**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?

A relational database management system (RDBMS) is a type of software that is used to manage and organize data in a relational database. In a relational database, data is stored in tables with columns and rows, and these tables are related to each other using common fields or keys. An RDBMS provides tools for managing, organizing, and querying data in these tables, and can also enforce data integrity rules and support transactions.

Advantages of a database management system over a file system include:

Data Integrity: An RDBMS can enforce rules to ensure that data is accurate, consistent, and complete. For example, it can ensure that a record cannot be inserted into a table if it violates a unique constraint or if it references a non-existent record in another table.

Concurrent Access: An RDBMS allows multiple users or applications to access the same data simultaneously, without the risk of data corruption or inconsistency that can occur in a file system.

Scalability: An RDBMS can handle large amounts of data and scale to support increasing numbers of users and applications.

Security: An RDBMS can provide access control mechanisms to ensure that only authorized users can access or modify data.

Data Consistency: An RDBMS ensures that data is consistent throughout the system, and can rollback transactions if errors occur to maintain data consistency.

Flexibility: An RDBMS allows users to easily manipulate, retrieve and update data without having to know the internal workings of the system.

Overall, an RDBMS is a powerful and efficient way to manage data, and can provide significant advantages over a file system for applications that require large amounts of data, concurrency, and data integrity.

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

ACID is an acronym that stands for Atomicity, Consistency, Isolation, and Durability. These properties are essential for database management systems to ensure that data remains consistent and accurate even in the event of hardware or software failures, concurrent access, or other issues.

Atomicity: This property ensures that a transaction is treated as a single, indivisible unit of work. It means that either all the changes made by the transaction are committed to the database, or none of them are committed. If any part of the transaction fails, the entire transaction is rolled back, and the database is restored to its previous state.

Consistency: This property ensures that a transaction brings the database from one valid state to another valid state. It means that the data in the database must satisfy all the rules and constraints imposed on it, even after the transaction is completed.

Isolation: This property ensures that each transaction is executed in isolation from other transactions. It means that the changes made by one transaction are not visible to other transactions until the first transaction is committed. This prevents concurrency issues such as dirty reads, non-repeatable reads, and phantom reads.

Durability: This property ensures that once a transaction is committed, its changes are permanent and cannot be lost, even in the event of a system failure or a power outage. The changes made by the transaction must be recorded in non-volatile memory, such as a hard disk, and must be available for future transactions to read.

In summary, the ACID properties are a set of guidelines that ensure that database transactions are reliable, consistent, and recoverable. These properties are crucial for maintaining the integrity of data in a database management system.

1. Explain the concept of normalization.

Normalization is a technique used in databases to organize data into multiple tables and reduce data redundancy. The process of normalization involves breaking down a larger table into smaller tables with fewer columns, creating relationships between these tables, and applying specific rules to ensure data consistency and integrity.

The main purpose of normalization is to minimize data duplication and reduce the chances of data inconsistencies and anomalies that may occur due to data redundancy. This is accomplished by separating data into smaller, related tables and applying normalization rules, which help to eliminate data inconsistencies and anomalies that might arise due to data redundancy.

Normalization involves a set of guidelines or rules that are applied to a database schema. There are several levels of normalization, starting with first normal form (1NF), second normal form (2NF), third normal form (3NF), and so on. Each level of normalization builds upon the previous one, with the ultimate goal of ensuring that each piece of data is stored in only one place and that all the data is logically organized and easy to query.

Overall, normalization is a crucial aspect of database design that helps to ensure data consistency, integrity, and reliability. It can also improve database performance and scalability by reducing the amount of data that needs to be stored and processed.

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

There are several types of query languages used in relational databases, including DQL, DML, DCL, and DDL. Each of these languages has a specific purpose and syntax.

Data Query Language (DQL): DQL is a language used to retrieve data from a database. It includes commands like SELECT, WHERE, and FROM, which are used to specify which data to retrieve and how to filter it. DQL is commonly used in applications that require data analysis and reporting.

Data Manipulation Language (DML): DML is a language used to modify data in a database. It includes commands like INSERT, UPDATE, and DELETE, which are used to add, modify, and remove data from a database. DML is commonly used in applications that require data entry and maintenance.

Data Control Language (DCL): DCL is a language used to control access to data in a database. It includes commands like GRANT and REVOKE, which are used to grant or revoke access to specific tables or data in a database. DCL is commonly used in applications that require secure access to sensitive data.

Data Definition Language (DDL): DDL is a language used to define the structure of a database. It includes commands like CREATE, ALTER, and DROP, which are used to create and modify tables, indexes, and other database objects. DDL is commonly used in applications that require database schema changes.

In summary, DQL is used for retrieving data, DML is used for modifying data, DCL is used for controlling access to data, and DDL is used for defining the structure of a database.

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

A primary key is a column or set of columns in a table that uniquely identifies each row in the table. A composite key, on the other hand, is a key that is made up of two or more columns in a table, which together uniquely identify each row in the table.

The main difference between a primary key and a composite key is that a primary key is a single column or set of columns that uniquely identifies each row in the table, while a composite key is made up of two or more columns in a table.

For example, consider a table called "orders" that contains the following columns: order\_id, customer\_id, and order\_date. If the order\_id column is designated as the primary key for this table, each row in the table will be uniquely identified by the order\_id column. However, if the order\_id and customer\_id columns together uniquely identify each row in the table, these columns can be designated as a composite key.

Instances where a primary key and composite key might be used include:

In a customer orders table, the primary key might be the order\_id column, which is automatically assigned when a new order is created. However, if there are multiple orders per customer, a composite key made up of the customer\_id and order\_id columns might be used to ensure that each order is uniquely identified.

In a product inventory table, the primary key might be the product\_id column, which uniquely identifies each product in the inventory. However, if there are multiple inventory locations for each product, a composite key made up of the product\_id and location columns might be used to ensure that each inventory item is uniquely identified.

Overall, both primary keys and composite keys serve to ensure that each row in a table is uniquely identified, but they differ in terms of the number of columns used to accomplish this goal.

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

CREATE TABLE example\_table (

id INTEGER PRIMARY KEY,

name VARCHAR(50) DEFAULT 'Unknown',

email VARCHAR(100) UNIQUE

);