SQL

Important Topics:

1. What does SQL stand for?

- SQL stands for **Structured Query Language**.
- The original name of the language was SEQUEL, created for the IBM System R research database in 1970, but due to copyright issues, they changed the name to SQL.

2. What is Data?

- Data is distinct pieces of information, which can be facts, figures, or details that are stored in or used by a computer.

3. What is a Database?

- A database is a well-organized collection of data that is stored in an electronic format.
- SQL database is an electronic system that allows users to easily access, manipulate, and update the data.

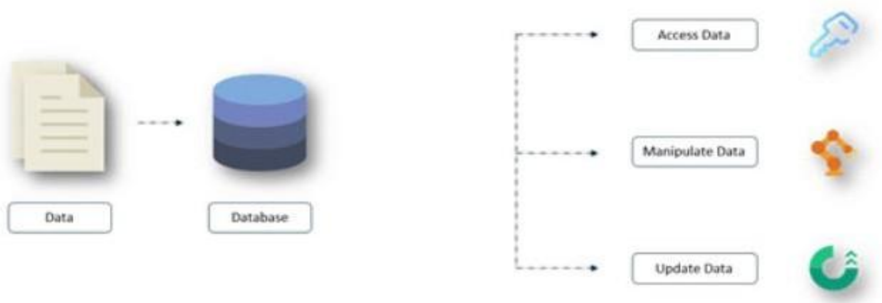


*fig.*1.Database

4. What is a Database Management System?

- It is a system software used for creating data in a systematic way and managing databases.
- Almost all modern databases are managed by a Database Management System (DBMS).

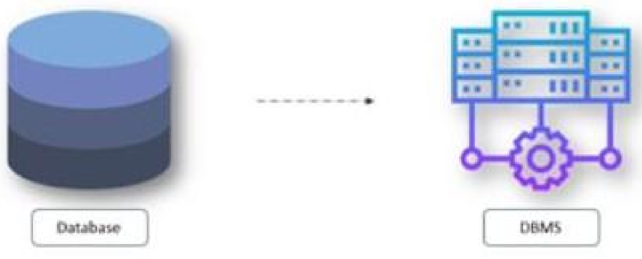


fig.2.Database Management System

➢ **Example:**

Consider a School SQL Database which has a record of the present students and the previously studied students in the Student Details table. Similarly, it may contain Faculty Details, Management Details, Staff Details, and many more depending on the school's requirement. As the data is in huge amounts, to manage it we need a database management system.

5. How Does SQL Database Manage Data?

➢ DBMS provides, for both users and programmers, a fundamental way to create, retrieve, update, and manage data.

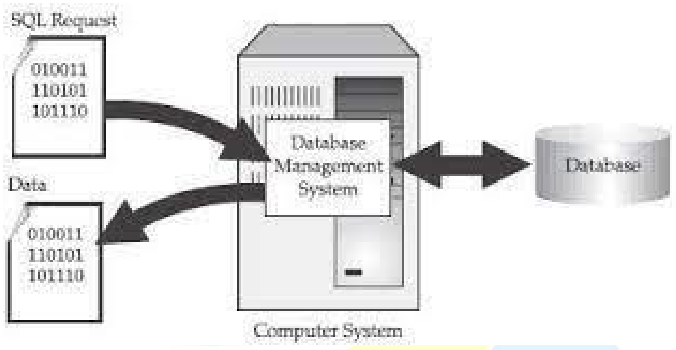


fig.3.SQL manage data

6. Types of Databases in SQL?

➢ SQL mainly consists of two databases.

➢ They are, Relational (SQL) and Non Relational (NoSQL).

7. What is SQL?

➢ SQL is a programming language used to manage and interact with databases.

➢ It allows you to **retrieve, insert, update**, and **delete** data in a structured way.

8. What is RDBMS?

➢ RDBMS stands for **Relational Database Management System.**

➢ It's a software system that organizes data in tables with rows and columns, uses SQL for data operations, and maintains data integrity through relationships between tables.

➢ Examples include MySQL, PostgreSQL, Oracle, SQL Server, and SQLite.

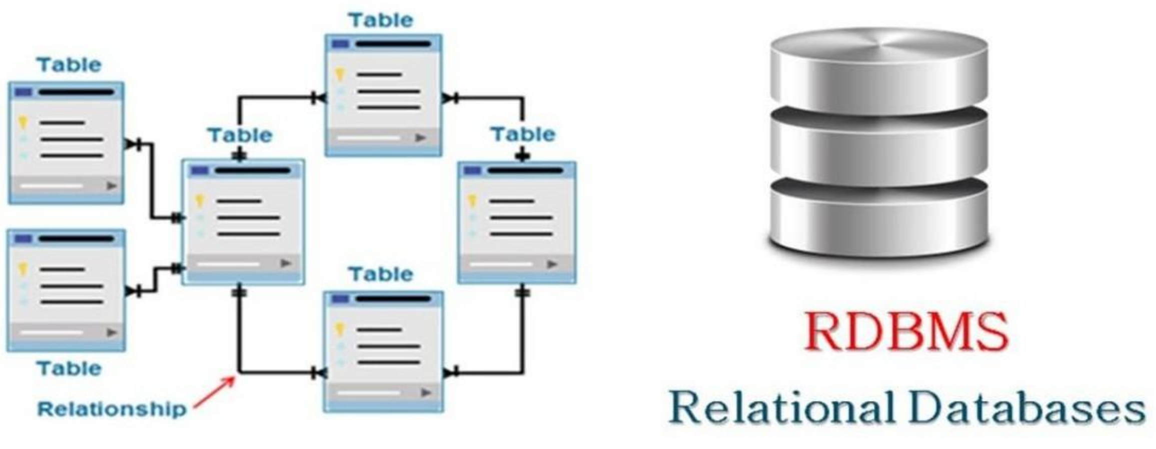


fig.4.RDBMS

9. SQL Process?

The SQL process involves the following steps:

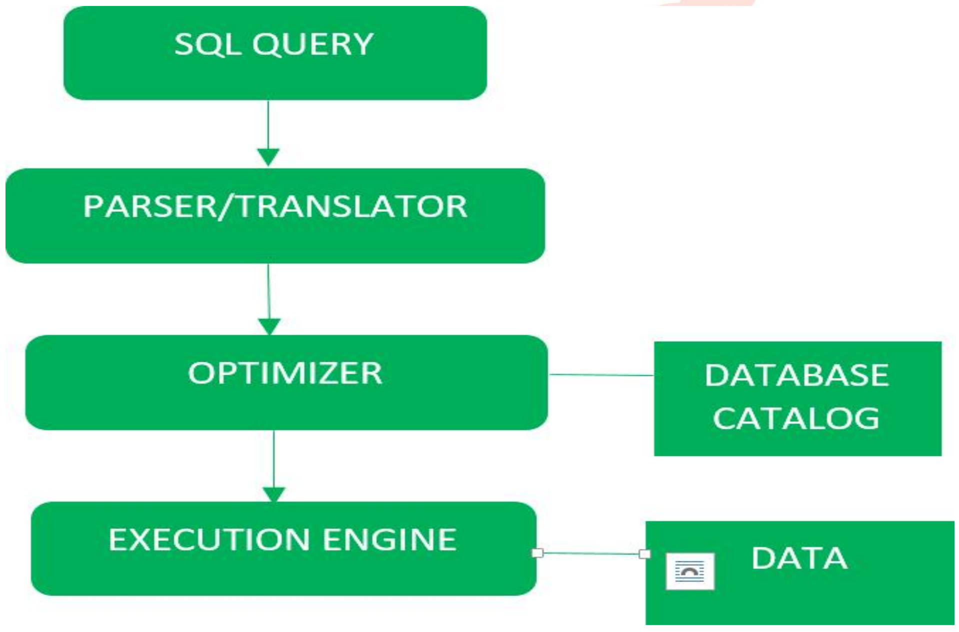


fig.5.SQL Process

- Sending an SQL query to the database.
- Parsing the query to check its validity.
- Optimizing the query for efficient execution.
- Executing the query to retrieve or modify data.
- Processing the results (if applicable).
- Managing transactions and connections.

10. Why SQL?

➢ Since SQL can work with any database, it is the most widely used language for database access. The databases in which you interact are programs that allow clients to logically store and manage information. With SQL, we can have the following advantages.

- SQL offers data access to users in relational databases.
- Users may use this feature to define the data.
- Identifying and modifying the data in a database is easy with SQL
- We can create, delete, alter the data in the database anytime.
- It Allows SQL modules, libraries, and pre-compilers to be embedded within other languages.
- Database views, stored procedures, and functions can be built with SQL

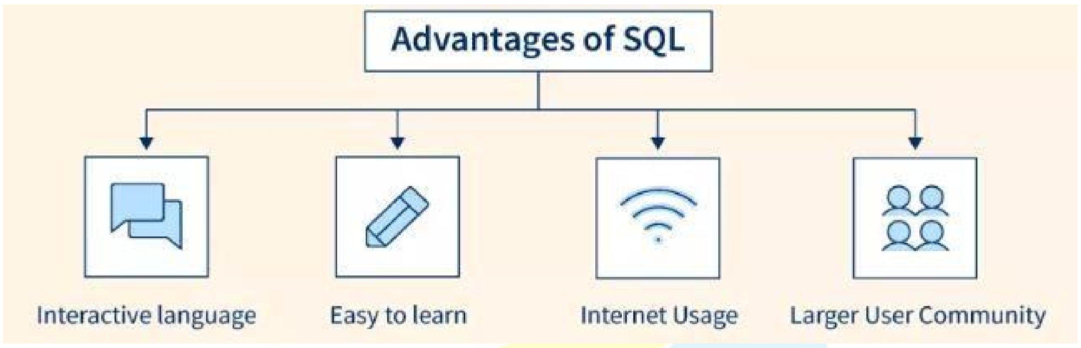


fig.6.Advantages of SQL

11. Features of SQL?

➢ SQL (Structured Query Language) has several key features that make it a powerful and widely-used language for managing and interacting with databases:

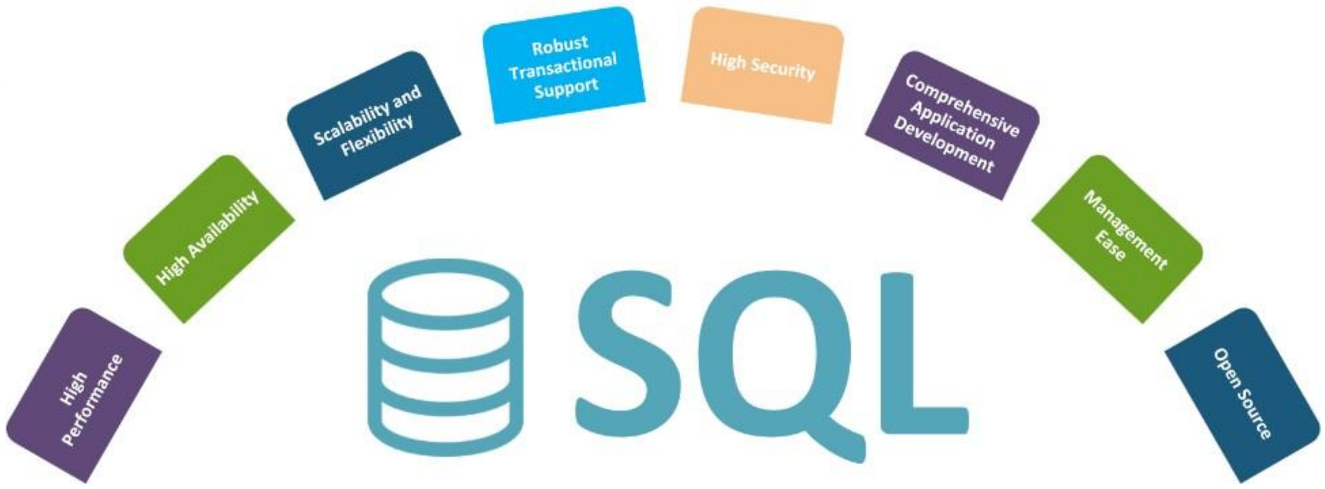

High Performance
High Availability
Scalability and Flexibility
Robust Transactional Support
High Security
Comprehensive Application Development
Management Ease
Open Source
SQL

fig.7.SQL Features & Characteristics

➢ **High Performance:**

SQL provides high-performance programming capability for highly transactional, heavy workload, and high usage database systems.

- **High Availability:**

1) SQL databases are made highly available by setting up backups, copies, and duplicates of the data across multiple servers.
2) If one server fails, another can take over, ensuring the database stays accessible even during failures or disasters. This way, the data remains available and the system continues to work smoothly without interruptions.

- **Scalability and Flexibility:**

It is very easy to create new tables and previously created or not used tables can be dropped or deleted in a database.

- **Robust Transactional Support:**

SQL programming can handle large records and manage numerous transactions.

- **High Security:**

It is very easy to provide permissions on tables, procedures, and views, hence, SQL gives security to your data.

- **Comprehensive Application Development:**

SQL is used by many programmers to program apps to access a database. No matter what the size of an organization, SQL works for every small or large organization.

- **Management Ease:**

SQL is used in almost every Relational Database Management System. “Select“, “Create”, “Insert”, “Drop”, “Update”, and “Delete” are the standard and common SQL commands that help us to manage large amounts of data from a database very quickly and efficiently.

- **Open Source:**

SQL is an open-source programming language for building relational database management system.

12. What are the different categories of SQL commands?

SQL commands are traditionally divided into four categories:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)
- Transaction Control Language (TCL)

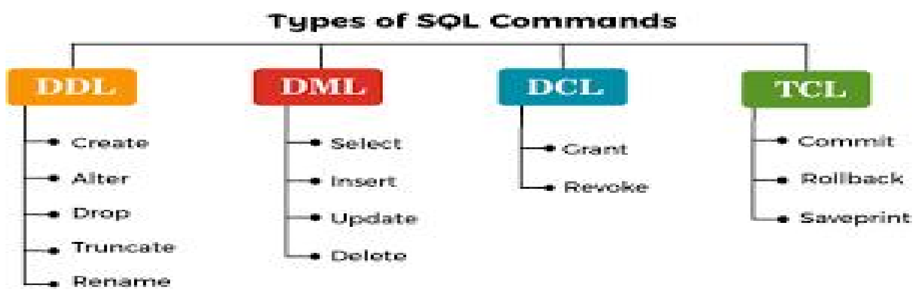


fig.8.Types of SQL Commands

- **DDL:** Used for defining and managing database objects like tables, indexes, etc.
- **DML:** Used for querying and modifying data within the database.
- **DCL:** Used for managing permissions and access to the database.
- **TCL:** Essential for ensuring the integrity and consistency of data in the database when multiple SQL statements need to be executed together as a unit.

13. What are the different types of DDL Commands?

- It allows users to create, modify, and delete database objects such as tables, indexes, views, and schema.
- Common DDL commands include:

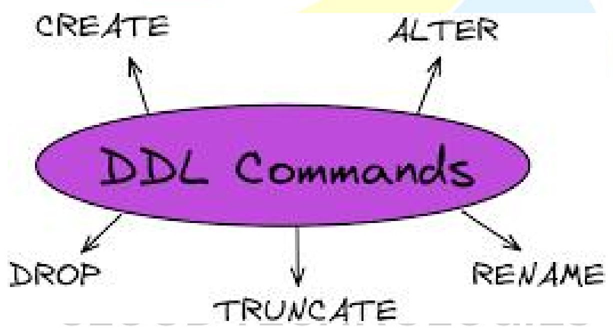


fig.9.DDL Commands

1) **CREATE:** Used to create new database objects like tables, indexes, and views.
2) **ALTER:** Used to modify the structure of existing database objects.
3) **DROP:** Used to delete database objects and remove them from the database.
4) **TRUNCATE:** Used to remove all data from a table, but not the table structure itself.
5) **RENAME:** Different database management systems (DBMS) may have their own specific ways of renaming database objects, but there is no single standardized RENAME command in SQL.

14. What are the different types of DML Commands?

- These commands are used to insert, retrieve, modify, and delete data within database tables.

- DML commands are essential for performing day-to-day operations on data within a database and are an integral part of SQL for interacting with and managing data.
- Common DML commands include:

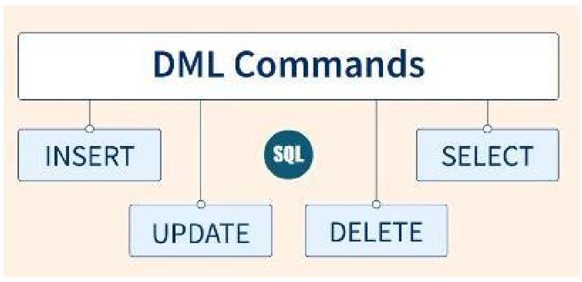


*fig.*10.DML Commands

1) **SELECT:** Used to retrieve data from one or more database tables based on specified conditions.
2) **INSERT:** Used to add new records (rows) into a table.
3) **UPDATE:** Used to modify existing records in a table.
4) **DELETE:** Used to remove records from a table.

15. What are the different types of DCL Commands?

- These commands grant or revoke privileges to users and control their level of access to specific database objects or operations.
- The main DCL commands are:

GRANT
GRANT
REVOKE
REVOKE

*fig.*11.DCL Commands

1) **GRANT:** Used to give specific privileges or permissions to users or roles. For example, granting SELECT permission on a table to a user.
2) **REVOKE:** Used to take away previously granted privileges or permissions from users or roles. For example, revoking the previously granted SELECT permission.

- DCL commands are essential for ensuring the security and integrity of the database by controlling who can access, modify, or view specific data within the database.
- Database administrators use DCL commands to manage user access and privileges to maintain data confidentiality and prevent unauthorized actions.

16. What are the different types of TCL Commands?

- ➢ A transaction is a sequence of one or more SQL statements that need to be executed as a single unit of work.
- ➢ The main TCL commands are:

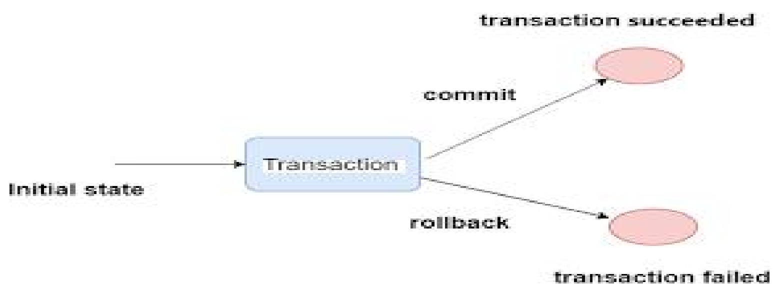


*fig.*12.TCL Commands

1) **COMMIT:** This command is used to permanently save the changes made during a transaction. Once a COMMIT is executed, the changes become permanent and cannot be rolled back.
2) **ROLLBACK:** This command is used to undo the changes made during a transaction and restore the data to its original state before the transaction began. It effectively cancels the transaction.
3) **SAVEPOINT:** This command is used to set a marker within a transaction. If a part of the transaction encounters an error or needs to be rolled back, you can roll back to the savepoint without undoing the entire transaction.

17. Explain the differences between DDL, DML, and DCL in SQL?

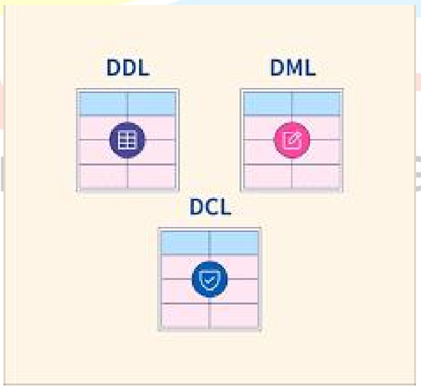


*fig.*13.Differences

1) **DDL:** Used to define database structure (CREATE, ALTER, DROP).
2) **DML:** Used to interact with data (SELECT, INSERT, UPDATE, DELETE).
3) **DCL:** Used to control database access (GRANT, REVOKE).

18. What is a database schema?

- ➢ A database schema is a logical container for database objects like tables, views, and indexes. It defines the structure and organization of a database.

19. What is a table in SQL?

➢ A table in SQL is a collection of data organized in rows and columns. Each column represents an attribute, and each row represents a record.

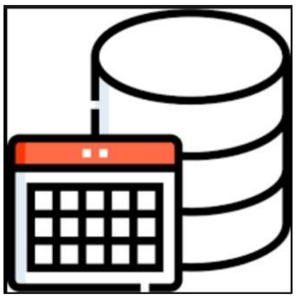


fig.14.Representation of Table & Database

20. How do you create a new database in SQL?

➢ To create a new database, you use the 'CREATE DATABASE' command followed by the database name.

Database in SQL
CREATE DATABASE db_name;
DROP DATABASE db _name;
SQL
col 2= val 2 WHERE condition;
SELECT * FROM tb1;

fig.15.Database in SQL

➢ **Syntax:** CREATE DATABASE database_name;

21. Which Command is used to check the databases we have?

➢ **Show Databases;** This command is used to display all the databases.

22. How do we delete database in SQL?

➢ To delete an entire database, we use the 'DROP DATABASE' command followed by the database name.

*fig.*16.Delete database

➢ **Syntax:** DROP DATABASE database_name;

23. What are the basic data types in SQL?

➢ Common data types include INTEGER,CHAR,VARCHAR, DATE, FLOAT, and BOOLEAN.

Data Type	Description	Storage
Bit	Integer value 0, 1 or NULL	
Small Int	Values from -32,768 and 32,767	2 bytes
Integer	Values from -2,147,483,648 and 2,147,483,647	4 bytes
Big Int	Values from -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
Decimal(p,s)	Fixed Precision and Scale Numbers Allows numbers from -10^38 +1 to 10^38 –1 The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default valueis 0	5-17 bytes
Float(n)	Floating precision number data from -1.79E + 308 to 1.79E + 308. The n parameter indicates whether the field should hold 4 or 8 bytes. Float (24) holds a 4-byte field and float (53) holds an 8-byte field. Default value of n is 53.	4 or 8bytes
Char(n)	Fixed width character string	Defined size
varchar(n)	Variable width character string	2 bytes + number of characters
Boolean	When you create a table with a boolean data type, MySQL outputs data as 0, if false, and 1, if true.	2 bytes
Datetime	Allows from January 1, 1753 to December 31, 9999	8 bytes
Timestamp	timestamp value is based upon an internal clock and does not correspond to real time	8 bytes
Date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
Time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes

➢ Examples of each DataType is as follows:

1) **INTEGER:** CREATE TABLE my_table (id INT);

2) **DECIMAL:** CREATE TABLE my_table (price DECIMAL(10, 2));

3) **FLOAT:** CREATE TABLE my_table (weight FLOAT);
4) **CHAR:** CREATE TABLE my_table (code CHAR(6));
5) **VARCHAR:** CREATE TABLE my_table (name VARCHAR(50));
6) **BOOLEAN:** CREATE TABLE my_table (is_active BOOLEAN);
7) **DATE:** CREATE TABLE my_table (birth_date DATE);
8) **DATETIME:** CREATE TABLE my_table (created_at DATETIME);
9) **TIMESTAMP:** CREATE TABLE my_table (updated_at TIMESTAMP);
10) **TIME:** CREATE TABLE my_table (start_time TIME);

24. Why we use VARCHAR instead of CHAR Datatype?

➢ We use VARCHAR instead of CHAR datatype because VARCHAR is more storage-efficient for variable-length data, allows flexibility in data length, and does not require padding, leading to better performance and data integrity.

25. Difference between VARCHAR and CHAR?

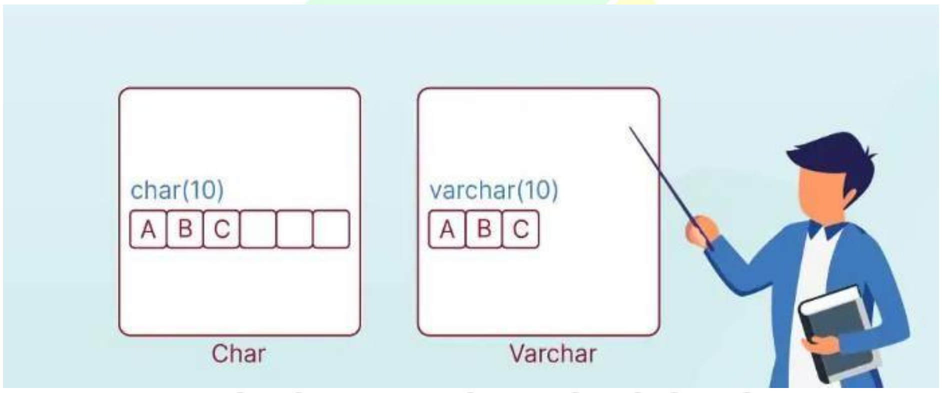


*fig.*17.CHAR Vs VARCHAR

➢ **VARCHAR** is a variable-length character data type that stores strings of varying lengths, using only the necessary storage for each value.

➢ **CHAR** is a fixed-length character data type that always allocates a specific amount of storage regardless of the actual data length, potentially leading to wasted space for shorter values.

26. How we use the date format in MySQL?

➢ MySQL typically uses the "**YYYY-MM-DD**" format for dates.

*fig.*18.Date Format in MySQL

- **Syntax:**

```
INSERT INTO table_name (date_column) VALUES ('2023-07-24');
```

27. How do you CREATE a new table in an existing database?

- To create a new table, you use the 'CREATE TABLE' command with the table name and column definitions.
- **Syntax:**

```
CREATE  TABLE  table_name
    (       column1        datatype
    constraints, column2  datatype
    constraints,
    ...
);
```

- **Example Query:**

```
CREATE TABLE employees
    ( employee_id INT PRIMARY
    KEY,first_name VARCHAR(50),
    last_name VARCHAR(50),
    department VARCHAR(50)
);
```

28. How do you INSERT data into a table?

- To insert data into a table, you use the 'INSERT INTO' command followed by the table name and the values to be inserted.
- **Syntax:**

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

- **Example:**

```
INSERT INTO
        employees
        (employee_id, first_name, last_name, department)
VALUES
        (1, 'John', 'Doe', 'HR');
```

29. How to SELECT data from table?

- The 'SELECT' statement is used in SQL to retrieve data from a database table.
- It allows you to specify which columns you want to retrieve and apply conditions to filter the data based on certain criteria.
- **Syntax:**

```
SELECT column1, column2, column3, ...
FROM table_name;
```

- **Example:**

```
SELECT first_name, last_name, department
FROM employees;
(or)
SELECT * FROM employees; //* indicates all columns
```

- **Example Output:**

```
| first_name | last_name | department |
|------------|-----------|------------|
| John       | Doe       | HR         |
| Jane       | Smith     | HR         |
| Jim        | Johnson   | HR         |
```

30. How do you UPDATE existing data in a table?

- To update data in a table, you use the 'UPDATE' command with the table name, SET clause, and the condition to identify the rows to be updated.
- **Syntax:**

```
UPDATE table_name
```

```
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

- **Example:**

```
UPDATE employees
SET department = 'Finance'
WHERE employee_id = 2;
```

- **Example Output:**

```
| first_name | last_name | department |
|------------|-----------|------------|
| John       | Doe       | HR         |
| Jane       | Smith     | Finance    |
| Jim        | Johnson   | HR         |
```

31. How do you DELETE data from a table?

- To delete data from a table, you use the 'DELETE FROM' command with the table name and a condition to specify the rows to be deleted.
- **Syntax:**

```
DELETE FROM table_name
WHERE condition;
```

- **Example:**

```
DELETE FROM employees
WHERE employee_id = 2;
```

32. Explain the WHERE clause and how it is used in SQL queries.

- The WHERE clause filters rows based on a specified condition in SQL queries.
- **Syntax:**

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

33. How do you Modify data in a table?

- The 'ALTER' statement in SQL is used to modify the structure of an existing database object.
- **Syntax:**

```
ALTER TABLE table_name ADD column_name datatype;
```

34. How do we change the datatype of a particular column after creation of a table?

- The 'ALTER' statement in SQL is used to modify the structure of an existing database object.
- **Syntax:**

```
ALTER TABLE table_name MODIFY COLUMN column_name datatype;
```

35. How DROP Command is used?

- The 'DROP' statement in SQL is used to remove existing database objects, such as tables, views, indexes, or databases themselves.
- The DROP statement is a powerful and potentially destructive command, as it permanently removes the specified database objects from the database.
- **Syntax:**

```
DROP TABLE table_name;
```

36. How TRUNCATE is used?

- The TRUNCATE statement in SQL is used to quickly remove all rows from a table, effectively deleting all data in the table, while keeping the **table's structure intact.**
- **Syntax:**

```
TRUNCATE TABLE table_name;
```

37. How we can remove all rows from table using DELETE Command?

- **Syntax:**

```
DELETE FROM table_name;
```

38. Difference Between TRUNCATE,DELETE & DROP Commands?

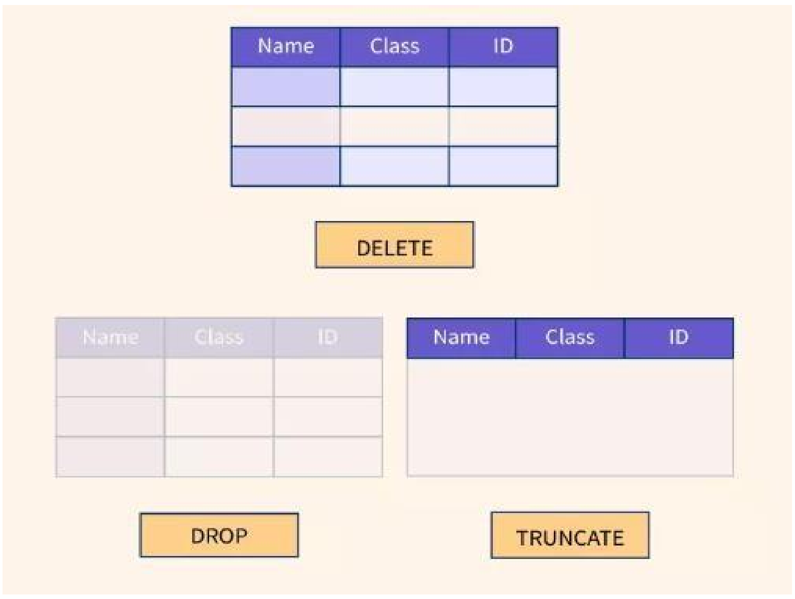


*fig.*19.DELETE,DROP,TRUNCATE

- **TRUNCATE:** Quickly removes all rows from a table, keeping the table structure. Faster for large tables. Cannot be rolled back.
- **DELETE:** Removes specific rows from a table based on conditions. Slower than TRUNCATE, logs individual deletions, can be rolled back.
- **DROP:** Permanently removes entire database objects (tables, views, etc.) from the database. Cannot be undone.

39. How GRANT and REVOKE Commands used?

- The GRANT and REVOKE statements in SQL are used to manage permissions and privileges on database objects, such as tables, views, procedures, and functions.
- **Syntax for GRANT:**

  GRANT permission(s) ON object_name TO user_or_role;
- **Syntax for REVOKE:**

  REVOKE permission(s) ON object_name FROM user_or_role;

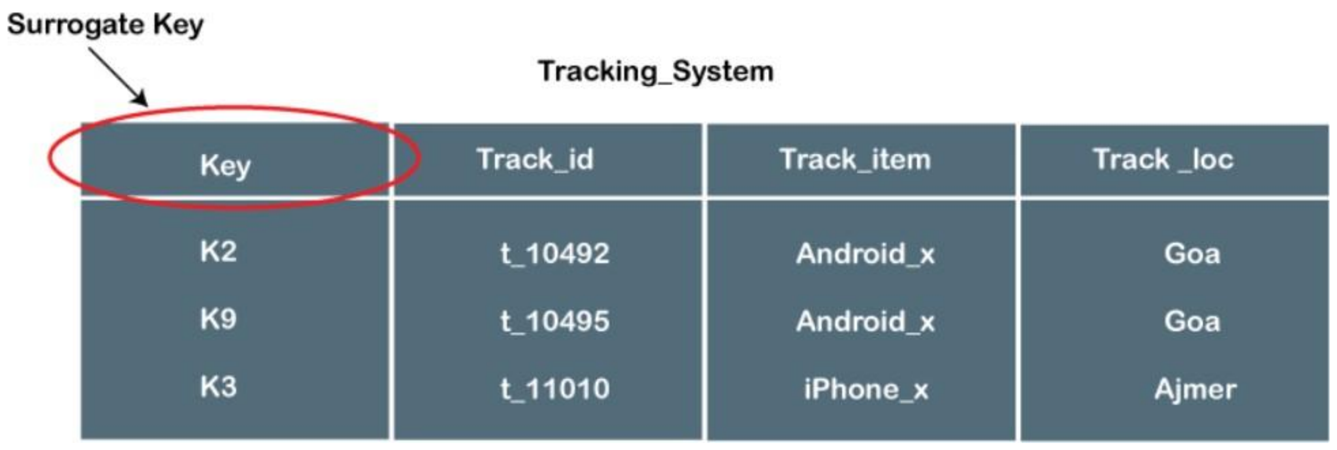
40. What is a primary key, and why is it important?

- A primary key is a unique identifier for each record in a table. It ensures data integrity and helps in data retrieval and table relationships.
- Primary Key contains Unique and Not Null values.

41. What is a foreign key, and how does it relate to other tables?

- A foreign key is a column in a table that refers to the primary key in another table. It establishes a relationship between the two tables.

42. What is a Surrogate Key?


Surrogate Key
Tracking_System

Key	Track_id	Track_item	Track _loc
K2	t_10492	Android_x	Goa
K9	t_10495	Android_x	Goa
K3	t_11010	iPhone_x	Ajmer

fig.20.Surrogate Key

- A surrogate key is a unique identifier generated by the system to uniquely identify each record in a database table. It has no inherent business meaning and is used solely for the purpose of data management and record identification.
- **Example:** System date & time stamp

43. What is a Unique key?

- A Unique key is a database constraint that ensures each value in a specific column or combination of columns is unique and not duplicated within a table. It prevents duplicate entries and helps maintain data integrity.
- **Example**:

```
CREATE TABLE employees
    ( employee_id INT PRIMARY
    KEY,first_name VARCHAR(50),
    last_name VARCHAR(50),
    UNIQUE (employee_id) );
```

44. What is a Composite Key?

- A composite key is a combination of two or more columns in a database table used together to uniquely identify each row. It provides a way to ensure data uniqueness and integrity for complex relationships within the table.
- **Syntax:**

```
CREATE TABLE table_name
   (column1 datatype,
   column2 datatype,
   ...,
   PRIMARY KEY (column1, column2, ...)
);
```

- **Example:**

```
CREATE TABLE Employees
   ( EmployeeID INT,
   EmployeeName VARCHAR(50),
   DepartmentID INT,
   Salary DECIMAL(10, 2),
   PRIMARY KEY (DepartmentID, EmployeeID)
);
```

In this example, the PRIMARY KEY declaration with (DepartmentID, EmployeeID) as its argument creates the composite key.

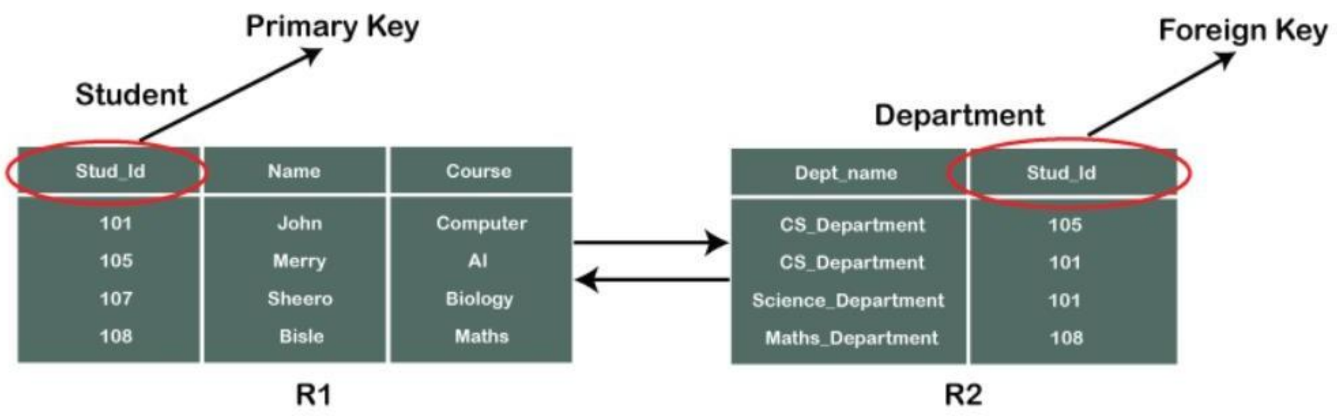
45. What is an Alternate Key?

- An alternate key is a unique key in a database table that can be used as an alternative to the primary key for identifying each row. It offers flexibility, performance benefits, and helps maintain data integrity in certain scenarios.

46. Difference between Surrogate Key and Primary Key?

- The main difference between a surrogate key and a primary key is that a primary key is a unique identifier with meaningful business significance, while a surrogate key is an artificially generated unique identifier with no inherent business meaning.

47. Difference between Primary key and Foreign Key?



Stud_Id	Name	Course
101	John	Computer
105	Merry	AI
107	Sheero	Biology
108	Bisle	Maths

Dept_name	Stud_Id
CS_Department	105
CS_Department	101
Science_Department	101
Maths_Department	108

fig.21.Primary Key Vs Foreign Key

- **Primary Key:** A primary key is a column or a combination of columns that uniquely identifies each row in a table. It ensures data uniqueness within the table.
- **Foreign Key:** A foreign key is a column or a combination of columns in one table that establishes a link or relationship with the primary key of another table. It enables the establishment of relationships between related data in different tables.

48. Difference between SQL and MySQL ?

- **SQL** is a language used to talk to databases and perform actions like storing, retrieving, updating, and deleting data.
- **MySQL** is a specific type of database software that understands the SQL language, allowing you to manage and interact with data using SQL commands. It is just one of many database systems that use SQL. Think of MySQL as a tool that speaks the SQL language and handles the data storage and retrieval tasks for you.

49. How do you Sort data in SQL?

- In SQL, you can sort data in a result set using the ORDER BY clause.
- The ORDER BY clause is typically used at the end of a SQL query to specify the column or columns by which you want to sort the data. It arranges the rows in ascending order by default, but you can also specify the sorting order as ascending or descending.
- **Syntax:**

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC|DESC], column2 [ASC|DESC], ...;
```

- **Example:**

```
SELECT name, age, salary
FROM  employees
ORDER BY age;
```

50. How do you fetch unique records from a table?

- To fetch unique records from a table in SQL, you can use the '**DISTINCT**' keyword in your SELECT statement.
- The 'DISTINCT' keyword ensures that only unique values are returned for the specified columns, removing any duplicates from the result set.
- **Syntax:**

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

- **Example:**

```
SELECT DISTINCT category
FROM products;
```

This query will return a list of unique category names from the "products" table, eliminating any duplicate category entries.

51. What are the different types of Operators?

1) **Arithmetic Operators:** +, -, *, /, %
   - Ex: select salary+1000 from emp
2) **Comparison Operators:** <, <=, >, >=, =, <>,
   - Ex: select * from emp where salary <= 5000
3) **Logical Operators:** AND, OR, NOT
   - Ex: select * from emp where did = 10 and salary <=5000
4) **List Comparison Operators:** IN, ANY, ALL
   - Ex: select * from emp where did in(10,20,30)
5) **Between Operator:** To check the range
   - Ex: select * from emp where did between 10 and 30

52. What is Pattern Matching?

- Pattern matching is the process of searching for specific patterns or sequences of characters within a given text or data.
- In SQL, Pattern matching is done by using LIKE operator.

53. What is the purpose of LIKE Operator?

- The LIKE operator in SQL is used for pattern matching within a column when performing queries.
- It allows you to search for specific patterns or substrings within the values of a column.
- The **LIKE** operator is commonly used with string data types, such as **VARCHAR** or **CHAR.**
- **Syntax:**

```
SELECT column1, column2, ...
FROM table_name
WHERE column_name LIKE pattern;
```

- **Example:**

```
SELECT name
FROM employees
WHERE name LIKE 'J%';
```

This query will return all names from the "employees" table that begin with the letter "J."

54. What are constraints in SQL?

- Constraints in SQL are rules applied to database tables to ensure data integrity and consistency. They enforce limitations on columns, such as unique values, non-null entries, and relationships between tables. Constraints prevent invalid data and maintain data accuracy and reliability.

55. What are the different types of SQL constraints?

- There are several types of SQL constraints that can be applied to database tables to enforce data integrity and maintain consistency. The main types of SQL constraints are:
  - **Primary Key Constraint**: Ensures the uniqueness of values in a column or a combination of columns, serving as a unique identifier for each row in the table.
  - **Foreign Key Constraint:** Establishes a relationship between two tables, ensuring that values in one table's column (foreign key) correspond to values in another table's primary key.
  - **Unique Constraint:** Ensures that all values in a column or a group of columns are unique, preventing duplicate entries.
  - **Check Constraint:** Specifies a condition that must be satisfied for data to be valid in a column. It allows custom data validation beyond data type constraints.
  - **Not Null Constraint:** Ensures that a column does not contain null values, requiring valid data to be provided during insert or update operations.

56. What is the purpose of the GROUP BY clause?

- The purpose of the GROUP BY clause in SQL is to group rows with similar values in one or more columns and apply aggregate functions to each group. It allows you to perform operations on groups of data rather than individual rows, enabling you to summarize and analyze data based on common characteristics.
- When using the GROUP BY clause, the result set is divided into groups based on the values of the specified column(s). Then, aggregate functions, such as SUM, COUNT, AVG, MIN, MAX, etc., are applied to each group, providing summarized information for that group.
- **Syntax:**

```
SELECT column1, aggregate_function(column2)
FROM table_name
GROUP BY column1;
```

- **Example:**

```
SELECT region, SUM(revenue) AS total_revenue
FROM sales
GROUP BY region;
```

This query will group the data by the "region" column, calculate the total revenue for each region, and present the result with the total revenue for each region.

57. What are Aggregate functions?

- Aggregate functions in SQL are functions that operate on a set of values and return a single value as the result.
- These functions allow you to perform calculations on groups of data or an entire column, providing summary information about the dataset.

58. What are the different types of Aggregate Functions?

The commonly used Aggregate functions in SQL are:

- **SUM:** Calculates the sum of all values in a numeric column. It is used to find the total of a set of numeric values.
  - **Syntax:**

```
SELECT SUM(column_name) AS result_alias
```

```
FROM table_name;
```

- **Example:**

```
SELECT SUM(amount) AS total_sales
FROM sales;
```

- **COUNT:** Counts the number of rows in a result set or the number of non-null values in a column. It is used to determine the number of occurrences of a specific value or the total number of rows in a table
  - **Example:**

```
SELECT COUNT(*) as Employee_Count FROM employee;
```

- **AVG:** Computes the average (mean) of all values in a numeric column. It is used to find the average value of a set of numeric data.
  - **Example:**

```
SELECT AVG(salary) FROM employee;
```

- **MIN:** Returns the minimum value from a numeric or character column. It is used to find the smallest value in a set of data.
  - **Example:**

```
SELECT MIN(salary) FROM employee;
```

- **MAX:** Returns the maximum value from a numeric or character column. It is used to find the largest value in a set of data.
  - **Example:**

```
SELECT MAX(salary) FROM employee;
```

- **COUNT DISTINCT:** Counts the number of distinct (unique) values in a column. It is used to find the number of unique occurrences of a specific value.
  - **Example:**

```
SELECT COUNT(DISTINCT customer_id) AS unique_customers FROM orders;
```

- **FIRST:** Returns the first value in a column. It is used to obtain the first occurrence of a value in a sorted result set.
  - **Example:**

```
SELECT department, MIN(first_name) AS first_employee
FROM employees
GROUP BY department;
```

The above query results in the first value of table i.e., Min value…

- **LAST:** Returns the last value in a column. It is used to obtain the last occurrence of a value in a sorted result set.

- **Example:**

```
SELECT department, MAX(first_name) AS last_employee
FROM employees
GROUP BY department;
```

The above query results in the last value of table i.e., Max value…

- **GROUP_CONCAT:** Concatenates strings from a column into a single string with optional separator. It is used to create comma-separated or other types of concatenated strings.
  - **Example:**

```
SELECT department, GROUP_CONCAT(first_name) AS employee_names
FROM employees
GROUP BY department;
```

59. What is the difference between the LIKE and = operators in SQL?

- The 'LIKE' operator is used for pattern matching, while the '=' operator is used for exact matches.

60. How do you use the ORDER BY clause to sort query results?

- The 'ORDER BY' clause is used to sort query results based on specified columns in ascending or descending order.
- **Syntax:**

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;
```

61. What is the GROUP BY clause, and how is it used in SQL?

- The 'GROUP BY' clause is used to group rows with the same values in one or more columns.
- **Syntax:**

```
SELECT column1, column2, ..., aggregate_function(column)
FROM table_name
GROUP BY column1, column2, ...;
```

- **Example:**

```
SELECT department, AVG(salary) AS average_salary
FROM employees
GROUP BY department;
```

62. Explain the HAVING clause and its purpose in SQL queries.

- The ‘HAVING’ clause filters grouped data based on a condition in SQL queries.
- It works in conjunction with the GROUP BY clause to filter grouped data.
- **Syntax:**

```
SELECT column1, column2, ..., aggregate_function(column)
FROM table_name
GROUP BY column1, column2, …
HAVING condition;
```

- **Example:**

```
SELECT department, COUNT(*) AS num_employees
FROM employees
GROUP BY department
HAVING COUNT(*) > 5;
```

The HAVING clause filters the result and only displays departments with more than 5 employees.

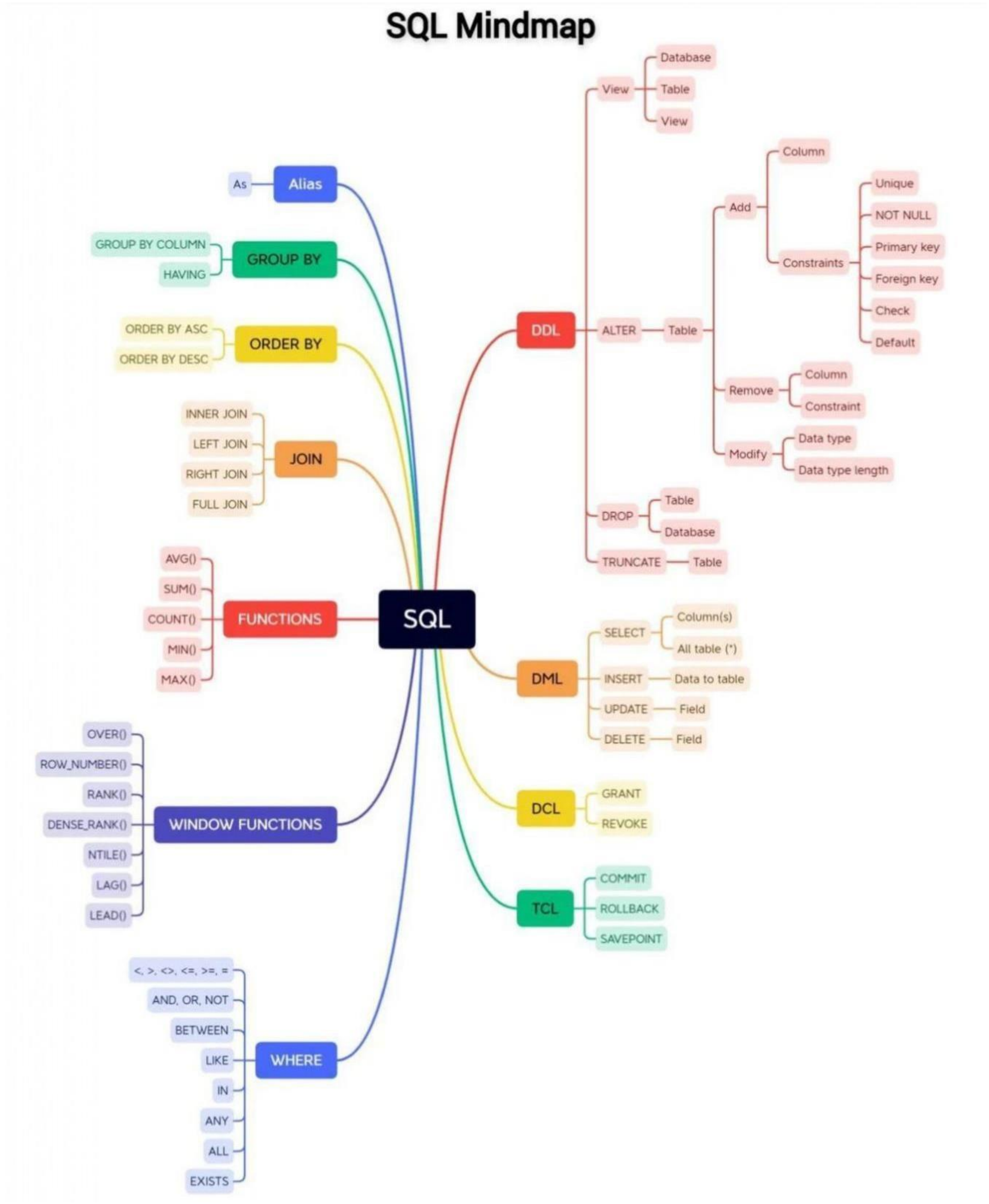


*fig.*22.SQL Mind Map

63. What is the purpose of the WHERE clause in SQL?

➢ The WHERE clause is used to filter rows from a table based on specified conditions, allowing us to retrieve only the data that meets the criteria defined in the WHERE clause.

64. How does the WHERE clause work in a SQL query?

- The WHERE clause filters the result set by specifying conditions that each row must meet. Rows that satisfy the conditions are included in the final result, while those that do not meet the conditions are excluded.

65. Explain the difference between the WHERE clause and the HAVING clause.

- The WHERE clause filters individual rows in a query before any grouping or aggregation, while the HAVING clause filters the results based on aggregated values after grouping has taken place.

66. How can you use comparison operators (e.g., =, <>, >, <, >=, <=) with the WHERE clause?

- Comparison operators are used to compare a column's value with a specific value or another column's value in the WHERE clause.
- **Example:**

```
SELECT * FROM employees WHERE age > 30;
```

67. What is the significance of the LIKE operator in the WHERE clause, and how is it used?

- The LIKE operator is used for pattern matching with string values. It allows the use of wildcard characters (%) to match a portion of the string. For example, WHERE name LIKE 'J%' matches names starting with 'J'.
- **Example:**

```
SELECT * FROM customers WHERE first_name LIKE 'J%';
```

68. How do you filter NULL values using the WHERE clause?

- To filter NULL values, you can use the IS NULL or IS NOT NULL operators in the WHERE clause.
- **Example:**

```
SELECT * FROM orders WHERE customer_id IS NOT NULL;
```

69. How do logical operators (AND, OR, NOT) work with the WHERE clause?

- Logical operators are used to combine multiple conditions in the WHERE clause.
- **Example:**

```
SELECT * FROM employees WHERE age > 30 AND department = 'Sales';
```

70. How can you use the BETWEEN operator with the WHERE clause?

- The BETWEEN operator selects values within a specified range, inclusive.
- **Example:**

```
SELECT * FROM products WHERE price BETWEEN 100 AND 500;
```

71. What is the purpose of the IN operator in the WHERE clause?

- The IN operator is used to filter rows based on a list of specific values for a particular column.
- **Example:**

```
SELECT * FROM products WHERE category IN ('Books', 'Toys');
```

72. How can you optimize a query with multiple WHERE conditions for better performance?

- To optimize a query with multiple WHERE conditions, you can create composite indexes that cover all the columns used in the WHERE clause.
- This helps the database system perform index scans efficiently.

73. Explain the importance of using indexes with the WHERE clause to improve query performance.

- Indexes with the WHERE clause improve query performance by quickly locating rows.

74. What is an Index in MySQL?

- In MySQL, indexes are data structures that improve the performance of queries by allowing the database engine to quickly locate and access the rows that match specific conditions.

75. How to create an Index?

- To create an index on a column or a set of columns, you use the CREATE INDEX statement.
- **Example:**

```
CREATE INDEX idx_name ON employees (name);
```

76. What are the Types of Indexes?

- **B-tree Index:** The most common and default index type in databases. Suitable for general-purpose indexing.
  - **Example:** CREATE INDEX idx_category ON products (category);
- **Unique Index:** Ensures that indexed column(s) contain unique values, preventing duplicates.
  - **Example:** CREATE UNIQUE INDEX idx_product_name ON products (product_name);
- **Composite Index (Multi-column Index):** An index on multiple columns, improving queries involving all or some of the indexed columns.
  - **Example:** CREATE INDEX idx_category_price ON products (category, price);
- **Full-Text Index:** Used for efficient searching of text-based data, supporting advanced text queries.
  - **Example:** CREATE FULLTEXT INDEX idx_product_name ON products (product_name);
- **Spatial Index:** Optimizes spatial queries involving geometric objects, helpful for GIS data types.

- **Example:** CREATE SPATIAL INDEX idx_location ON products (location);

77. What is a View in MySQL?

- In MySQL, a view is a virtual table created by a stored SELECT query, which can be treated and used like a regular table in subsequent queries.
- A view does not store any data on its own but provides an abstracted representation of the underlying data from one or more tables.
- **Syntax:**

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

78. Define Normalization?

- Normalization is the process of organizing data in a relational database to eliminate redundancy and improve data integrity.

79. Types of Normal Forms?

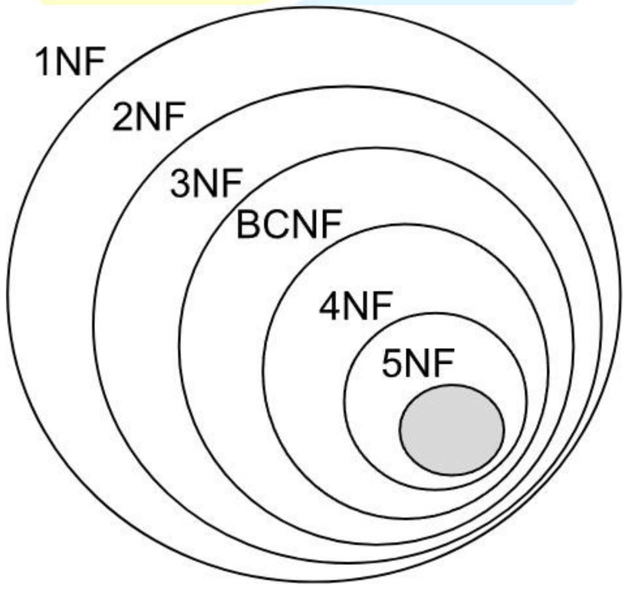


*fig.*23.Types of Normal Forms

- **First Normal Form (1NF):** Ensures each cell in a table contains only one value (no repeating groups or arrays).
- **Second Normal Form (2NF):** All non-key attributes depend on the entire primary key (no partial dependencies).

- **Third Normal Form (3NF):** All non-key attributes depend only on the primary key (no transitive dependencies).
- **Boyce-Codd Normal Form (BCNF):** Every determinant is a candidate key (stronger than 3NF).
- **Fourth Normal Form (4NF):** No multi-valued dependencies (no non-key attribute depends on multiple values of another non-key attribute).
- **Fifth Normal Form (5NF):** All join dependencies are removed.

80. Define Denormalization?

- In MySQL, denormalization is the process of intentionally introducing redundancy into a database by combining tables or duplicating data to improve query performance and simplify data retrieval. It sacrifices some normalization benefits for faster read operations and easier data access.

81. How can data be copied from one table to another table?

- Using the below Query we can copy entire data from one table to another table
- **Syntax:**

```
CREATE TABLE new_table AS SELECT * FROM source_table;
```

82. How to delete duplicate rows in SQL Server?

- **Step 1:** Create a CTE that assigns row numbers to duplicate rows based on the defined order.

```
WITH CTE_Duplicates AS (SELECT column1,column2,column3,ROW_NUMBER() OVER
(PARTITION BY column1, column2 ORDER BY column1) AS RowNumber FROM your_table)
```

- **Step 2:** Use the CTE to delete duplicate rows where RowNumber is greater than 1 (keeping only the first occurrence).

```
DELETE FROM CTE_Duplicates WHERE RowNumber > 1;
```

83. Define Sub Query?

- A subquery, also known as a nested query or inner query, is a query nested inside another SQL query.
- Subqueries provide a powerful way to filter and manipulate data, and they can be used in various scenarios like filtering, aggregation, joining, and more.
- **Example:**
- SELECT emp_name FROM employees WHERE department_id = (SELECT department_id FROM departments WHERE department_name = 'Sales');

84. Define Join in MySQL?

- The JOIN operation allows you to retrieve data from multiple tables simultaneously and is an essential tool for querying and analyzing data stored across different tables in a relational database.

85. Types of Joins?

- **INNER JOIN:** Returns only the matching rows from both tables.

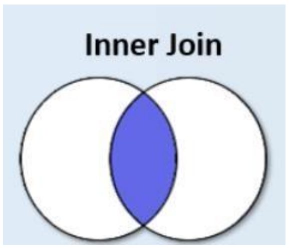


fig.24.Inner Join

- **Example:**

```
SELECT students.student_id, students.student_name, courses.course_name
FROM students
INNER JOIN courses
ON students.course_id = courses.course_id;
```

- **RIGHT JOIN (RIGHT OUTER JOIN):** Returns all the rows from the right table and the matching rows from the left table.

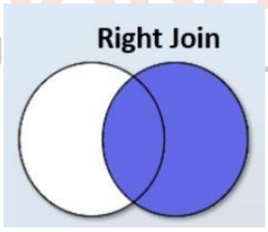


fig.25.Right Join

- **Example:**

```
SELECT employees.employee_id, employees.employee_name, departments.department_name
FROM employees
RIGHT JOIN departments
ON employees.department_id = departments.department_id;
```

- **LEFT JOIN (LEFT OUTER JOIN):** Returns all the rows from the left table and the matching rows from the right table.

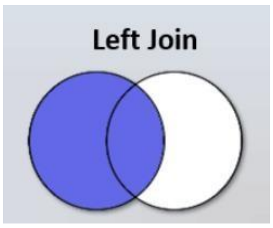


fig.26.Left Join

- **Example:**

```
SELECT employees.employee_id, employees.employee_name, departments.department_name
FROM employees
LEFT JOIN departments
ON employees.department_id = departments.department_id;
```

- **FULL JOIN (FULL OUTER JOIN):** Returns all the rows when there is a match in either the left or right table.

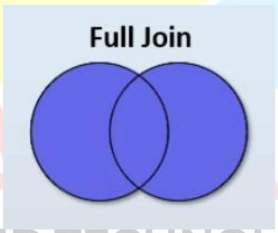


fig.27.Full Join

- **Example:**

```
SELECT employees.employee_id, employees.employee_name, departments.department_name
FROM employees
FULL JOIN departments
ON employees.department_id = departments.department_id;
```

- **SELF JOIN:**A self-join is when a single database table is joined with itself to compare or combine data from different rows within the same table.

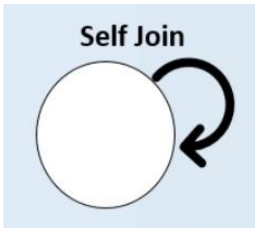


fig.28.Self Join

- **Example:**

  SELECT t1.column1, t1.column2, t2.column3

  FROM your_table t1

  JOIN your_table t2 ON t1.related_column = t2.related_column;

- **CROSS JOIN:**A cross join combines every row from one table with every row from another table, creating a Cartesian product of all possible combinations.

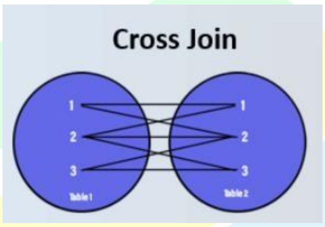


fig.29.Cross Join

- **Example:**

  SELECT columns

  FROM table1

  CROSS JOIN table2;

86. Different Date Functions?

- These functions are used to perform operations like extracting parts of dates, formatting date values, calculating differences between dates, and more. Here are some commonly used date functions in SQL:
- **CURRENT_DATE():** Returns the current date (excluding time) from the system's clock.
- **Syntax:**SELECT CURRENT_DATE();
- **CURRENT_TIME():** Returns the current time (excluding date) from the system's clock.
- **Syntax:**SELECT CURRENT_TIME();
- **CURRENT_TIMESTAMP():** Returns the current date and time from the system's clock.
- **Syntax:**SELECT CURRENT_TIMESTAMP();

- **DATEPART():** Extracts a specific part of a date (e.g., year, month, day) from a date value.
- **Syntax:**DATEPART(date_part, date_expression);
- **DATEADD():** Adds a specific interval to a date (e.g., adding days, months, years).
- **Syntax:**DATEADD(date_part, interval, date_expression);
- **DATEDIFF():** Calculates the difference between two dates in terms of a specified date part (e.g., days, months, years).
- **Syntax:**DATEDIFF(date_part, start_date, end_date);
- **DATE_FORMAT():** Formats a date value into a specific string format.
- Syntax:DATE_FORMAT(date, format);
- **DAY(), MONTH(), YEAR():** Extracts the day, month, and year from a date value, respectively.
- **Syntax:**DAY(date),MONTH(date),YEAR(date);
- **GETDATE():** Similar to CURRENT_TIMESTAMP(), returns the current date and time from the system's clock.
- **Syntax:**SELECT GETDATE();

87. Window Functions?

- Here are some commonly used window functions in MySQL:
- **ROW_NUMBER():** Assigns a unique number to each row, starting from 1, based on the specified order.
- **Example:**
- SELECT product, date, amount,ROW_NUMBER() OVER (ORDER BY date) AS row_num FROM sales;
- **RANK():** Assigns a unique rank to each row based on the specified order, with gaps in the ranking in case of ties.
- **Example:**
- SELECT product,date,amount,RANK() OVER (ORDER BY amount DESC) AS rank FROM sales;
- **DENSE_RANK():** Similar to RANK, but without gaps in the ranking in case of ties.
- **Example:**
- SELECT product,date,amount,DENSE_RANK() OVER (ORDER BY amount DESC) AS dense_rank FROM sales;

88. Define Triggers

- Triggers in MySQL are database objects that automatically respond to specific events (e.g., INSERT, UPDATE, DELETE) on a table, executing predefined actions in response to those events. They are used to enforce rules, maintain data integrity, and automate tasks within the database.