**DATABASE MANGEMENT SYSTEM**

A Database Management System (DBMS) is a software suite designed to manage and manipulate data efficiently and securely. It provides an interface for users to interact with the database and facilitates various operations such as storing, retrieving, updating, and deleting data.

**Components of DBMS:**

1. **Data Definition Language (DDL):** DDL is used to define the structure and schema of the database. It includes commands like CREATE, ALTER, and DROP to create and modify database objects such as tables, indexes, and views.
2. **Data Manipulation Language (DML):** DML is used to manipulate the data stored in the database. It includes commands like SELECT, INSERT, UPDATE, and DELETE to retrieve, add, modify, and delete data.
3. **Data Query Language (DQL):** DQL is a subset of DML used to retrieve data from the database. The most commonly used command in DQL is SELECT, which allows users to retrieve data based on specified criteria.
4. **Data Control Language (DCL):** DCL is used to control access to the database. It includes commands like GRANT and REVOKE to grant or revoke privileges such as SELECT, INSERT, UPDATE, and DELETE to users and roles.

**Types of DBMS:**

1. **Relational DBMS (RDBMS):** RDBMS stores data in a tabular format, where data is organized into rows and columns. It uses structured query language (SQL) for database operations. Examples include MySQL, Oracle, and PostgreSQL.
2. **NoSQL DBMS:** NoSQL (Not Only SQL) databases are designed to handle large volumes of unstructured and semi-structured data. They offer flexible schemas and horizontal scalability. Examples include MongoDB, Cassandra, and Redis.
3. **Object-oriented DBMS (OODBMS):** OODBMS stores data in the form of objects, which encapsulate data and behavior. It supports inheritance, encapsulation, and polymorphism. Examples include db4o and ObjectDB.
4. **Graph DBMS:** Graph DBMS stores data in the form of nodes and edges, representing entities and relationships between them. It is suitable for data with complex relationships, such as social networks and recommendation systems. Examples include Neo4j and Amazon Neptune.

**Advantages of DBMS:**

1. **Data Integrity:** DBMS ensures data integrity by enforcing constraints and relationships between data entities.
2. **Data Security:** DBMS provides access control mechanisms to restrict unauthorized access to sensitive data.
3. **Data Consistency:** DBMS maintains data consistency by enforcing ACID (Atomicity, Consistency, Isolation, Durability) properties during transactions.
4. **Data Backup and Recovery:** DBMS allows for easy backup and recovery of data, minimizing the risk of data loss.

**Disadvantages of DBMS:**

1. **Complexity:** Implementing and managing a DBMS can be complex and require specialized skills.
2. **Cost:** Licensing fees and infrastructure costs associated with DBMS can be high, especially for enterprise-level systems.
3. **Performance Overhead:** DBMS imposes overhead in terms of resource consumption and processing time.
4. **Vendor Lock-in:** Switching between DBMS vendors can be challenging due to differences in data formats and query languages.

**RELATIONAL DATABASE MANAGEMENT SYSTEM**

Relational Database Management Systems (RDBMS) are a type of database management system that organizes data into tables with rows and columns, where each row represents a record and each column represents an attribute or field of the record. Here are some key characteristics and components of RDBMS:

**1. Tabular Structure:** RDBMS stores data in tables, also known as relations, where each table consists of rows (tuples) and columns (attributes). The columns have predefined data types to enforce data integrity.

**2. SQL (Structured Query Language):** RDBMS uses SQL for defining, manipulating, and querying data. SQL provides a standard interface for interacting with the database, including commands for creating, querying, updating, and deleting data.

**3. Data Integrity:** RDBMS enforces data integrity through various constraints such as primary keys, foreign keys, unique constraints, and check constraints. These constraints ensure that the data stored in the database remains accurate and consistent.

**4. Relationships:** RDBMS allows the establishment of relationships between tables using foreign keys. These relationships enable the retrieval of related data through joins, which combine data from multiple tables based on common attributes.

**5. ACID Properties:** RDBMS ensures data consistency and reliability by adhering to the ACID properties of transactions:

* Atomicity: Transactions are either completed successfully (commit) or completely undone (rollback) in case of failure.
* Consistency: Transactions bring the database from one consistent state to another consistent state.
* Isolation: Transactions are executed in isolation from each other to prevent interference.
* Durability: Once a transaction is committed, its effects are permanent and survive system failures.

**6. Concurrency Control:** RDBMS employs concurrency control mechanisms to manage simultaneous access to data by multiple users or processes. Techniques like locking and multi-version concurrency control (MVCC) ensure data consistency and prevent conflicts.

**7. Examples of RDBMS:** Some popular RDBMS include:

* MySQL: An open-source RDBMS commonly used for web applications and small to medium-sized databases.
* PostgreSQL: An open-source RDBMS known for its advanced features, extensibility, and support for ACID transactions.
* Oracle Database: A commercial RDBMS widely used in enterprise environments, offering scalability, reliability, and comprehensive features.
* Microsoft SQL Server: A commercial RDBMS developed by Microsoft, commonly used in Windows-based environments for various applications.

RDBMSs provide a structured and efficient way to organize, manage, and retrieve data, making them suitable for a wide range of applications, from simple data storage to complex enterprise systems.

**Difference between dbms and rdbms**

**Database Management System (DBMS):**

1. **Definition:** DBMS is a software system that enables users to define, create, maintain, and control access to databases.
2. **Data Model:** DBMS can support various data models, including hierarchical, network, and relational models.
3. **Data Structure:** Data in DBMS may not necessarily be organized in a tabular format. It can be structured or unstructured.
4. **Query Language:** DBMS may have its own query language, but it's not necessarily based on the relational model. Examples include IMS, dBase, and Microsoft Access.
5. **Scalability:** DBMS may lack scalability features typically associated with relational databases.
6. **Data Integrity:** While DBMS ensures basic data integrity, it might not enforce all relational constraints like referential integrity.

**Relational Database Management System (RDBMS):**

1. **Definition:** RDBMS is a specific type of DBMS that manages data organized into tables with rows and columns, based on the relational model.
2. **Data Model:** RDBMS strictly adheres to the relational model, where data is stored in tables with predefined relationships between them.
3. **Data Structure:** Data in RDBMS is organized in a tabular format, with tables representing entities and attributes representing columns.
4. **Query Language:** RDBMS uses SQL (Structured Query Language) as its standard language for querying and managing the database.
5. **Scalability:** RDBMS typically offers horizontal scalability through features like sharding and replication, allowing for efficient handling of large datasets.
6. **Data Integrity:** RDBMS enforces relational integrity constraints, such as primary key constraints, foreign key constraints, and unique constraints, ensuring data consistency and accuracy.

In summary, while both DBMS and RDBMS are software systems designed to manage databases, RDBMS is a specific type of DBMS that strictly adheres to the relational model, organizes data in tables, and uses SQL as its query language.

**Normalization**

Normalization is a database design technique used to organize tables and minimize redundancy within a relational database. It involves breaking down a table into smaller, more manageable tables and defining relationships between them to ensure data integrity and reduce data duplication. Normalization is typically carried out in multiple stages, known as normal forms, each addressing different types of data redundancies.

Here are the common normal forms:

**First Normal Form (1NF):**

* Eliminates repeating groups within a table by ensuring that each column contains atomic values (i.e., values cannot be further divided).
* Each row should have a unique identifier (primary key).
* Example: Splitting a single table containing customer information and orders into two separate tables, with a unique customer identifier linking them.

**Second Normal Form (2NF):**

* Meets the requirements of 1NF.
* Ensures that all non-key attributes are fully functionally dependent on the primary key.
* In other words, each non-key attribute should be dependent on the entire primary key, not just part of it.
* Example: If a table has a composite primary key (i.e., consisting of multiple columns), and some attributes depend only on part of that composite key, those attributes should be moved to a separate table.

**Third Normal Form (3NF):**

* Meets the requirements of 2NF.
* Eliminates transitive dependencies, ensuring that non-key attributes are not dependent on other non-key attributes.
* In other words, if an attribute is not part of the primary key, it should not be dependent on another non-key attribute.
* Example: If a table contains information about employees and their departments, where the department manager's phone number is stored, it should be moved to a separate table that contains only department-related information.

**Boyce-Codd Normal Form (BCNF):**

* A stricter version of 3NF.
* Every determinant must be a candidate key (a unique identifier) in the table.
* Ensures that there are no non-trivial dependencies between candidate keys.
* Example: If a table has multiple candidate keys, each non-trivial functional dependency between them should be identified and addressed.

**Fourth Normal Form (4NF) and Fifth Normal Form (5NF):**

* These normal forms address more complex data dependencies and are less commonly encountered in practice.
* 4NF deals with multivalued dependencies, and 5NF addresses join dependencies.

By applying normalization techniques, databases can achieve better organization, reduced redundancy, improved data integrity, and increased flexibility in data manipulation and querying. However, it's important to balance normalization with performance considerations, as overly normalized databases can sometimes lead to complex queries and performance overhead.

Top of Form

What is SQL?

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

**What Can SQL do?**

* SQL can execute queries against a database
* SQL can retrieve data from a database
* SQL can insert records in a database
* SQL can update records in a database
* SQL can delete records from a database
* SQL can create new databases
* SQL can create new tables in a database
* SQL can create stored procedures in a database
* SQL can create views in a database
* SQL can set permissions on tables, procedures, and views

**CREATE DATABASE**

CREATE DATABASE databasename;

EX:CREATE DATABASE demo;

**SQL DROP DATABASE Statement**

The DROP DATABASE statement is used to drop an existing SQL database.

DROP DATABASE databasename;

Eg:DROP DATABASE demo;

**SQL CREATE TABLE Statement**

The CREATE TABLE statement is used to create a new table in a database.

Syntax

CREATE TABLE table\_name ( column1 datatype, column2 datatype, column3 datatype, .... );

**SQL DROP TABLE Statement**

The DROP TABLE statement is used to drop an existing table in a database.

DROP TABLE table\_name;

Example

DROP TABLE Shippers;

**SQL TRUNCATE TABLE**

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

TRUNCATE TABLE table\_name;

**SQL ALTER TABLE Statement**

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

ALTER TABLE table\_name ADD column\_name datatype;

**SQL Constraints**

SQL NOT NULL Constraint

By default, a column can hold NULL values.

The NOT NULL constraint enforces a column to NOT accept NULL values.

Example

CREATE TABLE Persons ( ID int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255) NOT NULL, Age int );

**SQL PRIMARY KEY Constraint**

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

**SQL FOREIGN KEY Constraint**

A FOREIGN KEY is a key used to link two tables together.

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

**The SQL INSERT INTO Statement**

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO table\_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

**SQL SELECT Statement**

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

SELECT Syntax

SELECT column1, column2, ... FROM table\_name;

SELECT \* FROM table\_name;

**SQL SELECT DISTINCT Statement**

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax

SELECT DISTINCT column1, column2, ... FROM table\_name;

**SQL WHERE Clause**

WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

Example

SELECT \* FROM Customers WHERE Country='Mexico';

**SQL AND, OR and NOT Operators**

The WHERE clause can be combined with AND, OR, and NOT operators.

• The AND and OR operators are used to filter records based on more than one condition:

• The AND operator displays a record if all the conditions separated by AND are TRUE.

• The OR operator displays a record if any of the conditions separated by OR is TRUE.

• The NOT operator displays a record if the condition(s) is NOT TRUE

**SQL ORDER BY Keyword**

SELECT \* FROM Customers WHERE Country='Germany' AND City='Berlin';

**SQL ORDER BY Keyword**

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

•ORDER BY Syntax

• SELECT column1, column2, ... FROM table\_name ORDER BY column1, column2, ... ASC|DESC;

**SQL UPDATE Statement**

The UPDATE statement is used to modify the existing records in a table.

UPDATE Syntax

UPDATE table\_name SET column1 = value1, column2 = value2, ... WHERE condition;

**SQL DELETE Statement**

The DELETE statement is used to delete existing records in a table.

DELETE Syntax

DELETE FROM table\_name WHERE condition;

Example

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

**SQL MIN() and MAX() Functions**

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

MIN() Syntax

SELECT MIN(column\_name) FROM table\_name WHERE condition;

**SQL COUNT(), AVG() and SUM() Functions**

The COUNT() function returns the number of rows that matches a specified criterion.

The AVG() function returns the average value of a numeric column.

The SUM() function returns the total sum of a numeric column.

**SQL LIKE Operator**

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

% - The percent sign represents zero, one, or multiple characters

\_ - The underscore represents a single character

**SQL IN Operator**

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

IN Syntax

SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1, value2, ...);

**SQL BETWEEN Operator**

The BETWEEN operator selects values within a given range.

The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

BETWEEN Syntax

SELECT column\_name(s) FROM table\_name WHERE column\_name BETWEEN value1 AND value2;

**SQL Aliases**

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of the query.

Alias Column Syntax

SELECT column\_name AS alias\_name FROM table\_name;

**SQL Joins**

**SQL INNER JOIN Keyword**

The INNER JOIN keyword selects records that have matching values in both tables.

INNER JOIN Syntax

SELECT column\_name(s) FROM table1 INNER JOIN table2 ON table1.column\_name = table2.column\_name;

Example

SELECT Orders.OrderID, Customers.CustomerName FROM Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

Note: The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns. If there are records in the "Orders" table that do not have matches in "Customers", these orders will not be shown!

**SQL LEFT JOIN Keyword**

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.

LEFT JOIN Syntax

SELECT column\_name(s) FROM table1 LEFT JOIN table2 ON table1.column\_name = table2.column\_name;

**SQL RIGHT JOIN Keyword**

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

RIGHT JOIN Syntax

SELECT column\_name(s) FROM table1 RIGHT JOIN table2 ON table1.column\_name = table2.column\_name;

**SQL FULL OUTER JOIN Keyword**

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

Note: FULL OUTER JOIN can potentially return very large result-sets!

Tip: FULL OUTER JOIN and FULL JOIN are the same.

FULL OUTER JOIN Syntax

SELECT column\_name(s) FROM table1 FULL OUTER JOIN table2 ON table1.column\_name = table2.column\_name WHERE condition;

**THANK YOU**