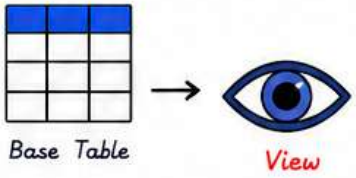
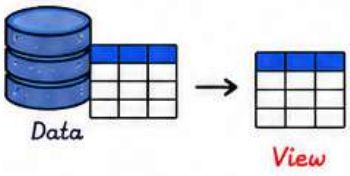
Chapter 16: MySQL Views

16.1 Introduction to Views

* A view is a virtual table based on the result of a SELECT query.
* It does not store data physically.
* Views provide a convenient way to simplify complex queries.
* They can be used to restrict access to data.
* Views are always up-to-date because they are based on the underlying tables.
* A view can be used like a table in SELECT queries.
* You cannot directly insert, update or delete data in a view (unless it is an updatable view).





Example 1 : A view that shows employee names and departments.

Query :

```
CREATE VIEW emp_dept_view AS
SELECT name, department
FROM employees;
```

Key Points :

- ☑ Views are based on SELECT queries.
- ☑ They do not store data.
- ☑ They help simplify queries and provide security.



16.2 Creating Views

* A view is created using the CREATE VIEW statement.
* You specify the name of the view and the SELECT query that defines the view.
* The view name acts like a table name in future queries.
* If the underlying tables change, the view reflects the changes automatically.

Example 2 : Create a view that shows employee names and their salaries.

Query :

```
CREATE VIEW emp_salary_view AS
SELECT name, salary
FROM employees;
```

Key Points :

- ☑ Use CREATE VIEW to define a view.
- ☑ The view is stored in the database.
- ☑ You can query the view like a table.
- ☑ The view is always up-to-date.



Example 3 : Query the view created above.

Query :

```
SELECT * FROM emp_salary_view;
```

Result :

name	salary
Alice	60000
Bob	55000
Cathy	70000



Note : You can create views with joins, calculations and conditions to make complex queries easier to manage.

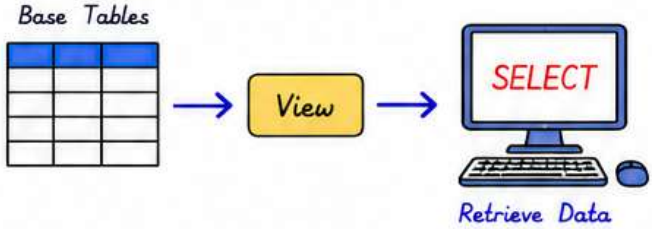
Key Takeaway :

- ☑ A view is a virtual table based on a SELECT query.
- ☑ Views simplify queries and improve security.
- ☑ Views are created using CREATE VIEW.
- ☑ You can query a view just like a table.



16.3 Selecting from Views

* Once a view is created, you can retrieve data from it using the SELECT statement.
* Views behave like virtual tables.
* You can use SELECT with conditions, sorting, and other clauses on views.
* The data in the view reflects the current data in the underlying tables.



Example 1 : Select all data from a view.

Query :

```
SELECT *
FROM emp_salary_view;
```

Result :

name	salary
Alice	60000
Bob	55000
Cathy	70000

Example 2 : Select specific columns from a view.

Query :

```
SELECT name
FROM emp_salary_view;
```

Result :

name
Alice
Bob
Cathy

Example 3 : Select data from a view with a condition.

Query :

```
SELECT *
FROM emp_salary_view
WHERE salary > 55000;
```

Result :

name	salary
Alice	60000
Cathy	70000

Example 4 : Select data from a view with sorting.

Query :

```
SELECT *
FROM emp_salary_view
ORDER BY salary DESC;
```

Result :

name	salary
Cathy	70000
Alice	60000
Bob	55000

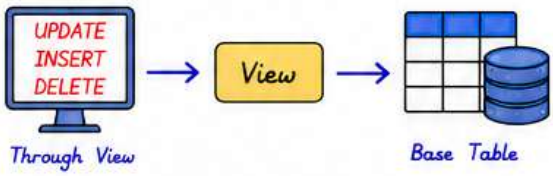
Key Points :

- ☑ You can query a view using the SELECT statement.
- ☑ Views can be used with WHERE, ORDER BY, GROUP BY, etc.
- ☑ The result is always based on the current data in the underlying tables.
- ☑ Views simplify complex queries and improve readability.



16.4 Updating Views

* If a view is based on a single table, you can update, insert or delete data through the view.
* If a view is based on multiple tables or contains GROUP BY, DISTINCT, aggregate functions, etc., it is not updatable.
* When you update data through a view, the changes are applied to the underlying base table.



Example 1 : Update salary of an employee using a view.

Query :

```
UPDATE emp_salary_view
SET salary = 65000
WHERE name = 'Bob';
```

Explanation :

The salary of Bob is updated in the underlying table through the view.

Result (emp_salary_view) :

name	salary
Alice	60000
Bob	65000
Cathy	70000

Example 2 : Insert a new employee using a view.

Query :

```
INSERT INTO emp_salary_view (name, salary)
VALUES ('David', 50000);
```

Result (emp_salary_view) :

name	salary
Alice	60000
Bob	65000
Cathy	70000
David	50000

Example 3 : Delete an employee using a view.

Query :

```
DELETE FROM emp_salary_view
WHERE name = 'Alice';
```

Result (emp_salary_view) :

name	salary
Bob	65000
Cathy	70000
David	50000

Key Points :

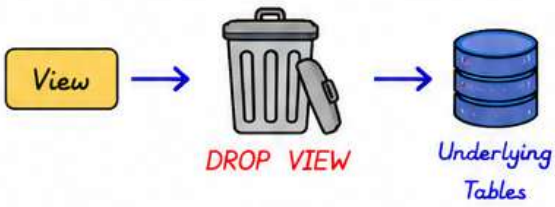
- ☑ You can update, insert or delete data through a view only if it is based on a single table.
- ☑ Changes made through the view affect the underlying base table.
- ☑ Views based on multiple tables, GROUP BY, DISTINCT, etc. are read-only.



Note : Always ensure the view is updatable before performing UPDATE, INSERT or DELETE operations.

16.5 Dropping Views

- When a view is no longer needed, you can remove it using DROP VIEW statement.
- Dropping a view does not affect the underlying tables.
- Once a view is dropped, it cannot be used in any query.
- If a view is dropped and recreated, you must grant permissions again if needed.



Example 1 : Drop a single view.

Query :

```
DROP VIEW emp_salary_view;
```

Explanation :

This will remove the emp_salary_view from the database.

Example 2 : Check if a view exists.

Query :

```
SHOW FULL TABLES
WHERE Table_type = 'VIEW';
```

Result :

Tables_in_company	Table_type
emp_dept_view	VIEW
emp_salary_view	VIEW

Example 3 : Drop a view if it exists.

Query :

```
DROP VIEW IF EXISTS emp_dept_view;
```

Explanation :

The view will be dropped only if it exists, avoiding an error.

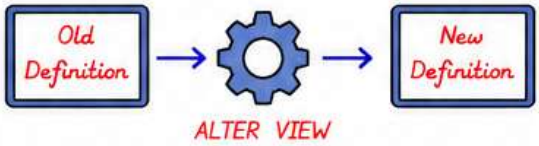


Key Points :

- ☑ Use DROP VIEW to remove a view from the database.
- ☑ Dropping a view does not affect the base tables.
- ☑ A dropped view cannot be used in queries.
- ☑ Use IF EXISTS to avoid errors if the view does not exist.

16.6 ALTER VIEW

- ALTER VIEW is used to modify the definition of an existing view.
- You can change the SELECT query that defines the view.
- The view name remains the same.
- All permissions on the view remain intact after alteration.



Example 1 : Modify a view to include an additional column.

Query :

```
ALTER VIEW emp_dept_view AS
SELECT name, department, salary
FROM employees;
```

Explanation :

The emp_dept_view is modified to include the salary column.

Result (emp_dept_view) :

name	department	salary
Alice	Sales	60000
Bob	Sales	55000
Cathy	HR	70000
David	HR	65000

Example 2 : Change the condition in a view.

Query :

```
ALTER VIEW high_salary_view AS
SELECT name, salary
FROM employees
WHERE salary >= 60000;
```

Explanation :

The view now shows only employees with salary greater than or equal to 60000.

Result (high_salary_view) :

name	salary
Alice	60000
Cathy	70000



Example 2 : Add a calculated column in the view.

Query :

```
ALTER VIEW emp_bonus_view AS
SELECT name, salary,
  salary * 0.10 AS bonus
FROM employees;
```

Explanation :

A new calculated column bonus (10% of salary) is added to the view.

Result (emp_bonus_view) :

name	salary	bonus
Alice	60000	6000
Bob	55000	5500
Cathy	70000	7000

Key Points :

- ☑ Use ALTER VIEW to change the definition of an existing view.
- ☑ You can modify columns, conditions and calculations in a view.
- ☑ The view name and permissions remain the same after alteration.
- ☑ Changes in the underlying tables automatically reflect in the view.

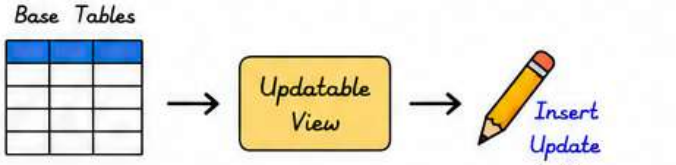


Key Takeaway : Views make complex queries simple and secure. You can create, query, update, modify and drop views easily using SQL statements.



16.7 Updatable Views

- A view is updatable if it meets certain conditions.
- Data in an updatable view can be inserted, updated or deleted.
- An updatable view must be based on a single table.
- It must not contain GROUP BY, DISTINCT, aggregate functions or subqueries.
- Most simple views are updatable.



Conditions for an Updatable View :

- ☑ Based on a single table.
- ☑ Does not use GROUP BY.
- ☑ Does not use DISTINCT.
- ☑ Does not use aggregate functions.
- ☑ Does not use subqueries.
- ☑ Does not use UNION, INTERSECT, EXCEPT.



Example 1 : Update data through an updatable view.

Query :

```
UPDATE emp_view
SET salary = 65000
WHERE name = 'Bob';
```

Explanation :

Since emp_view is based on a single table and has no restrictions, the update is allowed and applied to the base table.

Result :

name	department	salary
Alice	Sales	60000
Bob	Sales	65000
Cathy	HR	70000

Example 2 : Insert data through an updatable view.

Query :

```
INSERT INTO emp_view (name, department, salary)
VALUES ('David', 'HR', 55000);
```

Result :

name	department	salary
Alice	Sales	60000
Bob	Sales	65000
Cathy	HR	70000
David	HR	55000

Explanation :

The new row is inserted into the underlying table through the view.

Example 3 : Delete data through an updatable view.

Query :

```
DELETE FROM emp_view
WHERE name = 'Alice';
```

Result :

name	department	salary
Bob	Sales	65000
Cathy	HR	70000
David	HR	55000

Explanation :

The row is deleted from the base table through the view.

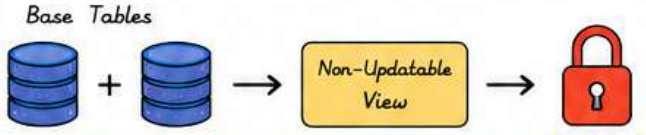
Key Points :

- ☑ Updatable views allow INSERT, UPDATE and DELETE operations.
- ☑ They must be based on a single table and follow certain rules.
- ☑ Changes made through the view affect the underlying base table.
- ☑ Most simple views that follow the rules are updatable.



16.8 Non-Updatable Views

- A view is non-updatable if it does not meet the conditions for an updatable view.
- You cannot insert, update or delete data through a non-updatable view.
- These views are usually created using GROUP BY, DISTINCT, aggregate functions, subqueries, joins, or set operations.
- They are used only for retrieving data.



Common Reasons for Non-Updatable Views :

- ☑ Uses GROUP BY clause.
- ☑ Uses DISTINCT.
- ☑ Uses aggregate functions (SUM, AVG, COUNT, etc.).
- ☑ Uses subqueries.
- ☑ Uses joins of multiple tables.
- ☑ Uses UNION, INTERSECT or EXCEPT.

Example 1 : A view using GROUP BY is not updatable.

Query :

```
CREATE VIEW dept_count_view AS
SELECT department, COUNT(*) AS total_emp
FROM employees
GROUP BY department;
```

Try to Update :

```
UPDATE dept_count_view
SET total_emp = 10
WHERE department = 'Sales';
```

Result / Error :

Error: This view is not updatable because it uses GROUP BY.



Example 2 : A view using JOIN is not updatable.

Query :

```
CREATE VIEW emp_dept_view AS
SELECT e.name, d.department
FROM employees e
JOIN departments d ON e.dept_id = d.dept_id;
```

Try to Insert :

```
INSERT INTO emp_dept_view (name, department)
VALUES ('David', 'Sales');
```

Result / Error :

Error: This view is not updatable because it uses JOIN.



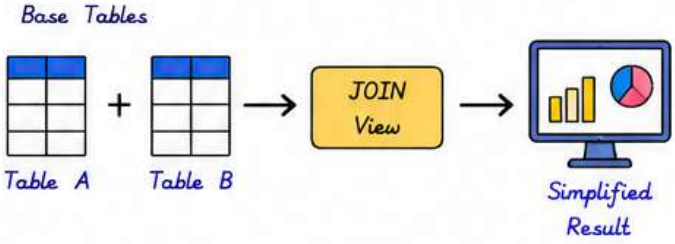
Key Points :

- ☑ Non-updatable views are used only for retrieving data.
- ☑ They cannot be used to insert, update or delete data.
- ☑ They are created using GROUP BY, DISTINCT, aggregate functions, joins, subqueries or set operations.
- ☑ They are useful for reporting and analysis.



16.9 Views with JOINs

- Views can be created using JOINs between multiple tables.
- These views simplify complex queries that involve relationships between tables.
- Such views are usually non-updatable.
- JOIN views are used mainly for retrieving and reporting data.



Example 1 : Create a view to display employee names with their department names.

Query :

```
CREATE VIEW emp_dept_view AS
SELECT e.name, e.salary, d.dept_name
FROM employees e
JOIN departments d ON e.dept_id = d.dept_id;
```

Result (emp_dept_view) :

name	salary	dept_name
Alice	60000	Sales
Bob	55000	Sales
Cathy	70000	HR
David	65000	HR

Explanation :

The view combines data from employees and departments tables using JOIN.

Example 2 : Create a view using multiple JOINs.

Query :

```
CREATE VIEW emp_project_view AS
SELECT e.name, p.project_name, d.dept_name
FROM employees e
JOIN departments d ON e.dept_id = d.dept_id
JOIN projects p ON e.emp_id = p.emp_id;
```

Result (emp_project_view) :

name	project_name	dept_name
Alice	Website Revamp	Sales
Bob	Mobile App	Sales
Cathy	HR Portal	HR
David	Recruitment System	HR

Explanation :

The view displays employee names along with their projects and departments.

Key Points :

- ☑ Views can be created using JOINs between multiple tables.
- ☑ JOIN views are usually non-updatable.
- ☑ They help simplify complex queries and improve readability.
- ☑ They are used mainly for reporting and analysis.

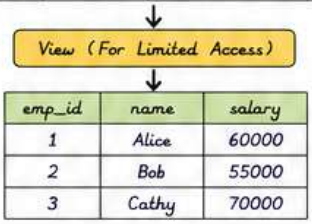


16.10 Views for Security

- Views can be used to control access to data.
- Users can be given access to a view instead of the actual base tables.
- This hides sensitive data and improves security.
- Views allow you to show only required columns or rows to specific users.

Base Table (employees)

emp_id	name	salary	email	dob	address
1	Alice	60000	alice@mail.com	01-01-1995	Hyderabad
2	Bob	55000	bob@mail.com	05-02-1994	Delhi
3	Cathy	70000	cathy@mail.com	10-03-1996	Mumbai



Example 1 : Create a view to allow users to see only employee_id, name and salary.

Query :

```
CREATE VIEW emp_public_view AS
SELECT emp_id, name, salary
FROM employees;
```

Result (emp_public_view) :

emp_id	name	salary
1	Alice	60000
2	Bob	55000
3	Cathy	70000

Explanation :

Only selected columns are visible through the view.

Example 2 : Create a view to show employees from a specific department only.

Query :

```
CREATE VIEW hr_view AS
SELECT emp_id, name, salary, email
FROM employees
WHERE dept_id = 2;
```

Result (hr_view) :

emp_id	name	salary	email
3	Cathy	70000	cathy@mail.com
4	David	65000	david@mail.com

Explanation :

The view shows only employees from the HR department.



Secure Access

Key Points :

- ☑ Views help in restricting access to sensitive data.
- ☑ Users are given access to views instead of base tables.
- ☑ You can show only required columns or rows to specific users.
- ☑ This improves data security and reduces the risk of unauthorized access.



Note : Always grant permissions on views instead of base tables for better control and security.