

6

Subqueries

Objectives

After completing this lesson, you should be able to do the following:

- **Describe the types of problems that subqueries can solve**
- **Define subqueries**
- **List the types of subqueries**
- **Write single-row and multiple-row subqueries**

Using a Subquery to Solve a Problem

“Who has a salary greater than Jones’?”

Main Query



“Which employees have a salary greater than Jones’ salary?”

Subquery



“What is Jones’ salary?”


Subqueries

```
SELECT select_list
FROM   table
WHERE  expr operator
       (SELECT select_list
        FROM   table);
```

- The subquery (inner query) executes once before the main query.
- The result of the subquery is used by the main query (outer query).

Using a Subquery

```
SQL> SELECT  ename
      2  FROM    emp      2975
      3  WHERE   sal >
      4          (SELECT sal
      5              FROM    emp
      6              WHERE   empno=7566) ;
```



ENAME

KING

FORD

SCOTT

Guidelines for Using Subqueries

- **Enclose subqueries in parentheses.**
- **Place subqueries on the right side of the comparison operator.**
- **Do not add an ORDER BY clause to a subquery.**
- **Use single-row operators with single-row subqueries.**
- **Use multiple-row operators with multiple-row subqueries.**

Types of Subqueries

- **Single-row subquery**



- **Multiple-row subquery**



- **Multiple-column subquery**



Single-Row Subqueries

- **Return only one row**
- **Use single-row comparison operators**

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

Executing Single-Row Subqueries

```
SQL> SELECT      ename, job
  2  FROM          emp
  3  WHERE         job =
  4      (SELECT    job
  5          FROM    emp
  6          WHERE   empno = 7369)
  7  AND          sal >
  8      (SELECT    sal
  9          FROM    emp
 10          WHERE   empno = 7876) ;
```

CLERK

1100

ENAME	JOB
-----	-----
MILLER	CLERK

Using Group Functions in a Subquery

```
SQL> SELECT  ename, job, sal
2  FROM emp
3  WHERE      sal =
4             (SELECT MIN(sal)
5             FROM
```

```
emp) ;
```

800



ENAME	JOB	SAL
-----	-----	-----
SMITH	CLERK	800

HAVING Clause with Subqueries

- The Oracle Server executes subqueries first.
- The Oracle Server returns results into the HAVING clause of the main query.

```
SQL> SELECT deptno, MIN(sal)
2 FROM emp
3 GROUP BY deptno
4 HAVING MIN(sal) >
5 (SELECT MIN(sal)
6 FROM emp
7 WHERE deptno = 20);
```

800



What Is Wrong with This Statement?

```
SQL> SELECT empno, ename  
2 FROM emp  
3 WHERE sal =  
4 (SELECT MIN(sal)  
5 FROM emp  
6 GROUP BY deptno);
```

Single-row operator with
multiple-row subquery

ERROR:

ORA-01427: single-row subquery returns more than
one row

no rows selected

Will This Statement Work?

```
SQL> SELECT ename, job
2  FROM emp
3  WHERE job =
4      (SELECT job
5  FROM emp
6  WHERE ename='SMYTHE' );
```

no rows selected

Subquery returns no values

Multiple-Row Subqueries

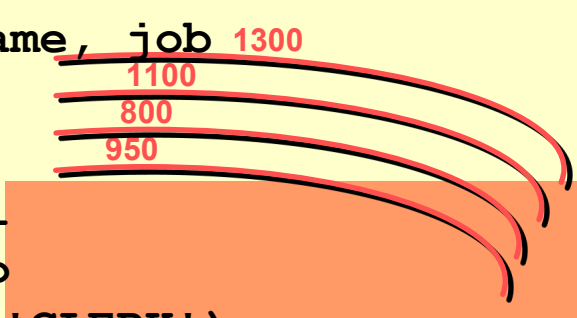
- **Return more than one row**
- **Use multiple-row comparison operators**

Operator	Meaning
IN	Equal to any member in the list
ANY	Compare value to each value returned by the subquery
ALL	Compare value to every value returned by the subquery

Feature	IN	ANY
Used With	= only	=, >, <, >=, <=, <>
Checks	Exact matches	Comparison with each value
True When	The value exists in the list	The condition holds for at least one value
Example	x IN (1, 2, 3)	x > ANY (1, 2, 3) (Equivalent to x > 1)

Using ANY Operator in Multiple-Row Subqueries

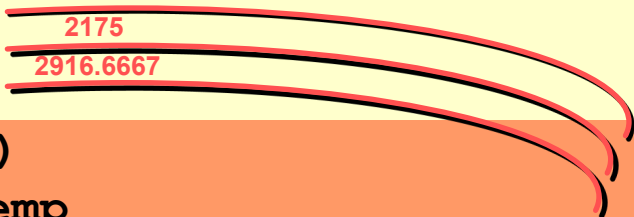
```
SQL> SELECT empno, ename, job 1300
2 FROM emp 1100
3 WHERE sal < ANY 800
4 (SELECT sal 950
5 FROM emp
6 WHERE job = 'CLERK')
7 AND job <> 'CLERK';
```



EMPNO	ENAME	JOB
7654	MARTIN	SALESMAN
7521	WARD	SALESMAN

Using ALL Operator in Multiple-Row Subqueries

```
SQL> SELECT empno, ename, job 1566.6667
2 FROM emp 2175
3 WHERE sal > ALL 2916.6667
4 (SELECT avg(sal)
5 FROM emp
6 GROUP BY deptno);
```



EMPNO	ENAME	JOB
7839	KING	PRESIDENT
7566	JONES	MANAGER
7902	FORD	ANALYST
7788	SCOTT	ANALYST

Summary

Subqueries are useful when a query is based on unknown values.

```
SELECT select_list
FROM   table
WHERE  expr operator
       (SELECT select_list
        FROM   table);
```

Practice Overview

- **Creating subqueries to query values based on unknown criteria**
- **Using subqueries to find out what values exist in one set of data and not in another**

6

Creating Views

Objectives

After completing this lesson, you should be able to do the following:

- **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**

Database Objects

Object	Description
Table	Basic unit of storage; composed of rows and columns
View	Logically represents subsets of data from one or more tables
Sequence	Generates primary key values
Index	Improves the performance of some queries
Synonym	Alternative name for an object

EMP Table

EMPVU10 View

Why Use Views?

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

Simple Views and Complex Views

Feature	Simple Views	Complex Views	
Number of tables	One	One or more	
Contain functions	No	Yes	
Contain groups of data	No	Yes	
DML through view	Yes	Not always	

Creating a View

- You embed a subquery within the **CREATE VIEW** statement.

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

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

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            emp
  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            emp
  5  WHERE            deptno = 30;
View created.
```

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

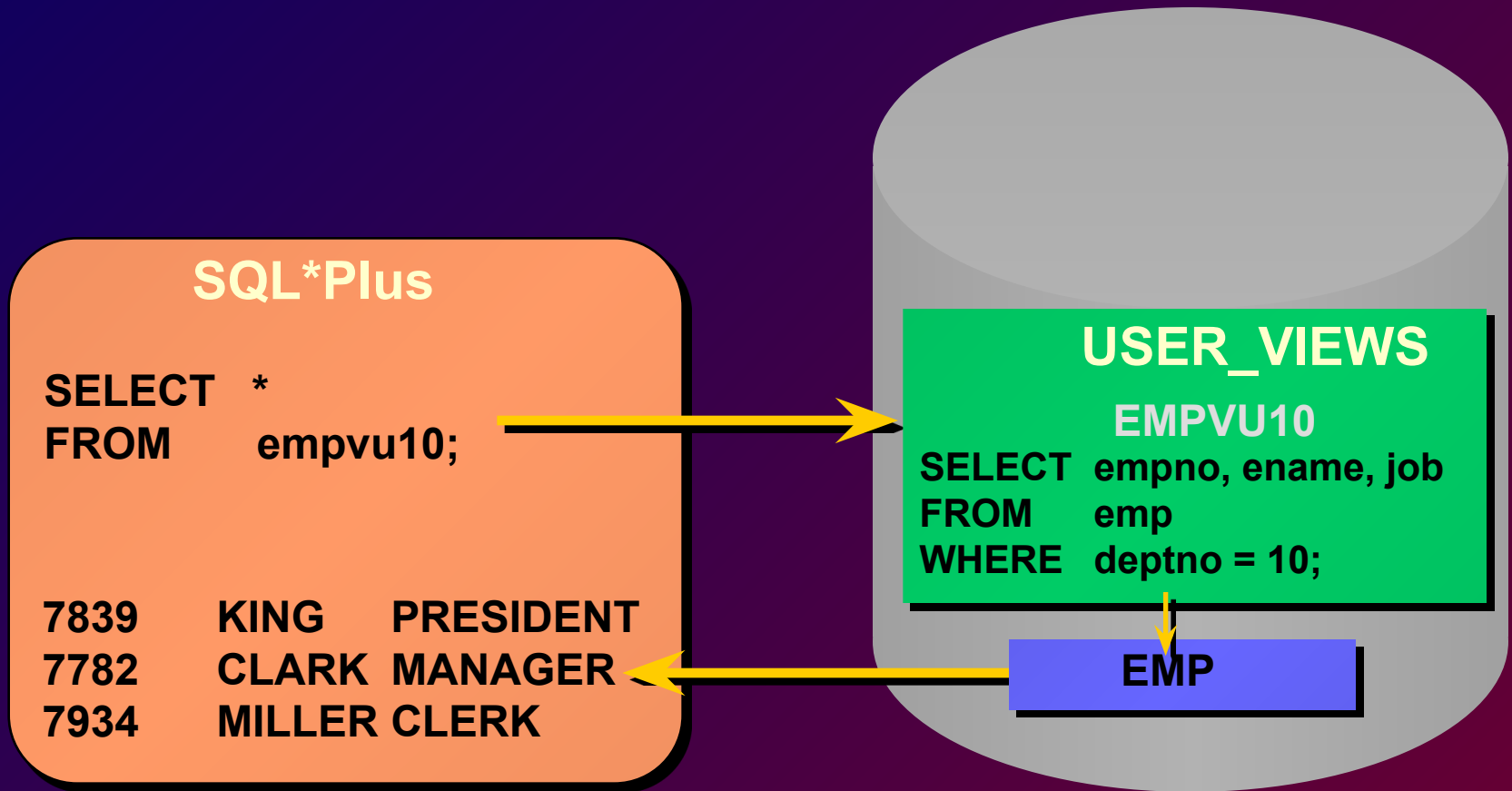
Retrieving Data from a View

```
SQL>  SELECT *  
      2  FROM salvu30;
```

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



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        emp
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.**

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      emp e, dept d
6 WHERE     e.deptno = d.deptno
7 GROUP BY  d.dname;
```

View created.

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**

Rules for Performing DML Operations on a View

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

Using the WITH CHECK OPTION Clause

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

```
SQL> CREATE OR REPLACE VIEW empvu20
  2  AS SELECT      *
  3  FROM            emp
  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            emp
  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.

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**

Practice Overview

- **Creating a simple view**
- **Creating a complex view**
- **Creating a view with a check constraint**
- **Attempting to modify data in the view**
- **Displaying view definitions**
- **Removing views**

