Joins and Nested Queries

Denormalized vs.Normalized Databases

Normalized databases are designed to minimize redundancy, while denormalized databases are designed to optimize read time.

In a traditional normalized database with data like Courses and Teachers, Courses might contain a column called TeacherID, which is a foreign key to Teacher. One benefit of this is that information about the teacher (name, address, etc.) is only stored once in the database.The drawback is that many common queries will require expensive joins.
Instead, we can denormalize the database by storing redundant data. For example, if we knew that we would have to repeat this query often, we might store the teacher's name in the Courses table. Denormalization is commonly used to create highly scalable systems.

SQL Syntax and Variations

As you read these queries, don't be surprised by minor variations in syntax. There are a variety of flavors of SQL, and you might have worked with a slightly different one. The examples in this book have been tested against Microsoft SQL Server.

Developers commonly use both the implicit join and the explicit join in SQLqueries.

Both syntaxesare shown below.

```
 /* Explicit Join */
SELECT CourseName, TeacherName
FROM Courses INNER JOIN Teachers
ON Courses.TeacherID = Teachers.TeacherID


 /* Implicit Join */
```

SELECT CourseName, TeacherName

FROM Courses, Teachers

WHERE Courses.TeacherID = Teachers.TeacherID

The two statements above are equivalent, and it's a matter of personal preference which one you choose. Forconsistency,we will stick to the explicit join.

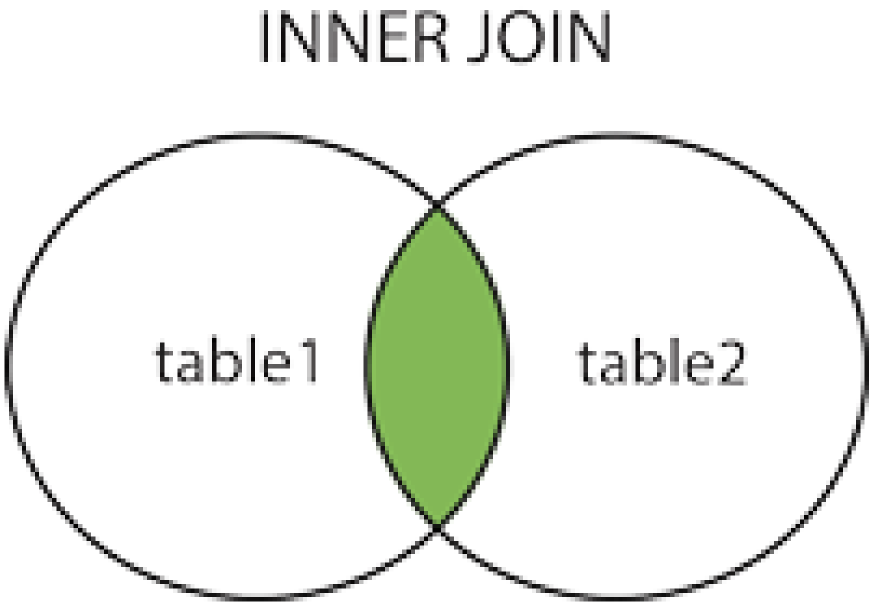
SQL JOIN

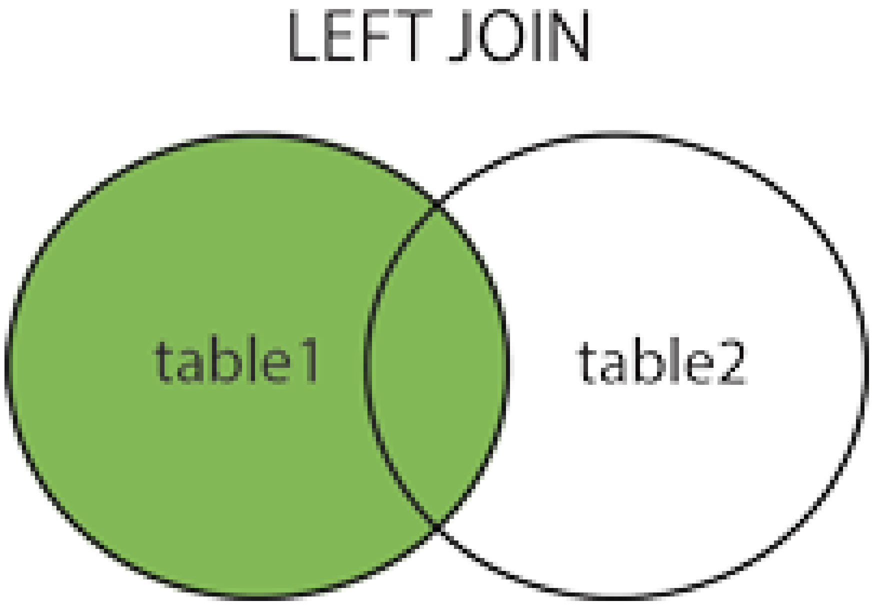
A `JOIN` clause is used to combine rows from two or more tables, based on a related column between them.

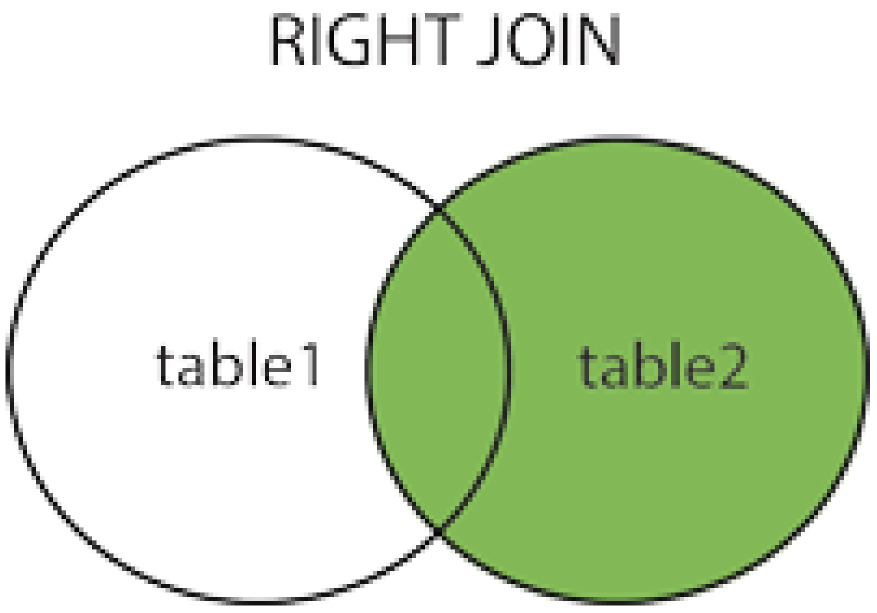
Different Types of SQL JOINs

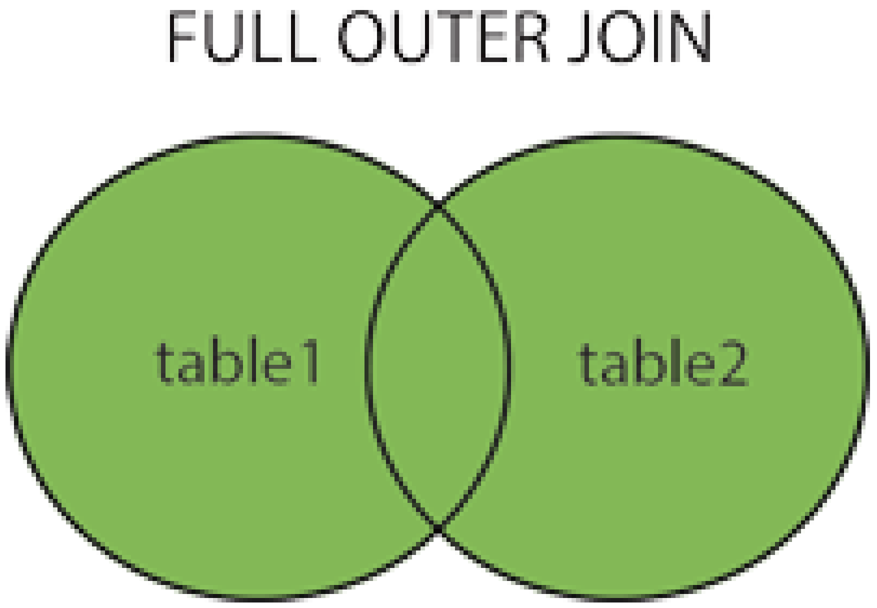
Here are the different types of the JOINs in SQL:

- `(INNER) JOIN`: Returns records that have matching values in both tables
- `LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table
- `RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from the left table
- `FULL (OUTER) JOIN`: Returns all records when there is a match in either left or right table

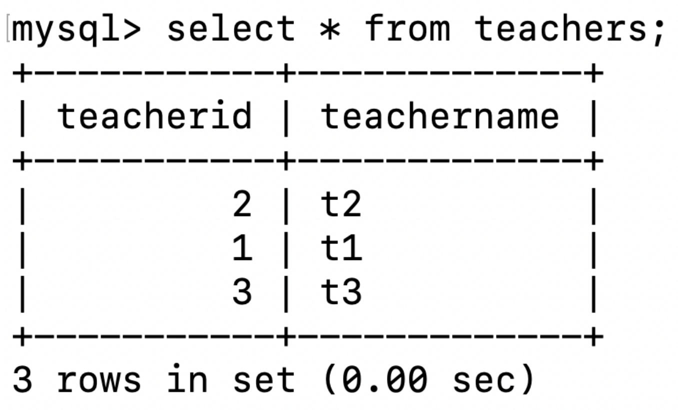

INNER JOIN
table1
table2


LEFT JOIN
table1
table2

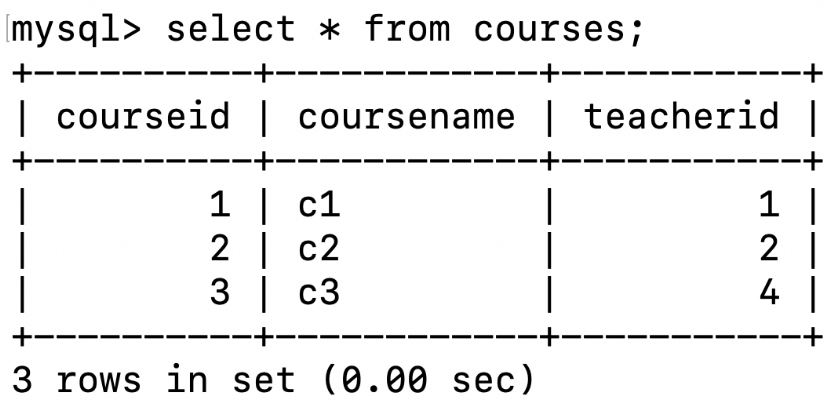




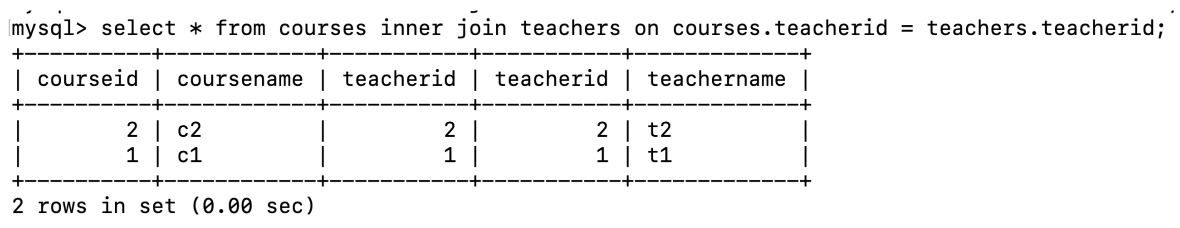
Examples:



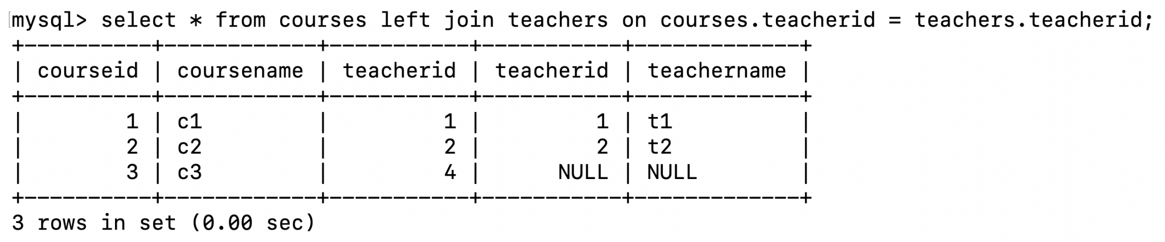
```
mysql> select * from teachers;
+-----------+-------------+
| teacherid | teachername |
+-----------+-------------+
|         2 | t2          |
|         1 | t1          |
|         3 | t3          |
+-----------+-------------+
3 rows in set (0.00 sec)
```



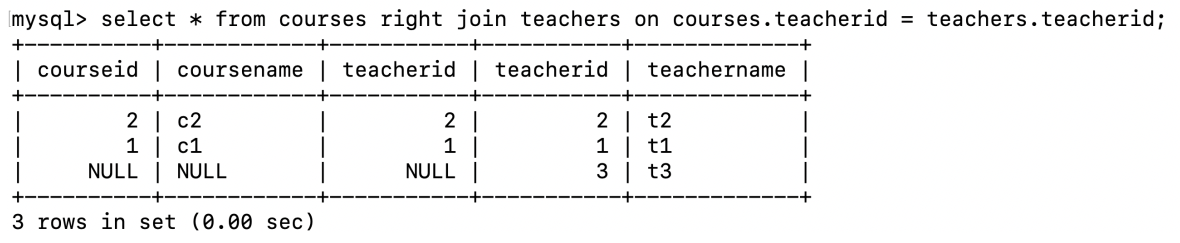
```
mysql> select * from courses;
+----------+------------+-----------+
| courseid | coursename | teacherid |
+----------+------------+-----------+
|        1 | c1         |         1 |
|        2 | c2         |         2 |
|        3 | c3         |         4 |
+----------+------------+-----------+
3 rows in set (0.00 sec)
```



```
mysql> select * from courses inner join teachers on courses.teacherid = teachers.teacherid;
+----------+------------+-----------+-----------+-------------+
| courseid | coursename | teacherid | teacherid | teachername |
+----------+------------+-----------+-----------+-------------+
|        2 | c2         |         2 |         2 | t2          |
|        1 | c1         |         1 |         1 | t1          |
+----------+------------+-----------+-----------+-------------+
2 rows in set (0.00 sec)
```



```
mysql> select * from courses left join teachers on courses.teacherid = teachers.teacherid;
+----------+------------+-----------+-----------+-------------+
| courseid | coursename | teacherid | teacherid | teachername |
+----------+------------+-----------+-----------+-------------+
|        1 | c1         |         1 |         1 | t1          |
|        2 | c2         |         2 |         2 | t2          |
|        3 | c3         |         4 |      NULL | NULL        |
+----------+------------+-----------+-----------+-------------+
3 rows in set (0.00 sec)
```

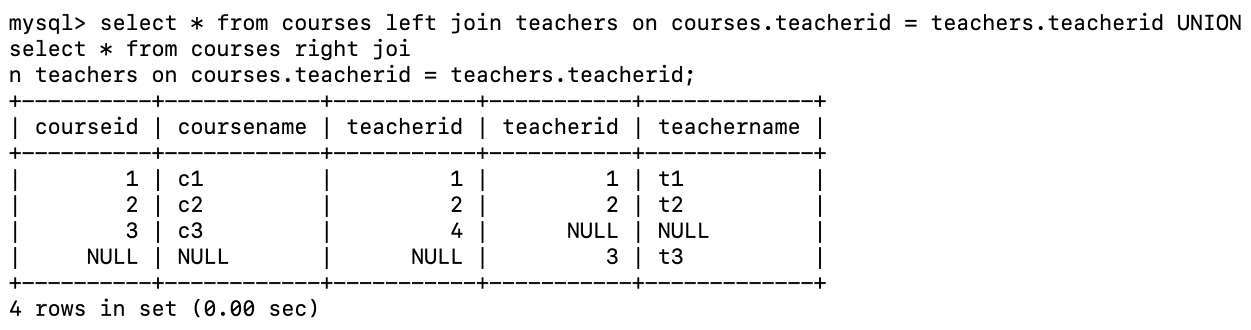


```
mysql> select * from courses right join teachers on courses.teacherid = teachers.teacherid;
+----------+------------+-----------+-----------+-------------+
| courseid | coursename | teacherid | teacherid | teachername |
+----------+------------+-----------+-----------+-------------+
|        2 | c2         |         2 |         2 | t2          |
|        1 | c1         |         1 |         1 | t1          |
|     NULL | NULL       |      NULL |         3 | t3          |
+----------+------------+-----------+-----------+-------------+
3 rows in set (0.00 sec)
```

Note: We don’t have full outer join support in MySQL.

With two tables t1, t2:

```
SELECT * FROM t1
LEFT JOIN t2 ON t1.id = t2.id
UNION
SELECT * FROM t1
RIGHT JOIN t2 ON t1.id = t2.id
```



```
mysql> select * from courses left join teachers on courses.teacherid = teachers.teacherid UNION
select * from courses right joi
n teachers on courses.teacherid = teachers.teacherid;
+----------+------------+-----------+-----------+-------------+
| courseid | coursename | teacherid | teacherid | teachername |
+----------+------------+-----------+-----------+-------------+
|        1 | c1         |         1 |         1 | t1          |
|        2 | c2         |         2 |         2 | t2          |
|        3 | c3         |         4 |      NULL | NULL        |
|     NULL | NULL       |      NULL |         3 | t3          |
+----------+------------+-----------+-----------+-------------+
4 rows in set (0.00 sec)
```

Introduction to the MySQL Subquery

A MySQL subquery is a query nested within another query such as `SELECT`, `INSERT`, `UPDATE` or `DELETE`. Also, a subquery can be nested within another subquery.

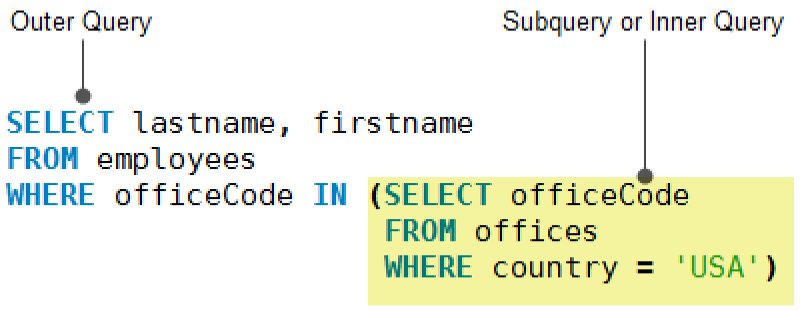
A MySQL subquery is called an inner query while the query that contains the subquery is called an outer query. A subquery can be used anywhere that expression is used and must be closed in parentheses.

For example, the following query uses a subquery to return the employees who work in the offices located in the USA.

```
SELECT
    lastName, firstName
FROM
    employees
WHERE
    officeCode IN (SELECT
            officeCode
        FROM
            offices
        WHERE
            country = 'USA');
Code language: SQL (Structured Query Language) (sql)
```

In this example:

- The subquery returns all *office codes* of the offices located in the USA.
- The outer query selects the last name and first name of employees who work in the offices whose office codes are in the result set returned by the subquery.



```
SELECT lastname, firstname
FROM employees
WHERE officeCode IN (SELECT officeCode
                     FROM offices
                     WHERE country = 'USA')
```

When executing the query, MySQL evaluates the subquery first and uses the result of the subquery for the outer query.

```
mysql> select * from offices;
+------------+----------+
| officecode | country  |
+------------+----------+
|          3 | India    |
|          4 | Thailand |
|          2 | USA      |
+------------+----------+
3 rows in set (0.00 sec)

mysql> select * from employees;
+-----------+----------+------------+
| firstname | lastname | officecode |
+-----------+----------+------------+
| f1        | l1       |          1 |
| f3        | l3       |          3 |
| f2        | l2       |          2 |
+-----------+----------+------------+
3 rows in set (0.00 sec)

mysql> SELECT
    ->     lastName, firstName
    -> FROM
    ->     employees
    -> WHERE
    ->     officeCode IN (SELECT
    ->             officeCode
    ->         FROM
    ->             offices
    ->         WHERE
    ->             country = 'USA');
+----------+-----------+
| lastName | firstName |
+----------+-----------+
| l2       | f2        |
+----------+-----------+
1 row in set (0.00 sec)
```

References:

https://stackoverflow.com/questions/4796872/how-can-i-do-a-full-outer-join-in-mysql

https://www.w3schools.com/sql/sql_join_left.asp

book - Cracking the Coding Interview.

https://www.mysqltutorial.org/mysql-subquery/