

MySQL

Lecture 2 ,3

by
T.K.Malwatta
Senior Lecturer/HOD –ICT
University of Vocational Technology

Pattern Matching

MySQL provides standard SQL pattern matching as well as a form of pattern matching based on extended regular expressions similar to those used by Unix utilities

SQL pattern matching enables you to use “_” to match any single character and “%” to match an arbitrary number of characters (including zero characters). In MySQL, SQL patterns are case-insensitive by default. Some examples are shown here. You do not use = or <> when you use SQL patterns; use the LIKE or NOT LIKE comparison operators instead. LIKE or NOT LIKE statements to match field values using wildcards. The % sign is used for denoting wildcards and can represent multiple characters.

To find names beginning with “N”:

```
Mysql > select * from Emp where fname like 'N%';
```

```
Mysql > select * from Emp where fname not like 'N%';
```

To find names ending with “al”:

```
Mysql > select * from Emp where fname like '%al';
```

To find names containing a “w”:

```
Mysql > select * from Emp where fname like '%w%';
```

To find names containing exactly five characters, use five instances of the “_” pattern character:

2 Mysql > select * from Emp where fname like '_____';

Change table headings

```
ALTER TABLE table_name CHANGE [COLUMN]  
old_column_name new_column_name data_definition;
```

```
mysql> alter table emp change column city town varchar(20);
```

Add column & data to the existing table

```
mysql> alter table emp add salary int;  
mysql> update emp set salary='20000' where fname='ajith';
```

Mathematical functions in MySql (Counting Rows, Min, Max, Sum)

Counting the total number of employees. COUNT(*) counts the number of rows, so the query to count your employees looks like this:

```
mysql> select count(*) from emp;
```

GROUP BY command will create groups in the field name specified and will count the number of records in the groups

```
mysql> select city , count(*) from emp group by city; // count No. of cities in the table  
mysql> select sex , count(*) from emp group by sex;  
mysql> select count(fname) from emp;
```

```
mysql> select city,sex, count(*) from emp group by city,sex;
```

city	sex	count(*)
Colombo	M	1
Galle	M	1
Maharagama	F	1
Moratuwa	M	2

```
4 rows in set (0.00 sec)
```

```
mysql>
```

```
mysql>select min(salary) from emp;
```

```
mysql>select max(salary) from emp;
```

```
mysql>select sum(salary) from emp
```

We can use average, floor(highest integer value) , Mod functions also.

BETWEEN

This conditional statement is used to select data where a certain related constraint falls between a certain range of values. The following example illustrates it's use.

```
mysql> select fname,lname from emp where salary between 20000 and 40000;
```

IN

```
Mysql> select fname,lname from emp where city IN('Colombo','Galle');
```

Rename Table

```
Alter table emp rename as emp1;
```

Limit

As your tables grow, you'll find a need to display only a subset of data. This can be achieved with the LIMIT clause. For example, to list only the names of first 3 employees in our table, we use LIMIT with 3 as argument.

```
mysql >select fname,lname from emp limit 3;
```

These are the first 3 entries in our table.

You can couple LIMIT with ORDER BY. Thus, the following displays the 4 most senior employees.

```
mysql >select fname,lname,age from emp order by age desc limit 4;
```

fname	lname	age
ajith	priyantha	38
Kanthi	wass	34
kanal	priyantha	28
piyal	silva	27

4 rows in set (0.00 sec)

```
mysql>
```

Extracting Subsets

Limit can also be used to extract a subset of data by providing an additional argument. The general form of this LIMIT is:

SELECT (whatever) from table LIMIT starting row, Number to extract;

```
mysql >select fname,lname,age from emp limit 2,3;
```

Delete Null values from tables

```
mysql >delete from emp1 where fname is null;
```

Create a Mysql table with a primary key

Primary key uniquely identify a row in a table. One or more columns may be identified as the primary key. The values in a single column used as the primary key must be unique

```
CREATE TABLE test  
(  
  id int(10),  
  name varchar(40),  
  birthdate DATE,  
  PRIMARY KEY (id)  
);
```

```
CREATE TABLE test  
(  
  id int(10) primary key,  
  name varchar(40),  
  birthdate DATE  
);
```

Additional columns can be identified as part of the primary key with a comma separated list in the PRIMARY KEY command, like PRIMARY KEY (contact_id, name).

```
mysql> create table test(  
-> id int(10),  
-> name varchar(30),  
-> birthdate DATE,  
-> primary key(id)  
-> );  
Query OK, 0 rows affected (0.11 sec)  
  
mysql> describe test;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| id    | int(10) | NO | PRI | 0 | |  
| name  | varchar(30) | YES | | NULL | |  
| birthdate | date | YES | | NULL | |  
+-----+-----+-----+-----+-----+-----+  
3 rows in set (0.02 sec)  
  
mysql>
```

Auto Increment ID fields

```
CREATE TABLE stud  
(id int not null auto_increment primary key,  
name CHAR(30) ) ;
```

“**not null**” argument, which means that an application using this database will throw an error if you try to insert data into a row in this table and leave the value for this field blank.

auto_increment: When MySQL comes across a column with an auto_increment attribute, it generates a new value that is one greater than the largest value in the column. Thus, we don't need to supply values for this column, MySQL generates it for us! Also, it follows that each value in this column would be unique.

primary key: helps in indexing the column that help in faster searches. Each value has to be unique.

It is possible to write the INSERT INTO statement in two forms.

```
INSERT INTO table_name  
VALUES (value1, value2, value3,...)
```

Or

```
INSERT INTO table_name (column1, column2,col3.)  
VALUES (value1, value2, value3,...)
```

id	name	marks	address	phone
1	David	56	Malabe	2567442
2	Nicol	34	Galle	4866322
3	Dilan	56	Matara	2784335
4	Rayan	78	Kalaniya	4877284
5	Marina	34	Dehiwala	2894343

Table Joins in Mysql

Basically it is the combining of two rows based on the comparative values in selected columns.

There are 4 types of joins

1. Outer joint (Cross joint)
2. Inner joint
3. Left joint
4. Right joint

Cross Joint

tables involves combining rows from two tables. The most basic of join types is the cross-join. The cross-join simply assigns a row from one table to every row of the second table. This is of little or no use in real terms, but for the purposes of completeness, the syntax for a cross-join is as follows:

Syntax

```
SELECT <column_name>  
FROM <table1>, <table2>
```

To give an example lets look at two simple tables.

Table 1- product

prod_id	prod_name	prod_desc	sup_id
1	mouse	genius	3
2	cd	writable	3
3	harddisk	10gb	4
4	monitor	panasonic	1
5	keyboard	panasonic	2

Table2-suppliers

sup_id	sup_name	cup_address	contact_person
1	IBM	Colombo 5	Nimal
2	PC-House	Ratnapura	Kamal
3	Autodesk	Galle	Piyal
4	Microsoft	Matara	Tharaka

To find the suppliers of each product ,here we use cross joint

-select prod_name,sup_name from suppliers,product; or

-select prod_name,sup_name from suppliers join product;



The screenshot shows the MySQL Command Line Client window. The title bar is blue with the text "MySQL Command Line Client". The window displays the result of a cross join query between the 'suppliers' and 'product' tables. The output is a table with two columns: 'prod_name' and 'sup_name'. The data is as follows:

prod_name	sup_name
mouse	IBM
mouse	PC-House
mouse	Autodesk
mouse	Microsoft
cd	IBM
cd	PC-House
cd	Autodesk
cd	Microsoft
harddisk	IBM
harddisk	PC-House
harddisk	Autodesk
harddisk	Microsoft
monitor	IBM
monitor	PC-House
monitor	Autodesk
monitor	Microsoft
keyboard	IBM
keyboard	PC-House
keyboard	Autodesk
keyboard	Microsoft

At the bottom of the window, it says "20 rows in set (0.01 sec)".

Inner Joint

Inner joint (Equi join) rows from one or more tables based on comparison between a specific column in each table

Syntax

SELECT column_names FROM table1, table2 WHERE (table1.column = table2.column);

For example, to extract the product name and supplier name for each row in our product table we would use the following command

SELECT prod_name, sup_name, sup_address FROM product, suppliers WHERE (product.sup_id = suppliers.sup_id);

Note that we have to use what is known as the *fully qualified name* for the supplier_id column in each table since both tables contain a *supplier_id*. A fully qualified column name is defined by specifying the table name followed by a dot (.) and then the column name. They return only the records that satisfy the join requirements

The result of the above command is to produces a list of products and the name and address of the supplier for each product:

where (product.sup_id = suppliers.sup_id)

prod_name	sup_name	sup_address
monitor	IBM	Colombo 5
keyboard	PC-House	Ratnapura
mouse	Autodesk	Galle
cd	Autodesk	Galle
harddisk	Microsoft	Matara

Left Join

Another way to join tables is use a LEFT JOIN in the select statement. The LEFT JOIN causes the tables to be joined before any WHERE clause is used. The syntax for this type of join is:

SELECT column names FROM table1 LEFT JOIN table2 ON (table1.column = table2.column);

Therefore, we can perform a LEFT JOIN that gives us the same result as our Equi-Join:

**SELECT prod_name, sup_name, sup_address FROM product LEFT JOIN suppliers
ON (product.sup_id = suppliers.sup_id);**

One key different with the LEFT JOIN is that it will also list rows from the first table for which there is no match in the second table. For example, suppose we have product in our *product* table for which there is no matching supplier in the *supplier* table. When we run our SELECT statement the row will still be displayed, but with NULL values for the supplier columns since no such supplier exists:

```
+-----+-----+-----+
| prod_name | sup_name | cup_address |
+-----+-----+-----+
| mouse     | Autodesk | Galle       |
| cd        | Autodesk | Galle       |
| harddisk  | Microsoft| Matara      |
| monitor   | IBM      | Colombo 5   |
| keyboard  | PC-House | Ratnapura    |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Right Join

The opposite effect can be achieved using a RIGHT JOIN, whereby all the rows in a the second table (i.e. our supplier table) will be displayed regardless of whether that supplier has any products in our product table:

**SELECT prod_name, sup_name, sup_address FROM product RIGHT JOIN suppliers
ON (product.sup_id = suppliers.sup_id);**

```
+-----+-----+-----+
| prod_name | sup_name | cup_address |
+-----+-----+-----+
| monitor   | IBM      | Colombo 5   |
| keyboard  | PC-House | Ratnapura    |
| mouse     | Autodesk | Galle       |
| cd        | Autodesk | Galle       |
| harddisk  | Microsoft| Matara      |
+-----+-----+-----+
5 rows in set (0.00 sec)
```