EECS 484 W17 - Homework 4

Deadline -- 11:55 PM on Monday, March 27, 2017

Please read the following instructions before starting the homework. Failing to follow the submission rules will lead to -4 points (e.g., wrong file format, wrong group submission).

You are allowed to work in pairs for this homework.

Deliverables: Please submit your answers through Canvas as a single PDF file and please show your work for the problems. Your solutions can be hand written or created electronically, as long as they are clear. Please use a high quality camera or a scanner found in the library/computer labs and scan your answers into one PDF file if they are hand drawn. Correct solutions that are hard to read may lose points.

If you are working in pair, make sure to **JOIN THE SAME GROUP** on Canvas (you need to join the group every time for different homework and projects) and make only one group submission.

No late days for homework! *If you miss the due date, you get 0 points. No exceptions on this.*

By submitting this homework, you are agreeing to abide by the Honor Code:
I have neither given nor received unauthorized aid on this assignment, nor have I concealed any violations of the Honor Code.

UPDATE & CLARIFICATION

3.25

Question 4: ignore the size of pointers between leaf nodes.

3.26

Question 2: The last attribute in the given query was changed to S.age instead of S.rating

Question 1 (20 points):

In class, we developed a cost model for various relational database operations, such as joins. We counted the total number of I/Os (i.e., writing or reading a page) required to perform each operation. From there, we could estimate the relative performance of two query plans. If we were to consider a nested-loop join vs. a hash join, for example, we might guess that using the hash join would be better because it would require fewer I/Os.

In modern magnetic disks, the cost of performing a sequential read is approximately the same as that of performing a sequential write; however, you might imagine another kind of secondary storage device where this is not true. For example, it is widely accepted that when using flash memory devices, it is cheaper to read a block than it is to write a block. Therefore, you might also imagine that replacing our magnetic disk with another device would affect our cost estimates, which could in turn affect the optimizer's plan (i.e., choice of algorithms).

For this question, suppose that you are given two relations, R and S, and that you want to perform a natural join between the two. Suppose the following is also true of the two relations:

Number of records in R = 50,000
Number of pages in R = 5,000
Number of records in S = 120,000
Number of pages in S = 15,000

Given the above, answer the following questions. In all cases, please ignore the cost of *writing* the final result back to secondary storage.

1.1) (6 points) Block Nested Loops Join
Supposed that you have B = 800 buffer pages available. Choose the outer relation such that the total cost is minimized. How many (a) total reads and (b) total writes are required (ignoring the cost of writing the final result)?

1.2) (7 points) Grace Hash Join
How many (a) total reads and (b) total writes are required (ignoring the cost of writing the final result)?

1.3) (7 points) Suppose we give you a new disk, and that reading a page from this disk will take 1 µs, and writing a page to disk will take X µs. Given your above analysis, for which values of X would you choose the block nested loops join? For which values of X would you choose the hash join? Round X to the nearest hundredth (e.g., if X = 13.56666, use 13.57 as your final value for X). *Please make sure to both show your work and explain your answer for full credit.*

<u>Question 2 (20 points):</u>
Consider the following relational schema and SQL query:

Sailors (sid, sname, rating, age)
Boats (bid, bname, color)
Reserves (sid, bid, rday)

SELECT S.sname, B.bname
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid AND B.bid = R.bid AND B.color = ‘blue’ AND S.rating > 3.5 and S.age < 35

2.1) (10 points) How many different join orders does a System R style query optimizer consider when deciding how to process the given query? Assume that cross-products are disallowed. List each of these join orders.

2.2) (10 points) There are several possible indexes that would be helpful. List five attributes on which you would build indexes.

<u>Question 3 (10 points):</u>
Suppose we are using the Dell DVD Store database from project 3. Consider the following query:

```
SELECT tax
FROM Customers C, Orders O
WHERE C.customerid = O.customerid
        AND C.country = 'China'
        AND O.totalamount > 100;
```

Using Postgres and the query execution plan notation discussed in class, draw the execution plan selected by the query optimizer. Please include details such as the optimizer's choice of algorithms (e.g., file scan, hash join). Note: please run the following commands before running the above query (this must be done in the same directory where you have the dell dvd store files from project 3):

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

Also make sure that you re-enable any join algorithms you may have disabled while working on the project, such as in Exercise 4: Join Selection.

Question 4 (30 points):

Consider the following relational schema and SQL query:

Students(sid, sname, gpa)
Takes(sid, cid)
Class(cid, cname, ctype)

```
SELECT S.sname, C.cname
FROM Students S, Takes T, Class C
WHERE S.sid = T.sid AND T.cid = C.cid AND C.ctype = ‘Social’ AND S.GPA > 3.5
```

Then consider a query plan that works as follows:
Join(Join(Selection(S), T), Selection(C)) with projects and selections cascaded and pushed in as far as possible.

Suppose we use Grace Hash Join for both joins, an unclustered hash index with a search key of ctype to access C, a clustered B+ tree index with a search key of GPA to access S, and a scan to access T. Assume the following:

- The B+ tree index on GPA is of height 2 with 64 leaf nodes. Each node occupies one page, and only the leaf nodes are on disk (i.e., all of the internal nodes are stored in RAM).
- Each GPA is stored to 2 decimal places (e.g., 3.75), and there are 400 distinct values for the GPA attribute, with the lowest GPA attribute being 0.01 and the highest being 4.00. Assume that GPA is evenly distributed (i.e., they are all equally likely)
- The ctype attribute takes 50 distinct values, all equally likely.
- Each attribute (search key) has a size of 16 bytes.
- Each RID (pointer) has a size of 32 bytes.
- Our system has a page size of 4096 bytes and a memory buffer size of 12 pages. Double buffering is not used and the I/O block size is a single page.
- Each page in the B+ tree has a fill factor of 0.71.
- There are 15K (i.e., 15 * 1024) tuples in the Class relation, and 240K (i.e., 240 * 1024) tuples in the Takes relation.
- All indexes use Alternative 2, as discussed in class.

For each join, you may choose either operand as the inner one (whichever results in a lower cost!), and you should pipeline the execution plan as far as possible. Keep in mind that if one pass of hashing a relation results in partitions that are too large for available memory, you may need additional passes to hash each partition into smaller partitions.

Given the above, answer the following questions:

4.1) (6 points) How many tuples are in S?

4.2) (6 points) How many data pages does S have?

4.3) (6 points) How many data pages does T have?

4.4) (6 points) Based on the above query, what would be the cost to scan the relation S (# of reads), and then materialize (# of writes for the SELECTED tuples only) the results? Please clearly show what each cost is (e.g., # of reads = ##, # of writes = ##) along with your work.

4.5) (6 points) What would be the number of I/O operations of a grace join in (a) the partition phase and (b) the probe phase for the join(selection(S), T) part of the query plan, according to the above query? Please clearly show both costs (e.g., partition phase = ##, probe phase = ##) along with your work.

<u>Question 5 (20 points):</u>

Consider a table, Employees, where there is a B+ tree on attribute salary. Suppose we also have the following query:

SELECT * FROM Employees ORDER BY salary

In addition, we are given the following parameters:

- Height of the B+ tree = 4 (Note: height does not include the root node)
- # of index leaf pages = 400
- # of data record pages = 8,000
- # of tuples per page = 100

Given that the first level (root) of the B+ tree is always in memory, provide the cost of executing the above query using the B+ tree when:

(A) (8 points) The index is clustered

(B) (8 points) The index is unclustered

(C) (4 points) When would you consider doing an external sort instead of using the B+ tree? Justify.