Objectives

- Identify a Cartesian join
- Create an equality join using the WHERE clause
- Create an equality join using the JOIN keyword
- Create a non-equality join using the WHERE clause
- Create a non-equality join using the JOIN…ON approach
Objectives (continued)

• Create a self-join using the WHERE clause
• Create a self-join using the JOIN keyword
• Distinguish an inner join from an outer join
• Create an outer join using the WHERE clause
• Create an outer join using the OUTER keyword
• Use set operators to combine the results of multiple queries
Purpose of Joins

• Joins are used to link tables and reconstruct data in a relational database
• Joins can be created through:
  – Conditions in a WHERE clause
  – Use of JOIN keywords in FROM clause
Cartesian Joins

• Created by omitting joining condition in the WHERE clause or through CROSS JOIN keywords in the FROM clause
• Results in every possible row combination \((m \times n)\)
Cartesian Join Example: Omitted Condition

```
SELECT title, name
FROM books, publisher;
```
Cartesian Join Example: CROSS JOIN Keywords

```
SELECT isbn, title, location, ' ' Count
FROM books CROSS JOIN warehouses
ORDER BY location, title;
```
Equality Joins

- Link rows through equivalent data that exists in both tables
- Created by:
  - Creating equivalency condition in the WHERE clause
  - Using NATURAL JOIN, JOIN…USING, or JOIN…ON keywords in the FROM clause
Equality Joins: WHERE Clause Example

SELECT title, name
FROM books, publisher
WHERE books.pubid = publisher.pubid;
Qualifying Column Names

- Columns in both tables must be qualified

```
SELECT title, pubid, name
FROM books, publisher
WHERE books.pubid = publisher.pubid;
```

**FIGURE 9-8** Column ambiguously defined error
WHERE Clause Supports Join and Other Conditions

![SQL query](image)

**FIGURE 9-9** Including search and join conditions in a WHERE clause
Joining More Than Two Tables

• Joining 4 tables requires 3 join conditions

```
SELECT lastname, firstname, title
FROM customers c, orders o, orderitems oi, books b
WHERE c.customer# = o.customer#
  AND o.order# = oi.order#
  AND oi.isbn = b.isbn
ORDER BY lastname, firstname;
```
Equality Joins: NATURAL JOIN

```sql
SELECT title, pubid, name
FROM publisher NATURAL JOIN books;
```
No Qualifiers with a NATURAL JOIN

```
SELECT title, p.pubid, name
FROM publisher p NATURAL JOIN books b;
```

**FIGURE 9-14** Column qualifier error with a NATURAL JOIN

ERROR at line 1:
ORA-25155: column used in NATURAL join cannot have qualifier
Equality Joins: JOIN...USING

```
SELECT title, pubid, name
FROM publisher JOIN books
USING (pubid);
```
Equality Joins: JOIN…ON

- Required if column names are different

```
SELECT title, pubid, name
FROM publisher2 p JOIN books b
  ON p.id = b.pubid;
```
JOIN Keyword Overview

• Use JOIN…USING when tables have one or more columns in common
• Use JOIN…ON when same named columns are not involved or a condition is needed to specify a relationship other than equivalency (next section)
• Using the JOIN keyword frees the WHERE clause for exclusive use in restricting rows
Non-Equality Joins

- In WHERE clause, use any comparison operator other than the equal sign
- In FROM clause, use JOIN…ON keywords with a non-equivalent condition
Non-Equality Joins: WHERE Clause Example

```sql
SELECT title, gift
FROM books, promotion
WHERE retail BETWEEN minretail AND maxretail;
```
Non-Equality Joins: JOIN…ON
Example

```
SELECT title, gift
FROM books JOIN promotion
    ON retail BETWEEN minretail AND maxretail;
```
Self-Joins

• Used to link a table to itself
• Requires the use of table aliases
• Requires the use of a column qualifier
## Customer Table Example

**Customer 1003**

(Leila Smith) has referred two customers
(Tammy Giana and Jorge Perez)

<table>
<thead>
<tr>
<th>CUSTOMER#</th>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>ADDRESS</th>
<th>CITY</th>
<th>STATE</th>
<th>ZIP</th>
<th>REFERRED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>MORALES</td>
<td>BONITA</td>
<td>P.O. BOX 651</td>
<td>EASTPOINT</td>
<td>FL</td>
<td>32328</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>THOMPSON</td>
<td>RYAN</td>
<td>P.O. BOX 9835</td>
<td>SANTA MONICA</td>
<td>CA</td>
<td>90404</td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td>SMITH</td>
<td>LEILA</td>
<td>P.O. BOX 66</td>
<td>TALLAHASSEE</td>
<td>FL</td>
<td>32306</td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td>PIERSON</td>
<td>THOKAS</td>
<td>68921 SOUTH AVENUE</td>
<td>BOISE</td>
<td>ID</td>
<td>83707</td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>GIRARD</td>
<td>CINDY</td>
<td>P.O. BOX 851</td>
<td>SEATTLE</td>
<td>WA</td>
<td>98115</td>
<td></td>
</tr>
<tr>
<td>1006</td>
<td>CRUZ</td>
<td>MESHIA</td>
<td>62 DIRT ROAD</td>
<td>ALBANY</td>
<td>NY</td>
<td>12211</td>
<td></td>
</tr>
<tr>
<td>1007</td>
<td>GIANA</td>
<td>TAMMY</td>
<td>9153 MAIN STREET</td>
<td>AUSTIN</td>
<td>TX</td>
<td>78710</td>
<td>1003</td>
</tr>
<tr>
<td>1008</td>
<td>JONES</td>
<td>KENNETH</td>
<td>P.O. BOX 137</td>
<td>CHEYENNE</td>
<td>WY</td>
<td>82003</td>
<td></td>
</tr>
<tr>
<td>1009</td>
<td>PEREZ</td>
<td>JORGE</td>
<td>P.O. BOX 8564</td>
<td>BURBANK</td>
<td>CA</td>
<td>91510</td>
<td>1003</td>
</tr>
<tr>
<td>1010</td>
<td>LUCAS</td>
<td>JAKE</td>
<td>114 EAST</td>
<td>SAVANNAH</td>
<td>GA</td>
<td>30314</td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td>MCGOVERN</td>
<td>REESE</td>
<td>P.O. BOX 18</td>
<td>CHICAGO</td>
<td>IL</td>
<td>60906</td>
<td></td>
</tr>
<tr>
<td>1012</td>
<td>MCKENZIE</td>
<td>WILLIAM</td>
<td>P.O. BOX 971</td>
<td>BOSTON</td>
<td>MA</td>
<td>02110</td>
<td></td>
</tr>
<tr>
<td>1013</td>
<td>NGUYEN</td>
<td>NICHOLAS</td>
<td>357 WHITE EAGLE AVE</td>
<td>CLERMONT</td>
<td>FL</td>
<td>34711</td>
<td>1006</td>
</tr>
</tbody>
</table>

**Customer 1006**

(Mesha Cruz) has referred one customer
(Nicholas Nguyen)

**FIGURE 9-22** Two columns of the same table are related (Partial table shown)
Self-Joins: WHERE Clause Example

```sql
SELECT r.firstname, r.lastname, c.lastname Referred
FROM customers c, customers r
WHERE c.referred = r.customer#;
```

**FIGURE 9-23** A self-join constructed in the WHERE clause
Self-Joins: JOIN...ON Example

```
SELECT r.firstname, r.lastname, c.lastname Referred
FROM customers c JOIN customers r
    ON c.referred = r.customer#;
```

**FIGURE 9-24** A self-join using the JOIN...ON keywords
Outer Joins

• Use outer joins to include rows that do not have a match in the other table
• In WHERE clause, include outer join operator (+) immediately after the column name of the table with missing rows to add NULL rows
• In FROM clause, use FULL, LEFT, or RIGHT with OUTER JOIN keywords
Outer Joins: WHERE Clause Example

```
SELECT lastname, firstname, order#
FROM customers c, orders o
WHERE c.customer# = o.customer#(+)
ORDER BY c.customer#;
```
Outer Joins: OUTER JOIN
Keyword Example

```
SELECT lastname, firstname, order#
FROM customers c LEFT OUTER JOIN orders o
ON c.customer# = o.customer#
ORDER BY c.customer#;
```
Outer Joins (continued)

• If multiple join conditions are used, the outer join condition may be required in all the join conditions to retain non-matching rows
Set Operators

- Used to combine the results of two or more SELECT statements

<table>
<thead>
<tr>
<th>SET OPERATOR</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNION</td>
<td>Returns the results of the combined SELECT statements and suppresses duplicates</td>
</tr>
<tr>
<td>UNION ALL</td>
<td>Returns the results of the combined SELECT statements but does not suppress duplicates</td>
</tr>
<tr>
<td>INTERSECT</td>
<td>Returns only the rows included in the results of both SELECT statements</td>
</tr>
<tr>
<td>MINUS</td>
<td>Removes the results of the second query that are also found in the first query and displays only the rows that are uniquely returned by only the first query</td>
</tr>
</tbody>
</table>

**FIGURE 9-29** List of set operators

Oracle 10g: SQL
Set Operators: UNION Example

FIGURE 9-30 Producing an unduplicated combined list using the UNION set operator
Set Operators: INTERSECT

Example

```
SELECT customer# FROM customers
INTERSECT
SELECT customer# FROM orders;
```
Set Operators: MINUS Example

**FIGURE 9-33** Subtracting result sets using the MINUS set operator
Summary

- Data stored in multiple tables regarding a single entity can be linked together through the use of joins.
- A Cartesian join between two tables returns every possible combination of rows from the tables; the resulting number of rows is always $m \times n$.
- An equality join is created when the data joining the records from two different tables are an exact match.
- A non-equality join establishes a relationship based upon anything other than an equal condition.
- Self-joins are used when a table must be joined to itself to retrieve needed data.
Summary (continued)

- Inner joins are categorized as being equality, non-equality, or self-joins
- An outer join is created when records need to be included in the results without having corresponding records in the join tables
  - The record is matched with a NULL record so it will be included in the output
- Set operators such as UNION, UNION ALL, INTERSECT, and MINUS can be used to combine the results of multiple queries