W4111 – Introduction to Databases Sections 002, V002; spring 2022

Homework 4a – Written Assignment

Instructions

- The homework submission date/time is 2022-MAY-01 at 11:59 PM.
- Submission format is a PDF version of this document with your answers. Place your answers in the document after the questions.
- The name of your PDF must be <UNI>_S22_W4111_HW4a_Written.pdf. For example, mine would be dff9_S22_W4111_HW3a_Written.pdf
- You must use the Gradescope functions to mark the location of your questions/answers in the submitted PDF. Failure to mark pages will cause point deductions. **Please, please read the countless Ed posts, TA produced instructions and videos, etc. to prepare your submission.**
- You can use online sources but you must cite your sources. You may not cut and paste text.
- Questions typically require less than five sentences for an answer. You will lose points if your answer runs on and wanders.

  "Verbosity wastes a portion of the reader's or listener's life."

Questions

Question 1: Explain why a sparse index must also be a clustering index.

Answer:

A sparse index will only have the pointers for a few records, if it is not a clustering index then some of the records will get omitted and not be referenced. so, it must be a clustering index to refer to all the records.

Question 2: Briefly explain sparse, multi-level indexes and their benefits. Why can the outer index be sparse?

Answer: Sparse index is the index table that will contain the pointers for a few of the records sorted in some fashion, similarly multi-level index is a recursive indexing method based on the same concept. This will help save memory on storing the index tables and will speed up the process of accessing a record compared to a table without indexing. The outer index can be sparse as each index will point to the first record in the respective cluster and in that way all the records can still be accessed.

Question 3: Indexes can significantly improve performance? What are some disadvantages of having many indexes?

Answer: Partitioning an index-organized table is not allowed. Indexing will decrease the performance on insert, delete and update queries due to additional overhead. We can’t directly perform any other indexing on an indexed table. We need a primary key on the table with a unique value to index a table.

Question 4: Briefly compare the pros and cons of B^+-Tree versus a Hash Index.

Answer:

The advantage of the B+ tree is that as all the values are stored in a sequential linked list it is very efficient for queries that involve ranges. Since it is a balanced tree structure the insert/delete/update does not affect the tree performance. Disadvantages being an extra insertion and deletion overheads and memory overhead.

Hash indexes allow for very fast access and generally provide O(1) performance and the disadvantage is that as the data grows there is a higher chance of hash collision and a leads to overhead for resolution.

Question 5: Briefly explain the concepts of covering index/covered query.

Answer:

If all the columns in the select query are available in the index without further looking up then it is called a covering index, this would significantly improve the querying performance. As the covering index has all the required values it is covering the query, so the query is called a covered query.

Question 6: Briefly explain the three main steps/stages in query processing.

Answer:

1. Parsing and translation: Here the query is translated to its internal form which is then translated to relational algebra and in this step parser checks for syntaxes, verifies names, relations etc
2. Optimization: We know that there can be many equivalent expressions for the same query, in the optimization, this along with the optimized algorithm for executing a relational operator is selected, this is annotated as the evaluation plan
3. Evaluation: The query execution engine executes the selected evaluation plan and returns the result.

Question 8: Explain the role of equivalent queries in query optimization.

Answer:

The equivalent query is one of the main steps in query optimization where a list of equivalence rules is followed to reduce the expression to its optimal expression. The most optimal expression is reached when minimality is achieved which is when the expression cannot be further reduced using an equivalence rule. It plays a significant role in query optimization as it has a high potential in reducing the time required in solving a complexly written query that has a simper equivalent.

Question 9: Assume that the ON clause in a JOIN on tables A and B compares columns a1 with b1 and column a2 with b2. What two properties must the ON condition have for the optimizer to be able to use a Hash Join?

Answer:

The property that is important for hash join is that the smaller size output of a1 comparison with b1 or the a2 comparison with b2 should be able to be fit in the memory and the second property is that the outputs of the two comparison subsets should be unique for the creation of a hash function.

Question 10: What is index selectivity? How does it factor into query optimization for JOINs?

Answer:

Index selectivity is the measure of the effectiveness of an index to select required record, so generally if an index returns less number of records then it’s highly selective. So, during query optimization having a column that is used for join as indexed will significantly improve the performance of that join. Having a highly selective index improves the performance of join hence if required a composite highly selective index should be created to optimize the performance of join queries.