1. Mardel Maduro
2. Student # 3349958
3. COMP 418
4. Assignment 2

## Part 1. Concepts and principles

### **Question 1**

**Explain what a typical optimizer does, and what strategies can be used to generate a good plan.**

- A typical optimizer uses relational algebraic equivalences to queries, making multiple plans for arriving to the necessary solution and using the most efficient one. This allows for the system to make better use of resources to fulfil the query and provide the user with results faster.

The strategies used to generate a good plan are :

- Enumerating the alternative plans for evaluating the expression. This is done by considering a subset of all the possible plans since the full set of plans may be large and should be reduced to improve efficiency.

- Estimating the cost of each enumerated plan, comparing them to see which one has the lowest cost and choosing that one as the plan to use.

### **Question 2**

**Briefly explain the concept of blocked I/O and why it is cheaper to read pages using blocked I/O rather than reading them using several independent requests.**

- Blocked I/O involves the handling of I/O in blocks instead of on a per page basis. In this type of I/O handling the disk system reads and writes in buffer blocks. This is cheaper than reading using several independent request since it requires less accesses since it can access several contiguous pages as a unit (a block ) and handle them with one transfer request as opposed to multiple ones if we had to access each page separately.

### **Question 3**

**Explain why two different hash functions are used in hash join, and how to select them.**

- Two different hash functions are used in hash join in order to reduce CPU cost. The first is called hash function H which is used to partition relations which is different from the second one which is H2. H2 is used to build an in-memory hash table for partition R. H2 is used to distribute tuples in partitions. We select between them by implementing phases and having H run in phase 1 (partition phase) and H2 run in phase 2 (distribution phase).

### **Question 4**

**For each of the following join techniques, discuss the contexts for which they are most suitable and for which they are not (list their pros and cons): hash join, sort-merge join, and block nested loops join.**

-

|  |  |  |
| --- | --- | --- |
| Technique | Pros | Cons |
| Hash Join | -is faster for joining a very small table to a large one  -is good for equality joins | -doesn’t support non-equality joins |
| Sort-Merge Join | - suitable for both equality and non-equality based joins  -Leaves the results sorted | -expensive when you have to use external sorting (there isn’t enough memory of in-memory sorting) |
| Blocked Nested Loops Join | - efficiency once the relations will fit in memory and MRU replacement is being used. | -not the best if indexes are available |

### **Question 5**

**Define the concept of reduction factor, and explain how it is calculated for equality clauses using the statistics available in database catalogs (consider only the WHERE clauses: column = value and column1 = column1).**

-The reduction factor is a ratio which describes the relationship between the expected result size and the input size considering only the selection represented by the term. In situations where column=value, it is calculated using the following equations:

if (index on column)

RF = 1/Nkeys(I)

else

RF = 1/10

## Part 2. Design considerations and calculation (60 marks)

### **Question 1**

**As a database administrator, you are in charge of regularly tuning the performance of the system. After a long period of operation, you noticed that queries making use of a specific file start to slow down. This particular file used to be sorted but is not anymore. After examining the type of queries processed on the file, you decided that sorting the query would improve its speed. Given that there is a B+ tree index on the file with the same search key, you know you have two options:**

* **Retrieving the records in order through the index.**
* **Retrieving the records in random order and then sorting them.**

**Compare the estimated cost of the two approaches to figure out which one is better to use. (Note: you are not required to do an exact calculation of the cost. It’s rather a general comparison).**

**-** Given the 2 options above I would retrieve the records in order though the index. I would do this since getting the records through the index would mean the columns would already be sorted and be containing unique values. This means I wouldn’t have to go through the extra work of resorting them like I would have to do if I retrieved them at random. The estimated cost of the index approach would be constant since thanks to the index we would know where to go to retrieve the record we need and it would already be in order, where as with the random then sorting approach we would have to spend extra time sorting the random assortment of records.jj

### **Question 2**

**As a system administrator for the accounting firm AFS, you are usually asked to import data from a regular source and upload the records to a heap file that needs to be sorted after the import. You want to write a script that will do that automatically, so you need to estimate the cost of the sort operation. Your DBMS uses external sort and makes efficient use of the available buffer space when it sorts a file. Information about the newly loaded file and the DBMS software available to operate on it are as follows:**

**The number of records in the file is 4500. The sort key for the file is 4 bytes long. You can assume that rids are 8 bytes long and page ids are 4 bytes long. Each record is a total of 48 bytes long. The page size is 512 bytes. Each page has 12 bytes of control information on it. Four buffer pages are available.**

1. **How many sorted subfiles will there be after the initial pass of the sort, and how long will each subfile be?**

**-** # of records per page = 10

# of total pages = 4500 /10 = 450

# of buffer pages (pages per run) = 4

# of runs = 450/4 = 133 sorted runs (subfiles) of 4 pages except for the last one which is only 2 pages long.

1. **How many passes (including the initial pass just considered) are required to sort this file?**

- Number of Passes = logb-1N+1 --> ceiling(log4-1133) +1 = 5+1 = 6 passes

1. **What is the total I/O cost for sorting this file?**

**-** Cost = 2 \* N \* 6 = 2 \* 450 \*6 = 5,400 I/Os

### **Question 3**

**As a database administrator, if your DMBS offers the possibility of using any of the join algorithms (sort-merge join, hash join, nested loops join), which one would you use in each of the following situations:**

1. **The join condition is not equality.**

**-** I would use a sort-merge join if the join condition was non equality.

1. **The join condition is not equality and you have a B+ tree index.**

**-**I would use a nested-loop join if the join condition was not equality and I had a B+ tree index

1. **The join condition is equality.**

**-** I would use a hash join if the join condition was equality

### **Question 4**

**A programmer team in your organization is developing a web-based application that will be accessible to the customers. In order to make it efficient they need estimate the cost of different operations the system needs to do before answering a user request. One of the services to users involves joining two relatively large relations (RA and RB), so they asked you as a system administrator to provide an estimate of the cost for this join operation. You know that the DMBS supports only nested loops join algorithms. The cost metric is the number of page I/Os unless otherwise noted, and the cost of writing out the result should be uniformly ignored.**

Given the following information about the relations to be joined:

Relation RA contains 10,000 tuples and has 10 tuples per page.  
Relation RB contains 2,000 tuples and also has 10 tuples per page.  
Both relations are stored as simple heap files.  
Neither relation has any indexes built on it.  
52 buffer pages are available.

1. **What is the cost of joining RA and RB using a page-oriented simple nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?**

**-** RA contains 1000 pages RB contains 200 pages available buffer pages = 52

Total Cost = outer pages + (outer pages \* inner pages) -> (using page at a time)

With RA as outer page

= 1000 + (1000 \*200) = 201,000 I/Os

With RB as outer page

= 200 + (200 \*1000) = 200,200 I/Os (so would chose this one, marginally better is still better)

Minimum number of buffer pages needed = 3 pages (1 for each relation and 1 for output)

1. **What is the cost of joining RA and RB using a block nested loops join? What is the minimum number of buffer pages required for this cost to remain unchanged?**

**- -** RA contains 1000 pages RB contains 200 pages available buffer pages = 52

Total Cost = outer pages + (ceiling(outer pages/B-2)\*inner pages)

With RB as outer page

= 200 + ((ceiling(200/52-2)) 1000) = 200 + (4\*1000) = 4200 I/Os

Minimum number of buffer pages needed = 52 pages (if it was lower, the term (ceiling(outerpage/B-2)) would increase meaning we wouldn’t have the same cost. For example if we had 51 pages, the term would increase to 5 and the cost would then be 5200 I/Os.

### **Question 5**

For the purpose of tuning your database that has begun to slow down for some queries, you are looking for the best plan to execute the following queries on the relation described below:

The relation you are dealing with is Employee with attributes ename, title, dname, and address; all are string fields of the same length.

The ename attribute is a candidate key.  
The relation contains 10,000 pages.  
There are 10 buffer pages.

(When answering the questions, make sure to describe the plan you have in mind.)

The first query is:

SELECT E.title, E.ename FROM Employee E WHERE E.title=‘Administrator’

Assume that only 10% of Employee tuples meet the selection condition.

1. Suppose that a clustered B+ tree index on ename is (the only index) available. What is the cost of the best plan?

- Since the where part of the query doesn’t include anything to do with ename, the index on ename will not assist the running of the query. We will still have to go through all the records by name and check if title = administrator. Meaning we will have to do a file scan, which will have the cost of 10,000 I/Os since there are 10,000 pages and we would have to check all of them.

1. Suppose that a clustered B+ tree index on title is (the only index) available. What is the cost of the best plan? **(5 marks)**

- Since the where part of the query is making use of title and the B+ tree is indexed on title we get to make use of the indexing to speed up the query. The index allows for a reduction of having to read all 10,000 pages to only having to look at 1000. As for the tree traversal with 10,000 pages the tree would be of height 2, meaning to get to the first title= ‘Administrator’ will take 2 I/Os. This coupled with the cost of writing out (title,name) , which is 1000\*0.5\*0.5 = 250 and the cost to traverse the B+ tree to get to the corresponding node makes the total cost of the best plan 1252 I/Os. (0.5 because of (title,name) being half the size of the full tuple and the other 0.5 based on name having a reduction factor of the type ‘column-in’.

The second query is:

SELECT E.ename FROM Employee E WHERE E.title=‘Administrator’ AND E.dname=‘Finance’

Assume that only 10% of Employee tuples meet the condition E.title =’Administrator’, only 10% meet E.dname =’Finance’ and only 5% meet both conditions.

1. Suppose that a clustered B+ tree index on dname is (the only index) available. What is the cost of the best plan?

- Same as with question 2, we get to speed the query up by having the B+ tree being indexed by one of the columns within the where part. The B+ tree is indexed on dname and as such we get to reduce the number of pages we’re looking at from 10,000 to 1000. As for the tree traversal with 10,000 pages the tree would be of height 2, meaning to get to the first dname= ‘Finance’ will take 2 I/Os. This coupled with the cost of writing out (ename) , which is 1000\*0.25\*0.5 = 125 and the cost to traverse the B+ tree to get to the corresponding node makes the total cost of the best plan 1127 I/Os. (0.25 because of (ename) being 1/4 the size of the full tuple and the other 0.5 based on name having a reduction factor of the type ‘column-in’.

1. Suppose that a clustered B+ tree index on <dname, title, ename> is (the only index) available. What is the cost of the best plan?

- We again get to speed the query up by having the B+ tree being indexed, but this time even more since it’s indexes by both of the columns within the where part as well as the column used for the select part. This means we can perform an index-only scan. The cost for this scan would be 132 IO’s since it would 2 I/Os to access the first record that matches both and then 130 I/Os to get the required record indexes since its 10,000\*0.05\*0.25. (O.05 since 5% of the records meet both conditions and 0.25 since the query is 1/4 the size of the full tuple.)