RDBMS

Reading Material

PW SKILLS

Topics Covered

- Understanding the concept of RDBMS
  - What is RDBMS?
  - Definition and importance of Data Modelling
  - Different Types of Data Models
  - What is the Relationship in RDBMS & Its Types
  - Keys in RDBMS
  - ACID
  - Normalization
  - Triggers
  - Constraints
  - Indexes
  - Join
  - Dependencies
  - Transaction and Concurrency Control
  - Database security

What is RDBMS

An RDBMS is a software system that manages and organizes data in a structured, tabular format, following the relational model. It offers advantages such as data integrity, security, scalability, concurrency control, and a standardized language. RDBMSs enforce rules and constraints to maintain data consistency and accuracy while providing robust security features. They can handle large amounts of data and can scale vertically or horizontally.

RDBMSs are commonly used in financial institutions, e-commerce platforms, and healthcare organizations. The core components of an RDBMS include tables, rows, columns, keys, and relationships. These components ensure data consistency, accuracy, scalability, and concurrency control. Examples of RDBMS usage include financial institutions, e-commerce platforms, and healthcare organizations.

Definition and importance of Data Modelling

At its core, data modeling is the process of defining and designing the structure of a database. It's like creating a map that outlines how data elements relate to each other and how they should be stored and accessed. Data modeling is crucial because it lays the foundation for building databases that accurately reflect real-world scenarios, making it easier to manage and query data effectively.

Different Types of Data Models

1. Conceptual Data Model
2. Logical Data Model
3. Physical Data Model

What is the Relationship in RDBMS & Its Types

Relationships in a Relational Database Management System (RDBMS) define how data is associated or related between tables. They establish connections or links between the data stored in different tables, allowing you to combine and retrieve information from multiple sources.

three main types

1. **One-to-One:** One record in the first table is associated with one and only one record in the second table.
2. **One-to-Many:** One record in the first table is associated with multiple records in the second table.
3. **Many-to-Many:** Multiple records in the first table are associated with multiple records in the second table.

Keys in RDBMS

1. **Primary Key:** A unique identifier for each row in a table.
2. **Foreign Key:** A column (or combination of columns) in one table that refers to the primary key in another table.
3. **Candidate Key:** A column (or combination of columns) that can uniquely identify each row in a table.
4. **Super Key:** A combination of columns that can uniquely identify each row in a table.

ACID

ACID stands for Atomicity, Consistency, Isolation, and Durability. These are the four fundamental properties that ensure database transactions are processed reliably.

1. **Atomicity:** A transaction is an indivisible unit of work, either all operations are completed, or none are.
2. **Consistency:** A transaction must transform the database from one valid state to another valid state.
3. **Isolation:** Concurrent transactions must not interfere with each other.
4. **Durability:** Once a transaction is committed, its effects are permanently stored in the database.

Normalization

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves breaking down a database into smaller tables and defining relationships between them.

1. **First Normal Form (1NF):** Eliminates duplicate rows and repeating groups.
2. **Second Normal Form (2NF):** Eliminates partial dependencies.
3. **Third Normal Form (3NF):** Eliminates transitive dependencies.

Triggers

A trigger is a special kind of stored procedure in a database that is automatically executed when a specific event occurs. Triggers can be defined as responding to INSERT, UPDATE, or DELETE operations on a table. They are commonly used for enforcing complex business rules, maintaining data integrity across multiple tables, and auditing or logging changes to data.

There are different types of triggers:

1. **BEFORE triggers:** These triggers are executed before the triggering event (INSERT, UPDATE, or DELETE) is performed on the table.
2. **AFTER triggers:** These triggers are executed after the triggering event is performed on the table.
3. **INSTEAD OF triggers:** These triggers are used with views and are executed instead of the triggering event on the underlying tables.

Triggers can be powerful tools for data validation, automatic generation of values, and maintaining referential integrity, but they should be used judiciously as they can impact database performance if not designed properly.

Constraints

Constraints are rules or restrictions enforced on the data in a database to maintain data integrity and consistency. There are several types of constraints in an RDBMS:

1. **Primary Key:** Ensures that the values in a column (or combination of columns) are unique and non-null.
2. **Foreign Key:** Establishes a link between two tables by referencing the primary key of another table.
3. **Unique:** Ensures that the values in a column (or combination of columns) are unique.
4. **Check:** Defines a condition that each row in the table must satisfy.
5. **Not Null:** Ensures that a column cannot have null values.

Constraints are essential for maintaining data integrity and ensuring that the data in the database follows the defined rules and relationships.

Indexes

An index is a database object that stores a subset of data from a table in a structured and efficient way. Indexes are used to improve the performance of queries by allowing the database to quickly locate and retrieve specific data without having to scan the entire table.

There are different types of indexes:

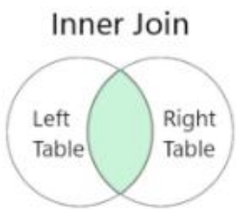
1. **Clustered Index:** Physically reorders the rows in a table based on the index key values.
2. **Non-Clustered Index:** Stores the index key values and row locators (pointers) to the actual data rows.
3. **Unique Index:** Ensures that the index key values are unique.
4. **Composite Index:** An index created on multiple columns.

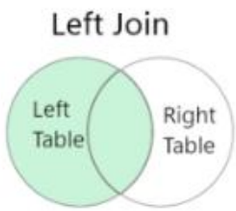
Indexes can significantly improve query performance, especially for large tables and complex queries involving joins, sorting, or range searches. However, indexes also increase the storage requirements and can impact insert, update, and delete operations due to the overhead of maintaining the index structures.

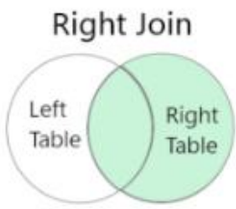
Join

A join is an operation that combines rows from two or more tables based on a related column between them. Common types of joins include:

1. **Inner Join:** Returns only the matching rows from both tables.
2. **Left Join:** Returns all rows from the left table, and matching rows from the right table.
3. **Right Join:** Returns all rows from the right table, and matching rows from the left table.
4. **Full Join:** Returns all rows from both tables, combining matching and non-matching rows.







Dependencies

1. **Functional Dependency:** A relationship between two sets of attributes in a relation, where the values of one set (determinant) uniquely identify the values of the other set (dependent).
2. **Inclusion Dependency:** A relationship between two sets of attributes in a relation, where the values of one set are a subset of the values in the other set.
3. **Multivalued Dependency:** A relationship between two sets of attributes in a relation, where the values of one set (determinant) uniquely identify a set of values in the other set (dependent).
4. **Join Dependency:** A relationship between two sets of attributes in a relation, where all attributes are functionally dependent on the combination of primary keys from the two tables being joined.

Transactions and Concurrency Control

Transactions are a sequence of operations performed as a single logical unit of work. Concurrency control ensures that multiple transactions can access the same data simultaneously without causing data inconsistency or corruption.

1. **Locking:** A mechanism that prevents other transactions from accessing or modifying data that is currently being used by another transaction.
2. **Timestamping:** A method of assigning a unique timestamp to each transaction, which is used to determine the order of execution and resolve conflicts.
3. **Optimistic vs. Pessimistic Concurrency Control:** Optimistic concurrency control assumes conflicts are rare and checks for conflicts at the end of a transaction. Pessimistic concurrency control assumes conflicts are common and uses locking to prevent them.

Database Security

Database security involves measures to protect the data stored in a database from unauthorized access, modification, or destruction.

1. **Access Control:** Restricting access to the database and its contents based on user roles and permissions.
2. **Authentication:** Verifying the identity of users or applications attempting to access the database.
3. **Encryption:** Encoding data to prevent unauthorized access and ensure data confidentiality.
4. **Auditing:** Monitoring and recording activities and events in the database for security and compliance purposes.