# 13-View

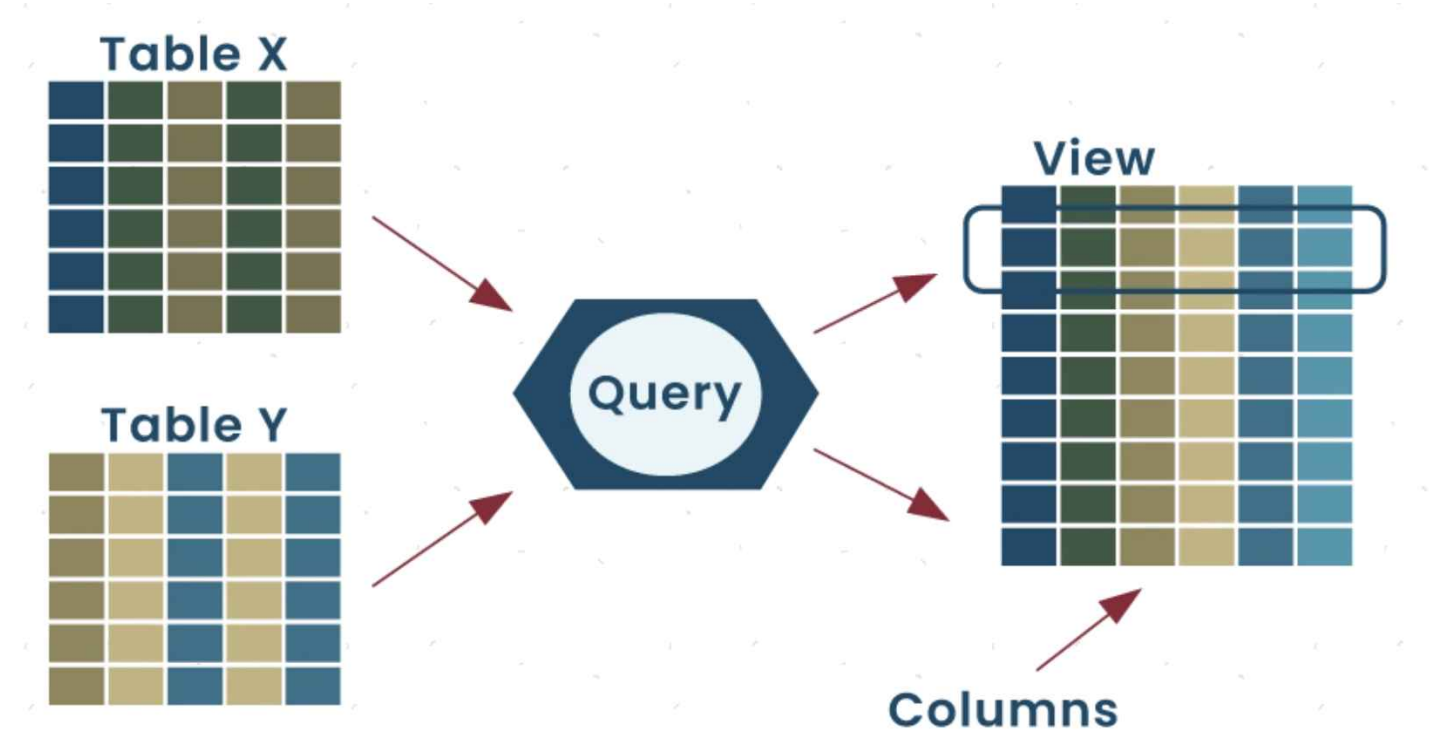
*Author: Vincent Lau*



## Learning Objectives

- Understand the purpose of using View.
- Understand how to implement View.
- Understand the relationship between Physical Tables and View.

## Introduction




In a database, a view is a virtual table that is derived from one or more existing tables (or other views) through a query. **Views do not store any data themselves;** they provide a way to **present data from underlying tables in a customized and controlled manner**. Views can be used to **simplify complex queries**, **restrict access to sensitive data, and encapsulate logic for data retrieval**.

# Create View

Here's an introduction to views along with code examples:

## Creating a View:

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

## Example of Creating a View:

Suppose you have a database with a table named "employees" that includes columns "employee_id," "first_name," "last_name," "department," and "salary." You want to create a view that only shows the "employee_id," "first_name," and "last_name" columns for employees in the "Sales" department:

```sql
CREATE VIEW sales_employee_view
AS
SELECT employee_id, first_name, last_name
FROM employees
WHERE department = 'Sales';
```

# Query View

```sql
SELECT *
FROM sales_employee_view;
```

# Why using View

## Advantanges

### Simplifies Queries

Views can simplify complex queries by encapsulating frequently used logic and complex joins into a single view.

### Data Abstraction

Views provide an abstraction layer, allowing users to interact with data without knowing the underlying structure.

### Data Security

Views can restrict access to sensitive data by exposing only specific columns or rows to certain users.

### Consistency

Views ensure that users see consistent data, as logic can be centralized in the view definition.

### Logical Data Independence

Views provide a level of separation between the physical data structure and how it's presented to users.

## Downsides

### Performance Impact

While modern database systems optimize view execution, complex views can still introduce additional overhead and potentially impact query performance.

### Maintenance Overhead

Maintaining views requires ensuring that they reflect changes in underlying tables, which can be challenging in dynamic environments.

### Limited Complex Operations

Some complex operations or aggregations may be challenging to achieve within a view, requiring direct queries on base tables.

### Potential Abstraction

Overuse of views can lead to a lack of understanding of the actual data structure and relationships.

### Dependent Queries

If many parts of an application depend on a view, changing its structure might impact multiple components.

## Views vs Tables

### Views

#### 1. Virtual Tables

Views are virtual tables created by executing a SELECT query on one or more existing tables. They do not store data themselves; they store the query definition.

#### 2. Data Storage

Views do not store data. Instead, they present data from underlying tables based on the query defined during view creation.

#### 3. Data Modification

Depending on the complexity and definition of the view, some views might allow limited data modification through DML (Data Manipulation Language) operations (e.g., INSERT, UPDATE, DELETE). However, not all views are updatable, especially if they involve multiple tables or complex logic.

#### 4. Data Abstraction

Views provide an abstraction layer, allowing users to interact with a subset of data or a specific perspective of data without needing to know the details of the underlying tables.

### 5. Complex Queries

Views can encapsulate complex queries, aggregations, and joins, simplifying the process of querying for end users.

### 6. Security and Permissions

Views can be used to restrict data access by exposing only certain columns or rows to specific users or roles.

### 7. Dynamic Data

Views reflect the current state of the underlying data; changes in the base tables are immediately reflected in the view.

## Tables

### 1. Physical Storage

Tables store actual data in the database. They are the primary containers for storing and managing data.

### 2. Data Modification

Tables allow direct manipulation of data through DML operations (INSERT, UPDATE, DELETE).

### 3. Data Structure

Tables define the structure of the data, including columns, data types, constraints, and indexes.

### 4. Primary Storage

Data is stored in tables and forms the backbone of the database.

### 5. Data Integrity

Tables enforce primary key and uniqueness constraints, ensuring data integrity.

### 6. Indexes

Tables can be indexed to optimize data retrieval and query performance.

### 7. Data History

Tables store historical data, and each row represents a discrete piece of information.

## Summary of their differences

In summary, views provide a way to create virtual tables that present data from existing tables, offering benefits like data abstraction, query simplification, and controlled data exposure. On the other hand, tables are the physical storage units for data and directly hold the information within the database. The choice between using views and tables depends on the specific requirements of your application, including data access patterns, security considerations, and query complexity.