Опис на партиционирањето во BlinkBuy

Вовед

Партиционирањето е напредна техника во PostgreSQL која овозможува поделба на една голема логичка табела на помали физички делови наречени партиции. Ова е клучно за системи како BlinkBuy кои работат со милиони записи, бидејќи директно ги подобрува перформансите преку механизмот наречен **partition pruning** (прескокнување на нерелевантните податоци).

Во нашиот проект користевме два главни типа на партиционирање:

1. **LIST Partitioning:** Идеално за категории со стабилни вредности (го искористивме за атрибутите на производите).
2. **RANGE Partitioning:** Најдобар избор за временски (audit) податоци кои природно растат со тек на време (го искористивме за историите на цени и статуси).

1. Табела: PRODUCT_ATTRIBUTE_VALUES (LIST Partitioning)

Ова е најголемата табела во BlinkBuy со околу **15 милиони редови**. Бидејќи корисниците постојано филтрираат производи според спецификации (RAM, Процесор, Боја), пребарувањето низ една гигантска табела би го забавило целиот систем.

Примена

Преку **LIST партиционирање по attribute_id**, ги одвоивме најчесто пребаруваните спецификации во сопствени физички табели (pav_ram, pav_processor, pav_color).

- **Резултат:** Кога купувачот бара „16GB RAM“, базата воопшто не ги „гледа“ останатите 13.5 милиони редови за бои или екрани, веднаш пристапува до партицијата за RAM.

```sql
-- 1. create master table
CREATE TABLE PRODUCT_ATTRIBUTE_VALUES_P (
  variant_id   int8 NOT NULL,
  attribute_id int8 NOT NULL,
  attr_value   varchar(255) NOT NULL,
  -- pk contains attribute_id
  PRIMARY KEY (variant_id, attribute_id)
) PARTITION BY LIST (attribute_id);

-- 2. create partitions for main attributes
CREATE TABLE pav_processor     PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (1);
CREATE TABLE pav_ram           PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (2);
CREATE TABLE pav_storage       PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (3);
CREATE TABLE pav_screen_size   PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (4);
CREATE TABLE pav_display_type  PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (5);
CREATE TABLE pav_battery       PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (6);
CREATE TABLE pav_gpu           PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (7);
CREATE TABLE pav_color         PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (8);
CREATE TABLE pav_efficiency    PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (9);
CREATE TABLE pav_power         PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P FOR VALUES IN (10);

-- 3. default partition for other attr
CREATE TABLE pav_default       PARTITION OF PRODUCT_ATTRIBUTE_VALUES_P DEFAULT;
```

```sql
EXPLAIN ANALYZE SELECT * FROM product_attribute_values WHERE attribute_id = 2;
```

Output | Result 59 | Query Plan | ======

QUERY PLAN

```
Seq Scan on pav_ram product_attribute_values  (cost=0.00..29289.00 rows=1500000 width=26) (actual time=0.019..171…
  Filter: (attribute_id = 2)
Planning Time: 0.198 ms
Execution Time: 223.375 ms
```

Базата скенира само **една** партиција (Seq Scan on pav_ram), а не ги допира другите. Со тоа времето на извршување е значително побрзо.

2. Табела: PRODUCT_PRICE_HISTORY (RANGE Partitioning)

Оваа табела содржи **2.25 милиони редови** и служи за следење на сите промени на цените. Податоците тука се од типот time-series.

Примена

Користевме **RANGE партиционирање по месеци** врз колоната change_date.

- **Перформанси:** Анализите на цените најчесто се прават за конкретен месец или квартал. Со овој пристап, базата пребарува само во релевантниот месечен опсег.
- **Одржување:** Старите податоци (на пр. од пред 2 години) може лесно да се избришат или архивираат со „откачување“ на целата партиција, без да се забавува главниот систем со тешки DELETE наредби.

```sql
-- new partitioned table
CREATE TABLE PRODUCT_PRICE_HISTORY_P (
  history_id  int8 NOT NULL,
  variant_id  int8 NOT NULL,
  old_price   numeric(12, 2) NOT NULL,
  new_price   numeric(12, 2) NOT NULL,
  change_date timestamp with time zone NOT NULL,
  -- pk includes date
  PRIMARY KEY (history_id, change_date)
) PARTITION BY RANGE (change_date);


SELECT
    date_trunc('month', change_date) AS month_start,
    count(*) AS num_records
FROM myschema.product_price_history
GROUP BY month_start
ORDER BY month_start;

-- monthly partitions
CREATE TABLE pph_2025_05 PARTITION OF PRODUCT_PRICE_HISTORY_P FOR VALUES FROM ('2025-05-01') TO ('2025-06-01');
CREATE TABLE pph_2025_06 PARTITION OF PRODUCT_PRICE_HISTORY_P FOR VALUES FROM ('2025-06-01') TO ('2025-07-01');


CREATE TABLE pph_2025_07 PARTITION OF PRODUCT_PRICE_HISTORY_P FOR VALUES FROM ('2025-07-01') TO ('2025-08-01');
CREATE TABLE pph_2025_08 PARTITION OF PRODUCT_PRICE_HISTORY_P FOR VALUES FROM ('2025-08-01') TO ('2025-09-01');
CREATE TABLE pph_2025_09 PARTITION OF PRODUCT_PRICE_HISTORY_P FOR VALUES FROM ('2025-09-01') TO ('2025-10-01');
CREATE TABLE pph_2025_10 PARTITION OF PRODUCT_PRICE_HISTORY_P FOR VALUES FROM ('2025-10-01') TO ('2025-11-01');
CREATE TABLE pph_2025_11 PARTITION OF PRODUCT_PRICE_HISTORY_P FOR VALUES FROM ('2025-11-01') TO ('2025-12-01');
CREATE TABLE pph_2025_12 PARTITION OF PRODUCT_PRICE_HISTORY_P FOR VALUES FROM ('2025-12-01') TO ('2026-01-01');
```

```sql
EXPLAIN ANALYZE
SELECT * FROM PRODUCT_PRICE_HISTORY_P
WHERE change_date BETWEEN '2025-11-01' AND '2025-11-30';
```

Output | Result 61 | Query Plan | Result 62

CSV

QUERY PLAN

```
Seq Scan on pph_2025_11 product_price_history_p  (cost=0.00..10920.00 rows=450000 width=40) (actual time=0.021..6…
  Filter: ((change_date >= '2025-11-01 00:00:00+00'::timestamp with time zone) AND (change_date <= '2025-11-30 00…
Planning Time: 0.179 ms
Execution Time: 76.018 ms
```

3. Табела: ORDER_STATUS_HISTORY (RANGE Partitioning)

Со **2.06 милиони редови**, оваа табела го следи животниот циклус на секоја нарачка (од CART до COMPLETED).

Примена

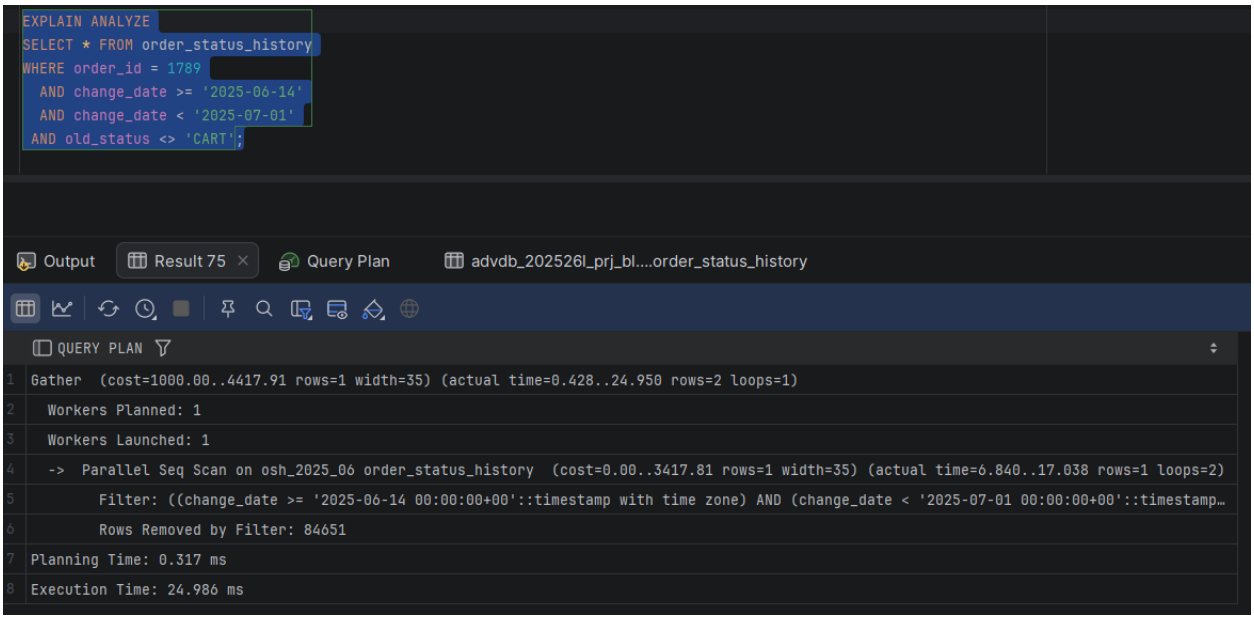
Повторно применивме **RANGE партиционирање по месеци**.

- **Логистичка ефикасност:** Бидејќи пратките во Македонија се доставуваат брзо, целиот историјат на една нарачка најчесто се наоѓа во рамките на една иста месечна партиција.
- **Write-Performance:** Куририте кои го ажурираат статусот на пратките во моментот, пишуваат само во „тековната“ партиција за овој месец. Тоа значи дека индексите со кои работи базата се мали и брзи, што овозможува моментално ажурирање на статусите на мобилната апликација на купувачот.

```
SET search_path TO myschema;

-- Креирање на мастер табелата за историја на статуси
CREATE TABLE ORDER_STATUS_HISTORY_P (
  history_id  int8 NOT NULL,
  order_id    int8 NOT NULL,
  old_status  varchar(255) NOT NULL,
  new_status  varchar(255) NOT NULL,
  change_date timestamp with time zone NOT NULL,
  -- PK мора да го содржи датумот поради партиционирањето
  PRIMARY KEY (history_id, change_date)
) PARTITION BY RANGE (change_date);

-- Партиции за 2025 година
CREATE TABLE osh_2025_05 PARTITION OF ORDER_STATUS_HISTORY_P FOR VALUES FROM ('2025-05-01') TO ('2025-06-01');
CREATE TABLE osh_2025_06 PARTITION OF ORDER_STATUS_HISTORY_P FOR VALUES FROM ('2025-06-01') TO ('2025-07-01');
CREATE TABLE osh_2025_07 PARTITION OF ORDER_STATUS_HISTORY_P FOR VALUES FROM ('2025-07-01') TO ('2025-08-01');
CREATE TABLE osh_2025_08 PARTITION OF ORDER_STATUS_HISTORY_P FOR VALUES FROM ('2025-08-01') TO ('2025-09-01');
CREATE TABLE osh_2025_09 PARTITION OF ORDER_STATUS_HISTORY_P FOR VALUES FROM ('2025-09-01') TO ('2025-10-01');
CREATE TABLE osh_2025_10 PARTITION OF ORDER_STATUS_HISTORY_P FOR VALUES FROM ('2025-10-01') TO ('2025-11-01');
CREATE TABLE osh_2025_11 PARTITION OF ORDER_STATUS_HISTORY_P FOR VALUES FROM ('2025-11-01') TO ('2025-12-01');
CREATE TABLE osh_2025_12 PARTITION OF ORDER_STATUS_HISTORY_P FOR VALUES FROM ('2025-12-01') TO ('2026-01-01');
```



Иако табелата order_status_history ги содржи сите промени, вклучувајќи го и иницијалниот статус CART, за потребите на **логистичката анализа и оптимизацијата на партициите**, системот го дефинира почетокот на оперативниот циклус од статусот PLACED. Од технички аспект, партиционирањето по change_date овозможува овие записи да бидат физички групирани според моментот на нивното настанување. Сепак, при пребарување и анализа на перформансите (на пр. брзина на обработка на нарачка), намерно се применува филтрирање на статусот CART. Ова е неопходно бидејќи CART претставува пред-продажна активност која може да трае со денови или месеци, па нејзиното вклучување во пресметките за 'време на испорака' би дало неточни резултати.