The 14 Codd rules are:

Information rule: All information in the database should be represented explicitly at the logical level.

Access rule: Each and every data item (i.e., value) must be addressable by a unique combination of primary key values.

Subschema rule: A relational database schema is a high-level view of the data. A subschema is a subset of the schema that reflects the information needs of a particular user or group of users.

Update rule: All changes to the database must be made through the schema.

Physical data independence: The physical storage of data should be independent of the logical structure of the data.

Logical data independence: The logical structure of the data should be independent of the physical storage of the data.

View rule: A view is a virtual table that is derived from one or more base tables.

Modification rule: The insertion, deletion, or modification of data in one table should not cascade to other tables.

Integrity rule: The database should maintain the integrity of the data.

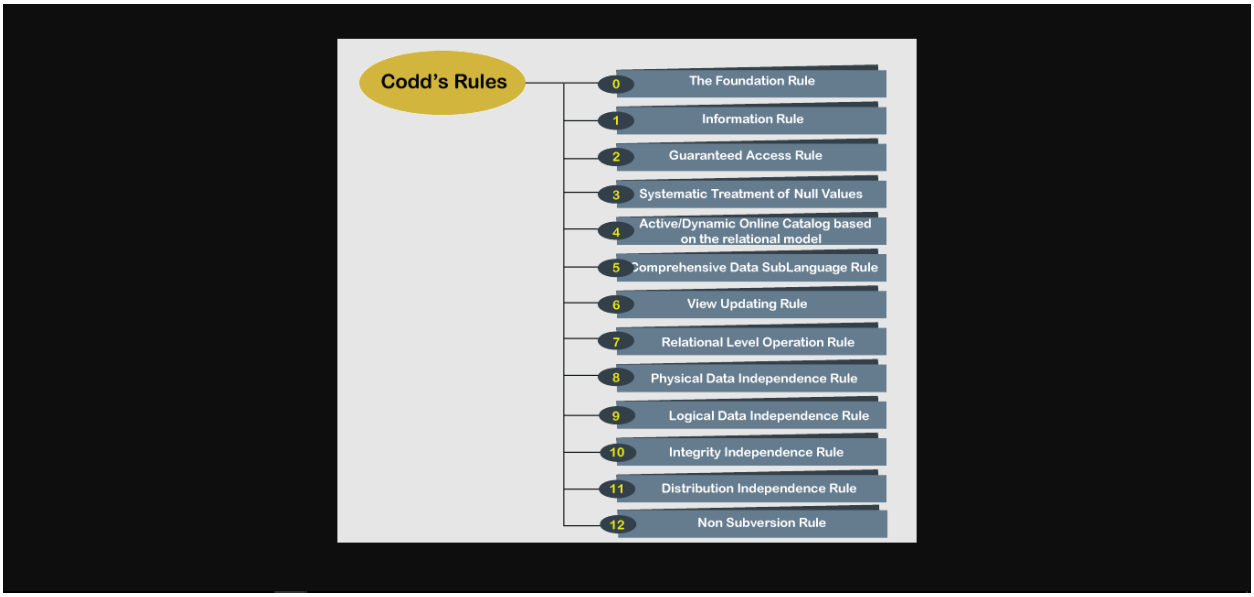
Distribution rule: The distribution of data over multiple sites should be transparent to the user.

Heterogeneity rule: The database should be able to store and access data from multiple data sources.

Concurrency rule: The database should be able to handle multiple users accessing and modifying data simultaneously.

Security rule: The database should protect the data from unauthorized access.

Simplicity rule: The database should be simple to understand and use.



Discuss the characters someone need to escape in XML documents?

There are five characters that need to be escaped in XML documents:

Ampersand (&)

Less than sign (<)

Greater than sign (>)

Double quote (")

Apostrophe (')

These characters are special to XML because they are used in the syntax of the language to define elements, attributes, and other constructs. For example, the less than sign (<) is used to start an element, and the greater than sign (>) is used to end an element.

If these characters are used in the content of an XML document, they must be escaped using either numeric character references (NCRs) or character entity references.

Numeric character references are represented by an ampersand (&), followed by a number, followed by a semicolon (;). The number is the ASCII code for the

character that is being escaped. For example, the NCR for the ampersand character is &38;.

Character entity references are represented by an ampersand (&), followed by the name of the character entity, followed by a semicolon (;). For example, the character entity reference for the ampersand character is &.

Discuss the various types of interactions catered by DBMS?

1. Data definition

Data definition operations allow users to create, modify, and delete database objects, such as tables, views, and stored procedures. This includes defining the structure of the database, such as the columns and data types for each table.

Examples of data definition operations include:

CREATE TABLE

ALTER TABLE

DROP TABLE

CREATE VIEW

DROP VIEW

CREATE PROCEDURE

DROP PROCEDURE

2. Data update

Data update operations allow users to insert, update, and delete data from the database. This includes adding new records, changing existing records, and removing records that are no longer needed.

Examples of data update operations include:

INSERT INTO

UPDATE

DELETE

3. Data retrieval

Data retrieval operations allow users to query the database to retrieve data. This includes selecting specific records, filtering data based on certain criteria, and performing aggregations on the data.

Examples of data retrieval operations include:

SELECT

WHERE

ORDER BY

GROUP BY

HAVING

4. Data administration

Data administration operations allow users to manage the database, such as creating and managing user accounts, backing up and restoring the database, and monitoring the performance of the database.

Examples of data administration operations include:

CREATE USER

DROP USER

GRANT

REVOKE

BACKUP

RESTORE

SHOW DATABASES

SHOW TABLES

SHOW INDEXES

DBMSs also typically provide a variety of other features to support these types of interactions, such as:

Transactions: Transactions allow users to group multiple data operations together and ensure that they are all completed successfully or rolled back if any error occurs.

Concurrency control: Concurrency control mechanisms allow multiple users to access the database simultaneously without interfering with each other's data.

Security: DBMSs provide a variety of security features to protect the database from unauthorized access and modification.

What are the features of Database language?

Data definition: Database languages allow users to define the structure of a database, including the tables, columns, and data types for each column.

Data manipulation: Database languages allow users to insert, update, delete, and retrieve data from a database.

Data querying: Database languages allow users to query the database to retrieve specific data based on certain criteria.

Data integrity: Database languages help to ensure the integrity of the data in a database by enforcing constraints on the data, such as unique values and required fields.

Security: Database languages can be used to control access to the data in a database and to protect it from unauthorized modification.

There are five main types of SQL joins:

1. Inner join: An inner join returns all rows from both tables where the join condition is met. The join condition is typically a comparison between two columns in the two tables. For example, you could use an inner join to combine data from a customer table and an order table, matching the customer ID column in both tables.

2. Left join: A left join returns all rows from the left table, even if there is no matching row in the right table. The right table is typically a reference table, such as a product table or a country table. For example, you could use a left join to combine data from a customer table and an order table, returning all customers even if they have not placed any orders.

3. Right join: A right join returns all rows from the right table, even if there is no matching row in the left table. The left table is typically a transaction table, such as an order table or a sales table. For example, you could use a right join to combine

data from a customer table and an order table, returning all orders even if the customer does not exist in the customer table.

4. Full outer join: A full outer join returns all rows from both tables, even if there is no matching row in the other table. This is the most inclusive type of join, and it is often used to generate

5. Self join: A self join is a join between two tables that are the same. This type of join is often reports that show all of the data in both tables.used to perform complex queries on a single table. For example, you could use a self join to find all of the customers who have placed multiple orders.

The main difference between SQL and MySQL is that SQL is a language, while MySQL is a software implementation of that language. SQL is a standard language that is supported by many different RDBMSs, including MySQL, PostgreSQL, and Microsoft SQL Server. MySQL is a specific RDBMS that is known for its open-source nature, performance, and scalability.

A transaction is a logical unit of work in a database system. It is a sequence of operations that are performed on the database as a single unit. Transactions are used to ensure that the data in the database is consistent and that any changes to the data are made atomically.

There are five main states of a transaction:

Active: The transaction is being executed.

Partially committed: The transaction has completed all of its operations, but the changes to the database have not yet been made permanent.

Committed: The transaction has completed all of its operations and the changes to the database have been made permanent.

Failed: The transaction has failed and the changes to the database have been rolled back.

Aborted: The transaction has been aborted and the changes to the database have been rolled back.

A transaction is a unit of work performed within a database management system against a database as a single logical operation. A transaction typically represents

any change in a database. A transaction is characterized by its atomicity, consistency, isolation, and durability, commonly known as the ACID properties:

Atomicity: Either all of the operations in a transaction succeed, or none of them succeed. This is also known as all-or-nothing.

Consistency: A transaction transitions the database from one valid state to another valid state.

Isolation: Transactions are isolated from each other, so that one transaction cannot see the uncommitted changes of another transaction.

Durability: Once a transaction has committed, its changes are permanent and cannot be rolled back.

A centralized database is a database that is stored and managed on a single server. All of the data in the database is stored on the same server, and all of the transactions that are performed on the database are processed by the same server.

A decentralized database is a database that is distributed across multiple servers. The data in the database is stored on multiple servers, and each server can process transactions independently.

There are several reasons why databases need to be normalized:

To reduce data redundancy: Data redundancy occurs when the same data is stored in multiple places in the database. This can lead to inconsistencies in the data and make it difficult to update and maintain the database.

To improve data integrity: Data integrity refers to the accuracy and consistency of the data in the database. Normalization can help to improve data integrity by reducing data redundancy and by ensuring that all data items are stored in the most appropriate place.

To improve database performance: Normalization can improve database performance by reducing the amount of data that needs to be stored and by making it easier for the database to retrieve the data that it needs.

To make the database easier to understand and maintain: Normalization can make the database easier to understand and maintain by organizing the data into logical tables and relationships.

There are several different levels of database normalization, each of which provides additional benefits. The most common levels of normalization are:

First normal form (1NF): 1NF ensures that all data items in a table are atomic and that there are no repeating groups.

Second normal form (2NF): 2NF ensures that all non-key attributes in a table are fully dependent on the primary key.

Third normal form (3NF): 3NF ensures that all non-key attributes in a table are directly dependent on the primary key and not on any other non-key attribute.

Explain the following SQL terms: Unions, Having, Group By, Distinct, and SQL Joins?

volume_up

Here is an explanation of the following SQL terms:

Unions

The UNION operator combines the results of two or more SELECT statements into a single result set. The result set of the UNION operator will contain all of the rows from the SELECT statements, including any rows that are duplicated in multiple SELECT statements.

To use the UNION operator, you must specify the SELECT statements that you want to combine, and you must ensure that the SELECT statements have the same number of columns and the same data types in the corresponding columns.

Here is an example of a UNION query:

SQL

```
SELECT name, email
```

```
FROM customers
UNION
SELECT name, email
FROM employees;
```
Use code with caution. Learn more

content_copy

This query will return a single result set that contains all of the customers and employees from the customers and employees tables.

Having

The HAVING clause is used to filter the groups of rows returned by the GROUP BY clause. The HAVING clause can be used to filter the groups based on the values of the aggregate functions that are used in the SELECT statement.

Here is an example of a HAVING clause:

SQL

```
SELECT product_category, SUM(amount) AS total_sales
FROM orders
GROUP BY product_category
HAVING SUM(amount) > 10000;
```
Use code with caution. Learn more

content_copy

This query will return the total sales for each product category, but it will only return the product categories that have total sales greater than 10000.

Group By

The GROUP BY clause is used to group the rows in a table based on the values in one or more columns. The GROUP BY clause can be used with aggregate functions to calculate summary statistics for the groups of rows.

Here is an example of a GROUP BY clause:

SQL

```
SELECT product_category, SUM(amount) AS total_sales
FROM orders
GROUP BY product_category;
```

Use code with caution. Learn more

content_copy

This query will return the total sales for each product category.

Distinct

The DISTINCT keyword is used to return only the unique rows from a result set. The DISTINCT keyword can be used with or without a GROUP BY clause.

Here is an example of a DISTINCT keyword:

SQL

```
SELECT name
FROM customers
DISTINCT;
```

Use code with caution. Learn more

content_copy

This query will return a list of all of the unique customers in the customers table.

SQL Joins

SQL joins are used to combine data from two or more tables based on a common field between the tables. There are four main types of SQL joins:

- Inner join: An inner join returns all of the rows from both tables that match the common field.
- Left join: A left join returns all of the rows from the left table, even if there are no matching rows in the right table.

- Right join: A right join returns all of the rows from the right table, even if there are no matching rows in the left table.
- Full outer join: A full outer join returns all of the rows from both tables, even if there are no matching rows in either table.

Here is an example of an inner join:

SQL

```
SELECT customers.name, orders.product_name
FROM customers
INNER JOIN orders ON customers.customer_id = orders.customer_id;
```