Long Questions

1. Explain multi-valued dependencies in a relational database and how they can be handled through decomposition.
2. Discuss the concept of the Fourth Normal Form (4NF) in database normalization.
3. Define the Fifth Normal Form (5NF) in database normalization and explain when it is applicable.
4. How can SQL queries be optimized for better performance?
5. Discuss the advantages of using stored procedures in SQL databases.
6. Define referential integrity in SQL databases and its importance.
7. What are common SQL constraints, and how are they used in database design?
8. How do you create a UNIQUE constraint in SQL, and what does it enforce?
9. Explain the purpose of a FOREIGN KEY constraint in SQL and provide an example.
10. Discuss the role of indexing in SQL databases and how it can improve query performance.
11. Assume you have a table named "Orders" with columns: OrderID (int), CustomerID (int), OrderDate (date), and TotalAmount (decimal). Write an SQL query to calculate the average total amount of all orders and the count of orders placed by each customer. Include customers who haven't placed any orders, treating their count as 0.
12. "You have a table named ""Employees"" with columns: EmployeeID (int, primary key), FirstName (text), LastName (text), and Salary (decimal). Implement the following:

    a. Write an SQL statement to add a constraint to ensure that the ""Salary"" column is always greater than or equal to 30000.

b. Create an SQL trigger that fires before an update operation on the ""Salary"" column to prevent salary reductions below 30000."

13. "Given two tables, ""A"" and ""B,"" with identical structures (columns: ID and Name), write SQL queries for the following:

    a. Retrieve all distinct records that appear in either ""A"" or ""B"" using UNION.
    b. Retrieve all distinct records that appear in both ""A"" and ""B"" using INTERSECT.
    c. Retrieve all distinct records from ""A"" that do not appear in ""B"" using EXCEPT."

14. Assume you have two tables, "Customers" and "Orders." Write an SQL query to retrieve the names of customers who have placed orders with a total amount greater than the average total amount of all orders.
15. Given a table "StudentCourses" with columns: StudentID (int), CourseID (int), and CourseName (text), where StudentID is the primary key and CourseID determines the CourseName, normalize the table to the Third Normal Form (3NF).
16. What is the significance of a transaction in a database management system?
17. Explain the ACID properties of a transaction.
18. What are the different states a transaction can go through during its execution?
19. Why is the transaction log important in a database system, and what information does it record?
20. Define the concept of a savepoint within a transaction.
21. What is concurrency control, and why is it necessary in a database system?
22. Explain the challenges posed by concurrent execution of transactions in a database system.

23. What is serializability in the context of concurrent transactions, and why is it important?
24. Describe the concept of recoverability in concurrent transactions.
25. Explain the implementation of isolation levels in a database system and their significance.
26. What are lock-based protocols in concurrency control, and how do they work?
27. Describe two-phase locking and its role in maintaining serializability.
28. Explain the concept of deadlock handling strategies in a lock-based concurrency control system.
29. What is shared vs. exclusive locking, and how do they affect concurrency?
30. Explain the concept of lock granularity in lock-based protocols.
31. What are timestamp-based protocols in concurrency control, and how do they work?
32. Explain the concepts of read and write timestamps in timestamp-based protocols.
33. Describe the two main timestamp-based protocols: Thomas's Write Rule and the Strict Two-Phase Locking Protocol.
34. What is the significance of validation-based protocols in concurrency control?
35. Explain the concept of multiple granularity locking in database systems.
36. What are the key considerations in database recovery and maintaining atomicity?
37. Describe log-based recovery in database systems and its role in ensuring durability.
38. How do databases recover with concurrent transactions after a system failure?
39. What are the challenges and solutions related to maintaining data consistency in a distributed database system?

40. Explain the concept of distributed recovery and the techniques used to ensure data consistency in a distributed database system.
41. You have a database with a table named "Orders" containing columns: OrderID (int), CustomerName (text), and OrderDate (date). Write an SQL transaction that inserts a new order with the provided data into the "Orders" table while ensuring atomicity and durability. If the transaction fails, it should not leave any partial changes.
42. "Assume you have a table named ""Inventory"" with columns: ProductID (int), ProductName (text), and Quantity (int). Implement a lock-based protocol to ensure mutual exclusion when two transactions try to update the quantity of a product simultaneously. Write SQL code for the lock-based protocol.
43. "Implement a timestamp-based protocol in SQL for managing concurrent transactions. Write SQL code to assign timestamps to transactions and ensure serializability by allowing transactions with higher timestamps to proceed.
44. Implement a log-based recovery mechanism in SQL for a database that includes a "TransactionLog" table to record transactions. Write SQL code to simulate the recovery process, rolling back uncommitted transactions after a crash.
45. "You have a database with two tables: ""Employees"" and ""Projects."" The ""Employees"" table contains columns: EmployeeID (int, primary key), EmployeeName (text), and Salary (decimal). The ""Projects"" table contains columns: ProjectID (int, primary key), ProjectName (text), and EmployeeID (int, foreign key). Implement a multiple granularity locking mechanism in SQL to ensure proper locking of both individual rows in the ""Employees"" table and the entire ""Projects"" table while allowing concurrentaccess. Write SQL code for transactions that demonstrate this multiple granularity locking mechanism. One transaction should read

employee information and the other should update the project's employee assignment.

46. What is the significance of external storage in database management?
47. Explain the concept of file organization in databases and its importance.
48. How do cluster indexes differ from primary and secondary indexes, and when are they used?
49. Describe common index data structures in databases and their roles.
50. What is hash-based indexing, and in which scenarios is it suitable for indexing data?
51. Explain tree-based indexing and its variations, such as B-trees and B+ trees.
52. Compare various file organizations (e.g., heap files, sorted files, clustered files) regarding their advantages and disadvantages.
53. What are primary and secondary indexes, and how do they enhance data access efficiency in databases?
54. How do database indexes contribute to performance tuning, and what factors should be considered when designing them?
55. Provide insights into understanding tree-based indexes, such as B-trees and B+ trees.
56. What are Indexed Sequential Access Methods (ISAM), and when are they employed in database systems?
57. Explain the characteristics and advantages of B+ trees as a dynamic index structure in databases.
58. What is the role of external storage in a database system, and why is it essential?
59. Describe the concept of file organization in a database system and its impact on data management.
60. How do cluster indexes differ from primary and secondary indexes, and in what scenarios are they beneficial?

61. Explain the role of index data structures in a database system and provide examples of common index structures.
62. What is hash-based indexing, and when should it be considered as an indexing method?
63. Describe tree-based indexing and provide insights into the differences between B-trees and B+ trees.
64. Compare and contrast different file organizations, index types, and their impact on database performance tuning.
65. How do primary and secondary indexes contribute to enhancing data access efficiency in a database system?
66. In what ways do database indexes play a role in performance tuning, and what considerations are important when designing indexes?
67. Provide intuitive explanations for understanding tree-based index structures like B-trees and B+ trees.
68. What are Indexed Sequential Access Methods (ISAM), and when are they used in database systems?
69. Explain the characteristics and advantages of B+ trees as dynamic index structures in databases.
70. Describe the significance of external storage in a database system and its role in ensuring data durability and availability.
71. Write a class in your preferred programming language that implements a hash table data structure. Include methods for insertion, deletion, and searching using hash-based indexing.
72. Write functions to insert, delete, and search for elements in a binary search tree. Demonstrate how these operations maintain the tree's properties.
73. Develop a class or module to implement a B+ tree data structure. Include methods to insert, delete, and search for elements. Ensure the tree maintains balance and handles splitting and merging nodes correctly.

74. Create a program that simulates an ISAM file organization. Implement methods for inserting records, searching for records based on a key, and performing range queries efficiently.
75. Write a program that reads data from an external file and builds indexes (e.g., hash indexes or tree-based indexes) for quick searching. Implement functions to search for records based on different criteria and measure the performance of your indexing approach.