Lesson 7
More SQL: Complex Queries
Joining Data from MultipleTables

OUTLINE

- 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
- 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

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 * n)

Cartesian Join Example: Omitted Condition

- Not including a joining condition in a WHERE clause (implicit)

```
SELECT *
FROM department, project;
```

- That CROSS JOIN query generates 99 rows.
- (There were 9 department rows and 11 project rows, thus giving 9× 11 = 99rows.)

Cartesian Join Example: CROSS JOIN Keywords

- Using the JOIN method with the CROSS JOIN keywords (explicit)

```
SELECT *
FROM department
CROSS JOIN project;
```

Equality Joins

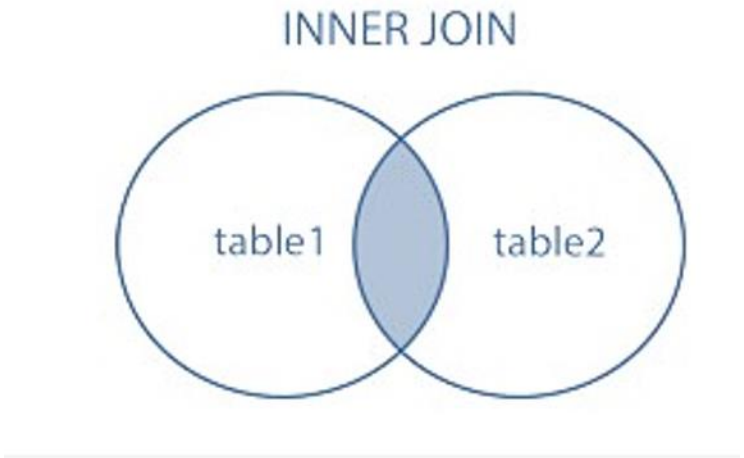
- Link rows through equivalent data that exists in both tables
- Created by:
  - Creating equivalency condition in the WHERE clause
  - Using NATURAL JOIN, JOIN…USING, or JOIN…ON
  - keywords in the FROM clause

Inner Joins

- An INNER join is the type of join most often needed.
- It requires a value in the row of one table to match a value in a row contained in the other table.
- An inner join creates a join by using a commonly named and defined column.

Inner Joins (continued)

- An inner join can be created by two methods:
  - Using the WHERE clause (SQL/86 standard). Using the JOIN method with the NATURAL JOIN, JOIN . . .ON, or JOIN . . . USING keywords. (SQL/92
  - introduced the INNER JOIN and ON clauses for performing an inner join).



Equality Joins: WHERE Clause Example

```
SELECT e.fname, e.lname
FROM employee e, dependent d
WHERE e.ssn = d.empssn
AND e.lname = d.dependent_name;
```

Qualifying Column Names

- Columns in both tables must be qualified
- Suppose that a table *mytable1* contains columns a and b, and a table *mytable2* contains columns b and c.
- References to columns a or c are unambiguous, but references to b must be qualified as either *mytbl1.b* or *mytbl2.b*:

```
SELECT a, mytable1.b, mytable2.b, c
FROM mytable1
INNER JOIN mytable2 ... ;
```

WHERE Clause Supports Join and Other Conditions

```
SELECT fname, lname , e.dno
FROM employee e
JOIN department d ON (e.dno = d.dnumber)
WHERE dname='Research';
```

Joining More Than Two Tables

- Joining four tables requires three join conditions

```
SELECT d.dname,lname,fname, pname
FROM department d,project p ,works_on a,employee e
WHERE d.dnumber = p.dnum
AND p.pnumber = a.pno
AND a.essn = e.ssn
ORDER by dname,lname,fname;
```

Equality Joins: JOIN…USING

- A second way to express a join is through the
- USING keyword.
- That query returns only the rows with matching values in the column indicated in the USING clause- and that column must exist in both tables.
- The syntax is:

```
SELECT   columnlist
FROM table1
JOIN   table2
USING  (common-column)
```

Equality Joins: JOIN…USING (continued)

```
SELECT dname, location
FROM department
JOIN dept_locations USING  (dnumber);
```

Equality Joins: JOIN…ON

- Another way to express a join when the tables have no common attribute names is to use the JOIN ON operand.
- The join condition will typically include an equality comparison expression of two columns.
- The syntax is:

```
SELECT column-list
FROM table1
JOIN table2
ON join-condition
```

Equality Joins: JOIN…ON (continued)

- Required if column names are different

```
SELECT e.fname, e.lname , d.dependent_name
FROM employee e
JOIN dependent d ON e.ssn = d.empssn;
```

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

```
SELECT fname,lname
FROM employee e
JOIN department d ON (e.dno = d.dnumber)
WHERE (dname <> 'Research');
```

Self-Joins

- Used to link a table to itself Requires the use of table aliases
- Requires the use of a column qualifier

Self-Joins: WHERE Clause Example

```
SELECT e.fname, e.lname, s.fname, s.lname
FROM employee e , employee s
WHERE e.superssn = s.ssn;
```

Self-Joins: JOIN…ON Example

```
SELECT e.fname, e.lname, s.fname, s.lname
FROM employee e
JOIN employee s ON e.superssn = s.ssn;
```

Outer Joins

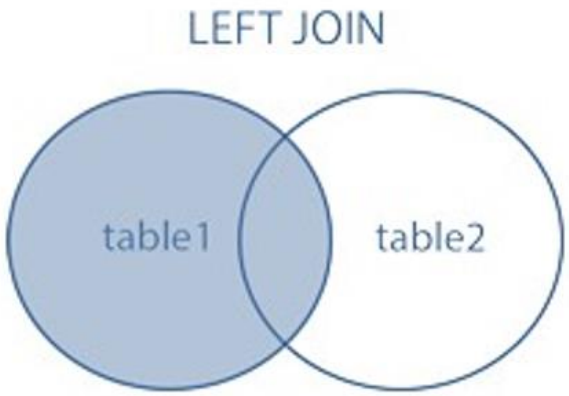
- Use outer joins to include rows that do not have a match in the other table
- In FROM clause, use FULL, LEFT, or RIGHT with OUTER JOIN keywords
- If multiple join conditions are used, the outer join condition may be required in all of the join conditions to retain nonmatching rows

INNER and OUTER Joins

- INNER JOIN **(versus** OUTER JOIN**)**
  - Default type of join in a joined table
  - Tuple is included in the result only if a matching tuple exists in the other relation
- LEFT OUTER JOIN
  - Every tuple in left table must appear in result
  - If no matching tuple
    - Padded with NULL values for attributes of right table
- RIGHT OUTER JOIN
  - Every tuple in right table must appear in result
  - If no matching tuple
    - Padded with NULL values for attributes of left table

Left Outer Joins

- The LEFT OUTER JOIN returns not only the rows matching the join condition, but also the rows in the left side table with unmatched values in the right side table.
- When you use a left outer join, the result set includes all the rows from the first, or left, table.



Left Outer Joins (continued)

- LEFT OUTER JOIN

  Every tuple in left table must appear in result If no matching tuple

  - Padded with NULL values for attributes of right table

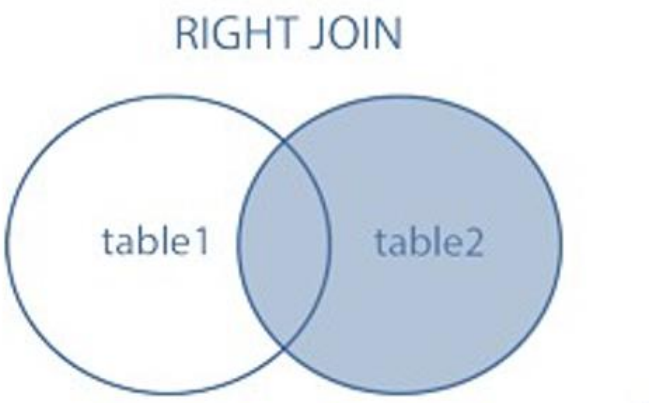
- syntax is:

  SELECT column-list
  FROM table1
  LEFT [OUTER] JOIN table2
  ON join-condition

  - table1 and table2 are the tables to join.
  - LEFT specifies a left outer join.

Right Outer Joins

- The RIGHT OUTER JOIN returns not only the rows matching the join condition, but also the rows in the right side table with unmatched values in the left side table.
- When you use a right outer join, the result set includes all the rows from the second, or right, table.



Right Outer Joins (continued)

- Right OUTER JOIN

  Every tuple in right table must appear in result If no matching tuple

  - Padded with NULL values for attributes of left table
- The syntax is:

  SELECT column-list

  FROM table1

  RIGHT [OUTER] JOIN

  table2

  ON join-condition

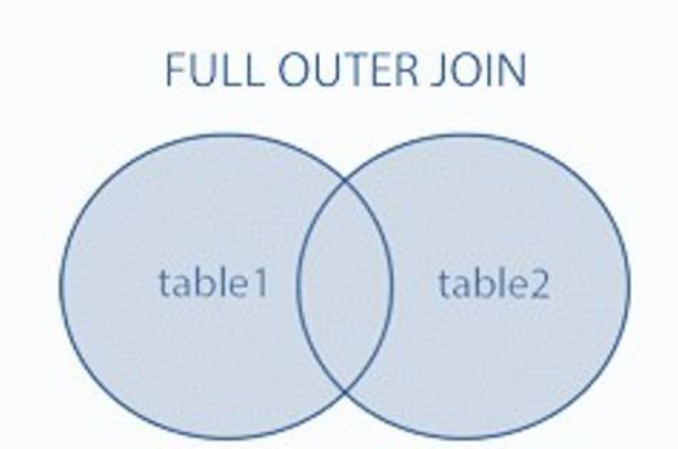
  - table1 and table2 are the tables to join.
  - RIGHT specifies a right outer join.

Outer Joins: OUTER JOIN Keyword Example

```
SELECT dname, d.dnumber, pname, p.dnum
FROM project p
RIGHT JOIN department d
ON (d.dnumber = p.dnum);
```

Full Outer Joins

- The FULL OUTER JOIN keyword returns all rows from the left table (table1) and from the right table (table2).
- The FULL OUTER JOIN keyword combines the
- result of both LEFT and RIGHT joins.
- **Note**: **MySQL doesn't support FULL OUTER JOIN**



Outer Joins (continued)

- If multiple join conditions are used, the outer join condition may be required in all of the join conditions to retain nonmatching rows

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* * *n*
- An equality join is created when the data joining the records from two different tables are an exact match

Summary (continued)

- 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
- 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