

SQL Basics and the Query Life Cycle

- **The SQL Query Life Cycle: How Your Queries Are Processed**

- **Parsing:** The database server first checks if your SQL query is syntactically correct. Think of it like a spell-checker for code. If there are typos or grammatical errors in your SQL, this is where they'll be caught.
- **Binding:** The server then checks if the objects you're referring to (tables, columns) actually exist and if you have the necessary permissions to access them. It also resolves object names to their internal identifiers.
- **Optimization:** The database's optimizer analyzes the query and tries to find the most efficient way to execute it. There might be multiple ways to get the same result, and the optimizer chooses the one that will likely be the fastest (e.g., deciding whether to use an index).
- **Execution Plan Generation:** Based on the optimized approach, the server creates a detailed plan outlining the steps it will take to execute the query (e.g., which tables to access, in what order, what kind of operations to perform).
- **Execution:** This is where the actual work happens. The database engine follows the execution plan, retrieving and manipulating the data as required by the query.
- **Fetching Results:** Once the data has been processed, the server prepares the result set to be sent back to the user or application that issued the query.
- **Transaction Management (if applicable):** If the query is part of a transaction (a sequence of operations treated as a single unit), the database ensures that the ACID properties are maintained throughout the process.
- **Logging & Auditing (if enabled):** The server might log the query and any changes made for recovery purposes or for auditing who accessed what data.
- **Result Return:** Finally, the database sends the result set back to the client.

○

- **Basic SQL Commands (from your examples):**

- **Data Definition Language (DDL):** Commands used to define the structure of the database.
 - `CREATE DATABASE cohort;` - Creates a new database named 'cohort'.
 - `SHOW DATABASES;` - Lists all the databases on the server.
 - `USE cohort;` - Selects 'cohort' as the active database.
 - `CREATE TABLE Students (...);` - Creates a new table named 'Students' with specified columns and constraints.
 - `ALTER TABLE Students ADD COLUMN ...;` - Modifies the structure of the 'Students' table by adding a column.
 - `DROP TABLE Students;` - Deletes the 'Students' table entirely.
- **Data Manipulation Language (DML):** Commands used to manipulate the data within the tables.
 - `INSERT INTO Students (...) VALUES (...);` - Adds new rows of data into the 'Students' table.
 - `UPDATE Students SET ... WHERE ...;` - Modifies existing data in the 'Students' table.
 - `DELETE FROM Students WHERE ...;` - Removes rows from the 'Students' table.
- **Data Query Language (DQL):** Primarily the `SELECT` statement used to retrieve data.
 - `SELECT * FROM cohort.Students;` - Retrieves all columns (*) and all rows from the 'Students' table in the 'cohort' database.
 - `SELECT COUNT(*) FROM cohort.Students;` - Counts the total number of rows in the 'Students' table.
- **Managing Databases and Tables:**
 - `SHOW TABLES;` - Lists all tables in the currently selected database.

Introduction to SQL

What is SQL?

- Structured Query Language.
- The standard language for managing and manipulating relational databases.
- Used to create, access, and modify data in a relational database management system (RDBMS).

SQL Statements

- **DDL (Data Definition Language):**
 - CREATE: Creates a new database, table, or other object.
 - Example: CREATE TABLE Students (StudentID INT, Name VARCHAR(255), Age INT);
 - ALTER: Modifies an existing object.
 - Example: ALTER TABLE Students ADD COLUMN Email VARCHAR(255);
 - DROP: Deletes an existing object.
 - Example: DROP TABLE Students;
- **DML (Data Manipulation Language):**
 - SELECT: Retrieves data from one or more tables.
 - Example: SELECT * FROM Students;
 - INSERT: Inserts new data into a table.
 - Example: INSERT INTO Students (StudentID, Name, Age) VALUES (1, 'John Doe', 20);
 - UPDATE: Modifies existing data in a table.
 - Example: UPDATE Students SET Age = 21 WHERE StudentID = 1;
 - DELETE: Deletes data from a table.
 - Example: DELETE FROM Students WHERE StudentID = 1;
- **DCL (Data Control Language):**
 - GRANT: Gives users privileges.
 - Example: GRANT SELECT ON Students TO user1;
 - REVOKE: Takes back privileges.
 - Example: REVOKE SELECT ON Students FROM user1;
- **TCL (Transaction Control Language):**
 - COMMIT: Saves all transactions to the database.
 - ROLLBACK: Undoes all transactions to the last COMMIT.
 - SAVEPOINT: Sets a point to which you can roll back.

Basic SQL Queries

- Retrieve all columns and rows from a table:
SELECT * FROM table_name;
- Retrieve specific columns from a table:
SELECT column1, column2 FROM table_name;
- Filter rows using a WHERE clause:
SELECT * FROM table_name WHERE condition;
- Sort the result set using ORDER BY:
SELECT * FROM table_name ORDER BY column1 ASC, column2 DESC;
- Aggregate data using GROUP BY and aggregate functions (e.g., COUNT, SUM, AVG, MIN, MAX):
SELECT column1, COUNT(*) FROM table_name GROUP BY column1;

Sample Question Answer

```
CREATE TABLE Students (
  rollId INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  dept TEXT NOT NULL
);
```

```
ALTER TABLE Students
ADD COLUMN city VARCHAR(50);
```

```
ALTER TABLE Students
ADD COLUMN percentage_in_12th DECIMAL(5, 2);
```

```
ALTER TABLE Students
MODIFY COLUMN dept VARCHAR(100);
```

```
-- Insert dummy values
INSERT INTO Students (rollId, name, dept, city, percentage_in_12th) VALUES
(0001, 'kunal ', 'CSE', 'Delhi', 92.10),
(0002, 'aayush ', 'EE', 'Mumbai', 89.60),
(0003, 'alekya ', 'EE', 'Bangalore', 94.30),
(0004, 'anil ', 'CSE', 'Chennai', 87.80),
(0005, 'Upasna ', 'Mech', 'Kolkata', 90.25);
```

```
-- Show the students table
```

```
SELECT * from Students;
```

```
-- Filter people having percentage greater than 80
```

```
SELECT *  
FROM Students  
WHERE percentage_in_12th > 80;
```

```
-- Tell the top 3 people
```

```
SELECT *  
FROM Students  
ORDER BY percentage_in_12th DESC  
LIMIT 3;
```

```
-- Get the average percentage of all people
```

```
SELECT AVG(percentage_in_12th)  
FROM Students;
```

Links:

https://www.w3schools.com/sql/sql_having.asp