Using Joins

Session 10

Objectives

- *Explain the different types of table joins in MySQL*
- *Explain the use of Equi-Join*
- *Explain the use of Inner Join*
- *Explain the use of Outer-Join*
- *Explain the use of Self-Join*
- *Explain the use of multiple SELECT queries in a single SELECT query*
- *Explain the use of UNION with the query*

Joining Tables in MySQL

- Combining two or more records of different tables from the same database, into one comprehensive structure, is known as joining of tables
- Joining tables enables ease of manipulation, increases the speed of access, and reduces data redundancy
- The concept of joining tables is to display a single table that will be derived from the base of all related tables linked to each other with a common attribute
- You can join the tables either using the `WHERE` clause with the `SELECT` command or by using the `JOIN` keyword

Using Equi-Join

- In Equi-Join, MySQL combines records from two or more tables based on a common column
- The syntax for Equi-Join is:

```
SELECT column_name,...[*] FROM table1 [as alias],table2 [as
alias] WHERE (join_condition);
```

where,

`SELECT` – retrieves data from the specified column and the table

`column_name` – specifies the name of the column to retrieve from the table

`table1` – specifies the name of the table that contains the columns

`as alias` – specifies the reference name for the table

`WHERE` – specifies the conditions that need to be satisfied before retrieving data

`join_condition` – compares data from the tables as specified in this clause

Using Equi-Join

- You can specify the `join_condition` using any one of the following clauses:
  - ON clause:
    - Specifies the conditional expression that can be used in a `WHERE` clause
    - If there is no matching record present in the specified table according to the query, then the row with all columns set to `NULL` will be used
  - USING clause:
    - Specifies the common attribute of the joining tables

Using Equi-Join

- For example, to display the first name, department name, and designation of all employees from the EMP_DETAILS and the EMP_DEPARTMENT tables of EMPLOYEE database enter the following command at the command prompt:

```
SELECT EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.D_NAME,
EMP_DEPARTMENT.DESIGNATION FROM EMP_DETAILS,
EMP_DEPARTMENT WHERE (EMP_DETAILS.E_ID =
EMP_DEPARTMENT.E_ID);
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT EMP_DETAILS.E_FNAME,
    -> EMP_DEPARTMENT.D_NAME,
    -> EMP_DEPARTMENT.DESIGNATION
    -> FROM EMP_DETAILS, EMP_DEPARTMENT
    -> WHERE
    -> (EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID);
+---------+-------------+--------------+
| E_FNAME | D_NAME      | DESIGNATION  |
+---------+-------------+--------------+
| JACK    | RESEARCH    | MANAGER      |
| PETER   | MARKETING   | ASST.MANAGER |
| JENNIE  | DEVELOPMENT | MANAGER      |
| SAMUEL  | SALES       | MANAGER      |
| MARY    | SALES       | MANAGER      |
+---------+-------------+--------------+
5 rows in set (0.01 sec)

mysql>
```

Using INNER JOIN

- The most common join operation used in applications is an `INNER JOIN`
- An `INNER JOIN` creates a virtual table by combining the fields of both tables that satisfy the query for both the tables
- An `INNER JOIN` creates a virtual table by combining the fields of both tables that satisfy the query for both the tables
- The only difference is in the syntax

Using INNER JOIN

- The syntax for inner join is:

```
SELECT column_name1, column_name2, column_name3FROM table_name1
INNER JOIN table_name2 ON table_name1.primary_keyfield =
table_name2.foreign_keyfield;
```

where,

`SELECT` – retrieves data from the table

`column_name` – specifies the name of the column

`table_name` – specifies the name of the table that contain data

`INNER JOIN` – creates a virtual table by combining the fields from both tables that satisfy the query

`table_name1.primary_keyfield= table_name2.foreign_keyfield` – specifies the relationship between the tables

Using INNER JOIN

- For example, to display the first names, departments, and designations of all employees from `EMP_DETAILS` and `EMP_DEPARTMENT` tables from EMPLOYEE database, enter the following command at the command prompt:

```
SELECT EMP_DETAILS.E_FNAME,EMP_DEPARTMENT.D_NAME,
EMP_DEPARTMENT.DESIGNATION FROM EMP_DETAILS INNER JOIN
EMP_DEPARTMENT ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT EMP_DETAILS.E_FNAME,
    -> EMP_DEPARTMENT.D_NAME,
    -> EMP_DEPARTMENT.DESIGNATION
    -> FROM
    -> EMP_DETAILS INNER JOIN EMP_DEPARTMENT
    -> ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
+---------+-------------+--------------+
| E_FNAME | D_NAME      | DESIGNATION  |
+---------+-------------+--------------+
| JACK    | RESEARCH    | MANAGER      |
| PETER   | MARKETING   | ASST.MANAGER |
| JENNIE  | DEVELOPMENT | MANAGER      |
| SAMUEL  | SALES       | MANAGER      |
| MARY    | SALES       | MANAGER      |
+---------+-------------+--------------+
5 rows in set (0.00 sec)

mysql>
```

Using INNER JOIN

- You can combine records from three or more tables and display the output in a single result set
- For example, to view all employees who have a basic salary greater than $2,000, you will retrieve data from three tables (EMP_DETAILS, EMP_DEPARTMENT, and EMP_SALARY) and combine them into a single result set
- To join data from three tables, enter the following command at the command prompt:

```
SELECT EMP_DETAILS.E_FNAME,EMP_DEPARTMENT.D_NAME,
 EMP_SALARY.BASIC_SALARY FROM EMP_DETAILS INNER JOIN
 (EMP_DEPARTMENT INNER JOIN EMP_SALARY ON
 (EMP_DEPARTMENT.E_ID = EMP_SALARY.E_ID) AND
 (EMP_SALARY.BASIC_SALARY > 2000)) ON EMP_DETAILS.E_ID =
 EMP_DEPARTMENT.E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help

mysql> SELECT EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.D_NAME,
    -> EMP_SALARY.BASIC_SALARY
    -> FROM EMP_DETAILS
    -> INNER JOIN
    -> (EMP_DEPARTMENT INNER JOIN EMP_SALARY ON
    -> (EMP_DEPARTMENT.E_ID = EMP_SALARY.E_ID)
    -> AND
    -> (EMP_SALARY.BASIC_SALARY > 2000))
    -> ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
+---------+-------------+--------------+
| E_FNAME | D_NAME      | BASIC_SALARY |
+---------+-------------+--------------+
| PETER   | MARKETING   |      2500.00 |
| JENNIE  | DEVELOPMENT |      3000.00 |
| SAMUEL  | SALES       |      3500.00 |
| MARY    | SALES       |      4000.00 |
+---------+-------------+--------------+
4 rows in set (0.00 sec)

mysql>
```

Using INNER JOIN

- To view employee details using table aliases, enter the following command at the command prompt:

```
SELECT E.E_FNAME, E.E_LNAME, S.BASIC_SALARY FROM
EMP_DETAILS AS E, EMP_SALARY AS S WHERE E.E_ID =
S.E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help

mysql> SELECT E.E_FNAME, E.E_LNAME, S.BASIC_SALARY
    -> FROM EMP_DETAILS AS E,
    -> EMP_SALARY AS S
    -> WHERE E.E_ID = S.E_ID;
+---------+----------+--------------+
| E_FNAME | E_LNAME  | BASIC_SALARY |
+---------+----------+--------------+
| JACK    | WILLIAMS |      2000.00 |
| PETER   | ADAMS    |      2500.00 |
| JENNIE  | NORTON   |      3000.00 |
| SAMUEL  | BUSH     |      3500.00 |
| MARY    | GRAF     |      4000.00 |
+---------+----------+--------------+
5 rows in set (0.00 sec)

mysql>
```

Using INNER JOIN

- You can join numeric fields together, as long as they are of same data type such as `AutoNumber` or `Long`
- In case of non-numeric data, the fields must be of same type and length and should contain same kind of data
- In the `ON` clause, you can use relational operators such as `=`, `<, >, <=, >=, or <>`
- The relational operator, =, checks for equality of `E_ID` of both the tables

Using INNER JOIN

- To display the designations of those employees whose E_ID of EMP_DETAILS table does not match with the E_ID of EMP_DEPARTMENT table, enter the following command at the command prompt:

```
SELECT DISTINCT EMP_DETAILS.E_ID,EMP_DETAILS.E_FNAME,
EMP_DEPARTMENT.DESIGNATION FROM EMP_DETAILS INNER JOIN
EMP_DEPARTMENT ON EMP_DETAILS.E_ID <> EMP_DEPARTMENT.E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT DISTINCT
    -> EMP_DETAILS.E_ID,
    -> EMP_DETAILS.E_FNAME,
    -> EMP_DEPARTMENT.DESIGNATION
    -> FROM EMP_DETAILS
    -> INNER JOIN
    -> EMP_DEPARTMENT ON
    -> EMP_DETAILS.E_ID <> EMP_DEPARTMENT.E_ID;
+------+---------+--------------+
| E_ID | E_FNAME | DESIGNATION  |
+------+---------+--------------+
|  101 | JACK    | ASST.MANAGER |
|  101 | JACK    | MANAGER      |
|  102 | PETER   | MANAGER      |
|  103 | JENNIE  | MANAGER      |
|  103 | JENNIE  | ASST.MANAGER |
|  104 | SAMUEL  | MANAGER      |
|  104 | SAMUEL  | ASST.MANAGER |
|  105 | MARY    | MANAGER      |
|  105 | MARY    | ASST.MANAGER |
|  106 | BOB     | MANAGER      |
|  106 | BOB     | ASST.MANAGER |
|  107 | GEORGE  | MANAGER      |
|  107 | GEORGE  | ASST.MANAGER |
+------+---------+--------------+
13 rows in set (0.00 sec)
```

Using INNER JOIN

- You can also use the AND or the OR operator with INNER JOIN command
- For example, to view all employees whose E_ID values are same and who live in California city, enter the following command at the command prompt:

```
SELECT EMP_DETAILS.E_FNAME,
EMP_DEPARTMENT.D_NAME,EMP_DETAILS.E_ADDRESS FROM
EMP_DETAILS INNER JOIN EMP_DEPARTMENT ON (EMP_DETAILS.E_ID
= EMP_DEPARTMENT.E_ID) AND (EMP_DETAILS.E_ADDRESS =
'CALIFORNIA ');
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help

mysql> SELECT EMP_DETAILS.E_FNAME, EMP_DEPARTMENT.D_NAME,
    -> EMP_DETAILS.E_ADDRESS FROM EMP_DETAILS
    -> INNER JOIN
    -> EMP_DEPARTMENT ON
    -> (EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID)
    -> AND (EMP_DETAILS.E_ADDRESS = 'CALIFORNIA');
+---------+----------+------------+
| E_FNAME | D_NAME   | E_ADDRESS  |
+---------+----------+------------+
| JACK    | RESEARCH | CALIFORNIA |
+---------+----------+------------+
1 row in set (0.00 sec)

mysql>
```

Using INNER JOIN

- MySQL also supports multi-table delete statements using the `DELETE` command
- The syntax to delete matching rows from two tables is:

```
DELETE [LOW PRIORITY|QUICK]
[IGNORE]tbl_name[.*][,tbl_name[.*]...] FROM
table_references [WHERE where_definition];
```

where,

`DELETE` – removes data from the table

`tbl_name[.*]` – specifies the names of the table that contain data to delete

`table_references` – specifies the tables that are linked with a key

`WHERE` – defines the condition to satisfy before deleting records from the table

Using INNER JOIN

- This command deletes matching rows from the tables specified in the FROM clause and before the USING clause
- For example, to delete records from the EMP_DEPARTMENT table that have matching values with EMP_DETAILS table, enter the following command at the command prompt:

```
DELETE FROM EMP_DEPARTMENT
USING EMP_DETAILS
INNER JOIN EMP_DEPARTMENT
ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help

mysql> DELETE FROM EMP_DEPARTMENT
    -> USING EMP_DETAILS
    -> INNER JOIN EMP_DEPARTMENT
    -> ON EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
Query OK, 5 rows affected (0.01 sec)

mysql>
```

Using OUTER JOIN

- An `OUTER JOIN` displays rows of the tables that may not have any matching value in the other tables to appear in the result set
- To use an `OUTER JOIN`, the tables must have one or more columns in common
- The two types of **OUTER JOIN** are as follows:
  - **LEFT JOIN**:
    - It is also known as **LEFT OUTER JOIN**. It displays all the records from the table specified on the left of the `OUTER JOIN` operator in the result set, with or without any matching records from the table specified on the right
    - If no matching record is found in any of the rows from the table placed on the right, then the corresponding columns of that row will contain `NULL` values

- The syntax for `LEFT OUTER JOIN` is:

```
SELECT COLUMN1, COLUMN2 FROM TABLE1 LEFT OUTER
JOIN TABLE2 ON (JOIN CONDITION);
```

where,

`SELECT` – retrieves data from columns in the table

`COLUMN1,…` – specifies the name of the column to retrieve data from

`LEFT OUTER JOIN` - displays all rows from the table specified on the left of the `OUTER JOIN` operator

`TABLE1` – specifies the name of the table

`JOIN CONDITION` – specifies the conditions to retrieve data from tables

Table lists the JOIN conditions that can be specified in the ON clause

Join Condition	Description
BETWEEN	Compares values that lie in between a specific range
COMPARISON	Compares two values using operators such as =, >, <, >=, <=, <>
EXIST	Determines whether a value exist in the given table
IN	Determines whether a value exist in the list of values or a table
IS NULL	Compares a value with an empty or NULL value
LIKE	Compares one value with another
SOME/ANY/ALL	Performs quantified comparisons

Using OUTER JOIN

- To view all the records from `EMP_DETAILS` and `EMP_DEPARTMENT` tables by joining them using `LEFT OUTER JOIN`, enter the following command at the command prompt:

```
SELECT EMP_DETAILS.E_ID, EMP_DEPARTMENT.D_NAME,
EMP_DEPARTMENT.DATE_OF_JOIN FROM EMP_DETAILS LEFT
OUTER JOIN EMP_DEPARTMENT ON EMP_DETAILS.E_ID =
EMP_DEPARTMENT.E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help

mysql> SELECT EMP_DETAILS.E_ID, EMP_DEPARTMENT.D_NAME,
    -> EMP_DEPARTMENT.DATE_OF_JOIN FROM EMP_DETAILS
    -> LEFT OUTER JOIN
    -> EMP_DEPARTMENT ON
    -> EMP_DETAILS.E_ID = EMP_DEPARTMENT.E_ID;
+------+-------------+--------------+
| E_ID | D_NAME      | DATE_OF_JOIN |
+------+-------------+--------------+
|  101 | RESEARCH    | 0000-00-00   |
|  102 | MARKETING   | 0000-00-00   |
|  103 | DEVELOPMENT | 0000-00-00   |
|  104 | SALES       | 0000-00-00   |
|  105 | SALES       | 0000-00-00   |
+------+-------------+--------------+
5 rows in set (0.00 sec)

mysql>
```

Using OUTER JOIN

- You can specify conditions in a join with the ON and USING clauses in the SELECT command
- The ON clause specifies the conditions to join tables
- The USING clause specifies the requirement for matching columns to be present in both the tables
- To view all the records from EMP_DETAILS and EMP_DEPARTMENT tables by joining them with the USING clause, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS LEFT JOIN EMP_DEPARTMENT
USING(E_ID);
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT * FROM EMP_DETAILS LEFT JOIN EMP_DEPARTMENT
    -> USING(E_ID);
+------+---------+----------+------------+------------+----------
---+--------------+--------------+
| E_ID | E_FNAME | E_LNAME  | E_ADDRESS  | E_PHONE_NO | D_NAME
   | DATE_OF_JOIN | DESIGNATION  |
+------+---------+----------+------------+------------+----------
---+--------------+--------------+
|  101 | JACK    | WILLIAMS | CALIFORNIA |     762340 | RESEARCH
   | 0000-00-00   | MANAGER      |
|  102 | PETER   | ADAMS    | NEW JERSEY |     628491 | MARKETING
   | 0000-00-00   | ASST.MANAGER |
|  103 | JENNIE  | NORTON   | TROY       |     638510 | DEVELOPME
NT | 0000-00-00   | MANAGER      |
|  104 | SAMUEL  | BUSH     | NEW YORK   |     438610 | SALES
   | 0000-00-00   | MANAGER      |
|  105 | MARY    | GRAF     | DOWN TOWN  |     248523 | SALES
   | 0000-00-00   | MANAGER      |
+------+---------+----------+------------+------------+----------
---+--------------+--------------+
5 rows in set (0.01 sec)

mysql>
```

Using OUTER JOIN

- The syntax for joining three tables using the `LEFT OUTER JOIN` is:

```
SELECT COLUMN1, COLUMN2 FROM TABLE1
LEFT OUTER JOIN TABLE2 ON (JOIN CONDITION)
LEFT OUTER JOIN TABLE3 ON (JOIN CONDITION);
```

where,

`SELECT` – retrieves data from the table

`COLUMN1,…` – specifies the name of the column to be retrieved

`TABLE1` – specifies the name of the table that contain the columns

`LEFT OUTER JOIN` – compares the records from tables using the conditions specified

`JOIN CONDITON` – contains the clauses to satisfy, before retrieving data from the tables

Using OUTER JOIN

- For example, to view all the records from EMP_DETAILS, EMP_SALARY, and EMP_DEPARTMENT tables by joining them with the common attribute E_ID, enter the following command at the command prompt:

```
SELECT E_FNAME,E_LNAME,E_ADDRESS,
BASIC_SALARY FROM EMP_DETAILS
LEFT OUTER JOIN
EMP_SALARY ON
EMP_DETAILS.E_ID = EMP_SALARY.E_ID
LEFT OUTER JOIN
EMP_DEPARTMENT ON
EMP_SALARY.E_ID = EMP_DEPARTMENT.E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT E_FNAME,E_LNAME,E_ADDRESS,
    -> BASIC_SALARY FROM EMP_DETAILS
    -> LEFT OUTER JOIN
    -> EMP_SALARY ON
    -> EMP_DETAILS.E_ID = EMP_SALARY.E_ID
    -> LEFT OUTER JOIN
    -> EMP_DEPARTMENT ON
    -> EMP_SALARY.E_ID = EMP_DEPARTMENT.E_ID;
+---------+----------+------------+--------------+
| E_FNAME | E_LNAME  | E_ADDRESS  | BASIC_SALARY |
+---------+----------+------------+--------------+
| JACK    | WILLIAMS | CALIFORNIA |      2000.00 |
| PETER   | ADAMS    | NEW JERSEY |      2500.00 |
| JENNIE  | NORTON   | TROY       |      3000.00 |
| THOMAS  | COOK     | WASHINGTON |      3500.00 |
| GEORGE  | BLAIR    | CALIFORNIA |      4000.00 |
| ADAMS   | EVE      | INDIANA    |         NULL |
+---------+----------+------------+--------------+
6 rows in set (0.01 sec)

mysql>
```

- **RIGHT OUTER JOIN:**
  - It displays all records from the table specified on the right of the `OUTER JOIN` operator in the result set, with or without any matching records from the table specified on the left
  - If there is no matching record found in any of the row from the table placed on the left, then the corresponding columns of that row will contain `NULL` values

Using OUTER JOIN

- The syntax for `RIGHT OUTER JOIN` is:

```
SELECT COLUMN1, COLUMN2 FROM TABLE1
RIGHT OUTER JOIN TABLE2 ON (JOIN CONDITION);
```

where,

`SELECT` – retrieves data from the table

`COLUMN1,…` – specifies the name for the column to retrieve data from

`TABLE1` – specifies the name of the table that contains the columns

`RIGHT OUTER JOIN` – specifies the comparison type to fetch data from the tables

Using OUTER JOIN

- For example, to display E_LNAME, E_ADDRESS, and E_PHONE_NO from EMP_DETAILS table where the employee ids range from 102 to 105, you will compare the values of EMP_DEPARTMENT table with the EMP_DETAILS table and display the output

```
SELECT DISTINCT E_LNAME,E_ADDRESS,E_PHONE_NO,D_NAME
FROM EMP_DETAILS E
RIGHT OUTER JOIN
EMP_DEPARTMENT D ON
(E.E_ID BETWEEN 102 AND 105);
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT DISTINCT
    -> E_LNAME, E_ADDRESS, E_PHONE_NO,
    -> D_NAME FROM EMP_DETAILS E
    -> RIGHT OUTER JOIN
    -> EMP_DEPARTMENT D ON
    -> (E.E_ID BETWEEN 102 AND 105);
+---------+------------+------------+-------------+
| E_LNAME | E_ADDRESS  | E_PHONE_NO | D_NAME      |
+---------+------------+------------+-------------+
| ADAMS   | NEW JERSEY |     628491 | RESEARCH    |
| NORTON  | TROY       |     638510 | RESEARCH    |
| BUSH    | NEW YORK   |     438610 | RESEARCH    |
| GRAF    | DOWN TOWN  |     248523 | RESEARCH    |
| ADAMS   | NEW JERSEY |     628491 | MARKETING   |
| NORTON  | TROY       |     638510 | MARKETING   |
| BUSH    | NEW YORK   |     438610 | MARKETING   |
| GRAF    | DOWN TOWN  |     248523 | MARKETING   |
| ADAMS   | NEW JERSEY |     628491 | DEVELOPMENT |
| NORTON  | TROY       |     638510 | DEVELOPMENT |
| BUSH    | NEW YORK   |     438610 | DEVELOPMENT |
| GRAF    | DOWN TOWN  |     248523 | DEVELOPMENT |
| ADAMS   | NEW JERSEY |     628491 | SALES       |
| NORTON  | TROY       |     638510 | SALES       |
| BUSH    | NEW YORK   |     438610 | SALES       |
| GRAF    | DOWN TOWN  |     248523 | SALES       |
+---------+------------+------------+-------------+
16 rows in set (0.01 sec)
```

Using OUTER JOIN

- For example, to display E_LNAME, E_ADDRESS, and E_PHONE_NO from EMP_DETAILS table where the employee ids are greater than 102, perform a join operation of the EMP_DEPARTMENT table with the EMP_DETAILS table

```
SELECT DISTINCT E_LNAME,E_ADDRESS,E_PHONE_NO FROM
EMP_DETAILS E
RIGHT OUTER JOIN EMP_DEPARTMENT D ON (E.E_ID > 102);
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT DISTINCT E_LNAME,
    -> E_ADDRESS,
    -> E_PHONE_NO
    -> FROM
    -> EMP_DETAILS E
    -> RIGHT OUTER JOIN
    -> EMP_DEPARTMENT
    -> D ON (E.E_ID > 102);
+---------+-----------+------------+
| E_LNAME | E_ADDRESS | E_PHONE_NO |
+---------+-----------+------------+
| NORTON  | TROY      |     638510 |
| BUSH    | NEW YORK  |     438610 |
| GRAF    | DOWN TOWN |     248523 |
+---------+-----------+------------+
3 rows in set (0.00 sec)

mysql>
```

Using OUTER JOIN

- A `SUBSTRING` function extracts string or character literal from a column of a table, specified in the column reference
- The `FROM` clause of `SUBSTRING` function specifies the character position from where the extracted string should start
- The `FOR` clause of the function specifies the length of the extracted string

Using OUTER JOIN

- To use the SUBSTRING function with the RIGHT OUTER JOIN, enter the following command at the command prompt:

```
SELECT D.D_NAME,
D.DATE_OF_JOIN,
D.DESIGNATION
FROM EMP_DEPARTMENT D
RIGHT OUTER JOIN
EMP_DETAILS E
ON(SUBSTRING(E.E_ADDRESS,1,3) = 'CAL');
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT D.D_NAME,D.DATE_OF_JOIN,D.DESIGNATION FROM EMP_DEPARTMENT D RIGHT
OUTER JOIN EMP_DETAILS E ON(SUBSTRING(E.E_ADDRESS,1,3) = 'CAL');
+-------------+--------------+--------------+
| D_NAME      | DATE_OF_JOIN | DESIGNATION  |
+-------------+--------------+--------------+
| RESEARCH    | 2009-06-04   | MANAGER      |
| MARKETING   | 2009-07-05   | ASST.MANAGER |
| DEVELOPMENT | 2009-08-07   | MANAGER      |
| SALES       | 2009-07-07   | MANAGER      |
| SALES       | 2009-02-09   | MANAGER      |
| PRODUCTION  | 2009-04-03   | MANAGER      |
| SALES       | 2009-05-03   | MANAGER      |
| MARKETING   | 2005-05-15   | MANAGER      |
| SALES       | 2001-04-16   | TRAINEE      |
| NULL        | NULL         | NULL         |
| NULL        | NULL         | NULL         |
| NULL        | NULL         | NULL         |
| RESEARCH    | 2009-06-04   | MANAGER      |
| MARKETING   | 2009-07-05   | ASST.MANAGER |
| DEVELOPMENT | 2009-08-07   | MANAGER      |
| SALES       | 2009-07-07   | MANAGER      |
| SALES       | 2009-02-09   | MANAGER      |
| PRODUCTION  | 2009-04-03   | MANAGER      |
| SALES       | 2009-05-03   | MANAGER      |
```

Using Self-Join

- A self-join statement joins or compares a table with itself
- It means that self-join compares records of a column with the other
- The syntax for self-join is:

```
SELECT DISTINCT COLUMN1, COLUMN2 FROM TABLE WHERE
(JOIN CONDITION);
```

where,

`SELECT` – retrieves data from the table

`DISTINCT` – displays unique values

`COLUMN1,…` – specifies the name for the column to retrieve data from

`TABLE` – specifies the name of the table that contains the columns

Using Self-Join

- For example, to view all the employees, who are located in the same city, enter the following command at the command prompt:

```
SELECT DISTINCT E1.E_ID, E1.E_FNAME, E1.E_LNAME,
E1.E_ADDRESS
FROM EMP_DETAILS AS E1, EMP_DETAILS AS E2
WHERE E1.E_ADDRESS = E2.E_ADDRESS
AND E1.E_ADDRESS IS NOT NULL
AND E2.E_ADDRESS IS NOT NULL
AND E1.E_ID <> E2.E_ID
ORDER BY E1.E_ADDRESS, E1.E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT DISTINCT E1.E_ID,E1.E_FNAME,E1.E_LNAME,E1.E_ADDRESS
    -> FROM EMP_DETAILS AS E1,EMP_DETAILS AS E2
    -> WHERE E1.E_ADDRESS = E2.E_ADDRESS
    -> AND E1.E_ADDRESS IS NOT NULL
    -> AND E2.E_ADDRESS IS NOT NULL
    -> AND E1.E_ID <> E2.E_ID
    -> ORDER BY E1.E_ADDRESS,E1.E_ID;
+------+---------+----------+------------+
| E_ID | E_FNAME | E_LNAME  | E_ADDRESS  |
+------+---------+----------+------------+
|  101 | JACK    | WILLIAMS | CALIFORNIA |
|  106 | BOB     | WILLIAMS | CALIFORNIA |
+------+---------+----------+------------+
2 rows in set (0.00 sec)

mysql>
```

Subqueries

- When a SELECT query is placed inside another SELECT query, then the inner SELECT query is said to be a subquery
- The syntax for the subquery is:

```
SELECT [*] column_name… FROM table_name1
WHERE column_name IN
(SELECT column_name FROM table_name2);
```

where,

SELECT – retrieves data from the table

* - retrieves all the records from the table

column_name – specifies the name of the column

IN – compares data from both the tables

table_name1 – specifies the name of the table

Subqueries

- Consider an example, where you want to view the details of all the employees from all the departments
- In this case, you will compare the employee `ID` in the `EMP_DEPARTMENT` and the `EMP_DETAILS` table
- You will display the results only if the `E_ID` from `EMP_DEPARTMENT` matches with `E_ID` of `EMP_DETAILS`
- To view all records from `EMP_DETAILS` and `EMP_DEPARTMENT` tables using subqueries, enter the following command at the command prompt:

```
SELECT * FROM EMP_DETAILS WHERE E_ID IN (SELECT E_ID
FROM EMP_DEPARTMENT);
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT * FROM EMP_DETAILS
    -> WHERE E_ID IN
    -> (SELECT E_ID FROM EMP_DEPARTMENT);
+------+---------+----------+------------+------------+
| E_ID | E_FNAME | E_LNAME  | E_ADDRESS  | E_PHONE_NO |
+------+---------+----------+------------+------------+
|  101 | JACK    | WILLIAMS | CALIFORNIA |     762340 |
|  102 | PETER   | ADAMS    | NEW JERSEY |     628491 |
|  103 | JENNIE  | NORTON   | TROY       |     638510 |
|  104 | SAMUEL  | BUSH     | NEW YORK   |     438610 |
|  105 | ,MARY   | GRAF     | DOWN TOWN  |     248523 |
+------+---------+----------+------------+------------+
5 rows in set (0.03 sec)

mysql>
```

- You can use the `NOT IN` operator to display the unmatched values of tables.
- For example, to view `E_ID` of `EMP_DEPARMENT` table that do not match with the `E_ID` of `EMP_DETAILS` table, enter the following command at the command prompt:

```
SELECT * FROM EMP_DEPARTMENT
WHERE E_ID NOT IN
(SELECT E_ID FROM EMP_DETAILS);
```

Figure displays the output of the command

```
mysql> SELECT * FROM EMP_DEPARTMENT
    -> WHERE E_ID NOT IN
    -> (SELECT E_ID FROM EMP_DETAILS);
+------+-------------+--------------+-------------+
| E_ID | D_NAME      | DATE_OF_JOIN | DESIGNATION |
+------+-------------+--------------+-------------+
|  107 | PRODUCTION  | 2009-10-10   | MANAGER     |
|  108 | DEVELOPMENT | 2009-12-10   | MANAGER     |
|  109 | PRODUCTION  | 2009-12-10   | MANAGER     |
+------+-------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql>
```

Subqueries

- MySQL also supports the use of SELECT statement within the REPLACE and INSERT queries
- It means that when a SELECT query is used within an INSERT or REPLACE query, then that SELECT query is said to be subquery of INSERT or REPLACE query
- For example, to insert department names of all employees from EMP_DEPARTMENT table to EMP_SALARY table, enter the following command at the command prompt:

```
INSERT INTO EMP_SALARY(D_NAME) SELECT D_NAME FROM
EMP_DEPARTMENT;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help

mysql> INSERT INTO EMP_SALARY(D_NAME) SELECT D_NAME FROM EMP_DEPART
MENT;
Query OK, 5 rows affected, 1 warning (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 1

mysql>
```

Unions

- Union is used to combine many result sets used with `SELECT` command into a single result set
- The syntax for the `UNION` command is:

```
SELECT * FROM TABLE1
UNION [ALL | DISTINCT]
SELECT * FROM TABLE2
UNION [ALL | DISTINCT];
```

where,

`SELECT` – retrieves data from the table

`*` - retrieves all data from the table

`TABLE1` – specifies the name of the table that contains data

`UNION` – combines query results into a single result

Unions

- The columns specified in the SELECT command should be of the same type
- MySQL will use the column names from the first SELECT query as the column names for the results returned
- MySQL provides the ALL keyword to display matching rows from all the SELECT statements
- If this keyword is not used, only unique rows are displayed
- For example, to view E_ID and E_FNAME from EMP_DETAILS table and E_ID and D_NAME from EMP_DEPARTMENT table, enter the following command at the command prompt:

```
SELECT E_ID, E_FNAME FROM EMP_DETAILS
UNION
SELECT E_ID, D_NAME FROM EMP_DEPARTMENT;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> SELECT E_ID,E_FNAME FROM EMP_DETAILS
    -> UNION
    -> SELECT E_ID, D_NAME FROM EMP_DEPARTMENT;
+------+-------------+
| E_ID | E_FNAME     |
+------+-------------+
|  101 | JACK        |
|  102 | PETER       |
|  103 | JENNIE      |
|  104 | SAMUEL      |
|  105 | MARY        |
|  101 | RESEARCH    |
|  102 | MARKETING   |
|  103 | DEVELOPMENT |
|  104 | SALES       |
|  105 | SALES       |
+------+-------------+
10 rows in set (0.08 sec)

mysql>
```

Unions

- The `ORDER BY` clause can also be used to sort the entire UNION results
- The `SELECT` statements must be included within parenthesis
- MySQL supports the `LIMIT` clause to restrict the number of records returned by the query
- These values must be in the form of non-negative integer constants
- The first value in the argument defines the first row to return and the second value defines the maximum number of rows to return

- For example, to sort the records retrieved from the EMP_DETAILS table where the employee addresses are California and Troy, enter the following command at the command prompt:

```
(SELECT * FROM EMP_DETAILS WHERE E_ADDRESS = 'TROY' ORDER BY
E_ID LIMIT 2)UNION(SELECT * FROM EMP_DETAILS WHERE E_ADDRESS =
'CALIFORNIA' ORDER BY E_ID LIMIT 1)ORDER BY E_ID;
```

Figure displays the output of the command

```
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
mysql> (SELECT * FROM EMP_DETAILS
    -> WHERE E_ADDRESS = 'TROY' ORDER BY E_ID LIMIT 2)
    -> UNION
    -> (SELECT * FROM EMP_DETAILS
    -> WHERE E_ADDRESS = 'CALIFORNIA' ORDER BY
    -> E_ID LIMIT 1)
    -> ORDER BY E_ID;
+------+---------+----------+------------+------------+
| E_ID | E_FNAME | E_LNAME  | E_ADDRESS  | E_PHONE_NO |
+------+---------+----------+------------+------------+
|  101 | JACK    | WILLIAMS | CALIFORNIA |     762340 |
|  103 | JENNIE  | NORTON   | TROY       |     638510 |
+------+---------+----------+------------+------------+
2 rows in set (0.00 sec)

mysql>
```

Summary

- Joining of tables means combining two or more records from different tables of the same database into one comprehensive structure
- Joining of tables is performed either using `WHERE` clause with the `SELECT` command or using the `JOIN` keyword. This will enhance manipulation, increase access speed, and reduce data redundancy
- In an Equi-Join, the comparison is made between two columns that contain similar values
- The ON clause specifies the conditional expression to be used in a WHERE clause. It also supports the use of relational operators such as =, <, >, <=, >=, or <>
- The `USING` clause can be used in a join only when the tables have matching columns

Summary

- Numeric fields can be joined together, as long as they are of same data type such as, AutoNumber or Long
- Incase of non-numeric data, the fields must be of same type and length, and should contain same kind of data
- An `INNER JOIN` creates a virtual table by combining the fields of both tables that satisfy the query for both the tables
- The `AND` or the `OR` operators can also be used with `INNER JOIN` command
- An `OUTER JOIN` is used to join two tables, a source and joining table, which have one or more columns in common
- The rows of the tables that may not have any matching value in the other tables are also displayed in the result set

Summary

- MySQL supports two types of `OUTER JOINS`: `LEFT OUTER JOIN` and `RIGHT OUTER JOIN`
- The `LEFT OUTER JOIN` displays all rows from the table specified on the left of the `OUTER JOIN` operator in the result set, with or without any matching record with the table specified on the right
- The `RIGHT OUTER JOIN` displays all rows from the table specified on the right of the `OUTER JOIN` operator in the result set, with or without any matching record with the table specified on the left
- When using the `OUTER JOIN`, if there is no match found in any of the rows from the table placed on the right, then the corresponding columns of that row will contain NULL values
- A Self-Join is a query that is used to join or compare a table to itself

Summary

- The `DISTINCT` clause lists unique records from tables
- A query used inside another select query is known as subquery
- The `UNION` clause can be used to combine results from different `SELECT` commands
- The `ORDER BY` clause must be used with a parenthesis to sort results displayed by the `UNION` option
- The `LIMIT` clause restricts the number of results displayed by the query