Table Joins and Indexes in SQL

सीबीएसई पाठ्यक्रम पर आधारित

कक्षा -11

द्वारा:

संजीव भदौरिया

स्नातकोत्तर शिक्षक (संगणक विज्ञान)

के० वि० बाराबंकी (लखनऊ संभाग)

परिचय

- कभी कभी हमें ऐसी इनफार्मेशन की आवश्यकता पड़ती है जिसमे डाटा एक से अधिक tables से आता है |
- यदि tables किसी common field से आपस में जुडी हों तो आप tables को आसानी से join कर सकते हैं और डाटा या इनफार्मेशन निकल सकते हैं |
- इस अध्याय में हम SQL के अंतर्गत tables को आपस में join करना तथा उनसे इनफार्मेशन निकलना सीखेंगे |
- इस अध्याय में हम SQL में indexes भी सीखेंगे जिसके कारण बड़े databases से query को प्रोसेस कराना तेज़ हो जाता है |

JOINS

- Join एक प्रकार की query होती है जो दो या अधिक tables के rows इकठ्ठा करती है |
- एक join-query में FROM Clause में एक से अधिक tables की सूची देते हैं |
- कई tables के डाटा को एक साथ लेन के फंक्शन को joining कहते हैं | उदहारण के लिए –

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

- दो tables के unrestricted join या cartesian product से दोनों tables के समस्त row से सभी संभव जोड़ बन जाते हैं |

Table को join के लिए तैयार करना

```
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)
```

उपरोक्त उदहारण में 2 table बनाई गयी हैं जिसमें emp1 table में primary key है empcode तथा dept table में foreign key है deptic जो कि emp1 table के empcode को refer कर रही है |

Table को join के लिए तैयार करना

```
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)
```

इस table का अध्ययन करें तो हम को समझ आजायेगा की table एक दूसरे से कैसे जुडी हैं |

अब ये देखते हैं की join query कैसे काम करती है |

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;

Command चलने से समस्त record के combination के साथ हर column के डाटा को शो कर दिया |

इनको फ़िल्टर करने के लिए query में condition लगाई जाती है |

Join query

ये पता करना हो कि कौन सा employee किस department में प्रभारी है तो निम्न query को run करना होगा |

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

जब दोनों table में सामान name से field हों तो perticular table की field को दर्शाने के लिए <TableName>.<ColName> के प्रारूप में लिखते हैं जैसे - *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)
```

यह एक प्रकार का **Equi- Join** का उदहारण है जहाँ tables को column की values को बराबरी के साथ access करके जोड़ा जाता है | तथा यह **Natural- Join** का भी उदहारण है जहाँ joining tables एक एक ही column का प्रयोग किया जाता है |

Join query के साथ additional search

यदि ये पता करना हो कि Ankit किस department में प्रभारी है तो निम्न query को run करना होगा |

```
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)
```

Table Aliases का प्रयोग करना

- कभी कभी ऐसा होता है की tables का नाम बहुत बड़ा होता है तथा अलग अलग tables में column के name सामान होते हैं |
- ऐसे में '.' के साथ बार बार name लिखने में समय बर्बाद होता है तो हम table के alias बना लेते हैं जो की अस्थाई lebel होते हैं |
- इसके लिए निम्न उदहारण को ध्यान से देखें

ये table के Alias name हैं |

```
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)
```

दो से अधिक tables को जोड़ना

```
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)
```

यह एक प्रकार का Non-Equi-Join का उदहारण है जहाँ बिना '=' का प्रयोग किये condition दी गयी है |

JOIN Clause के द्वारा tables को जोड़ना

- SQL दो tables को जोड़ने के लिए कुछ विशेष clauses भी प्रदान करता है - JOIN (और NATURAL JOIN) clause .

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

- इसमें cartesian product बनाने के लिए -
  SELECT * FROM emp1 JOIN dept;

- CROSS JOIN बनाने के लिए –
  SELECT * FROM emp1 CROSS JOIN dept;

- EQUI- JOIN बनाने के लिए –
  SELECT * FROM emp1 e. JOIN dept d
  ON (e.empcode=d.deptic);

- NATURAL JOIN बनाने के लिए -
  SELECT * FROM emp1 NATURAL JOIN dept;

इनके output भी वैसे ही आएंगे जैसे की हम पीछे देख चुके हैं |

LEFT-JOIN

- जब हम LEFT-JOIN का प्रयोग करते हैं तो पहली table से समस्त row return होती हैं चाहे दूसरी table से row match हो या न हों |
- पहली table के unmatched row के लिए दूसरी table के column में NULL दिखाई देता है|

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

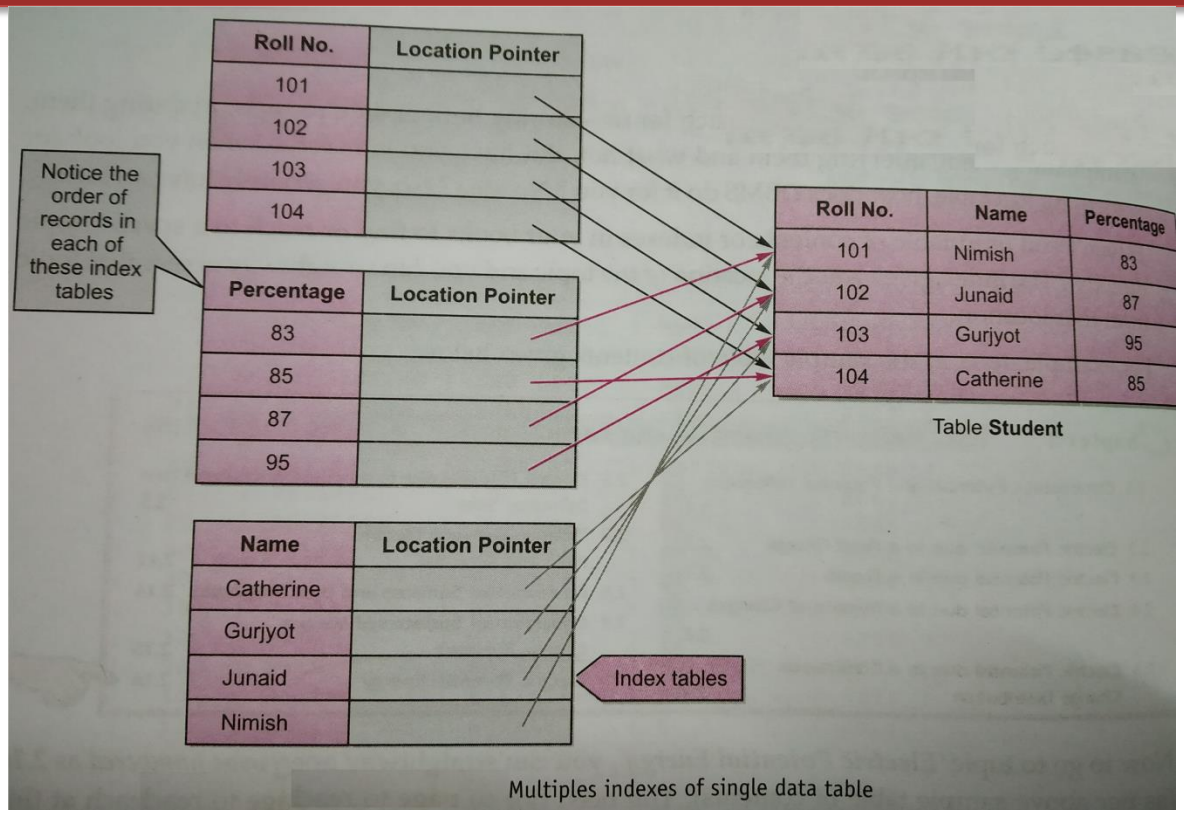
Right-JOIN

- जब हम RIGHT-JOIN का प्रयोग करते हैं तो दूसरी table से समस्त row return होती हैं चाहे पहली table से row match हो या न हों |
- दूसरी table के unmatched row के लिए पहली table के column में NULL दिखाई देता है|

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

Database में Indexes

- Index एक प्रकार का data structure होता है जिसका प्रयोग database किसी table से records को अधिक तेज़ी खोजने में करता है |
- एक index अपने index field में actual table में record के लोकेशन की sort की हुई या ordered values को रखता है |
- Database में index स्वयं एक table होती है जो अपने column में values को एक निश्चित क्रम में संगहीत रखती है |



MySQL में Indexes बनाना

- MySQL में Index हम दो प्रकार से बना सकते हैं –
  1. Table को create करते समय
  2. किसी पहले से उपस्थित table पर index बनाना

- पहले वाले case में syntax निम्नवत होगा –

```
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>, . . ));
```

- उदहारण :

```
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) ) );
```

MySQL में Indexes बनाना

- MySQL में Index हम दो प्रकार से बना सकते हैं –
  1. Table को create करते समय
  2. किसी पहले से उपस्थित table पर index बनाना

- दूसरे वाले case में syntax निम्नवत होगा –

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

अथवा unique values को रखने के लिए

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

उदहारण :

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

अथवा

```
Create Unique index Player_idx on Players (Teamno);
```

Database में Indexes

- Indexes को देखने के लिए निम्न commands का प्रयोग करें –

  mysql> SHOW INDEXES FROM <TableName>;

उदाहरण :

mysql> SHOW INDEXES FROM Players;

- Indexes को delete करने के लिए निम्न commands का प्रयोग करें –

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

उदाहरण :

mysql> Drop Index Player_idx ON Players;

- Indexes को rename करने के लिए निम्न commands का प्रयोग करें –

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

उदाहरण :

mysql> ALTER TABLE Players RENAME INDEX Player_idx
TO Player_new_idx;

Indexes के Advantages & Disadvantages

- **Advantages** :
  - Indexes के साथ queries बेहतर कार्य करती हैं |
  - Indexes के साथ डाटा की प्राप्ति अधिक तेज़ी से होती है |
  - Sorting के उद्देश्य से index बहुत मदद् गार है
  - Index, Database में uniquely identifiable records को uniquely खोजने के गारंटी देता है |

- **Disadvantages** :
  - Indexes के साथ insert, update और delete की कार्यक्षमता घट जाती है | क्योंकि प्रत्येक insert/update/delete operation के साथ index table को भी update होना पड़ता है जिसके कारण समय अधिक लग जाता है |
  - Index storage की जगह घेरती हैं जिसके कारण जैसे जैसे डाटा की संख्या या column की संख्या बढ़ती है तो जगह भी ज्यादा घिरती है |

अतः यह सलाह दी जाती है की जब ज्यादा ज़रुरत हो तभी index का प्रयोग करना चाहिए|

धन्यवाद

और अधिक पाठ्य-सामग्री हेतु निम्न लिंक पर क्लिक करें -

www.pythontrends.wordpress.com