MySQL

NOW

MySQL
DDL
Triggers
Data Types
DML

OPEN SOURCE
DEPARTMENT

Information
Technology
Institute

Day 1 Contents

- Why MySQL?
- History of MySQL.
- RDBMS concepts and terminology
- Installing & Logging.
- Creating Users and maintaining their privileges.
- DDL (CREATE DB/TABLE, ALTER, DROP,….)
- Columns Data Types
- Indexing

Why MySQL?

- Designed for high volume environments.
- Cross platform
- Low / No Cost.
- Stability
- Open Source

https://www.mysql.com/why-mysql/case-studies/

https://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems

History of MySQL

- Original development of MySQL by Michael Widenius and David Axmark beginning in 1994 and First internal release on 23 May 1995.
- He has a daughter My (after whom MySQL was named)
- Michael Widenius founding the MySQL AB (company in Swedish)

https://en.wikipedia.org/wiki/Michael_Widenius

https://dev.mysql.com/doc/refman/5.7/en/history.html

History of MySQL

- Sun Microsystems acquired MySQL AB on 26 February 2008 for approximately $1 billion.
- Oracle Corporation then acquired Sun in 2010 for $5.6 billion.
- The day Oracle announced the purchase of Sun, Michael "Monty" Widenius forked MySQL, launching MariaDB, and took a swath of MySQL developers with him
- MariaDB is a community-developed fork of the MySQL relational database management system intended to remain free under the GNU GPL

History of MySQL

- **MySQL Enterprise Server**

  Available only with the MySQL Enterprise subscription.

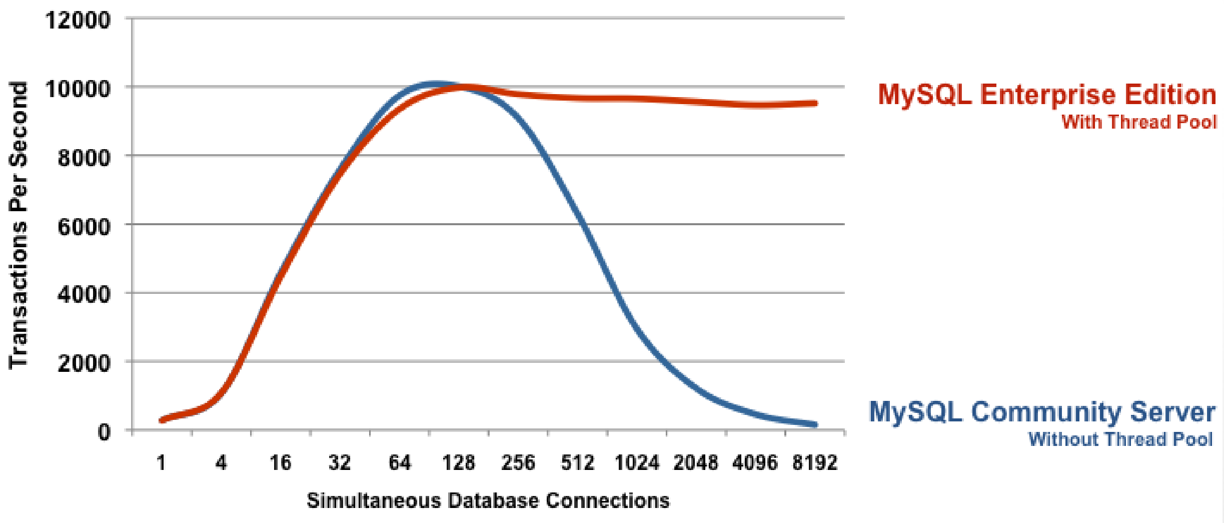
  Annual Subscription USD 5,000-10,000

- **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. that is free of charge.

Why MySQL?

- Performance
  - http://www.mysql.com/why-mysql/benchmarks/



RDBMS

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.

Relational database concepts

- 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.
- Values
  - Each row consists of a set of individual values that correspond to columns. Each value must have the data type specified by its column.

Relational database concepts

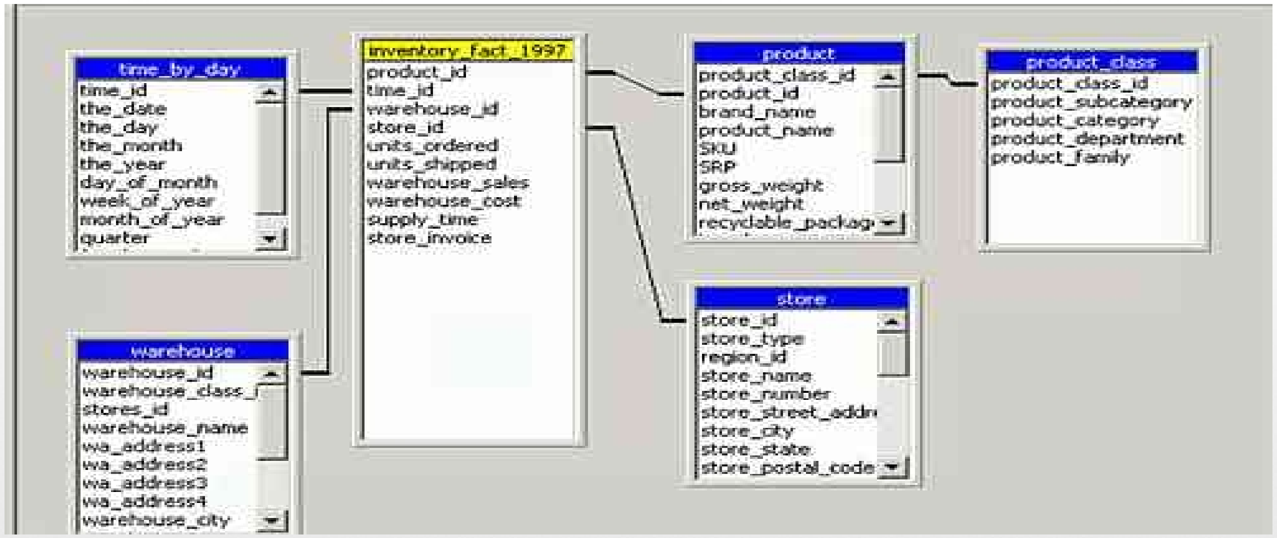
- Primary Key
  - A primary key is unique. A key value can not occur twice in one table. With a key, you can find at most one row.
- Foreign Key
  - A foreign key is the linking pin between two tables.
- Referential Integrity
  - Referential Integrity makes sure that a foreign key value always points to an existing row.

Relational database concepts

- Index
  - it is a data structure that improves the speed of data retrieval operations on a database table
  - Disadvantages: Storage Size & Insertion Time

Relational database concepts

- Schemas
  - 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.



Relational database

- RDBMS software:
  - 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.
  - C/C++, Java Interface

ERD

Entity Relationship Diagram

ERD Symbols and Notations

- Entity
  - An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. Entities are represented in ER diagrams by a rectangle and named using *singular* nouns.

ERD Symbols and Notations

- Attribute
  - An attribute is a property or characteristic of an entity, relationship. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item.
- Multivalued Attribute
  - If an attribute can have more than one value it is called an multivalued attribute. For example a person entity can have multiple hobbies values.

ERD Symbols and Notations

- Derived Attribute

An attribute based on another attribute. This is found rarely in ER diagrams. Such as calculations

SUCH AS call duration, age.

- Relationship

A relationship describes how entities interact.

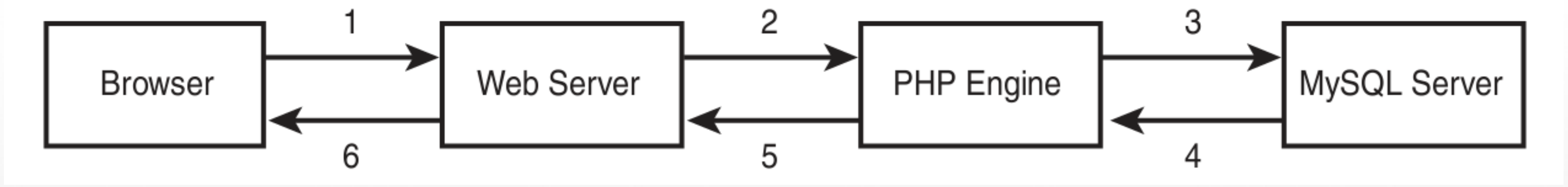
Example: Student, address, track, staff, courses, desk

Designing Your Database

- Simple tables
  - that describe a real-world object. They might also contain keys to other simple objects with which they have a one-to-one or one-to-many relation-ship.
- Linking tables
  - that describe a many-to-many relationship between two real objects.

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.



Installing

- For Ubuntu/Debian :

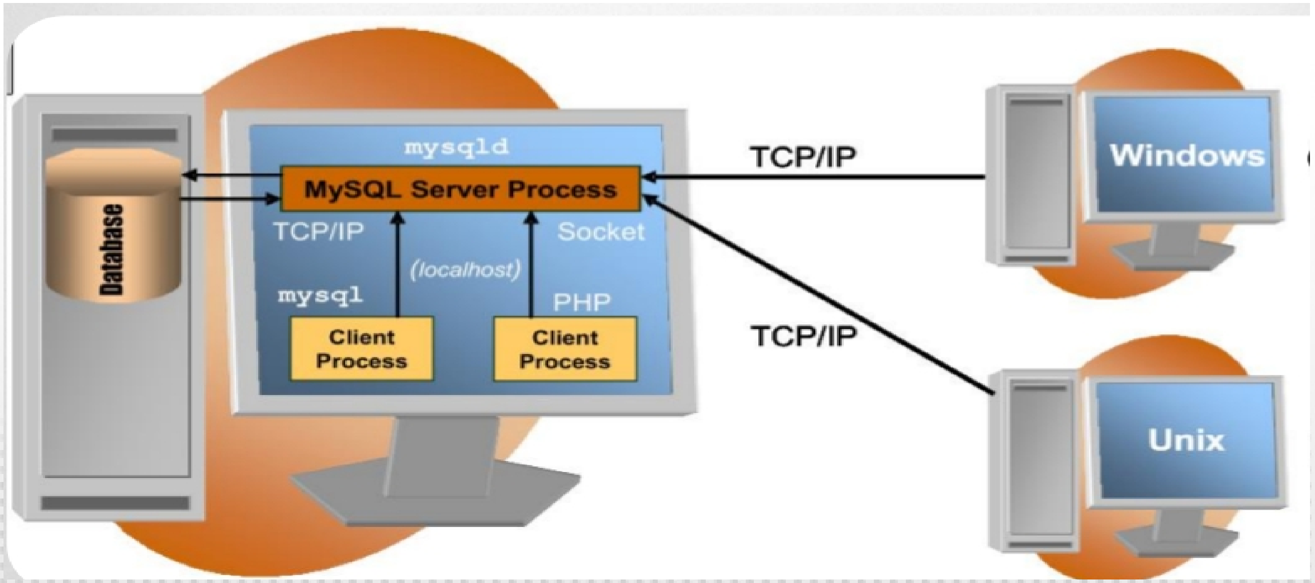
```
$ sudo apt-get install mariadb-server
```

- For CentOS/Red Hat Distros:

```
$ sudo yum install mariadb-server
$ sudo systemctl enable mariadb
$ sudo systemctl start mariadb
```

MySQL Terminology - 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.



MySQL Terminology

- Mysql is an interactive client that lets you issue queries and see the results.
- Two administrative clients are:
  - mysqldump, a backup program that dumps table contents into a file.
  - mysqladmin, which enables you to check on the status of the server and performs other administrative tasks such as telling the server to shut down.

How MySQL stores data?

- A MySQL server can store several databases
- Databases are stored as directories
  - Default is at /usr/local/mysql/var/
  - Or /var/lib/mysql (Ubuntu /Debian)
- Tables are stored as files inside each database (directory)

Installing & Logging

- To Log into MySQL

```
$ mysql -h hostname -u username –p
```

- you should get a response something like this:

```
Enter password:
```

INVOKING CLIENT PROGRAMS

- You can exit mysql client using

```
mysql> exit
mysql>quit
mysql>\q
```

- The default statement terminators are: \g and ;

- The \G is also recognized s statement terminator, It displays the result in a vertical style:

```
mysql> SELECT VERSION(), DATABASE()\G
```

THE mysql PROMPTS

- There are several types of prompts, The following table shows each of these prompts

Option	Meaning
mysql>	Ready for new statement
->	Continue entering statement until a terminator like is entered
‘>	Waiting for end of single-quoted string (')
“>	Waiting for end of double-quoted string or identifier (")
`>	Waiting for end of backtick-quoted identifier (`)
/*>	Waiting for end of C-style comment (*/)

Installing & Logging

If all goes well, you should see a response something like this:

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.41-MariaDB MariaDB Server

Copyright (c) 2000, 2014, Oracle, MariaDB Corporation Ab and
   others.

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

MariaDB [(none)]>
```

Creating Users

- Create One or more users:

```
CREATE USER account [IDENTIFIED BY [PASSWORD] 'password']
[,account [IDENTIFIED BY[PASSWORD] 'password'] ] ...

CREATE USER ‘iti' IDENTIFIED BY ‘os123';
```

Try @ Lab

?

Get Time, Date,

Current user,

MySQL Version

Using prompt?

Creating Databases and Users

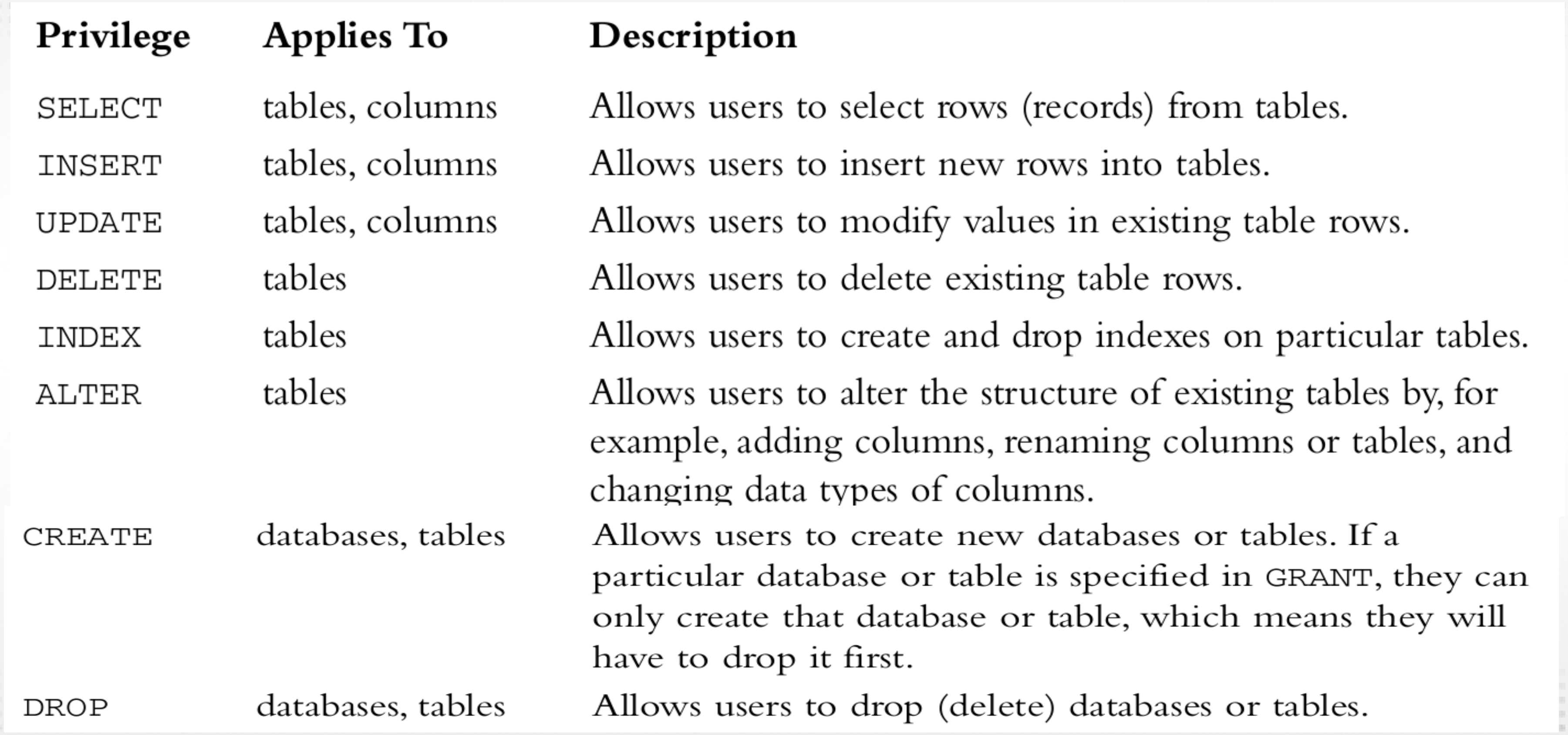
- The GRANT and REVOKE commands enable you to give rights to and take them away from MySQL users at these levels of privilege:
- Database
- Table
- Column

Creating Databases and Users

- The GRANT command creates users and gives them privileges. The general form is

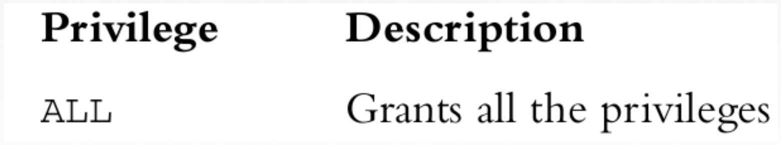
```
GRANT privileges [columns]
ON item
TO user_name
[WITH limit_options]
```

Users Privileges



Privilege	Applies To	Description
SELECT	tables, columns	Allows users to select rows (records) from tables.
INSERT	tables, columns	Allows users to insert new rows into tables.
UPDATE	tables, columns	Allows users to modify values in existing table rows.
DELETE	tables	Allows users to delete existing table rows.
INDEX	tables	Allows users to create and drop indexes on particular tables.
ALTER	tables	Allows users to alter the structure of existing tables by, for example, adding columns, renaming columns or tables, and changing data types of columns.
CREATE	databases, tables	Allows users to create new databases or tables. If a particular database or table is specified in GRANT, they can only create that database or table, which means they will have to drop it first.
DROP	databases, tables	Allows users to drop (delete) databases or tables.

Special Privileges



Privilege	Description
`ALL`	Grants all the privileges

Limit Options

- With LIMIT OPTIONS

```
MAX_QUERIES_PER_HOUR n

MAX_UPDATES_PER_HOUR n

MAX_CONNECTIONS_PER_HOUR n
```

Creating Databases and Users

- The REVOKE Command : The opposite of GRANT is REVOKE. You use it to take privileges away from a user. It is similar to GRANT in syntax:

```
REVOKE privileges [(columns)]
ON item
FROM user_name
```

- You can revoke all other privileges by adding:

```
REVOKE All , GRANT OPTION
FROM ‘user_name’
```

Examples

```
  grant all
-> on *
-> to iti;

  revoke all privileges, grant option
-> from iti;
```

Creating Databases

- General Form :

```
CREATE DATABASE [IF NOT EXISTS] db_name
[CHARACTER SET charset] ;
```

  - A character set is a set of symbols and encodings.
  - The most used Types:

    - utf8
    - ascii

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).
```

- Statement must end with (‘;’). The semicolon tells mysql that the statement is complete. Another way to terminate a statement is to use \g (“go”) rather than a semicolon.

Character Set

- Character sets can be specified at the server, database, table, column:

```
CREATE DATABASE db_name CHARACTER SET charset;



CREATE TABLE tbl_name (...) CHARACTER SET charset;


c CHAR(10) CHARACTER SET charset
```

Show Database Definition

- To see the definition for an existing database:

```
SHOW CREATE DATABASE db_name \G
************* 1. row *************
       Database: db_name
Create Database: CREATE DATABASE `db_name`
DEFAULT CHARACTER SET latin1 */
1 row in set (0.00 sec)
```

Using the Right Database

- To Select a database

```
use db_name;
```

- Alternatively, you can do that when you log in:

```
mysql -D dbname -h hostname -u username –p
```

- 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 datbse:

```
ALTER DATABASE [db_name] [CHARACTER SET charset];
```

If you omit the database name, ALTER DATABASE applies to the default database.

Creating Tables

- The general form of a CREATE TABLE statement is:

```
CREATE TABLE [IF NOT EXISTS] tbl_name (column_specs);
```

- Show table definition:

```
SHOW CREATE TABLE t;
```

- To create an empty copy of an existing table:

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

Creating Tables

- To create an empty copy of a table and then populate it from the original table:

```
CREATE TABLE new_tbl_name LIKE tbl_name;

INSERT INTO new_tbl_name SELECT * FROM tbl_name;
```

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

```
CREATE TEMPORARY TABLE tbl_name;
```

Creating Database Tables

- As Example:

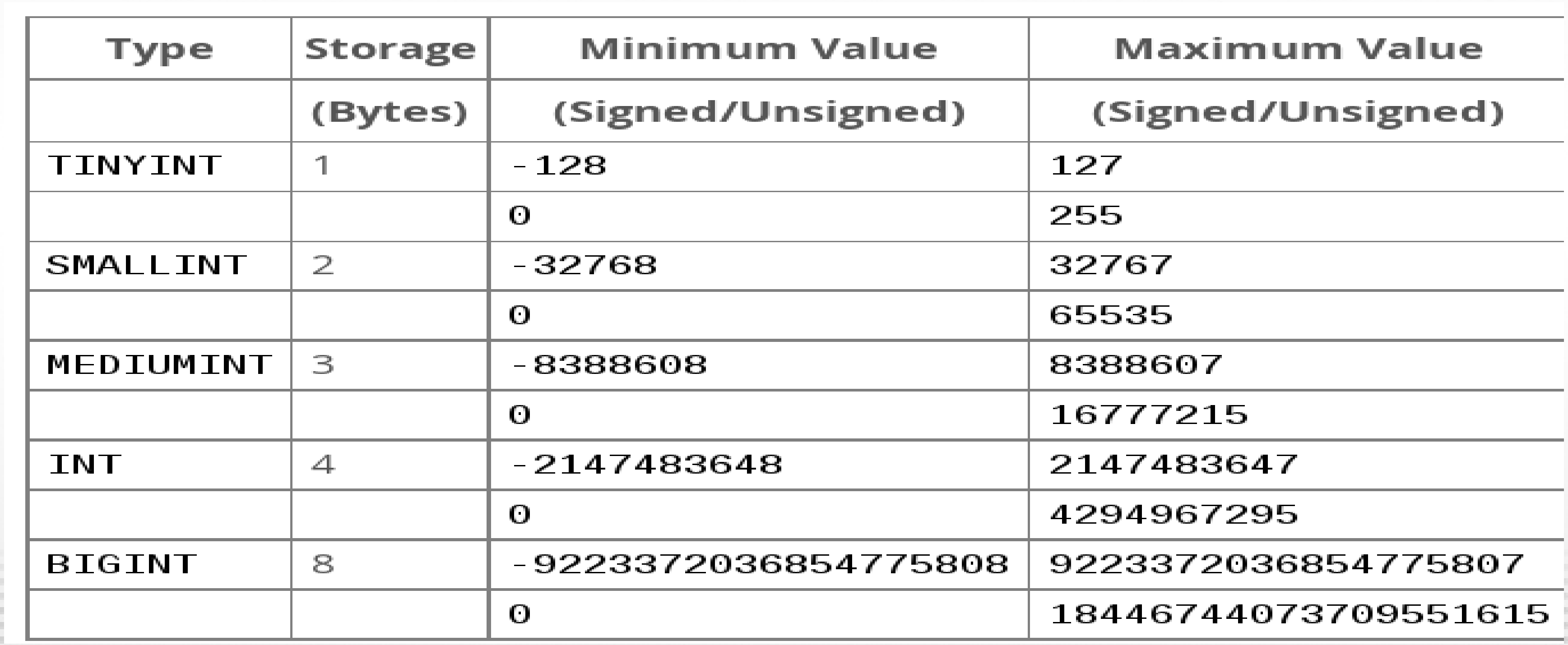
```
create table customers
(       customerid int unsigned primary key,
        name char(50) ,
        address char(100) ,
        city char(30)
);
create table orders
(       orderid int primary key,
        customerid int ,
        amount float(6,2),
        date date
);
```

Column Data Types

- The three basic column types in MySQL are numeric, date and time, and string.
- Each of the three types comes in various storage sizes. When you are choosing a column type, the principle is generally to choose the smallest type that your data will fit into.

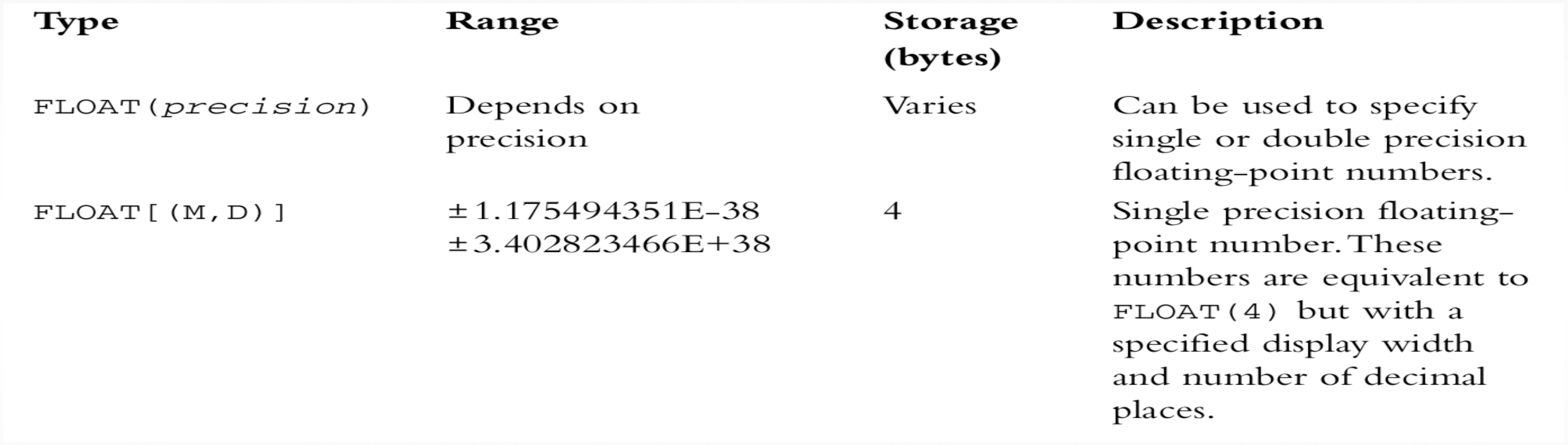
Column Data Types

- **Numeric Types**
  - The numeric types are either integers or float

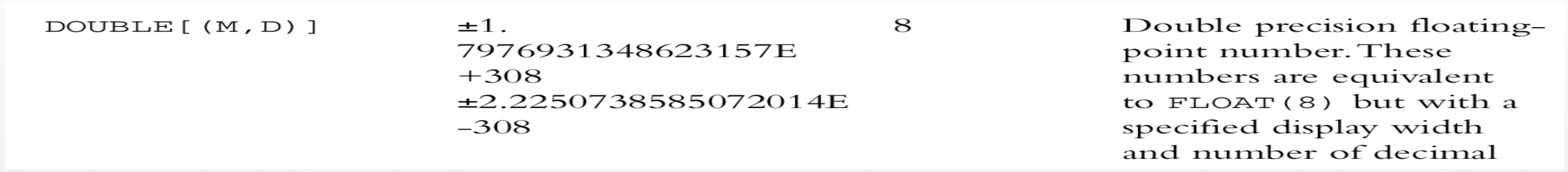


Type	Storage	Minimum Value	Maximum Value
	(Bytes)	(Signed/Unsigned)	(Signed/Unsigned)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

Column Data Types



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.



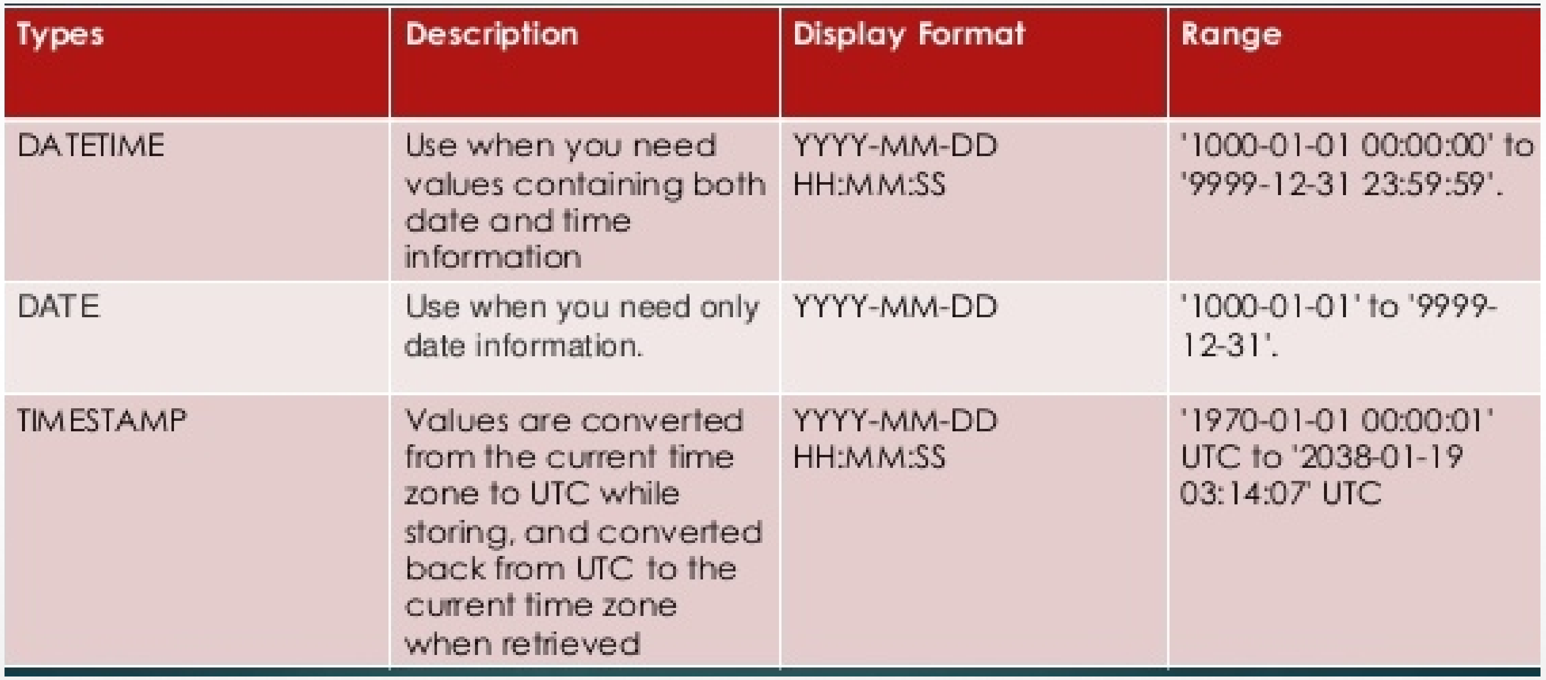
`DOUBLE[(M,D)]`	±1. 7976931348623157E +308 ±2.2250738585072014E –308	8	Double precision floating-point number. These numbers are equivalent to `FLOAT(8)` but with a specified display width and number of decimal

Column Data Types

Name	Storage Size	Description	Range
smallserial	2 bytes	small autoincrementing integer	1 to 32767
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

Column Data Types

- Date and Time Types



Types	Description	Display Format	Range
DATETIME	Use when you need values containing both date and time information	YYYY-MM-DD HH:MM:SS	'1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
DATE	Use when you need only date information.	YYYY-MM-DD	'1000-01-01' to '9999-12-31'.
TIMESTAMP	Values are converted from the current time zone to UTC while storing, and converted back from UTC to the current time zone when retrieved	YYYY-MM-DD HH:MM:SS	'1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC

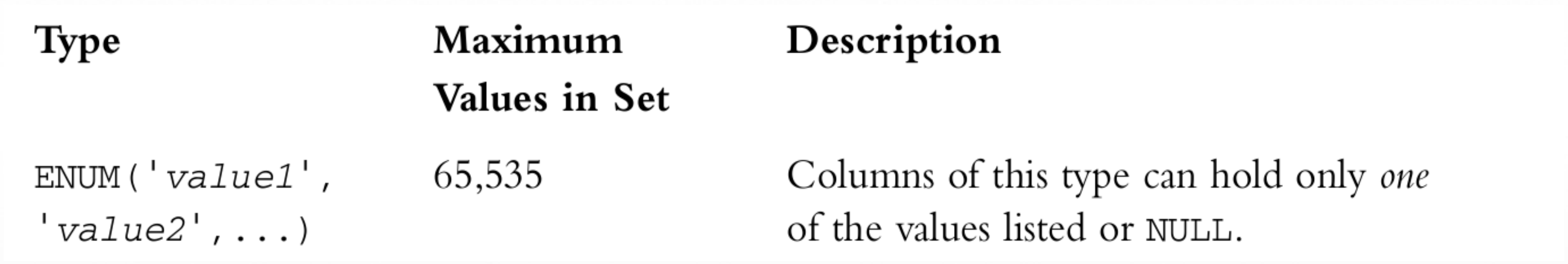
Character data types

- An attempt to store a longer string will result in an error, unless the excess characters are all spaces, in this case string will be truncated to the maximum
- If the string is shorter than the declared length, values of type character will be space-padded; values of type character varying will simply store the shorter string.

Name	Description
character varying(n), varchar(n)	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

Column Data Types

- ENUM



Type	Maximum Values in Set	Description
`ENUM('value1', 'value2',...)`	65,535	Columns of this type can hold only *one* of the values listed or `NULL`.

Show & Describe

- You can see more information about a particular table

```
DESCRIPE table_name;
DESC table_name;
```

Show & Describe

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

```
SHOW TABLES;
```

- you can also use show to see a list of databases by typing

```
SHOW DATABASES;
```

- Display information about columns or indexes in a table

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

Creating Indexes

- If you find that you are running many queries on a column that is not a key, you may want to add an index on that column to improve performance.

```
CREATE INDEX index_name
ON table_name (index_column_name , ...])
```

Creating Indexes

- To create an index during table creation:

```
CREATE TABLE tbl_name
(
... column definitions ...
INDEX index_name (index_columns),
UNIQUE index_name (index_columns),
PRIMARY KEY (index_columns),
FOREIGN KEY (index_column) REFERENCES tbl_name
(index_column)
..
);
```

Creating Indexes

- Another way to create an index by altering the table:

```
ALTER TABLE tbl_name ADD INDEX index_name (index_columns);
ALTER TABLE tbl_name ADD PRIMARY KEY (index_columns);
ALTER TABLE tbl_name ADD UNIQUE index_name
(index_columns);
ALTER TABLE tbl_name ADD FOREIGN KEY (index_col)
REFERENCES tbl_name  (index_columns);
```

Dropping Indexes

- To drop an index:

```
DROP INDEX index_name ON tbl_name;
```

You can also use Alter:

```
ALTER TABLE tbl_name DROP INDEX index_name;
ALTER TABLE tbl_name DROP PRIMARY KEY;
ALTER TABLE tbl_name DROP FOREIGN KEY constraint_name
```