

Ex.No.: 5		CREATING VIEWS
Date:		

After the completion of this exercise, students will be able to do the following:

- Describe a view
- Create, alter the definition of, and drop a view
- Retrieve data through a view
- Insert, update, and delete data through a view
- Create and use an inline view

### View

A view is a logical table based on a table or another view. A view contains no data but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called base tables.

### Advantages of Views

- To restrict data access
- To make complex queries easy
- To provide data independence
- To present different views of the same data

### Classification of views

1. Simple view
2. Complex view

Feature	Simple	Complex
No. of tables	One	One or more
Contains functions	No	Yes
Contains groups of data	No	Yes
DML operations thr' view	Yes	Not always

### Creating a view

#### Syntax

CREATE OR REPLACE FORCE/NOFORCE VIEW view\_name AS Subquery WITH CHECK  
OPTION CONSTRAINT constraint WITH READ ONLY CONSTRAINT constraint;

**FORCE** - Creates the view regardless of whether or not the base tables exist.

**NOFORCE** - Creates the view only if the base table exist.

WITH CHECK OPTION CONSTRAINT-specifies that only rows accessible to the view can be inserted or updated.

WITH READ ONLY CONSTRAINT-ensures that no DML operations can be performed on the view.

**Example: 1 (Without using Column aliases)**

Create a view EMPVU80 that contains details of employees in department80.

**Example 2:**

```
CREATE VIEW empvu80 AS SELECT employee_id, last_name, salary FROM employees  
WHERE department_id=80;
```

**Example:1 (Using column aliases)**

```
CREATE VIEW salvu50  
AS SELECT employee_id, id_number, last_name NAME, salary *12 ANN_SALARY  
FROM employees  
WHERE department_id=50;
```

**Retrieving data from a view**

**Example:**

```
SELECT * from salvu50;
```

**Modifying a view**

A view can be altered without dropping, re-creating.

**Example: (Simple view)**

Modify the EMPVU80 view by using CREATE OR REPLACE.

```
CREATE OR REPLACE VIEW empvu80 (id_number, name, sal, department_id)  
AS SELECT employee_id, first_name, last_name, salary, department_id  
FROM employees  
WHERE department_id=80;
```

### Example: (complex view)

```
CREATE VIEW dept_sum_vu (name, minsal, maxsal, avgsal)
AS    SELECT d.department_name, MIN(e.salary), MAX(e.salary), AVG(e.salary)
FROM employees e, department d
WHERE e.department_id=d.department_id
GROUP BY d.department_name;
```

### Rules for performing DML operations on view

- Can perform operations on simple views
- Cannot remove a row if the view contains the following:
  - Group functions
  - Group By clause
  - Distinct keyword
- Cannot modify data in a view if it contains
  - Group functions
  - Group By clause
  - Distinct keyword
  - Columns contain by expressions
- Cannot add data thr' a view if it contains
  - Group functions
  - Group By clause
  - Distinct keyword
  - Columns contain by expressions
  - NOT NULL columns in the base table that are not selected by the view

### Example: (Using the WITH CHECK OPTION clause)

```
CREATE OR REPLACE VIEW empvu20
AS    SELECT *
FROM employees
WHERE department_id=20
WITH CHECK OPTION CONSTRAINT empvu20_ck;
```

Note: Any attempt to change the department number for any row in the view fails because it violates the WITH CHECK OPTION constraint.

### Example – (Execute this and note the error)

```
UPDATE empvu20 SET department_id=10 WHERE employee_id=201;
```

### Denying DML operations

Use of WITH READ ONLY option.

Any attempt to perform a DML on any row in the view results in an oracle server error.

Try this code:

```
CREATE OR REPLACE VIEW empvu10(employee_number, employee_name, job_title)
AS SELECT employee_id, last_name, job_id
FROM employees
WHERE department_id=10
WITH READ ONLY;
```

Find the Solution for the following:

1. Create a view called EMPLOYEE\_VU based on the employee numbers, employee names and department numbers from the EMPLOYEES table. Change the heading for the employee name to EMPLOYEE.

```
CREATE OR REPLACE VIEW EMPLOYEE_VU AS
SELECT Employeeid, Lastname || ' ' || firstname AS
EMPLOYEE, Departmentid
FROM EMPLOYEES;
```

2. Display the contents of the EMPLOYEES\_VU view.

```
SELECT *
FROM EMPLOYEES_VU;
```

3. Select the view name and text from the USER\_VIEWS data dictionary views.

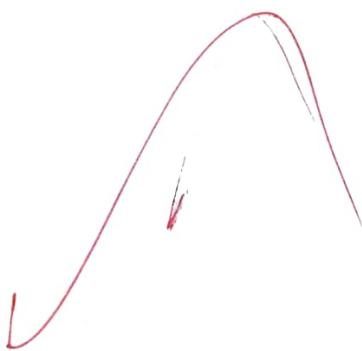
```
SELECT view-name, text
FROM user-views
WHERE view-name = 'EMPLOYEES_VU';
```

4. Using your EMPLOYEES\_VU view, enter a query to display all employees names and department.

```
SELECT EMPLOYEE, Departmentid
FROM EMPLOYEE_VU;
```

5. CREATE OR REPLACE VIEW DEPT\_50  
(EMPNO, EMPLOYEE, DEPTNO) AS  
SELECT Employeeid, Lastname || ', ' || Firstname,  
Departmentid  
FROM EMPLOYEES  
WHERE Departmentid = 50  
WITH CHECK OPTION  
CONSTRAINT dept50-chk;

8. CREATE OR REPLACE VIEW SALARYVU AS  
SELECT  
e. Lastname AS Employee,  
d. DepartmentName AS Department,  
e. Salary AS salary,  
j. GradeLevel AS Grade  
FROM employee e  
JOIN DEPARTMENTS d ON e.departmentid =  
d.departmentid  
JOIN JOB\_GRADES j  
ON e.salary BETWEEN j.LowestSalary  
AND j.HighestSalary;



5. Create a view named DEPT50 that contains the employee number, employee last names and department numbers for all employees in department 50. Label the view columns EMPNO, EMPLOYEE and DEPTNO. Do not allow an employee to be reassigned to another department through the view.

6. Display the structure and contents of the DEPT50 view.

```
DESC DEPT 50;  
SELECT * FROM DEPT 50;
```

7. Attempt to reassign Matos to department 80.

```
UPDATE DEPT 50  
SET DEPTNO = 80  
WHERE EMPLOYEE LIKE " Matos ,%";
```

8. Create a view called SALARY\_VU based on the employee last names, department names, salaries, and salary grades for all employees. Use the Employees, DEPARTMENTS and JOB\_GRADE tables. Label the column Employee, Department, salary, and Grade respectively.

*Bfj*

Evaluation Procedure	Marks awarded
Query(5)	
Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	

RESULT :  
Thus all the above SQL statements  
were executed.