**(67506) Databases – Spring 2022 – Exercise (3)**

**Muaz Abdeen 300575297**

**Mohammad Ghanayem 208653220**

***Question (1):***

authors (name, conference, year, institution, count, adjustedcount)

conferences (conference, area, subarea)

institutions (institution, region, country)

First, some primary calculations:

For there are rows per block, and for there are rows per block.

Therefore, , , , .

1. BNL cost: The outer table is the one with less blocks, that is, conferences ().
2. Hash: Notice that: , but , then we can join using hash tables. The cost is:
3. Notice that: and .

Since the sequences generated by divide table into partitions, that is , are greater than the buffer size, we can’t sort it. So, **we CAN’T execute the Sort Merge join**.

1. BNL cost: The outer table is the one with less blocks, that is, conferences ().
2. Hash: Notice that: , but , then we can join using hash tables. The cost is:
3. Notice that: and . then we can join using Sort Merge. Now, since we can’t merge the sorted sequences before merging each table sequences to one sorted table, so the cost is:
4. BNL: minimal buffer size: . Since the buffer must have at least one page for table , and one page for table , and a page for the output.
5. Hash-Join: minimal buffer size: . Since we must have at least one table such that its biggest bin size is less that the buffer size.

1. Sort-Merge: minimal buffer size: . Since we must be able to sort each table, that is, each table number of sorted sequences must be less than the buffer size.

So, for both tables can be sorted, and we can execute Sort-Merge-Join.

1. To execute Sort-Merge-Join in more efficient way, in addition to the condition in the previous section, it must hold:

So, the minimal buffer size is: .

***Question (2):***

1. Output size of block.

Since each block in contains rows, then there is  **block**.

1. Output size of blocks.

Since each block in contains rows, then there are  **blocks**.

1. Output size of rows.

\* is a key so .

1. The most efficient algorithm is:

From previous sections we conclude that pushing down the selection decreases the size of tables to be joined, so whenever we can push, we do that.

First, notice that for the expression we have:

, we can read it by full table scan only since there is no index.

, and as calculated in section (2).

Second, for the expression we have:

, by full scan, and by index scan, since it is given that the cost of accessing the index is negligible.

, and as calculated in section (1).

Now, for the **BNL** query plan the cost is:

We used as outer relation since the expression has smaller size .

We can notice that the queries **SM** and **Hash** can’t be more efficient since they require at least 3 passes over relations, whereas BNL required a single pass in our case.

For the **INL** query plan the cost is:

We used as outer relation and as inner relation since there is an index on the join attribute in the relation only (). For the cost of selection, notice that for each tuple in there is one matching tuple in , since is a key, and because the cost of accessing the index is negligible, for every tuple we just calculate the cost of getting the corresponding block from the disk (each tuple may be in a different block), so .

Finally, we conclude that **BNL** is the optimal query plan with cost .

Diagram

Description automatically generated

1. The cost of the most efficient algorithm is:

**BNL** is the optimal query plan with cost .

***Question (3):***

1. In there are rows, in there are rows, so we got that rows, and rows.

Now to find out how much rows results, we compute the following ( is a key for ):

1. In our results we get that each row is of size bytes, so in each block there are rows, so we got that the size of our result in blocks is  **blocks**.
2. To begin with, we need to compute the costs and sizes for each .

Now, we want to compute :

There is matching rows, after running projection each row needs 20 bytes, so we got blocks.

So, we have:

Let’s compute the cost of each possibility:

Two possibilities for BNL Join:

For the next possibility, Sort Join, we should check if the condition holds, so:

So, SM not working.

Now we going to check the condition for Hash Join, if it is holds or not:

And the cost is:

And from all these checks we got that the most effective way to compute the results is

**Hash Join.**

This is the query plan tree:

Diagram

Description automatically generated with medium confidence

***Question (4):***

1. The query is canceled due to timeout error.

ERROR: canceling statement due to statement timeout

The query plan according to EXPLAIN command:

Text

Description automatically generated

1. I change the query like this:

SELECT DISTINCT \*

FROM authors a1 NATURAL JOIN (SELECT year, max(adjustedcount) AS adjustedcount

FROM authors a2

GROUP BY year) D;

I think the cause is we don’t rescan the table sequentially for every tuple in the table, this what the condition WHERE does, instead my new query scans the table once to create the table D, then joins both authors and D.

The query plan is:

Text, letter

Description automatically generated

1. Text

   Description automatically generatedCreate index on authors(year, adjustedcount);

It took 2112.418 ms = 2.112418 seconds to run the query from section 1, it ran faster because of the fact that for each row in the table it uses the index to scan the wanted year and adjustedcount, and that’s absolutely faster than full table scan, and that’s it.