**Views**

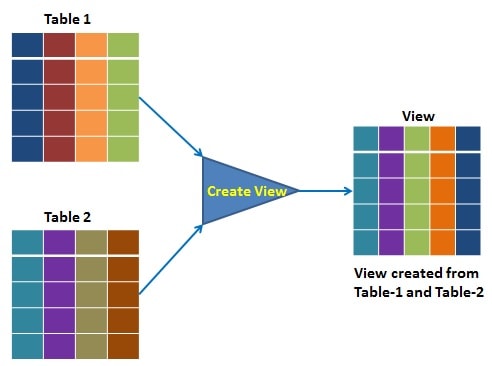
Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition.

Syntax

## CREATE [OR REPLACE] VIEW view\_name AS

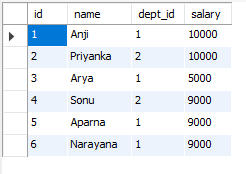
## SELECT columns

FROM tables [WHERE conditions];



Example

Step-1 select \*from employee; (already created data)



Step-2 CREATE VIEW view\_name AS

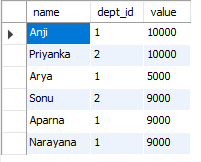
SELECT

name, dept\_id, salary AS value

FROM

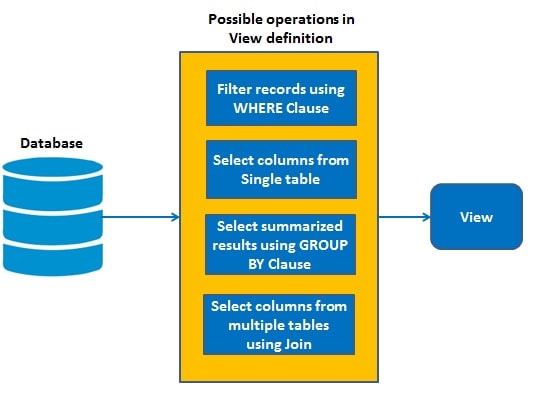
employee;

step-3 select \* from view\_name;



The view has primarily two purposes:

* Simplify the complex SQL queries.
* Provide restriction to users from accessing sensitive data.



**Advantages**

* Enforce Business Rules: By placing complicated or misunderstood business logic into the view, ***you can be sure to present a unified portrayal of the data***which increases use and quality.
* Security: Each user can be given permission to access the database only through a small set of views that contain the specific data the user is authorized to see, thus restricting the user's access to stored data
* Query Simplicity

A view can draw data from several different tables and present it as a single table, turning multi-table queries into single-table queries against the view.

* Structural simplicity

Views can give a user a "personalized" view of the database structure, presenting the database as a set of virtual tables that make sense for that user.

* Consistency

A view can present a consistent, unchanged image of the structure of the database, even if the underlying source tables are split, restructured, or renamed.

* Data Integrity

If data is accessed and entered through a view, the DBMS can automatically check the data to ensure that it meets the specified integrity constraints.

* Logical data independence.

View can make the application and database tables to a certain extent independent. If there is no view, the application must be based on a table. With the view, the program can be established in view of above, to view the program with a database table to be separated.

Disadvantages of views

* Performance

Views create the appearance of a table, but the DBMS must still translate queries against the view into queries against the underlying source tables. If the view is defined by a complex, multi-table query then simple queries on the views may take considerable time.

* Update restrictions

When a user tries to update rows of a view, the DBMS must translate the request into an update on rows of the underlying source tables. This is possible for simple views, but more complex views are often restricted to read-only.

Types of views in sql

Views are of two types-

* System defined view
* User-defined view

System defined view -- Already defined in the system

User-defined view -- Defined by the user

### **System Defined View**

Three types-

* Information schema view
* Catalog view
* Dynamic management view

|  |  |  |
| --- | --- | --- |
| **Information schema view** | **Catalog view** | **Dynamic management view** |
| They are twenty in number. They are used to show information about a database. | hows information that helps to understand the health of the database. | out administrator information of a database. |
| They are located under the master database. They have a prefix called INFORMATION\_SCHEMA. | They are also located under the master database. | Returns information that can be used to monitor the health of a server. |

User-defined view

* **Simple View:** A view based on only a single table, which doesn't contain GROUP BY clause and any functions.
* **Complex View:** A view based on multiple tables, which contain GROUP BY clause and functions.
* **Inline View:** A view based on a subquery in FROM Clause, that subquery creates a temporary table and simplifies the complex query.
* **Materialized View:** A view that stores the definition as well as data. It creates replicas of data by storing it physically.

EXAMPLE FOR SIMPLE VIEW

Step-1 : CREATE or replace VIEW view\_name AS

SELECT

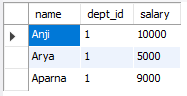
name, dept\_id, salary

FROM

employee where dept\_id=1;

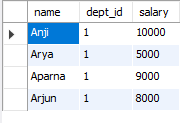
step-2

SELECT \* FROM view\_name;



Insert into sample view

insert into view\_name values ('Arjun','1','8000');



EXAMPLE FOR COMPLEX VIEW

CREATE OR REPLACE VIEW com\_view AS

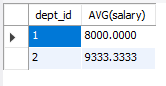
SELECT

dept\_id, AVG(salary)

FROM

employee

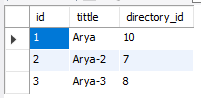
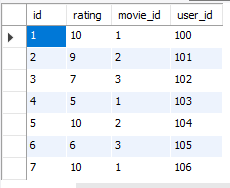
GROUP BY dept\_id;



Note: complex view is only used to read data

EXAMPLE FOR INLINE VIEW

Creating two tables

SELECT

MAX(avg\_rating) AS max\_avg\_rating

FROM (

SELECT

AVG(rating) AS avg\_rating

FROM rating

GROUP BY movie\_id

) AS avg\_movie\_ratings;



SYNTAX FOR MATERIALIZED VIEW

Create materialized view v build [clause]

Refresh [type] on [triggers] as<,query expression>