Chapter 2 (cont)
Displaying Data from Multiple Tables

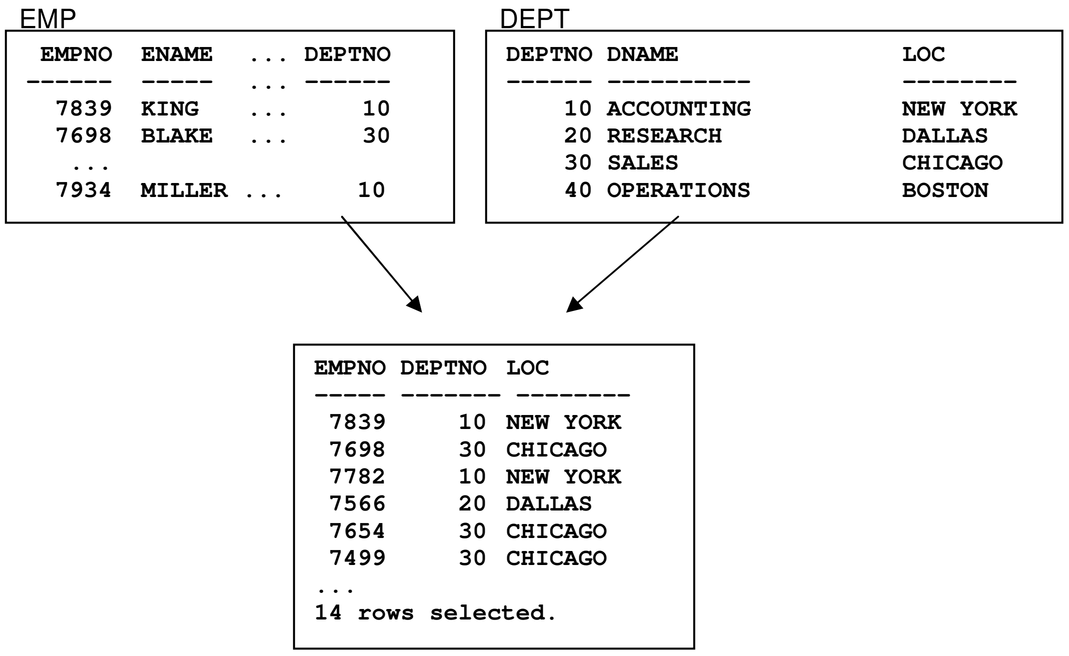
Objectives

- Write SELECT statements to access data from more than one table using equality and nonequality joins
- Generate a Cartesian product of all rows from two or more tables
- View data that generally does not meet a join condition by using OUTER joins
- Join a table to itself using a self –join

Join Standards

- Before the Oracle 9i release, the join syntax was different from the ANSI standards. This syntax is still very common within the BC Government.
- With Oracle 9i and later releases, can use either the Oracle Join Syntax or the SQL 1999 compliant syntax.
- We will look at both

Obtaining Data from Multiple Tables



EMP

```
 EMPNO  ENAME   ... DEPTNO
------  -----   ... ------
  7839  KING    ...     10
  7698  BLAKE   ...     30
   ...
  7934  MILLER ...      10
```

DEPT

```
DEPTNO DNAME               LOC
------ ----------          --------
    10 ACCOUNTING          NEW YORK
    20 RESEARCH            DALLAS
    30 SALES               CHICAGO
    40 OPERATIONS          BOSTON
```

```
EMPNO DEPTNO  LOC
----- ------- --------
 7839     10  NEW YORK
 7698     30  CHICAGO
 7782     10  NEW YORK
 7566     20  DALLAS
 7654     30  CHICAGO
 7499     30  CHICAGO
...
14 rows selected.
```

What Is a Join?

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

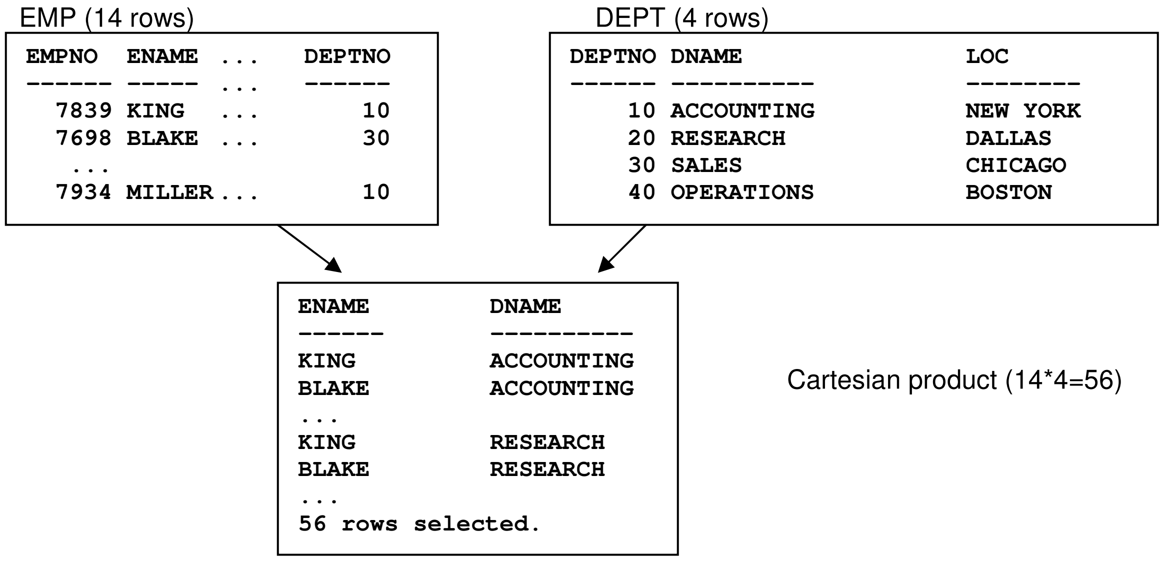
```
SELECT     table1.column, table2.column
FROM       table1, table2
WHERE      table1.column1 = table2.column2;
```

  – Write the join condition in the WHERE clause. (Oracle Join Syntax)
  – 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



EMP (14 rows)

```
EMPNO  ENAME ...    DEPTNO
------ ----- ...    ------
  7839 KING  ...        10
  7698 BLAKE ...        30
   ...
  7934 MILLER...        10
```

DEPT (4 rows)

```
DEPTNO DNAME              LOC
------ ----------         --------
    10 ACCOUNTING         NEW YORK
    20 RESEARCH           DALLAS
    30 SALES              CHICAGO
    40 OPERATIONS         BOSTON
```

```
ENAME        DNAME
------       ----------
KING         ACCOUNTING
BLAKE        ACCOUNTING
...
KING         RESEARCH
BLAKE        RESEARCH
...
56 rows selected.
```

Cartesian product (14*4=56)

Cross Joins (99)

- A Cartesian product is also known as a cross join.
- In SQL:1999 use the words CROSS JOIN explicitly to create a cartesian product.
- Note that the join occurs in the FROM clause.

```
SELECT ename, dname
FROM emp CROSS JOIN dept;
```

Types of Joins

- Equijoin
- Non-equijoin
- Outer join
- Self join

What Is an Equijoin?

- If the query is relating two tables using an equality operator (=), it is an Equijoin.

EMP

```
EMPNO ENAME    DEPTNO
------ ------- -------
  7839 KING         10
  7698 BLAKE        30
  7782 CLARK        10
  7566 JONES        20
```

Foreign Key

DEPT

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

Primary Key

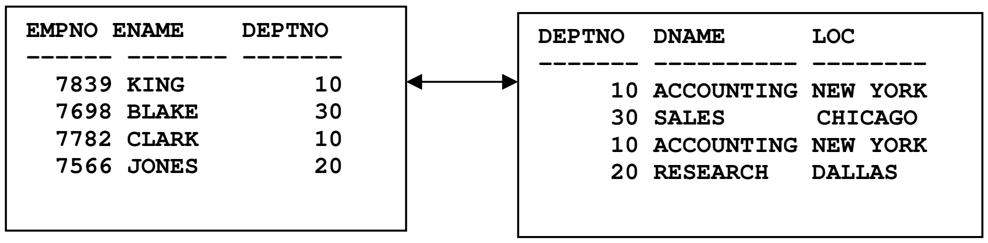
Retrieving Records with Equijoins

```
SQL> SELECT   emp.empno, emp.ename, emp.deptno,
  2           dept.deptno, dept.loc
  3  FROM     emp, dept
  4  WHERE    emp.deptno=dept.deptno;


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.



```
EMPNO ENAME    DEPTNO
------ ------- -------
  7839 KING         10
  7698 BLAKE        30
  7782 CLARK        10
  7566 JONES        20
```

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

Using Table Aliases

- Simplify queries by using table aliases.

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
  2              dept.deptno, dept.loc
  3  FROM   emp, dept
  4   WHERE  emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno,
  2         d.deptno, d.loc
  3  FROM   emp e, dept d
  4  WHERE  e.deptno=d.deptno;
```

Inner Join

- The ANSI/ISO SQL:1999 standard defines the join we just saw as a Natural Join.
- A inner join is a join based on all the columns in two tables that have the same name and same data types.
- A inner join selects rows from the two tables that have equal values in all the matched columns.

```
SELECT department_name, city
FROM departments INNER JOIN locations;
```

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

Becomes:

```
SELECT empno, ename, deptno, loc
FROM   emp INNER JOIN dept;
```

The USING clause (99)

- The USING clause specifies the columns that have to be used for an equijoin between two tables.
- The column name should be the same in both tables, and should have compatible data types.
- The columns reference by the USING clause should not have qualifiers anywhere in the SQL statement, including a WHERE clause.
- For example:
  - You want the last names of all employees and the names of the departments in which they work.
  - Will use the departments and employees tables.

The USING clause (99)

- Can you use a natural join?
- Check the structure of both tables.
- The two tables have two columns in common, manager_id and department_id. But only want to join on department_id.
- Would do the following:

```
SELECT e.last_name, d.department_name
FROM hr.employees e INNER JOIN hr.departments d
USING (department_id);
```

- How would you write this in the previous Oracle way?

The ON Clause (99)

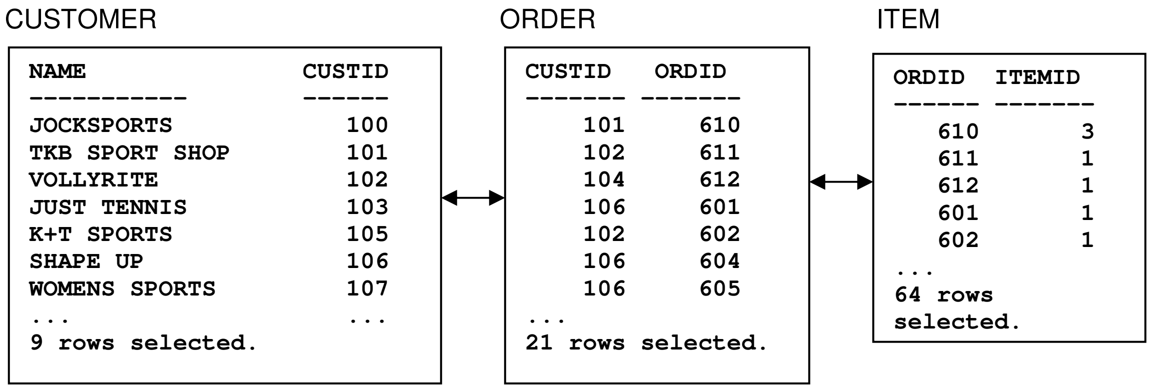
- Can use the ON clause to specify the join condition for the join of two tables.
- This clause separates the join condition from other filter conditions.

- Example:
  - Suppose you need to list the last names of all employees and the departments in which they work.
  - Need to perform an equijoin of the employees and department tables based on the department_id column.

```
SELECT e.last_name, d.department_name
FROM hr.employees e INNER JOIN hr.departments d
ON (e.department_id = d.department_id);
```

- What is the difference between the USING clause and the ON clause?
  - Can use USING only if the column names are the same in both tables.
  - If the column names are different, must use ON.

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     604
    106     605
...
21 rows selected.
```

ITEM

```
ORDID  ITEMID
------ -------
   610       3
   611       1
   612       1
   601       1
   602       1
...
64 rows
selected.
```

…where customer.custid = ord.custid and ord.ordid = item.ordid

Multitable Join

- Want to join more than two tables.
- In SQL:1999 compliant syntax, join are performed from left to right.
- Example:
  - – List the names of employees, the names of the departments in which they work, and the name of the city in which the department is located.
    - Need the employees, departments, and locations table.

```
SELECT employee_id, city, department_name
FROM hr.locations l INNER JOIN hr.departments d
ON (d.location_id = l.location_id)
INNER JOIN hr.employees e
ON (d.department_id = e.department_id);
```

or

```
SELECT employee_id, city, department_name
FROM hr.locations l INNER JOIN hr.departments d
USING (location_id)
INNER JOIN hr.employees e
USING (department_id);
```

- It is important to note that the first join to be performed is *locations INNER JOIN departments*
- This join can reference columns in locations and departments, but not employees.
- The second join can reference columns from all three tables.

Multitable Join - Oracle

```
SELECT employee_id, city, department_name
FROM hr.locations l, hr.departments d, hr.employees e
WHERE d.location_id = l.location_id
AND d.department_id = e.department_id;
```

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.
```

Want Salaries between HISAL & LOSAL

SALGRADE

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

Retrieving Records with Non-Equijoins

```
SQL>  SELECT           e.ename, e.sal, s.grade
   2  FROM             emp e, salgrade s
   3  WHERE            e.sal
   4  BETWEEN          s.losal AND s.hisal;
```

```
ENAME          SAL     GRADE
---------- ------- ---------
JAMES          950         1
MARTIN        1250         2
ALLEN         1600         3
TURNER        1500         3
BLAKE         2850         4 …
```

Outer Joins

– To see data from one table even if there is no corresponding row in the joining table.

No employee in the Operations department

EMP

```
ENAME        DEPTNO
-----        ------
KING         10
BLAKE        30
CLARK        10
JONES        20
...
```

DEPT

```
DEPTNO DNAME
------ ----------
10     ACCOUNTING
30     SALES
10     ACCOUNTING
20     RESEARCH
...
40     OPERATIONS
```

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.ename, d.deptno, d.dname
  2  FROM            emp e, dept d
  3  WHERE           e.deptno(+) = d.deptno
  4  ORDER BY        e.deptno;


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

Get all departments even if there isn't a matching employee.

Using Outer Joins

– What would happen for the following?

```
SQL> SELECT             e.ename, d.deptno, d.dname
  2  FROMemp e, dept d
  3  WHERE              e.deptno = d.deptno(+)
  4  ORDER BY           e.deptno;
```

Outer Join (99)

- In SQL:1999, the syntax of the outer joins has changed.
- Need to know if you want a:
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join

Left Outer Join (99)

- A left outer join returns all the rows from the table specified on the left side of the JOIN keyword, and the rows that satisfy the join condition from the table that is specified on the right of the join keyword.

Left Outer Join (99) Example

- Suppose you need a list of all employees and the departments they belong to. This should include the employees who have not yet been assigned to any department.

```
SELECT e.last_name,d.department_name
FROM hr.employees e LEFT OUTER JOIN hr.departments d
ON (e.department_id = d.department_id);
```

- And the equivalent previous Oracle SQL?

Right Outer Join (99)

- A right outer join returns all the rows from the table specified on the right side of the JOIN keyword, and the rows that satisfy the join condition from the table that is specified on the left of the join keyword.

Right Outer Join (99) Example

Suppose you need a list of all employees and the departments they belong to. This should include the departments to which no employees have yet been assigned

```
SELECT e.last_name,d.department_name
FROM hr.employees e RIGHT OUTER JOIN hr.departments d
ON (e.department_id = d.department_id);
```

- And the Oracle version?

Full Outer Join (99)

- A full outer join returns all the rows from the tables that satisfy the join condition as well as the results of both the left and the right outer joins.
- This is new to Oracle9i +

Full Outer Join (99) Example

- Suppose you need a list of all employees and the departments they belong to. This should include the employees who have not yet been assigned to any department.

```
SELECT e.last_name,d.department_name
FROM hr.employees e FULL OUTER JOIN hr.departments d
ON (e.department_id = d.department_id);
```

- No direct previous Oracle version.
- Would need to use a UNION as follows:

```
SELECT e.last_name,d.department_name
FROM hr.employees e, hr.departments d
WHERE e.department_id(+) = d.department_id
UNION
SELECT e.last_name,d.department_name
FROM hr.employees e, hr.departments d
WHERE e.department_id = d.department_id(+);
```

Self Joins

– Joining a Table to Itself

EMP(WORKER)

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698

EMP(MANAGER)

EMPNO	ENAME
7839	KING
7839	KING
7839	KING
7698	BLAKE
7698	BLAKE

“MGR in the WORKER table is equal to EMPNO in the MANAGER table”

Joining a table to itself

```
SQL> SELECT worker.ename||' works for '||manager.ename
  2  FROM        emp worker, emp manager
  3  WHERE       worker.mgr = manager.empno;
```

```
WORKER.ENAME||'WORKSFOR'||MANAG
-------------------------------
BLAKE works for KING
CLARK works for KING
JONES works for KING
MARTIN works for BLAKE
...
13 rows selected.
```

Joining a Table to Itself (99)

```
SELECT e.last_name ||   ' works for ' || m.last_name
FROM hr.employees e INNER JOIN hr.employees m
ON (e.manager_id = m.employee_id);
```

Set Operators

- Used to select data from multiple tables
- Combine the results of two queries into one
- Are compound queries
- All set operators have equal precedence – if multiples, evaluate from left to right
- Data types of the resulting columns should match in both queries

Set Operators

- Oracle has four set operators
  - UNION: returns all unique rows selected by either query
  - UNION ALL: returns all rows, including duplicates selected by either query
  - INTERSECT: returns rows selected from both queries
  - MINUS: returns unique rows selected by the first query but not the rows selected by the second query

UNION

– All unique rows selected by either query.

EAST_ORDER

```
PRODID       NAME
-----        ------
 1101        BILL
 1102        SCOTT
 1101        SCOTT
 1103        JANE
 1103        MARY
```

WEST_ORDER

```
PRODID       NAME
-----        ------
 1103        BILL
 1102        MARY
 1101        STEVE
 1103        SCOTT
```

```
SELECT name FROM east_order
UNION
SELECT name FROM west_order;
```

UNION ALL

– ALL rows selected by either query.

```
SELECT name FROM east_order
UNION
SELECT name FROM west_order;
```

INTERSECT

– Rows selected from both queries

```
SELECT name FROM east_order
INTERSECT
SELECT name FROM west_order;
```

MINUS

– Unique rows selected by first query but not rows selected by second.

```
SELECT name FROM east_order
MINUS
SELECT name FROM west_order;
```

SUMMARY

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

For joins: Equijoin, Non-equijoin, Outer join, Self join

```
SELECT     column FROM     table
{UNION, UNION ALL, INTERSECT, MINUS}
SELECT     column FROM     table;
```

SET OPERATORS