Carnegie Mellon University
Department of Computer Science
15-415/615 - Database Applications
C. Faloutsos, A. Pavlo , Spring 2015

Homework 5 (by Elomar Souza)

Due: **hard and e-copy**, at **1:30pm, Mar. 17th, 2015**

IMPORTANT – what to hand in:

1. **Hard copy** of your answers (including SQL statements and database output, whenever stated by the question), in **class** at 1:30pm, on Thursday, Mar. 17th, 2015.
2. **tar-file**, with your queries, via *Blackboard*, before the due date (Mar. 17th, 2015, 1:30pm). See page 2 ('*What to deliver*'), for details.

Reminders

- *Plagiarism*: Homework is to be done **individually**.
- *Typeset* all of your answers. Handwritten work will get zero points.
- *Late homeworks*: Standard policy: email (a) to all TAs, (b) with the subject line exactly `15-415 Homework Submission (HW 5)`, and (c) the count of slip-days you are using.

For your information:

- Graded out of **100** points
- **4** questions total
- Expected effort: ≈ 4-9h (30min-1h to set up PostgreSQL; 1-2h per question).

Revision : 2015/02/25 21:51

Question	Points	Score
EXPLAIN and ANALYZE	25	
Using indexes	25	
Joins	25	
Optimizing Joins	25	
Total:	100	

Preliminaries

Database set-up

Before starting the homework, follow the instructions for importing the data you'll work with, available at `http://www.cs.cmu.edu/~christos/courses/dbms.S15/hws/HW5/postgresql-setup.html`.

For your convenience, a file with all the queries used in the homework is available at `http://www.cs.cmu.edu/~christos/courses/dbms.S15/hws/HW5/hw5-queries.sql`. You can copy the queries from the file to your `psql` session, to avoid copying-from-pdf errors.

What to deliver: Check-list

The two items, as mentioned on the front page:

1. **Hard copy**:
   - What: hard copy of your answers (including **SQL queries**, and **database output**, whenever stated by the question)
   - When: Mar. 17th, 2015, 1:30pm
   - Where: in class

   As before, please separate your answers for each question, providing on all answers the usual information (`course#`, `Homework#`, `Question#`, `Andrew ID`, `name`).
2. **tar-file**:
   - What: A `tar` file (`<your-andrew-id>.tar`) with all your code - no need to include the answers.
   - When: Mar. 17th, 2015, 1:30pm
   - Where: on *Blackboard*, under 'Assignments'/'Homework #5'

   Please use the template provided at `www.cs.cmu.edu/~christos/courses/dbms.S15/hws/HW5/hw5.tar` – just insert your SQL query to each place-holder `.sql` file.

Introduction

The purpose of this homework is to make you familiar with the query execution engine of PostgreSQL. In particular, you will have to analyze a few queries, and answer questions regarding their performance when turning different knobs of the execution engine.

In order to answer the questions, you might find the following documentation links useful:

- Documentation of `EXPLAIN ANALYZE`:
  http://www.postgresql.org/docs/8.4/static/sql-explain.html.
- Making sense of the `EXPLAIN ANALYZE` output:
  http://www.postgresql.org/docs/8.4/static/performance-tips.html.
- PostgreSQL query planner documentation:
  http://www.postgresql.org/docs/8.4/static/runtime-config-query.html.

Homework 5 continues…

- How to create an index:
  http://www.postgresql.org/docs/8.4/static/sql-createindex.html.
- The system table `pg_class`:
  http://www.postgresql.org/docs/8.4/static/catalog-pg-class.html.

When creating an index, do not alter PostgreSQL default options (type B-Tree, unclustered).

Database schema

The database you'll use in this exercise tracks users behavior on an ecommerce website. As typical for both large (e.g. Amazon) and small e-commerce portals, it tracks **user sessions**, **clicks**, and **purchases**.

The `sessions` table has data on each particular visit to the website. Its fields are: `sessid` (int, primary key), `zipcode` (int), `sex` (char), `agegroup` (varchar), and `browser` (varchar). For example, the tuple

(1, 90755, 'M', "5-19", "Chrome")

means that the #1 session happened at zipcode 90755, and was a visit from a Male, age between 5 and 19, using Chrome.

The `clicks` table tracks the time, item, and category of each click made during a particular session. Its fields are: `sessid` (int), `created` (timestamp), `itemid` (int), and `catid` (int).

The `purchases` table tracks the item, price, and quantity of each purchase made during a particular session. Its fields are: `sessid` (int), `created` (timestamp), `itemid` (int), `price` (numeric), and `quantity` (int).

Homework 5 continues…

Question 1: EXPLAIN and ANALYZE . . . . [25 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

In this question, you'll learn how to use `EXPLAIN` and `ANALYZE` to understand the impact of indexes on simple queries.

Answer the questions based on the query below:

```
SELECT * FROM clicks
WHERE itemid BETWEEN 214800000 AND 214819999;
```

(a) [**3 points**] Provide the execution plan of the query. Provide the SQL statement you used and its output.

(b) Based on the execution plan:

  i. [**1 point**] What was the estimated cost of the query? (in arbitrary units)

  ii. [**1 point**] What was the total runtime? (in ms)

(c) [**3 points**] Create an index on the attribute `itemid` on the table `clicks`.[1] Provide the SQL statement.

(d) [**3 points**] Provide the new execution plan of the query, with the index in place.

(e) Based on the new execution plan:

  i. [**1 point**] What was the estimated cost of the query? (in arbitrary units)

  ii. [**1 point**] What was the total runtime? (in ms)

  iii. [**1 point**] What was the estimated number of tuples to be output?

  iv. [**1 point**] What was the actual number of tuples to be output?

(f) Use the table `pg_class` to answer the following questions:

  i. [**2 points**] How many pages are used to store the table `clicks`?

  ii. [**2 points**] How many tuples are in the table `clicks`, according to `pg_class`?

  iii. [**2 points**] Is that number always equal to the result of running `SELECT COUNT(*) FROM clicks`?

  iv. [**2 points**] How many pages are used to store the index you created?

  v. [**2 points**] How many tuples are in the index?

[1] Using the default PostgreSQL options.

Question 2: Using indexes .... [25 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

In this question, you'll learn the conditions under which indexes may or may not be used by the query optimizer.

Make sure you have an index on the column `clicks.itemid`, created in Q1-(c).

(a) For each of those queries, answer (yes) if the index you created for Q1, item (c) was used or (not) if it wasn't:

i. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid BETWEEN 214800000 AND 214819999;
```
ii. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid > 214800000;
```
iii. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid BETWEEN 214800000 AND 214819999
AND created > '2014-04-01';
```
iv. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid BETWEEN 214800000 AND 214819999
AND created IS NULL;
```
v. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid BETWEEN 214800000 AND 214819999
OR created > '2014-04-01';
```
vi. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid = 214507226;
```
vii. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid != 214507226;
```

(b) [**1 point**] Create an index on the column `created` on the table `clicks`.[2] Provide the SQL command.

(c) For each of those queries, answer (1) if only the index on `itemid` was used, (2) if only the index on `created` was used, (3) if both were used, or (4) if neither one of the indexes were used:

i. [**1 point**]
```
SELECT * FROM clicks
```

[2]Using the default PostgreSQL options.

Question 2 continues...

```
WHERE itemid BETWEEN 214800000 AND 214819999
AND created > '2014-04-01';
```

ii. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid BETWEEN 214800000 AND 214819999
AND created BETWEEN '2014-04-01' AND '2014-04-02';
```

iii. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid BETWEEN 214800000 AND 214819999
OR created BETWEEN '2014-04-01' AND '2014-04-02';
```

iv. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid < 214819999
OR created BETWEEN '2014-04-01' AND '2014-04-02';
```

v. [**1 point**]
```
SELECT * FROM clicks
WHERE itemid < 214819999
AND created BETWEEN '2014-04-01' AND '2014-04-02';
```

(d) For the query `SELECT * FROM clicks WHERE created BETWEEN '2014-04-01' AND '2014-04-02';` answer the following questions:

i. [**1 point**] Was the index on `created` used?

ii. [**1 point**] What percentage of the total records in the table `clicks` was returned?

(e) For the query `SELECT * FROM clicks WHERE created BETWEEN '2014-04-01' AND '2015-12-31';` answer the following questions:

i. [**1 point**] Was the index on `created` used?

ii. [**1 point**] What percentage of the total records in the table `clicks` was returned?

(f) For the query `SELECT * FROM clicks WHERE itemid BETWEEN 214800000 AND 214819999 ORDER BY itemid;`, answer the following questions:

i. [**1 point**] Which method was used for sorting?

ii. [**1 point**] Where did the sorting happen – memory or disk?

iii. [**1 point**] How much space was used for sorting?

iv. [**1 point**] What was the total runtime? (in ms)

(g) Increase PostgreSQL working memory with the command `SET work_mem = '25MB';`. For the same query as above, answer the following questions:

i. [**1 point**] Which method was used for sorting?

ii. [**1 point**] Where did the sorting happen – memory or disk?

iii. [**1 point**] How much space was used for sorting?

iv. [**1 point**] What was the total runtime? (in ms)

Question 2 continues. . .

(h) **[0 points]** 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).

Homework 5 continues. . .

Question 3: Joins....................................[25 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

In this question, you'll learn more about the different methods used by PostgreSQL for executing joins.

Make sure you reset `work_mem` to its default value, as per Q2-(h).

Answer the questions based on the query below:

```
SELECT sessions.*, purchases.*
FROM sessions,  purchases
WHERE sessions.sessid = purchases.sessid;
```

(a) Provide the query plan for the query above, and answer the following questions:

  i. [**2 points**] Which join method was used – nested loop, merge, or hash?

  ii. [**1 point**] What was the estimated cost of the query? (in arbitrary units)

  iii. [**1 point**] What was the total runtime? (in ms)

(b) [**5 points**] Create an index that improves the total runtime of this query.[3]. Provide the SQL statement.

(c) Provide the new query plan with the index in place, and answer the following questions:

  i. [**2 points**] Which join method was used – nested loop, merge, or hash?

  ii. [**1 point**] What was the estimated cost of the query? (in arbitrary units)

  iii. [**1 point**] What was the total runtime? (in ms)

(d) Execute the command `SET enable_mergejoin = false;` to disable merge joins. Provide the new query plan, and answer the following questions:

  i. [**2 points**] Which join method was used – nested loop, merge, or hash?

  ii. [**1 point**] What was the estimated cost of the query? (in arbitrary units)

  iii. [**1 point**] What was the total runtime? (in ms)

(e) Execute the command `SET enable_hashjoin = false;` to disable hash joins. Provide the new query plan, and answer the following questions:

  i. [**2 points**] Which join method was used – nested loop, merge, or hash?

  ii. [**1 point**] What was the estimated cost of the query? (in arbitrary units)

  iii. [**1 point**] What was the total runtime? (in ms)

(f) Execute the command `SET enable_indexscan = false; SET enable_bitmapscan = false;` to disable index scans. Provide the new query plan, and answer the following questions:

  i. [**2 points**] Which join method was used – nested loop, merge, or hash?

  ii. [**2 points**] What was the estimated cost of the query? (Feel free to find out the actual time, but be aware that it takes a while.)

[3]Using the default PostgreSQL options.

Question 3 continues...

(g) [**0 points**] Execute these commands to re-enable the different joins (or your answers for the next questions will turn out wrong):

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

Homework 5 continues…

Question 4: Optimizing Joins........................[25 points]

On separate page, with '[course-id] [hw#] [question#] [andrew-id] [your-name]'

In this question you will optimize a query with multiple joins.

Make sure you reset the optimizer to the default settings, as per Q3-(g).

Answer the questions based on the query below:

```
SELECT c.itemid, s.browser, s.agegroup,
MAX(c.created) - MIN(c.created) as date_span
FROM sessions s, clicks c
WHERE s.sessid = c.sessid
AND (s.zipcode BETWEEN 15000 AND 16000
  OR s.agegroup = (SELECT MAX(agegroup) FROM sessions))
GROUP BY c.itemid, s.browser, s.agegroup
ORDER BY c.itemid, s.browser, s.agegroup;
```

(a) [**1 point**] Destroy any indexes created on the previous questions. Provide the SQL commands.

(b) Provide the query plan for the query above and answer the following questions:
   i. [**1 point**] What was the estimated cost of the query? (in arbitrary units)
   ii. [**1 point**] What was the total runtime? (in ms)

(c) [**20 points**] Optimize the total runtime of the query. You're allowed to (i) use any techniques discussed on this homework, and (ii) tweak the query itself, without modifying the results. The query will be ran against the **same database dump** provided. Provide a list of the optimizations you made, all the SQL statements to implement them, and the final query used. *Hint:* You should see performance gains of ~10 times on your assigned GHC machine.

(d) Provide the new query plan with your optimizations in place and answer the following questions:
   i. [**1 point**] What was the estimated cost of the query? (in arbitrary units)
   ii. [**1 point**] What was the total runtime? (in ms)