SQL | Views

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. In this article we will learn about creating , deleting and updating Views.

Sample Tables:

StudentDetails

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

StudentMarks

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

CREATING VIEWS

We can create 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;
```

view_name: Name for the View
table_name: Name of the table
condition: Condition to select rows

Examples:

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

- 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

- **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

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 (Using SHOW FULL TABLES):

```
use "database_name";
show full tables where table_type like "%VIEW";
```

Syntax (Using information_schema) :

```
select * 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";
```

DELETING VIEWS

We have learned about creating a View, but what if a created View is not needed any more? Obviously we will want to delete it. SQL allows us to delete an existing View. We can delete or drop a View using the DROP statement. **Syntax**:

```
DROP VIEW view_name;
```

view_name: Name of the View which we want to delete.

For example, if we want to delete the View **MarksView**, we can do this as:

```
DROP VIEW MarksView;
```

UPDATING VIEWS

There are certain conditions needed to be satisfied to update a view. If any one of these conditions is **not** met, then we will not be allowed to update the view.

1. The SELECT statement which is used to create the view should not include GROUP BY clause or ORDER BY clause.
2. The SELECT statement should not have the DISTINCT keyword.
3. The View should have all NOT NULL values.
4. The view should not be created using nested queries or complex queries.
5. 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.

- We can use the **CREATE OR REPLACE VIEW** statement to add or remove fields from a view. **Syntax**:

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

- For example, if we want to update the view **MarksView** and add the field AGE to this View from **StudentMarks** Table, we can do this as:

```
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

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

Syntax:

```
INSERT INTO view_name(column1, column2 , column3,..)
VALUES(value1, value2, value3..);
```

`view_name`: Name of the View

Example: 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

- **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 delete the row from the actual table and the change is then reflected in the view.

Syntax:

```
DELETE FROM view_name
WHERE condition;
```

view_name:Name of view from where we want to delete rows
condition: Condition to select rows

Example: 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

WITH CHECK OPTION

The WITH CHECK OPTION clause in SQL is a very useful clause for views. It is applicable to an updatable view. If the view is not updatable, then there is no meaning of including this clause in the CREATE VIEW statement.

- The WITH CHECK OPTION clause is used to prevent the insertion of rows in the view where the condition in the WHERE clause in 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.

Example: In the below example we are creating a View SampleView from StudentDetails Table with 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 null value in the NAME column then it will give an error because the view is created with the condition for NAME column as NOT NULL. For example,though the View is updatable but 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.*

Uses of a View: A good database should contain views due to the given reasons:

1. **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.
2. **Hiding data complexity** – A view can hide the complexity that exists in multiple tables join.
3. **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.
4. **Store complex queries** – Views can be used to store complex queries.
5. **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 select statement. Thus, renaming helps to hide the names of the columns of the base tables.
6. **Multiple view facility** – Different views can be created on the same table for different users.

This article is contributed by **Harsh Agarwal**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or if you want to share more information about the topic discussed above.

Last Updated : 21 Mar, 2023

197

Similar Reads