30 00:00:00+00'::timestamp with time zone))
        Rows Removed by Filter: 1499156
        Buffers: shared hit=3 read=31277
Planning Time: 0.073 ms
JIT:
  Functions: 6
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 1.026 ms (Deform 0.153 ms), Inlining 0.000 ms, Optimization 3.529 ms, Emission 41.953 ms, Total 46.508 ms
Execution Time: 480.203 ms
```

Секционированная таблица:

```
VACUUM ANALYZE bookings\_copy.bookings;
EXPLAIN ANALYZE
SELECT \* FROM bookings\_copy.bookings WHERE book\_date BETWEEN '2025-10-01' AND '2025-10-30';
```

```
Seq Scan on bookings\_2025\_10 bookings  (cost=0.00..9278.38 rows=407842 width=21) (actual time=0.381..30.611 rows=407769.00 loops=1)
  Filter: ((book\_date >= '2025-10-01 00:00:00+00'::timestamp with time zone) AND (book\_date <= '2025-10-30 00:00:00+00'::timestamp with time zone))
  Rows Removed by Filter: 26390
  Buffers: shared read=2766
Planning:
  Buffers: shared hit=174 read=11
Planning Time: 5.812 ms
Execution Time: 42.177 ms
```

Видим, что время извлечения данных уменьшилось на порядок!

Выводы

Секционирование улучшает производительность в случае если запросы работают с данными
умещающимися внутрь одной секции (то есть в фильтре используется поле разбиения),
в противном случае секционирование даже вредно.
Поэтому очень важно продумать стратегию разбиения данных.

Спасибо за внимание!