**VIEWS IN ORACLE**

After a table is created and populated with data, it may become necessary to prevent all users from accessing all column of a table, for data security reasons. This would mean creating several tables having the appropriate number of columns and assigning specific users to each table, as required. This will answer data security requirements very well but will give rise to a great deal of redundant data being resident in tables, in the database.

To reduce redundant data to the minimum possible, Oracle allows the creation of an object called a View.

**VIEW:**

* It can be defined as a stored select statement.
* It will not hold data or store data by itself.
* It is a logical table based on one or more tables or views.
* View can be created based on a table called a BASE TABLE.
* View can be created as Object Views or Relational Views.
* The Object Views support

\*LOBs \*Object Types \*REFs

\*Nested Tables \*Varray

* DML, DESC, SELECT allowed on views.

**PREREQUISITES:**

* Should have **CREATE VIEW** System privilege.
* To create a subview, we need **UNDER ANY VIEW** system privilege
* The OWNER of the Schema should have SELECT,INSERT,UPDATE or DELETE Rows from all tables or Views on which the VIEW is based.
* The above privileges must be granted the privileges directly, rather than a role.

**Advantages of Views:**

•Provides high security while sharing data between users

•USER\_VIEWS hold the details of views. (date dictionary table).

•View can be used to make simple queries to retrieve the result of complicated queries.

•Views provide groups of user’s accesses according to their particular criteria.

* Views are stored in user\_views, tab, user\_objects, dba\_objects.

**Syntax:**

**CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW view\_name**

**[(AliasName[,AliasName])] As SubQuery**

**[WITH {CHECK OPTION/READ ONLY}**

**CONSTRAINT ConstraintName];**

\***OR REPLACE:**

* Re-creates the view if it already exists.

\***FORCE:**

* Specifies the View Should be create if the base table does not Exists.

**NOFORCE:**

* Creates the view only if the base tables exist, which is the default.

**Alias Name:**

* Specifies names for the expressions selected by the view query.

**WITH CHECK OPTION:**

* Specifies that only rows accessible to the view can be INSERTED or UPDATED or DELETED.

**CONSTRAINT:**

* Constraint is the name assigned to the WITH CHECK OPTION or READ ONLY constraint.

**WITH READ ONLY:**

* Ensures that no DML operations can be performed on this view.

**TYPES OF VIEWS:**

**Simple Views:**

• The view which is created without following clauses.

* Join condition.
* Group by
* Having clause
* Set operators
* Distinct
* Group function
* If view contain simple subquery i.e also called as simple view.

Note: through simple view we can perform DML operation.

**Complex View**

* The view which is created with any restriction clause is called complex view.
* Complex Views can contain Sub Queries.
* The Subquery can contain,Queries
* That Retrieve from multiple Base Tables
* Group Rows using a GROUP BY or DISTINCT Clause.
* Contain a Functional call.

Sql>Create VIEW Employees As Select Empno "Emp Id", Ename Name,

Sal "Basic Salary",

Job Designation From Emp;

**Selecting Data From A View:**

Sql> Select Name, Job From Employees;

Sql> Select "Emp Id", name,"Basic Salary"\*12 From Employees;

Sql> select "Emp Id",Name,

To\_char("Basic Salary",'99,99,999.99') Monthly,

"Basic Salary"\*12 Annual From Employees

Where "Basic Salary">2500;

**Complex View:**

**Example:**

Sql>Create VIEW EmpInfo As

Select E.Empno EmployeeNo, E.Ename Name, D.Deptno Departmentld, D.Dname DepartmentName From Emp E, Dept D

Where D.deptno = E.Deptno ORDER BY D.Deptno;

Sql>Create VIEW EmpGrades As Select E.Ename Name, E.Sal Basic, S.Grade Garde

From Emp E,Salgrade S

Where E.Sal BETWEEN S.Losal AND S.Hisal ORDER BY S.Grade;

Sql> Create VIEW EmpManagers As Select Rownum SeriaINo, Initcap(E.Ename)||'Works Under'|| M.Ename "Employee And Managers" From Emp E,Emp M Where E.Mgr=M.Empno;

Sql>Create or Replace VIEW EmpAccount As

Select Ename,Deptno,Sal Monthly, Sal\*12 Annual From Emp Where Deptno=(Select Deptno

From Dept Where Dname='ACCOUNTING') ORDER BY Annual;

Sql> Create VIEW PayInfo As

Select Empno Ecode,Sal Basic,

Sal\*0.25 Da,Sal \* 0.35 HRA,

Sal \* 0.12 PF, Sal + Sal \*0.25 + Sal \*0.35-sal \*0.12 GROSS From Emp;

Sql> Create VIEW Dept\_Analysis As SELECT Deptno,count(\*) NoEmp,

MIN(Sal) Low\_Pay,MAX(Sal) High\_Pay,SUM(Sal) Tot\_Pay,AVG(Sal) Avg\_Pay FROM Emp Group By Deptno

Sql> Create or Replace VIEW CumSum As Select B.Sal,SUM(A.Sal) As Cum\_Sal From Emp A, Emp B Where A.RowId<=B.RowId

GROUP BY B.RowId,B.Sal;

Sql>Create or Replace VIEW OrgDesignations As Select Job From Emp Where Deptno = 10 UNION Select Job From Emp Where Deptno IN(20,30);

**VIEW in Data Dictionary:**

• Once the View has been created, we can query upon the DATA DICTIONARY table called USER\_VIEWS to see the Name and definition of the View.

• The TEXT of the SELECT statement that constitutes the VIEW is stored in a LONG Column.

**Retrieving Data from a View:**

• Retrieves the VIEW definition from the Data Dictionary table USER\_VIEWS.

• Check the Access privileges for the view.

• We can retrieve data from a view as you would from any table.

• We can display either the contents of the entire view or just specific rows and columns.

Sql> Select GRANTEE From AII\_Tab\_Privs

Where Grantor='C##ANKUR' AND Table\_Name = ' OrgDesignations';

**Creating Views With Columns Declarations:**

* When a VIEW is being created, we can specify the Names of the Columns, that it can project, along with the VIEW'S definition.

Sql>CREATE OR REPLACE VIEW EmpV

(id\_number, name, sal, department\_id) AS SELECT empno, ename, sal, deptno FROM emp WHERE deptno =30;

Sql>Select \*From EmpV;

Sql>Create View VDept20 As Select \* From Emp

Where Deptno=20;

**Inserting The Total Data of Employees From Department 20 Using View:**

Sql>Create table dept20

As Select \* from VDept20;

Sql> Create Table EmpGrades(Employee, Designation, BasicSalary, Garde )As

Select Ename, Job, Sal, Grade From Emp E, Salgrade S

Where E.Sal BETWEEN S.Losal AND S.Hisal;

**DROPING A VIEW:**

* A VIEW can remove without losing data, because a view is based underlying tables in the database.
* The DROP VIEW statement is used to remove a View permanenty.
* Dropping a View has no affect on the tables upon which the View is created.
* VIEWS or APPLICATIONS based on deleted Views become invalid.
* We need DROP ANY VIEW privileges to remove the VIEWS.

**Syntax:**

>DROP VIEW view\_name;

**Example:**

>DROP VIEW EmpGrades;

**Rules for Performing DML Operations On A View:**

* DML operations can be performed upon a table through VIEW.
* A row can be removed from a VIEW unless it contains.
* JOIN Condition.
* GROUP By Clause
* HAVING Clause
* SET Operators
* DISTINCT/UNIQUE Key Word
* The Column defined by Expressions
* Data can be added through a view, unless it contains any of the above rules and there are NOT NULL Columns.

**Example:**

Sql>Create View Vsalval As

Select Empno,Ename,Sal,Job From Emp Where Sal>7000;

Sql>Desc Vsalval;

* By executing this statement empno, ename, sal,job will be displayed because the Vsalval view contains four columns only.

**Adding the WITH READ ONLY option:**

• We can ensure that no DML operations occur on your view by creating it with the WITH READ ONLY option.

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

**Example**:

Sql> Create Or Replace View empvu10

(employee\_number, emplyoyee\_name, job\_title)

AS Select "Emp Id",name,designation

from employees

Where designation='SALESMAN'

WITH READ ONLY;

**View On View:**

• We can create view on their view.

Sql>Create View V1

As

Select \*from emp;

Sql> Create view V2

As

Select Empno,Ename,Job,Sal\*0.22 Bonus

From v1;

Sql > Drop view Emp;

**MATERIALIZED VIEWS:**

* It is introduced in Oracle (8i).
* It holds data.
* No DML allowed.
* Equivalent to snap-shop (DBA object)
* Materialized Views are used in DATA WAREHOUSES.
* They are used to increase the speed of queries on Very large databases.

**Query Rewrite:**

• Materialized Views improve query performance by PRECALCUTING Expensive JOIN and AGGREGATION operations on the DATABASE PRIOR to Execution time and stores the results in the DATABASE.

• The Query OPTIMIZER can make use of MATERIALIZED VIEWS by automatically recognizing when an Existing MATERIALIZED VIEW can and should be used to satisfy a Request.

• After above process is completed then QUERY OPTIMIZER transparently rewrites the request to use the MATERIALIZED VIEW.

• QUERIES are then directed to the MATERIALIZED VIEWS and not to the underling DETAIL TABLES or VIEWS.

• REWRITING QUERIES to use MATERIALIZED VIEWS rather than detail relations, results in a significant performance gain.

**PREREQUISITES FOR MATERIALIZED VIEWS PRIVILEGES:**

• Needs create materialized view privilege.

• Grant Query rewrite to scott;

• Alter Session Set QUERY REWRITE ENABLED=TRUE;

**Set the InitSid.ORA File:**

OPTIMIZER\_MODE = CHOOSE

JOB\_QUEUE\_INTERVAL=3600

JOB\_QUEUE\_PROCESSES=1

QUERY\_REWRITE\_ENABLED=TRUE

QUERY\_REWRITE\_INTEGRITY=ENFORCED

**Syntax:**

Sql>Create MATERIALIZED VIEW <View name>

[ENABLED QUERY REWRITE]

[Refresh On Commit]

As

<Select statements>;

**Example:**

Sql> Create MATERIALIZED VIEW Emp\_Dno

refresh on commit

As

Select Deptno,Sum(Sal),Count(Empno)

From Emp

Group By Deptno;

**Creating Optimizer Statistics and Refreshing Materialised Views:**

Sql> EXECUTE DBMS\_MVIEW.REFRESH('Emp\_Dno');

**OCP Eaxmples:**

1)Which two statements about views are true?

A. They hide data complexity

B. DML operation On based table of view can not retrieve by Simple View.

C. You can Create View with out a Base Table.

D. DML operation can perform through Materialized View when it create REFRESH ON COMMIT option.

2) Which two statements about views are true? (Choose two.)

A. A view can be created as read only.

B. A view can be created as a join on two or more tables.

C. A view cannot have an ORDER BY clause in the SELECT statement.

D. A view cannot be created with a GROUP BY clause in the SELECT

statement.

1. A view must have aliases defined for the column names in the

SELECT statement.

3) Which View statements give the out in ascending 0rder by Ename

A. CREATE or replace VIEW my\_view AS SELECT \* FROM emp ORDER BY ename;

B. CREATE or replace VIEW my\_ view AS ( SELECT\*FROM emp ORDER BY ename);

C. A And B

D.None

4) You created a view called EMP\_DEPT\_VU that contains three columns from the EMPLOYEES and DEPARTMENTS tables:

EMPLOYEE\_ID, EMPLOYEE\_NAME AND DEPARTMENTAL\_NAME.

The DEPARTMENT\_ID column of the EMPLOYEES table is the foreign key to the primary key

DEPARTMENTJD column of the DEPARTMENTS table.

You want to modify the view by adding a fourth column MAM

of NUMBER data type from the EMPLOYEES tables.

How can you accomplish this task?

1. ALTER VIEW EMP\_dept\_\_vu (ADD manger\_id NUMBER);
2. MODIFY VIEW EMP\_dept\_vu (ADD manager\_jd NUMBER);
3. ALTER VIEW emp\_dept\_vu AS

SELECT emplyoyee\_id, employee\_name,

Department\_name, manager\_id

FROM employee e, department d

WHERE e.department\_id=d.department\_id;

E. CREATE OR REPLACE VIEW emp\_dept\_vu AS

SELECT employee\_id, employee\_name,

Department\_name, maanger\_id

FROM employee e, department d

WHERE e.department\_id=d.department;

1. YOU must remove the existing view first ,and then run the CREATE VIEW

Command with a new column list to modify a view.

5) Which two statements about views are true?

A. They hide data compiexit.

B. DML operation On based table of view can not retrieve by Simple View.

C. You can Create View with out a Base Table.

D. DML operation can perform through Materialized View when it create

REFRESH ON COMMIT option.

6) What is necessary for your query on an existing view to execute successfully?

A. The underlying tables must have data.

B. You need SELECT privileges on the view.

C. The underlying tables must be in the same schema.

D. You need SELECT privileges only on the underlying tables.

7) Which view statements is true?

A. Operation performed on view dose not affect the base table.

B. View cannot be created without a table.

C. Only Key Preserved table of Join view can be updated.

D. View can be thought of as a Stored Query or Virtual Table

8) To impose constraint on view\_\_\_\_\_\_\_\_\_\_option is used.

A. With read only

B. With Check Option

C Join View

D. None

9)You need to create a view EMP\_VU. The view should allow the users to manipulate the records of only the employees that are working for departments 10 or 20

Which SQL statement would you use to create the,

A. CREATE VIEW emp\_.vu AS SELECT \* FROM employees

WHERE departmentjd IN (10,20);

B. CREATE VIEW emp\_vu AS

SELECT \* FROM employees

WHERE department\_id IN (10,20)

WITH READ ONLY;

1. CREATE VIEW emp, vu AS

SELECT \* FROM employees

WHERE department\_id IN (10,20)

WITH CHECK OPTION;

1. CREATE FORCE VIEW emp\_vu AS

SELECT f FROM employees

WHERE departmentjd IN (10,20);

1. CREATE FORCE VIEW emp\_vu AS

SELECT \* FROM employees

WHERE department id IN (10,20)

NO UPDATE;

10) Examine the structure if the EMPLOYEES table:

Column name Data Type Remarks

EMPLOYEE ID NUMBER NOT NULL, Primary Key

EMP\_NAME VARCHAR2(30)

JOB\_ID VARCHAR2(20) NOT NULL

SAL NUMBER

MGR\_ID NUMBER References EMPLOYEE\_ID column

DEPARTMENT\_ID NUMBER Foreign key to DEPARTMENT\_ID

column of the DEPARTMENTS table

You need to create a view called EMP\_VU that allows the user to insert rows through the view. Which SQL statement, when used to create the EMP VU view allows the user to insert rows?

1. CREATE VIEW emp Vu AS

SELECT employee\_id, emp\_name,

Department\_id

FROM employees

WHERE mgr\_id IN (102, 120);

1. CREATE VIEW emp\_Vu\_AS

SELECT employee\_id, emp\_name, job\_id

Department\_id

FROM employees

WHERE mgr\_id IN (102, 120);

1. CREATE VIEW emp\_Vu\_AS

SELECT department\_id, SUM(sal) TOTALSAL

FROM employees

WHERE mgr\_id IN (102, 120);

GROUP BY department\_id;

1. CREATE VIEW emp\_Vu\_AS

SELECT employee\_id, emp\_name, job\_id

DISTINCT department\_id

FROM employees