Lab Week – 3
Displaying Data from Multiple Tables

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Displaying Data from Multiple Tables

- In this session:
  - Using SELECT statements to access data from more than one table using equality and non-equality joins
  - View data that generally does not meet a join condition by using outer joins
  - Join a table to itself

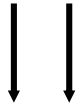
Obtaining Data from Multiple Tables

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700



EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

What Is a Join?

- Use a join to query data from more than one table.

```
SELECT    table1.column, table2.column
FROM      table1, table2
WHERE     table1.column1 = table2.column2;
```

  - Write the join condition in the WHERE clause.
  - Prefix the column name with the table name when the same column name appears in more than one table.

Cartesian Product

- A Cartesian product is formed when:
  - A join condition is omitted
  - A join condition is invalid
  - All rows in the first table are joined to all rows in the second table
- To avoid a Cartesian product, always include a valid join condition in a WHERE clause.

Generating a Cartesian Product

EMPLOYEES (20 rows)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

DEPARTMENTS (8 rows)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows selected.

Cartesian product:
20 x 8 = 160 rows

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700
...		

160 rows selected.

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

What Is an Equijoin?

EMPLOYEES

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20

...
14 rows selected.

DEPARTMENTS

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS

...
14 rows selected.

Foreign key Primary key

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Retrieving Records with Equijoins

```
SELECT   employee_id, last_name,
         employees.department_id, d.location_id
FROM     employees e, departments d
WHERE    e.department_id = d.department_id;
```

```
EMPNO ENAME   DEPTNO DEPTNO LOC
----- ------ ------ ------ ----------
 7839 KING         10     10 NEW YORK
 7698 BLAKE        30     30 CHICAGO
 7782 CLARK        10     10 NEW YORK
 7566 JONES        20     20 DALLAS
...
14 rows selected.
```

Qualifying Ambiguous Column Names

- Use table prefixes to qualify column names that are in multiple tables.
- Improve performance by using table prefixes.
- Distinguish columns that have identical names but reside in different tables by using column aliases.

Using Table Aliases

- Simplify queries by using table aliases.

```
SELECT e.employee_id, e.last_name, d.department_no,
       d.location_id
FROM   employees e, departments d
WHERE  e.department_id=d.department_id;
```

```
SELECT e.empno, e.ename, e.deptno,
       d.deptno, d.loc
FROM   emp e, dept d
WHERE  e.deptno=d.deptno;
```

Additional Search Conditions Using the AND Operator

EMPLOYEES

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20

...
14 rows selected.

DEPARTMENTS

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS

...
14 rows selected.

Additional Search Conditions Using the AND Operator

```
SELECT   e.employee_id, e.last_name,
d.department_no, d.location_id
FROM     EMPLOYEES E, DEPARTMENTS D
WHERE    e.department_id = d.department_id
AND      last_name = 'KING';
```

Joining More Than Two Tables

CUSTOMER

NAME	CUSTID
JOCKSPORTS	100
TKB SPORT SHOP	101
VOLLYRITE	102
JUST TENNIS	103
K+T SPORTS	105
SHAPE UP	106
WOMENS SPORTS	107
...	...

9 rows selected.

ORDER

CUSTID	ORDID
101	610
102	611
104	612
106	601
102	602
106	
106	
...	

21 rows s

ITEM

ORDID	ITEMID
610	3
611	1
612	1
601	1
602	1
...	

64 rows selected.

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Joining More Than Two Tables…

```
SELECT   Name, ItemId
FROM Customer C, Ord O, Item I
WHERE    C.custId = O.custId
AND      O.ordId = I.ordId;
```

Non-Equijoins

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950

...
14 rows selected.

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

"salary in the EMP table is between low salary and high salary in the SALGRADE table"

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Retrieving Records with Non-Equijoins

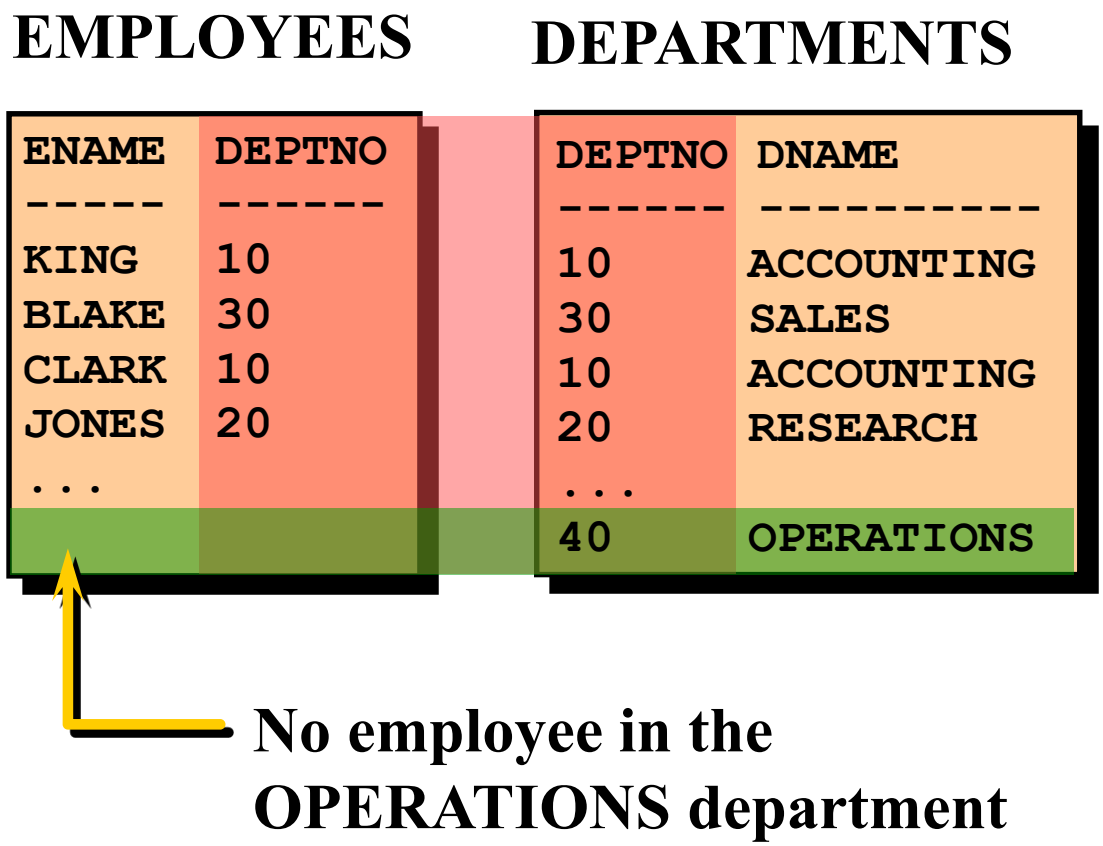
```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRA
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

...

20 rows selected.

Outer Joins


EMPLOYEES
DEPARTMENTS
ENAME DEPTNO
KING 10
BLAKE 30
CLARK 10
JONES 20
...
DEPTNO DNAME
10 ACCOUNTING
30 SALES
10 ACCOUNTING
20 RESEARCH
...
40 OPERATIONS
No employee in the
OPERATIONS department

Outer Joins

- You use an outer join to also see rows that do not usually meet the join condition.
- Outer join operator is the plus sign (+).

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

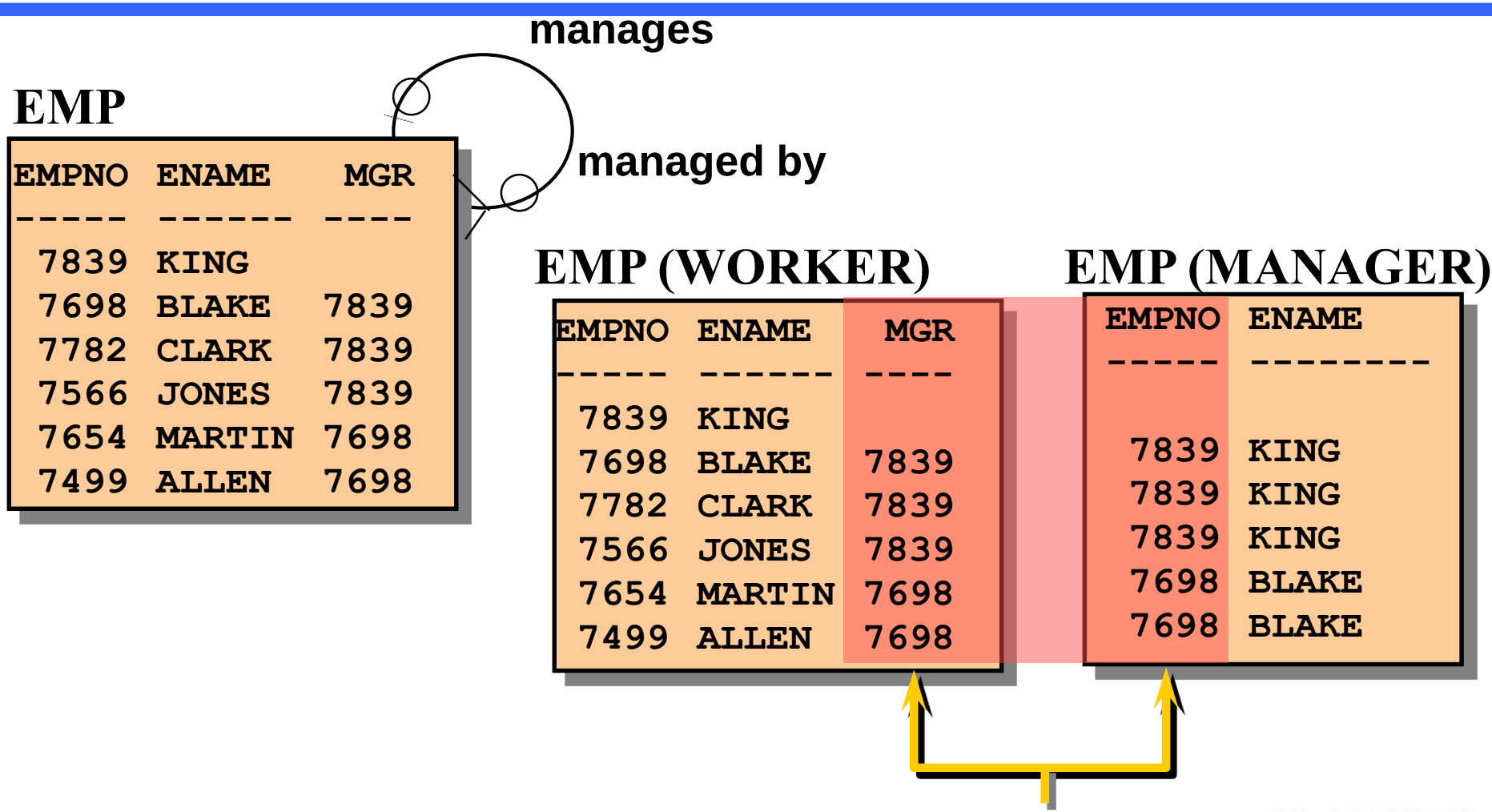
Using Outer Joins

```
SQL> SELECT e.last_name, d.department_id, d.dname
  2  FROM     employees e, departments d
  3  WHERE    e.department_id(+) = d.department_id
  4  ORDER BY e.department_id;
```

```
ENAME          DEPTNO DNAME
---------- --------- -------------
KING               10 ACCOUNTING
CLARK              10 ACCOUNTING
...
                   40 OPERATIONS
15 rows selected.
```

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Self Joins



"MGR in the WORKER table is equal to EMPNO in the MANAGER table"

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

Joining a Table to Itself (Self Joins)

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     (e.manager_id = m.employee_id);
```

EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

...

19 rows selected.

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?

Subquery Syntax

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

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

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

Types of Subqueries

- **Single-row subquery**

- **Multiple-row subquery**

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

AUT UNIVERSITY
TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

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

...

10 rows selected.

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
```

Lab Activities

- Complete exercise handed out in lab