Course Chapter Map

Learning Objectives

- Combine data from multiple tables, using the **JOIN** keyword
- Explain the concept of a **JOIN**
- Use **OUTER** and **INNER** joins

Combining Multiple Tables (1/2)

- **SELECT**s can be use to retrieve data from multiple tables
  - Some questions cannot be answered by querying only one table
- A *join* operation is used to combine tables for a query
  - Joins data in one table with data in another table

Combining Multiple Tables (2/2)

- Tables to be joined

```
table1                                  table2
+----+----+                             +----+----+
| i1 | c1 |                             | i2 | c2 |
+----+----+                             +----+----+
|  1 | a  |             AND             |  2 | c  |
|  2 | b  |                             |  3 | b  |
|  3 | c  |                             |  4 | a  |
+----+----+                             +----+----+
3 rows in set (0.00 sec)                3 rows in set (0.00 sec)
```

- Joined tables *(cross join results in a 'Cartesian Product')*

```
SELECT * FROM table1, table2;
+----+----+----+----+
| i1 | c1 | i2 | c2 |
+----+----+----+----+
|  1 | a  |  2 | c  |
|  2 | b  |  2 | c  |
|  3 | c  |  2 | c  |
|  1 | a  |  3 | b  |
|  2 | b  |  3 | b  |
|  3 | c  |  3 | b  |
|  1 | a  |  4 | a  |
|  2 | b  |  4 | a  |
|  3 | c  |  4 | a  |
+----+----+----+----+
9 rows in set (0.00 sec)
```

This example is a *cross join*, which results in a *Cartesian Product.* Also known as an *Unqualified Join*.

Categories of Joins

- Cross join
  - Combines all rows from one table with all rows of another table
  - Unqualified join
- Inner join
  - matching rows from two tables
  - Qualified join
- Outer join
  - matching and non-matching rows from two tables
  - Qualified join
- Qualified joins
  - Retain only specific row pairs according to the 'join condition'
    - A **city** (***City*** *table*) is a capital of a **country** (***Country*** *table*) *and* a **country** (***Country*** *table*) has **cities** (***City*** *table*)

Inner Joins

- Identifies combinations of matching rows from two tables
- Two different types of syntax
  - Comma separated
  - **INNER JOIN** Keywords

Comma Joins

- List tables to be joined with a comma separator
- Two separate queries can be joined into one

```
mysql> SELECT Code, Name, Language
    -> FROM Country, CountryLanguage
    ->      WHERE Continent='Africa'
    ->      AND Code = CountryCode;
+------+--------------+------------------+
| Code | Name         | Language         |
+------+--------------+------------------+
| DZA  | Algeria      | Arabic           |
| DZA  | Algeria      | Berberi          |
| AGO  | Angola       | Ambo             |
| AGO  | Angola       | Chokwe           |
...
| BEN  | Benin        | Adja             |
| BEN  | Benin        | Aizo             |
...
| BWA  | Botswana     | Khoekhoe         |
| BWA  | Botswana     | Ndebele          |
...
```

Table Name Aliases

- Table reference can be aliased
  - *tbl_name* **AS** *alias_name*
  - *tbl_name alias_name*
- Examples

```
mysql> SELECT t1.Name, t2.CountryCode
     -> FROM Country AS t1, City AS t2
     -> WHERE t1.Name = t2.Name;
+------------+-------------+
| Name       | CountryCode |
+------------+-------------+
| Djibouti   | DJI         |
| Mexico     | PHL         |
| Gibraltar  | GIB         |
| Armenia    | COL         |
| Kuwait     | KWT         |
| Macao      | MAC         |
| San Marino | SMR         |
| Singapore  | SGP         |
+------------+-------------+
8 rows in set (0.47 sec)
```

OR

```
mysql> SELECT t1.Name, t2.CountryCode
    -> FROM Country t1, City t2
    -> WHERE t1.Name = t2.Name;
```

INNER JOIN Keywords

- **INNER JOIN** replaces the comma separator
- **FROM** clause
- Examples

```
mysql> SELECT t1.Name, t2.CountryCode
  -> FROM Country AS t1 INNER JOIN City AS t2
  -> ON t1.Name = t2.Name;
+------------+-------------+
| Name       | CountryCode |
+------------+-------------+
| Djibouti   | DJI         |
| Mexico     | PHL         |
| Gibraltar  | GIB         |
| Armenia    | COL         |
| Kuwait     | KWT         |
| Macao      | MAC         |
| San Marino | SMR         |
| Singapore  | SGP         |
+------------+-------------+
8 rows in set (0.34 sec)
```

OR

```
mysql> SELECT t1.Name, t2.CountryCode
    -> FROM Country AS t1
    -> INNER JOIN City AS t2
    -> USING(Name);
```

JOIN Keyword

- Equivalent to **INNER JOIN**
- Example

```
mysql> SELECT COUNT(City.Name)
    -> FROM City
    -> JOIN Country
    -> ON CountryCode = Code
    -> WHERE Continent = 'South America';
+-------------------+
| COUNT(City.Name)  |
+-------------------+
|               470 |
+-------------------+
1 row in set (0.42 sec)
```

- **ON** condition -> *How*
- **WHERE** clause -> *Which*

OUTER JOIN Keywords

- Finds tables with and without matching rows
- **LEFT JOIN**
- **RIGHT JOIN**
- Example →

```
mysql> SELECT Name, Language
    -> FROM Country
    -> LEFT JOIN CountryLanguage
    -> ON Code = CountryCode;
+------------------------------+------------------+
| Name                         | Language         |
+------------------------------+------------------+
| Aruba                        | Dutch            |
| Aruba                        | English          |
| Aruba                        | Papiamento       |
| Aruba                        | Spanish          |
| Afghanistan                  | Balochi          |
...
| Antarctica                   | NULL             |
| French Southern territories  | NULL             |
| Antigua and Barbuda          | Creole English   |
| Antigua and Barbuda          | English          |
| Australia                    | Arabic           |
| Australia                    | Canton Chinese   |
| Australia                    | English          |
| Australia                    | German           |
...
990 rows in set (0.00 sec)
```

LEFT JOIN

- Comparison for join based on the first (or left) table
- Use **WHERE** clause to find mismatches
- **LEFT JOIN** example

```
mysql> SELECT Name, Language
    -> FROM Country
    -> LEFT JOIN CountryLanguage
    -> ON Code = CountryCode
    -> WHERE CountryCode IS NULL;
+----------------------------------------------+----------+
| Name                                         | Language |
+----------------------------------------------+----------+
| Antarctica                                   | NULL     |
| Bouvet Island                                | NULL     |
| British Indian Ocean Territory               | NULL     |
| South Georgia and the South Sandwich Islands | NULL     |
| Heard Island and McDonald Islands            | NULL     |
| French Southern territories                  | NULL     |
+----------------------------------------------+----------+
6 rows in set (0.01 sec)
```

RIGHT JOIN

- Roles of tables reversed from **LEFT JOIN**
- Example

```
mysql> SELECT Name, Language
    -> FROM Country
    -> RIGHT JOIN CountryLanguage
    -> ON Code = CountryCode
    -> WHERE CountryCode IS NULL;
Empty set (0.00 sec)
```

OUTER JOIN: USING and NULL

- USING
  - Single/Multiple columns must exist in both tables

    ```
    … a LEFT JOIN b USING (c1,c2,c3)
    ```

- **NULL**
  - No matching row for right table of LEFT JOIN gives NULL result
  - Used to find rows with no counterpart in other table
  - Example

```
SELECT table1.id * FROM table1
LEFT JOIN table2
ON table1.id=table2.id
WHERE table2.id IS NULL
```

Further Practice: Chapter 10

- Comprehensive exercises

Chapter Summary

- Combine data from multiple tables, using the **JOIN** keyword
- Explain the concept of a **JOIN**
- Use **OUTER** and **INNER** joins