**Views in SQL**

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;**