

EXERCISE-11

CREATING VIEWS

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.

- 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 view Employee_id as select employee_id, employee_name as employee, dept_id from EMPLOYEES;

2. Display the contents of the EMPLOYEES_VU view.

*Select * from EMPLOYEE_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 = 'EMPLOYEE_VU';

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

Select employee, department-id from EMPLOYEE_VU;

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.

Create view Dept AS select employee-id AS Emp-No, last-name AS employee, dept-id AS DeptNo from employee where dept-id = 50; constraint dept_50 check (dept-id = 50) with check option

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 Dept No = 80 where EMPLOYEE = 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.

*Create view salary-vu AS
select e.last-name AS employee, d.dept-name AS
Department, e.salary AS salary, j.grade-level AS
Grade
from Employee
Join dept d ON e.department-id = d.dept-id
Join Job_grades j ON e.salary BETWEEN j.low-sal AND j.high-sal;*

Evaluation Procedure	Marks awarded
Query(5)	5
Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	R Pn 9/9/25