**SQL Assignment 1**

1. What is a relational database management system (RDBMS)? What are the advantages of a database management system over a file system?

Ans - A Relational Database Management System (RDBMS) is a software system designed to manage relational databases. It's a type of database management system (DBMS) that organizes and manages data in a structured format based on the relational model, where data is stored in tables that are related to each other using common fields.

Key features of an RDBMS include:

* Data Structuring: Organizes data into tables, rows, and columns, allowing for efficient storage and retrieval.
* Data Integrity: Enforces data integrity through constraints like primary keys, foreign keys, and unique constraints to maintain data accuracy and consistency.
* Data Querying: Supports a structured query language (SQL) to perform queries for data retrieval, manipulation, and management.
* ACID Properties: Ensures transactions (groups of database operations) adhere to Atomicity, Consistency, Isolation, and Durability properties for data reliability.

Advantages of RDBMS over file systems:

* Data Integrity: RDBMS offers robust data integrity through normalization, constraints, and relationships, ensuring accurate and consistent data storage.
* Data Abstraction: Allows users to interact with data through a high-level query language (SQL) without needing to understand the underlying data storage mechanisms.
* Efficient Data Retrieval: RDBMS supports indexing, allowing for quick and efficient data retrieval, even from large datasets.
* Data Security: Provides features like user authentication, access control, and encryption to secure sensitive data.
* Scalability and Flexibility: RDBMS systems are scalable and adaptable to changing business needs, supporting both small-scale and enterprise-level applications.
* Concurrency Control: Ensures multiple users can access the database simultaneously without affecting data consistency.

1. In a database management system, explain the ACID properties.

Ans - ACID is an acronym that stands for Atomicity, Consistency, Isolation, and Durability. These properties are crucial characteristics of transactions within a database management system (DBMS), ensuring reliability, consistency, and integrity of data despite system failures, errors, or concurrent access by multiple users.

1. **Atomicity:** This property ensures that a transaction is treated as a single unit of work. It means that all the operations within a transaction are executed entirely or not executed at all. If any part of the transaction fails, the entire transaction is rolled back (undone) to its initial state, maintaining the database's consistency.

It involves the following two operations.

—Abort: If a transaction aborts, changes made to the database are not visible.

—Commit: If a transaction commits, changes made are visible.

Atomicity is also known as the ‘All or nothing rule’.

* Example: A bank transfer from Account A to Account B.
* Explanation**:** Suppose a transaction involves debiting $500 from Account A and crediting the same amount to Account B. If either the debit or credit operation fails due to a system error or interruption, the entire transaction should be rolled back to maintain consistency. Atomicity ensures that either both the debit and credit operations occur successfully or none at all.

1. **Consistency:** The consistency property guarantees that the database remains in a valid state before and after the transaction. In other words, if the data is consistent before a transaction begins, it should remain consistent after the transaction is completed, irrespective of any failure or concurrent transactions.  It refers to the correctness of a database.

Example: Maintaining a balance of total assets and liabilities in a banking system.

Explanation: A banking system must ensure that the total sum of assets equals the total sum of liabilities at all times. Even during multiple concurrent transactions involving deposits, withdrawals, or transfers, the system must maintain this consistency to avoid any discrepancies.

1. **Isolation:** Isolation ensures that the operations within a transaction are performed independently of other concurrent transactions. It prevents interference between transactions, maintaining data integrity and preventing concurrency-related issues like dirty reads, non-repeatable reads, and phantom reads.

Example: Simultaneous transactions involving Account A and Account B.

Explanation: Imagine two customers initiate transactions at the same time, transferring money between Account A and Account B. Isolation ensures that each transaction is executed independently, preventing interference between the transactions. It ensures that the actions within one transaction do not affect the other until they are completed and committed.

1. **Durability:** Durability guarantees that once a transaction is committed, its effects persist even in the event of system failures or crashes. The changes made by the committed transaction are permanently stored in the database and will not be lost.

Example: Recording a completed transaction in a banking database.

Explanation: After a customer completes a transaction (e.g., a fund transfer), it's committed to the database and becomes permanent. Even in the event of a system crash or power failure immediately after the commit, the details of the transaction remain intact and are not lost. Durability ensures that the committed changes persist in the database, guaranteeing their permanence.

These ACID properties collectively ensure that transactions in a database are reliable, maintain data integrity, and provide a robust foundation for data management even in complex and concurrent environments. They are fundamental for maintaining the accuracy, reliability, and consistency of data within a database system.

1. Explain the concept of normalization.

Ans – It is the process of organizing the data in a database. It helps in removing the duplicate values in the database. Normalization divides the large table into smaller tables and links them using relationships.

The normal form is used to reduce redundancy from the database table. Normalization is the name given to the process of simplifying the relationship among data elements in a record.

In simple words we can say,

Normalization is the process of organizing data to minimize.

* Redundancy/duplication/repetition.
* Insertion, deletion, updating anomalies.

There are six Normal forms which are as follows –

|  |  |
| --- | --- |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | A relation is in 1NF if it contains an atomic value. In simple words, a cell cannot hold multiple values. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| [3NF](https://www.javatpoint.com/dbms-third-normal-form) | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |
| BCNF | A stronger definition of 3NF is known as Boyce Codd's normal form. Every function dependency A -> B, then A has to super key of that table |
| [4NF](https://www.javatpoint.com/dbms-forth-normal-form) | A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless. |

1. Explain the many types of query languages used in relational databases. DQL, DML, DCL, and DDL are some examples.

Ans - In relational databases, there are several types of query languages that serve distinct purposes in managing and manipulating data. Here are the common types:

1. Data Query Language (DQL):
   * Purpose: DQL is used to retrieve data from the database.
   * Example: SQL (Structured Query Language) SELECT statement is a typical example of DQL, used to fetch data from tables based on specified criteria.
2. Data Manipulation Language (DML):
   * Purpose: DML is used for manipulating data within the database.
   * Examples: SQL commands such as INSERT, UPDATE, and DELETE are part of DML. These commands add, modify, or delete data records in the tables.
3. Data Definition Language (DDL):
   * Purpose: DDL is used to define the structure and schema of the database.
   * Examples: SQL commands like CREATE, ALTER, DROP, and TRUNCATE are part of DDL. These commands create, modify, or delete database objects like tables, indexes, views, or constraints.
4. Data Control Language (DCL):
   * Purpose: DCL is used for managing access to data stored in the database.
   * Examples: SQL commands like GRANT and REVOKE fall under DCL. These commands grant or revoke access privileges and permissions to users or roles within the database.

1. What is the difference between the main key and a composite key? Give instances of how primary key and composite are used.

Ans - The main difference between a primary key and a composite key lies in their composition and purpose within a database:

1. Primary Key:
   * A primary key is a unique identifier for a single record in a table.
   * It ensures data integrity by uniquely identifying each row in a table and prevents duplicate or null values.
   * A primary key constraint ensures that the key values are unique and not null.
   * In most cases, a primary key is composed of a single column.

Example of Primary Key: In a "Students" table, the column "Student\_ID" can be designated as the primary key. Each student's ID would be unique, acting as a primary identifier for that student's record. This ensures that no two students can have the same ID within the table.

1. Composite Key:
   * A composite key is composed of two or more columns that together uniquely identify a record within a table.
   * It is used when a single column cannot uniquely identify a record, but the combination of multiple columns can.
   * The composite key constraint ensures the uniqueness of the combined values across these columns.
   * It's helpful in cases where individual columns may have duplicate values, but their combinations are unique.

Example of Composite Key: Consider an "Orders" table in an e-commerce database. A composite key could consist of "Order\_ID" and "Product\_ID." Here, a single order ID might correspond to multiple products, but the combination of both the order ID and product ID will uniquely identify a particular product within an order.

In summary, while a primary key uniquely identifies a single record in a table using a single column, a composite key uniquely identifies a record using a combination of multiple columns. Both primary and composite keys play crucial roles in maintaining data integrity and ensuring unique identification of records within a database table.

1. Create a table with a primary key, a column default value, and a column unique constraint in SQL.

Ans - CREATE TABLE Students

( Student\_ID INT PRIMARY KEY,

Name VARCHAR(50) DEFAULT 'John Doe',

Email VARCHAR(100) UNIQUE );

Explanation:

* **Students** is the name of the table being created in the student database.
* **Student\_ID** column is defined as an integer and designated as the primary key using **PRIMARY KEY**.
* **Name** column is of type VARCHAR(50) and has a default value 'John Doe' assigned using **DEFAULT**.
* **Email** column is a VARCHAR(100) type and is specified as unique using **UNIQUE**, ensuring uniqueness across all values in this column.