MySQL

For Developers

CONTENTS

- History
- About MySQL
- RDBMS
- ERD
- MySQL Terminologies
- Users and Privileges
- Databases
- Tables
- Column data types
- Indexes

HISTORY

1994

Michael Widenius and David Axmark started the development of MySQL

2005

Oracle Corporation acquired Innobase OY, the company that developed the InnoDB

2006

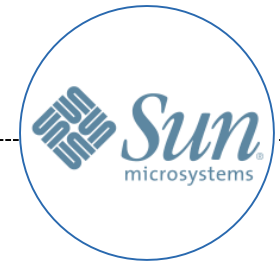
Oracle Corporation acquired Sleepycat Software, makers of the Berkeley DB

2012

MySQL creators announced the development of MariaDB as a fork from MySQL

2009

Oracle Cooperation acquired Sun Microsystems



2008

Sun Microsystems acquired MySQL AB

MYSQL POPULARITY

- It is the world's most popular open source database and has won the Linux Journal Readers' Choice Award on a number of occasions. It is used in Google, Wikipedia, Facebook and Yahoo!

MYSQL PRODUCTS

- **MySQL Enterprise Server**

Available only with the MySQL Enterprise subscription.

- **MySQL Community Server**

The MySQL database server for open source developers and technology enthusiasts who want to get started with MySQL. Supported by the large MySQL open source community. Under the General Public License (GPL), benefits to the open source community include a commercial-grade framework that is free of charge.

WHY MYSQL?

- Ease of Use
- Low Cost
- Availability of Support
- Portability
  - MySQL can be used on many different Unix systems as well as under Microsoft Windows.

RDBMS

- R for Relational:
  - It's called relational because all the data is stored into different tables and relations are established using primary keys or other keys known as foreign keys.
- DB for Database
  - It is a collection of data stored in different tables to be easy to be accessed.
- MS for Management System:
  - An application has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

RDBMS

- 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.
  - Interprets an SQL query and combines information from various tables.

RELATIONAL DATABASE CONCEPTS

- Tables
  - A Collection of related data. The table has a name; a number of columns and a number of rows, a table in a database looks like a simple spreadsheet.
- Columns
  - Each column in the table has a unique name and contains different data. Additionally, each column has an associated data type as an integer, strings or Timestamp and so on . Columns are sometimes called fields or attributes.
- Rows
  - are a group of related data. Because of the tabular format, each row has the same attributes. Rows are also called records or tuples.

RELATIONAL DATABASE CONCEPTS

- 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 (Composite Key)
  - 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.

RELATIONAL DATABASE CONCEPTS

- Schemas
  - The complete set of table designs for a database is called the database schema. It is akin to a blueprint for the database. A schema should show the tables along with their columns, and the primary key of each table and any foreign keys. A schema does not include any data, but you might want to show sample data with your schema to explain what it is for.

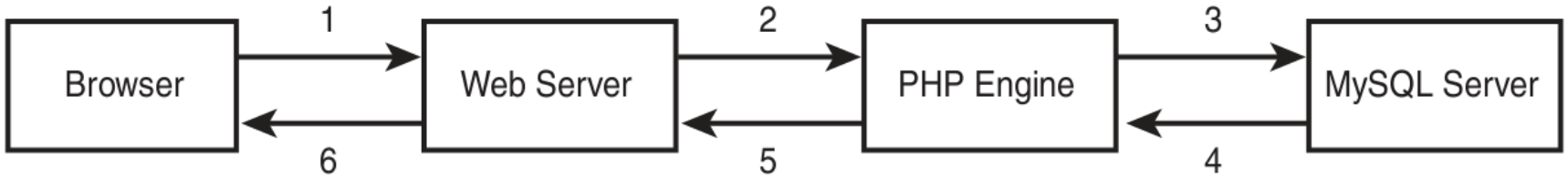
DESIGNING YOUR WEB DATABASE

- Think About the Real-World Objects You are Modeling.
- Avoid Storing Redundant Data.
- Use Atomic Column Values.
- Choose Sensible Keys.
- Avoid Designs with Many Empty Attributes.

MySQL ERD

WEB DATABASE ARCHITECTURE

- This system consists of two objects: a web browser and a web server. A communication link is required between them. A web browser makes a request of the server. The server sends back a response.

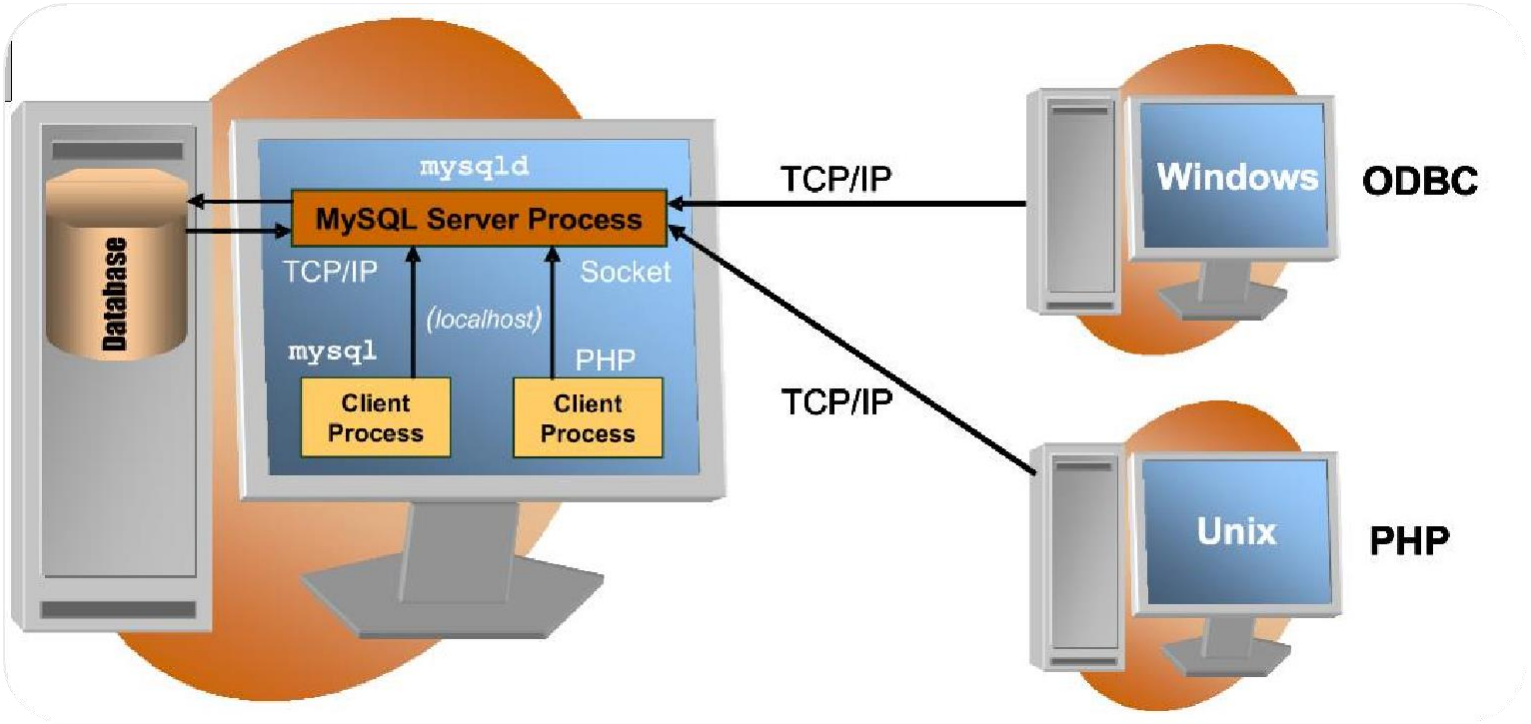


INSTALLATION

- https://dev.mysql.com/downloads/installer/

CLIENT/SERVER ARCTICTURE

- MySQL operates using a client/server architecture.
- The server (**`mysqld`**) runs on the machine where your databases are stored. It listens for client requests on port 3306.



CREATING DATABASES

- At the MySQL command prompt, type

```
CREATE DATABASE `db_name`;
```

- You should see a response like this :

```
Query OK, 1 row affected (0.0 sec).
```

- To use a reserved word as a database, table, column, or index identifier, use (``)
- If unquoted, an identifier must follow these rules:
  - An identifier may contain all alphanumeric characters, the underline character ('_'), and the dollar sign ('$').
  - An identifier cannot consist entirely of digits.
  - An identifier cannot be a reserved word like (select,order,..etc)

USING THE RIGHT DATABASE

- To Select a database

```
use db_name;
```

- You can also use qualified names that identify both the database and the table:

```
SELECT * FROM db_name.tbl_name;
```

- To Know which database is selected:

```
SELECT DATABASE();
```

DROPPING AND ALTERING DATABASE

- To drop a database simply type:

```
DROP DATABASE db_name;
```

- To Alter a databse:

```
ALTER DATABASE [db_name];
```

Alter is used to modify an existing table, like adding a row or column. **Alter** preserves the existing data.

Drop is used to remove a table completely.

- If you omit the database name, It applies to the default database.
- You cannot use `ALTER DATABASE` to rename a database. To do that, create a database with the new name, reload the data into the new database, and drop the old database
- To list all the databases:

```
SHOW DATABASES;
```

CREATING TABLES

- The general form of a `CREATE TABLE` statement is:

```
CREATE TABLE <table> (
  <column name> <column type> [<column options>],
  [<column name> <column type> [<column options>],…,]
  [<list of table constraints and or indexes>] )
  [<table options>]
```

- Show table definition:

```
SHOW CREATE TABLE tablename
```

To create an empty copy of an existing table:

```
CREATE TABLE new_tbl_name LIKE tbl_name;
```

CREATING TABLES

- To create a temporary table which disappears automatically when your connection to the server terminates:

```
CREATE TEMPORARY TABLE tbl_name;
```

CREATING TABLES

- As Example:

```
mysql> CREATE TABLE CountryLanguage (
->      CountryCode CHAR(3) NOT NULL,
->      Language CHAR(30) NOT NULL,
->      IsOfficial ENUM('True', 'False') NOT
  NULL DEFAULT 'False',
->      Percentage FLOAT(3,1) NOT NULL,
->      PRIMARY KEY(CountryCode, Language)
-> )
```

COLUMN OPTIONS

- A table must have at least one column.
- Each column has to have a name and a data type.
- In addition, a column definition can have several options as
  - `NULL` and `NOT NULL`
  - `DEFAULT`
  - `AUTO_INCREMENT`
  - Constraints `(PRIMARY KEY, UNIQUE, FOREIGN KEY)`

COLUMN DATA TYPES

- In MySQL the data types available can be broken down into four major categories:

Type	Description
Numeric	Numeric values (Integers, Floating-Point and Fixed-Point)
Character	Text strings
Binary	Binary data strings
Temporal	Time and dates

NUMERIC DATA TYPES

- Integer

Type	Range	Storage (Bytes)	Description
`TINYINT[(M)]`	–127..128 or 0..255	1	Very small integers
`BIT`			Synonym for `TINYINT`
`BOOL`			Synonym for `TINYINT`
`SMALLINT[(M)]`	–32768..32767 or 0..65535	2	Small integers
`MEDIUMINT[(M)]`	–8388608.. 8388607 or 0..16777215	3	Medium-sized integers
`INT[(M)]`	$-2^{31}..2^{31}-1$ or $0..2^{32}-1$	4	Regular integers
`INTEGER[(M)]`			Synonym for `INT`
`BIGINT[(M)]`	$-2^{63}..2^{63}-1$ or $0..2^{64}-1$	8	Big integers

NUMERIC DATA TYPES

- Float

Type	Range	Storage (bytes)	Description
FLOAT(*precision*)	Depends on precision	Varies	Can be used to specify single or double precision floating-point numbers.
FLOAT[(M,D)]	±1.175494351E-38 ±3.402823466E+38	4	Single precision floating-point number. These numbers are equivalent to FLOAT(4) but with a specified display width and number of decimal places.
DOUBLE[(M,D)]	±1.7976931348623157E +308 ±2.2250738585072014E -308	8	Double precision floating-point number. These numbers are equivalent to FLOAT(8) but with a specified display width and number of decimal places.
DOUBLE PRECISION[(M,D)]	As above		Synonym for DOUBLE[(M, D)].
REAL[(M,D)]	As above		Synonym for DOUBLE[(M, D)].
DECIMAL[(M[,D])]	Varies	M+2	Floating-point number stored as char. The range depends on M, the display width.
NUMERIC[(M,D)]	As above		Synonym for DECIMAL.
DEC[(M,D)]	As above		Synonym for DECIMAL.

CHARACTER DATA TYPES

- Character

Type	Range	Description
[NATIONAL] CHAR(*M*) [BINARY | ASCII | UNICODE]	0 to 255 characters	Fixed-length string of length *M*, where *M* is between 0 and 255. The NATIONAL keyword specifies that the default character set should be used. This is the default in MySQL anyway, but is included because it is part of the ANSI SQL standard. The BINARY keyword specifies that the data should be treated as case sensitive. (The default is case sensitive.) The ASCII keyword specifies that the latin1 character set will be used for this column. The UNICODE keyword specifies that the ucs character set will be used.
CHAR		Synonym for CHAR(1).
[NATIONAL] VARCHAR(*M*) [BINARY]	1 to 255 characters	Same as above, except they are variable length.

BINARY DATA TYPES

- Text and Binary

Type	Maximum Length (Characters)	Description
TINYBLOB	$2^8 - 1$ (that is, 255)	A tiny binary large object (BLOB) field
TINYTEXT	$2^8 - 1$ (that is, 255)	A tiny TEXT field
BLOB	$2^{16} - 1$ (that is, 65,535)	A normal-sized BLOB field
TEXT	$2^{16} - 1$ (that is, 65,535)	A normal-sized TEXT field
MEDIUMBLOB	$2^{24} - 1$ (that is, 16,777,215)	A medium-sized BLOB field
MEDIUMTEXT	$2^{24} - 1$ (that is, 16,777,215)	A medium-sized TEXT field
LONGBLOB	$2^{32} - 1$ (that is, 4,294,967,295)	A long BLOB field
LONGTEXT	$2^{32} - 1$ (that is, 4,294,967,295)	A long TEXT field

	Values in Set	
ENUM('*value1*', '*value2*',...)	65,535	Columns of this type can hold only *one* of the values listed or NULL.
SET('value1', 'value2',...)	64	Columns of this type can hold a set of the specified values or NULL.

TEMPORAL DATA TYPES

- Date and Time

Type	Range	Description
`DATE`	1000-01-01 9999-12-31	A date. Will be displayed as `YYYY-MM-DD`.
`TIME`	-838:59:59 838:59:59	A time. Will be displayed as `HH:MM:SS`. Note that the range is much wider than you will probably ever want to use.
`DATETIME`	1000-01-01 00:00:00 9999-12-31 23:59:59	A date and time. Will be displayed as `YYYY-MM-DD HH:MM:SS`.
`TIMESTAMP[(M)]`	1970-01-01 00:00:00	A timestamp, useful for transaction reporting. The display format depends on the value of *M* (see Table 9.8, which follows).
	Sometime in 2037	The top of the range depends on the limit on Unix.
	timestamps.	
`YEAR[(2|4)]`	70–69 (1970–2069) 1901–2155	A year. You can specify two- or four-digit format. Each has a different range, as shown.

SHOW & DESCRIBE

- You can see more information about a particular table

```
DESCRIBE table_name;
DESC table_name;
```

- You can view the tables in the database by typing;

```
SHOW TABLES;
SHOW TABLES FROM database_name;
```

- Display information about columns or indexes in a table

```
SHOW COLUMNS FROM tbl_name;
SHOW INDEX FROM tbl_name;
```

INSERT Command

Person table

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo

✓ INSERT INTO "table_name" VALUES ("value1", "value2", ...)

- **Insert a New Row:**

INSERT INTO Person VALUES ('Saleh', 'Ahmed', 'Moharam bak', 'Alex.')

Person table

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Saleh	Ahmed	Moharam bak.	Alex.

INSERT Command (cont.)

- Insert Data in Specified Columns:

Person table

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo

- **Insert a New Row:**

INSERT INTO Person (LastName, City) VALUES ('Hassan', 'Cairo')

Person table

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Hassan			Cairo.

Update Command

- ✓ UPDATE "table_name"
  SET "column_1" = {new value}
  [WHERE {condition}]

Example (1)

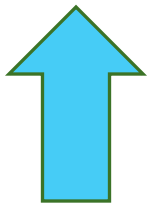
```
UPDATE Person
SET City= 'cairo'
```

All records will be updated

Example (2)

```
UPDATE Person
SET City= 'cairo'

Where FirstName = ‘Ahmed’
```



Only records with first name ‘Ahmed’ will be updated

Update Command (cont.)

✓ Update several Columns in a Row:

LastName	FirstName	Address	City
El-Sayed	Mohamed	Nasr City	Cairo
Saleh	Ahmed	Moharam bak.	Alex.

```
UPDATE Person
SET    Address = '241 El-haram ',  City = 'Giza'
WHERE  LastName = 'El-Sayed'
```

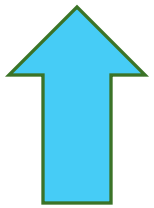
LastName	FirstName	Address	City
El-Sayed	Mohamed	241 El-haram	Giza
Saleh	Ahmed	Moharam bak.	Alex.

Delete Command

- DELETE FROM "table_name"
  [WHERE {condition}]

Example (1)

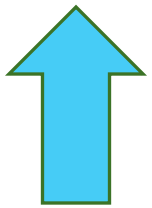
DELETE FROM Person



All records will be deleted

Example (2)

DELETE FROM Person
Where FirstName = ‘Ahmed’



Only records with first name ‘Ahmed’ will be deleted

Simple Queries

Select <attribute list >
From < table list>
[Where <condition>]

- ✓ select *
  from department;

- ✓ select emp_id, emp_name, dept_id
  from employee;

- ✓ select distinct dept_id
  from employee;

Simple Queries (cont.)

```
Select  dept_id, dept_name
from    department
where   location = ‘Cairo’;
```

Comparison Conditions

- = Equal.
- > greater than.
- >= greater than or equal.
- < less than.
- <= less than or equal.
- <>not equal.

```
Select   last_name, salary
from     employee
where  salary >1000
```

Logical Conditions

- AND.

```
Select   last_name, salary
from     employee
where   city = 'cairo' and salary > 1000;
```

- OR.

```
Select   last_name, salary
from     employee
where   city = 'cairo' OR salary > 1000;
```

- NOT.

```
Select   emp_id, last_name, salary, manager_id
From     employee
where    manager_id NOT IN (100, 101, 200);
```

Other Comparison Conditions

- Between …… AND ….. (between two values - Inclusive).

```
Select   last_name, salary
from     employee
where    salary between 1000 and 3000;
```

- IN (set) (Match any of a list of values)

```
Select    emp_id, last_name, salary, manager_id
From      employee
where     manager_id IN (100, 101, 200);
```

- Like (Match a character Pattern)

```
Select   first_name
from     employee
where    first_name Like '_s%';
```

Arithmetic Expressions

```
Select   last_name, salary, salary + 300
from     employee;
```

- Order of precedence: * , / , +, -
- You can enforce priority by adding parentheses.

```
Select    last_name, salary, 10 * (salary + 300)
from      employee;
```

Order by Clause

- It is used to sort results either in ascending or descending order.

✓
```
Select      fname, dept_id, hire_date
From        employee
Order by    hire_date  [ ASC ];
```

✓
```
Select      fname, dept_id, hire_date
From        employee
Order by    hire_date  DESC;
```

✓
```
Select      fname, dept_id, salary
From        employee
Order by    dept_id, Salary  DESC;
```

Aggregate Functions

COUNT , SUM , MAX, MIN, AVG

```
Select   count (*)
From     employee

Select   count (Address)
From     employee

Select   Sum (salary) , Max (salary), Min (salary), Avg (salary)
From     employee
```

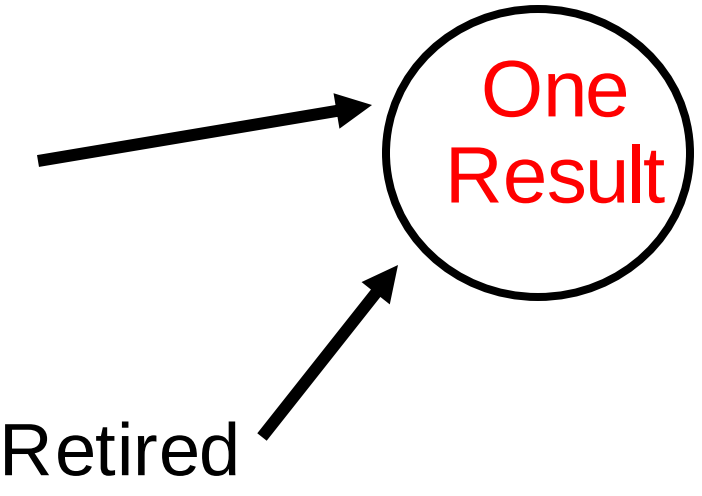
Union operator

- It’s used to combine results from different queries.
- Display names for all employees who is or was working in the organization.

```
SELECT Name FROM Employees

UNION

SELECT Name FROM Employees_Retired
```



Union operator (cont.)

- Number of columns and data types in the different queries must be the same.
- The UNION operator selects only distinct values by default.
- To allow duplicate values, use UNION ALL.

Sub-Queries

- Find the names of the employees working in "sales" department.

```
Select   name
from     Employee
Where    Dno in
                    ( select Dnumber
                      from Department
                      where dname = 'sales')
```

Sub-Queries (cont.)

- Display department name with the highest paid employee.

1- Get the Highest Salary.
2- Get Deptno for this Employee.
3- Get Department name.

```
SELECT dname FROM dept
WHERE deptno IN (SELECT deptno FROM emp
WHERE sal = (SELECT MAX(sal) FROM EMP));
```

Sub-Queries (cont.)

- Find the names of employees whose salary is greater than the salary of the employees in department 5.

```
Select    Lname , Fname
From      Employee
Where     salary > ( select   Max(salary)  from   Employee
                     where  Dno=5)
```

Subqueries

Name	Sample Syntax	Description
ANY	SELECT c1 FROM t1 WHERE c1 > ANY (SELECT c1 FROM t2);	Returns true if the comparison is true for any of the rows in the subquery.
IN	SELECT c1 FROM t1 WHERE c1 IN (SELECT c1 from t2);	Equivalent to =ANY.
SOME	SELECT c1 FROM t1 WHERE c1 > SOME (SELECT c1 FROM t2);!	Alias for ANY; sometimes reads better to the human ear
ALL	SELECT c1 FROM t1 WHERE c1 > ALL (SELECT c1 from t2);	Returns true if the comparison is true for all of the rows in the subquery.

Data Definition Language

- Create.
- Drop.
- Alter.

Create Command

- Create table "table_name”

  (“column name” data type, “column name” data type, ...)

Example (1)

```
CREATE TABLE customer
(ID int  Not Null, First_Name char(50), Last_Name char(50),
 City char(25), Birth_Date date, Primary key (ID));
```

Example (2)

```
CREATE TABLE customer
(ID int Primary key, First_Name char(50), Last_Name

 char(50),  City char(25), Birth_Date date);
```

Drop command

- ✓ Drop table “table name”
- ✓ Drop table Customer

Alter command

- ✓ ALTER TABLE table_name ADD column_name datatype
- ✓ ALTER TABLE table_name DROP COLUMN column_name

- Example:
- ✓ ALTER TABLE Customer ADD Address char(40)
- ✓ ALTER TABLE Customer DROP COLUMN Address