SQL – Basics

Presentation by Uplatz
Contact us: https://training.uplatz.com
Email: info@uplatz.com
Phone: +44 7836 212635

Overview

- What is SQL?
- What can SQL do?
- Introduction to RDBMS
- SQL and it’s process
- Acid property
- ER Model and relationships
- Database Normalization

What is SQL

- SQL stands for Structured Query Language.
- SQL lets you access and manipulate databases.
- SQL is the language to retrieve, update, modify, delete or insert any new information in database.
- SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

What can SQL do?

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows users to create and drop databases and tables.
- Allows users to retrieve, insert, update, delete, create new tables
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.
- SQL can execute queries against a database
- SQL can set permissions on tables, procedures, and views

SQL is a Standard-BUT....

- Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

- However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

Introduction to RDBMS

RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

SQL Process

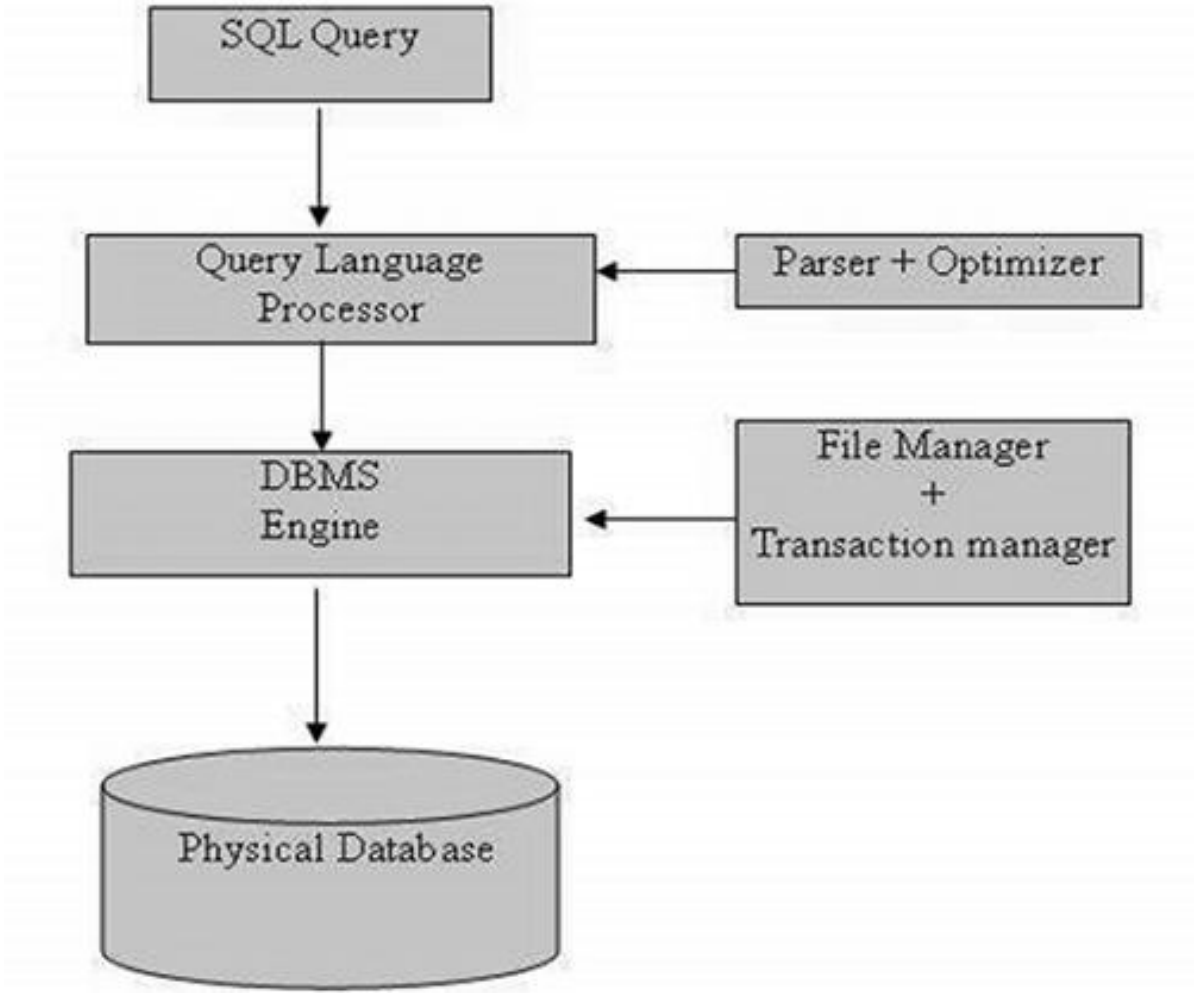
When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

There are various components included in this process.

These components are −

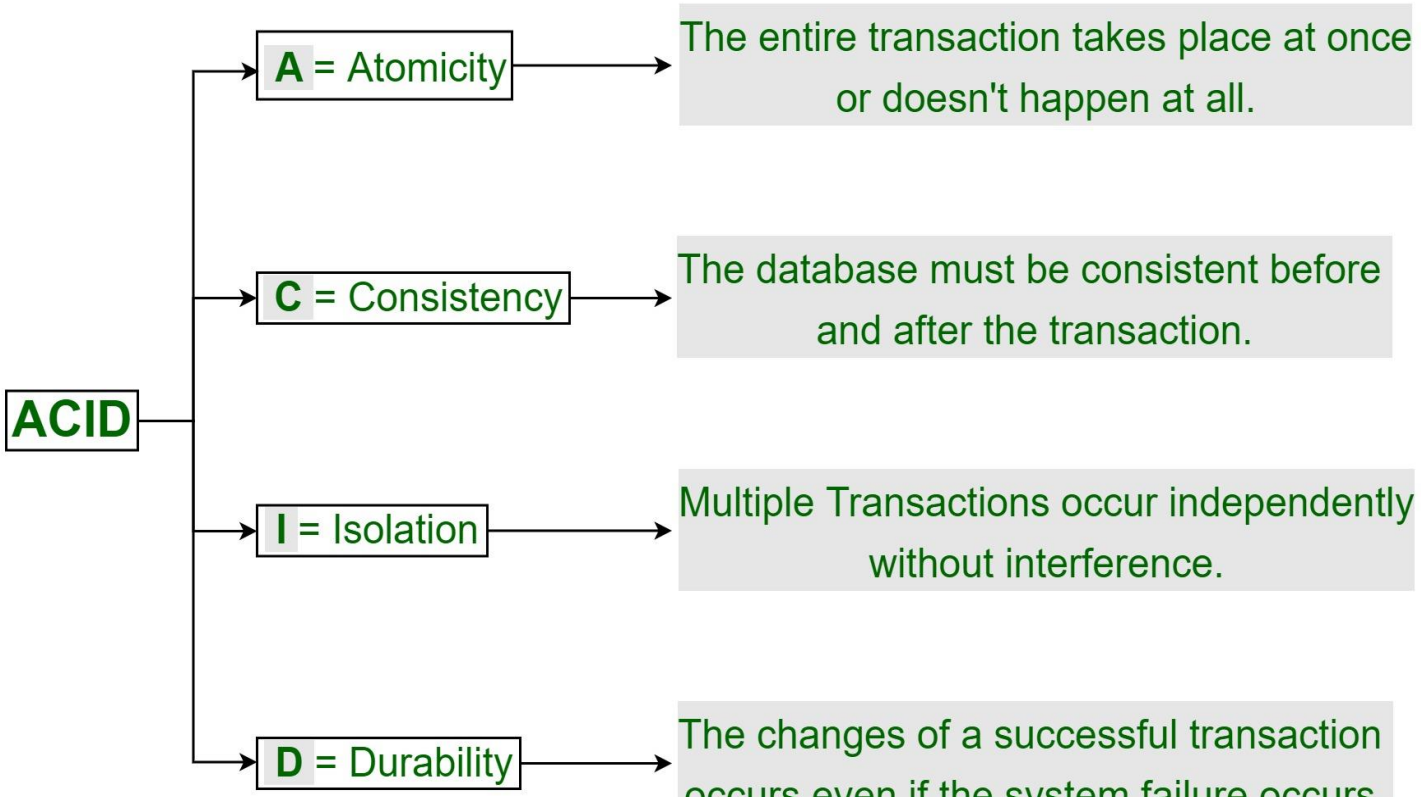
- **Query Dispatcher**
- **Optimization Engines**
- **Classic Query Engine**
- **SQL Query Engine, etc.**

A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.



Uplatz

ACID Properties in DBMS



Atomicity

Uplatz

By this, we mean that either the entire transaction takes place at once or doesn't happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.

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

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

Atomicity is also known as the 'All or nothing rule'.

Consider the following transaction **T** consisting of **T1** and **T2**: Transfer of 100 from account **X** to account **Y**.

If the transaction fails after completion of **T1** but before completion of **T2**.(say, after **write(X)** but before **write(Y)**), then amount has been deducted from **X** but not added to **Y**. This results in an inconsistent database state. Therefore, the transaction must be executed in entirety in order to ensure correctness of database state.

Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

Consistency

This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database. Referring to the example above,

The total amount before and after the transaction must be maintained.

Total before T occurs = 500 + 200 = 700.

Total after T occurs = 400 + 300 = 700.

Therefore, database is consistent. Inconsistency occurs in case T1 completes but T2 fails. As a result T is incomplete.

Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

Isolation

T	T''
Read (X) X: = X*100 Write (X) Read (Y) Y: = Y – 50 Write	Read (X) Read (Y) Z: = X + Y Write (Z)

This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.

Let **X**= 500, **Y** = 500.
Consider two transactions **T** and **T''.**

Suppose **T** has been executed till **Read (Y)** and then **T''** starts. As a result , interleaving of operations takes place due to which **T''** reads correct value of **X** but incorrect value of **Y** and sum computed by
T'': (X+Y = 50, 000+500=50, 500)
is thus not consistent with the sum at end of transaction:
T: (X+Y = 50, 000 + 450 = 50, 450).
This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory.

Uplatz

Durability

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

The **ACID** properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored.

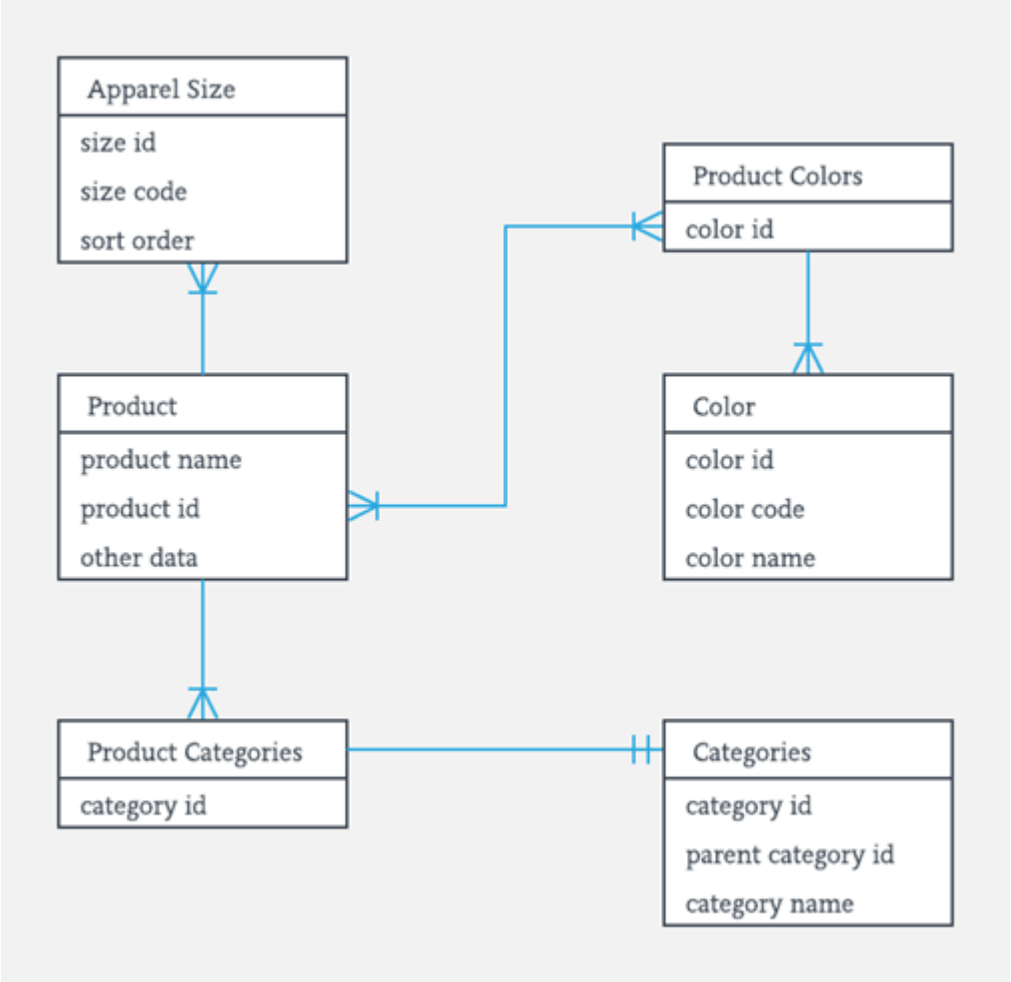
What is the ER Model?

Uplatz

ENTITY RELATIONAL (ER) MODEL is a high-level conceptual data model diagram. ER modeling helps you to analyze data requirements systematically to produce a well-designed database. The Entity-Relation model represents real-world entities and the relationship between them. It is considered a best practice to complete ER modeling before implementing your database.

ER modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

ENTITY-RELATIONSHIP DIAGRAM (ERD) displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.



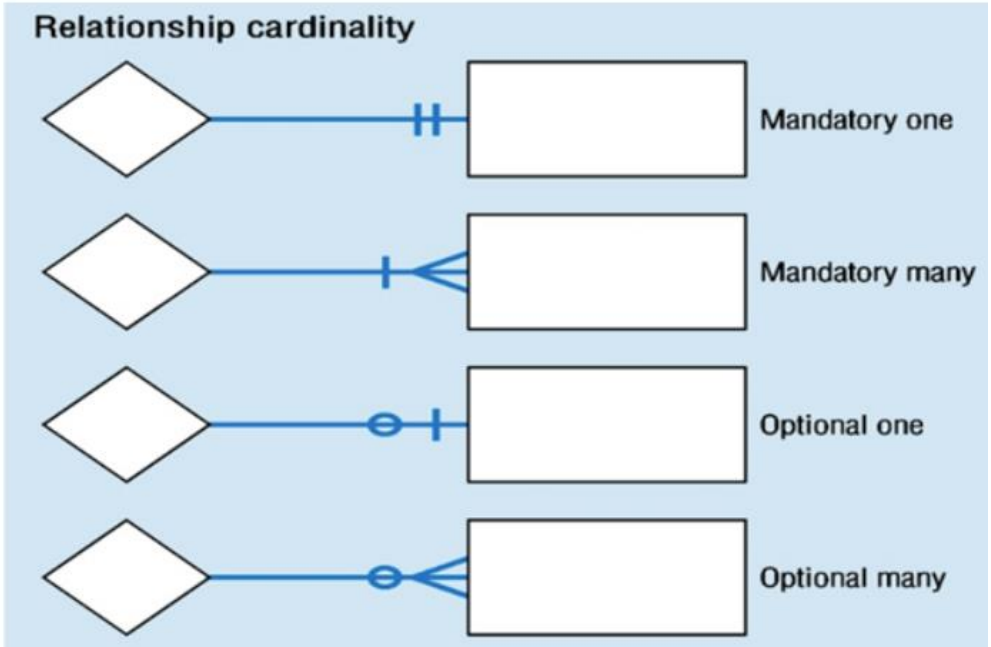
Facts about ER Diagram Model

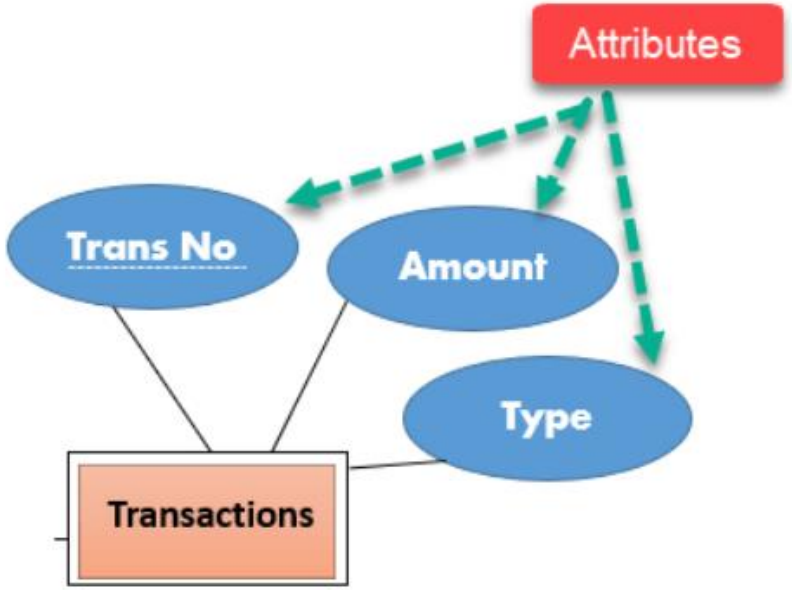
Uplatz

- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modeling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities

Components of the ER Diagram

- Entities
- Attributes
- Relationships





Best practices for developing effective ER Diagram

- Eliminate any redundant entities or relationships
- You need to make sure that all your entities and relationships are properly labeled
- There may be various valid approaches to an ER diagram. You need to make sure that the ER diagram supports all the data you need to store
- You should assure that each entity only appears a single time in the ER diagram
- Name every relationship, entity, and attribute are represented on your diagram
- Never connect relationships to each other
- You should use colors to highlight important portions of the ER diagram

Database Normalization

Database normalization is the process of efficiently organizing data in a database. There are two reasons of this normalization process −

- Eliminating redundant data, for example, storing the same data in more than one table.
- Ensuring data dependencies make sense.

Both these reasons are worthy goals as they reduce the amount of space a database consumes and ensures that data is logically stored. Normalization consists of a series of guidelines that help guide you in creating a good database structure.

Normalization guidelines are divided into normal forms; think of a form as the format or the way a database structure is laid out. The aim of normal forms is to organize the database structure, so that it complies with the rules of first normal form, then second normal form and finally the third normal form.

For our course - MySQL

MySQL is an open source SQL database, which is developed by a Swedish company – MySQL AB. MySQL is pronounced as "my ess-que-ell," in contrast with SQL, pronounced "sequel."

MySQL is supporting many different platforms including Microsoft Windows, the major Linux distributions, UNIX, and Mac OS X.

MySQL has free and paid versions, depending on its usage (non-commercial/commercial) and features. MySQL comes with a very fast, multi-threaded, multi-user and robust SQL database server.

Uplatz

Advantages of MySQL

Features

High Performance.

High Availability.

Scalability and Flexibility Run anything.

Robust Transactional Support.

Web and Data Warehouse Strengths.

Strong Data Protection.

Comprehensive Application Development.

Management Ease.

Open Source Freedom and 24 x 7 Support.

Lowest Total Cost of Ownership.

Thank you