MySQL

For Developers

COURSE MATERIALS

You can access the course materials via this link

http://goo.gl/3R3eWh

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
- Source Code
- 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.

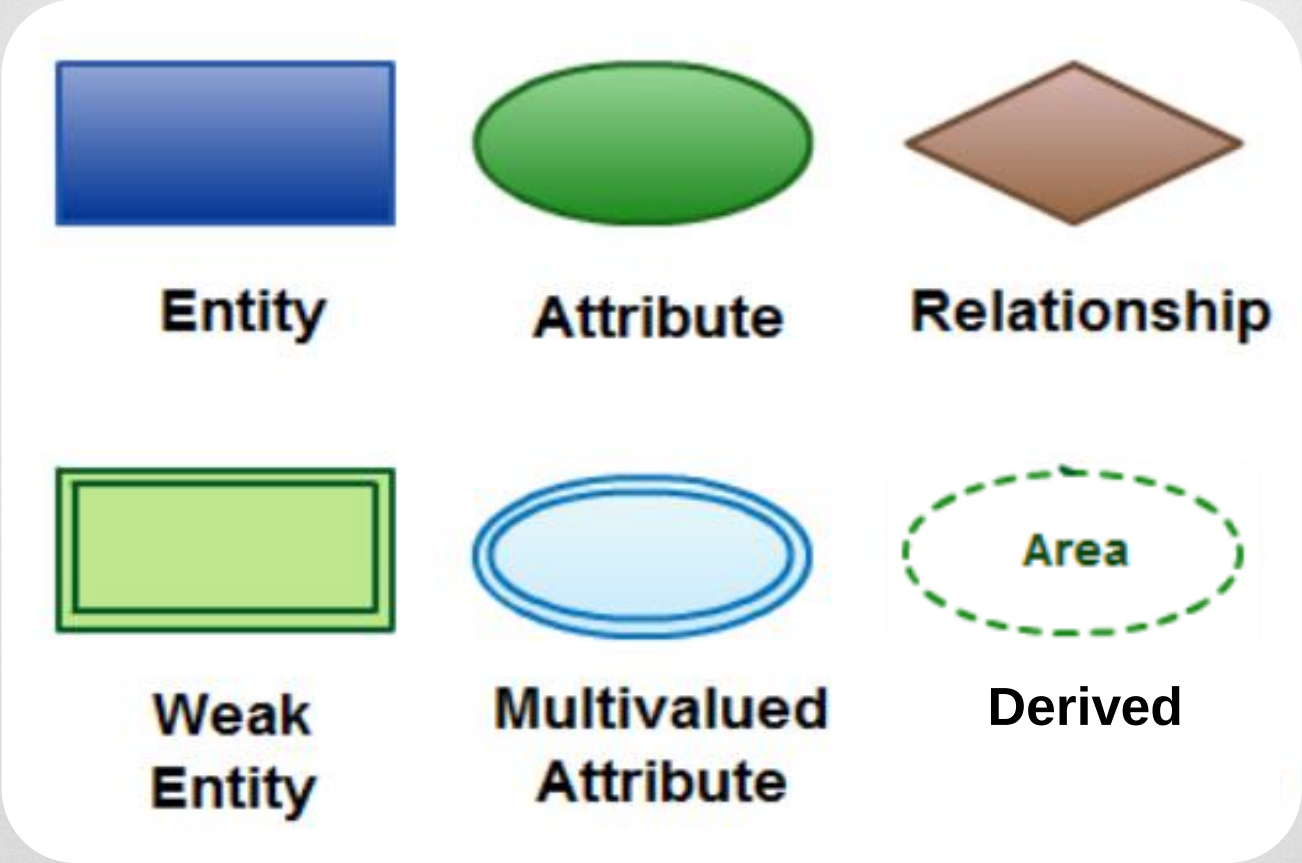
RELATIONAL DATABASE CONCEPTS

- 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.

ERD

Entity Relationship Diagram

ERD SYMBOLS AND NOTATIONS



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.
- Weak Entity
  - A weak entity is an entity that depends on the existence of another entity. In more technical terms it can defined as an entity that cannot be identified by its own attributes. Such as order -> order items

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

ERD SYMBOLS AND NOTATIONS

- Relationship
  - A relationship describes how entities interact.
- Recursive Relationship
  - If the same entity participates more than once in a relationship it is known as a recursive relationship.

RELATIONAL DATABASE CONCEPTS

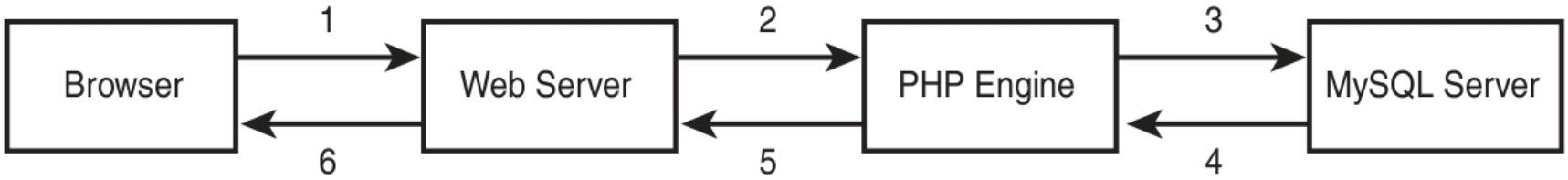
- Relationships types
  - one-to-one relationship.
  - one-to-many relationship.
  - many-to-many relationship.

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.

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

- For Ubuntu/Debian :

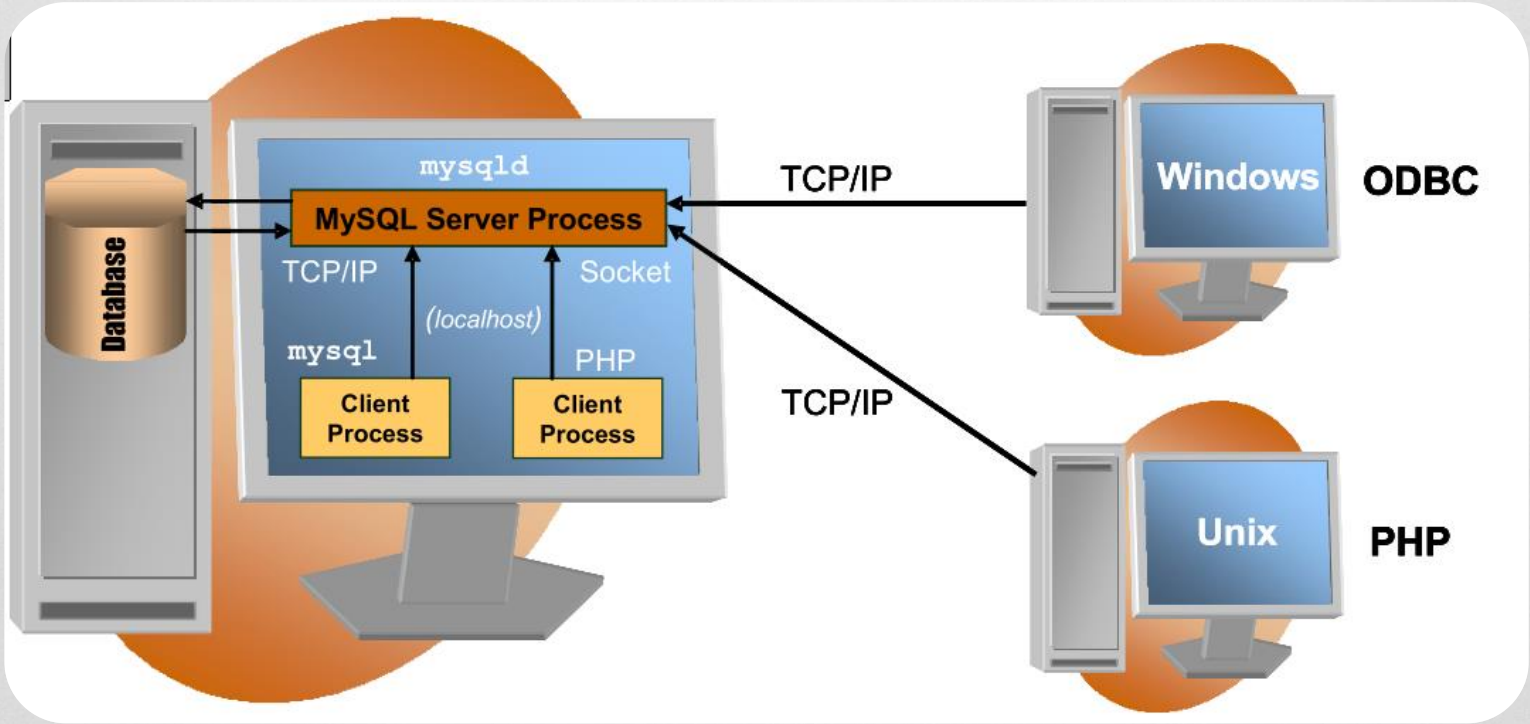
```
$ sudo apt-get install mysql-server
```

- For CentOS/Red Hat Distros:

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

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.



CLIENT/SERVER ARCTICTURE

- `mysql` is an interactive client that lets you issue queries and see the results.
- Some administrative clients are:
  - `mysqldump`, a backup program that dumps table contents into a file.
  - `mysqlimport` for importing data files
  - `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.
  - `mysqlcheck` for checking the integrity of the database files.

INVOKING CLIENT PROGRAMS

- To determine the options supported by a MySQL program:

```
$ mysql --help
```

- To determine the version of a program

```
$ mysql --version
```

```
mysql  Ver 14.14 Distrib 5.1.30, for Win32
(ia32)
```

- To Log into MySQL

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

- you should get a response something like this:

```
Enter password:
```

INVOKING CLIENT PROGRAMS

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

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

Your MySQL connection id is 1 to server
version: 5.1.25-rc-community MySQL Community
Server (GPL)

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

INVOKING CLIENT PROGRAMS

- Command line options:

Option	Meaning
`--protocol`	The protocol to use for the connection (tcp, socket, pipe, memory)
`--host | -h`	The host where the server is running
`--port | -P`	The port number for TCP/IP connections, default 3306
`--socket | -S`	The Unix socket filename or Windows named-pipe name
`--compress | -C`	This option causes all communication between the client and the server to be compressed. The result is a reduction in the number of bytes sent over the connection, which can be helpful on slow networks.
`--username | -u`	The name of the user connecting to MySQL
`--password | -p`	The password for the user connecting to MySQL

INVOKING CLIENT PROGRAMS

- You can exit `mysql` client using

```
mysql> exit
```

```
mysql>quit
```

```
mysql>\q
```

- The default statement terminators are: `\g` and **;**
- To change default statement terminators , use:

```
mysql>DELIMITER go
```

```
mysql>\d go
```

- The `\G` sequence is also recognized as statement terminator, It displays the result in a vertical style, showing each column value on a separate line, separating each row with a line of asterisks

INVOKING CLIENT PROGRAMS

- You can exit `mysql` client using

```
mysql> exit
```

```
mysql>quit
```

```
mysql>\q
```

- The default statement terminators are: `\g` and `;`
- To change default statement terminators , use:

```
mysql>DELIMITER go
```

```
mysql>\d go
```

- 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 (*/)

- To change the default prompt, use:

```
mysql> prompt (\u@\h) [\d]\>
```

- To revert to the default prompt, use: `\R` or `prompt`

CREATING USERS

- Create One or more users:

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

[,account [IDENTIFIED BY[PASSWORD]
'password'] ] ...


CREATE USER ‘islam'@'localhost' IDENTIFIED BY
‘os123';
```

MARINATING PRIVILEGES

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

MARINATING PRIVILEGES

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

```
GRANT privileges [columns]

ON item

TO user_name [IDENTIFIED BY 'password']

[REQUIRE ssl_options]

[WITH [GRANT OPTION | limit_options]  ]
```

MARINATING PRIVILEGES

- Three basic types of privileges exist in MySQL: privileges suitable for granting to regular users, privileges suitable for administrators, and a couple of special privileges.

MARINATING 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.

MARINATING PRIVILEGES

Privilege	Description
CREATE TEMPORARY TABLES	Allows an administrator to use the keyword TEMPORARY in a CREATE TABLE statement.
FILE	Allows data to be read into tables from files and vice versa.
LOCK TABLES	Allows the explicit use of a LOCK TABLES statement.
PROCESS	Allows an administrator to view server processes belonging to all users.
RELOAD	Allows an administrator to reload grant tables and flush privileges, hosts, logs, and tables.
REPLICATION CLIENT	Allows use of SHOW STATUS on replication masters and slaves. Replication is explained in Chapter 12.
REPLICATION SLAVE	Allows replication slave servers to connect to the master server. Replication is explained in Chapter 12.
SHOW DATABASES	Allows a list of all databases to be seen with a SHOW DATABASES statement. Without this privilege, users see only databases on which they have other privileges.
SHUTDOWN	Allows an administrator to shut down the MySQL server.
SUPER	Allows an administrator to kill threads belonging to any user.

MARINATING PRIVILEGES

Privilege	Description
`ALL`	Grants all the privileges listed in Tables 9.1 and 9.2. You can also write `ALL PRIVILEGES` instead of `ALL`.
`USAGE`	Grants no privileges. This privilege creates a user and allows her to log on, but it doesn't allow her to do anything. Usually, you will add more privileges later.

LIMIT OPTIONS

- With `GRANT OPTION` allows the user to give his/her own privileges to another user
- With LIMIT OPTIONS

```
MAX_QUERIES_PER_HOUR n

MAX_UPDATES_PER_HOUR n

MAX_CONNECTIONS_PER_HOUR n
```

MARINATING PRIVILEGES

- 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
```

- If you have given the WITH GRANT OPTION clause, you can revoke this (along with all other privileges) by adding:

```
REVOKE All , GRANT OPTION

FROM 'user_name'
```

EXAMPLES

```
mysql>GRANT all
-> ON *.*
-> TO `islam`@`localhost` identified by
  'is123'
-> WITH GRANT OPTIONS;


mysql>REVOKE all, GRANT OPTION
-> FROM `islam`@`localhost`;
```

CREATING DATABASES

- General Form :

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

CREATE SCHEMA [IF NOT EXISTS] db_name
[CHARACTER SET charset] [COLLATE collation];
```

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)

CASE SENSITIVITY

- The rules that determine whether an identifier is case sensitive depend on what kind of identifier it is:
  - For database and table identifiers, case sensitivity depends on the operating system and filesystem of the server host, and on the setting of the lower_case_table_names system variable.
  - Column, index, and stored routine identifiers are not case sensitive.
  - Column aliases are not case sensitive.
  - The case-sensitivity of trigger identifiers is dependent upon the operating system (like it is the case for table identifiers). However, that is unaffected by any setting of the lower_case_table_names system variable

CHARACTER SET AND COLLATION

- A character set is a set of symbols and encodings.
- A collation is a set of rules for comparing characters in a character set.
- To find out which character sets and collations are available, use these statements:

```
SHOW CHARACTER SET;

SHOW COLLATION;
```

CHARACTER SET AND COLLATION

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

```
CREATE DATABASE db_name CHARACTER SET charset
COLLATE collation;

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

c CHAR(10) CHARACTER SET charset COLLATE
collation
```

CHARACTER SET AND COLLATION

- 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`
/*!40100 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 databse:

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

- 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 t\G
```

- 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 AS 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 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)
-> )
-> ENGINE = MyISAM
-> COMMENT = 'Lists Language Spoken'
-> CHARSET utf8 ;
```

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 Sometime in 2037 timestamps.	A timestamp, useful for transaction reporting. The display format depends on the value of *M* (see Table 9.8, which follows). The top of the range depends on the limit on Unix.
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;
```

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 [UNIQUE|FULLTEXT] INDEX index_name

ON table_name (index_column_name
  [ASC|DESC], ...])
```

You can not make this for primary key index!

CREATING INDEXES

Index type	Description
Primary key	Requires that each value or set of values be unique in the columns on which the primary key is defined. In addition, null values are not allowed. Also, a table can include only one primary key.
Foreign key	Enforces the relationship between the referencing columns in the child table where the foreign key is defined and the referenced columns in the parent table.
Regular	A basic index that permits duplicate values and null values in the columns on which the index is defined.
Unique	Requires that each value or set of values be unique in the columns on which the index is defined. Unlike primary key indexes, null values are allowed.
Full-text	Supports full-text searches of the values in the columns on which the index is defined. A full-text index permits duplicate values and null values in those columns. A full-text index can be defined only on MyISAM tables and only on CHAR, VARCHAR, and TEXT columns.

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),
FULLTEXT index_name (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;
```

- To drop a primary key on table:

```
DROP INDEX `PRIMARY` 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
```