**VIEW**

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Views, which are a type of virtual tables allow users to do the following −

* Structure data in a way that users or classes of users find natural or intuitive.
* Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
* Summarize data from various tables which can be used to generate reports.

Creating Views

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

To create a view, a user must have the appropriate system privilege according to the specific implementation.

The basic **CREATE VIEW** syntax is as follows −

CREATE VIEW view\_name AS

SELECT column1, column2.....

FROM table\_name

WHERE [condition];

You can include multiple tables in your SELECT statement in a similar way as you use them in a normal SQL SELECT query.

Example

Consider the CUSTOMERS table having the following records −

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

Following is an example to create a view from the CUSTOMERS table. This view would be used to have customer name and age from the CUSTOMERS table.

SQL > CREATE VIEW CUSTOMERS\_VIEW AS

SELECT name, age

FROM CUSTOMERS;

Now, you can query CUSTOMERS\_VIEW in a similar way as you query an actual table. Following is an example for the same.

SQL > SELECT \* FROM CUSTOMERS\_VIEW;

This would produce the following result.

+----------+-----+

| name | age |

+----------+-----+

| Ramesh | 32 |

| Khilan | 25 |

| kaushik | 23 |

| Chaitali | 25 |

| Hardik | 27 |

| Komal | 22 |

| Muffy | 24 |

+----------+-----+

The WITH CHECK OPTION

The WITH CHECK OPTION is a CREATE VIEW statement option. The purpose of the WITH CHECK OPTION is to ensure that all UPDATE and INSERTs satisfy the condition(s) in the view definition.

If they do not satisfy the condition(s), the UPDATE or INSERT returns an error.

The following code block has an example of creating same view CUSTOMERS\_VIEW with the WITH CHECK OPTION.

CREATE VIEW CUSTOMERS\_VIEW AS

SELECT name, age

FROM CUSTOMERS

WHERE age IS NOT NULL

WITH CHECK OPTION;

The WITH CHECK OPTION in this case should deny the entry of any NULL values in the view's AGE column, because the view is defined by data that does not have a NULL value in the AGE column.

Updating a View

A view can be updated under certain conditions which are given below −

* The SELECT clause may not contain the keyword DISTINCT.
* The SELECT clause may not contain summary functions.
* The SELECT clause may not contain set functions.
* The SELECT clause may not contain set operators.
* The SELECT clause may not contain an ORDER BY clause.
* The FROM clause may not contain multiple tables.
* The WHERE clause may not contain subqueries.
* The query may not contain GROUP BY or HAVING.
* Calculated columns may not be updated.
* All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

So, if a view satisfies all the above-mentioned rules then you can update that view. The following code block has an example to update the age of Ramesh.

SQL > UPDATE CUSTOMERS\_VIEW

SET AGE = 35

WHERE name = 'Ramesh';

This would ultimately update the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 35 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

Inserting Rows into a View

Rows of data can be inserted into a view. The same rules that apply to the UPDATE command also apply to the INSERT command.

Here, we cannot insert rows in the CUSTOMERS\_VIEW because we have not included all the NOT NULL columns in this view, otherwise you can insert rows in a view in a similar way as you insert them in a table.

Deleting Rows into a View

Rows of data can be deleted from a view. The same rules that apply to the UPDATE and INSERT commands apply to the DELETE command.

Following is an example to delete a record having AGE = 22.

SQL > DELETE FROM CUSTOMERS\_VIEW

WHERE age = 22;

This would ultimately delete a row from the base table CUSTOMERS and the same would reflect in the view itself. Now, try to query the base table and the SELECT statement would produce the following result.

+----+----------+-----+-----------+----------+

| ID | NAME | AGE | ADDRESS | SALARY |

+----+----------+-----+-----------+----------+

| 1 | Ramesh | 35 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

+----+----------+-----+-----------+----------+

Dropping Views

Obviously, where you have a view, you need a way to drop the view if it is no longer needed. The syntax is very simple and is given below −

DROP VIEW view\_name;

Following is an example to drop the CUSTOMERS\_VIEW from the CUSTOMERS table.

DROP VIEW CUSTOMERS\_VIEW;

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

Screenshot (57)

StudentMarks

Screenshot (58)

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:

Screenshot (57)

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:

Screenshot (64)

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:

Screenshot (59)

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.

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.

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,coulmn2,..

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:

Screenshot (60)

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 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:

Screenshot (62)

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:

Screenshot (57)

WITH CHECK OPTION

The WITH CHECK OPTION clause in SQL is a very useful clause for views. It is applicable to a 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.