EPAM Systems, RD Dep., RD Dep.

POSTGRESQL FOR DWH AND ETL BUILDING

Partitioning and Parallel Execution

CONTENTS

1. PARTITIONING

Task Results: Provide queries where needed. Describe what happened and why with screenshots where needed.

1.1 TASK 1: USE INHERITANCE

Create table:

```
CREATE TABLE SALES_INFO
(
     id        INTEGER,
     category  VARCHAR(1),
     ischeck   BOOLEAN,
     eventdate DATE
);
```

Apply partitioning by using inheritance:

1. Create 4-5 child tables with partitioning by **eventdate** column. One partition is one year.
2. Create partition function for your tables. Use following as a template:

```
CREATE OR REPLACE FUNCTION partition_sales_info() RETURNS trigger
     as $$
  BEGIN
    IF (new.eventdate >= '2022-01-01'::DATE AND
        new.eventdate < '2023-01-01'::DATE) THEN
         INSERT INTO sales_info_2022 VALUES (new.*) ;
    ELSEIF (new.eventdate >= '2021-01-01'::DATE AND
            new.eventdate < '2022-01-01'::DATE) then
         INSERT INTO sales_info_2021 VALUES (new.*) ;
    ….
    ELSE
         RAISE EXCEPTION 'Out of range';
END IF;

RETURN NULL;
END;
$$ language plpgsql;
```

3. Create trigger for your function and tables. Use following as a template:

```
CREATE TRIGGER partition_sales_info_trigger
    BEFORE INSERT ON sales_info
        FOR EACH ROW EXECUTE PROCEDURE partition_sales_info();
```

4. Generate test data and insert in SALES_INFO table:

```
INSERT INTO SALES_INFO(id,category, ischeck, EventDate)
SELECT id
     ,('{"A","B","C","D","E","F","J","H","I","J","K"}'::text[])[(
     (RANDOM())*10)::INTEGER] category
      ,((1*(RANDOM())::INTEGER)<1) ischeck
      ,(NOW() - '10 day'::INTERVAL * (RANDOM()::int * 100))::
     DATE EventDate
FROM generate_series(1,10000000) id;
```

5. Update some rows in SALES_INFO and set another eventdate.

6. Create table SALES_INFO_SIMPLE with the same structure as SALES_INFO but without partitioning. Insert test data from the 5th step. Compare plans of different queries:
    - Select all
    - Select with range of dates
    - Select exact date
    - Count of all rows
    - Count of rows with range of dates
7. Delete one of partition (the oldest one). Create some general table like sales_info_3000 with the same structure as sales_info and add it as new partition.

1.2 TASK2: USE DECLARATIVE PARTITIONING

1. Create table SALES_INFO_DP with structure:
```
id        INTEGER,
category  VARCHAR(1),
ischeck   BOOLEAN,
eventdate DATE
```
   And make it partitioned by eventdate.
2. Create 4-5 child tables with partitioning by **eventdate** column. One partition is one year. Each child table should be partitioned by list on **category** column. Use 2 lists of values and one default partition here. As a result you should have SALES_INFO_DP table with composite partitioning by range and list.
3. Add date to partitioned table:

```
INSERT INTO SALES_INFO_DP(id,category, ischeck, EventDate)
SELECT id
    ,('{"A","B","C","D","E","F","J","H","I","J","K"}'::text[])[((
    RANDOM())*10)::INTEGER] category
      ,((1*(RANDOM())::INTEGER)<1) ischeck
      ,(NOW() - '10 day'::INTERVAL * (RANDOM()::int * 100))::
    DATE EventDate
FROM generate_series(1,10000000) id;
```

4. Update some rows in SALES_INFO_DP and set another category.
5. Compare plans of different queries for tables SALES_INFO_DP and SALES_INFO_SIMPLE:
    - Select all
    - Select with range of dates
    - Select exact date
    - Select exact category
    - Select a list of categories
    - Select a list of categories in exact date
    - Count of all rows
    - Count of rows with range of dates

6. For one of the child tables with range partition by eventdate split one list partition for two. For example:

   SALES_INFO_DP_2020_A PARTITION OF SALES_INFO_DP_2020 FOR VALUES IN ('A','B','C','D','E') => SALES_INFO_DP_2020_A PARTITION OF SALES_INFO_DP_2020 FOR VALUES IN ('A','B','C') and SALES_INFO_DP_2020_A PARTITION OF SALES_INFO_DP_2020 FOR VALUES IN ('D','E')

   Return partition ('A','B','C','D','E'). Drop newly created (for ('A','B','C') and ('D','E')).

2. PARALLEL EXECUTION

2.1 TASK 3: USE PARALLEL QUERING

1. Add parallel workers:
   ```
   set max_parallel_workers_per_gather=4;
   ```
2. Analyze plans for tables SALES_INFO, SALES_INFO_DP and SALES_INFO_SIMPLE by querying:
   a. Select all from tables
   b. Add order by eventadate
   c. Select count of all rows
   d. Add range of dates
   e. Add grouping by category
   f. Join SALES_INFO and SALES_INFO_DP on id and count rows on exact date.
3. Add indexes on any of table with partitions. Check how plans are change.