EPAM Systems, RD Dep., RD Dep.

POSTGRESQL DB FOR DWH AND ETL BUILDING

PostgreSQL Relational Structures

CONTENTS

1. PREREQUISITE TASK

Connect to your test_db and lab schema which was created on the previous module. Check search path parameter, add your schema into path if it is needed.

Please DROP a table person from previous module and recreate it. Fill the person table with data:

```
CREATE TABLE labs.person (
  id integer NOT NULL,
  name varchar(15)
);

INSERT INTO person VALUES(1, 'Bob');
INSERT INTO person VALUES(2, 'Alice');
INSERT INTO person VALUES(3, 'Robert');
```

Read about `EXPLAIN ANALYZE` command. Do not dive deep for now.

2. TABLES

2.1 TASK 1 – PERFOMANCE WITH UNLLOGED TABLES

Task Result: Add your personal description of what happened and why with screenshots where needed.

1. Create simple table

   ```
   labs.test_simple(a int,b int)
   ```

2. Perform insert and check how much time it takes:

   ```
   insert into test_simple values (generate_series(1,1000000));
   ```

   then

   ```
   insert into test_simple values (generate_series(1,5000000));
   ```

3. Create UNLOGGED table

   ```
   test_unlogged(a int,b int)
   ```

4. Perform insert and check how much time it takes:

   ```
   insert into test_unlogged values (generate_series(1,1000000));
   ```

   then

   ```
   insert into test_simple values (generate_series(1,5000000));
   ```

2.2 TASK 2 – UNDERSTANDING INHERITED TABLES

Task Result: Provide queries where needed. Add your personal description of what happened and why with screenshots where needed.

1. Create inherited table for person:

   ```
   CREATE TABLE labs.users(user_id INT GENERATED BY DEFAULT AS IDENTITY
   PRIMARY KEY,  login VARCHAR(30)) INHERITS (person);
   ```

2. Perform insert into new table:

   ```
   INSERT INTO users VALUES (1, 'new Bob', 'NewLogin');
   INSERT INTO users VALUES (999, 'TestUser', 'TestLogin');
   ```

3. Perform selects:

   ```
   SELECT * FROM users;
   SELECT * FROM person;
   SELECT * FROM ONLY person;
   ```

4. Modify data in person table, check result:

   ```
   UPDATE person
   SET name = 'not Bob'
   where id = 1;
   ```

   Rewrite `UPDATE` to change name for row where id=1 for person table only.

5. Alter person table and check structure and constraints on both users and person tables:

   ```
   ALTER TABLE person ADD COLUMN status integer DEFAULT 0;
   ALTER TABLE person ADD CONSTRAINT status CHECK (status in (0,1)) NO
   INHERIT;
   ALTER TABLE person ADD CONSTRAINT id UNIQUE (id, name);
   ```

3. INDEXES

3.1 TASK 3 - BRIN VS B-TREE

Task Result: Provide queries where needed. Add your personal description of what happened and why with screenshots where needed*.

* You could also use EXPLAIN ANALYZE for SELECT queries and check the plans differences.

1. Create table test_index:

```
CREATE TABLE labs.test_index (
    num       float NOT NULL,
    load_date timestamptz NOT NULL
);
```

2. Fill the table with a lot of test data:

   ```
   INSERT INTO test_index(num, load_date)
   SELECT random(), x
   FROM generate_series('2017-01-01 0:00'::timestamptz,
       '2021-12-31 23:59:59'::timestamptz, '10 seconds'::interval) x;
   ```

   Check the table size:

   ```
   SELECT pg_size_pretty(pg_relation_size('test_index'));
   ```

3. Perform select and check how much time it takes:

   ```
   SELECT date_trunc('year', load_date), max(num)
   FROM test_index
   WHERE load_date BETWEEN '2021-09-01 0:00' AND '2021-10-31 11:59:59'
   GROUP BY 1
   ORDER BY 1;
   ```

4. Create B-Tree Index on test_index table for load_date column. How long is this operation take? Repeat step 3. Check the index size(see step 1, change 'test_index' to name of your index). Drop the B-Tree index.

5. Create BRIN Index on test_index table for load_date column. How long is this operation take? Repeat step 3. Check the index size(see step 1, change 'test_index' to name of your index). Drop the BRIN index.

6. DROP test_index table.

3.2 TASK 4 - GIN VS GIST

Task Result: Provide queries where needed. Add your personal description of what happened and why with screenshots where needed.

1. Create a table and fill with test data:

   ```
   CREATE TABLE test_index AS SELECT id, md5(id::text) as t_hash
   FROM generate_series(1, 10000000) AS id;
   ```

   Check the table size(see Task 3).

2. Perform select and check how much time it takes:

   ```
   SELECT * FROM test_index WHERE t_hash LIKE '%ceea167a5a%';
   ```

3. Create GIST Index on test_index table for t_hash column. How long is this operation take? Repeat step 1. Check the index size(see Task 3). Drop the GIST index.

   ```
   CREATE EXTENSION pg_trgm;

   CREATE INDEX idx_text_index_gist ON test_index USING gist(t_hash
   gist_trgm_ops);
   ```

4. Create GIN Index on test_index table for t_hash column. How long is this operation take? Repeat step 1. Check the index size(see Task 3). Drop the GIN index.

```
CREATE INDEX idx_text_index_gin ON test_index USING gin (t_hash
gin_trgm_ops);
```

5. DROP test_index table.

4. FOREIGN DATA WRAPPERS

4.1 TASK 5 – CSV FILE AS A TABLE

Task Result: Provide queries where needed. Add your personal description of what happened and why with screenshots where needed.

1. Install file_fdw extension to database and create the SERVER to use, every FOREIGN TABLE requires a server:

```
CREATE EXTENSION file_fdw;
CREATE SERVER test_import FOREIGN DATA WRAPPER file_fdw;
```

2. Create the foreign table to connect to the CSV file (cities_list.csv):

```
CREATE FOREIGN TABLE labs.test_foreign_table
(
    LatD INT,
    LatM INT,
    LatS INT,
    NS  TEXT,
    LonD INT,
    LonM INT,
    LonS INT,
    EW TEXT,
    City TEXT,
    State TEXT
)
SERVER test_import
OPTIONS
(
    filename '\path_to_file\cities_list.csv',
    format 'csv',
    header 'true',
    delimiter ','
);
```

3. Select data from the foreign table and count of rows.

```
SELECT * FROM test_foreign_table;
SELECT count(*) FROM test_foreign_table;
```

4. Create Materialized view to select from this foreign table. Select from MView.

```
select count(*) from mview
```

5. Delete from file some rows. Select count again.

6. Refresh MView to update count of rows.