EECS 484 W13 Project 3

Query Optimization

Introduction

In this assignment, we will work with a small subset of an online e-commerce dataset called Dell DVD Store (http://linux.dell.com/dvdstore/). This dataset has information about customers, products, and orders. Although there are 8 tables in the dataset, our exercises will focus on the following two tables (you can see the schema and indices on a table by doing `\d+ tablename`):

```
Customers (customerid, firstname, lastname, city, state, country, …)
Orders (orderid, orderdate, customerid, netamount, tax, totalamount)
```

Getting Started

We will be using PostgreSQL 8.4 (also known as Postgres) database system as it has very powerful tools for query performance analysis and optimization. See the following Google Doc for instructions on how to access Postgres and initialize your database.
https://docs.google.com/document/d/1jqmAGdq4JpYhplkd1GqPPu-VpQiZlE6sjbEl-zGgRVA/edit?usp=sharing

Helpful Links

The following are links to Postgres documentations relevant to this assignment. Be prepared to look up the documentation as you work on the exercises.

- Full PostgreSQL 8.4.16 documentation:
http://www.postgresql.org/docs/8.4/static/index.html

- Statistics used by the query planner
http://www.postgresql.org/docs/8.4/static/planner-stats.html

- How to manipulate the query planner (such as disabling the use of a join algorithm)
http://www.postgresql.org/docs/8.4/static/runtime-config-query.html

- Syntax of EXPLAIN command
http://www.postgresql.org/docs/8.4/static/sql-explain.html

- How to use EXPLAIN command and interpret its output
http://www.postgresql.org/docs/8.4/static/using-explain.html

- Creating an index

http://www.postgresql.org/docs/8.4/static/sql-createindex.html

- Creating a clustered index

http://www.postgresql.org/docs/8.4/interactive/sql-cluster.html

What to Submit

You need to submit three documents:

1) P3answers.pdf
    - Contains the answers to the following 7 exercises and SQL commands that you used to answer each exercise. Please try to keep your commands as neat as possible.

2) P3output.pdf
    - Contains the outputs of EXPLAIN commands for all the exercises (numbered by exercise #).

3) P3queries.sql
    - Contains all SQL queries in a proper sequence (including reloading of database, ENABLE and DROP commands). It should be a script that does all the given exercises in a sequence. Include some comments to indicate the question to which each query applies.

Exercises

Load the Dell DVD Store dataset into Postgres (not sqlplus or sqlite) as described in the Google Doc (see *Initializing the Database*) that are posted with this homework. Then, answer the following questions.

Note:

- Bitmap index is disabled in the driver file (setup_db.sql) because it is not covered in this class.
- The answers you get for various exercises may differ slightly across different runs.

Exercise 1: System Catalog Queries (10 Points)

Use the Postgres `\d+` command or catalog commands to answer the following questions. You will be using `pg_class` and `pg_stats` tables.

a) **(2 points)** How many indexes are built on the Customers relation?
Solution:

b) **(2 points)** Name two relations and two indexes (from the tables and indexes created for DVD Store data) that occupy the most number of disk pages.

Solution:

c) **(3 points)** Find out the number of distinct values in each column of the Customers table using information stored in the Postgres catalog (`pg_stats` table). Write down the query that you used to obtain this information. Based on what you found, in addition to the attributes that are already indexed, suggest two other attributes that are suitable for building a B-tree index. B-trees are good for range-style queries and sorting. Hash indexes are good for equality-type queries.
Solution:

d) **(3 points)** Find out the number of distinct values in each column of the Customers table ***without*** using the Postgres catalog. Write down the query that you used to obtain this information. How does the expected count from the catalog compare with the actual count values? Show the comparison in a table.
Solution:

Exercise 2: Equality Query (10 points)

We will use the EXPLAIN command to examine possible optimizations for the following query:

```
SELECT * FROM customers WHERE country = 'Japan';
```

Note: You need to run `VACUUM` and `ANALYZE` after adding or dropping indexes in Postgres. These commands update statistics about tables so that Postgres can estimate the cost of various queries accurately. If you get warnings that certain tables can only be vacuumed by superuser, ignore them.

```
VACUUM;
ANALYZE;
```

a) **(2 points)** What is the estimated cardinality of this query (number of rows expected in the result)? How does the estimated cardinality compares with the actual value?
Solution:

b) **(4 points)** Write down the formula and then show the calculation by which the optimizer computes the expected execution cost for the above query. You can use the default time measuring units of Postgres. See the following Postgres documentation on how to come up with the formula.
http://www.postgresql.org/docs/8.4/static/using-explain.html
Solution:

c) **(4 points)** Using the query execution plan notation discussed in class, draw the execution plan selected by the query optimizer (see lecture notes on Relational Algebra or textbook Chapter 13)
Solution:

Exercise 3. Equality Query with Indexes (15 Points)

Run the following commands to reload the entire Dell DVD Store database and update stored statistics.

```
\i setup_db.sql;
VACUUM;
ANALYZE;
```

Create a non-clustered index called `customers_country` on attribute `country` of `customers` table, then update the statistics using VACUUM and ANALYZE.

```
CREATE INDEX customers_country ON customers(country);
VACUUM;
ANALYZE;
```

a) **(1 point)** How many disk pages does the `customers_country` index occupy?
Solution:

b) **(3 points)** Rerun EXPLAIN from Exercise 2. Which access method does the query optimizer selects now? What is the estimated total cost of executing the best plan for this query?

Note: If you have already clustered your `customers` tables along the `country` attribute during some previous execution of Exercise 3(d), then you must reload the dataset by doing:

```
\i setup_db.sql
```

Otherwise, the dataset is already clustered and so even if you drop and recreate the index as non-clustered index, it will actually be a clustered index because the underlying data is already clustered.
Solution:

c) **(4 points)** How does the non-clustered index compare to using no index as in Exercise 2? If it does not lead to significant benefits, explain possible reasons.
Solution:

Now, change the `customers_country` index into a clustered index, and then update the statistics using as follows:

```
CLUSTER customers_country ON customers;
VACUUM;
```

```
ANALYZE;
```

d) **(3 points)** Rerun EXPLAIN from Exercise 2. What access method does the query optimizer selects now? What is the estimated total cost of executing the best plan for these queries?
Solution:

e) **(4 points)** How does the best plan of clustered index compare to the best plan of non-clustered index for the query from Exercise 2? Explain any differences.
Solution:

Exercise 4: Join Query (10 Points)

Run the following commands to reload the entire Dell DVD Store database and update stored statistics.

```
\i setup_db.sql;
VACUUM;
ANALYZE;
```

Answer the following questions based on the following query.

```
SELECT totalamount
FROM Customers C, Orders O
WHERE C.customerid = O.customerid AND C.country = 'Japan';
```

a) **(3 points)** Using the query execution plan notation discussed in class, draw the execution plan selected by the query optimizer.
Solution:

b) **(2 points)** Which join algorithm does the query optimizer use? What is its total estimated cost? What is the estimated result cardinality of this query?
Solution:

c) **(3 points)** Disable the join algorithm selected by the optimizer in the answer to part (b) by using the SET command (see accompanying Postgres documentation). Which join algorithm is used now? What is the total estimated cost? Draw the query tree again (as in part (a)).
Solution:

d) **(2 points)** Now disable both join algorithms that are used in part (b) and (c). Which join algorithm is used now? What is the total estimated cost?
Solution:

Exercise 5. Join Selection (15 Points)

Run the following commands to reload the entire Dell DVD Store database and update stored statistics.

```
\i setup_db.sql;
VACUUM;
ANALYZE;
```

Enable all the disabled join algorithms in Exercise 4 and consider the following two queries.

Query 5.1

```
SELECT AVG(totalamount) as avgOrder, country
FROM Customers C, Orders O
WHERE C.customerid = O.customerid
GROUP BY country
ORDER BY avgOrder;
```

Query 5.2

```
SELECT *
FROM Customers C, Orders O
WHERE C.customerid = O.customerid
ORDER BY C.customerid;
```

a) **(5 points)** What join algorithm does the query optimizer select for Query 5.1? What is the total expected cost? Disable the selected join algorithm. What join algorithm is selected now and what is the expected total cost (not just the cost of join)?
Solution:

b) **(5 points)** Enable the join algorithms disabled in part (a). What join algorithm does the query optimizer select for Query 5.2? What is the total expected cost? Disable the selected join algorithm. What join algorithm is selected now and what is the expected total cost (not just the cost of join)?
Solution:

c) **(5 points)** If you find that the optimizer selected different join algorithms for Query 5.1 and 5.2, explain why the optimizer may have selected different join algorithms.
Solution:

Exercise 6. Correlated Subquery (15 Points)

Run the following commands to reload the entire Dell DVD Store database and update stored statistics.

```
\i setup_db.sql;
VACUUM;
ANALYZE;
```

A *correlated sub-query* (also known as a synchronized sub-query) is a sub-query (a query nested inside another query) that uses values from the outer query in its WHERE clause. The sub-query is evaluated once for each row processed by the outer query. Correlated sub-queries are normally very inefficient because they require per-row processing instead of set-oriented processing. We can do it more efficiently by observing that correlated sub-queries have the following structure:

```
for each (val in X) {
   SubResult = CS(val)
   Process the SubResult
}
```

Here `val` is the correlation attribute (`C.customerid` in Query 6.1) and X is the set of values with which the correlated query is invoked. To optimize the execution time, we need to decouple the execution of CS from the execution of outer block. We can do this by creating a view that stores the computed values `CS(val)` of all distinct values of `val` in X.

In this exercise, we will look into optimization of correlated sub-queries. Consider the following query that retrieves all customers who ordered more than 4 items.

Query 6.1

```
SELECT C.customerid, C.lastname
FROM Customers C
WHERE
4 < (
      SELECT COUNT(*)
      FROM Orders O
      WHERE O.customerid = C.customerid);
```

a) **(1 point)** Using EXPLAIN, what is the estimated total cost of executing the best plan for Query 6.1?
Solution:

b) **(5 points)** Write an SQL query to create a view `OrderCount(customerid, numorders)` that represents the total number of orders for each customer.
Solution:

c) **(5 points)** Write a SQL query (call it Query 6.2) using the view `OrderCount` that will produce the same result as Query 6.1 without using any nested sub-queries.
Solution:

d) **(4 points)** What is the estimated total cost for executing Query 6.2? How does it compare with the estimated cost for the nested query from part (a)?
Solution:

Exercise 7. Query Optimization (15 Points)

Run the following commands to reload the entire Dell DVD Store database and update stored statistics.

```
\i setup_db.sql;
VACUUM;
ANALYZE;
```

Let us define ORank (Order Rank) of a customer X as the total number of customers who have more number of orders than customer X. For example, Smith's ORank is 32 if there are 32 customers with greater number of orders than that of Smith. Consequently, customers with the highest number of orders will have ORank equal to zero. Note that there might be multiple users with the same ORank (those with the same number of orders).

Query 7.1 finds the `customerid`, `lastname`, and `numorders` (number of orders) of customers from Japan with ORank <= 5. In this query, we have restricted ourselves to only the customers from Japan; otherwise the execution time will be extremely long. Therefore, for this question, ORank is defined with respect to other customers from Japan only.

Query 7.1

```
SELECT customerid, lastname, numorders
FROM (
SELECT C.customerid, C.lastname, count(*) as numorders
           FROM Customers C, Orders O
           WHERE C.customerid = O.customerid AND C.country = 'Japan'
           GROUP BY C.customerid, lastname) AS ORDERCOUNTS1
     WHERE 5 >= (SELECT count(*)
           FROM (
           SELECT C.customerid, C.lastname, count(*) as numorders
           FROM Customers C, Orders O
```

```
            WHERE C.customerid=O.customerid AND C.country = 'Japan'
            GROUP BY C.customerid, lastname) AS ORDERCOUNTS2
WHERE ORDERCOUNTS1.numorders < ORDERCOUNTS2.numorders)

ORDER BY customerid;
```

a) **(1 point)** Using EXPLAIN, what is the estimated total cost of executing the best plan for Query 7.1?
Solution:

b) **(5 points)** Write a SQL query to create a view `MoreFrequentJapanCustomers(customerid, oRank)` that finds the ORank of each customer in Japan with respect to other customers from Japan. You may find it useful to create a view `OrderCountJapan(customerid, numorders)` where you store the number of orders by each customer from Japan and use that view to create your `MoreFrequentJapanCustomers` view.
Solution:

c) **(5 points)** Write a SQL query (call it Query 7.2) using the views from part (b) that will produce the same result as Query 7.1, but in a more efficient way without using any nested sub-queries.
Solution:

d) **(4 points)** Using EXPLAIN, what is the estimated total cost of executing the best plan for Query 7.2? How does it compare to the answer from part (a). Explain possible reasons for the differences.
Solution: