In SQL, the term **virtual** can be related to several concepts, but it is often associated with **views**, **virtual tables**, or **virtual columns**. These concepts provide an abstraction over the physical data stored in the database, allowing users to work with data in a more flexible way.

**1. Virtual Tables (Views)**

A **view** in SQL is often referred to as a **virtual table** because it is not a real table in the database. Instead, it is a saved query that you can treat like a table. When you query a view, the DBMS executes the underlying SQL query to retrieve the data.

**Key Characteristics of Views:**

* A **view** does not store data itself. It is essentially a stored SQL query.
* You can use views to simplify complex queries, encapsulate logic, and present data in a more user-friendly way.
* Views can include **JOINs**, **GROUP BY**, and other operations that aggregate or combine data from multiple tables.

**Creating a View:**

CREATE VIEW Employee\_View AS

SELECT name, department, salary

FROM Employees

WHERE salary > 50000;

In this example, Employee\_View is a **virtual table** that includes employees with a salary greater than 50,000. Whenever you query Employee\_View, it executes the stored SQL statement.

**Querying a View:**

SELECT \* FROM Employee\_View;

This query retrieves the data from the view, just like querying a regular table, but behind the scenes, the DBMS runs the SQL query defined in the view.

**2. Virtual Columns**

Some database management systems support **virtual columns**, which are columns that are not physically stored in the database but are **calculated dynamically** when needed. These columns are often derived from other columns in the table.

**Key Characteristics of Virtual Columns:**

* They don't take up physical storage space.
* They can be used for calculations, like creating a column that shows the result of an arithmetic operation or a string concatenation.
* Virtual columns are computed when queried but are not stored in the database.

**Creating Virtual Columns:**

In databases like Oracle, you can define virtual columns using the GENERATED ALWAYS keyword.

CREATE TABLE Employees (

employee\_id INT,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

salary DECIMAL(10, 2),

full\_name VARCHAR(100) GENERATED ALWAYS AS (first\_name || ' ' || last\_name) VIRTUAL

);

In this example, the full\_name column is a **virtual column** that is automatically generated based on the first\_name and last\_name columns, and it will be computed whenever the data is queried, but it's not physically stored in the database.

**Querying Virtual Columns:**

SELECT employee\_id, full\_name, salary FROM Employees;

This will display the full\_name dynamically created by combining first\_name and last\_name.

**3. Virtual Tables in Other Contexts**

* **Materialized Views**: Unlike regular views, materialized views are stored physically in the database. The data in a materialized view is refreshed periodically based on the underlying query. This can improve performance, especially for complex queries that don’t change frequently.
* **In Memory Tables (Virtual Tables)**: Some DBMSs, such as SQL Server or MySQL, allow the creation of in-memory tables that can be thought of as "virtual" in the sense that they may reside entirely in memory rather than on disk, for faster access.

**4. Benefits of Virtual Tables and Virtual Columns**

* **Simplification**: Views can encapsulate complex logic, making it easier for developers and users to interact with the database.
* **Security**: By restricting access to specific columns or rows through views, you can control what data different users can see without altering the underlying tables.
* **Performance**: Materialized views can improve query performance by storing the results of complex queries and updating them periodically.

**5. Limitations**

* **Performance**: Views can introduce overhead because the database must execute the underlying SQL query each time the view is queried.
* **Non-updatable Views**: Some views (e.g., ones that involve complex joins or aggregations) cannot be used for updates, inserts, or deletes, as the underlying data is not directly modifiable.

**Conclusion**

In SQL, the term **virtual** is typically used in the context of **views** (virtual tables) or **virtual columns**. These features provide a way to abstract and simplify data manipulation without directly storing the data, which can be beneficial for both performance and usability. Understanding how to use and implement virtual concepts in SQL can help in optimizing queries and managing large datasets effectively.