

# Establishing a Database Connection and Working With Database

# Database Overview

- A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.
- Other kinds of data stores can be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.
- So nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or foreign keys.

# Database Overview

A Relational Database Management System (RDBMS) is a software that:

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

# RDBMS Terminology

- Before we proceed to explain MySQL database system, let's revise few definitions related to database.
- **Database:** A database is a collection of tables, with related data.
- **Table:** A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- **Column:** One column (data element) contains data of one and the same kind, for example the column postcode.
- **Row:** A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.

# RDBMS Terminology

- **Redundancy:** Storing data twice, redundantly to make the system faster.
- **Primary Key:** A primary key is unique. A key value can not occur twice in one table. With a key, you can find at most one row.
- **Foreign Key:** A foreign key is the linking pin between two tables.
- **Compound Key:** A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- **Index:** An index in a database resembles an index at the back of a book.
- **Referential Integrity:** Referential Integrity makes sure that a foreign key value always points to an existing row.

# MySQL Database

- MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:
- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.

# MySQL Database

- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

# MySQL Datatypes

- Database table contains multiple columns with specific data types such as numeric or string. MySQL provides more data types other than just numeric or string. Each data type in MySQL can be determined by the following characteristics:
  - - Kind of values it can represent.
  - - The space that takes up and whether the values are fixed-length or variable-length.
  - - Does the values of the data type can be indexed.
  - - How MySQL compares the value of a specific data type.



# MySQL Numeric Data Types

- **INT** – A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
- **TINYINT** – A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.
- **SMALLINT** – A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.

# MySQL Numeric Data Types

- **MEDIUMINT** – A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.
- **BIGINT** – A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.
- **FLOAT(M,D)** – A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

# MySQL Numeric Data Types

- **DOUBLE(M,D)** – A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
- **DECIMAL(M,D)** – An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

# Date and Time Types

- **DATE** – A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** – A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** – A timestamp between midnight, January 1st, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 ( YYYYMMDDHHMMSS ).

# Date and Time Types

- **TIME** – Stores the time in a HH:MM:SS format.
- **YEAR(M)** – Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

# String Types

- **CHAR(M)** – A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** – A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.
- **BLOB or TEXT** – A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two is that the sorts and comparisons on the stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.

# String Types

- **TINYBLOB or TINYTEXT** – A BLOB or TEXT column with a maximum length of 255 characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB or MEDIUMTEXT** – A BLOB or TEXT column with a maximum length of 16777215 characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.

# String Types

- **LONGBLOB or LONGTEXT** – A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LONGBLOB or LONGTEXT.
- **ENUM** – An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.



# Work with MySQL three way

- We can work with MySQL three ways as below.
  1. **PHPMyAdmin** : Using PHPMyAdmin we can connect to mysql and create tables add rows in GUI Format.
  2. **MySQL Console** : Using MySQL Console we can access database using command mode. We need to write script to access to create database, update etc.
  3. **PHP** : we can access database using mysql PHP functions to make connection, get rows , update rows, delete rows etc.

# Database and Database Management System

- Database is simply a collection of data. In relational database, data is organized into tables.

Enrollment Number	Student Name	Gender	Mobile Number
101	Alpesh	Male	9999999999
102	Vijay	Male	8888888888
103	Seema	Female	8933333333
	...	...	...

- Database Management System (DBMS) is software to maintain and utilize the collections of data (SQL Server, Oracle, DB2, MySQL)

# MySQL Introduction

- MySQL is a database management system
- SQL stands for the Structured Query Language. It defines how to insert, retrieve, modify and delete data
- Free from [www.mysql.com](http://www.mysql.com)
- Reference sites
  - NASA, Yahoo!, Compaq, Motorola

# Basic MySQL Operations

- Create table
- Insert records
- Load data
- Retrieve records
- Update records
- Delete records
- Modify table
- Join table
- Drop table
- Optimize table
- Count, Like, Order by, Group by
- More advanced ones (sub-queries, stored procedures, triggers, views ...)

# How MySQL stores data (by default)

- A MySQL server can store several databases
- Databases are stored as directories
  - Default is at `/usr/local/mysql/var/`
- Tables are stored as files inside each database (directory)
- For each table, it has three files:
  - `table.FRM` file containing information about the table structure
  - `table.MYD` file containing the row data
  - `table.MYI` containing any indexes belonging with this table, as well as some statistics about the table.

# Login

- `mysql -h hostname -u username -p [password]`

- Example

```
% mysql -u usr Name -p
```

```
Enter password: passowrd
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

```
Your MySQL connection id is 23 to server version: 3.23.41.
```

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

```
mysql>
```

# Create User and Database

- `mysql>use mysql;`
  - Use database mysql, used by the system
- `mysql>insert into user (Host, User, Password) values ('localhost', 'test1', password('pass1'));`
  - Create a new database user test1
  - An alternative
    - `GRANT USAGE ON *.* TO 'test1'@'localhost' IDENTIFIED BY 'pass1';`

# Create User and Database (cont.)

- `mysql>insert into db (Host, Db, User, Select_priv, Insert_priv, Update_priv, Delete_priv, Create_priv, Drop_priv) values ('localhost', 'testdb', 'test1', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y');`
  - Create a new database testdb for user test1
- `mysql>flush privileges`
  - Reloads the privileges from the grant tables in the database mysql
- An alternative
  - `GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP ON testdb.* TO 'test1'@'localhost' IDENTIFIED BY 'pass1';`



# Create Database

What are the current databases at the server?

```
mysql> show databases;
```

```
+-----+
```

```
| Database |
```

```
+-----+
```

```
| mysql    |      mysql is a database (stores users' password ...) used by system.
```

```
| test     |
```

```
+-----+
```

Create a database (make a directory) whose Name is MyDB

```
mysql> create database MyDB;
```

Select database to use

```
mysql> use MyDB;
```

Database changed

What tables are currently stored in the MyDB database?

```
mysql> show tables;
```

Empty set (0.00 sec)

# Create Table

- CREATE TABLE Table\_Name (column\_specifications)
- Example

```
mysql> CREATE TABLE student
```

```
-> (
```

```
-> EnrollmentNumber INT UNSIGNED NOT NULL,
```

```
-> Student Name      VARCHAR(20) NOT NULL,
```

```
-> Gender      VARCHAR(50),
```

```
-> MobileNumber      VARCHAR(5)
```

```
-> );
```

```
Query OK, 0 rows affected (0.00 sec)
```

EnrollmentNumber	Student Name	Gender	MobileNumber
------------------	--------------	--------	--------------

# Display Table Structure

```
mysql> show tables;
```

```
+-----+  
| Tables_in_MyDB |  
+-----+  
| student        |  
+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql> describe student;
```

```
+-----+-----+-----+-----+-----+-----+  
| Field          | Type                | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| EnrollmentNumber | int(10) unsigned    |      |     | 0        |       |  
| Student Name     | varchar(20)         |      |     |          |       |  
| Gender          | varchar(50)         | YES  |     | NULL     |       |  
| MobileNumber     | varchar(5)          | YES  |     | NULL     |       |  
+-----+-----+-----+-----+-----+-----+
```

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

# Modify Table Structure

- ALTER TABLE table\_name Operations

```
mysql> alter table student add primary key (EnrollmentNumber);
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> describe student;
```

Field	Type	Null	Key	Default	Extra
EnrollmentNumber	int(10) unsigned		<b>PRI</b>	0	
StudentName	varchar(20)				
Gender	varchar(10)	YES		NULL	
MobileNumber	varchar(5)	YES		NULL	

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

# Insert Record

- `INSERT INTO table_name SET col_Student Name1=value1, col_Student Name2=value2, col_Student Name3=value3, ...`
- Example

```
mysql> INSERT INTO student SET EnrollmentNumber=101, StudentName='Alpesh',  
    Gender='Male', MobileNumber='9844883374';
```

Query OK, 1 row affected (0.00 sec)

EnrollmentNumber	Student Name	Gender	MobileNumber
101	Alpesh	Male	9844883374

# Retrieve Record

- SELECT what\_columns  
FROM table or tables  
WHERE condition
- Example

Enrollment Number	Student Name	Gender	Mobile Number
101	Alpesh	Male	9999999999
102	Vijay	Male	8888888888
103	Seema	Female	8933333333
	...	...	...

```
mysql> SELECT Gender, MobileNumber  
FROM student WHERE Student  
Name='Alpesh';
```

```
+-----+-----+  
| Gender| MobileNumber|  
+-----+-----+  
| Male | 99999999999 |  
+-----+-----+  
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM student;
```

# Update Record

- UPDATE table\_name

SET which columns to change

WHERE condition

- Example

```
mysql> UPDATE student SET MobileNumber='9844883374' WHERE StudentName='Alpesh';
```

```
Query OK, 1 row affected (0.00 sec)
```

```
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM student WHERE Student Name='Alpesh';
```

```
+-----+-----+-----+-----+
```

```
| StudentName | EnrollmentNumber | Gender | MobileNumber |
```

```
+-----+-----+-----+-----+
```

```
| Alpesh | 101 | Male | 9844883374 |
```

```
+-----+-----+-----+-----+
```

```
1 row in set (0.00 sec)
```

# Delete Record

- DELETE FROM table\_Student Name WHERE condition
- Example

```
mysql> DELETE FROM student WHERE Student Name='Shannon';  
Query OK, 1 row affected (0.00 sec)
```

```
Mysql> DELETE FROM student;
```

**Will delete ALL student records!**



# Drop Table

- DROP TABLE table\_Student Name
- Example

```
mysql> drop table student;
```

```
Query OK, 0 rows affected (0.00 sec)
```

- Logout MySQL

```
mysql> quit;
```

# Buck Load

- Load batch data instead of inserting records one by one
- Example

```
mysql> LOAD DATA LOCAL INFILE "student.txt" INTO TABLE student;  
Query OK, 21 rows affected (0.01 sec)  
Records: 21 Deleted: 0 Skipped: 0 Warnings: 0
```

```
mysql> LOAD DATA LOCAL INFILE "project.txt" INTO TABLE project;  
Query OK, 7 rows affected (0.00 sec)  
Records: 7 Deleted: 0 Skipped: 0 Warnings: 0
```

# More Table Retrieval

- OR

```
mysql> select StudentName from student where Gender = 'BCB' OR Gender = 'CS';
```

- COUNT (Count query results)

```
mysql> select count(Student Name) from student where Gender = 'BCB' OR Gender = 'CS';
```

- ORDER BY (Sort query results)

```
mysql> select Student Name from student where Gender = 'BCB' OR Gender = 'CS' ORDER BY Student Name;
```

```
mysql> select Student Name from student where Gender = 'BCB' OR Gender = 'CS' ORDER BY Student Name DESC;
```

```
mysql> select * from student where Gender = 'BCB' OR Gender = 'CS' ORDER BY EnrollmentNumber ASC, Student Name DESC
```

- LIKE (Pattern matching)

```
mysql> select Student Name from student where Student Name LIKE "J%";
```

- DISTINCT (Remove duplicates)

```
mysql> select Gender from student;
```

```
mysql> select DISTINCT Gender from student;
```

# MySQL PHP Function

- **mysqli\_connect()** Opens a new connection to the MySQL server.
- **mysqli\_affected\_rows()** Returns the number of affected rows in the previous MySQL operation.
- **mysqli\_close()** Closes a previously opened database connection.

# MySQL PHP Function

- **mysqli\_errno()** Returns the last error code for the most recent function call.
- **mysqli\_error()** Returns the last error description for the most recent function call.
- **mysqli\_fetch\_all()** Fetches all result rows as an associative array, a numeric array, or both.

# MySQL PHP Function

- **mysqli\_fetch\_array()** Fetches a result row as an associative, a numeric array, or both.
- **mysqli\_fetch\_assoc()** Fetches a result row as an associative array.
- **mysqli\_fetch\_row()** Fetches one row from a result-set and returns it as an enumerated array.
- **mysqli\_free\_result()** Frees the memory associated with a result.

# MySQL PHP Function

- **mysqli\_num\_rows()** Returns the number of rows in a result set.
- **mysqli\_query()** Performs a query against the database.
- **mysqli\_real\_escape\_string()** Escapes special characters in a string for use in an SQL statement.
- **mysqli\_select\_db()** Changes the default database for the connection.

# PHP mysqli\_connect() Function

The `mysqli_connect()` function opens a new connection to the MySQL server.

## Syntax

```
mysqli_connect(host,username,password,new,flag);
```

```
mysqli_connect(host,username,password,dbname,port,socket);
```

## Parameter:

**Host:** Optional. Specifies a host name or an IP address.

**Username:** Optional. Specifies the MySQL username.

**Password:** Optional. Specifies the MySQL password.

**DBname:** Optional. Specifies the default database to be used.

**Port:** Optional. Specifies the port number to attempt to connect to the MySQL server.

**Socket:** Optional. Specifies the socket or named pipe to be used.

**Return Value:** Returns an object representing the connection to the MySQL server.



# PHP mysqli\_select\_db() Function

The mysqli\_select\_db() function is used to change the default database for the connection.

## Syntax

```
mysqli_select_db(dbname, connection);  
mysqli_select_db(connection,dbname);
```

## Parameter:

**Connection:** Required. Specifies the MySQL connection to use.

**Dbname:** Required. Specifies the default database to be used.

**Return Value:** TRUE on success. FALSE on failure.

# PHP mysqli\_query() Function

The mysqli\_query() function performs a query against the database.

## Syntax

```
mysqli_query(query, connection,resultmode);  
mysqli_query(connection,query,resultmode);
```

## Parameter

**Connection:** Required. Specifies the MySQL connection to use.

**Query:** Required. Specifies the query string

**Resultmode:** Optional. A constant. Either: MYSQLI\_USE\_RESULT (Use this if we have to retrieve large amount of data) or MYSQLI\_STORE\_RESULT (This is default).

**Return Value:** For successful SELECT, SHOW, DESCRIBE, or EXPLAIN queries it will return a mysqli\_result object. For other successful queries it will return TRUE. FALSE on failure.

# PHP mysqli\_affected\_rows() Function

- The mysqli\_affected\_rows() function returns the number of affected rows in the previous SELECT, INSERT, UPDATE, REPLACE, or DELETE query.

## Syntax

```
mysqli_affected_rows(connection);
```

## Parameter

Connection: Required. Specifies the MySQL connection to use.

See Example....

# PHP mysqli\_close() Function

- The mysqli\_close() function closes a previously opened database connection.

## Syntax

```
mysqli_close(connection);
```

## Parameter

**Connection:** Required. Specifies the MySQL connection to close.

**Return Value:** it returns Boolean value True on success and False in Failure.

See Example....

# PHP mysqli\_errno() Function

The mysqli\_errno() function returns the last error code for the most recent function call, if any.

## Syntax

```
mysqli_errno(connection);
```

## Parameter:

**Connection:** Required. Specifies the MySQL connection to use.

**Return Value:** Returns an error code value. Zero if no error occurred.

See Example....

# PHP mysqli\_error() Function

The `mysqli_error()` function returns the last error description for the most recent function call, if any.

## Syntax

```
mysqli_error(connection);
```

## Parameter:

**Connection:** Required. Specifies the MySQL connection to use.

**Return Value:** Returns a string with the error description. "" if no error occurred.

See Example....

# PHP mysqli\_fetch\_all() Function

- The mysqli\_fetch\_all() function fetches all result rows and returns the result-set as an associative array, a numeric array, or both.
- Note: This function is available only with MySQL Native Driver.

## Syntax

```
mysqli_fetch_all(result,resulttype);
```

## Parameter:

**Result:** Required. Specifies a result set identifier returned by mysqli\_query(), mysqli\_store\_result() or mysqli\_use\_result().

**Resulttype:** Optional. Specifies what type of array that should be produced. Can be one of the following values (1) MYSQLI\_ASSOC (2) MYSQLI\_NUM (3) MYSQLI\_BOTH

**Return Value:** Returns an array of associative or numeric arrays holding the result rows.

See Example....

# PHP mysqli\_fetch\_array() Function

- The mysqli\_fetch\_array() function fetches a result row as an associative array, a numeric array, or both.
- **Note:** Fieldnames returned from this function are case-sensitive.

## Syntax

```
mysqli_fetch_array(result,resulttype);
```

## Parameter:

**Result:** Required. Specifies a result set identifier returned by mysqli\_query(), mysqli\_store\_result() or mysqli\_use\_result().

**Resulttype:** Optional. Specifies what type of array that should be produced. Can be one of the following values (1) MYSQLI\_ASSOC (2) MYSQLI\_NUM (3) MYSQLI\_BOTH

**Return Value:** Returns an array of strings that corresponds to the fetched row. NULL if there are no more rows in result-set.

See Example....



# PHP `mysqli_fetch_assoc()` Function

- The `mysqli_fetch_assoc()` function fetches a result row as an associative array.
- Note: Fieldnames returned from this function are case-sensitive.

## Syntax

```
mysqli_fetch_assoc(result);
```

## Parameter

**Result:** Required. Specifies a result set identifier returned by `mysqli_query()`, `mysqli_store_result()` or `mysqli_use_result()`.

**Return Value:** Returns an associative array of strings representing the fetched row. NULL if there are no more rows in result-set.

See Example....

# PHP mysqli\_fetch\_row() Function

The `mysqli_fetch_row()` function fetches one row from a result-set and returns it as an enumerated array.

## Syntax

```
mysqli_fetch_row(result);
```

## Parameter:

**Result:** Required. Specifies a result set identifier returned by `mysqli_query()`, `mysqli_store_result()` or `mysqli_use_result()`.

**Return Value:** Returns an array of strings that corresponds to the fetched row. NULL if there are no more rows in result set.

See Example....

# PHP mysqli\_free\_result() Function

- The mysqli\_free\_result() function frees the memory associated with the result.

## Syntax

```
mysqli_free_result(result);
```

## Parameter:

**Result:** Required. Specifies a result set identifier returned by mysqli\_query(), mysqli\_store\_result() or mysqli\_use\_result().

**Return Value:** No return value.

# PHP mysqli\_num\_rows() Function

The mysqli\_num\_rows() function returns the number of rows in a result set.

## Syntax

```
mysqli_num_rows(result);
```

## Parameter:

**Result:** Required. Specifies a result set identifier returned by mysqli\_query(), mysqli\_store\_result() or mysqli\_use\_result().

**Return Value:** Returns the number of rows in the result set.

# PHP mysqli\_real\_escape\_string() Function

- The mysqli\_real\_escape\_string() function escapes special characters in a string for use in an SQL statement.

## Syntax

```
mysqli_real_escape_string(connection,escapestring);
```

## Parameter:

**Connection:** Required. Specifies the MySQL connection to use

**Escapestring:** Required. The string to be escaped. Characters encoded are NUL (ASCII 0), \n, \r, \, ', ", and Control-Z.

**Return Value:** Returns the escaped string.

See Example....

Thank you