SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Re-accredited by NAAC with A+ grade, Accredited by NBA(CSE, IT, ECE, EEE & Mechanical)
Approvedy by AICTE, New Delhi, Recognized by UGC, Affiliated to Anna University, Chennai

Department of MCA

DBMS SQL Views

Course Name : 23CAT603 - DATA BASE MANAGEMENT SYSTEM

Class : I Year / I Semester

Unit IV – SQL Views

SQL Views

Views in SQL are a type of **virtual table** that simplifies how users interact with data across one or more tables. Unlike **traditional tables**, a view in **SQL** does not store data on disk; instead, it dynamically retrieves data based on a pre-defined query each time it's accessed.

SQL views are particularly useful for managing complex queries, enhancing security, and presenting data in a simplified format. In this guide, we will cover the **SQL** create view statement, updating and deleting views, and using the WITH CHECK OPTION clause.

What is a View in SQL?

A view in SQL is a saved SQL query that acts as a virtual table. It can fetch data from one or more tables and present it in a customized format, allowing developers to:

Simplify Complex Queries: Encapsulate complex joins and conditions into a single object.

Enhance Security: Restrict access to specific columns or rows.

Present Data Flexibly: Provide tailored data views for different users.

SQL Views

Demo SQL Database

We will be using these **two SQL tables** for examples.

StudentDetails

```
-- Create StudentDetails table
CREATE TABLE StudentDetails (
   S_ID INT PRIMARY KEY,
   NAME VARCHAR(255),
   ADDRESS VARCHAR(255)
);

INSERT INTO StudentDetails (S_ID, NAME, ADDRESS)
VALUES
   (1, 'Harsh', 'Kolkata'),
   (2, 'Ashish', 'Durgapur'),
   (3, 'Pratik', 'Delhi'),
   (4, 'Dhanraj', 'Bihar'),
   (5, 'Ram', 'Rajasthan');
```

S_ID	NAME	ADDRESS
1	Harsh	Kolkata
2	Ashish	Durgapur
3	Pratik	Delhi
4	Dhanraj	Bihar
5	Ram	Rajasthan

SQL Views

StudentMarks

```
-- Create StudentMarks table
CREATE TABLE StudentMarks (
    ID INT PRIMARY KEY,
    NAME VARCHAR(255),
    Marks INT,
    Age INT
);

INSERT INTO StudentMarks (ID, NAME, Marks, Age)
VALUES
    (1, 'Harsh', 90, 19),
    (2, 'Suresh', 50, 20),
    (3, 'Pratik', 80, 19),
    (4, 'Dhanraj', 95, 21),
    (5, 'Ram', 85, 18);
```

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

SQL Views

CREATE VIEWS in SQL

We can create a view using **CREATE VIEW** statement. A View can be created from a single table or multiple tables.

Syntax:

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

Parameters:

view_name: Name for the View

table_name: Name of the table

condition: Condition to select rows

SQL CREATE VIEW Statement Examples

Let's look at some examples of CREATE VIEW Statement in SQL to get a better understanding of how to create views in SQL.

SQL Views

Example 1: Creating View From a Single Table

In this example, we will create a View named DetailsView from the table StudentDetails.

Query:

CREATE VIEW DetailsView **AS SELECT** NAME, ADDRESS **FROM** StudentDetails **WHERE** S_ID < 5;

To see the data in the View, we can query the view in the same manner as we query a table.

SELECT * **FROM** DetailsView;

Output:

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

SQL Views

Example 2: Create View From Table

In this example, we will create a view named StudentNames from the table StudentDetails.

Query:

CREATE VIEW StudentNames **AS SELECT** S_ID, NAME **FROM** StudentDetails **ORDER BY** NAME; If we now query the view as,
SELECT * **FROM** StudentNames;

Output:

S_ID	NAMES
2	Ashish
4	Dhanraj
1	Harsh
3	Pratik
5	Ram

SQL Views

Example 3: Creating View From Multiple Tables

In this example we will create a View named MarksView from two tables StudentDetails and StudentMarks. To create a View from multiple tables we can simply include multiple tables in the SELECT statement.

Query:

CREATE VIEW MarksView AS **SELECT** StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS **FROM** StudentDetails, StudentMarks **WHERE** StudentDetails.NAME = StudentMarks.NAME; To display data of View MarksView:

SELECT * **FROM** MarksView;

Output:

NAME	ADDRESS	MARKS
Harsh	Kolkata	90
Pratik	Delhi	80
Dhanraj	Bihar	95
Ram	Rajasthan	85

SQL Views

Listing all Views in a Database

We can list View using the **SHOW FULL TABLES** statement or using the **information_schema table**.

A View can be created from a single table or multiple tables.

Syntax:

USE "database_name"; **SHOW FULL** TABLES **WHERE** table_type LIKE "%VIEW"; **Using information_schema**

SELECT table_name **FROM** information_schema.views **WHERE** table_schema = 'database_name'; OR **SELECT** table_schema, table_name, view_definition **FROM** information_schema.views **WHERE** table_schema = 'database_name';

DELETE VIEWS in SQL

SQL allows us to delete an existing View. We can delete or drop View using the **DROP statement**.

Syntax:

DROP VIEW view_name; **Example**

In this example, we are deleting the View **MarksView.**

DROP VIEW MarksView;

SQL Views

UPDATE VIEW in SQL

If you want to update the existing data within the view, use the **UPDATE** statement.
Syntax:

UPDATE view_name **SET** column1 = value1, column2 = value2...., columnN = valueN **WHERE** [condition];**Note:** Not all views can be updated using the UPDATE statement.

If you want to update the view definition without affecting the data, use the **CREATE OR REPLACE VIEW** statement. you can use this syntax

CREATE OR REPLACE VIEW view_name **AS SELECT** column1, column2, ... **FROM** table_name **WHERE** condition;

SQL Views

Rules to Update Views in SQL:

Certain conditions need to be satisfied to update a view. If any of these conditions are **not** met, the view can not be updated.

The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.

The SELECT statement should not have the DISTINCT keyword.

The View should have all NOT NULL values.

The view should not be created using nested queries or complex queries.

The view should be created from a single table. If the view is created using multiple tables then we will not be allowed to update the view.

Examples

Let's look at different use cases for updating a view in SQL. We will cover these use cases with examples to get a better understanding.

SQL Views

Example 1: Update View to Add or Replace a View Field

We can use the **CREATE OR REPLACE VIEW** statement to add or replace fields from a view. If we want to update the view **MarksView** and add the field AGE to this View from **StudentMarks** Table, we can do this by:

CREATE OR REPLACE VIEW MarksView AS **SELECT** StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS, StudentMarks.AGE **FROM** StudentDetails, StudentMarks **WHERE** StudentDetails.NAME = StudentMarks.NAME; If we fetch all the data from MarksView now as:

SELECT * **FROM** MarksView; **Output:**

NAME	ADDRESS	MARKS	AGE
Harsh	Kolkata	90	19
Pratik	Delhi	80	19
Dhanraj	Bihar	95	21
Ram	Rajasthan	85	18

SQL Views

Example 2: Update View to Insert a row in a view

We can insert a row in a View in the same way as we do in a table. We can use the INSERT INTO statement of SQL to insert a row in a View.

In the below example, we will insert a new row in the View DetailsView which we have created above in the example of “creating views from a single table”.

INSERT INTO DetailsView(NAME, ADDRESS) VALUES("Suresh","Gurgaon"); If we fetch all the data from DetailsView now as,

SELECT * FROM DetailsView;

Output:

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar
Suresh	Gurgaon

SQL Views

Example 3: Deleting a row from a View

Deleting rows from a view is also as simple as deleting rows from a table. We can use the DELETE statement of SQL to delete rows from a view. Also deleting a row from a view first deletes the row from the actual table and the change is then reflected in the view.

In this example, we will delete the last row from the view DetailsView which we just added in the above example of inserting rows.

DELETE FROM DetailsView **WHERE** NAME="Suresh"; If we fetch all the data from DetailsView now as,

SELECT * **FROM** DetailsView;

Output:

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

SQL Views

WITH CHECK OPTION Clause

The **WITH CHECK OPTION** clause in SQL is a very useful clause for views. It applies to an updatable view.

The WITH CHECK OPTION clause is used to prevent data modification (using INSERT or UPDATE) if the condition in the WHERE clause in the CREATE VIEW statement is not satisfied.

If we have used the WITH CHECK OPTION clause in the CREATE VIEW statement, and if the UPDATE or INSERT clause does not satisfy the conditions then they will return an error.

WITH CHECK OPTION Clause Example:

In the below example, we are creating a View SampleView from the StudentDetails Table with a WITH CHECK OPTION clause.

CREATE VIEW SampleView AS **SELECT** S_ID, NAME **FROM** StudentDetails **WHERE** NAME IS NOT NULL **WITH CHECK OPTION**; In this view, if we now try to insert a new row with a null value in the NAME column then it will give an error because the view is created with the condition for the NAME column as NOT NULL. For example, though the View is updatable then also the below query for this View is not valid:

INSERT INTO SampleView(S_ID) **VALUES**(6); **NOTE**: The default value of NAME column is *null*.

SQL Views

Uses of a View

A good database should contain views for the given reasons:

Restricting data access – Views provide an additional level of table security by restricting access to a predetermined set of rows and columns of a table.

Hiding data complexity – A view can hide the complexity that exists in multiple joined tables.

Simplify commands for the user – Views allow the user to select information from multiple tables without requiring the users to actually know how to perform a join.

Store complex queries – Views can be used to store complex queries.

Rename Columns – Views can also be used to rename the columns without affecting the base tables provided the number of columns in view must match the number of columns specified in a select statement. Thus, renaming helps to hide the names of the columns of the base tables.

Multiple view facility – Different views can be created on the same table for different users.

Creating Triggers

The syntax for creating a trigger is

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

References

1. https://www.geeksforgeeks.org/sql-views/