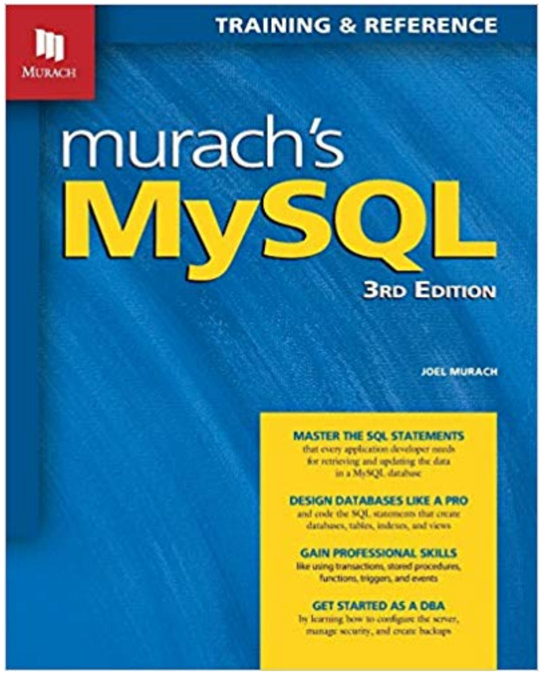
JOIN and the UNION operations

Topic 3

Lesson 3 – Retrieving data from more than one table

Chapter 4 Murach's MySQL



MURACH

TRAINING & REFERENCE

murach's MySQL

3RD EDITION

JOEL MURACH

MASTER THE SQL STATEMENTS
that every application developer needs
for retrieving and updating the data
in a MySQL database

DESIGN DATABASES LIKE A PRO
and code the SQL statements that create
databases, tables, indexes, and views

GAIN PROFESSIONAL SKILLS
like using transactions, stored procedures,
functions, triggers, and events

GET STARTED AS A DBA
by learning how to configure the server,
manage security, and create backups

UNION Operation in SQL

UNION is a set operation available in the SQL SELECT statement. It allows you to “glue” multiple compatible results together.

It is typically considered analogous to addition, since if one relation A has n distinct tuples and another relation B has m distinct tuples, then A UNION B has at most n + m tuples

UNION syntax

```
SELECT_statement_1
UNION [ALL]
    SELECT_statement_2
[UNION [ALL]
    SELECT_statement_3]...
```

Rules for performing a UNION:

- UNION COMPATIBILITY:
  - Each result set must return the same number of fields
  - The corresponding fields in each result set must have compatible data types.
- The column names in the final result set are taken from the first SELECT clause (so the names of the fields can vary in each SELECT clause).

UNION ALL behavior

- UNION has one keyword: ALL.
- Without the ALL keyword, the UNION operation will remove all duplicate tuples from the result. It functions as a set operation.
- Sometimes, you want the duplicate rows. If this is the case, then use the keyword ALL. It functions as a multiset operation; duplicate elements are allowed.

UNION example

```
SELECT yo_grad FROM student_instance1
UNION
SELECT yr_o_grad FROM student_instance2;
```

ID	Name	School	Credits_Earned	Credits_Req	Yo_grad
7	Haines	Khoury	32	120	2021
8	Lee	D’Amore McKim	64	128	2020
9	Frred	D’Amore McKim	50	120	2020

ID	Name	School	Credits_Earned	Credits_Req	Yr_o_grad
1	Smith	Khoury	32	120	2019
2	Shah	D’Amore McKim	64	128	2019
3	Li	Khoury	50	120	2020

RESULT: UNION example

```
SELECT yo_grad FROM student_instance1
UNION
SELECT yr_o_grad FROM student_instance2;
```

ID	Name	School	Credits_Earned	Credits_Req	yo_grad
7	Haines	Khoury	32	120	2021
8	Lee	D'Amore McKim	64	128	2020
9	Frred	D'Amore McKim	50	120	2020

ID	Name	School	Credits_Earned	Credits_Req	yr_o_grad
1	Smith	Khoury	32	120	2019
2	Shah	D'Amore McKim	64	128	2019
3	Li	Khoury	50	120	2020

Yo_grad
2021
2020
2019

RESULT: UNION ALL example

```
SELECT yo_grad FROM student_instance1
UNION ALL
SELECT yr_o_grad FROM student_instance2;
```

ID	Name	School	Credits_Earned	Credits_Req	yo_grad
7	Haines	Khoury	32	120	2021
8	Lee	D'Amore McKim	64	128	2020
9	Frred	D'Amore McKim	50	120	2020

ID	Name	School	Credits_Earned	Credits_Req	yr_o_grad
1	Smith	Khoury	32	120	2019
2	Shah	D'Amore McKim	64	128	2019
3	Li	Khoury	50	120	2020

Yo_grad
2021
2020
2020
2019
2019
2020

JOIN OPERATION

JOIN functionality

- Many times, the data you want in your result is in multiple tables.
- The JOIN operation allows you to retrieve data from multiple tables and put semantic restrictions on the result tuples when needed.
- It is typically considered analogous to multiplication since if table A has m tuples and table B has n tuples, A JOIN B can have as many as m*n tuples.
- There are two different syntaxes for expressing a JOIN: the implicit join and the explicit join.
- The implicit join is the older method for expressing a join and is not used in industry.

Syntax for the Implicit JOIN

```
SELECT select_list
FROM table_1, table_2 [, table_3]...
WHERE table_1.column_name operator
  table_2.column_name
  [AND table_2.column_name operator
  table_3.column_name]...
```

We treat the table source as a comma separated list, just like the fields in the select list. In order to retrieve tuples that are about the same entity we create WHERE clauses that typically match primary key to foreign key.

Example: Implicit JOIN

We have the students table and the student_major table. We want a result that lists all student fields along with the student’s major in one result. We can use a JOIN to retrieve the data.

ID	Name	School	Credits_Earned	Credits_Req
1	Smith	Khoury	32	120
2	Shah	D’Amore McKim	64	128
3	Li	Khoury	50	120

ID	Major
1	CS
1	Accounting
2	CS
3	DS

Example Implicit JOIN

```
SELECT id, name, school, credits_earned, credit_req, major
  FROM student, student_major WHERE id = student_id;
```

id	name	school	credits_earned	credits_req
1	Smith	Khoury	32	120
2	Shah	D'Amore McKim	64	128
3	Li	Khoury	50	120

student_id	major
1	CS
1	Accounting
2	CS
3	DS

What is the result from the query?
How many columns?
How many rows?
What is the result without the WHERE clause?

Example Result: Implicit JOIN

```
SELECT id, name, school, credits_earned, credit_req, major
  FROM student, student_major WHERE id = student_id;
```

id	name	school	credits_earned	credits_req	major
1	Smith	Khoury	32	120	CS
1	Smith	Khoury	32	120	ACCOUNTING
2	Shah	D’Amore McKim	64	128	CS
3	Li	Khoury	50	120	DS

Example (2) : Implicit JOIN

```
SELECT id, name, school, credits_earned, credit_req, major
  FROM student_major, student;
```

Is this a legal (well-formed) query?

If so, what is the result?

Example (2) : FULL JOIN

Yes, this is a legal SQL statement. It is a FULL JOIN also known as a CROSS JOIN. However, the tuples do not semantically make sense.

A FULL JOIN is the cross product of the two relations. If one table has n tuples and the other has m tuples, then the result will have n x m tuples.

Example (2) : Result

Student_id	major	id	name	school	credits_earned	credits_req
1	CS	1	Smith	Khoury	32	120
1	CS	2	Shah	D’Amore McKim	64	128
1	CS	3	Li	Khoury	50	120
1	Accounting	1	Smith	Khoury	32	120
1	Accounting	2	Shah	D’Amore McKim	64	128
1	Accounting	3	Li	Khoury	50	120
2	CS	1	Smith	Khoury	32	120
2	CS	2	Shah	D’Amore McKim	64	128
2	CS	3	Li	Khoury	50	120
3	DS	1	Smith	Khoury	32	120
3	DS	2	Shah	D’Amore McKim	64	128
3	DS	3	Li	Khoury	50	120

Explicit INNER JOIN syntax

```
SELECT select_list
FROM table_1
    [INNER] JOIN table_2
        ON join_condition_1
   [[INNER] JOIN table_3
        ON join_condition_2]...
```

- Introduces the JOIN keyword, found between the table names involved in the JOIN and the optional [INNER] keyword.
- The ON clause specifies the JOIN condition. It is the qualifier that limits the JOINed tuples in the result.
- When the operation performed in the JOIN criteria is equality, it is called an EQUIJOIN.

Example Explicit INNER JOIN

```
SELECT id, name, school, credits_earned, credit_req, major
  FROM student INNER JOIN student_major
   ON id = student_id;
```

id	name	school	credits_earned	credits_req
1	Smith	Khoury	32	120
2	Shah	D'Amore McKim	64	128
3	Li	Khoury	50	120

student_id	major
1	CS
1	Accounting
2	CS
3	DS

Does this query produce the same result as the implicit JOIN result?

Example Result: Explicit INNER JOIN

```
SELECT id, name, school, credits_earned, credit_req, major
  FROM student INNER JOIN  student_major
             ON id = student_id;
```

id	name	school	credits_earned	credits_req	major
1	Smith	Khoury	32	120	CS
1	Smith	Khoury	32	120	ACCOUNTING
2	Shah	D'Amore McKim	64	128	CS
3	Li	Khoury	50	120	DS

If neither INNER or OUTER keywords are specified, the DEFAULT JOIN type is INNER.

A corresponding tuple is needed from both tables for a result tuple.

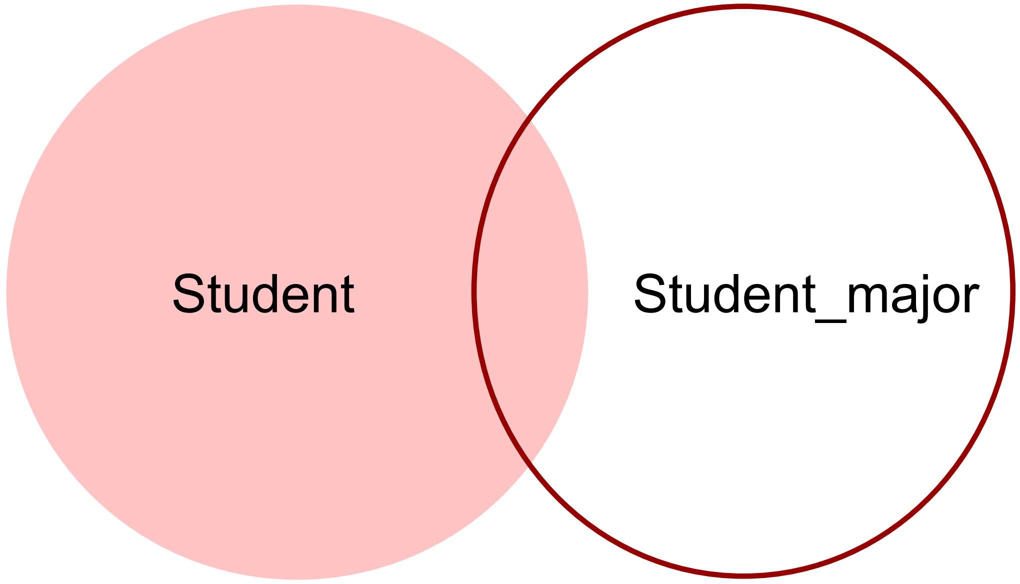
INNER versus OUTER JOIN

- An INNER JOIN produces a tuple in the result when there is a corresponding matching tuple in both the tables.
- An OUTER JOIN allows you to specify one of the tables as always contributing to the result. There are 2 types of OUTER JOINS: LEFT OUTER JOIN and RIGHT OUTER JOIN.
- LEFT OUTER JOIN specifies that the tuples in the table to the LEFT of the JOIN keyword always contributes to the result.
- RIGHT OUTER JOIN specifies that the tuples in the table to the RIGHT of the JOIN keyword always contributes to the result.

LEFT OUTER JOIN

LEFT outer join returns all rows in the left table and all the matching rows found in the right table.

Picture below shows a LEFT OUTER JOIN between student and student_major



Syntax for an OUTER JOIN

```
SELECT select_list
FROM table_1
    {LEFT|RIGHT} [OUTER] JOIN table_2
        ON join_condition_1
   [{LEFT|RIGHT} [OUTER] JOIN table_3
        ON join_condition_2]...
```

- Like the INNER keyword, the OUTER keyword is also optional. However, the LEFT and RIGHT keywords are not.
- LEFT JOIN retrieves unmatched tuples from the 1st table. RIGHT JOIN retrieves unmatched tuples from the 2nd table.
- We can have multiple JOIN clauses in the same SELECT statement.

Example: OUTER JOIN

```
SELECT * from student_major  AS sm
            RIGHT OUTER JOIN available_major AS m
            ON sm.major = m.major;
```

Student_major

ID	Major
1	CS
1	Accounting
2	CS
3	DS

Available_major

Major
CS
Accounting
DS
IS

ID	Sm.major	m.major
1	CS	CS
1	Accounting	Accounting
2	CS	CS
3	DS	DS
NULL	NULL	IS

Fields from the noncontributing table are set to NULL

Syntax for a CROSS (FULL) JOIN

```
SELECT select_list
  FROM table_1 CROSS JOIN table_2
```

EXAMPLE:

```
SELECT id, name, school, credits_earned,
  credit_req, major FROM student CROSS JOIN
  student_major;
```

- The ON clause is eliminated since we are not limiting the results from the cross product of the two tables.
- Just like the implicit full join, we have tuples created from 2 objects that are not related to each other.

Example (2) : Result

id	name	school	credits_earned	credits_req	Student_id	major
1	Smith	Khoury	32	120	1	CS
2	Shah	D’Amore McKim	64	128	1	CS
3	Li	Khoury	50	120	1	CS
1	Smith	Khoury	32	120	1	Accounting
2	Shah	D’Amore McKim	64	128	1	Accounting
3	Li	Khoury	50	120	1	Accounting
1	Smith	Khoury	32	120	2	CS
2	Shah	D’Amore McKim	64	128	2	CS
3	Li	Khoury	50	120	2	CS
1	Smith	Khoury	32	120	3	DS
2	Shah	D’Amore McKim	64	128	3	DS
3	Li	Khoury	50	120	3	DS

NATURAL JOIN

SYNTAX:

```
SELECT select_list
FROM table_1
       NATURAL JOIN table_2
      [NATURAL JOIN table_3]...
```

- A NATURAL JOIN allows the structure of the 2 tables to determine the JOIN criteria.
- It identifies all the field names in common between the two tables and performs a JOIN on those fields where all the common fields are equal. No ON clause is needed.
- The duplicated column is removed from the result.

Example: NATURAL JOIN

```
SELECT * FROM student AS s NATURAL JOIN
  student_major;
```

id	name	school	credits_earned	credits_req
1	Smith	Khoury	32	120
2	Shah	D'Amore McKim	64	128
3	Li	Khoury	50	120

id	major
1	CS
1	Accounting
2	CS
3	DS

Result: NATURAL JOIN

```
SELECT * FROM student AS s NATURAL JOIN
  student_major;
```

Notice the number of columns in the result

ID	Name	School	Credits_Earned	Credits_Req	MAJOR
1	Smith	Khoury	32	120	CS
1	Smith	Khoury	32	120	Accounting
2	Shah	D'Amore McKim	64	128	CS
3	Li	Khoury	50	120	DS

USING keyword

```
SELECT select_list
FROM table_1
    [{LEFT|RIGHT} [OUTER]] JOIN table_2
        USING (join_column_1[, join_column_2]...)
   [[{LEFT|RIGHT} [OUTER]] JOIN table_3
        USING (join_column_1[,
 join_column_2]...)]...
```

- If the criteria for the JOIN involves the same field name in the two tables, you can use the USING clause as opposed to the ON clause.
- It performs equality over the two same named columns from the tables.

Example: USING clause

```
SELECT * from student AS s INNER JOIN
  student_major USING (id);
```

id	name	school	credits_earned	credits_req
1	Smith	Khoury	32	120
2	Shah	D'Amore McKim	64	128
3	Li	Khoury	50	120

id	major
1	CS
1	Accounting
2	CS
3	DS

Result: USING clause

```
SELECT * from student s AS s
          JOIN student_major sm USING (id);
```

Notice the number of columns in the result

s.id	name	school	credits_earned	credits_req	sm.id	major
1	Smith	Khoury	32	120	1	CS
1	Smith	Khoury	32	120	1	Accounting
2	Shah	D'Amore McKim	64	128	2	CS
3	Li	Khoury	50	120	3	DS

Summary

In this module you learned:

- Implicit JOIN syntax
- Explicit JOIN syntax
- NATURAL JOIN operation
- INNER JOIN operation
- LEFT and RIGHT OUTER JOIN operation
- CROSS JOIN operation
- The USING clause

MySQL work

Let’s use the MySQL workbench to write SELECT statements using the ap database

Complete the join_exercises.sql exercises.