Homework 7

Due: Nov 13 in class.

The goal of this homework is to observe the behavior of a query optimizer. The related earlier homework on real world query optimizer behavior targeted the relationship between select predicates, secondary indexes and disk locality. This homework targets the join operator and the optimizer's cost function wrt the size of intermediate join results.

This is the first time this homework is being assigned. If you witness a behavior that is interesting and determine some additional queries or test cases that help reveal details, your feedback will be appreciated by me and future classes. Complementarily, if a path through this homework proves boring, don’t get stuck wondering what you were supposed to learn. In particular, the chunk on referential integrity may not have any impact on query plans. (I have no idea, yet, if certain ideas in the scientific literature have made their way into commercial practice. i.e. You are being asked to test that.)

Deliverable:

The deliverable for this homework is the physical plan determined by the query optimizer for each of the queries. This can be obtained using the SQL explain command, but graphical presentation of the plans are preferred.

For mySQL users, (and perhaps Oracle users), the native install does not provide graphical presentation of query plans. A convenient third-party tool is Aqualfolds Data Studio. Although it may be beyond the boundary of the formal eval license, I have a verbal permission for everyone in the class to download the 14 day eval copy and use it for class work.

Similar to the last implementation homework, come to class Saturday with hardcopy of your homework, 1 query plan per sheet. We’ll compare the results from the different RDBMSs.

Important:

You do not have to execute the queries. Thus, you may very well have to let your computer work ~8 hours to load the data into the test database (assuming ~1 hour to load 1,000,000 rows). But, the time it takes to ask the optimizer to produce a plan should just be moments. So, like the earlier homework, you may want to load the database with a small test example, do [some] of the homework, and then develop the final results. Further, since you are not running the queries, you need not execute many queries and average them.

Test data:

The test data will comprise the same data table replicated 8 times. Let that table be defined as

Data(pk, ht, tt, ot, filler)

Let there be 1,000,000 rows with a unique primary key.

Column ht should contain uniformly distributed values from 0 to 99,000 (i.e. hundred thousand),
Column tt, should contain uniformly distributed values from 0 to 9,000 (i.e. ten thousand),
Column 0t, should contain uniformly distributed values from 0 to 999 (i.e. one thousand),
Column filler – same as before

Note this schema is modeled after the schema in a famous database benchmark, “The Wisconsin Benchmark”,
The original paper: “Benchmarking Database Systems -- A Systematic Approach”
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.7764
Some ways better, a retrospective paper: "The Wisconsin Benchmark: Past, Present, and Future”

Physical Schema:
Create three copies of the test data table. Call them A, B and C. There should be no additional physical details beyond the basic table definition.

Create three more copies of the test data table. Call them A’, B’, C’, (or Aprime, Bprime, Cprime). Build secondary indices on each of the three columns, ht, tt, ot, for each of the three additional tables, for a total of nine indexes.

Create two more copies of the test data. Call them B”, C” (or Bdprime, Cdprime). For both B” and C”, declare the ht, tt and ot columns to be foreign key references on the primary key of table A. Build a secondary index on the ht, tt and ot columns of B only (not on C”).

Queries:
Determine physical query plans for SFW queries with the following where clauses

Section 1: How does size and index impact choice of join order and algorithm?

1. A.pk = B.pk
2. A.ht = B.ht
3. A’.ht = B’.ht

4. A.ht = B.ht = C.ht
5. A.ot = B.ot = C.ot
6. If 4 and 5 produce different plans, then
   A.tt = B.tt = C.tt, else simply report “ 4 and 5 produce the same plan”

7. A.pk = B.ht = C.ht
8. A.pk = B.ot = C.ot
9. Same as 6.

10. A’.ht = B’.ht = C’.ht
11. repeat 10 for the ot column, (i.e. 10 is to 11, as 4 is to 5)
12. repeat 7 as detailed in 6.

13. A.pk = B'.ht = C'.ht
14. Repeat 13 for the ot column, use A.pk for the “A” argument
15. Repeat 13 as detailed in 6

Section 2: Does the optimizer reorder joins? (keep the from clause the same)

16. A.ht = B.th = Cot
17. C.ot = B.th = A.ht

18. A.pk = B’.th = C’ot

Section 3: Semantic optimization test

20. A.pk = B”ht = C”ht
21. Repeat 11 for the ot column
22. Repeat 11 as detailed in 6

Important:

You will probably have to run an explicit database command for the database to examine itself and gather statistics for the optimizer. Be sure you do this after loading the database AND building the indexes, and before generating the query plans. The command is not part of the SQL standard. You will have to look up what you need to do for the vendor of the RDBMS that you are using.