Table Joins and Indexes in SQL

Based on CBSE Curriculum

Class ~~11~~

By-

Neha Tyagi

PGT CS

KV 5 Jaipur II Shift

Jaipur Region

Introduction

- Sometimes we need an information which is receiving data from multiple tables.
- If the tables have some common field then it become easier to join these tables and can retrieve the data.
- In this chapter, we will learn joining of tables and the process of information retrieval.
- In this chapter we will also learn indexes of SQL which facilitates database processing.

JOINS

- Join is a query which combine rows of two or more tables.
- In a join-query, we need to provide a list of tables in FROM Clause.
- The process of combining multiple tables in order to retrieve data is called joining. For ex-

```
SELECT * FROM emp1, dept;
```

- Unrestricted join or cartesian product of both the tables gives all possible concatenations of all the rows of both the tables.

Making Table ready for join

```
mysql> create table emp1(empcode INT(4) Primary key,
    -> Name CHAR(20) NOT NULL,
    -> City CHAR(10),
    -> Salary INT(6) check(Salary>15000),
    -> age INT(2) Check(age>16));
Query OK, 0 rows affected (0.08 sec)

mysql> CREATE TABLE DEPT(DeptID INT(3) PRIMARY KEY,
    -> DEPTNAME CHAR(10) NOT NULL,
    -> LOCATION CHAR(10),
    -> DEPTIC INT(4) REFERENCES emp1(empcode));
Query OK, 0 rows affected (0.11 sec)
```

In above given example there are two tables namely emp1 and dept.
In emp1 table, primary key is empcode and in table dept foreign key is empcode which is referring empcode of emp1 table.

Making Table ready for join

```
mysql> Select * from emp1;
+---------+--------+------+--------+------+
| empcode | Name   | City | Salary | age  |
+---------+--------+------+--------+------+
|    1001 | Amit   | BBK  |  16000 |   27 |
|    1002 | Suresh | LKO  |  26000 |   37 |
|    1003 | Ashish | KNP  |  26000 |   30 |
|    1004 | Harish | BBK  |  22000 |   28 |
|    1005 | Ankit  | LKO  |  24000 |   27 |
|    1006 | Sumit  | KNP  |  23000 |   25 |
+---------+--------+------+--------+------+
6 rows in set (0.00 sec)

mysql> Select * from dept;
+--------+------------+----------+--------+
| DeptID | DEPTNAME   | LOCATION | DEPTIC |
+--------+------------+----------+--------+
|    101 | Sales      | KNP      |   1002 |
|    102 | Purchase   | BBK      |   1004 |
|    103 | Admin      | LKO      |   1005 |
|    104 | Production | BBK      |   1003 |
+--------+------------+----------+--------+
4 rows in set (0.00 sec)
```

By analyzing the tables we will be informed that how table are joined here.

Now we will see that how join works.

Join query

```
mysql> select * from emp1,dept;
+---------+--------+------+--------+------+--------+------------+----------+----
----+
| empcode | Name   | City | Salary | age  | DeptID | DEPTNAME   | LOCATION | DEP
TIC |
+---------+--------+------+--------+------+--------+------------+----------+----
----+
|    1001 | Amit   | BBK  |  16000 |   27 |    101 | Sales      | KNP      |   1
002 |
|    1001 | Amit   | BBK  |  16000 |   27 |    102 | Purchase   | BBK      |   1
004 |
|    1001 | Amit   | BBK  |  16000 |   27 |    103 | Admin      | LKO      |   1
005 |
|    1001 | Amit   | BBK  |  16000 |   27 |    104 | Production | BBK      |   1
003 |
|    1002 | Suresh | LKO  |  26000 |   37 |    101 | Sales      | KNP      |   1
002 |
|    1002 | Suresh | LKO  |  26000 |   37 |    102 | Purchase   | BBK      |   1
004 |
|    1002 | Suresh | LKO  |  26000 |   37 |    103 | Admin      | LKO      |   1
005 |
|    1002 | Suresh | LKO  |  26000 |   37 |    104 | Production | BBK      |   1
003 |
|    1003 | Ashish | KNP  |  26000 |   30 |    101 | Sales      | KNP      |   1
002 |
|    1003 | Ashish | KNP  |  26000 |   30 |    102 | Purchase   | BBK      |   1
004 |
|    1003 | Ashish | KNP  |  26000 |   30 |    103 | Admin      | LKO      |   1
005 |
|    1003 | Ashish | KNP  |  26000 |   30 |    104 | Production | BBK      |   1
003 |
|    1004 | Harish | BBK  |  22000 |   28 |    101 | Sales      | KNP      |   1
002 |
|    1004 | Harish | BBK  |  22000 |   28 |    102 | Purchase   | BBK      |   1
004 |
|    1004 | Harish | BBK  |  22000 |   28 |    103 | Admin      | LKO      |   1
005 |
|    1004 | Harish | BBK  |  22000 |   28 |    104 | Production | BBK      |   1
003 |
|    1005 | Ankit  | LKO  |  24000 |   27 |    101 | Sales      | KNP      |   1
002 |
|    1005 | Ankit  | LKO  |  24000 |   27 |    102 | Purchase   | BBK      |   1
004 |
|    1005 | Ankit  | LKO  |  24000 |   27 |    103 | Admin      | LKO      |   1
005 |
|    1005 | Ankit  | LKO  |  24000 |   27 |    104 | Production | BBK      |   1
003 |
|    1006 | Sumit  | KNP  |  23000 |   25 |    101 | Sales      | KNP      |   1
002 |
|    1006 | Sumit  | KNP  |  23000 |   25 |    102 | Purchase   | BBK      |   1
004 |
|    1006 | Sumit  | KNP  |  23000 |   25 |    103 | Admin      | LKO      |   1
005 |
|    1006 | Sumit  | KNP  |  23000 |   25 |    104 | Production | BBK      |   1
003 |
+---------+--------+------+--------+------+--------+------------+----------+----
```

mysql> SELECT * from emp1, dept;

Data from each column with the combination of each record has show on execution of query.

Condition is to be used to filter this data.

Join query

To get the details about the departments and their in- charges, query will be-

mysql> SELECT name, deptname from emp1, dept
where emp1.empcode=dept.deptic;

When both the tables have same field name, then to show a field from particular table, use the following pattern to access a field- <Table name>.<Field Name>

Ex- *emp1.empcode*

```
mysql> select name, deptname from emp1, dept
    -> where emp1.empcode=dept.deptic;
+--------+------------+
| name   | deptname   |
+--------+------------+
| Suresh | Sales      |
| Harish | Purchase   |
| Ankit  | Admin      |
| Ashish | Production |
+--------+------------+
4 rows in set (0.00 sec)
```

This is an example of **Equi-Join,** in which columns are compared for equality and it is also an example of **Natural- Join**, in which only one of the identical columns exists.

Additional search with Join Query

If you want to get information that Ankit is in-charge of which department, following query will be used.-

mysql> SELECT name, deptname from emp1, dept where emp1.empcode=dept.deptic and emp1.name='Ankit';

```
mysql> select name, deptname from emp1, dept
    -> where emp1.empcode=dept.deptic
    -> and emp1.name='Ankit';
+--------+------------+
| name   | deptname   |
+--------+------------+
| Ankit  | Admin      |
+--------+------------+
1 row in set (0.02 sec)
```

Using Table Aliases

- Sometimes we have very big file names and more than one table have same filed names.
- In such cases, we need to write file names again and again which consumes time.
- This can be solved by using alias for tables.

See the following example-

These are table aliases.

```
mysql> SELECT E.name, E.Salary, D.DeptName
                    FROM emp1 E, dept D
                   WHERE E.empcode=D.deptic;
```

```
mysql> SELECT E.name, E.salary, D.deptname
    -> FROM emp1 E, dept D
    -> WHERE E.empcode=D.deptic;
+---------+--------+-------------+
| name    | salary | deptname    |
+---------+--------+-------------+
| Suresh  |  26000 | Sales       |
| Harish  |  22000 | Purchase    |
| Ankit   |  24000 | Admin       |
| Ashish  |  26000 | Production  |
+---------+--------+-------------+
4 rows in set (0.02 sec)
```

Joining more than one table

- Sometimes it is required to get the data from more than one table. In such cases , tables are to be joined on the basis of some matching columns.
- See the following example-

```
mysql> SELECT E.name, E.Salary, D.DeptName, G.grade
                    FROM emp1 E, dept D, salarygrade G
                WHERE E.empcode=D.deptic
                AND E.salary BETWEEN G.lowsal AND G.hisal;
```

```
mysql> Select E.name, E.salary, D.deptname, G.Grade
    -> from emp1 E, dept D, salarygrade G
    -> where E.empcode=D.deptic AND
    -> E.salary between G.lowsal and G.hisal;
+--------+--------+------------+-------+
| name   | salary | deptname   | Grade |
+--------+--------+------------+-------+
| Harish |  22000 | Purchase   |     2 |
| Ankit  |  24000 | Admin      |     2 |
| Suresh |  26000 | Sales      |     3 |
| Harish |  22000 | Purchase   |     3 |
| Ankit  |  24000 | Admin      |     3 |
| Ashish |  26000 | Production |     3 |
| Suresh |  26000 | Sales      |     4 |
| Ankit  |  24000 | Admin      |     4 |
| Ashish |  26000 | Production |     4 |
+--------+--------+------------+-------+
9 rows in set (0.01 sec)
```

This is an example of **Non-Equi-Join** where condition is given without using '='.

Joining table using JOIN Clause

- SQL provides some special clauses for joining of tables- JOIN (and NATURAL JOIN) clause .

```
mysql > SELECT * FROM <table1>
                    [CROSS] [NATURAL] JOIN <Table2>
                    [ON (<Join Condition>) | USING(<JoinFields>)];
```

- For creating cartesian product -
  SELECT * FROM emp1 JOIN dept;

- For creating CROSS JOIN -
  SELECT * FROM emp1 CROSS JOIN dept;

- For creating EQUI- JOIN-
  SELECT * FROM emp1 e. JOIN dept d
  ON (e.empcode=d.deptic);

- For creating NATURAL JOIN-
  SELECT * FROM emp1 NATURAL JOIN dept;

These will give the same output as we have seen in previous slides.

LEFT-JOIN

- When we use LEFT-JOIN, it returns all rows from first table whether it has matching rows in second table or not.
- It shows NULL in columns for the unmatched rows of first table.

mysql>SELECT <Col List> FROM <table1> LEFT JOIN <table2>
ON <joining Condition>

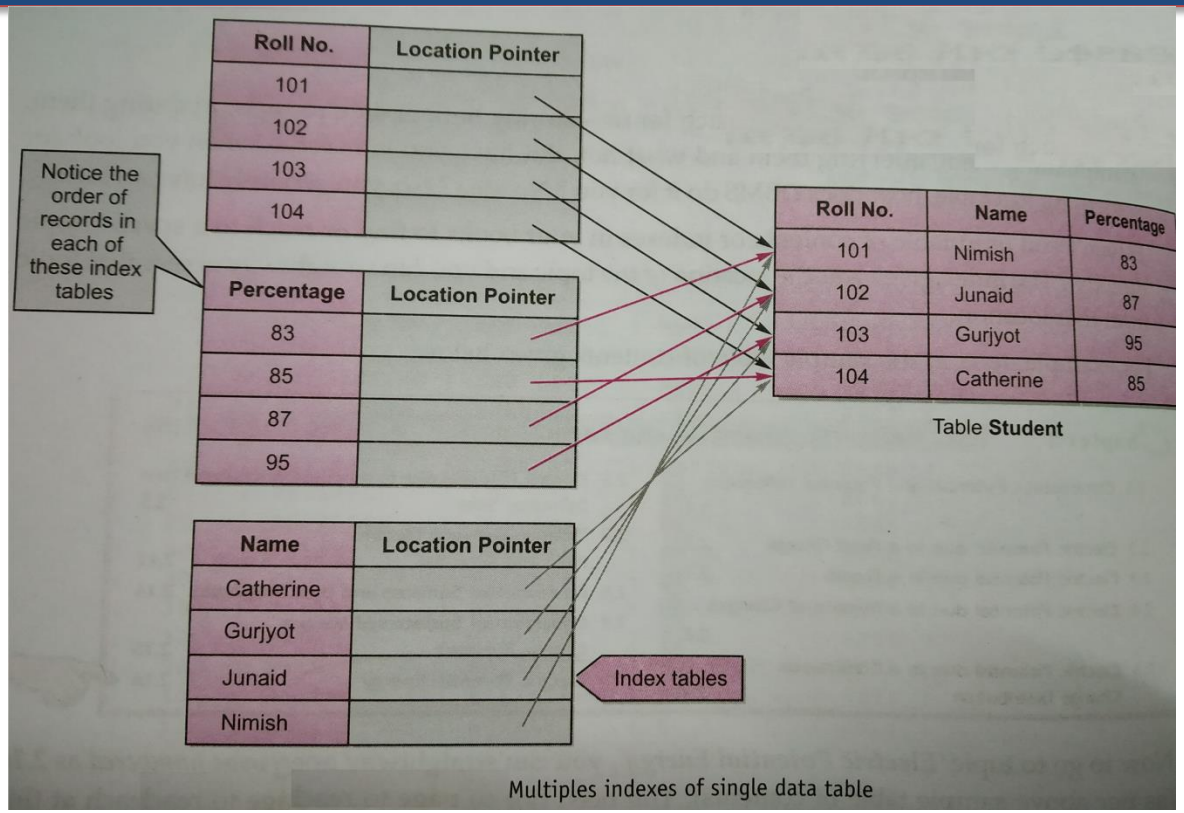
Right-JOIN

- When we use RIGHT-JOIN, it returns all rows from second table whether it has matching rows in first table or not.
- It shows NULL in columns for the unmatched rows of second table.

mysql>SELECT <Col List> FROM <table1> RIGHT JOIN <table2>
ON <joining Condition>

Indexes in Database

- Index is a data structure maintained by a database that helps to find records within a table more quickly.
- An index stores the sorted/ordered values within the index field and their location in the actual table.
- An index in a database is also a table which stores arranged values of one or more columns in a specific order.



Multiples indexes of single data table

Creation of Indexes in MySQL

- We can create indexes in MySQL by two methods-
  1. At the time of table creation.
  2. Creation of index at some already existing table.

- Syntax for first case–

```
CREATE TABLE <TableName>  (<Col1> <type(Size)> <Constraint>,
        <Col2> <type(Size)> <Constraint>,
        <Col3> <type(Size)> <Constraint>, . . . .
        INDEX <IndexName> (<IndexCol1Name>  <Length> <ASC/DESC>,
                        <IndexCol2Name>  <Length> <ASC/DESC>, . . ));
```

- Example :

```
mysql>Create table PLAYERS ( PLAYERNO  INT NOT NULL,
                             NAME  CHAR(15) NOT NULL,
                             DOB DATE,
                             SEX CHAR(1) NOT NULL,
                             ADDRESS VARCHAR(100) NOT NULL,
                             PHONE CHAR(10),
                             TEAM NO   CHAR(4) NOT NULL,
                             PRIMARY KEY (PLAYERNO),
                             INDEX Player_idx(NAME(5) ) );
```

Creation of Indexes in MySQL

- Syntax for second case-

```
CREATE  INDEX <IndexName> ON
                    <TableName> (<Col1name> [ASC|DESC],
                                 <Col2name> [ASC|DESC], . . );
```

Or to keep unique vales-

```
CREATE  UNIQUE INDEX <IndexName> ON
                    <TableName> (<Col1name> [ASC|DESC],
                                 <Col2name> [ASC|DESC], . . );
```

Example:

mysql>Create index Player_idx on Players (name(5));

OR

Create Unique index Player_idx on Players (Teamno);

Indexes in Database

- To show the Indexes, use the following command –

  mysql> SHOW INDEXES FROM <TableName>;

Example :

mysql> SHOW INDEXES FROM Players;

- To delete the Indexe, use the following command

  mysql> DROP INDEX <IndexName> ON <Tablename>;

Example:

mysql> Drop Index Player_idx ON Players;

- To rename the Indexe, use the following command

  mysql> ALTER TABLE <TableName> RENAME INDEX <OldName> TO <NewName>;

Example :

mysql> ALTER TABLE Players RENAME INDEX Player_idx TO Player_new_idx;

Advantages & Disadvantages of Indexes

- **Advantages** :
  - With Indexes, queries gives much better performance.
  - Data retrieval is much faster with Indexes.
  - Indexes are very useful for Sorting purpose.
  - Unique indexes guarantee uniquely identifiable records in the database.

- **Disadvantages** :
  - With Indexes, the performance of insert, update and delete decreases. As every time insert/update/delete operation happens, the index is to be updated accordingly.
  - Index consumes storage space and this increases with the number of fields used and the length of the table.

  Thus, it is advised that one should only create indexes only when actually needed.

Thank you

Please follow us on our blog

www.pythontrends.wordpress.com