Fair Use Notice:

DATABASE SYSTEMS (SW215)

VIEWS

By : HIRA NOMAN

VIEWS

- A view is a logical representation of another table or combination of tables.
- A view derives its data from the tables on which it is based. These tables are called base tables. Base tables might in turn be actual tables or might be views themselves.
- All operations performed on a view affect the base table of the view.
- You can use views in almost the same way as tables. You can query, update, insert into, and delete from views, just as you can standard tables.
- Views can provide a different representation (such as subsets or supersets) of the data that resides within other tables and views.
- Views are very powerful because they allow you to tailor the presentation of data to different types of users.

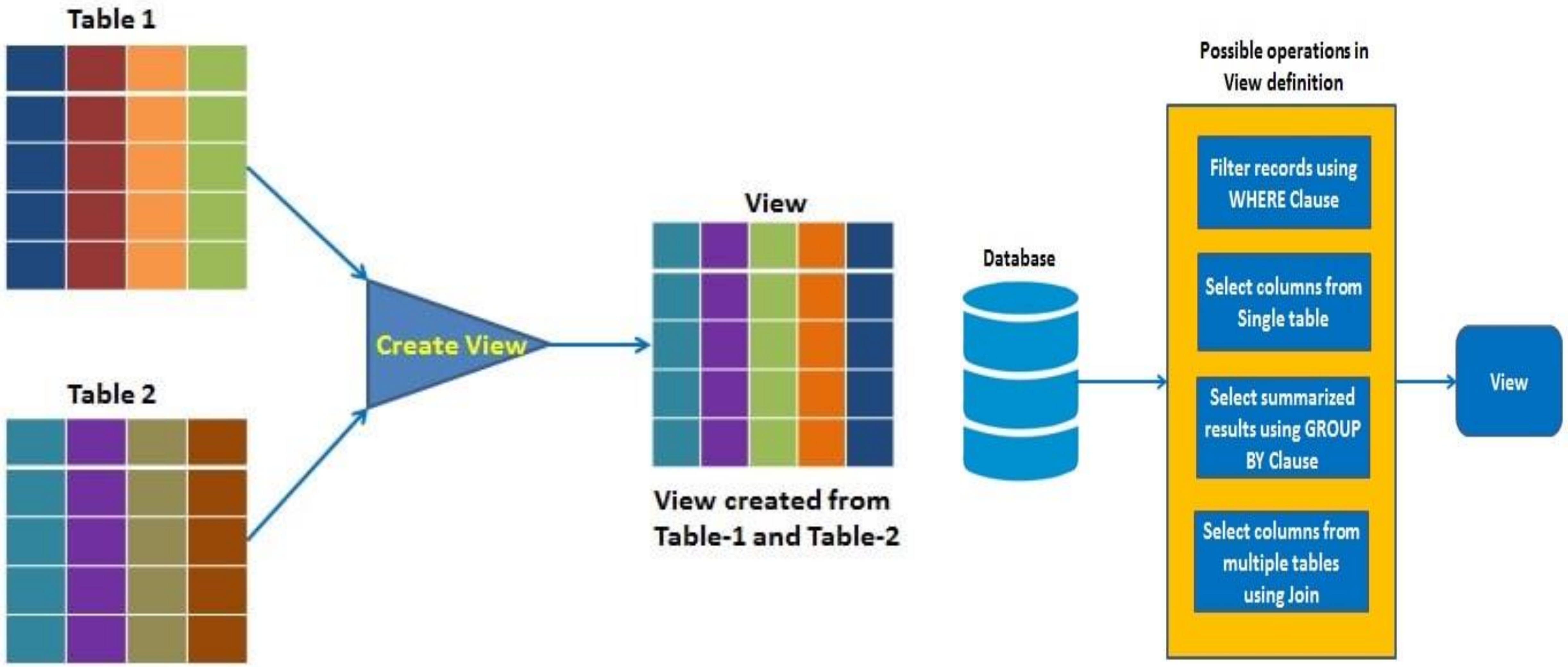


Table 1
Table 2
Create View
View
View created from Table-1 and Table-2
Database
Possible operations in View definition
Filter records using WHERE Clause
Select columns from Single table
Select summarized results using GROUP BY Clause
Select columns from multiple tables using Join
View

ORACLE VIEWS

When to use the Oracle view ?

- You can use views in many cases for different purposes. The most common uses of views are as follows:

1. Simplifying data retrieval.
2. Maintaining logical data independence.
3. Implementing data security.

1. **Simplifying data retrieval:**

- Views help simplify data retrieval significantly.
- First, you build a complex query, test it carefully, and encapsulate the query in a view. Then, you can access the data of the underlying tables through the view instead of rewriting the whole query again and again.

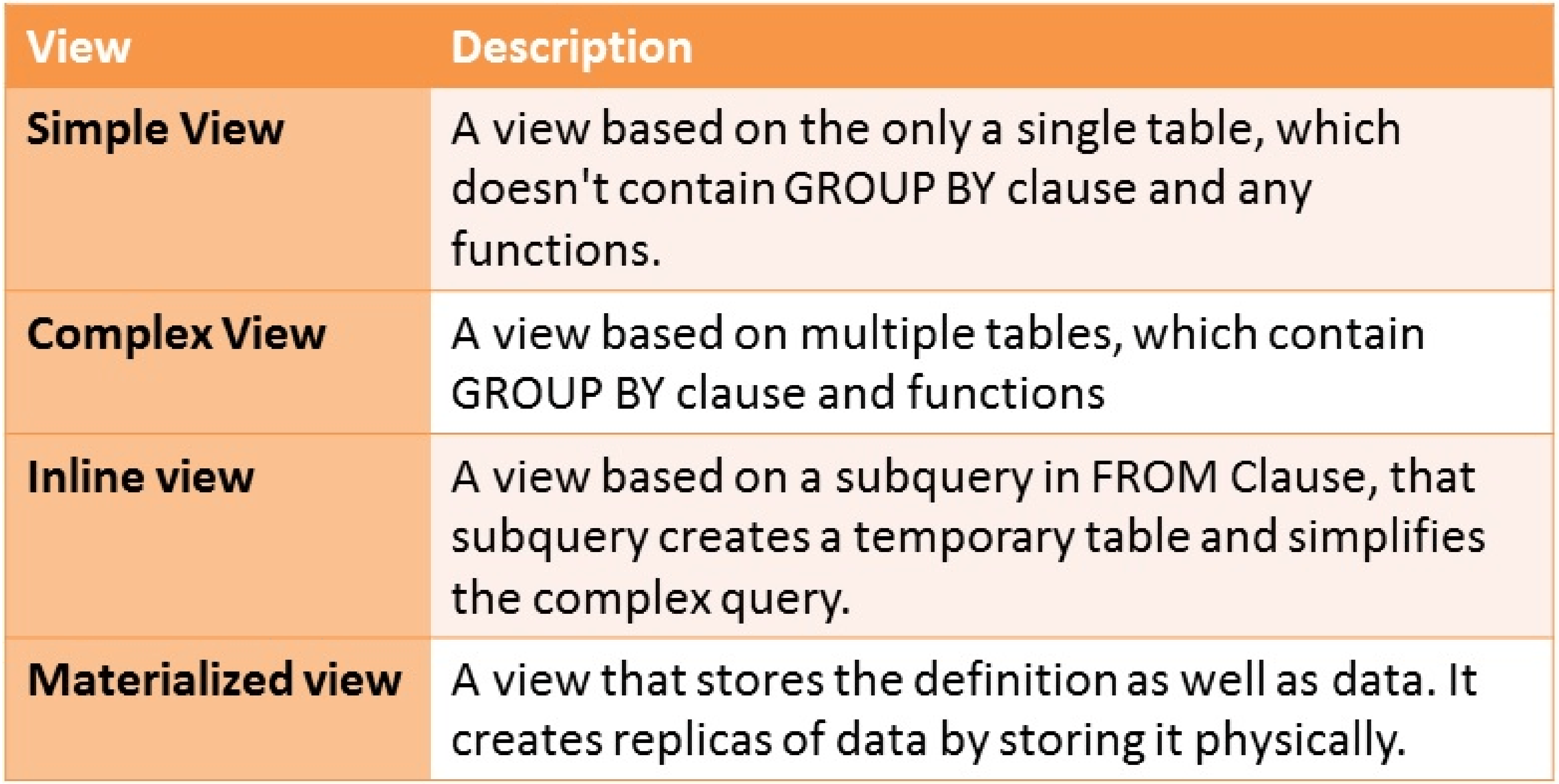
2. **Maintaining logical data independence:**

- You can expose the data from underlying tables to the external applications via views.
- Whenever the structures of the base tables change, you just need to update the view. The interface between the database and the external applications remains intact.
- The beauty is that you don’t have to change a single line of code to keep the external applications up and running.

3. Implementing data security:

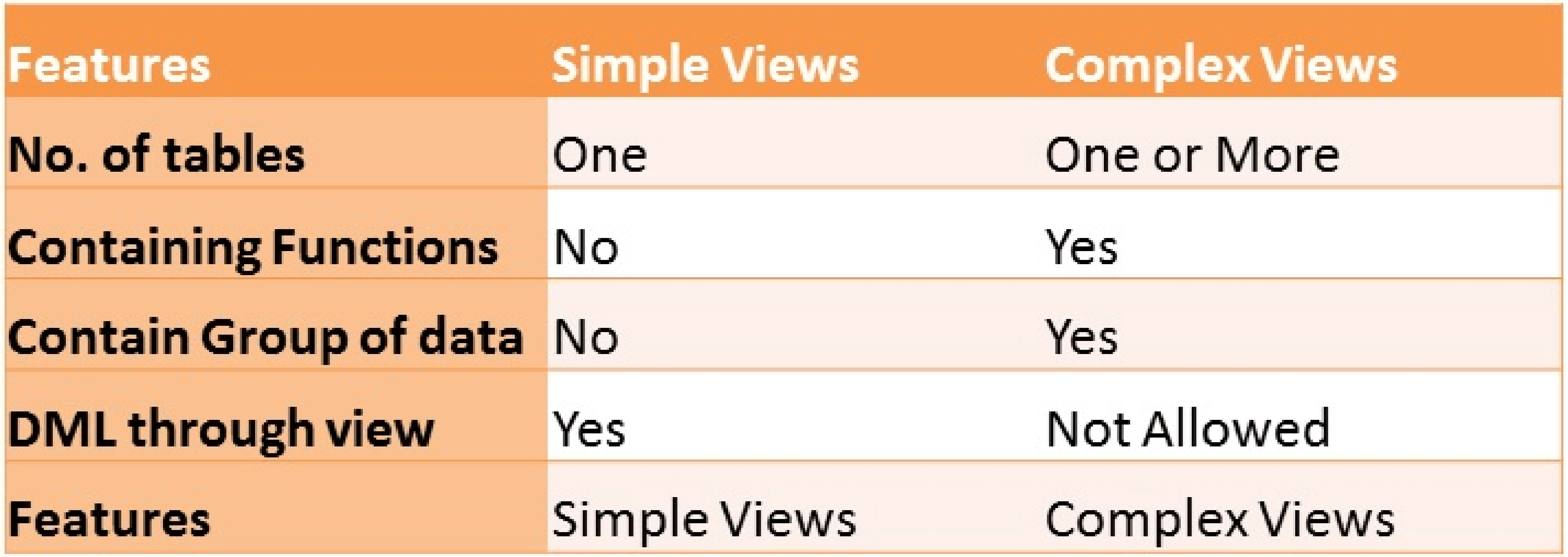
- Views allow you to implement an additional security layer. They help you hide certain columns and rows from the underlying tables and expose only needed data to the appropriate users.
- Oracle provides you the with GRANT and REVOKE commands on views so that you can specify which actions a user can perform against the view.
- Note that in this case, you don’t grant any privileges on the underlying tables because you may not want the user to bypass the views and access the base tables directly.

TYPES OF VIEWS



View	Description
Simple View	A view based on the 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.

SIMPLE VERSUS COMPLEX VIEWS



Features	Simple Views	Complex Views
No. of tables	One	One or More
Containing Functions	No	Yes
Contain Group of data	No	Yes
DML through view	Yes	Not Allowed
Features	Simple Views	Complex Views

INLINE VIEWS

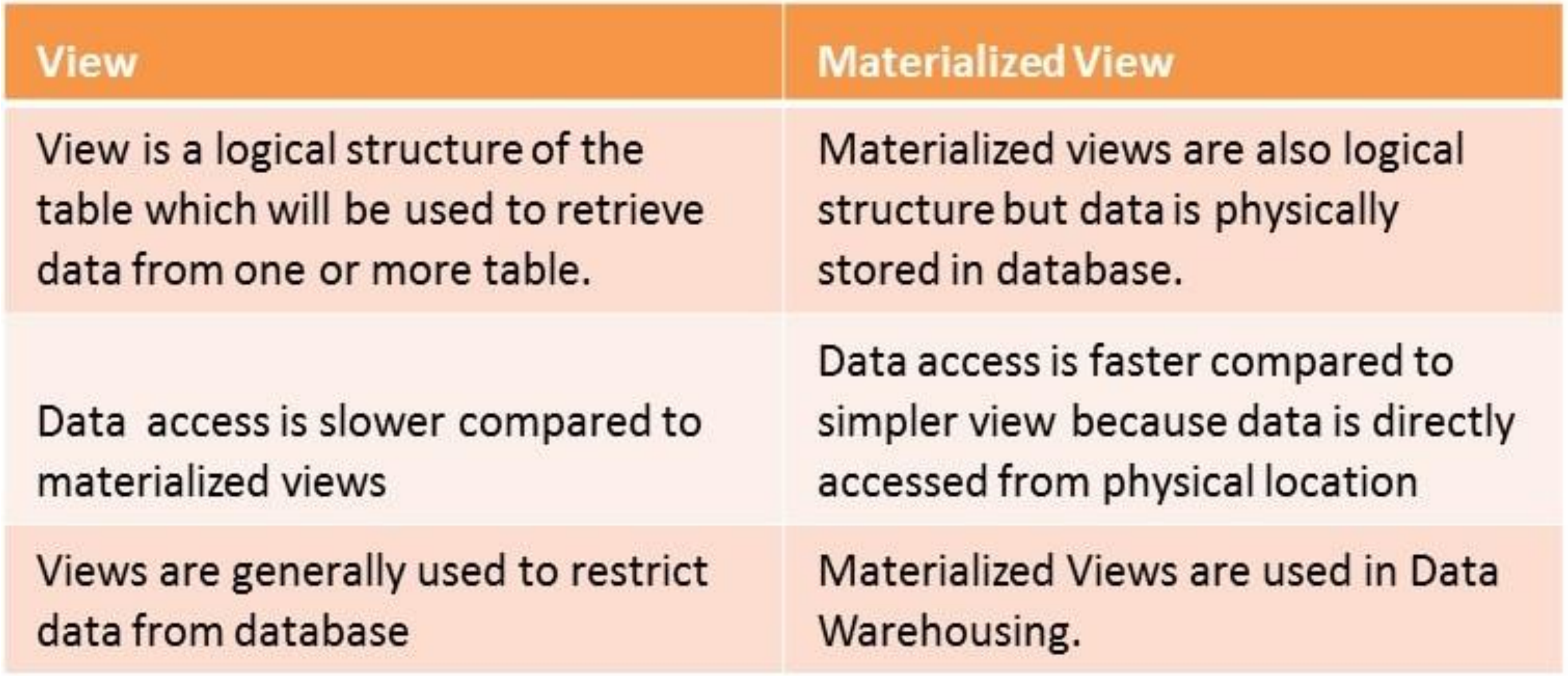
- An inline view is a SELECT statement in the FROM-clause of another SELECT statement to create a temporary table that could be referenced by the SELECT statement.
- Inline views are utilized for writing complex SQL queries without join and subqueries operations.
- This is called a temporary table because a duplicate copy of the data returned by the stored subquery isn't stored in the database.
- In Oracle community, this temporary table is called an inline view.
- It is referred to as a sub-select.

```
SELECT column_names, ...
FROM (subquery)
WHERE ROWNUM<= N;
```

MATERIALIZED VIEWS

- Materialized view replicates the retrieved data physically.
- This replicated data can be reused without executing the view again.
- This type of view is also known as "SNAPSHOTS".
- Materialized view reduce the processing time to regenerate the whole data.
- It helps remote users to replicate data locally and improve query performance.
- The challenging part here is to synchronize the changes in materialized views underlying tables

VIEWS VERSUS MATERIALIZED VIEWS



View	Materialized View
View is a logical structure of the table which will be used to retrieve data from one or more table.	Materialized views are also logical structure but data is physically stored in database.
Data access is slower compared to materialized views	Data access is faster compared to simpler view because data is directly accessed from physical location
Views are generally used to restrict data from database	Materialized Views are used in Data Warehousing.

CREATING VIEWS

SYNTAX:

CREATE [OR REPLACE] VIEW view_name [(column_aliases)] AS defining-query

[WITH READ ONLY]

[WITH CHECK OPTION]

OR REPLACE

- The OR REPLACE option replaces the definition of existing view.
- It is handy if you have granted various privileges on the view. Because when you use the DROP VIEW and CREATE VIEW to change the view's definition, Oracle removes the view privileges, which may not be what you want. To avoid this, you can use the CREATE OR REPLACE clause that preserves the view privileges.

FORCE

- Usually, you create a new view based on existing tables.
- However, sometimes, you may want to create a view based on the tables that you will create later or the tables that you don't have sufficient privileges to access at the time of creating the view. In these cases, you can use the FORCE option.

column-aliases

- Typically, the column names of a view are derived from the select list of the defining query.
- However, the column names of the defining query may contain functions or expressions that you cannot use for the view definition.
- To solve this problem, you have two options:

1. Use column aliases that adhere to the naming rules in the SELECT clause of the defining query.
2. Explicitly specify column aliases for the view's columns between the CREATE VIEW and AS clauses.

AS defining-query

- The defining query is a SELECT statement that defines the columns and rows of the view.

WITH READ ONLY

- The WITH READ ONLY clause prevents the underlying tables from changes through the view.

WITH CHECK OPTION

- The WITH CHECK OPTION clause protects the view from any changes to the underlying table that would produce rows which are not included in the defining query.

PREREQUISITES FOR VIEW CREATION

To create a view, you must meet the following requirements:

- To create a view in your schema, you must have the CREATE VIEW privilege. To create a view in another user's schema, you must have the CREATE ANY VIEW system privilege. You can acquire these privileges explicitly or through a role.

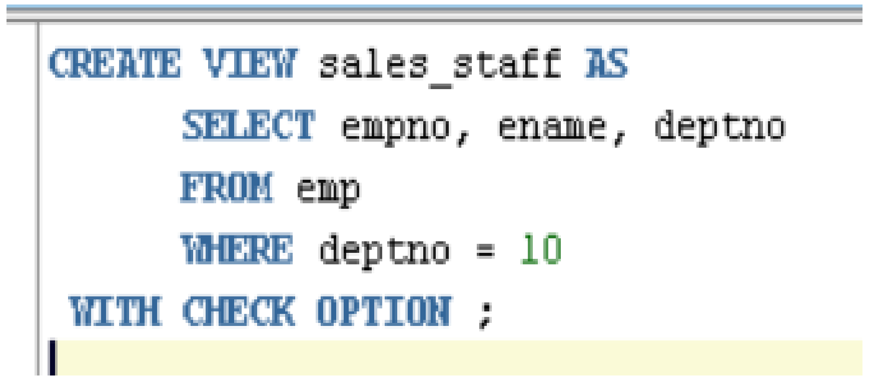
- The owner of the view (whether it is you or another user) must have been explicitly granted privileges to access all objects referenced in the view definition. The owner cannot have obtained these privileges through roles. Also, the functionality of the view depends on the privileges of the view owner. For example, if the owner of the view has only the INSERT privilege for Scott's emp table, then the view can be used only to insert new rows into the emp table, not to SELECT, UPDATE, or DELETE rows.

- If the owner of the view intends to grant access to the view to other users, the owner must have received the object privileges to the base objects with the GRANT OPTION or the system privileges with the ADMIN OPTION.

- You can create views using the CREATE VIEW statement. Each view is defined by a query that references tables, materialized views, or other views.

EXAMPLE:

```
CREATE VIEW sales_staff AS
     SELECT empno, ename, deptno
     FROM emp
     WHERE deptno = 10
 WITH CHECK OPTION CONSTRAINT sales_staff_cnst ;
```



```
CREATE VIEW sales_staff AS
        SELECT empno, ename, deptno
        FROM emp
        WHERE deptno = 10
 WITH CHECK OPTION ;
```

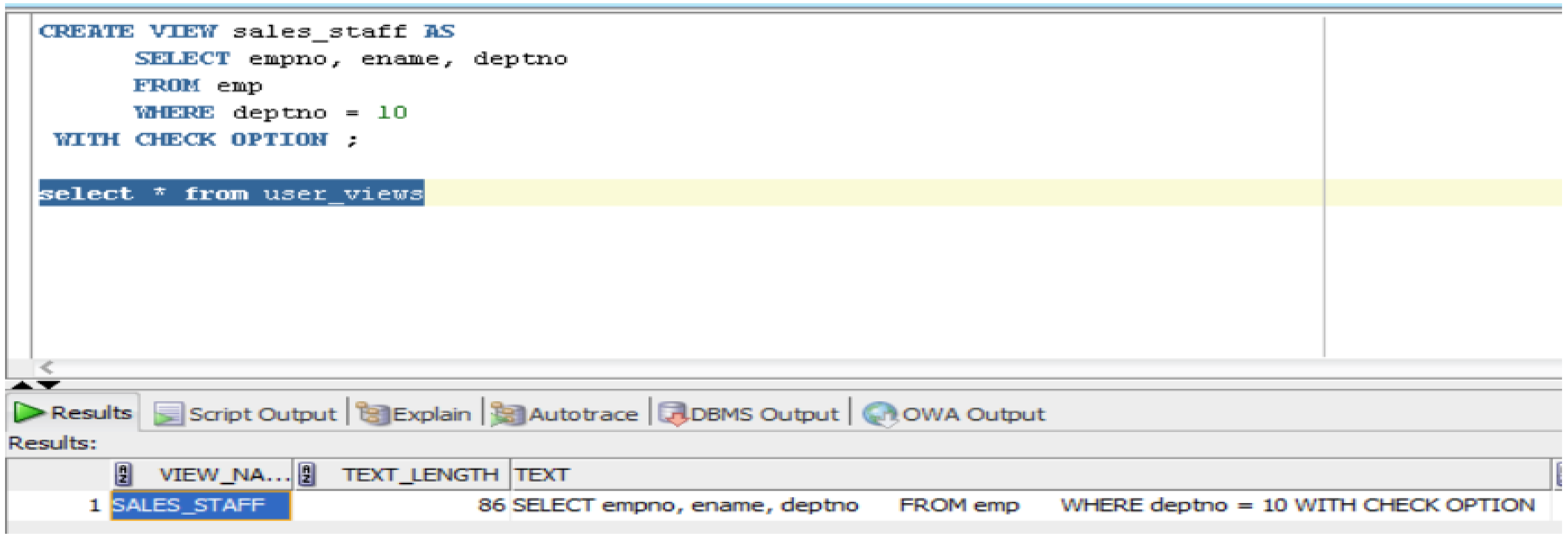
- The query creates a view that references only rows in department 10.
- Furthermore, the CHECK OPTION creates the view with the constraint (named sales_staff_cnst).
- The constraint ensures that any INSERT & UPDATE statement issued against the view cannot result in rows that the view cannot select. For example, the following INSERT statement successfully inserts a row into the emp table by means of the sales_staff view, which contains all rows with department number 10:

INSERT INTO sales_staff VALUES (7584, ‘WILLIAM', 10);

- However, the following INSERT statement returns an error because it attempts to insert a row for department number 30, which cannot be selected using the sales_staff view:

INSERT INTO sales_staff VALUES (7591, 'WILLIAM', 30);

- The view could have been constructed specifying the WITH READ ONLY clause, which prevents any updates, inserts, or deletes from being done to the base table through the view. If no WITH READ ONLY clause is specified, the view, with some restrictions, is inherently updatable.

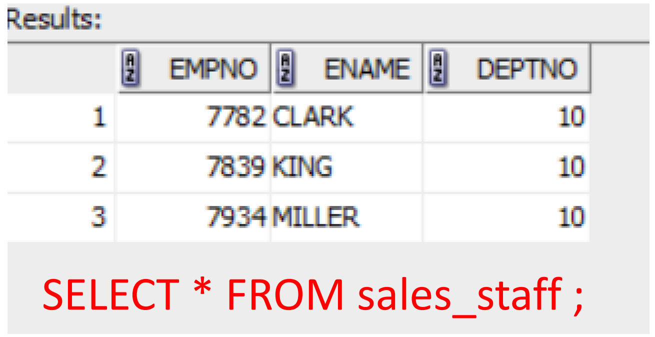


```
CREATE VIEW sales_staff AS
      SELECT empno, ename, deptno
      FROM emp
      WHERE deptno = 10
 WITH CHECK OPTION ;

select * from user_views
```

Results:

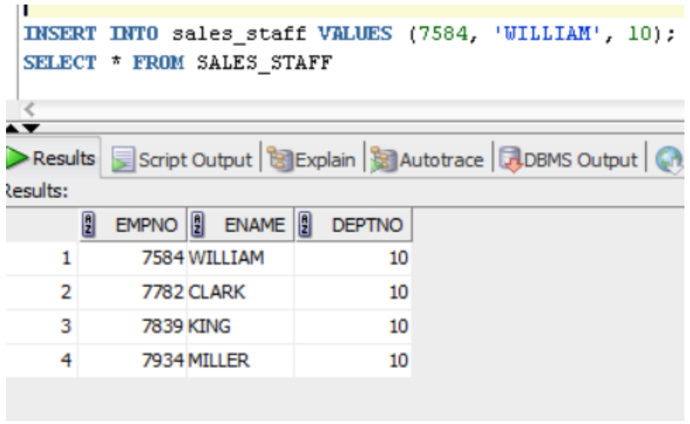
	VIEW_NA...	TEXT_LENGTH	TEXT
1	SALES_STAFF	86	SELECT empno, ename, deptno FROM emp WHERE deptno = 10 WITH CHECK OPTION



Results:

	EMPNO	ENAME	DEPTNO
1	7782	CLARK	10
2	7839	KING	10
3	7934	MILLER	10

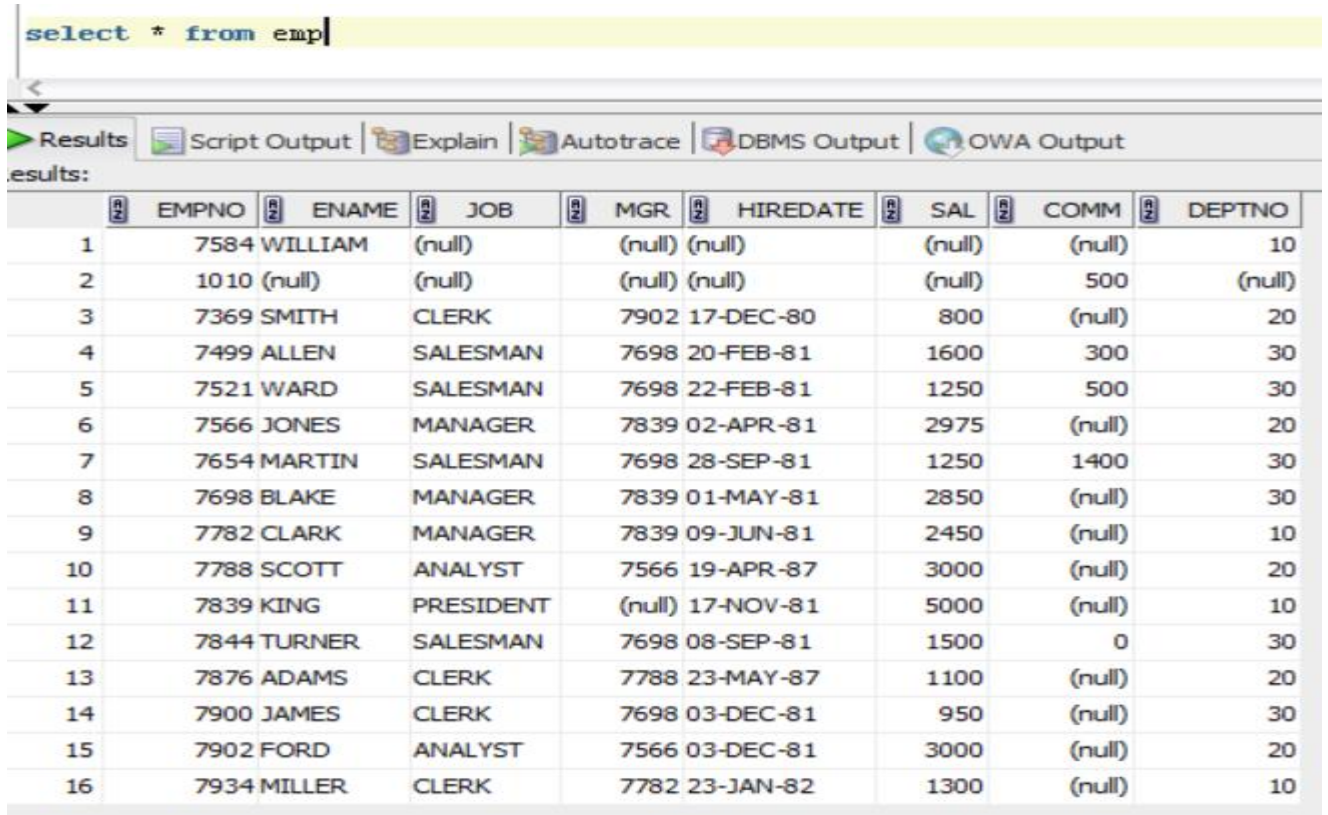
SELECT * FROM sales_staff ;



```
INSERT INTO sales_staff VALUES (7584, 'WILLIAM', 10);
SELECT * FROM SALES_STAFF
```

Results:

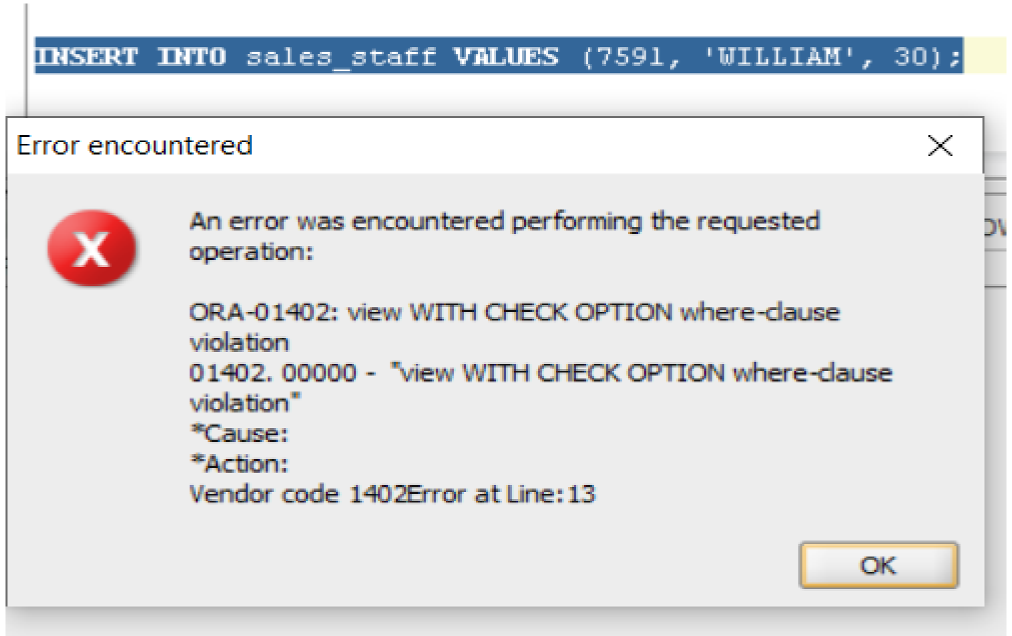
	EMPNO	ENAME	DEPTNO
1	7584	WILLIAM	10
2	7782	CLARK	10
3	7839	KING	10
4	7934	MILLER	10



```
select * from emp
```

Results:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7584	WILLIAM	(null)	(null)	(null)	(null)	(null)	10
2	1010	(null)	(null)	(null)	(null)	(null)	500	(null)
3	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20
4	7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
5	7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
6	7566	JONES	MANAGER	7839	02-APR-81	2975	(null)	20
7	7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
8	7698	BLAKE	MANAGER	7839	01-MAY-81	2850	(null)	30
9	7782	CLARK	MANAGER	7839	09-JUN-81	2450	(null)	10
10	7788	SCOTT	ANALYST	7566	19-APR-87	3000	(null)	20
11	7839	KING	PRESIDENT	(null)	17-NOV-81	5000	(null)	10
12	7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
13	7876	ADAMS	CLERK	7788	23-MAY-87	1100	(null)	20
14	7900	JAMES	CLERK	7698	03-DEC-81	950	(null)	30
15	7902	FORD	ANALYST	7566	03-DEC-81	3000	(null)	20
16	7934	MILLER	CLERK	7782	23-JAN-82	1300	(null)	10

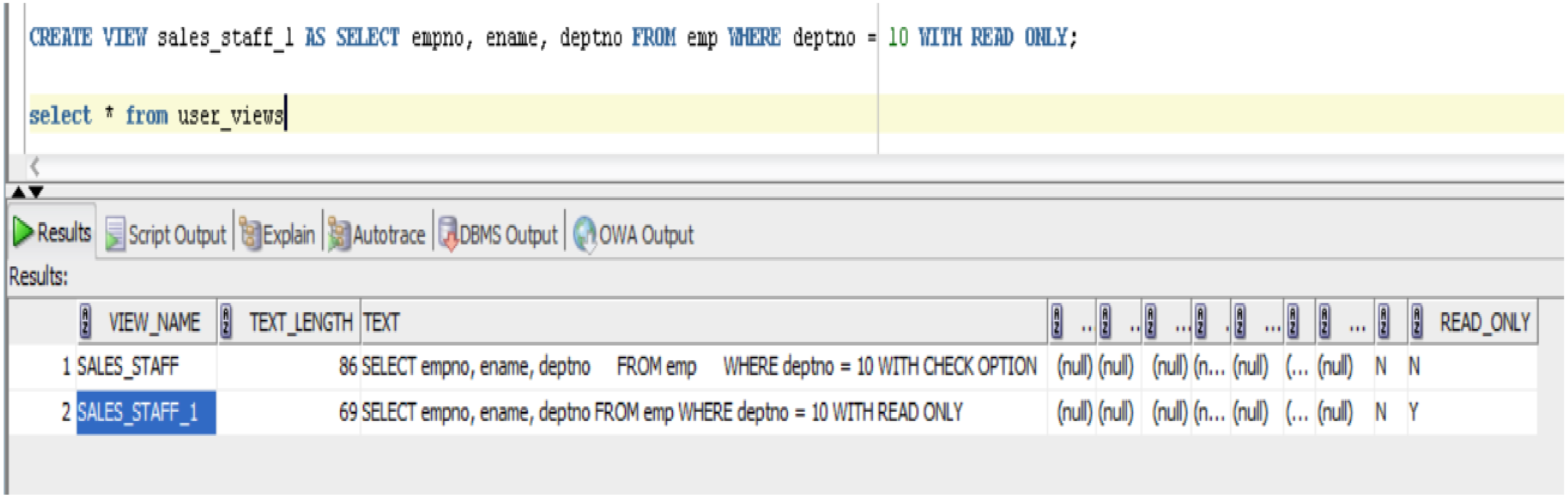


```
INSERT INTO sales_staff VALUES (7591, 'WILLIAM', 30);
```

Error encountered

An error was encountered performing the requested operation:

ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 - "view WITH CHECK OPTION where-clause violation"
*Cause:
*Action:
Vendor code 1402Error at Line:13


```
CREATE VIEW sales_staff_1 AS SELECT empno, ename, deptno FROM emp WHERE deptno = 10 WITH READ ONLY;

select * from user_views
```

Results:

	VIEW_NAME	TEXT_LENGTH	TEXT	..	..	...	.	...		...		READ_ONLY
1	SALES_STAFF	86	SELECT empno, ename, deptno FROM emp WHERE deptno = 10 WITH CHECK OPTION	(null)	(null)	(null)	(n...	(null)	(...	(null)	N	N
2	SALES_STAFF_1	69	SELECT empno, ename, deptno FROM emp WHERE deptno = 10 WITH READ ONLY	(null)	(null)	(null)	(n...	(null)	(...	(null)	N	Y

JOIN VIEWS

- You can also create views that specify more than one base table or view in the FROM clause.
- These are called join views

EXAMPLE:

```
CREATE VIEW division1_staff AS
    SELECT ename, empno, job, dname
    FROM emp, dept
    WHERE emp.deptno IN (10, 30) AND emp.deptno = dept.deptno ;
```

CREATING VIEWS WITH ERRORS

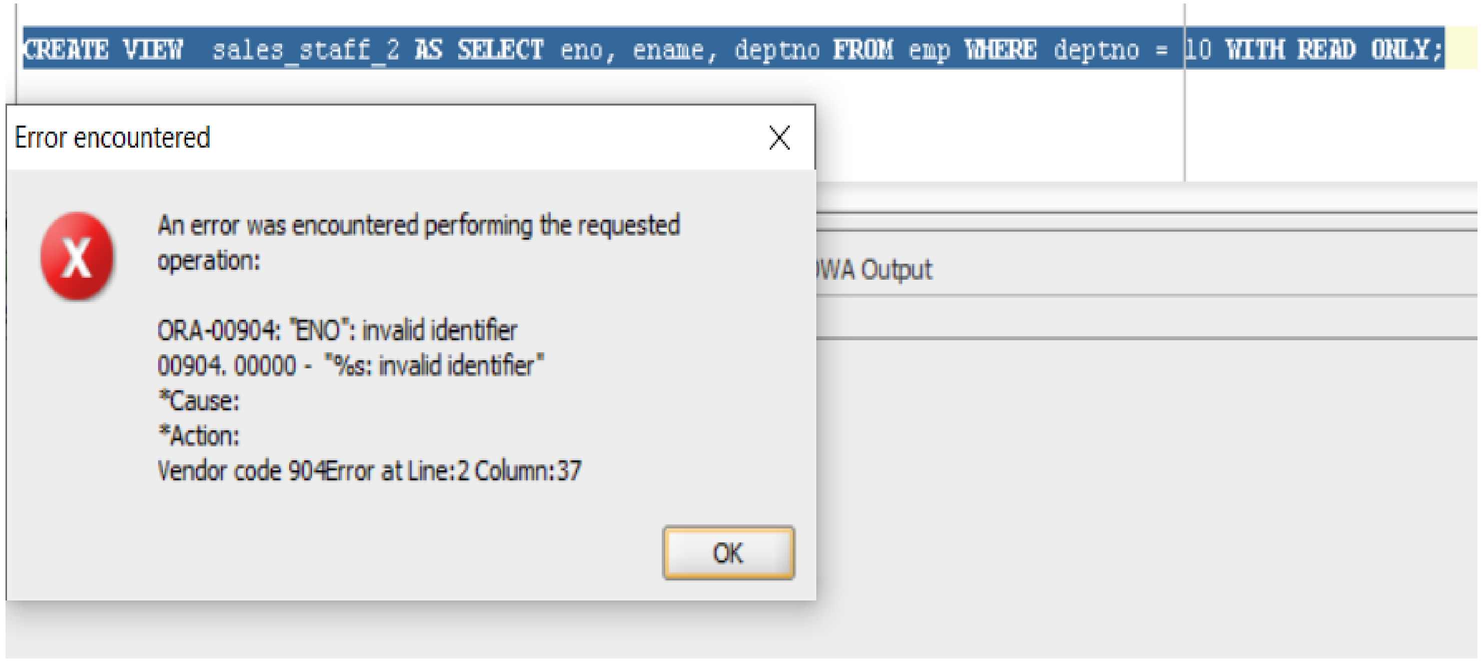
- If there are no syntax errors in a CREATE VIEW statement, the database can create the view even if the defining query of the view cannot be executed.

- In this case, the view is considered "created with errors." For example, when a view is created that refers to a nonexistent table or an invalid column of an existing table, or when the view owner does not have the required privileges, the view can be created anyway and entered into the data dictionary.
- However, such a view is not yet usable.

- To create a view with errors, you must include the FORCE clause of the CREATE VIEW statement.

CREATE FORCE VIEW AS ...;

- By default, views with errors are created as INVALID. When you try to create such a view, the database returns a message indicating the view was created with errors. If conditions later change so that the query of an invalid view can be executed, the view can be recompiled and be made valid (usable).



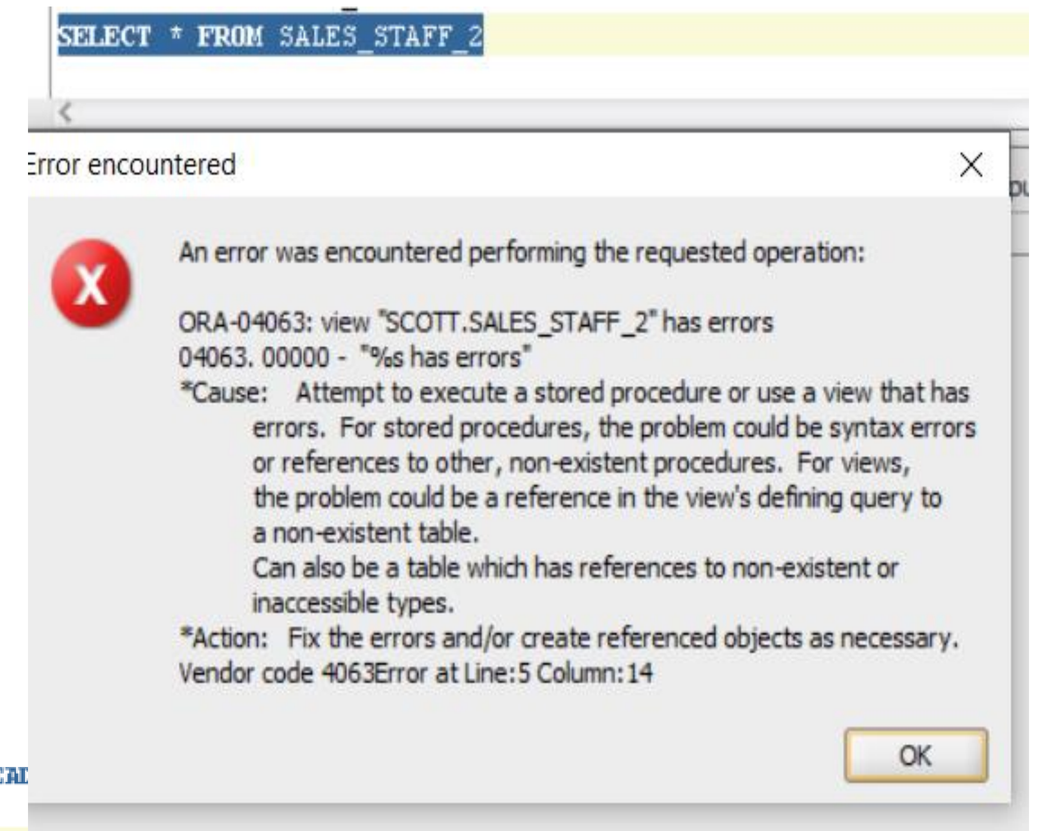
```
CREATE VIEW  sales_staff_2 AS SELECT eno, ename, deptno FROM emp WHERE deptno = 10 WITH READ ONLY;
```

Error encountered

An error was encountered performing the requested operation:

ORA-00904: "ENO": invalid identifier
00904. 00000 - "%s: invalid identifier"
*Cause:
*Action:
Vendor code 904Error at Line:2 Column:37

OK



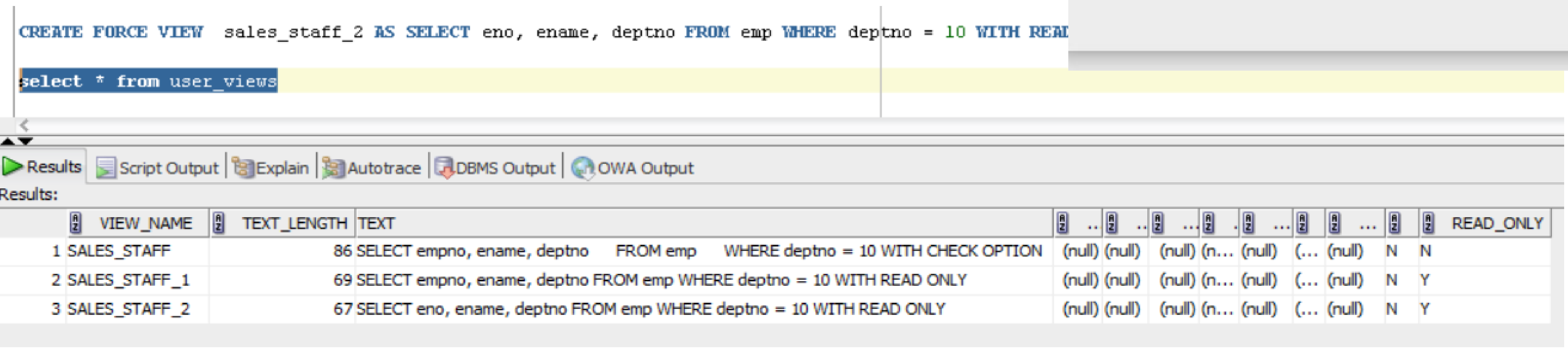
```
SELECT * FROM SALES_STAFF_2
```

Error encountered

An error was encountered performing the requested operation:

ORA-04063: view "SCOTT.SALES_STAFF_2" has errors
04063. 00000 - "%s has errors"
*Cause: Attempt to execute a stored procedure or use a view that has
           errors. For stored procedures, the problem could be syntax errors
           or references to other, non-existent procedures. For views,
           the problem could be a reference in the view's defining query to
           a non-existent table.
           Can also be a table which has references to non-existent or
           inaccessible types.
*Action: Fix the errors and/or create referenced objects as necessary.
Vendor code 4063Error at Line:5 Column:14

OK

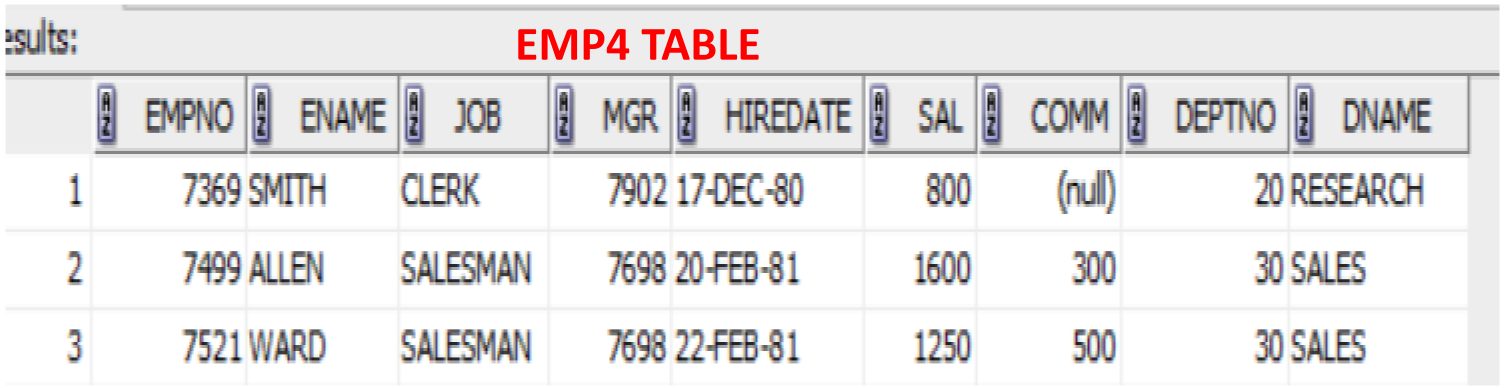


```
CREATE FORCE VIEW  sales_staff_2 AS SELECT eno, ename, deptno FROM emp WHERE deptno = 10 WITH READ

select * from user_views
```

Results:

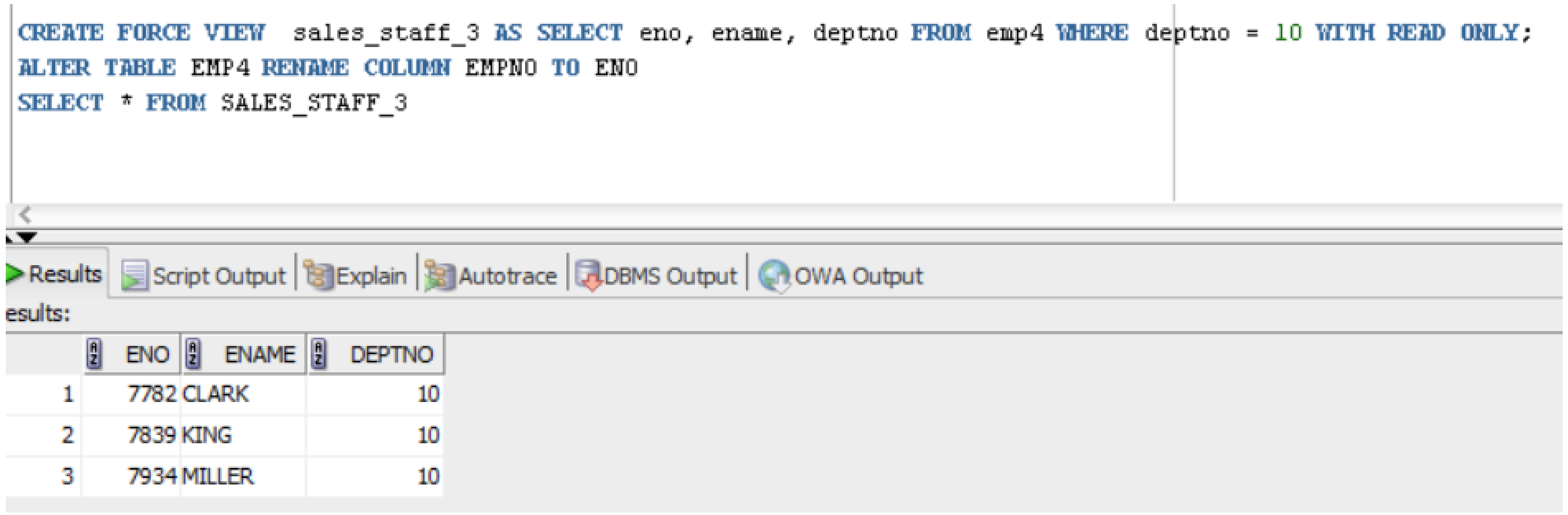
	VIEW_NAME	TEXT_LENGTH	TEXT	..	..	...	.	...		...		READ_ONLY
1	SALES_STAFF	86	SELECT empno, ename, deptno FROM emp WHERE deptno = 10 WITH CHECK OPTION	(null)	(null)	(null)	(n...	(null)	(...	(null)	N	N
2	SALES_STAFF_1	69	SELECT empno, ename, deptno FROM emp WHERE deptno = 10 WITH READ ONLY	(null)	(null)	(null)	(n...	(null)	(...	(null)	N	Y
3	SALES_STAFF_2	67	SELECT eno, ename, deptno FROM emp WHERE deptno = 10 WITH READ ONLY	(null)	(null)	(null)	(n...	(null)	(...	(null)	N	Y



esults:

EMP4 TABLE

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DNAME
1	7369	SMITH	CLERK	7902	17-DEC-80	800	(null)	20	RESEARCH
2	7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30	SALES
3	7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30	SALES



```
CREATE FORCE VIEW  sales_staff_3 AS SELECT eno, ename, deptno FROM emp4 WHERE deptno = 10 WITH READ ONLY;
ALTER TABLE EMP4 RENAME COLUMN EMPNO TO ENO
SELECT * FROM SALES_STAFF_3
```

esults:

	ENO	ENAME	DEPTNO
1	7782	CLARK	10
2	7839	KING	10
3	7934	MILLER	10

REPLACING VIEWS

- To replace a view, you must have all of the privileges required to drop and create a view.
- If the definition of a view must change, the view must be replaced; you cannot use an ALTER VIEW statement to change the definition of a view. You can replace views in the following ways:

1. You can drop and re-create the view.
   - When a view is dropped, all grants of corresponding object privileges are revoked from roles and users. After the view is re-created, privileges must be regranted.
2. You can redefine the view with a **CREATE VIEW** statement that contains the **OR REPLACE** clause. The OR REPLACE clause replaces the current definition of a view and preserves the current security authorizations.

```
CREATE OR REPLACE VIEW sales_staff AS
    SELECT empno, ename, deptno
    FROM emp
    WHERE deptno = 30
 WITH CHECK OPTION CONSTRAINT sales_staff_cnst ;
```

Before replacing a view, consider the following effects:

- Replacing a view replaces the view definition in the data dictionary. All underlying objects referenced by the view are not affected.

- If a constraint in the CHECK OPTION was previously defined but not included in the new view definition, the constraint is dropped.

- All views dependent on a replaced view become invalid (not usable).
- In addition, dependent PL/SQL program units may become invalid, depending on what was changed in the new version of the view.
- For example, if only the WHERE clause of the view changes, dependent PL/SQL program units remain valid. However, if any changes are made to the number of view columns or to the view column names or data types, dependent PL/SQL program units are invalidated.

VIEW RESTRICTIONS

Restrictions on DML operations for views use the following criteria in the order listed:

- If a view is defined by a query that contains **SET** or **DISTINCT** operators, a **GROUP BY** clause, or a group function, then rows cannot be inserted into, updated in, or deleted from the base tables using the view.

- If a view is defined using **WITH CHECK OPTION**, a row cannot be inserted into, or updated in, the base table (using the view), if the view cannot select the row from the base table.

- If a **NOT NULL** column that does not have a **DEFAULT** clause is omitted from the view, then a row cannot be inserted into the base table using the view.

ALTERING VIEWS

- You use the **ALTER VIEW** statement only to explicitly recompile a view that is invalid.

- If you want to change the definition of a view, then you need to create the view with REPLACE option.

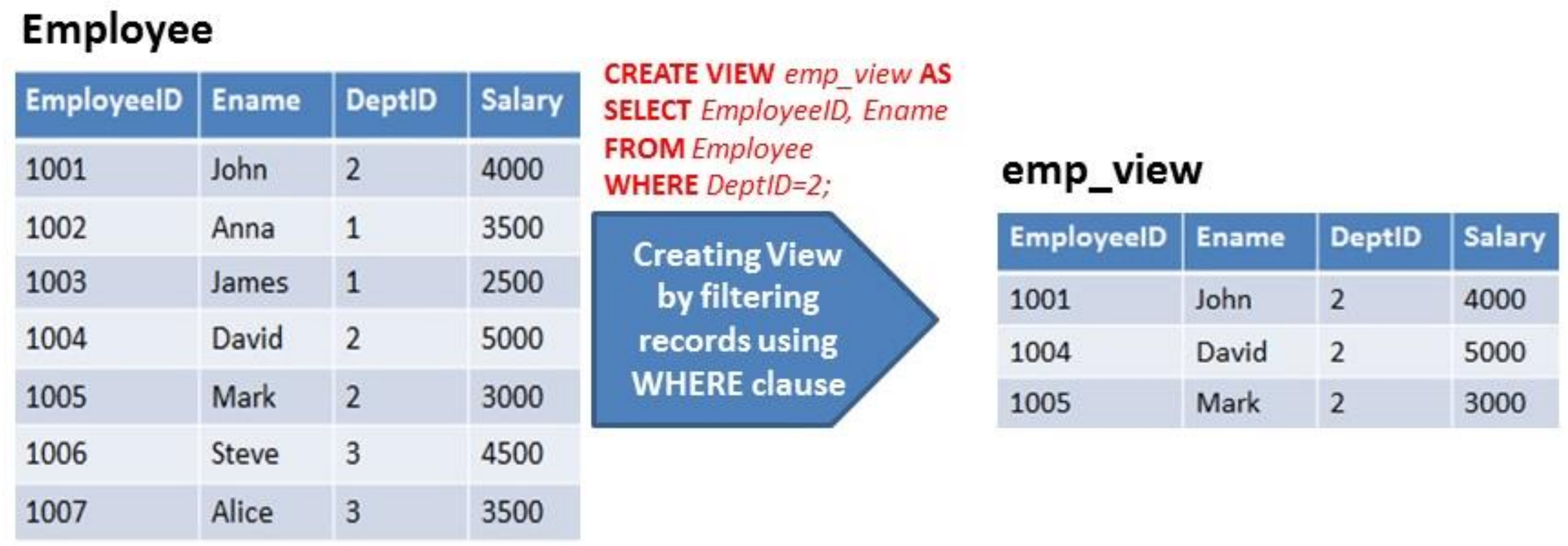
- To use the ALTER VIEW statement, the view must be in your schema, or you must have the ALTER ANY TABLE system privilege.

DROPPING VIEWS

- You can drop any view contained in your schema.
- To drop a view in another user's schema, you must have the DROP ANY VIEW system privilege.
- Drop a view using the DROP VIEW statement.

```
DROP VIEW emp_dept ;
```

SIMPLE VIEW



Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

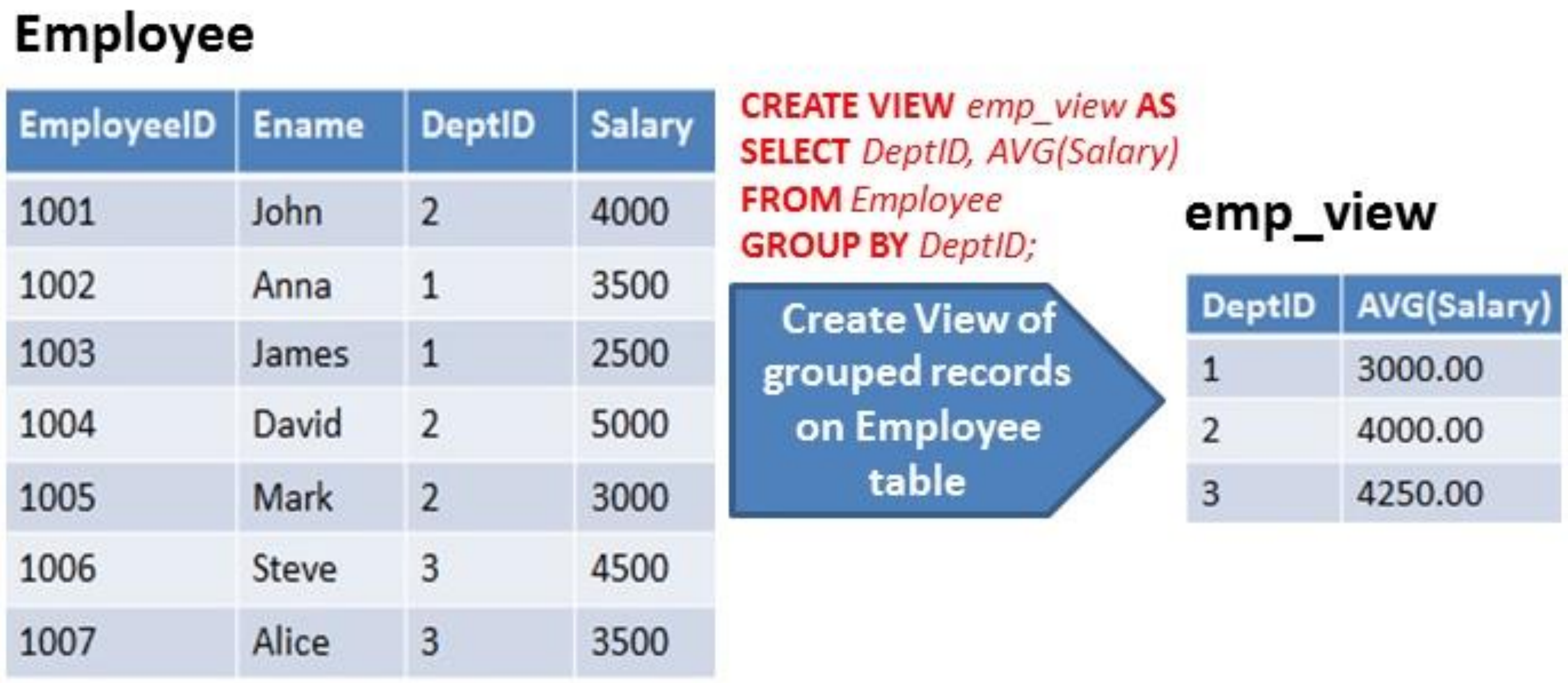
```
CREATE VIEW emp_view AS
SELECT EmployeeID, Ename
FROM Employee
WHERE DeptID=2;
```

Creating View by filtering records using WHERE clause

emp_view

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1004	David	2	5000
1005	Mark	2	3000

COMPLEX VIEWS



Employee

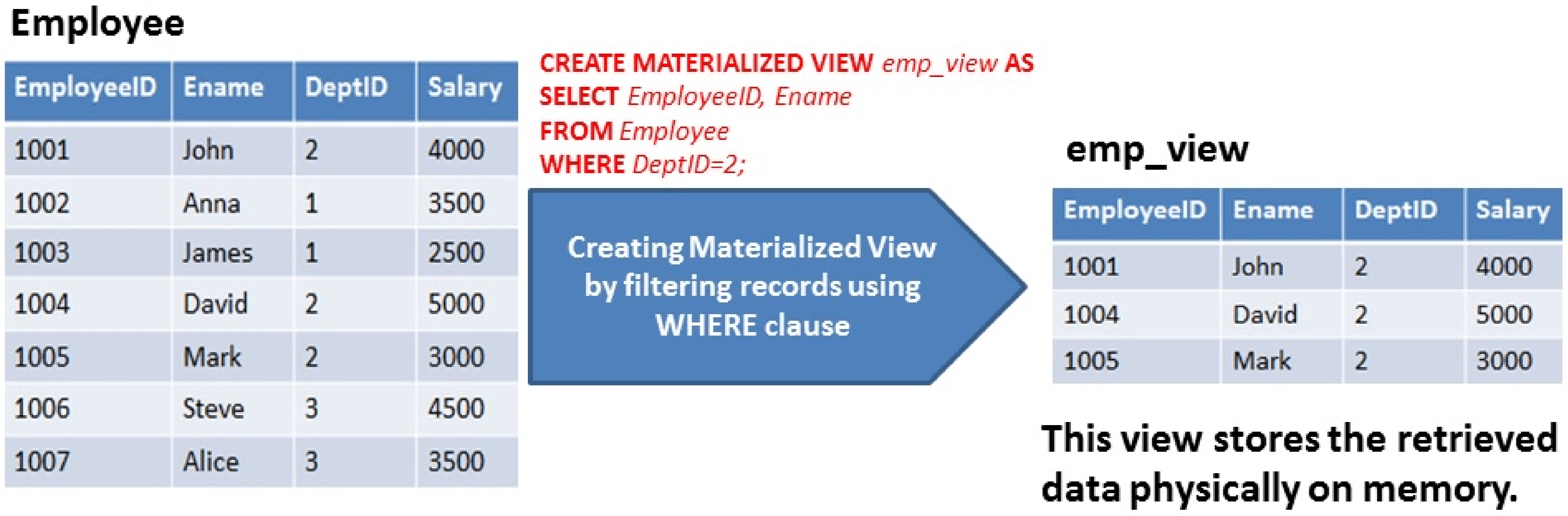
EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
CREATE VIEW emp_view AS
SELECT DeptID, AVG(Salary)
FROM Employee
GROUP BY DeptID;
```

emp_view

DeptID	AVG(Salary)
1	3000.00
2	4000.00
3	4250.00

MATERIALIZED VIEW



Employee

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1002	Anna	1	3500
1003	James	1	2500
1004	David	2	5000
1005	Mark	2	3000
1006	Steve	3	4500
1007	Alice	3	3500

```
CREATE MATERIALIZED VIEW emp_view AS
SELECT EmployeeID, Ename
FROM Employee
WHERE DeptID=2;
```

Creating Materialized View by filtering records using WHERE clause

emp_view

EmployeeID	Ename	DeptID	Salary
1001	John	2	4000
1004	David	2	5000
1005	Mark	2	3000

This view stores the retrieved data physically on memory.