Birla Institute of Technology & Science, Pilani, K. K. BIRLA Goa campus
Database Systems (CS F212) Second Semester 2019-2020
Lab-5: To study JOINS and SET Operations in SQL

In the previous labs we were getting data from one table at a time. This is good enough for simple tasks but in real world MYSQL usages, you will often need to get data from multiple mutually related tables in a single query.

A join enables you to retrieve records from two (or more) logically related tables in a single result set.
JOIN clauses are used to return the rows of two or more queries using two or more tables that share a meaningful relationship based on a common set of values. These values are usually the same column name and data type that appear in both the participating tables being joined.

These columns, or possibly a single column from each table, are called the join key or common key. Mostly *but not all of the time*, the join key is the primary key of one table and a foreign key in another table. The join can be performed as long as the data in the columns are *matching*.
It can be difficult when the join involves more than two tables. It is good practice to think of the query as a series of two table joins, when involvement of three or more tables in joins.

TYPES OF JOINS

- CROSS JOIN
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- NATURAL JOIN

We will work with the Library database from Lab 4 in this manual.

CROSS JOIN

Cross Join simply returns the cartesian product of the rows from the joined tables. Unlike other joins CROSS JOIN clause doesn't have any condition and hence it is not used widely.

Suppose the first table has n rows and the second table has m rows. The cross join that joins the first with the second table will return nxm rows.

The following show the syntax of the cross join clause:

```
SELECT columns
FROM table_1
CROSS JOIN table_2;
```

Suppose, we want to get the list of all customers records against all OrderBook records, we can use the following query:

```
mysql> Select * from Customer CROSS JOIN OrderBook;
```

```
mysql> Select * from Customer CROSS JOIN OrderBook;
+-----+--------+------------------------------+------+-------+------+------+------------+
| cid | cname  | address                      | age  | oisbn | ocid | qty  | orderdate  |
+-----+--------+------------------------------+------+-------+------+------+------------+
| c5  | John   | Pune, Shivajinagar           |   18 | A1234 | c1   |    1 | 2012-07-02 |
| c4  | Saloni | Hyderabad                    |   22 | A1234 | c1   |    1 | 2012-07-02 |
| c3  | Pooja  | sector no. 23, Vashi, Mumbai |   24 | A1234 | c1   |    1 | 2012-07-02 |
| c2  | Akbar  | D-20, Sainivas, Mumbai       |   19 | A1234 | c1   |    1 | 2012-07-02 |
| c1  | Amar   |  23, M.G. road, Ahmadabad    |   20 | A1234 | c1   |    1 | 2012-07-02 |
| c5  | John   | Pune, Shivajinagar           |   18 | A1234 | c2   |    2 | 2013-10-01 |
| c4  | Saloni | Hyderabad                    |   22 | A1234 | c2   |    2 | 2013-10-01 |
| c3  | Pooja  | sector no. 23, Vashi, Mumbai |   24 | A1234 | c2   |    2 | 2013-10-01 |
| c2  | Akbar  | D-20, Sainivas, Mumbai       |   19 | A1234 | c2   |    2 | 2013-10-01 |
| c1  | Amar   |  23, M.G. road, Ahmadabad    |   20 | A1234 | c2   |    2 | 2013-10-01 |
| c5  | John   | Pune, Shivajinagar           |   18 | A1236 | c1   |    5 | 2012-05-14 |
| c4  | Saloni | Hyderabad                    |   22 | A1236 | c1   |    5 | 2012-05-14 |
| c3  | Pooja  | sector no. 23, Vashi, Mumbai |   24 | A1236 | c1   |    5 | 2012-05-14 |
| c2  | Akbar  | D-20, Sainivas, Mumbai       |   19 | A1236 | c1   |    5 | 2012-05-14 |
| c1  | Amar   |  23, M.G. road, Ahmadabad    |   20 | A1236 | c1   |    5 | 2012-05-14 |
| c5  | John   | Pune, Shivajinagar           |   18 | A1236 | c5   |    4 | 2012-12-30 |
| c4  | Saloni | Hyderabad                    |   22 | A1236 | c5   |    4 | 2012-12-30 |
| c3  | Pooja  | sector no. 23, Vashi, Mumbai |   24 | A1236 | c5   |    4 | 2012-12-30 |
| c2  | Akbar  | D-20, Sainivas, Mumbai       |   19 | A1236 | c5   |    4 | 2012-12-30 |
| c1  | Amar   |  23, M.G. road, Ahmadabad    |   20 | A1236 | c5   |    4 | 2012-12-30 |
| c5  | John   | Pune, Shivajinagar           |   18 | A1236 | c3   |    2 | 2013-12-12 |
| c4  | Saloni | Hyderabad                    |   22 | A1236 | c3   |    2 | 2013-12-12 |
| c3  | Pooja  | sector no. 23, Vashi, Mumbai |   24 | A1236 | c3   |    2 | 2013-12-12 |
| c2  | Akbar  | D-20, Sainivas, Mumbai       |   19 | A1236 | c3   |    2 | 2013-12-12 |
| c1  | Amar   |  23, M.G. road, Ahmadabad    |   20 | A1236 | c3   |    2 | 2013-12-12 |
| c5  | John   | Pune, Shivajinagar           |   18 | A1238 | c4   |   10 | 2012-06-15 |
| c4  | Saloni | Hyderabad                    |   22 | A1238 | c4   |   10 | 2012-06-15 |
| c3  | Pooja  | sector no. 23, Vashi, Mumbai |   24 | A1238 | c4   |   10 | 2012-06-15 |
| c2  | Akbar  | D-20, Sainivas, Mumbai       |   19 | A1238 | c4   |   10 | 2012-06-15 |
| c1  | Amar   |  23, M.G. road, Ahmadabad    |   20 | A1238 | c4   |   10 | 2012-06-15 |
+-----+--------+------------------------------+------+-------+------+------+------------+
```

INNER JOIN

Inner Join returns rows from either table if and only if both tables meet the conditions specified. The inner join clause compares each row from the first table with every row from the second table. If values in both rows cause the join condition evaluates to true, the inner join clause creates a new row whose column contains all columns of the two rows from both tables and include this new row in the final result set. Inner Join is the most commonly used join and chances are that you might have used it already (*using WHERE t1.column=t2.column*) .
The following show the syntax of the cross join clause:

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

Suppose, we want to get the list of customers who have ordered a book along with the oisbn of book and customer names. This can be achieved using INNER JOIN by the following query:

```
mysql> SELECT C.cid, C.cname, OB.oisbn FROM Customer AS C INNER JOIN
OrderBook AS OB ON C.cid=OB.ocid;
+-----+--------+-------+
| cid | cname  | oisbn |
+-----+--------+-------+
| c1  | Amar   | A1234 |
| c1  | Amar   | A1236 |
| c2  | Akbar  | A1234 |
| c3  | Pooja  | A1236 |
| c4  | Saloni | A1238 |
| c5  | John   | A1236 |
+-----+--------+-------+
6 rows in set (0.00 sec)
```

LEFT JOIN

The above query gave us isbn of all the books which have been ordered by some customer. But what if we wanted to list all the books irrespective of whether some customer ordered it (with NULL in qty if no one ordered it)?

Left Join is used for this purpose. The LEFT JOIN is such a join which specifies that all records be fetched from the table on the left side of the join statement. If a record returned from the left table has no matching record in the table on the right side of the join, it is still returned, and the corresponding column from the right table returns a NULL value.

The following show the syntax of the cross join clause:

```
SELECT columns
FROM table1
LEFT JOIN table2 ON table1.column = table2.column;
```

Here is the result of the above query using Left Join:

```
mysql> SELECT B.isbn, B.title, OB.qty FROM Book AS B LEFT JOIN
OrderBook as OB ON B.isbn=OB.oisbn;
+-------+---------------------------------+------+
| isbn  | title                           | qty  |
+-------+---------------------------------+------+
| A1234 | Data Structures and Algorithms |    1 |
| A1234 | Data Structures and Algorithms |    2 |
| A1236 | Operating Systems              |    5 |
| A1236 | Operating Systems              |    4 |
| A1236 | Operating Systems              |    2 |
| A1238 | Applied Mathematics            |   10 |
| A1235 | Computer Networks              | NULL |
| A1237 | C                               | NULL |
+-------+---------------------------------+------+
8 rows in set (0.00 sec)
```

- Run the same query by replacing LEFT JOIN with INNER JOIN and compare the results.

RIGHT JOIN

The right join clause is similar to the left join clause except that the treatment of tables is reversed.

The right join starts selecting data from the right table instead of the left table.The right join clause selects all rows from the right table and matches rows in the left table. If a row from the right table does not have matching rows from the left table, the column of the left table will have NULL in the final result set.

The following show the syntax of the cross join clause:

```
SELECT columns
FROM table1
RIGHT JOIN table2 ON table1.column = table2.column;
```

The same result can be obtained by Right Join instead of Left Join by just interchanging the tables in the clause.

```
mysql> SELECT B.isbn, B.title, OB.qty FROM OrderBook AS OB RIGHT JOIN
Book AS B ON B.isbn=OB.oisbn;
+-------+---------------------------------+------+
| isbn  | title                           | qty  |
+-------+---------------------------------+------+
| A1234 | Data Structures and Algorithms |    1 |
| A1234 | Data Structures and Algorithms |    2 |
| A1236 | Operating Systems              |    5 |
| A1236 | Operating Systems              |    4 |
| A1236 | Operating Systems              |    2 |
| A1238 | Applied Mathematics            |   10 |
| A1235 | Computer Networks              | NULL |
| A1237 | C                               | NULL |
+-------+---------------------------------+------+
8 rows in set (0.00 sec)
```

To find the rows which are not in OrderBook table, you can use a WHERE clause with the IS NULL operator:

```
SELECT columns
FROM table1
RIGHT JOIN table2 ON table1.column = table2.column
WHERE column_table_1 IS NULL;
```

- Modify the above query to show isbn and title of only those books which were not ordered by anyone.

NATURAL JOIN

A NATURAL JOIN is such a join that performs the same task as an INNER or LEFT JOIN, in which the ON or USING clause refers to all columns that the tables to be joined have in common.

To use Natural Join:

➔ The associated tables have one or more pairs of identically named columns.

➔ The columns must be the same data type.

➔ Don't use ON clause in a natural join.

To show the example of Natural Join, let us create two new tables.

```
 CREATE TABLE students (
      student_id INT AUTO_INCREMENT,
      s_name VARCHAR(100),
      PRIMARY KEY (student_id)
);

CREATE TABLE courses (
      student_id INT AUTO_INCREMENT,
      c_name VARCHAR(100),
      PRIMARY KEY (student_id)
);

INSERT INTO students(s_name)
VALUES('Sagar'),('Rohit'),('Raghav'),('Ameya'),('Aditya');

INSERT INTO courses(c_name)
VALUES('OS'),('DSA'),('DBMS'),('OOP');
```

```
mysql> select * from students;
+------------+--------+
| student_id | s_name |
+------------+--------+
|          1 | Sagar  |
|          2 | Rohit  |
|          3 | Raghav |
|          4 | Ameya  |
|          5 | Aditya |
+------------+--------+
5 rows in set (0.00 sec)
```

```
mysql> select * from courses;
+------------+--------+
| student_id | c_name |
+------------+--------+
|          1 | OS     |
|          2 | DSA    |
|          3 | DBMS   |
|          4 | OOP    |
+------------+--------+
4 rows in set (0.00 se
```

To get all the unique columns from students and courses tables, the following query can be used:

```
mysql> select * from course NATURAL JOIN students;
+------------+--------+--------+
| student_id | c_name | s_name |
+------------+--------+--------+
|          1 | OS     | Sagar  |
|          2 | DSA    | Rohit  |
|          3 | DBMS   | Raghav |
|          4 | OOP    | Ameya  |
+------------+--------+--------+
4 rows in set (0.00 sec)
```

- The MySQL NATURAL JOIN is structured in such a way that columns with the same name of associate tables will appear once only. This is one difference between Natural Join and Inner Join. Let us try the same query using Inner Join and analyze the result.

```
mysql> select * from course INNER JOIN students on
course.student_id=students.student_id;
+------------+--------+------------+--------+
| student_id | c_name | student_id | s_name |
+------------+--------+------------+--------+
|          1 | OS     |          1 | Sagar  |
|          2 | DSA    |          2 | Rohit  |
|          3 | DBMS   |          3 | Raghav |
|          4 | OOP    |          4 | Ameya  |
+------------+--------+------------+--------+
4 rows in set (0.01 sec)
```

“ON” and “USING” Clauses

We can also use the “USING” clause instead of “ON” in the join queries. The difference with **USING is it needs to have identical names for matched columns in both tables.**

Example:

```
mysql> select * from course RIGHT JOIN students USING(student_id);
+------------+--------+--------+
| student_id | s_name | c_name |
+------------+--------+--------+
|          1 | Sagar  | OS     |
|          2 | Rohit  | DSA    |
|          3 | Raghav | DBMS   |
|          4 | Ameya  | OOP    |
|          5 | Aditya | NULL   |
+------------+--------+--------+
5 rows in set (0.00 sec)
```

- Try running the above INNER JOIN query with “USING” instead of “ON” clause and justify the difference.

INNER JOIN with Three Tables:

Let us go back to our library database. It is possible that we might need to show a column from each of the three tables as a result to our query. For example, let us show the name of the customer, the title of the book and the date at which he ordered a book. We already know the three tables are related to each other by primary and foreign keys. Therefore, we can join all the three tables to get the required result.

The same logic is applied which is done to join two tables i.e. minimum number of join statements to join n tables are (n-1).
Following is the query and result:

```
mysql> SELECT cname, title, orderdate FROM Customer C INNER JOIN
OrderBook OB ON C.cid=OB.ocid INNER JOIN Book AS B ON
B.isbn=OB.oisbn;
+--------+---------------------------------+------------+
| cname  | title                           | orderdate  |
+--------+---------------------------------+------------+
| Amar   | Data Structures and Algorithms | 2012-07-02 |
| Amar   | Operating Systems               | 2012-05-14 |
| Akbar  | Data Structures and Algorithms | 2013-10-01 |
| Pooja  | Operating Systems               | 2013-12-12 |
| Saloni | Applied Mathematics             | 2012-06-15 |
| John   | Operating Systems               | 2012-12-30 |
+--------+---------------------------------+------------+
6 rows in set (0.00 sec)
```

SET Operations

SQL Set Operations is used to combine data from two or more SQL Select Statements.

UNION

This set operator is used to combine the outputs of two or more queries into a single set of rows and columns having different records.

Syntax:

```
SELECT column_name FROM table1
UNION
SELECT column_name FROM table2;
```

- In the union operation, all the number of datatype and columns must be same in both the tables on which UNION operation is being applied.
- The union operation eliminates the duplicate rows from its resultset.

Suppose, we want to find the cids of customers who ordered a book before ‘2013-10-01’ or whose age is > 19. We can use UNION to combine the two results.

```
mysql>
SELECT ocid as cid FROM OrderBook WHERE orderdate<'2013-10-01'
UNION
SELECT cid from Customer WHERE age>19;
+-----+
| cid |
+-----+
| c1  |
| c4  |
| c5  |
| c3  |
+-----+
4 rows in set (0.22 sec)
```

UNION ALL

Union All operation is equal to the Union operation. It returns the set without removing duplication and sorting the data.

Syntax:

```
SELECT column_name FROM table1
UNION ALL
SELECT column_name FROM table2;
```

Result of above query using UNION ALL instead of UNION:

```
mysql> SELECT ocid as cid FROM OrderBook WHERE orderdate<'2013-10-01'
UNION ALL SELECT cid from Customer WHERE age>19;
+-----+
| cid |
+-----+
| c1  |
| c1  |
| c4  |
| c5  |
| c1  |
| c3  |
| c4  |
+-----+
7 rows in set (0.00 sec)
```

INTERSECT

The Intersect operation returns the common rows from both the SELECT statements. In the Intersect operation, the number of datatype and columns must be the same.

```
SELECT column_name FROM table1
INTERSECT
SELECT column_name FROM table2;
```

NOTE: MYSQL doesn't support INTERSECT Operation.

However, as the astute students may have noticed, the same query can be emulated using MYSQL JOIN clauses without using INTERSECT keyword.

```
SELECT DISTINCT
   column_name
FROM table1
   INNER JOIN table2 USING(id);
```

Suppose, we wanted to use INTERSECT instead of UNION in the last example. Here is the query for the same:

```
mysql> SELECT DISTINCT cid FROM OrderBook INNER JOIN Customer ON
OrderBook.ocid=Customer.cid WHERE orderdate<'2013-10-01' AND age>19;
+-----+
| cid |
+-----+
| c1  |
| c4  |
+-----+
2 rows in set (0.00 sec)
```

Some of the alert students might have also noticed that the same result can be obtained by using 'IN' and SQL Subqueries. You can try working it out on the above example.

MINUS

MINUS compares the results of two queries and returns distinct rows from the result set of the first query that does not appear in the result set of the second query.

```
SELECT column_name
FROM table_name1
MINUS
SELECT column_name
FROM table_name2;
```

Just like INTERSECT, MINUS is also not supported by MYSQL.

But by now we know that it can be emulated using JOINS.

```
SELECT
    column_name
FROM
    table_name1
LEFT JOIN
    table_name2 USING (id)
WHERE
    table_name2.column_name IS NULL;
```

Example:

```
mysql> SELECT cid FROM (SELECT ocid as cid FROM OrderBook WHERE
orderdate<'2013-10-01') AS T1 LEFT JOIN (SELECT cid FROM Customer
WHERE age>19) AS T2 USING(cid) WHERE T2.cid IS NULL;
+-----+
| cid |
+-----+
| c5  |
+-----+
1 row in set (0.00 sec)
```