The Sub Query

- Jayendra Khatod

Objectives

- Describe the types of problem 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 Abel's?

Main Query:

Which employees have salaries greater than Abel's salary?

Subquery

What is Abel's salary?

Using a Subquery

```
SELECT last_name
FROM   employees
WHERE  salary >
               (SELECT salary
                FROM   employees
                WHERE  last_name = 'Abel');
```

11000

LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

Guidelines for Using Subqueries

- **Enclose subqueries in parentheses.**
- **Place subqueries on the right side of the comparison condition.**
- **Use single-row operators with single-row subqueries and use multiple-row operators with multiple-row subqueries.**

Types of Subqueries

- **Single-row subquery**

- **Multiple-row 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

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary >
                (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```

ST_CLERK

2600

LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

Using Group Functions in a Subquery

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

2500

LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

The HAVING Clause with Subqueries

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

```
SELECT    department_id, MIN(salary)
FROM      employees
GROUP BY  department_id
HAVING    MIN(salary) >
                        (SELECT MIN(salary)
                         FROM   employees
                         WHERE  department_id = 50)
```

2500

What is Wrong with this Statement?

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                (SELECT   MIN(salary)
                 FROM     employees
                 GROUP BY department_id);
```

```
ERROR at line 4:
ORA-01427: single-row subquery returns more than
one row
```

Single-row operator with multiple-row subquery

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

Using the ANY Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

Using the ALL Operator in Multiple-Row Subqueries

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
                    (SELECT salary
                     FROM   employees
                     WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

Null Values in a Subquery

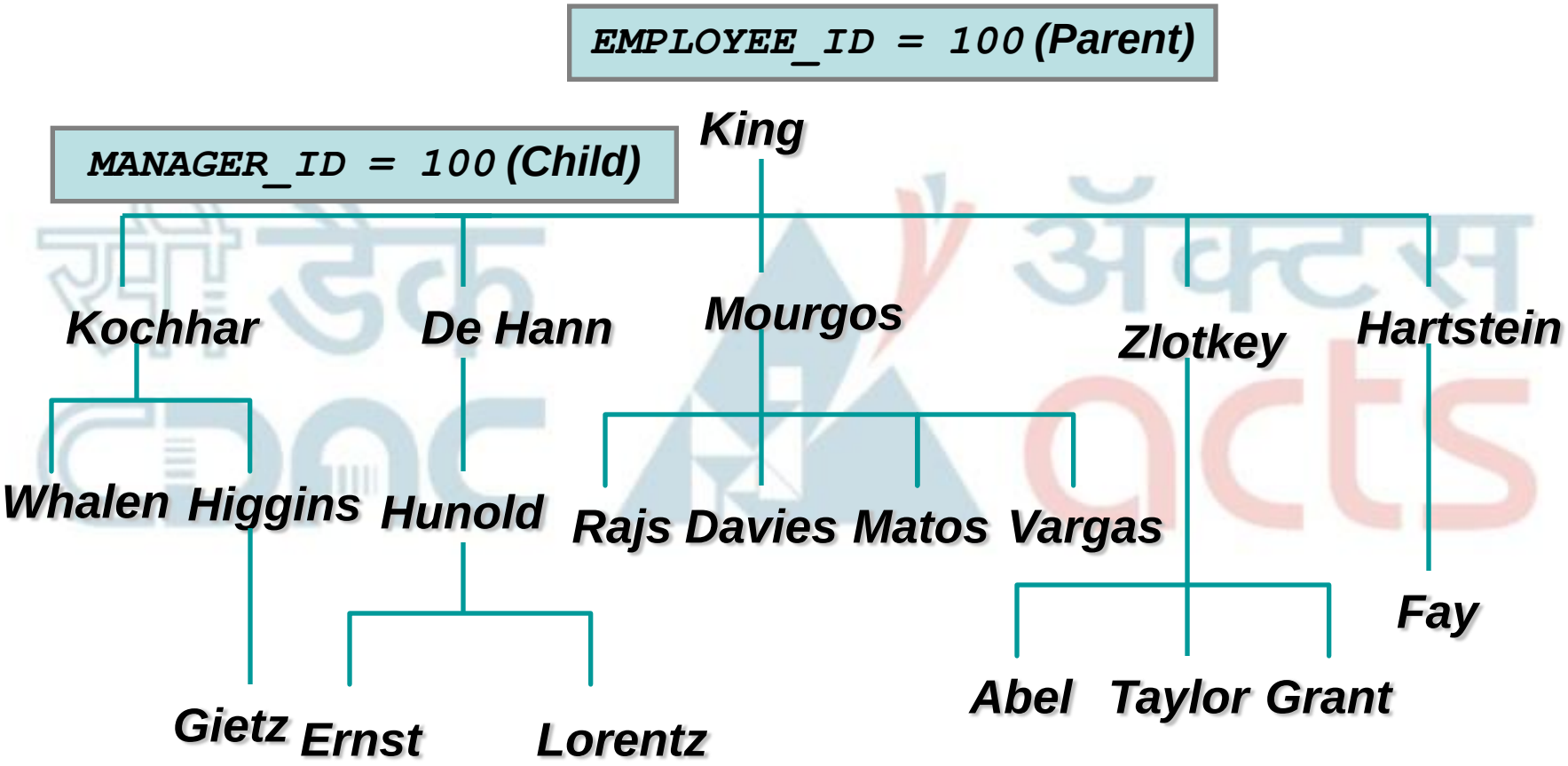
```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
                              (SELECT mgr.manager_id
                               FROM   employees mgr);

no rows selected
```

Hierarchical Reporting

- **Interpret the concept of a hierarchical query**
- **Create a tree-structured report**
- **Format hierarchical data**
- **Exclude branches from the tree structure**

Natural Tree Structure



Hierarchical Queries

```
SELECT [LEVEL], column, expr...
       FROM   table
       [WHERE condition(s)]
       [START WITH condition(s)]
       [CONNECT BY PRIOR condition(s)] ;
```

Starting Point

- **Specifies the condition that must be met**
- **Accepts any valid condition**

```
START WITH column1 = value
```

Using the EMPLOYEES table, start with the employee whose last name is Kochhar.

```
...START WITH last_name = 'Kochhar'
```

Walking the Tree

```
CONNECT BY PRIOR column1 = column2
```

Walk from the top down, using the `EMPLOYEES` table.

```
... CONNECT BY PRIOR employee_id = manager_id
```

Direction

Top down → ***Column1 = Parent Key***
Column2 = Child Key

Bottom up → ***Column1 = Child Key***
Column2 = Parent Key

Walking the Tree: From the Bottom Up

```
SELECT employee_id, last_name, job_id, manager_id
FROM   employees
START  WITH  employee_id = 101
CONNECT BY PRIOR manager_id = employee_id ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	MANAGER_ID
101	Kochhar	AD_VP	100
100	King	AD_PRES	

Walking the Tree: From the Top Down

```
SELECT employee_id, last_name, job_id,
       manager_id, level
FROM   employees
START  WITH  employee_id = 100
CONNECT BY PRIOR employee_id = manager_id
```

20 rows selected.

Summary

In this lesson, you should have learned how to:

- **Identify when a subquery can help solve a question**
- **Write subqueries when a query is based on unknown values**

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

Summary (Continued..)

- **You can Use hierarchical queries to view a hierarchical relationship between rows in a table.**
- **You specify the direction and starting point of the query.**
- **You can eliminate nodes or branches by pruning.**

```
SELECT [LEVEL], column, expr...
        FROM   table
    [WHERE condition(s)]
    [START WITH condition(s)]
[CONNECT BY PRIOR condition(s)] ;
```

Thank You !