****

USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science

**Fall 2019**

**Lab # 9**

**Objective:** Working on Views

.

**Name of Student:**

**E**

**Roll No: Section:**

02-DEC-2019

**Date of Experiment:**

**Date Submitted:**

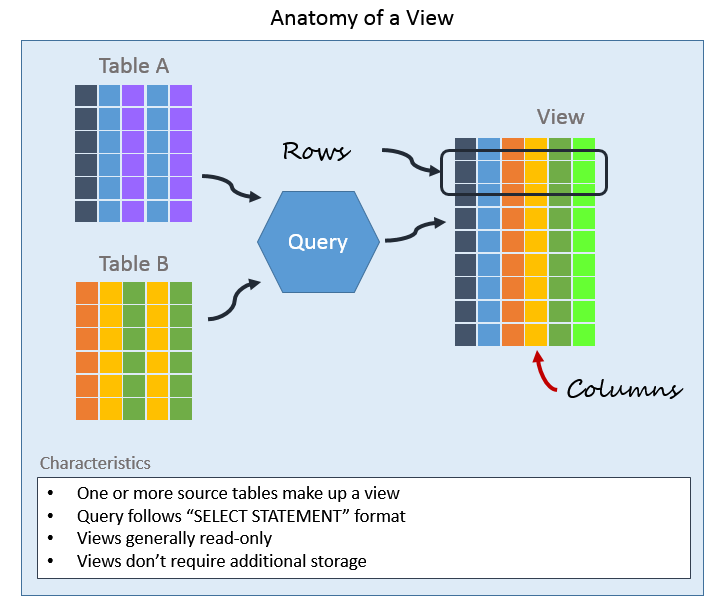
Marks Obtained/Remarks: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Creating Views**

**What is a View?**

A view is a logical table based on other tables or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed. The tables in which a view is based are called *base tables*. The view is stored as a SELECT statement in the data dictionary.



**Figure 9.1**

**Why Use Views?**

* Views can represent a subset of the data contained in a table. Consequently, a view can limit the degree of exposure of the underlying tables to the outer world: a given user may have permission to query the view, while denied access to the rest of the base table.
* Views can [join](https://en.wikipedia.org/wiki/Join_(SQL)) and simplify multiple tables into a single virtual table.
* Views can act as aggregated tables, where the [database engine](https://en.wikipedia.org/wiki/Database_engine) aggregates data ([sum](https://en.wikipedia.org/wiki/Summation), [average](https://en.wikipedia.org/wiki/Average), etc.) and presents the calculated results as part of the data.
* Views can hide the complexity of data. For example, a view could appear as Sales2000 or Sales2001, transparently [partitioning](https://en.wikipedia.org/wiki/Partition_(database)) the actual underlying table.
* Views take very little space to store; the database contains only the definition of a view, not a copy of all the data that it presents.
* Depending on the [SQL](https://en.wikipedia.org/wiki/SQL) engine used, views can provide extra security.
* To present different views of the same data to different users.

**Simple and Complex Views**

There are two classifications for views:

Simple and Complex.

The basic difference is related to the DML (insert, update and delete) operations.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Simple Views** | **Complex Views** |
| Number of tables | One | One or more |
| Contain functions | No | Yes |
| Contain groups of data | No | Yes |
| DML through view | Yes | Not always |

**Table 9.1**

**Creating a View**

We can create a view by embedding a subquery within the CREATE VIEW statement. The syntax is as follows:-

CREATE [OR REPLACE] VIEW view

AS subquery;

For example, to create a view, EMPVU10, that contains the employee number, name and job title for all the employees in department 10.

|  |
| --- |
| CREATE VIEW emp\_view\_dept10  AS SELECT empno, ename, job  FROM emp  WHERE deptno = 10; |

We can display the structure of the view by using the SQL\*Plus DESCRIBE command as follows:-

DESCRIBE emp\_view\_dept10

We can also create views by using column aliases in the subquery.

|  |
| --- |
| CREATE VIEW sal\_view\_dept30  AS SELECT empno EMPLOYEE\_NUMBER, ename NAME, sal SALARY  FROM emp  WHERE deptno = 30; |

Now select the columns from this view by the given alias names.

The data from the view would be retrieved as follows:- SELECT \* FROM sal\_view\_dept30;

**Views in the data dictionary**

Once a view has been created, we can query the data dictionary table called USER\_VIEWS to see the name of the view and the view definition. The text of the SELECT statement that constitutes the view is stored in a LONG column.

**Creating a Complex View**

A complex view contains columns from multiple tables and may also include group functions.

* To create a complex view to show employee number, employee name and department name, we would have to join EMP and DEPT tables as follows:-

|  |
| --- |
| CREATE VIEW EMP\_DEPT  AS  SELECT EMPNO, ENAME, DNAME  FROM EMP, DEPT  WHERE EMP.DEPTNO = DEPT.DEPTNO; |

* To create a complex view that contains group functions to display values from two tables.

|  |
| --- |
| CREATE VIEW dept\_sum\_vu (name, minsal, maxsal, avgsal)  AS SELECT d.dname, MIN(e.sal), MAX(e.sal), AVG(e.sal)  FROM EMP e, DEPT d  WHERE e.DEPTNO = d.DEPTNO  GROUP BY d.dname; |

**Removing a View**

We can remove a view without losing data because a view is based on underlying tables in the database. The syntax is

DROP VIEW view;

For example, to drop the emp\_view\_dept10 view,

|  |
| --- |
| DROP VIEW emp\_view\_dept10 |

**Updating a View**

CREATE OR REPLACE VIEW view\_name AS  
SELECT column1,column2, ...  
FROM table\_name  
WHERE condition;

**Example:**

Now we want to add the "Category" column to the "Current Product List" view. We will update the view

with the following SQL:

|  |
| --- |
| CREATE OR REPLACE VIEW [Current ProductList] AS SELECT ProductID, ProductName, Category FROM Products WHERE Discontinued = No; |

**Updating a View**

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

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

**Exercise:**

1. Create views for following purposes (use tables from lab 5 for below tasks)
2. To display each designation and number of employees with that particular designation.

1. To display employee number, employee name, project title and employee performance in that project.

1. To display employee number, employee name and number of projects in which employee performance is *excellent*.
2. What are the different uses of views? Differentiate between simple and complex views.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\* \* \* \* \*