Chapter 8:

MySQL – Revision Tour

Informatics Practices

Class XII (CBSE Board)

Revised as per CBSE Curriculum 2015

"Open Teaching-Learning Material"

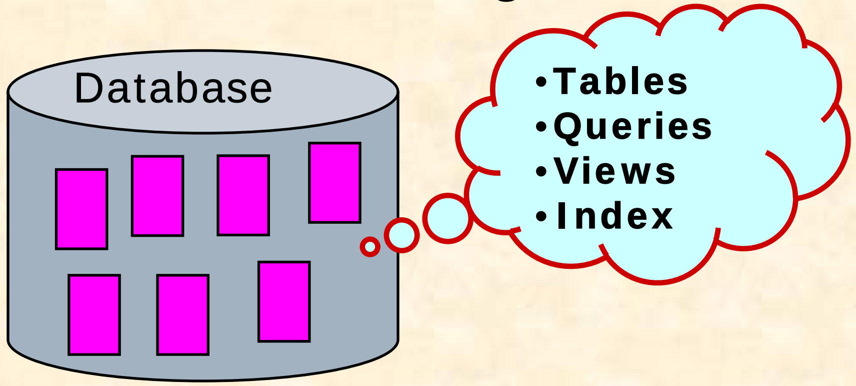
Visit www.ip4you.blogspot.com for more….

Authored By:- Rajesh Kumar Mishra, PGT (Comp.Sc.)
Kendriya Vidyalaya Upper Camp, Dehradun (Uttarakhand)
e-mail : rkmalld@gmail.com

What is the Database?

- A database is an organized collection of interrelated data stored together to serve applications. It work like a container which may contains the various database objects.
- Most of the databases stores data in the form of Relations (also called Tables). Such Database are known as Relational Database.
- A Software used to manage Relational database is called **RDBMS** (Relational Database Management System).

Some Commonly used RDBMS software are- **Oracle**, **MySQL**, **MS SQL Server**, **SyBase** and **Ingress** etc.



Why Database System is used? (Advantages)

- Databases reduces **Redundancy**

  It removes duplication of data because data are kept at one place and all the application refers to the centrally maintained database.
- Database controls **Inconsistency**

  When two copies of the same data do not agree to each other, then it is called Inconsistency. By controlling redundancy, the inconsistency is also controlled.
- Database facilitate **Sharing of Data**

  Data stored in the database can be shared among several users.
- Database ensures **Security**

  Data are protected against accidental or intentional disclosure to unauthorized person or unauthorized modification.
- Database maintains **Integrity**

  It enforces certain integrity rules to insure the validity or correctness of data. For ex. A date can't be like 25/25/2000.

Data Model

Data model describes 'How data is organized or stored' in the database. It may be-

- **Relational Data Model**

  In this model data is organized into **Relations** or **Tables** (i.e. Rows and Columns). A row in a table represents a relationship of data to each other and also called a **Tuple** or **Record**. A column is called **Attribute** or **Field.**

- **Network Data Model**

  In this model, data is represented by collection of records and relationship among data is shown by **Links**.

- **Hierarchical Data Model**

  In this model, Records are organized as **Trees**. Records at top level is called Root record and this may contains multiple directly linked children records.

- **Object Oriented Data Model**

  In this model, records are represented as a objects. The collection of similar types of object is called class.

Data Models

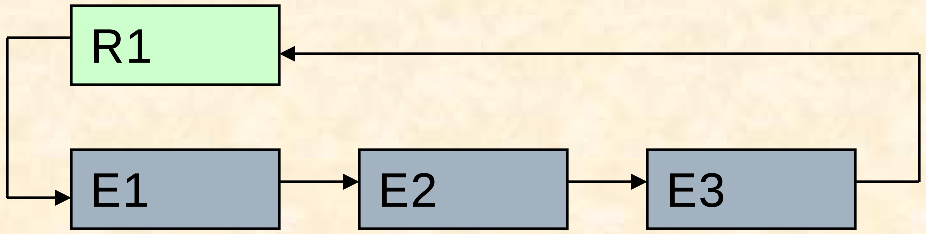
Attribute (Field)

Table (Relation)

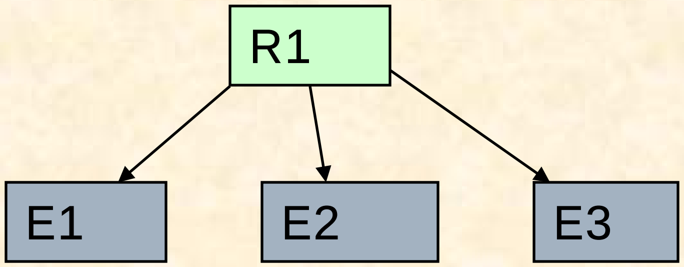
Entity (Record)

Name	Address	DOB	City	Phone
Amar	2/3 Chowk	01.04.1990	Kanpur	12345
Kailash	22 Katra	23.10.1992	Lucknow	67890

Relational Model



Network Model



Hierarchical Model

Representation of Records and Relationship in various Data Model

RDBMS Terminology

- **Relation (Table)**
  A Relation or Table is Two-Dimensional (Matrix) like structure arranged in Rows and Columns. It has the following properties-
  - Column homogeneous - All items in a column are of same data type.
  - Each column assigned a unique name and must have atomic (indivisible) value.
  - All rows of a relation are distinct i.e. no two identical rows (records) can exist in the Relation.
  - Ordering or Rows (Records) or Columns (fields) are immaterial.
- **Domain**
  It is collection (set) of possible values from which the value for a column is derived.
- **Tuple/ Entity/ Record :**
  A Row of a table is called Tuple or Record.
- **Attribute/ Field:**
  Column of a table is called Attribute or Field.
- **Degree :** Number of columns (attributes) in a table.
- **Cardinality :** Number of Records in a table.

Concept of Keys

In a Relation, each record must be unique i.e. no two identical records are allowed in the Database. A column or combination of column which identifies a record called Key of the Table. A key attribute must have unique (non-repeatable) value.

- **Primary Key**

  A set of one or more column that can uniquely identify a record in the relation is called Primary Key.

- **Candidate Key**

  A Column or group of columns which can be used as primary key are called Candidate keys, as they are candidate to become as Primary key.

- **Alternate Key**

  A Candidate Key that is not a Primary key is called Alternate key.

- **Foreign Key**

  A non-key column whose values are derived from the primary key of some other table is called Foreign key.

Introduction to MySQL

MySQL is an Open Source, Fast and Reliable Relational Database Management System (RDBMS) . It is alternative to many of the commercial RDBMS. The main features of MySQL are-

- **Open Source & Free of Cost:**

  It is Open Source and available free of cost. It is part of LAMP (Linux, Apache, MySQL, PHP/ Perl/ Python) Open Source group.

- **Portability:**

  It can be installed and run on any types of Hardware and OS like Linux, MS Windows or Mac etc.

- **Security :**

  It offers privilege and password system for authorization.

- **Connectivity**

  It may connect various types of client using different protocols and Programming Languages .

- **Query Language**

  It uses SQL (Structured Query Language) as query language, which is standardized by ANSI.

Types of SQL Commands

MySQL follows SQL specifications for its commands . These SQL commands can be categorized as -

- **Data Definition Language (DDL)**

  These SQL commands are used to create, alter and delete database objects like table, views, index etc.

  Example : CREATE , ALTER , DROP etc.

- **Data Manipulation Language (DML)**

  These commands are used to insert, delete, update and retrieve the stored records from the table.

  Ex. SELECT…., INSERT…, DELETE…, UPDATE…. etc.

- **Transaction Control Language (TCL)**

  These commands are used to control the transaction.

  Ex. COMMIT, ROLLBACK, SAVEPOINT etc.

- **Data Control Language (DCL)**

  These commands are used to manipulate permissions or access rights to the tables etc.

  Ex. GRANT , REVOKE etc.

Database Handling commands in MySQL

- **Creating a Database.**

  The following command will create School database in MySQL.

  mysql> **CREATE DATABASE School;**

- **Opening a database**

  To open an existing database, following command is used.

  mysql> **USE school ;**

- **Getting listings of database and tables**

  mysql> **SHOW DATABASES;**

  mysql> **SHOW TABLES;**

- **Deleting a Database and Table**

  mysql> **DROP DATABASE School;**

  mysql> **DROP TABLE Student;**

- **Viewing Table Structure**

  mysql> **DESCRIBE Student;**

Select database();
Shows the name of currently open database

Data type in MySQL

- **Numeric Data Types:**
  - **INTEGER or INT** – up to 11 digit number without decimal.
  - **SMALLINT** – up to 5 digit number without decimal.
  - **FLOAT (M,D) or DECIMAL(M,D) or NUMERIC(M,D)**

    Stores Real numbers upto **M** digit length (including .) with **D** decimal places.

    e.g. Float (10,2) can store 1234567.89
- **Date & Time Data Types:**
  - **DATE** - Stores date in YYYY-MM-DD format.
  - **TIME** - Stores time in HH:MM:SS format.
- **String or Text Data Type:**
  - **CHAR(Size)**

    A fixed length string up to 255 characters. (default is 1)
  - **VARCHAR(Size)**

    A variable length string up to 255 characters.

Char, **Varchar**, **Date** and **Time** values should be enclosed with single (' ') or double ("") quotes in MySQL.

Creating Tables

- **Creating Simple Tables:**

  CREATE TABLE < Table Name>

  (<Col name1><data type>[(size)][Constraints],….);

  Data types- INTEGER, NUMERIC(P,D), CHAR(n), VARCHAR(n), DATE etc.

```
mysql> CREATE TABLE Emp
      (empID integer,
       ename char(30),
       city char(25),
       pay decimal(10,2));
```

Emp			
empID	ename	city	pay

- **Creating Table from Existing Table:**

  CREATE TABLE <Table name> [AS] (<Select Query>);

```
CREATE TABLE Staff  ( Select empID, ename, pay  From Emp);
CREATE TABLE Staff AS ( Select * From Emp);
```

Staff table will be identical to Emp table.

Making Simple Queries Using SELECT

The SELECT command of SQL, empower you to make a request (queries) to retrieve stored records from the database.

The syntax of SQL is given below-

```
SELECT < [Distinct | ALL] *| column name(s)>
FROM <table(s)>
WHERE <condition>
ORDER BY <column name> [ASC | DESC] ;
```

Consider the table ***Student*** having some records as –

StID	Name	Fname	DOB	City	Class
S1	Amitabh	Harivansh Rai	1948-11-10	Allahabad	12
S2	Sharukh	Firoz	1970-05-10	Delhi	11
S3	Irphan	Akbar	1970-10-05	Jaipur	11
S4	Salman	Salim Javed	1972-04-10	Mumbai	10
S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

Making Simple Queries – Cont..

- **Selecting all columns**

If you want to view all columns of the student table, then you should give the following command-

```
mysql> SELECT * FROM Student ;
```

MySQL will display the all records with all columns in the Student table.

* Is used to represent all columns.

StID	Name	Fname	DOB	City	Class
S1	Amitabh	Harivansh Rai	1948-11-10	Allahabad	12
S2	Sharukh	Firoz	1970-05-10	Delhi	11
S3	Irphan	Akbar	1970-10-05	Jaipur	11
S4	Salman	Salim Javed	1972-04-10	Mumbai	10
S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

Making Simple Queries – Cont..

☐ Selecting columns

If you want to view only **Name** and **City** columns of the student table

mysql> **SELECT Name, City FROM Student ;**

Name	City
Amitabh	Allahabad
Sharukh	Delhi
Irphan	Jaipur
Salman	Mumbai
Abhishek	Mumbai

mysql> **SELECT City, Name FROM Student ;**

City	Name
Allahabad	Amitabh
Delhi	Sharukh
Jaipur	Irphan
Mumbai	Salman
Mumbai	Abhishek

Making Simple Queries – Cont..

- **Eliminating Duplicate values in a column - DISTINCT**

```
mysql> SELECT City  FROM Student ;
```

City
Allahabad
Delhi
Jaipur
Mumbai
Mumbai

Mumbai is repeated

```
mysql> SELECT DISTINCT City FROM Student ;
```

City
Allahabad
Delhi
Jaipur
Mumbai

Only Unique Cities are displayed

Making Simple Queries – Cont..

- **Doing simple calculations**

We can also perform simple calculations with SQL Select command. SQL provide a dummy table named DUAL, which can be used for this purpose.

```
mysql> SELECT 4*3 ;
```

We can also extend this idea with a columns of the existing table.

```
mysql> SELECT Name, Sal *12 FROM EMP ;
```

- **Using Column Aliases**

We can give a different name to a column or expression (Alias) in the output of a query.

Alias for Sal*12

```
mysql> SELECT Name, Sal*12 AS ‘Annual Salary’ FROM EMP;
mysql> SELECT Name, DOB AS ‘Date of Birth’ FROM Student;
mysql> SELECT 22/7 AS PI FROM Dual;
```

When Alias name is a single word then ‘ ‘ is not required.

Selecting Specific Rows – WHERE clause

- **WHERE <Condition>**

  We can select specific records by specifying condition with WHERE clause.

```
mysql> SELECT * FROM Student WHERE City='Mumbai';
```

StID	Name	Fname	DOB	City	Class
S4	Salman	Salim Javed	1972-04-10	Mumbai	10
S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

```
mysql> SELECT Name, Fname, City from Student
       WHERE Class >10;
```

Condition

Name	Fname	City	Class
Amitabh	Harivansh Rai	Allahabad	12
Sharukh	Firoz	Delhi	11
Irphan	Akbar	Jaipur	11

Selecting Specific Rows – WHERE clause

- **Relational Operators**

  We can use the following Relational operators in condition.

  =, > , < , >=, <=, <>, IS , LIKE, IN, BETWEEN

- **Logical Operators**

  We can use the following Logical Operators to connect two conditions.

  OR , AND , NOT (!)

```
mysql> SELECT Name, City from Student
       WHERE City <> 'Mumbai' AND Class>10;

mysql> SELECT * FROM Emp
       WHERE Sal >10000 OR Job ='Manager';

mysql> SELECT * FROM Student
       WHERE NOT Grade='A';
```

Selecting Specific Rows – WHERE clause

- **Specifying Range of Values – BETWEEN Operator**

```
    mysql> SELECT * FROM Emp
           WHERE Sal BETWEEN 5000 AND 10000 ;
```

The same query can also be written as -

```
    mysql> SELECT * FROM Emp
           WHERE Sal >= 5000 AND Sal<=10000 ;
```

Other Logical operators also can be applied-

```
      mysql> SELECT * FROM Emp
             WHERE  NOT Sal BETWEEN 5000 AND 10000 ;
```

- **Specifying List – IN Operator**

```
    mysql> SELECT * FROM Emp
           WHERE Sal IN (5000, 10000) ;
```

The same query can also be written as -

```
    mysql> SELECT * FROM Emp
           WHERE Sal = 5000 OR Sal =10000 ;

    mysql> SELECT * FROM Student
             WHERE  City IN (‘Mumbai’, ‘Delhi’,’Kanpur’) ;
```

Selecting Specific Rows – WHERE clause

- **Pattern Matching – LIKE Operator**

A string pattern can be used in SQL using the following wild card

- **% Represents a substring in any length**
- **_ Represents a single character**

Example.

'A%'	represents any string starting with 'A' character.
'_ _A'	represents any 3 character string ending with 'A'.
'_B%'	represents any string having second character 'B'
'_ _ _'	represents any 3 letter string.

A pattern is case sensitive and can be used with LIKE operator.

```
mysql> SELECT * FROM Student WHERE Name LIKE 'A%';
mysql> SELECT * FROM Student WHERE Name LIKE '%Singh%';
mysql> SELECT Name, City FROM Student
       WHERE Class>=9 AND Name LIKE '%Kumar%' ;
```

Selecting Specific Rows – WHERE clause

❑ Searching NULL Values – IS Operator

mysql> **SELECT * FROM Student WHERE City IS NULL ;**

The **NOT** Operator can also be applied -

mysql> **SELECT * FROM Student WHERE City IS NOT NULL;**

❑ Ordering Query Result – ORDER BY Clause

A query result can be orders in ascending (A-Z) or descending (Z-A) order as per any column. <u>Default is Ascending order.</u>

mysql> **SELECT * FROM Student ORDER BY City;**

To get descending order use **DESC** key word.

mysql> **SELECT * FROM Student ORDER BY City DESC;**

mysql> **SELECT Name, Fname, City FROM Student**
Where Name LIKE 'R%' ORDER BY Class;

Inserting Records in a Table

You can insert record in the table by using by using the following DML command.

INSERT INTO <Table Name> [<Column list>]

VALUES <list of values>

Suppose a table named STUDENT has been created with the following structure.

StID	NAME	FNAME	DOB	CITY	CLASS

We can insert a record as follows-

```
mysql> INSERT INTO Student VALUES
       ('s1','Amitabh', 'Harivansh','1955-10-25', 'Mumbai', 12);
 mysql> INSERT INTO Student VALUES
       ('s2','Sharukh Khan', NULL,'1972-5-25', 'Delhi', 10);
mysql> INSERT INTO Student (StID, FName, Name, Class)
       VALUES ('s3','Amitabh', 'Abhishek', 10);
```

Inserting Records from Other Table

You can insert all or selected record(s) in the table from another table by using Select … command in place of Values.

Suppose a table named NEWSTUDENT has been created and records to be inserted from OLDSTUDENT table having the same structure of columns.

```
mysql> INSERT INTO Newstudent VALUES
       (SELECET * FROM Oldstudent);
```

Both tables must have same structure

```
 mysql>INSERT INTO Newstudent VALUES
       (SELECT * FROM Oldstudent WHERE City=‘Mumbai’);
mysql> INSERT INTO Newstudent (StID, Name, Class)
       VALUES (Select StID, Name,Class FROM Oldstudent
       WHERE Class>=11);
```

Deleting Records from the Table

You can delete all or selected record(s) from the table by using the following DML command.

DELETE FROM <Table Name> [WHERE <Condition>]

```
mysql> DELETE FROM Student ;
```

This command will delete all records...

```
mysql> DELETE FROM Student WHERE  City='Mumbai' ;
mysql> DELETE FROM Student WHERE Class >=11 ;
```

- **You can recall (Undelete) records by giving ROLLBACK command.**

```
mysql> ROLLBACK ;
```

- **You can issue COMMIT command to record the changes permanently.**

```
mysql> COMMIT;
```

Modifying Records in the Table

You can modify the values of columns of all or selected records in the table by using the following DML command.

UPDATE <Table Name> SET <Column> = <Expression> [WHERE <Condition>]

```
mysql> UPDATE Student SET Class =10  ;

mysql> UPDATE Student SET FName= CONACT(‘Mr.’, FName’) ;

mysql> UPDATE Emp SET Sal = Sal+(Sal*10/100);

mysql> UPDATE Emp SET Sal = Sal+(Sal*10/100)
       WHERE Sal <=10000;

mysql> UPDATE Emp SET City = ‘Dehradun’
       WHERE CITY IS NULL;
```

Working with Functions

- **What is Function?**

  A function is a special types of command that performs some operation and returns a single value as a result.

  It is similar to method or function in JAVA, which can be called by giving some argument.

- **Types of Functions:**
  - Numeric Functions
  - String Functions
  - Date & Time Function
  - Aggregate Functions

Numeric Functions

These functions may accept some numeric values and performing required operation, returns numeric values as result.

Name	Purpose	Example
MOD (M, N)	Returns remainder of M divide by N	Select MOD(11,4) ; ➔ **3**
POWER (M, N) POW (M, N)	Returns M^N	Select POWER(3,2); ➔ **9**
ROUND (N [,M])	Returns a number rounded off up to M place. If M is -1, it rounds nearest 10. If M is not given, the N is rounded to the nearest Integer.	Select ROUND(15.193,1); ➔**15.2** Select ROUND(15.193); ➔**15**
SQRT (N)	Returns square root of N	Select SQRT(25); ➔ **5**
TRUNCATE(N,M)	Returns number after truncating M decimal place.	Select TRUNCATE(15.79,1) ➔ **15.7**

String Functions

Name	Purpose	Example
CONCAT(str1,str2)	Returns concatenated string i.e. str1+str2.	Select CONCAT(Name, City) from Student;
LOWER(str) / LCASE(str)	Returns the given string in lower case.	Select LOWER('ABC'); → abc
UPPER(str) / UCASE(str)	Returns the given String in upper case.	Select UPPER('abc'); → ABC
LTRIM(str) RTRIM(str) TRIM(str)	Removes Leading/Trailing/both spaces from given string.	Select TRIM(' ABC '); → 'ABC'
LEFT(str, N) RIGHT(str,N)	Returns the (N) characters from left/right from the given string.	Select LEFT('Computer',4); → Comp
SUBSTR(str,P,[N]) / MID (str,P,N)	Returns the substring for given position(P) and length (N). If M is (-ve) then backward position counted.	Select SUBSTR('Computer',3,2); → mp
INSTR(str1,str2)	Returns the index of first occurrence of str2 in str1.	Select INSTR('Common', 'm'); →3
LENGTH(str)	Returns the length of given string	Select LENGTH('Common'); →6

Date & Time Functions

Name	Purpose	Example
CURDATE() / CURRENT_DATE()	Returns the current date in **YYYY-MM-DD** format.	Select CURDATE(); ➔ 2013-10-02
NOW()	Returns the current date & Time as YYYY-MM-DD HH:MM:SS	Select NOW(); ➔ 2013-10-02 11:30:02
SYSDATE()	Returns the current date & Time as YYYY-MM-DD HH:MM:SS	Select SYSDATE(); ➔ 2013-10-02 11:30:10
DATE()	Returns the date part of a date-time expression.	Select DATE(SYSDATE()); ➔ 2013-10-02
MONTH() YEAR()	Returns the Month/Year from given date argument.	Select MONTH('2012-10-02'); ➔ 10
DAYNAME()	Returns the name of the weekday	Select DAYNAME(CURDATE()); ➔ SUNDAY
DAYOFMONTH()	Returns the day of month (1-31).	Select DAYOFMONTH(CURDATE());
DAYOFWEEK()	Returns the day of week (1-7).	Select DAYOFWEEK(CURDATE());
DAYOFYEAR()	Returns the day of year(1-366).	Select DAYOFYEAR(CURDATE());

Aggregate Functions

Name	Purpose	Example

SUM()	Returns the sum of given column.	Select SUM(Pay) from Emp;
		Select Sum(Pay), Sum(Net) from Emp;

MIN()	Returns the minimum value in the given column.	Select MIN(Pay) from Emp;
MAX()	Returns the maximum value in the given column.	Select MAX(Pay) from Emp;
AVG()	Returns the Average value of the given column.	Select AVG(Pay) from Emp;

COUNT()	Returns the total number of values/ records in given column.	Select COUNT(Name) from Emp;
		Select COUNT(*) from Emp;

Aggregate Functions should not be used with other columns which may have multiple values in the table. The following query is <u>illogical and wrong</u>. Why? Think yourself....

```
Select sum(pay), name from Employee;
```

Modifying Table Structure

You can alter (modify) the structure of existing table by the using ALTER TABLE.... Command of MySQL.

You can do the following with the help of ALTER TABLE.. Command.

- **Add a new Column or Constraints**
- **Modifying existing column (name, data type, size etc.)**
- **Delete an existing column or Constraints**
- **Changing Column Name**

ALTER TABLE <Table Name>

ADD|MODIFY|DROP|CHANGE <Column Definition(s)>

You can add/Delete/Modify multiple columns with single ALTER Command.

Modifying Table Structure cont..

- **Adding new column**

  ALTER TABLE <Table Name>
  ADD <Column>[<data type> <size>][<Constraints>]

```
mysql> ALTER TABLE Student ADD (TelNo Integer);
mysql> ALTER TABLE Student ADD (Age Integer DEFAUL 10);
```

- **Modifying Existing Column**

  ALTER TABLE <Table Name>
  MODIFY <Column>[<data type> <size>] [<Constraints>]

```
mysql> ALTER TABLE Student MODIFY Name VARCHAR(40);
mysql> ALTER TABLE Emp MODIFY (Sal DECIMAL (10,2));
```

- **Removing Column & Constraints**

  ALTER TABLE <Table Name>
  DROP <Column name> |<Constraints>

```
mysql> ALTER TABLE Student DROP TelNo;
mysql> ALTER TABLE Emp DROP JOB, DROP Pay;
```