**Database Management System**

**Introduction**

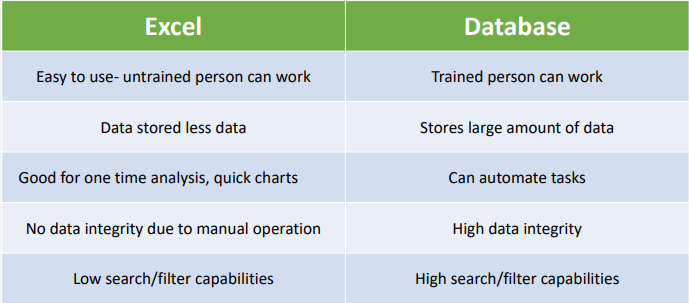
A database is an organized collection of structured data that is stored electronically. It serves as a repository for storing, managing, and retrieving data efficiently. Databases are widely used in various domains, including businesses, research and applications.

Here are some key concepts related to databases:

1. **Data Structure:** Databases organize data into structured formats, typically using tables composed of rows (records) and columns (fields). Each row represents a unique entity or record, while each column represents a specific attribute or piece of data.
2. **Relational Databases:** Relational databases are the most common type of database management system (DBMS). They use structured query language (SQL) to define, manipulate, and query data. Relational databases organize data into multiple tables with relationships established through keys (e.g., primary and foreign keys).
3. **Data Integrity:** Databases enforce data integrity by imposing rules and constraints on the stored data. This ensures that data is accurate, consistent, and reliable. Constraints can include primary key constraints, foreign key constraints, uniqueness constraints, and more.
4. **Querying and Manipulating Data:** Databases provide mechanisms to retrieve, modify, and delete data. SQL is the standard language used to interact with relational databases. It allows users to write queries to retrieve specific data subsets, perform calculations, update records, or delete data based on specific criteria.
5. **Indexing and Optimization:** Indexes in databases improve query performance by allowing faster data retrieval. They are created on specific columns to speed up searches and enhance data access efficiency. Database optimization techniques, such as query optimization and database normalization, are employed to improve performance and minimize redundancy.
6. **Data Security:** Databases implement security measures to protect data from unauthorized access, modification, or deletion. This includes user authentication, access control, encryption, and auditing mechanisms. Database administrators (DBAs) are responsible for managing security measures and ensuring data privacy.
7. **Scalability and Performance:** Databases should be designed and optimized to handle increasing amounts of data and user requests. Scalability refers to the ability of a database to handle growing workloads and user concurrency. Performance tuning techniques, such as optimizing queries, index usage, and database configuration, help enhance database performance.
8. **Database Management Systems (DBMS):** A DBMS is software that facilitates the creation, management, and utilization of databases. Examples of popular DBMSs include MySQL, Oracle, Microsoft SQL Server, PostgreSQL, and SQLite. Each DBMS has its own features, capabilities, and syntax variations.

Databases serve as critical components in various applications and play a fundamental role in data storage and retrieval. They provide a structured and efficient way to manage and analyze vast amounts of data, enabling businesses and organizations to make informed decisions based on their data assets.

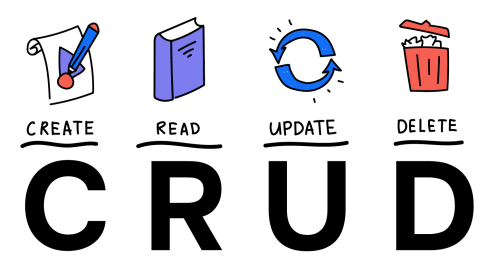
**Excel v/s Database**

****

**SQL**

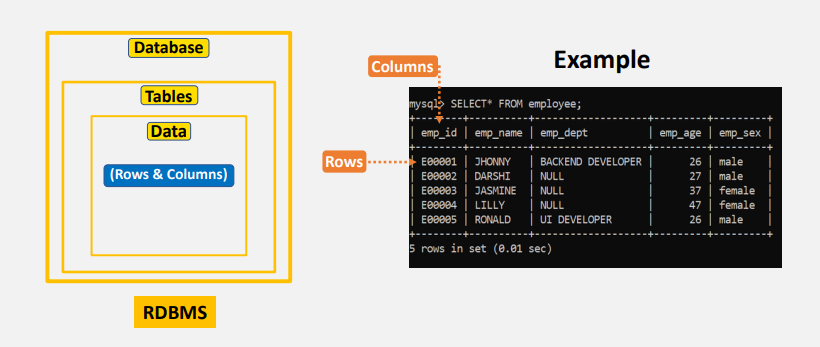
SQL is a programming language used to manage and manipulate relational databases. It provides a standardized way to communicate with and operate on databases, allowing users to create, retrieve, update and delete data.

**SQL Application**

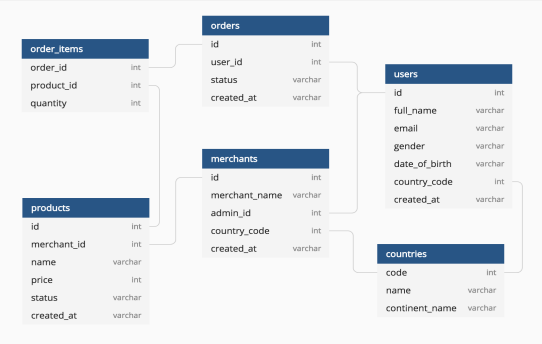


CRUD is an acronym for CREATE, READ(SELECT), UPDATE, and DELETE statements in SQL

**SQL Structure**



**Database Diagram**

****

Here are some key points about SQL:

1. **Purpose:** SQL is primarily use for managing and interacting with structured data stored in relational database management systems (RDBMS). It provides a set of commands and syntax for performing various operations on databases.
2. **Data Definition and Manipulation:** SQL enables the creation, modification, and deletion of database structures and objects. It allows users to define tables, columns, constraints, indexes, and relationships between tables. SQL also provides commands for inserting, updating, and deleting data in the database.
3. **Relational Databases:** SQL is closely associated with relational databases, which organize data into structured tables composed of rows and columns. Tables in a relational database have predefined relationships defined by primary and foreign keys, allowing for efficient data management and retrieval.
4. **Standardized Language:** SQL is an ISO/ANSI standard language, which means it has a standardized syntax and features across different database management systems. Although there may be variations and proprietary extensions between different DBMS implementations, the core SQL language remains consistent.
5. **Querying and Retrieving Data:** One of the key aspects of SQL is its ability to retrieve specific subsets of data from databases. The SELECT statement is use to retrieve data from one or more tables based on specified conditions. SQL allows complex filtering, sorting, aggregation, and joining of data to obtain the desired result set.
6. **Data Manipulation Language (DML) and Data Definition Language (DDL):** SQL is divide into two main categories: Data Manipulation Language (DML) and Data Definition Language (DDL). DML commands like SELECT, INSERT, UPDATE, and DELETE are used to manipulate data within tables. DDL commands like CREATE, ALTER, and DROP are used to define and modify the structure of databases and tables.
7. **Extensions and Advanced Features:** SQL has been extended and expanded over time to include advanced features like subqueries, views, stored procedures, user-defined functions, transactions, and more. These features enhance the power and flexibility of SQL for complex data operations and application development.

SQL is widely used in various industries and domains for managing and analyzing data. It is an essential skill for data professionals, database administrators, and software developers working with relational databases.

**SQL Commands**

There are mainly 3 types of SQL commands:

• DDL (Data Definition Language): create, alter, and drop

• DML (Data Manipulation Language): select, insert, update and delete

• DCL (Data Control Language): grant and revoke permission to users

**Creating Database & Tables**

* Data types
* Primary & Foreign keys
* Constraints
* SQL Commands
* CREATE
* INSERT
* UPDATE
* BACKUP
* DELETE
* ALTER
* DROP, TRUNCATE

**Data Types**

* Data type of a column defines what value the column can store in table.
* Defined while creating tables in database.
* Data types mainly classified into three categories + most used
* String: char, varchar, etc
* Numeric: int, float, bool, etc
* Date and time: date, datetime, etc
* Commonly used data types in SQL:
* int: used for the integer value
* float: used to specify a decimal point number
* bool: used to specify Boolean values true and false
* char: fixed length string that can contain numbers, letters, and special characters
* varchar: variable length string that can contain numbers, letters, and special characters
* date: date format YYYY-MM-DD
* datetime: date & time combination, format is YYYY-MM-DD hh:mm:ss

**Primary Key and Foreign Keys:**

**Primary key (PK):**

* A Primary key is a unique column we set in a table to easily identify and locate data in queries
* A table can have only one primary key, which should be unique and NOT NULL

**Foreign keys (FK):**

* A Foreign key is a column used to link two or more tables together
* A table can have any number of foreign keys, can contain duplicate and NULL values

**Constraints**

* Constraints are used to specify rules for data in a table
* This ensures the accuracy and reliability of the data in the table
* Constraints can be specified when the table is created with the CREATE TABLE statement, or,
* after the table is created with the ALTER TABLE statement
* Syntax

CREATE TABLE table\_name (

column1 datatype constraint,

column2 datatype constraint,

column3 datatype constraint,

....

);

Commonly used constraints in SQL:

* NOT NULL - Ensures that a column cannot have a NULL value
* UNIQUE - Ensures that all values in a column are different
* PRIMARY KEY - A combination of a NOT NULL and UNIQUE
* FOREIGN KEY - Prevents actions that would destroy links between tables (used to link multiple tables together)
* CHECK - Ensures that the values in a column satisfies a specific condition
* DEFAULT - Sets a default value for a column if no value is specified
* CREATE INDEX - Used to create and retrieve data from the database very quickly

1. **Database Creation:** Before you can start working with SQL, you need a database to work with. Most database systems have their own management tools, such as MySQL Workbench for MySQL or phpmyadmin, which provide an interface to create and manage databases. Install and set up the desired database system of your choice.

**CREATE DATABASE** database\_name;

1. **Creating a Table:** Once you have a database, you can create tables to store your data. A table consists of columns that define the structure of the data. For example, to create a table named "Employees" with columns for "ID," "Name," and "Age" in MySQL, you can use the following SQL statement:

CREATE TABLE table\_name (

Column\_name1 data\_type1 constraint1,

Column\_name2 data\_type2 constraint2,

Column\_name3 data\_type3 constraint3

);

1. **Inserting Data:** After creating a table, you can insert data into it using the INSERT statement. Each row of data corresponds to a record in the table. For example, to insert a new employee with an ID of 1, name of "John Doe," and age of 30, you can use the following SQL statement:

Syntax:

INSERT INTO table\_name(column1, column2,…)

VALUES(value1,value2,……)

Eg:

INSERT INTO Employees (ID, Name, Age)

VALUES (1, 'John Doe', 30);

1. **Retrieving Data with SELECT:** The SELECT statement allows you to retrieve data from one or more tables. To retrieve all records from the "Employees" table, you can use the following SQL statement:

SELECT column1, column2 FROM table\_name;

\* = for all record

This will return all columns and rows from the table.

1. **Filtering Data with WHERE:** You can use the WHERE clause to filter the results based on specific conditions. For example, to retrieve employees older than 25, you can modify the SELECT statement as follows:

SELECT column\_name FROM table\_name WHERE condition;

1. **Updating Data with UPDATE:** The UPDATE statement is used to modify existing data in a table. For example, to update the age of the employee with ID 1 to 35, you can use the following SQL statement:

UPDATE table\_name SET column\_name = value WHERE condition;

1. **Deleting Data with DELETE:** The DELETE statement is used to remove records from a table. For example, to delete the employee with ID 1, you can use the following SQL statement:

DELETE FROM table\_name WHERE condition;

1. **Advanced Queries:** SQL offers various advanced querying techniques, such as joins, aggregations, and subqueries. Joins allow you to combine data from multiple tables based on common columns, aggregations help summarize data using functions like COUNT, SUM, AVG, etc., and subqueries enable you to nest queries within other queries. Exploring these concepts will enhance your SQL skills.

This tutorial provides a basic overview of SQL. However, SQL is a vast language with many additional features and capabilities. You can further explore topics such as indexing, constraints, transactions, stored procedures, and more to deepen your understanding. Online resources, tutorials, and interactive platforms like SQLZoo, W3Schools, or Mode Analytics can provide comprehensive SQL tutorials and practice exercises to help you further enhance your skills.

Advanced queries in SQL go beyond basic data retrieval and involve more complex operations. Here are some common advanced querying techniques:

1. **Joins:** Joins allow you to combine data from multiple tables based on common columns. There are different types of joins, including:

* INNER JOIN: Returns only the matching records from both tables.
* LEFT JOIN: Returns all records from the left (or "left-hand") table and the matching records from the right table.
* RIGHT JOIN: Returns all records from the right (or "right-hand") table and the matching records from the left table.
* FULL JOIN: Returns all records when there is a match in either the left or right table.

For example, to retrieve data from two tables, "Orders" and "Customers," based on the customer ID, you can use an INNER JOIN like this:

SELECT table\_name1.columns, table\_name2.columns ////- \* can use

FROM table\_name1

INNER JOIN table\_name2

ON table\_name1.columns = table\_name2.columns;

1. **Subqueries:** Subqueries, also known as nested queries, allow you to include one query inside another query. Subqueries can be used in various ways, such as:

* Filtering: Use a subquery in the WHERE clause to filter the result set based on values from another table or query.
* Calculation: Use a subquery to perform calculations within a query, such as finding the maximum or average value.
* Inclusion: Use a subquery in the SELECT statement to retrieve a value that depends on the result of another query.

For example, to find customers who have placed more orders than the average number of orders, you can use a subquery like this:

SELECT CustomerID, CustomerName

FROM Customers

WHERE CustomerID IN (

SELECT CustomerID

FROM Orders

GROUP BY CustomerID

HAVING COUNT(OrderID) > (

SELECT AVG(OrderCount)

FROM (

SELECT CustomerID, COUNT(OrderID) AS OrderCount

FROM Orders

GROUP BY CustomerID

) AS Subquery

)

);

1. **Aggregations and Grouping:** SQL provides aggregate functions like COUNT, SUM, AVG, MIN, and MAX to perform calculations on groups of rows. Combined with the GROUP BY clause, you can group data based on one or more columns. For example, to calculate the total sales amount for each product category, you can use the following query:

SELECT Category, SUM(SalesAmount) AS TotalSales

FROM Products

GROUP BY Category;

1. **Window Functions:** Window functions allow you to perform calculations across a set of rows related to the current row. These functions operate on a "window" of rows defined by a specific ordering and partitioning. Window functions include ROW\_NUMBER, RANK, DENSE\_RANK, LAG, LEAD, and more. They are useful for tasks such as ranking, calculating moving averages, or finding the difference between consecutive rows.

For example, to calculate the rank of each employee based on their salary within their department, you can use the RANK() window function:

SELECT EmployeeID, EmployeeName, Department, Salary, RANK() OVER (PARTITION BY Department ORDER BY Salary DESC) AS SalaryRank

FROM Employees;

These are just a few examples of advanced SQL queries. There are many more features and techniques you can explore, such as common table expressions (CTEs), recursive queries, conditional expressions (CASE statements), and working with dates and times. By mastering these advanced querying techniques, you'll have a solid foundation for complex data retrieval and analysis tasks.