Views

Objectives

- Describe a view
- Create a view
- Retrieve data through a view
- Alter the definition of a view
- Insert, update, and delete data through a view
- Drop a view

VIEW

- A view, like a table, is a database object.
- However, views are not “real" tables.
  - They are logical representations of existing tables or of another view.
  - Views contain no data of their own.
  - They function as a window through which data from tables can be viewed or changed.
  - The view is a query stored as a SELECT statement in the data dictionary.
  - The tables on which a view is based are called "base tables".

Why Use Views?

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

Creating a View

- You embed a subquery within the CREATE VIEW statement.

- The subquery can contain complex SELECT syntax.
- The subquery cannot contain an ORDER BY clause.

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
  [(alias[, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY]
```

Guidelines For Creating A View

- The subquery that defines the view can contain complex SELECT syntax.
- The subquery that defines the view cannot contain an ORDER BY clause.
- You can use the OR REPLACE option to change the definition of the view without having to drop it or regrant object privileges previously granted on it.
- Aliases can be used for the column names in the subquery.
  - Aliases must be used for the expression columns

Creating a View

- Create a view, EMPVU10, that contains details of employees in department 10.

```
SQL> CREATE VIEW     empvu10
  2  AS SELECT       empno, ename, job
  3  FROM            employee
  4  WHERE           deptno = 10;
View created.
```

- Describe the structure of the view by using the SQL*Plus DESCRIBE command.

```
SQL> DESCRIBE empvu10
```

Creating a View

- Create a view by using column aliases in the subquery.

```
SQL> CREATE VIEW     salvu30
  2  AS SELECT    empno EMPLOYEE_NUMBER, ename NAME,
  3               sal SALARY
  4  FROM         employee
  5  WHERE        deptno = 30;
View created.
```

- Select the columns from this view by the given alias names.

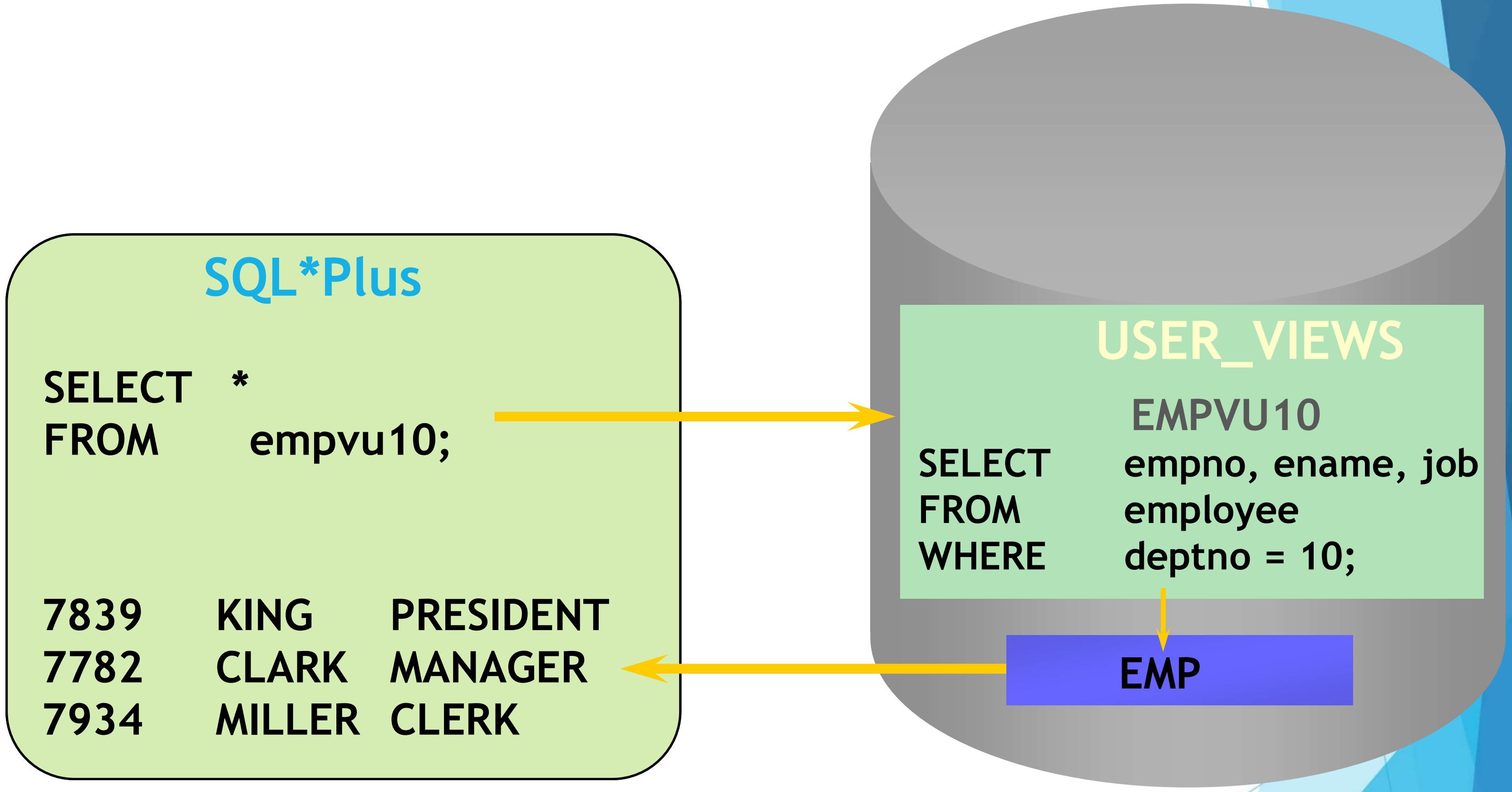
Retrieving Data from a View

```
SQL>  SELECT *
  2   FROMsalvu30;
```

```
EMPLOYEE_NUMBER NAME            SALARY
--------------- ---------- ----------
           7698 BLAKE            2850
           7654 MARTIN           1250
           7499 ALLEN            1600
           7844 TURNER           1500
           7900 JAMES             950
           7521 WARD             1250

6 rows selected.
```

Querying a View



```
SELECT   *
FROM      empvu10;
```

```
7839    KING     PRESIDENT
7782    CLARK    MANAGER
7934    MILLER   CLERK
```

```
SELECT      empno, ename, job
FROM        employee
WHERE       deptno = 10;
```

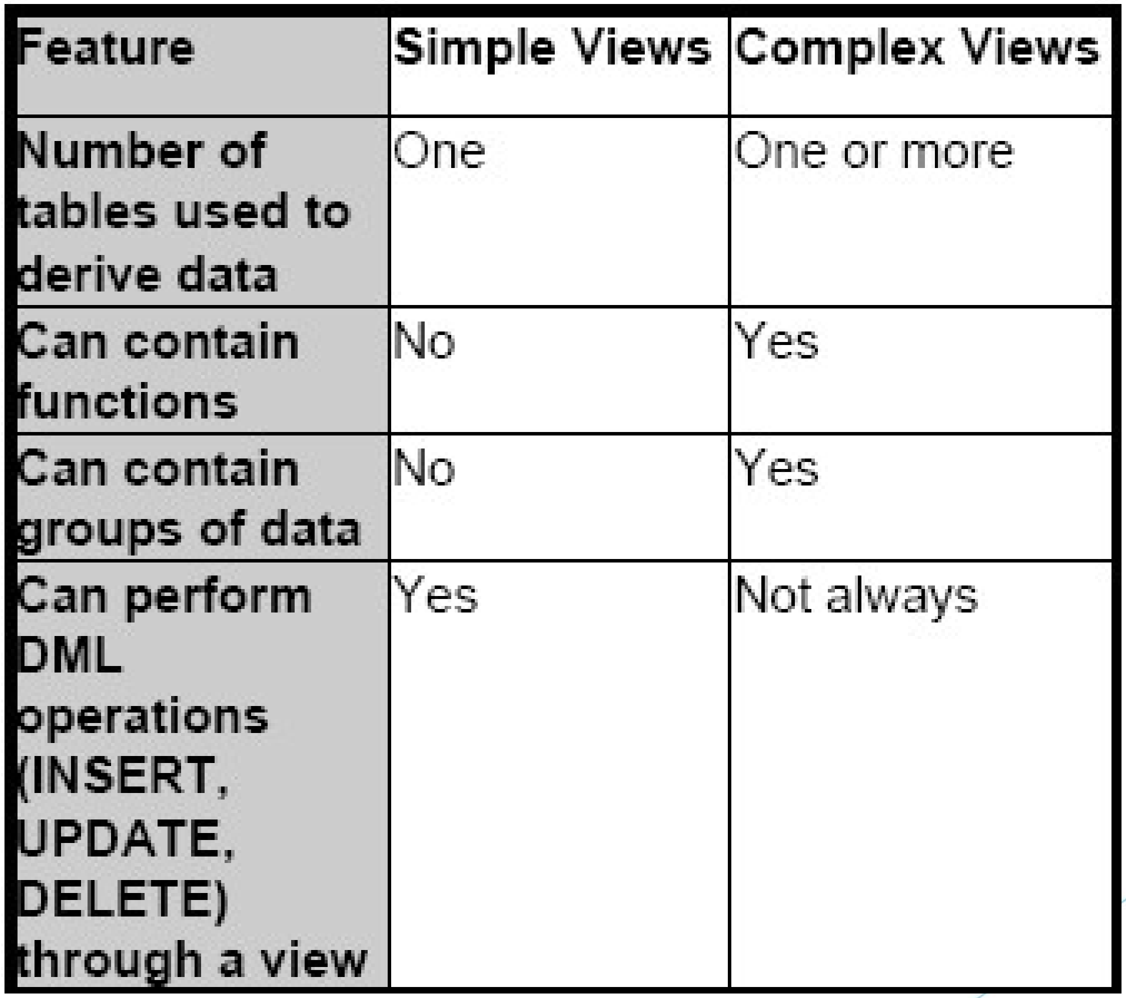
Modifying a View

- Modify the EMPVU10 view by using CREATE OR REPLACE VIEW clause. Add an alias for each column name.

```
SQL> CREATE OR REPLACE VIEW empvu10
  2           (employee_number, employee_name, job_title)
  3   AS SELECT   empno, ename, job
  4   FROM        employee
  5   WHERE       deptno = 10;
View created.
```

- Column aliases in the CREATE VIEW clause are listed in the same order as the columns in the subquery.

Simple And Complex Views



Feature	Simple Views	Complex Views
Number of tables used to derive data	One	One or more
Can contain functions	No	Yes
Can contain groups of data	No	Yes
Can perform DML operations (INSERT, UPDATE, DELETE) through a view	Yes	Not always

SIMPLE VIEW

- The view shown below is an example of a simple view.
- The subquery derives data from only one table and it does not contain a join function or any group functions.
- Because it is a simple view, INSERT, UPDATE, DELETE, and MERGE operations affecting the base tables could be performed through the view.

```
CREATE VIEW view_copy_d_cds
AS SELECT cd_number, title, producer, year
FROM d_cds;
```

Creating a Complex View

- Create a complex view that contains group functions to display values from two tables.

```
SQL> CREATE VIEW dept_sum_vu
  2                 (name, minsal, maxsal, avgsal)
  3  AS SELECT      d.dname, MIN(e.sal), MAX(e.sal),
  4                 AVG(e.sal)
  5  FROM           employee e, department d
  6  WHERE          e.deptno = d.deptno
  7  GROUP BY       d.dname;
View created.
```

DML Operations on a View

Rules for Performing DML Operations on a View

- You can perform DML operations on simple views.
- You cannot remove a row if the view contains the following:
  - Group functions
  - A GROUP BY clause
  - The DISTINCT keyword

DML Operations on a View

Rules for Performing DML Operations on a View

- You cannot modify data in a view if it contains:
  - Any of the conditions previously mentioned
  - Columns defined by expressions
  - The ROWNUM pseudo column
- You cannot add data if:
  - The view contains any of the conditions mentioned above or previously mentioned
  - There are NOT NULL columns in the base tables that are not selected by the view

WITH CHECK OPTION

▶You can ensure that DML on the view stays within the domain of the view by using the WITH CHECK OPTION.

```
SQL> CREATE OR REPLACE VIEW empvu20
  2  AS SELECT   *
  3  FROM             employee
  4  WHERE            deptno = 20
  5  WITH CHECK OPTION CONSTRAINT empvu20_ck;
View created.
```

- Any attempt to change the department number for any row in the view will fail because it violates the WITH CHECK OPTION constraint.

Denying DML Operations

▶You can ensure that no DML operations occur by adding the WITH READ ONLY option to your view definition.

```
SQL> CREATE OR REPLACE VIEW empvu10
  2        (employee_number, employee_name, job_title)
  3  AS SELECT   empno, ename, job
  4  FROM        employee
  5  WHERE       deptno = 10
  6  WITH READ ONLY;
View created.
```

- Any attempt to perform a DML on any row in the view will result in Oracle Server error ORA-01752.

Removing a View

- Remove a view without losing data because a view is based on underlying tables in the database.

```
DROP VIEW view;
```

```
SQL> DROP VIEW empvu10;
View dropped.
```

Summary

- A view is derived from data in other tables or other views.
- A view provides the following advantages:
  - Restricts database access
  - Simplifies queries
  - Provides data independence
  - Allows multiple views of the same data
  - Can be dropped without removing the underlying data