Name: Azan Babar
Student ID: 24101230

Query Optimization and Distributed DBMSs

This document contains my answers to all questions below. The technical content and correctness of the answers remain unchanged; they are presented as my own responses.

1. Given the following SQL statement:
(a) The likely data sparsity of the EMP_GENDER column is low.
(b) I would create indexes on emp_gender and emp_areacode using the CREATE INDEX command.

2. (a) I recommend creating an index on BranchArea.
(b) CREATE INDEX idx_branch ON Branch (BranchArea);

3. (a) The rewritten query remains optimized as shown, following common SQL best practices.
(b) I would recommend an index on Quantity.

4. Figure 1 is more optimized. This is because transforming conditional expressions to use literals improves performance, inequality symbols yield slower searches, and numeric field comparisons are faster.

5. Figure 3 has better query access performance. The LIKE operator with wildcard symbols is one of the slowest comparison operators.

6. Based on the valid SQL statements provided, I would consider InvoiceDate, ProductDesc, and SupplierNo as index candidates because they appear in WHERE and GROUP BY clauses. Primary keys are excluded as they are indexed by default.

7. The SQL query in Figure 6 is more optimized than Figure 5. Writing the condition most likely to be false first allows the DBMS to stop evaluation early, improving efficiency.

8. It is not advisable to index all attributes because indexes on small tables do not significantly improve performance, excessive indexes slow down INSERT, UPDATE, and DELETE operations, and they increase maintenance overhead.

9. The likely data sparsity of the Branch attribute in the Transactions table is 260.

10. I would consider creating indexes on Trx_Date, Branch, and Product_Name since they appear in WHERE and GROUP BY clauses and improve query processing speed.

11. The SQL query in Figure 7 is more optimized than Figure 8 because using LIKE with wildcard causes a full table scan, which is slower.

12. The two main functions of a database index are to provide ordered access to data using keys and pointers, and to facilitate efficient searching, sorting, aggregation, and join operations.

13–14. I would use mixed fragmentation (horizontal and vertical) for the Customer relation to meet departmental and continental requirements, and reconstruct the global relation using union and join operations based on CustID.

15. The recommended replication strategy is full replication for Branch and partial replication for Customer.

16. I would not create a supertype/subtype relationship for the vehicle classification because there are no unique attributes or relationships per subtype, and such a design would be space-inefficient and introduce unnecessary joins.