SQL Views vs MatViews

Difference Between SQL Views vs Materialized Views explain with example

SQL Views vs Materialized Views

SQL Views

A **SQL View** is a virtual table that is based on the result set of an SQL query. It does not store data physically in the database; instead, it provides a way to look at data from one or more tables. Views are dynamically generated when they are queried, meaning the query inside the view definition is executed every time the view is accessed.

Characteristics of SQL Views:

- Do not store data physically.
- Always reflect the current data of the underlying tables.
- Easier to manage as they are virtual and do not require additional storage.
- Can simplify complex queries by encapsulating them into a view.

Example of a SQL View:

Let's assume we have two tables, `**Employees**` and `**Departments**`.

Employees Table:

EmployeeID	Name	DepartmentID
1	John	1
2	Jane	2
3	Doe	1

Departments Table:

DepartmentID	DepartmentName
1	HR
2	IT

We can create a view to see the employees along with their department names:

```sql
CREATE VIEW EmployeeDepartments AS
SELECT e.EmployeeID, e.Name, d.DepartmentName
FROM Employees e
JOIN Departments d ON e.DepartmentID = d.DepartmentID;
```

Now, querying the view `**EmployeeDepartments**`:

```sql
SELECT * FROM EmployeeDepartments;
```

Would yield:

EmployeeID	Name	DepartmentName
1	John	HR
2	Jane	IT
3	Doe	HR

Materialized Views 

A **Materialized View** is a database object that contains the results of a query and stores them physically. Unlike regular views, the data in a materialized view is not dynamically generated; instead, it is periodically refreshed based on the defined refresh mechanism.

Characteristics of Materialized Views:

- Store data physically in the database.
- May not always reflect the most current data of the underlying tables (depending on the refresh frequency).
- Can improve performance for complex queries since the data is precomputed and stored.
- Require additional storage space.

Example of a Materialized View:

Using the same `**Employees**` and `**Departments**` tables, let's create a materialized view.

```sql
CREATE MATERIALIZED VIEW EmployeeDepartmentsMaterialized
AS
SELECT e.EmployeeID, e.Name, d.DepartmentName
FROM Employees e
JOIN Departments d ON e.DepartmentID = d.DepartmentID
WITH DATA;
```

To ensure the materialized view is refreshed periodically, you can use the following command:

```sql
REFRESH MATERIALIZED VIEW EmployeeDepartmentsMaterialized;
```

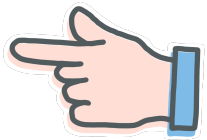
Now, querying the materialized view:

```sql
SELECT * FROM EmployeeDepartmentsMaterialized;
```


Would yield the same result:

EmployeeID	Name	DepartmentName
1	John	HR
2	Jane	IT
3	Doe	HR

However, the data comes from the materialized view storage, not directly from the underlying tables.

Key Differences 

1. **Storage**:
   - SQL Views: No physical storage, data is fetched dynamically.
   - Materialized Views: Physical storage, data is stored and can be refreshed periodically.
2. **Performance**:
   - SQL Views: Slower for complex queries as they are dynamically generated.
   - Materialized Views: Faster for read operations as data is precomputed.
3. **Data Freshness**:
   - SQL Views: Always show the latest data.
   - Materialized Views: Data may be stale depending on the refresh interval.
4. **Use Cases**:
   - SQL Views: Simplifying complex queries, providing abstraction.
   - Materialized Views: Performance improvement for complex queries with infrequent updates.