University of Edinburgh
School of Informatics
INFR11199 - Advanced Database Systems (Spring 2024)

Tutorial Sheet 4 - Playing with PostgreSQL

The purpose of this practical sheet is to familiarise you with the query execution engine of PostgreSQL. In particular, you will analyse a few queries and answer questions regarding their performance when turning different knobs of the execution engine. To answer the questions, you might find the following documentation links useful:

- Documentation of `EXPLAIN ANALYZE`:
  `https://www.postgresql.org/docs/14/sql-explain.html`.
- Making sense of the `EXPLAIN ANALYZE` output:
  `https://www.postgresql.org/docs/14/performance-tips.html`.
- PostgreSQL query planner documentation:
  `https://www.postgresql.org/docs/14/runtime-config-query.html`.
- How to create an index:
  `https://www.postgresql.org/docs/14/sql-createindex.html`.
- The system table `pg_class`:
  `https://www.postgresql.org/docs/current/catalog-pg-class.html`.

Prerequisites:

- Install PostgreSQL on your machine and start a PostgreSQL server (plenty of instructions online on how to do this, e.g., `http://postgresguide.com/setup/install.html`; any version will work). Make sure the command-line tool `psql` is working and you can use it to create tables and run queries.
- Download the bay-area-bike-sharing dataset from the course webpage. Unzip the archive and import the data into PostgreSQL using the provided scripts (e.g., by typing the command `psql < import.sql`).

1. **EXPLAIN and ANALYZE**

   For the following questions consider the query below:

   ```
   SELECT * FROM trip WHERE bike_id = 10;
   ```

   (a) Provide the PostgreSQL execution plan of the query and the SQL statement you use to generate the result.

   Based on the execution plan:

   i. What was the estimated cost (in arbitrary units)? ____________

   ii. What was the total runtime (in ms)? ____________

   iii. What was the estimated number of tuples to be output? ____________

   iv. What was the actual number of output tuples? ____________

(b) Create an index on the attribute `bike_id` on the table `trip`. Provide the SQL statement for that and the new execution plan of the query.

Based on the execution plan:

i. What was the estimated cost (in arbitrary units)? Faster than without index ______

ii. What was the total runtime (in ms)? ______

(c) Use the table `pg_class` to answer the following questions.

i. How many pages are used to store the index you created on table `trip`? Provide the answer and the query you use to generate the answer.

ii. How many tuples are in the index you created on column `bike_id`? Provide the answer and the query you use to generate the answer.

Without running ANALYZE, the pg_class does not have info about new deletions in the tuples

iii. How many tuples are in the table `weather`, according to `pg_class`?

iv. In the table `weather`, delete all records of which date is earlier than '2013-10-01'. Provide the SQL statement you use.

v. After deletion, rerun your query from step 3. Is the new result equal to the result of running `SELECT COUNT(*) FROM weather`?

◯ Yes ◯ No

vi. `ANALYZE` is a Postgres function used to collect statistics about a database. You want to use it especially after considerable number of modifications happen to that database. Run `ANALYZE`, and then rerun your query from step 3 again. Is the new result equal to the result of running `SELECT COUNT(*) FROM weather`?

◯ Yes ◯ No

2. **Using Indexes**

In this question, we will learn the conditions under which indexes may or may not be used by the query optimizer.

(a) Create an index on the column `start_station_name` on the table `trip`. Provide the SQL command you use.

(b) For each of those queries, answer Yes if the index you created on `trip.start_station_name` was used in the execution plan, or No otherwise:

i.
```
SELECT * FROM trip
 WHERE start_station_name like 'San';
```

◯ Yes ◯ No

ii.
```
SELECT * FROM trip
 WHERE start_station_name like '%San';
```

◯ Yes ◯ No

iii.
```
SELECT * FROM trip
 WHERE start_station_name BETWEEN 'San␣Francisco' AND 'San␣Jose'
   AND end_station_name > 'San';
```

◯ Yes ◯ No

iv.
```
SELECT * FROM trip
 WHERE start_station_name BETWEEN 'San Francisco' AND 'San Jose'
    OR end_station_name > 'San';
```

◯ Yes ◯ No

(c) Make sure you still have an index on the column `trip.bike_id` (you can verify this using `\di` in `psql`). For each of those queries, answer which indexes are used in their execution plans.

i.
```
SELECT * FROM trip
 WHERE start_station_name BETWEEN 'San Francisco' AND 'San Jose'
   AND bike_id < 10;
```

- ◯ Only the index on `start_station_name` was used.
- ◯ Only the index on `bike_id` was used.
- ◯ Both indexes were used.
- ◯ None of the indexes were used.

ii.
```
SELECT * FROM trip
 WHERE start_station_name BETWEEN 'San Francisco' AND 'San Jose'
   AND bike_id < 500;
```

- ◯ Only the index on `start_station_name` was used.
- ◯ Only the index on `bike_id` was used.
- ◯ Both indexes were used.
- ◯ None of the indexes were used.

iii.
```
SELECT * FROM trip
 WHERE start_station_name BETWEEN 'San Francisco' AND 'San Jose'
   AND bike_id BETWEEN 500 AND 510;
```

- ◯ Only the index on `start_station_name` was used.
- ◯ Only the index on `bike_id` was used.
- ◯ Both indexes were used.
- ◯ None of the indexes were used.

iv.
```
SELECT * FROM trip
 WHERE start_station_name > 'San Francisco'
   AND bike_id < 500;
```

- ◯ Only the index on `start_station_name` was used.
- ◯ Only the index on `bike_id` was used.
- ◯ Both indexes were used.
- ◯ None of the indexes were used.

(d) Answer the questions below for the query:

```
SELECT * FROM trip
 WHERE bike_id BETWEEN 10 AND 20;
```

i. Was the index on `bike_id` used? ◯ Yes ◯ No

ii. What percentage of the total records in the table `trip` was returned? Provide a percent and retain two significant figures. __________

(e) Answer the questions below for the query:

```
SELECT * FROM trip
 WHERE bike_id > 10;
```

i. Was the index on `bike_id` used? ◯ Yes ◯ No

ii. What percentage of the total records in the table `trip` was returned? Provide a percent and retain two significant figures. __________

(f) Answer the questions below for the query:

```
SELECT * FROM trip
 WHERE bike_id > 10 ORDER BY start_time;
```

i. Which method was used for sorting? __________
ii. Where did the sorting happen? ◯ Memory ◯ Disk
iii. How much space was used for sorting? __________
iv. What was the total runtime (in ms)? __________

(g) Display PostgreSQL working memory with `SHOW work_mem;`. Increase PostgreSQL working memory with the command `SET work_mem = '128MB';`. For the same query from part vi., answer the following questions:

i. Which method was used for sorting? __________
ii. Where did the sorting happen? ◯ Memory ◯ Disk
iii. How much space was used for sorting? __________
iv. What was the total runtime (in ms)? __________

(h) Execute the command `RESET work_mem;` to get PostgreSQL working memory back to the default value (or your answers for the next questions will turn out wrong).

3. **Joins**

In this question, we will learn about different methods used by PostgreSQL for executing joins. Make sure you reset `work_mem` to its default value (i.e., `RESET work_mem;`).

Answer the following questions based on the query below:

```
SELECT trip.*, station.city
  FROM trip, station
 WHERE trip.start_station_id = station.station_id
   AND bike_id < 200;
```

(a) Provide the query plan for the above query.

Based on the execution plan:

i. Which join method was used? ____________

ii. What was the estimated cost (in arbitrary units)? ____________

iii. What was the total runtime (in ms)? ____________

(b) Execute the command `SET enable_hashjoin = false;` to disable hash joins. Provide the new query plan.

Based on the execution plan:

i. Which join method was used? ____________

ii. What was the estimated cost (in arbitrary units)? ____________

iii. What was the total runtime (in ms)? ____________

(c) Execute the command `SET enable_mergejoin = false;` to disable merge joins. Provide the new query plan.

Based on the execution plan:

i. Which join method was used? ____________

ii. What was the estimated cost (in arbitrary units)? ____________

iii. What was the total runtime (in ms)? ____________

(d) Execute the command `SET enable_indexscan = false; SET enable_bitmapscan = false;` to disable index scans. Give the new plan.

Based on the execution plan:

i. Which join method was used? ____________

ii. What was the estimated cost (in arbitrary units)? ____________

iii. What was the total runtime (in ms)? ____________

(e) Execute these commands to re-enable the different joins.

```
RESET enable_mergejoin;
RESET enable_hashjoin;
RESET enable_indexscan;
RESET enable_bitmapscan;
```