Chapter- 6

DATABASE MANAGEMENT SYSTEM (DBMS)

The database can normally be described as a repository for data. A database is a collection of any related data. A database is a collection of seamless, logically consistent, naturally meaningful and real data.

Database is a collection of related data. Data is a collection of facts and figures that can be processed for providing information. Most data represent recordable facts. Data is helpful in providing information that is based on facts.

If we have the marks obtained by all the students, then we can extract the list of toppers and other desired information.Data is stored by the data management system in such a way that the information can easily be obtained and updated.

6.1 Data Base Management System (DBMS)

Data base management system is a software that has the functions of database storage, access, security, backup and such other features. Examples of some commonly used DBMS are: MySQL, Postgres SQL, SQL Server, Oracle etc.

DBMS is a collection of programs that enables the users to create and maintain the database. This software system gives us the following main features:

Defining: Facility for specifying type of data, structure of data and the constrains for the data being stored.

Constructing: Facility for storing the data in a storage device.

Manipulating: Facility for retrieving and updating the data present in the database and preparing reports.

6.2 Database Abstraction

Database system provides available data to users according to their requirement and hides the details about how data is stored and managed in the hardware. This concept/process is known as database abstraction.

6.2.1 DBMS Architecture

Data base management systems is described by three-level schemas (schema architecture). There are three layers (levels), which are as follow:

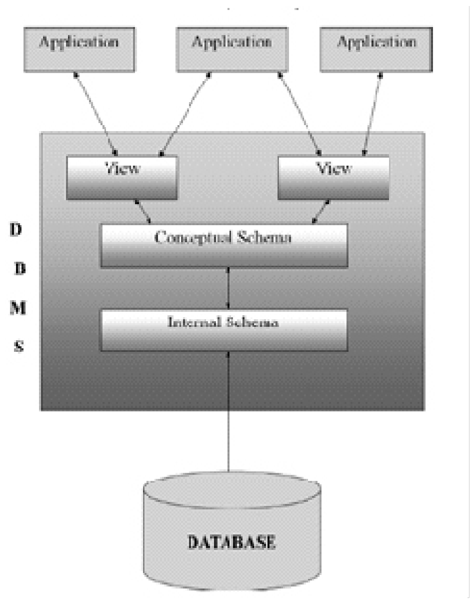


Fig. 6.1 Database Management System Architecture

1. PHYSICAL LEVEL or INTERNAL LEVEL

It denotes how the data is stored in the database. It is the lowest level of abstraction.

2. CONCEPTUAL or LOGICAL LEVEL

Conceptual abstraction is the next higher level of abstraction. This level represents what stored data and their relationship in a database. It is the job of database manager.

3. EXTERNAL or VIEW LEVEL

Most of the database users do not use the complete database. It means that partial data is used by them. This external level is used for this purpose. It is the highest level of abstraction. What is needed by a user, or on what part of data the user is working, only that part of the database is visible to the user. In this way, there can be many views of one database.

Database can be categorized on different criterion. Database Management System is categorized on different basis, which are as follow:

1. On the basis of Data Models

    1. Traditional Models: Relational, Network and Hierarchical Models

    2. Emerging Models: Object-oriented and Object-relational Models

2. On the basis of users

    Single and Multiple

3. On the basis of location

    Centrlised and Distributed databases

4. On the basis of purpose

6.2.2 Database System Structure

Databases system is divided into modules and each module has some or other responsibility of system control.

1.Storage Manager 2.Query Processor

1. Storage Manager

Storage manager is the module which makes interactions between low level data and application programs and queries. Storage manager is responsible for storing, retrieving and updating data in the database. It has following components.

1. Authorization and Integrity Manager:It tests the integrity constraints and checks the authorization of users to access data.

2. Transaction Manager: It ensures that no kind of change will be brought to the database until a transaction has been completed totally.

3. File Manager: It manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

4. Buffer Manager: It decides which data is in need to be cached in main memory and then fetch it up in main memory. This is very important as it defines the speed in which the database can be used.

The storage manager implements various data structures.

(1) Data Files - where the data is stored.

(2) Data Dictionary – where data about the data is stored.

(3) Indices –which help in accessing the data faster.

2. Query Processor

As query is very much necessary to find out only the data user need from loads of data of the database, query processor is very important to process these query requests. Query processor has following components.

1. DDL Interpreter:It interprets the DDL statements and records the definitions in data dictionary.

2. DML Compiler: It translates the DML statements in low-level instructions that the query evaluation understands. It also performs query optimization which actually picks up the lowest cost evaluation plan from various alternatives.

3. Query Evaluation Engine: It executes the low-level instructions compiled by the DML compiler.

6.3 Advantages of DBMS

1. No data redundancy and inconsistency: In an organization, it is quite normal to have a set of data being used by different departments for different purposes. When each department keep its own data, initially it is a copy of the same data. After some time, changesmay occur in the data which may not be updated everywhere. This leads to inconsistency. It increases the storage requirements and cost too. Data base management system saves us from this problem, because it reduces the issues of data redundancy.

2. Restricting Unauthorized Access and Security: All data present in the database should not be accessible to all users. Data Base Administrator restricts unauthorised access and provides security to the database.

3. Data Integrity: Security and integrity is maintained in th database. To maintain the integrity of data, some constraints or conditions are imposed.

4. Simple Access : Data is easily accessible in DBMS.

6.4 Characteristics of DBMS

Following are the characteristics of DBMS:

1. Most important characteristic of DBMS is that data redundancy can be controlled.
2. Data can be shared.
3. Data is more secure.
4. Better and fast processing of data.
5. Data is independent in DBMS.
6. Examples of Data Base Management System Applications

6.5 Examples of Data Base Management System Applications

Database is being in many organizations. Here are few examples:

1. In banking, database is used extensively. Any customer can do transactions at any branch of his/her bank. Earlier, when databases were not in use, customer had to go to his/her branch. Because of Database, customers are getting net-banking and such other facilities.
2. Airline reservation system is also based on database. Not only customer can book flight as per his/her convenience, but also the airlines employees can generate various reports which help in taking quick decisions.
3. In a university, information about teachers and students is stored in database. Details about various courses, how many students are studying in a course and other such details can be easily queried. Jobs related to examinations are also accomplished quickly.
4. Be it railway reservation or bus reservation, both are based on database. Passengers get the information quickly and at the same time, respective departments can get various reports as per their requirements.
5. Database makes it convenient to keep details of doctors, patients, various lab-test reports of patients, their medication, so that analysis-diagnosis of disease can take place easily.

6.6 Relational Data Base Management System

Relations are created and maintained in various tables stored in database system, this system is called Relational Data Base Management System (RDBMS). DBMS and RDBMS are used to physically store and maintain information in database. When the data increases in large amount, RDBMS is required for storing and managing in a better way. Relational data model contains information about indexes, keys, tables, and details about relationship among tables.

Edgar Frank Codd presented the principle of relational database in 1970s. Codd proposed a set of thirteen rules for relational model or principle. Relations among different data is the main requirement of relational model.

RDBMS can be considered as of next generation of DBMS. DBMS is used as a base model to store data in a relational database system. Although, instead of DBMS, RDBMS is used for complex business applications.

6.6.1 Entity Relationship Model (ER Model)

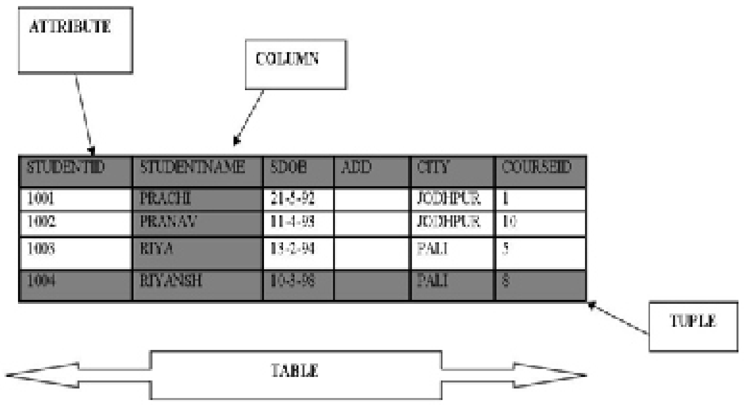
Entity relationship model is based on the concept of real-world entities and their relationships. While developing a database model of a real scenario, ER model creates entity set, relation set, normal attributes and constraints.

ER Model is used for conceptual design. It is based on

1. Entities and Attributes
2. Relationships among entities

Entity: In ER model, an entity has attributes which belong to real world. In each attribute, a set of its values is defined by domain.

For example, in a school database, student is an entity. Attributes of student can be name, age, course etc.



STUDENTID	STUDENTNAME	SDOB	ADD	CITY	COURSEID
1001	PRACHI	21-5-92		JODHPUR	1
1002	PRANAV	11-4-93		JODHPUR	10
1003	RIYA	13-2-94		PALI	5
1004	RIYANSH	10-3-98		PALI	8

Fig 6.2 Entity Relationship Model

Relationship: Logical connection among more than one entities is called relationship. Relations in entity are mapped with different methods. Mapping cardinalities provides the number of relations between two entities.

Following cardinalities are there:

One to One

One to Many

Many to One

Many to Many

One element of entity A is connected to maximum one element of entity B and one element of entity B is connected to maximum one element of entity A.

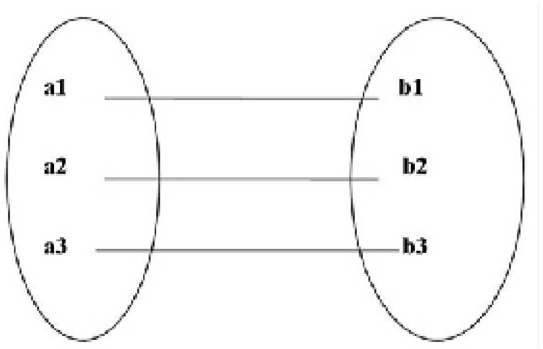


Fig. 6.3 One to One Relationship

One element of entity A may be connected to more than one element of entity B, but one element of entity B can be connected to maximum one element of entity A.

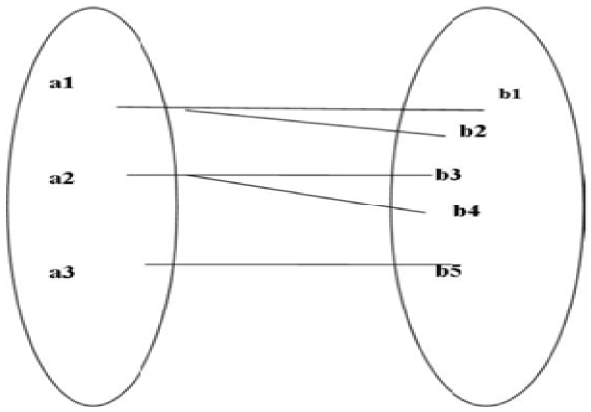


Fig. 6.4 One to Many Relationship

One element of entity A is connected to maximum one element of entity B, but one element of entity B may be connected to more than one element of entity A.

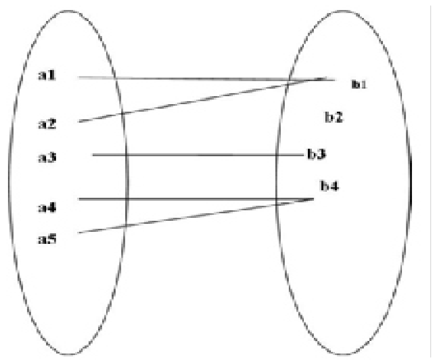


Fig. 6.5 Many to One Relationship

One element of entity A may be connected to more than one element of entity B and one element of entity B may be connected to more than one element of entity A.

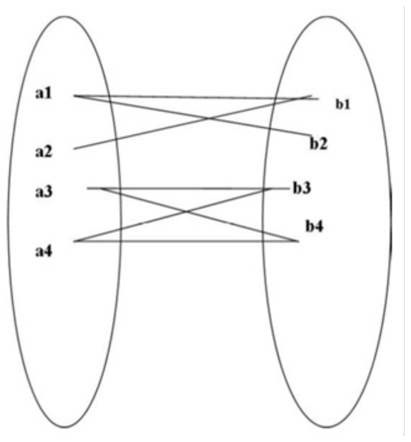


Fig. 6.6 Many to Many Relationship

Some important definitions

- Data: Raw facts and figures which are useful for an organization are called data. We can not make decisions based on data.

- Information: Well-processed data is called information.We can decide on the basis of information.

- ♦ Field/Attribute: It represents a particular element of data.It is also called data item
- ♦ Record:Collection of fields is called a record. A record can have fields of differentdatatypes
- ♦ File: The collection of similar types of records is called a file.
- ♦ Table:Group of rows and columns is called a table, in which useful data / information is stored. It is generally considered a passive unit which is stored on secondary device.
- ♦ Relation: Relation (collection of rows and columns) on which we can perform different operations.
- ♦ Database:It is a collection of logically related data.
- ♦ Tuple:A row in a relation is called a tuple.
- ♦ Domain:The database domain is the group of all acceptable values. Example: A field for gender may have permissible domain to have three values {male, female, unknown} in that column.

6.6.2 Keys

Any attribute in the table which uniquely identifies each record in the table is called key.

1. Primary Key

In a relational table, it is the first and foremost key which is used to uniquely identify a record. The primary key is of two types.

1. Simple primary key: The primary key which has one attribute is called simple primary key.
2. Composite primary key: The primary key which has more that one attribute is called composite primary key.

Defining primary key

1. The primary key is always unique.
2. There may be only one primary key in any table.

3. Composite primary key can have upto 16 fields combination for identifying a record uniquely.

2. Foreign Key

In relational database, a foreign key is a group of column(s) that provide link between the two or more tables. It is also known as referencing key. In other words, when a primary key of one table is used as a key in another table, it is called a foreign key. Foreign key provides the method for maintaining integrity in data.

3. Composite Primary Key

The composite primary key in the relational database table is a group of two or more columns that uniquely identifies each row in the table.

4. Super Key

Super Key is defined as a set of attributes within a table that uniquely identifies each record within a table. Super Key is a superset of Candidate key.

5. Candidate Key

Candidate keys are defined as the set of fields from which primary key can be selected. It is an attribute or set of attribute that can act as a primary key for a table to uniquely identify each record in that table.

6.7. SQL

Structured Query Language(SQL) is the language of database. Relational databases like MS-Access, MS-SQL Server and Oracle use SQL.

6.7.1 Database Languages

In any system, database languages are used to create and manage databases. Two types of languages are used in databases.

Data Definition Language (DDL)

DDL is used to define conceptual schema. It also provides information about how this schema is implemented in physical devices. The most important DDL satements in SQL are given below.

1. CREATE- To create objects in database

2. ALTER- To change in the structure of database

3. DROP- to remove objects from database

4. COMMENT- To put comment in data dictionary

5. RENAME- To rename the objects in database

Data Manipulation Language (DML)

DML is used for manipulating the data in database. Few of the statements used in DML are given below.

1. SELECT- To retrieve data from database.

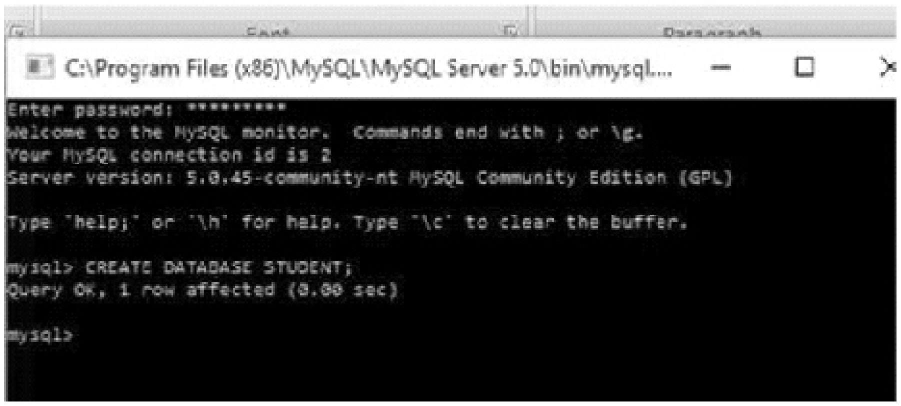
2. INSERT- To insert data in table.

3. UPDATE- To update the present data in a table.

4. DELETE- To delete records from a table.

6.8 MySQL

MySQL is adatabase management system which is used for managing relational databases. It is an open source software.



```
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE STUDENT;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Fig. 6.7 MySQL Interface

6.8.1 MySQL Commands

Create Database

There is a need to create a database before doing anything with the data. Database is a container of data. It stores details as per our requirements, for example about students, vendors, employees, customers or anything. Database in MySQL is a collection of tables, database views, triggers, stored procedures, etc., by which data is stored and manipulated. Following statement systaxis used to create a database in MySQL.

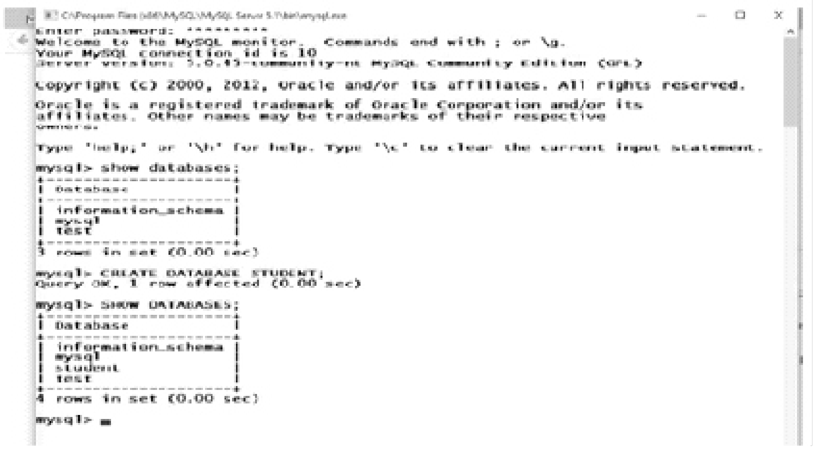
CRETAE DATABASE [IF NOT EXIST] [Name of Database]

'Name of Database' is the name by which we wish to create the database. Database name should be meaningful and descriptive as far as possible.

IF NOT EXIST is an optional clause. It checks for the name of database, if it already exists in the database server. If it already exists it stops the user to create a new database with the same name. Database server cannot have to two databases with the same name.

For example, to create a database with name 'student', the command is

CREATE DATABASE STUDENT;



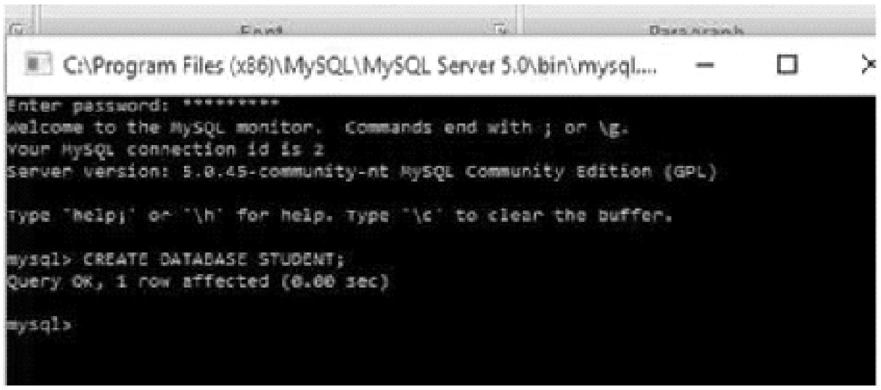
```
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)

mysql> CREATE DATABASE STUDENT;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| student            |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> _
```

Fig. 6.8

After successful execution of the command, MySQL displays a message saying if the database is created successful or not.

SHOW DATABASES;

This statement displays all database present in server. You can use this statement to check all databases or database you have created in the server. In the fresh installation of MySQL, when this statement is executed we get



```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql...
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> CREATE DATABASE STUDENT;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Fig.2.9 Making Database

USE DATABASE

Before working with a particular database, you have to tell which database will be used. You can work with the database that you want to use by using this statement, you can select the database named as 'student' using the following statement as follows.

USE STUDENT;

Now creating new table, placing data query or calling a stored procedure etc. shall be applicable to database STUDENT.

DATABASE DROP

Dropping a database means removing the database permanently. All data and related objects inside the database are permanently removed and this removal can not be reverted. Therefore, when this statement is given, a warning message appears on the screen. One should be extra careful while executing this statement. The syntax is

DROP DATABASE [IF EXISTS] database_name;

If we wish to remove STUDENT database, we write following statement.

e.g. DROP DATABASE STUDENT;

MySQL Data Types

The columns of a table may contain different types of values, such as numeric or string. MySQL provides more different data types than numeric or string. Each data type can be determined by the following specifications in MySQL

- ♦ The value represents what kind of values?
- ♦ Is the value of fixed length or variable length?
- ♦ Can the datatype of value be indexed?

Numeric Data

MySQL provides following numeric data types like SQL. Also, it provides facility to store single bit value with a data type called BIT. Numeric data types can be signed and unsigned. The table shows numeric data types available in MySQL.

Numeric Types	Description
TINYINT	A very small integer
SMALLINT	A small integer
MEDIUMINT	A medium-sized integer
INT	A standard integer
BIGINT	A large integer
DECIMAL	A fixed-point number
FLOAT	A single-precision floating point number
DOUBLE	A double-precision floating point number
BIT	A bit field

String Data

In MySQL, the string data type can be from simple text to binary data such as images and files. The string data type can be used for matching and searching values. The following table shows the string data type in MySQL

String Types	Description
CHAR	A fixed-length nonbinary (character) string
VARCHAR	A variable-length non-binary string
BINARY	A fixed-length binary string
VARBINARY	A variable-length binary string
TINYTEXT	A very small non-binary string
TEXT	A small non-binary string
MEDIUMTEXT	A medium-sized non-binary string
LONGTEXT	A large non-binary string
ENUM	An enumeration; each column value may be assigned one enumeration member
SET	A set; each column value may be assigned zero or more set members

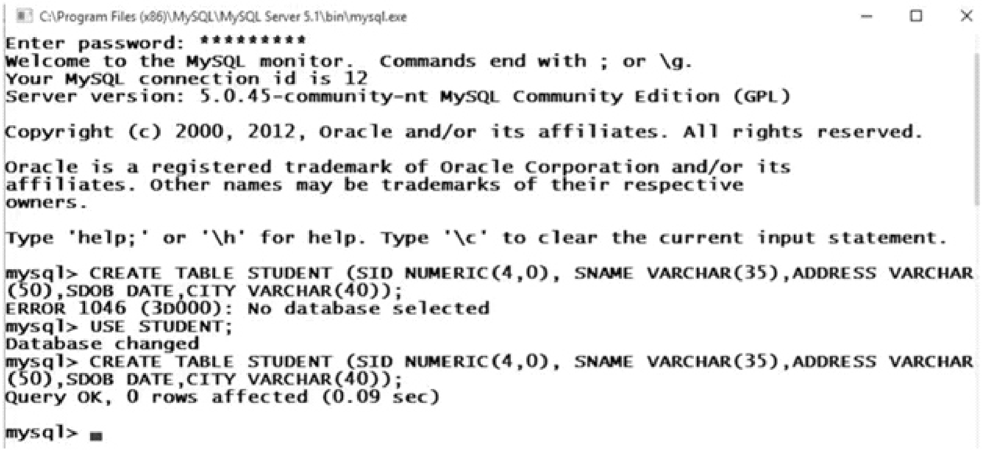
Date and Time Data Types

MySQL provides date and time data types as well as a combination of date and time data type. In addition, MySQL timestamp data type also provides the facility to track the changes made in a row in a table. To store only year without date and month, year data type can be used. The following table shows date and time data type in MySQL

Date and Time Types	Description
DATE	A date value in ;YYYY-MM-DD' format
TIME	A time value in ;hh:mm:ss' format
DATETIME	A date and time value in ;YYYY-MM-DD hh:mm:ss' format
TIMESTAMP	A timestamp value in ;YYYY-MM-DD hh:mm:ss' format
YEAR	A year value in YYYY or YY format

CREATE NEW TABLE IN A DATABASE

To make a new table within the database, the statement is 'create table' in MySQL. The statement of making tables is a complex statement, which is as follows.



```
C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE TABLE STUDENT (SID NUMERIC(4,0), SNAME VARCHAR(35),ADDRESS VARCHAR
(50),SDOB DATE,CITY VARCHAR(40));
ERROR 1046 (3D000): No database selected
mysql> USE STUDENT;
Database changed
mysql> CREATE TABLE STUDENT (SID NUMERIC(4,0), SNAME VARCHAR(35),ADDRESS VARCHAR
(50),SDOB DATE,CITY VARCHAR(40));
Query OK, 0 rows affected (0.09 sec)

mysql>
```

Figure 6.10

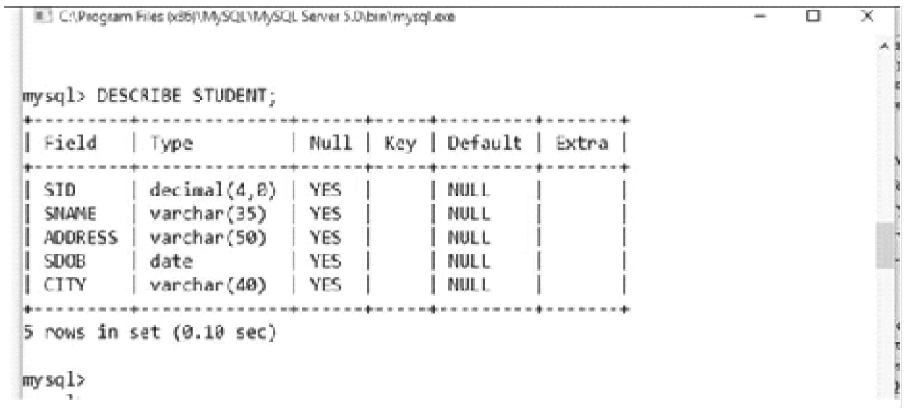
DESCRIBE TABLE

To see the attributes and their data types of a table, DESCRIBE command is used.

DESCRIBE TABLE_NAME

For example

DESCRIBE STUDENT;



```
C:\Program Files (x86)\MySQL\MySQL Server 5.0\bin\mysql.exe

mysql> DESCRIBE STUDENT;
+---------+--------------+------+-----+---------+-------+
| Field   | Type         | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| SID     | decimal(4,0) | YES  |     | NULL    |       |
| SNAME   | varchar(35)  | YES  |     | NULL    |       |
| ADDRESS | varchar(50)  | YES  |     | NULL    |       |
| SDOB    | date         | YES  |     | NULL    |       |
| CITY    | varchar(40)  | YES  |     | NULL    |       |
+---------+--------------+------+-----+---------+-------+
5 rows in set (0.10 sec)

mysql>
```

Figure 6.11 Describe Database

DROP TABLE

To remove an existing table, the following statement is used.

DROP [TEMPORARY] TABLE [IF EXISTS] table_name [, table_name]

For example, to remove STUDENT table, we write

DROP TABLE STUDENT;

DROP TABLE statement permanently deletes a table and its data from the database. This statement can be used for multiple tables also, in that case each table is separated by a comma.

ALTER TABLE

To change the structure of an existing table, the alter table statement is used. It allows to add a column, delete a column, change data type of a column, add a primary key, rename the table and do many other tasks. The syntax is

ALTER TABLE table_name action1[,action2,…]

To change the structure of an existing table, specify

- the name of table in which we wish to make changes.

- actions you want to apply to the table, any action, such as adding a new column, adding primary key, changing the name of the table, etc.

More than one action can be applied with a single 'alter table' statement. Each action is separated by a comma.

ADD COLUMN TO TABLE

After creating a table, due to some new requirements, there may be a requirement to add a new column, for example, to add a date_of_birth column to the STUDENT table, we write

ALTER TABLE STUDENT ADD COLUMN DOB date;

DOB column will be added after the last column in the table.

If the column to be added after a particular column, the name of that particular column will be written in the statement. For example, to add PINCODE colun, after

CITY, the statement will be

ALTER TABLE STUDENT ADD COLUMN PINCODE AFTER CITY;

DROP COLUMN

If we want to remove a column from the table, we can drop it, for example, suppose the table has a PINCODE column and now it is no more needed in the STUDENT table. The following statement will be written for the purpose.

ALTER TABLE STUDENT DROP COLUMN PINCODE;

RENAMING TABLE

To change the name ot the table, ALTER can be used, the statement will be

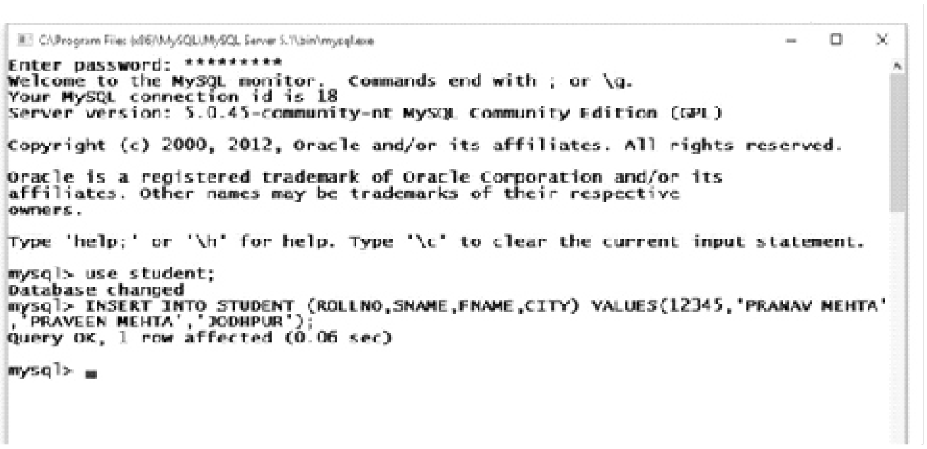
ALTER TABLE STUDENT RENAME TO STUINFO;

INSERT Statement

Insert statement is used to insert one or more rows into a table. The syntax of the insert statement is

INSERT INTO TABLE [COLUMN1, COLUMN2,.......] VALUES (VALUE1,VALUE2.....)

The list of columns is optional. After INSERT INTO, write list of columns of the specified table separated by commas within parentheses. Insert the values of columns separated by commas in the parentheses after the VALUES keyword.



```
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use student;
Database changed
mysql> INSERT INTO STUDENT (ROLLNO,SNAME,FNAME,CITY) VALUES(12345,'PRANAV MEHTA'
,'PRAVEEN MEHTA','JODHPUR');
Query OK, 1 row affected (0.06 sec)

mysql> ■
```

Figure 6.12 Inserting data in a table

For example:

If we wish to put in data of a student in the STUDENT table then the following INSERT statement will be

```
INSERT INTO STUDENT (ROLLNO,SNAME,FNAME,CITY)
VALUES (12345,'PRANAV MEHTA','PRAVEEN
MEHTA','JODHPUR');
```

List of columns is optional, hence, following is also valid.

```
INSERT INTO STUDENT VALUES (12345,'PRANAV
    MEHTA','PRAVEEN MEHTA','JODHPUR');
```

INSERT MULTIPLE ROWS

To add more than one rows in a table, at the same time, following syntax may be used.

```
INSERT INTO table (column1,column2...) VALUES (value1,value2,...),
(value1,value2,...), ……
```

For example, if we wish to add data of Pranav, Prachi and Rudra Pratap, we can write as

```
INSERT INTO STUDENT VALUES (12345,'PRANAV
MEHTA','PRAVEENMEHTA','JODHPUR'),(12346,'PRACHI MEHTA','PRAVEEN
MEHTA','JODHPUR'), (12347,'RUDRA PRATAP','RAKESH
MEHTA','JODHPUR');
```

Here, values of each rows are placed in brackets which are separated by comma.

In the INSERT statement, the value can also be inserted by the SELECT statement. It is possible to copy (partially, or full) the contents of a table to another.

```
INSERT INTO table_1  SELECT c1, c2, FROM table_2;
```

For example, to copy complete data of STUDENT, to another table TEMPSTU, we can write the following statement

```
INSERT INTO TEMPSTU SELECT * FROM STUDENT;
```

UPDATE

One of the important job while working with a database is data updating. To update some existing data in a table, we useUPDATE statement. We can use this statement to update one row, group of rows or all rows in the table. The UPDATE statement is as follows.

```
UPDATE table_name

SET

  column_name1 = expr1,

  column_name2 = expr2,

   ...

WHERE

  condition;
```

Firstly,after the update keyword, specify the table name you want to update. Second, specify the column which you want to modify and give new value with the SET statement. To update multiple columns,separate them by commas.

Third, specify which rows will be updated using a condition in the where clause. This clause is optional. If we leave the clause, all rows in the table will be updated.

Where clause is extremely important.It should be used with caution, missing this clause, will update all the row in the database.

Single Column Example:

In this example, we want to update Pranav's email with the new e-mail pranav@boser.edu.in.

```
UPDATE STUDENT SET EMAIL= 'pranav@boser-edu-in' WHERE
SNAME='PRANAV';
```

Multiple Column Example:

In this example, we want to update email and lastname of rollno 205502.

UPDATE SET LASTNAME='MOHNOT', EMAIL=' pranav@boser-edu-in' WHERE

ROLLNO=205502;

DELETE Statement

To remove data from one or more tables, the DELETE statement is used. A single DELETE statement can be used to remove records from multiple tables.

DELETE FROM table

[WHERE conditions]

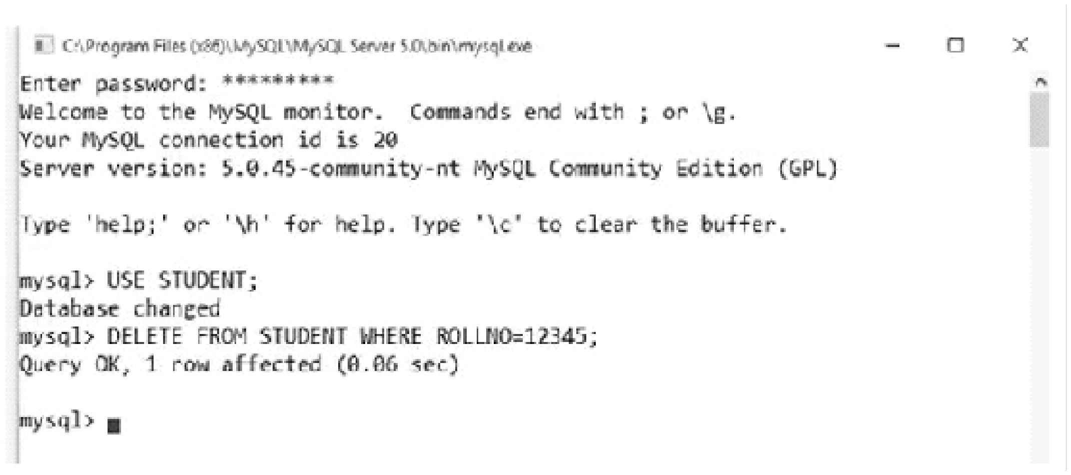
[ORDER BY ...]

[LIMIT rows]

WHERE Clause describes which row(s) you want to remove. If the record fulfills the condition given in WHERE clause, the row is deleted permanently from the table. If we do not use the WHERE clause, all records of that table will be removed. ROW_COUNT() function shall return the number of records deleted by the preceding statement.

For example, to delete the record of student who has rollno 123456, we write

DELETE FROM STUDENT WHERE ROLLNO=123456



```
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> USE STUDENT;
Database changed
mysql> DELETE FROM STUDENT WHERE ROLLNO=12345;
Query OK, 1 row affected (0.06 sec)

mysql> 
```

Figure 6.14 Removing record from database

To delete all records from the table, the statement is

```
DELETE FROM STUDENT;
```

All records from the stable STUDENT are deleted by this statement.

Removing Records from more than one table

```
DELETE table_1, table_2,...

FROM table-refs

[WHERE conditions]

DELETE FROM table_1, table_2,...

USING table-refs

[WHERE conditions]
```

For example, in a situation where one course is closed, to remove the record of that course and all records of students studying that course, the DELETE statement shall be

```
DELETE STUDENT, COURSE

FROM STUDENT, COURSE

WHERE STUDENT.COURSEID = COURSE.COURSEID AND
COURSE.COURSEID=1;
```

SELECT statement

The SELECT statement is used to obtain data from the table. A table is a combination of rows and columns which is like a spreadsheet. Most of all, we want to see a group of rows or group of columns or a combination of both of them. The yield of the SELECT statement is called RESULT SET.

The result set is a list of group rows that have the same number of columns.

For example, see table STUDENT, which has columns – rollno, stuname, dob, email, coursecode

ROLLNO	STUNAME	DOB	EMAIL	COURSE CODE
100201	Pranav	2000-2-15	pranav@boser.edu.in	1
100202	Prachi	2000-3-10	prachi@boser.edu.in	2
100203	Riya	2001-5-21	riya@boser.edu.in	1
100204	Rudra Pratap	2000-4-25	rudra@boser.edu.in	1
100205	Riyansh	2002-3-11	riyansh@boser.edi.in	2

Which columns and rows we want to see we use SELECT statement. For example, if wish to see only name of student and email or name of student and date of birth, SELECT statement helps in doing so. The syntax of SELECT statement is

```
SELECT
        column_1, column_2, column_3 ...
FROM
        table_1
        [INNER | LEFT |RIGHT] JOIN table_2 ON conditions
WHERE
        Conditions
GROUP BY column_1
HAVING group_conditions
ORDER BY column_1
```

SELECT statement has different clauses, which are mentioned below.

- The SELECT statement allows for partial query in a table of data, for that a list of columns separated by commas is specified in the SELECT clause or a * (asterisk) which means that you want to see all the columns.
- JOIN is used to receive data from other tables based on the conditions of relations.

- WHERE is a filter of ROWS in WHERE RESULT SET.
- GROUP BY creates a group of rows and the AGGREGATE function is held on each group.
- HAVING clause filtersthe groups made by GROUP BY.
- ORDER BY specifies a list of columns to sort.
- SELECT and FROM clauses are required in the statement, the other part is optional.
- SELECT statement permits to query partially about data in a table; for this purpose a list of columns separated by commas is specified.

Example

To see student name and email, we write the query as

```
SELECT
        STUNAME,EMAIL
        FROM STUDENT;
```

To see all columns, we can write name of each column or simply *.

```
        SELECT ROLLNO,STUNAME, FNAME,EMAIL,CITY
        FROM STUDENT;
        OR
        SELECT *
        FROM STUDENT;
```

These two queries will result the same.

GROUP BY

This part is optional in SELECT statement. It groups and summarises a number of rows. For every group there is one row as the output.In other words, it reduces the number of rows in the result set.

We often use aggregate functions such as SUM, AVG, MAX, MIN and COUNT with GROUP BY. The aggregate functions provides information about individual groups, selected by SELECT statement. The syntax is

```
SELECT

        c1, c2,..., cn, aggregate_function(ci)

FROM

        table

WHERE

        where_conditions

GROUP BY c1 , c2,...,cn;
```

The GROUP BY segment should be written after the FROM and WHERE clauses. After GROUP BY, columns separated by commas are written or an expression that you want to use as criteria is written.

Aggregate function gives a single value after processing a group of rows. The GROUP BY is generally used with an aggregate function which provides a single value for each subgroup.

For example,

If we want to know how many courses are there, we can write GROUP BY with COUNT as

```
SELECT COURSEID,COUNT(*)

        FROM COURSE

        GROUP BY COURSEID
```

If we want a group according to the courses then the following statement will be written

```
SELECT *

FROM STUDENT

GROUP BY COURSEID
```

If we want to know how many students in the course, we use the GROUP BY segment with the COURSE column as the following query.

```
SELECT COURSEID,COUNT(*)

FROM STUDENT

GROUP BY COURSEID
```

We use the HAVING block to filter the group given by GROUP BY. The following query will be written using the HAVING section to find the number of students born after year 2003.

```
SELECT DOB,YEAR(DOB)  AS YEAR, COUNT(*)

FROM STUDENT

GROUP BY YEAR

HAVING YEAR(DOB) > 2003;
```

The HAVING segment is used to filter the set or group of rows in the SELECT statement. The filter workson the column(s) in the GROUP BY block, if the GROUP BY block is omitted, then the HAVING segment acts like FROM.

The filter applies to each group of rows in the HAVING section,whereas the filter condition in WHERE section applies to every different row.

When you use the SELECT statement to query the data from a table, the result set does not occur in any sequence. To order result set, we use the ORDER BY. By using ORDER BY segment

· Result set can be sorted by a single column or multiple columns

· Result set, different columns, can be sorted in either ascending or descending order.

ORDER BY block is used as

```
SELECT column1,cloumn2, .........

FROM table_name

ORDER BY column1 [ASC|DSC],column2 [ASC|DSC], …
```

For example, we wish list the names of the students in ascending alphabetical order, the query will be as follows:

```
SELECT SNAME, ROLLNO

FROM STUDENT

ORDER BY SNAME;
```

If we wish list the names of the students in descending order, the query will be as follows:

```
SELECT SNAME, ROLLNO

FROM STUDENT

ORDER BY SNAME DESC;
```

If we need merit list of the candidates on the basis of total marks obtained, the query will be as follows.

```
SELECT ROLLNO, SNAME, TOTALMKS

FROM STUDENT

ORDER BY TOTALMKS;
```

6.8.2 MySQL FUNCTIONS

Commonly used functions are described below in brief.

AVG: Calculates the average of a set

For example, to find the average score of the first subject of the students

```
SELECT AVG(SUB1) FROM STUDENT;
```

To find the average score of the first subject of the students who secured first division.

```
SELECT AVG(SUB1) FROM STUDENT WHERE DIV='I';
```

Using the DISTINCT operator, you can use the average function to calculate the average of different scores.

```
SELECT AVG(DISTINCT SUB1) FROM STUDENT;
```

We often use the average function to compute the average value for each group of rows in a table in conjunction with the GROUP BY segment. According to the syllabus, the average score of the first subject of the student is to be calculated as

```
SELECT AVG(SUB1) FROM STUDENT GROUP BY COURSE;
```

In order to set conditions for the average values for groups, the function AVG can be use in the HAVING section. To find average score of the marks of the first subject of the first class students for all courses, the statement will be

```
SELECT AVG(SUB1) FROM STUDENT GROUP BY COURSE
HAVING AVG(SUB1)>59;
```

COUNT: Tells the number of rows in a table.

COUNT Function returns the number of rows in a table. It counts the rows which are satisfying given condition.

COUNT function works in many different ways. It provides value in BIGINT data type. It returns zero when no matching row is detected.

COUNT(*) returns the number of rows in a table. It includes the rows which have NULL values.

For example, to find total number of rows in table Students, we write

```
SELECT COUNT(*)

FROM STUDENTS;
```

To calculate, we can apply some condition using where, like

```
SELECT COUNT(*)

FROM STUDENTS

WHERE DIV='FIRST';
```

To calculate unique rows in a table, we use DISTINCT with COUNT, like

```
COUNT (DISTINCT expression)

SELECT COUNT(DISTINCT COURSEID)
```

FROM STUDENT;

Many a times, COUNT function is used with GROUP BY clause to identify data in groups.

SUM: Calculates the sum of a set.

SUM function is used for adding a set of values or expressions. If there is no value matching in the SELECT statement, the result of SUM function will be zero.We can use DISTINCT with SUM to find out the sum of only those values which are distinct. SUM function ignores NULL values for calculation.

SUM(DISTINCT expression)

For example, to find the sum of fee deposited by students, the value will be stored in TOTALFEE.

SELECT SUM(FEE) AS TOTALFEE

FROM STUDENT;

If we wish to sum the fee, course wise

SELECT SUM(FEE) AS TOTALFEE

FROM STUDENT

GROUP BY COURSEID;

SUM can be used with HAVING function to filter results based on a specific condition. The SUM function ignores NULL values in the calculation.

MIN: Finds the minimum from the set.

MIN function gives the minimum value in the set. It is very useful function in many situations like, finding the minimum marks obtained, or cheapest product etc. The syntax is

MIN(expression);

If we wish to find out the minimum marks obtained in first subject, we write

SELECT SNAME, MIN(SUB1)

FROM STUDENT;

MAX: Finds the maximum from the set.

MAX function gives the maximum value in the set. It is very useful function in many situations like, finding the highest marks obtained, or costliest product etc. The syntax is

MAX(expression);

If we wish to find out the maximum marks obtained in first subject, we write

SELECT SNAME, MAX(SUB1)

FROM STUDENT;

STRING Functions

CONCAT and CONCAT_WS are used to add two or more strings at once using the function.

CONCAT(expr1, expr2…)

The CONCAT function takes one or more strings and combines them into a single string. The CONCAT function, there is minimum requirement of one string otherwise it will give an error. The syntax is

CONCAT(sting1,string2,string3,.........);

SELECT CONCAT('Suncity','Jodhpur');

The result will be SuncityJodhpur

SELECT CONCAT('Suncity',' ;,'Jodhpur');

The result will be Suncity Jodhpur

SELECT CONCAT('Suncity',', ;,'Jodhpur');

The result will be Suncity.Jodhpur

CONCAT_WS combines two or more string values with a predefined separator.
CONCAT_WS(separator,string1,string2,string3,...)

SELECT CONCAT_WS('-','Pranav','Mehta');

Output will be :Pranav.Mehta

SELECT CONCAT_WS('-','Pranav','Mehta');

Output will be : Pranav-Mehta

LENGTH function: It is used for finding the length of string, i.e. how many letters, characters, spaces in the string. It is measured in bytes.

CHAR_LENGHT: This function also works to get the number of letters, characters, spaces in the string. It is measured in characters.

SELECT LENGTH('prachi');

Output will be 5

SELECT LENGTH('riyamehta');

Output will be 10

LEFT() function

It is used to return a string of specified length from left a string. LEFT function accepts two arguments. The syntax is

LEFT(str, length)

1. The str string is the one from which we have to get the substring.
2. The length specifies the number of characters which will be a positive integer. The str string from the function gives the left-most number of letters written in LENGTH. If the value of str or LENGTH is NULL, then it returns a NULL value.

If the LENGTH is zero or negative, the LEFT function returns an empty string. If LENGTH is greater than the length of the string, then the LEFT function returns the entire string.

RIGHT() function

It is used to return a string of specified length from right of a string. RIGHT Function accepts two arguments. The syntax is

RIGHT(str, length)

1. The str string is the one from which we have to get the substring.

2. The length specifies the number of characters which will be a positive integer. The str string from the function gives the right-most number of letters written in LENGTH. If the value of str or LENGTH is NULL, then it returns a NULL value.

If the LENGTH is zero or negative, the RIGHT function returns an empty string. If LENGTH is greater than the length of the string, then the RIGHT function returns the entire string.

TRIM

TRIM function is used to remove unwanted spaces from the beginning or end of a string.

SELECT TRIM ('Sample Text ');

Result: Sample Text

DATE

The date function is marked in YYYY-MM-DD format. The values ??of DATE can also be given in a string or number. This function accepts the value between 1000-01-01 to 9999-12-31.

MONTH

The MONTH function returns the value of the month from the value of DATE, it is between 1 and 12.

SELECT MONTH('2016-02-03')

Result: 2

YEAR

The YEAR function provides value for the year from the value of DATE.

SELECT YEAR('2016-02-03')

Result: 2016

DAY

The DAY function returns the value of the day of the month from the value of

DATE, it is between 1 and 31. This is synonymous with DAYOFMONTH ().

SELECT DAY ('2016-02-15')

Result: 15

SYSDATE

SYSDATE () provides current date and time value of the functioning system according to the string or number data as YYYY -MM-DD HH: MM: SS or YYYYMMDDHHMMSS.

SELECT SYSDATE ();

Result: 2016 -01 -16 13 Rs 47 per Rs 36

DATEDIFF

The DATEDIFF function calculates the number of days between the two dates, the date time, or the timestamp value.

Important Points

1. The database can normally be described as a repository for data.
2. Data is a collection of facts and figures that can be processed for providing information.
3. Data base management system is a software that has the functions of database storage, access, security, backup and such other features.
4. Data base management systems is described by three-level schemas (schema architecture).
5. Databases system is divided into two modules - Storage Manager and Query Processor.
6. Most important characteristic of DBMS is that data redundancy can be controlled.
7. Well-processed data is called information.
8. In any system, database languages are used to create and maintain a database.

9. Two types of languages are used in database – Data Definition Language and Data Manipulation Language.

10. Structured Query Language (SQL) is database language.

11. MySQL is a database management system used for managing relational database. It is open source software.

Exercises

Objective Type Question

1. Raw facts and figures are

   A. Data   B. Information   C. Snapshot   D. Report

2. The facility which permits accessing only a few records in database.

   A. Form   B. Report   C. Queries   D. Tables

3. What are relational databases?

   A. To store relational information at one place.

   B. Relation of one database with another

   C. A database to store human relations

   D. Any of the above

4. With this key, each record is identified uniquely

   A. Primary Key   B. Key Record

   C. Unique Key   D. Name of Field

5. Database language related to definition of schema and complete database structure is

   A. DCL   B. DML   C. DDL   D. All of the Above

Short Type Questions

1. What do you understand with database?
2. Explain database architecture.
3. How many data types are available in MySQL?

4. How records are inserted in MySQL? Explain.
5. What is the utility of UPDATE function?

Essay Type Questions

1. Explain database management system and its advantages.
2. Write the characteristics of database management system.
3. How many types of database languages are there? Explain in detail.
4. How a database of students can be created in MySQL? Explain in detail.
5. Explain string function in MySQL.

Answers Key

1.A 2. C 3. A 4. A 5. B