**Part 4**

**Basic of Database Management System**

|  |  |
| --- | --- |
| **4.1** | **Overview of DBMS** |
|  | A Database Management System (DBMS) is a software suite that facilitates the creation, organization, retrieval, and management of data in a database. The primary purpose of a DBMS is to provide an efficient and structured mechanism for data storage and retrieval while ensuring data integrity, security, and concurrency control.  A database is a structured collection of data organized for efficient retrieval, storage, and management.  Databases can be categorized into different types, such as relational databases, NoSQL databases, object-oriented databases, and more.  RDBMS is a type of DBMS that manages data in a tabular form with rows and columns.  It enforces the relational model, which defines relationships between tables using keys. |
| **4.2** | **MySQL** |
|  | MySQL is an open-source relational database management system (RDBMS) that is widely used for managing and organizing large sets of data. It is part of the LAMP (Linux, Apache, MySQL, PHP/Python/Perl) and MEAN (MongoDB, Express.js, AngularJS, Node.js) stack, making it a popular choice for web development.  MySQL is distributed under an open-source license, making it freely available for use, modification, and distribution.  MySQL follows the relational database model, organizing data into tables with rows and columns.  It supports SQL (Structured Query Language) for defining, manipulating, and querying the data.  MySQL is designed to run on various operating systems, including Linux, Windows, and macOS.  MySQL supports multiple concurrent users and provides mechanisms for managing access control and ensuring data integrity.  MySQL supports a wide range of data types, including numeric, string, date and time, and spatial data types. |
| **4.3** | **Overview of Workbench** |
|  | MySQL Workbench is a visual database design and modeling tool, as well as an integrated development environment (IDE) for MySQL. It provides a graphical user interface (GUI) that simplifies the tasks of designing, modeling, and managing MySQL databases.  MySQL Workbench allows users to visually design, model, and create databases using a diagrammatic representation of tables, relationships, and entities.  Users can define tables, columns, indexes, and relationships using a drag-and-drop interface.  The IDE includes a SQL editor for writing and executing SQL queries, scripts, and stored procedures.  Syntax highlighting, auto-completion, and error checking features assist developers in writing efficient and accurate SQL code. |
| **4.4** | **Database Design** |
|  | 1. Define the Purpose and Requirements:  Clearly understand the purpose of the database and gather requirements from stakeholders.  2.Entity-Relationship Diagram (ERD):  Create an Entity-Relationship Diagram to visually represent the entities (objects or concepts) and the relationships between them.  3. Normalization:  Apply normalization techniques to minimize data redundancy and dependency.  4. Data Types and Constraints:  Choose appropriate data types for each attribute (e.g., integer, varchar, date).  Apply constraints such as primary keys, foreign keys, unique constraints, and check constraints to enforce data integrity.  5. Table Design:  Create tables based on the entities identified in the ERD.  Assign primary keys to uniquely identify each record in a table.  Establish relationships between tables using foreign keys.  6. Indexing:  Identify columns that will be frequently used in search conditions and create indexes on those columns to improve query performance.  7. Views and Stored Procedures:  Use views to create virtual tables that simplify complex queries.  Implement stored procedures for frequently used operations to centralize business logic.  8. Security:  Implement security measures such as user authentication and authorization.  9.Backup and Recovery:  Implement a robust backup and recovery strategy to protect against data loss. |
|  | **Normalization** |
|  | 1. First Normal Form (1NF):  Eliminate duplicate columns from the same table.  Create a separate table for each set of related data, and assign a primary key to each table.  2. Second Normal Form (2NF):  Meet the requirements of 1NF.  Remove partial dependencies by putting any columns that depend on only part of a multi-column primary key into a separate table.  3. Third Normal Form (3NF):  Meet the requirements of 2NF.  Remove transitive dependencies by putting columns that are not dependent on the primary key into a separate table.  4. Boyce-Codd Normal Form (BCNF):  Meet the requirements of 3NF.  For each non-trivial functional dependency, the determinant must be a superkey.  5. Fourth Normal Form (4NF):  Meet the requirements of BCNF.  Address multi-valued dependencies by creating a separate table for the multi-valued attributes.  6. Fifth Normal Form (5NF):  Meet the requirements of 4NF.  Handle cases where multiple overlapping candidate keys cause redundancy. |
|  | **Keys** |
|  | 1. Primary Key (PK):  A primary key uniquely identifies each record in a table.  It must contain unique values and cannot have NULL values.  Each table can have only one primary key.  2. Foreign Key (FK):  A foreign key is a field in a table that refers to the primary key of another table.  It establishes a relationship between the two tables.  Foreign keys are used to maintain referential integrity between related tables.  3. Super Key:  A super key is a set of one or more attributes (columns) that, taken collectively, can uniquely identify a record.  It is a superset of a candidate key.  4. Candidate Key:  A candidate key is a minimal set of attributes that can uniquely identify a record.  In a table, there can be multiple candidate keys.  The primary key is chosen from the candidate keys.  5. Composite Key:  A composite key is a primary key composed of multiple columns.  Unlike a single-column primary key, a composite key uses a combination of two or more columns to uniquely identify a record.  6. Unique Key:  A unique key is a set of one or more columns that must contain unique values.  Unlike a primary key, it can allow NULL values. |
|  | **Group by & Having Clause** |
|  | the GROUP BY clause is used to group rows that have the same values in specified columns into summary rows, like "total sales per region" or "average salary per department."  SELECT column1, column2, ..., aggregate\_function(column)  FROM your\_table  GROUP BY column1, column2, ...;  The HAVING clause is used in conjunction with GROUP BY to filter the results based on a specified condition.  SELECT column1, column2, ..., aggregate\_function(column)  FROM your\_table  GROUP BY column1, column2, ...  HAVING condition; |
| **4.5** | **SQL Basic** |
|  | SQL, or Structured Query Language, is a programming language designed for managing and manipulating relational databases. It is widely used for tasks such as querying data, updating data, inserting data, and creating and modifying database structures  1. SELECT Statement  2. INSERT Statement  3. UPDATE Statement  4. DELETE Statement  5. CREATE TABLE Statement  6. ALTER TABLE Statement  7. DROP TABLE Statement  8. WHERE Clause  9. ORDER BY Clause  10. GROUP BY Clause  11. JOIN Clause  12. Aggregate Functions  13. Constraints  14. Indexing |
| **4.6** | **Data Sorting** |
|  | Sorting data in a database environment, such as MySQL Workbench, is typically done using the ORDER BY clause in SQL. The ORDER BY clause is used to sort the result set of a query based on columns.  -- Sorting in ascending order based on a column  SELECT column1, column2  FROM your\_table  ORDER BY column1;  -- Sorting in descending order based on a column  SELECT column1, column2  FROM your\_table  ORDER BY column1 DESC; |
| **4.7** | **Null Value & keyword** |
|  | In programming and database systems, "null" is a special marker used to indicate that a data value does not exist in the database. It represents the absence of a value or a blank value.  NULL:  Represents the absence of a value in a database column.  It is not the same as an empty string or zero.  It is often used to indicate missing or undefined data.  INSERT INTO your\_table (column\_name) VALUES (NULL);  Empty String (''):  Represents a string with zero length.  It is a valid value for a character or text column.  It is not the same as NULL and is considered a distinct value.  INSERT INTO your\_table (text\_column) VALUES ('');  **Keywords:**  In the context of programming and databases, a "keyword" is a reserved word that has a specific meaning and functionality within a particular language or system. Keywords are part of the syntax and structure of the language, and they cannot be used as identifiers (such as variable names or function names). Each keyword typically serves a specific purpose and is used to convey instructions or information to the interpreter or compiler.  In SQL (Structured Query Language), keywords are used to construct queries and interact with databases. Examples include SELECT, FROM, WHERE, JOIN, INSERT, UPDATE, DELETE, etc. |
| **4.8** | **Auto Increment** |
|  | "Auto-increment" is a feature in database systems that automatically generates unique values for a column in a table. This is commonly used for primary key columns, ensuring that each row in the table has a unique identifier.  CREATE TABLE your\_table (  id INT AUTO\_INCREMENT PRIMARY KEY,  name VARCHAR(255),  age INT  ); |
| **4.9** | **DDL, DML, DCL, TCL, DQL** |
|  | In database management and SQL (Structured Query Language), there are different types of commands categorized based on their functionality. These categories are commonly known as DDL, DML, DCL, TCL, and DQL:  DDL (Data Definition Language):  DDL deals with the structure and definition of the database schema. It includes commands for creating, altering, and deleting database objects.  Common DDL commands: CREATE, ALTER, DROP, TRUNCATE, RENAME, COMMENT, etc.  DML (Data Manipulation Language):  DML deals with the manipulation and retrieval of data stored in the database. It includes commands for inserting, updating, and deleting data.  Common DML commands: SELECT, INSERT, UPDATE, DELETE, etc.  DCL (Data Control Language):  DCL deals with the control and management of access to the data stored in the database. It includes commands for granting or revoking permissions and privileges.  Common DCL commands: GRANT, REVOKE, etc.  TCL (Transaction Control Language):  TCL deals with the management of transactions within a database. It includes commands for starting, committing, or rolling back transactions.  Common TCL commands: COMMIT, ROLLBACK, SAVEPOINT, etc.  DQL (Data Query Language):  DQL is not a formal category like the others, but it is commonly used to refer to the subset of SQL that deals specifically with querying data. The primary DQL command is SELECT. |
| **4.10** | **Limit** |
|  | the LIMIT clause is used in conjunction with the SELECT statement to restrict the number of rows returned by a query. This is useful when you only want to retrieve a specific number of rows from a result set.  SELECT column1, column2, ...  FROM your\_table  LIMIT number\_of\_rows; |
| **4.11** | **Aggregate function** |
|  | Aggregate functions in databases are used to perform a calculation on a set of values and return a single value.  COUNT:   * Counts the number of rows in a result set.   SUM:   * Calculates the sum of values in a numeric column.   AVG:   * Calculates the average (mean) of values in a numeric column.   MIN:   * Finds the minimum value in a column.   MAX:   * Finds the maximum value in a column. |
| **4.12** | **Subquery** |
|  | A subquery, also known as a nested query or inner query, is a query nested within another query in SQL. The result of the subquery is used by the outer query as a part of its condition or calculation.  SELECT column1, column2  FROM table1  WHERE column1 = (SELECT MAX(column1) FROM table1); |
| **4.13** | **Joins** |
|  | a join is a clause used to combine rows from two or more tables based on a related column between them. Joins are fundamental in relational databases for querying data across multiple tables.  1. INNER JOIN:  An INNER JOIN returns only the rows that have matching values in both tables.  2. LEFT JOIN (or LEFT OUTER JOIN):  A LEFT JOIN returns all rows from the left table (the first table mentioned) and the matched rows from the right table.  3. RIGHT JOIN (or RIGHT OUTER JOIN):  A RIGHT JOIN returns all rows from the right table and the matched rows from the left table.  4. FULL JOIN (or FULL OUTER JOIN):  A FULL JOIN returns all rows when there is a match in either the left or right table. Rows without a match in one of the tables will contain NULL values for columns from the table without a match.  5. CROSS JOIN:  A CROSS JOIN returns the Cartesian product of two tables, i.e., all possible combinations of rows from both tables. |
| **4.14** | **Unions** |
|  | **UNION:**  UNION operator is used to combine the result sets of two or more SELECT statements. It removes duplicate rows from the combined result set. Each SELECT statement within the UNION must have the same number of columns in the result sets with similar data types.  SELECT column1, column2, ...  FROM table1  UNION  SELECT column1, column2, ...  FROM table2;  **UNION ALL:**  If you want to include duplicate rows in the result set, you can use UNION ALL instead of UNION. UNION ALL does not remove duplicate rows and is generally faster than UNION. |
| **4.15** | **Index** |
|  | an index is a data structure that improves the speed of data retrieval operations on a database table. Indexes are created on one or more columns of a table, and they provide a quick and efficient way to look up data based on the values in those columns.  CREATE INDEX index\_name ON table\_name (column1, column2, ...);  You can remove an index using the DROP INDEX statement.  DROP INDEX index\_name ON table\_name; |
| **4.16** | **View** |
|  | a view is a virtual table derived from one or more base tables. Unlike a physical table, a view does not store the data itself but is a saved SQL query that produces a result set when queried.  Views are created using the CREATE VIEW statement in SQL. The view definition consists of a SELECT statement that specifies the columns and rows to include in the view.  CREATE VIEW employee\_view AS  SELECT employee\_id, first\_name, last\_name, department\_id  FROM employees  WHERE department\_id = 1;  You can remove a view using the DROP VIEW statement.  DROP VIEW employee\_view; |