Relational Database

In this lesson, we will discuss the relational databases.

We'll cover the following

What Is A Relational Database?

This is the most common & widely used type of database in the industry. A relational database saves data containing relationships. *One to One*, *One to Many*, *Many to Many*, *Many to One* etc. It has a relational data model. *SQL* is the primary data query language used to interact with relational databases.

MySQL is the most popular example of a relational database. Alright!! I get it but what are relationships?

What Are Relationships?

Let's say you as a customer buy five different books from an online book store. When you created an account on the book store you would have been assigned a customer id say C1. Now that C1[You] is linked to five different books B1, B2, B3, B4, B5.

This is a *one to many* relationship. In the simplest of forms, one table will contain the details of all the customers & another table will contain all the products in the inventory.

One row in the customer table will correspond to multiple rows in the product inventory table.

On pulling the user object with id C1 from the database we can easily find

On pulling the user object with id C1 from the database we can easily find what books C1 purchased via the relationship model.

Data Consistency

Besides, the relationships, relational databases also ensure saving data in a normalized fashion. In very simple terms, normalized data means a unique entity occurs in only one place/table, in its simplest and atomic form and is not spread throughout the database.

This helps in maintaining the consistency of the data. In future, if we want to update the data, we just update at that one place and every fetch operation gets the updated data.

Had the data been spread throughout the database in different tables. We would have to update the new value of an entity everywhere. This is troublesome and things can get inconsistent.

ACID Transactions

Besides normalization & consistency, relational databases also ensure *ACID* transactions.

ACID – Atomicity, Consistency, Integrity, Durability.

An acid transaction means if a transaction in a system occurs, say a financial transaction, either it will be executed with perfection without affecting any other processes or transactions.

The system will have a new state after the transaction which is durable & consistent. Or if anything, amiss happens during the transaction, say a minor system failure, the entire operation is rolled back.

When a transaction happens, there is an initial state of the system *State A* & then there is a final state of the system *State B* after the transaction. Both the states are consistent and durable.

A relational database ensures that either the system is in *State A* or *State B* at all times. There is no middle state. If anything fails, the system goes back to *State A*.

If the transaction is executed smoothly the system transitions from *State A* to

State B.