Programming Assignment #2.1 CAS CS 460

Query Optimization

Due: 11/30 11:59 pm on gradescope.

This programming assignment is for groups of two. If there is a strong reason you wish to work on it alone, please reach out to the teaching staff and explain why.

General

First make sure that you have PostgreSQL and MonetDB installed in a cloud environment (Azure). Instructions to get started on an Azure account are here. You can also find this **Azure Support Manual** in Piazza, under the Resources tab.

You will need PostgreSQL for both PA2.1 and PA2.2. MonetDB will only be used for PA2.2. Overall, the instructions provided in this document may differ depending on the Operating System you are using. You may need to make minor modifications to make the set-up work.

Query Optimization Task

1.1 Introduction

In this task, you will carry out several exercises involving the optimization of relational queries using the PostgreSQL query optimizer and the visualization command EXPLAIN. You need to read parts of the PostgreSQL documentation to be able to complete this task. To be specific, you need to get familiar with the EXPLAIN, ANALYZE and the INFORMATION_SCHEMA Table commands of PostgreSQL. (links are provided in the helpful resources section at the end of this document)

1.2 Setup

First, make sure that PostgreSQL service is started and runs in the background. Then, make sure you start **postgres**. Create a database called **TASK1DATABASE**, if you have not done so already. Then, you will need to download this file into your local machine and unzip it. This file can also be found in Piazza, under the Resources tab. You will then transfer this file into the remote virtual machine on Azure using the SCP command. Further, we will setup the created database with the data that we just downloaded. Follow the instructions to **setup the database**, **switching** to task1database and perform a **sanity check** as described in the Azure Support Manual.

Relation Schema:

We will use four tables in this experiment: part, supplier, partsupp, and lineitem.

```
part (p_partkey integer, p_name varchar(55),  p_mfgr character
(25), p_brand character(10), p_type varchar(25), p_size intege
```

```
r, p_containercharacter(10), p_retailprice numeric(20,2), p_co
mment varchar(23), primary key (p_partkey));

supplier ( s_suppkey integer, s_name char(25), s_address varch
ar(40), s_nationkey integer, s_phone character(15), s_acctbal
numeric(20,2), s_commentvarchar(101), primary key (s_suppkey));

partsupp (ps_partkey integer, ps_suppkey integer, ps_availqty
integer, ps_supplycost numeric(20,2), ps_comment varchar(199),
primary key(ps_partkey, ps_suppkey));

lineitem( l_orderkey integer, l_partkey integer, l_suppkey int
eger, l_linenumber integer, l_quantity numeric(20,2), l_extend
edprice numeric(20,2), l_discountnumeric(3,2), l_tax numeric(3
,2), l_returnflag character(1), l_linestatus character(1), l_s
hipdate date, l_commitdate date, l_receiptdate date, l_shipins
tructcharacter(25), l_shipmode character(10), l_comment varcha
r(44), primary key (l_orderkey, l_linenumber);
```

1.3 Exercises

When EXPLAIN is used with an explainable statement, PostgreSQL displays information from the optimizer about the statement execution plan. That is, PostgreSQL explains how it would process the statement, including information about how tables are joined and in which order. In general, use - **EXPLAIN (ANALYZE true, COSTS true, FORMAT json)** - to get the evaluation plan because it gives much more information about the plan. Use the actual execution of the query on terminal or profile information for query execution times.

For each of the following exercises, please provide screenshot(s) of the result after running the corresponding queries.

1.3.1 Statistics of the tables

We will first examine the statistics for table lineitem. Answer the following questions.

1. How many records are there actually in "lineitem"? What is the estimated value by the query optimizer? How do you find these values (command or SQL)?
2. Is the value used by the query optimizer exact? If not, why?

1.3.2 Index on perfect match query

We will check how index affects query optimization and performance.

First, examine the following query without index:

SELECT * FROM lineitem WHERE L_TAX = 0.07;

1. What is the estimated total cost of executing the best plan?
2. What is the estimated result cardinality for this plan?
3. How does the query optimizer obtain the result cardinality? Is it a reasonable one?
4. Which access method (access path) does the optimizer choose?

Create an index "ltax_idx" on the attribute "L_TAX".

5. Which access method does the optimizer consider to be the best now? Is the estimated result cardinality better now? Why?
6. Compare the two plans (without and with index). Explain briefly why access method in (5) is cheaper than the previous one.

1.3.3 Index on range select

Consider the following query:

```
SELECT * FROM lineitem WHERE L_QUANTITY < 45;
```

1. How many tuples does the query optimizer think will be returned? What is the estimated total cost?
2. What is the access method used?

Create an index "l_qty_idx" on the attribute "L_QUANTITY". Consider now the following query:

```
SELECT * FROM lineitem WHERE L_QUANTITY < 3;
```

3. What is the estimated total cost now? Is it correct? In what order would the tuples be returned by this plan?
4. Explain why one of the access methods is more expensive than the other.

1.3.4 Join algorithm

Consider the following query:

```
SELECT DISTINCT s_name FROM supplier, partsupp

WHERE s_suppkey = ps_suppkey AND ps_availqty < 40;
```

Answer the follow questions:

1. Write down the best plan estimated by the optimizer (in plan tree form- just provide a screenshot). What is the estimated total cost?
2. What is the join algorithm used in the plan? Explain how the system reads the two relations (what access path uses).
3. According to the optimizer, how many tuples will be retrieved from partsupp? How many from supplier? Are these estimations correct?

4. Can you add an index to improve the performance of the plan? Which index you will create and on which attribute? What is the new plan that is executed and what is its cost?
5. After you created the index, check the estimation of the tuples retrieved from partsupp. Is it correct? If yes, why?

1.4 Helpful Resources

Explain: https://www.postgresql.org/docs/9.4/using-explain.html

Row Estimation: https://www.postgresql.org/docs/8.1/planner-stats-details.html

https://www.postgresql.org/docs/8.3/row-estimation-examples.html

Analyze: https://www.postgresql.org/docs/9.1/sql-analyze.html

Information Schema: https://www.postgresql.org/docs/9.1/information-schema.html

Steps to execute a script:

- Open the terminal. Go to the directory where you have your **script**.
- **Run** the **script** by typing: ./ScriptName.sh