Mail

- You can send mail using the ***mail*** function
- PHP Mailer
  - https://github.com/PHPMailer/PHPMailer
  - http://www.learn2crack.com/2014/03/sending-mail-phpmailer.html
- Test mail server tool
  - http://www.toolheap.com/test-mail-server-tool/

Relational Database Systems

- Table (Relation)
  - Fundamental Unit of Storage
  - Rows/Columns(fields)
- Field Types
  - String
  - Integer
  - Float
  - enum
  - Etc.
- Primary Keys
- Operations
  - Select
  - Project
  - Join
- Query Language Used
  - SQL (Structured Query Language)
- Goal of System Design → Avoid redundancy

MySQL

- Database system we will be using
- **MySQL Console**
  - Allow us to execute commands (e.g., queries, create tables, etc.)
  - We start the console by executing mysql.exe
  - Once you set a password you can access the console by mysql.exe –u root -p
- Starting MySQL Console in XAMPP (Windows)
  - We start the console by executing mysql.exe –u root
  - In xampp you can find mysql.exe under C:\xampp\mysql\bin
    - C:\xampp\mysql\bin\mysql.exe –u root
  - Make sure you have started MySQL Server (e.g., via XAMPP Control Panel Application in Windows)
- Starting MySQL Console in XAMPP (Mac)
  - Folder /Applications/XAMPP/xamppfiles/bin
  - **./mysql –u root -p**
  - Make sure you have started MySQL Server
- Beep Disabling (sound when a mistake in done in a mysql command)
  - In Windows
    - Access The Device Manager
    - View “Show Hidden Devices”
    - Under “Non-Plug and Play Drivers” double-click on Beep
    - Right click, Properties, Select the Driver tab and Stop

MySQL

- Commands (end with semicolon, non-case sensitive)
  - show databases; → let us know the databases in the system
  - **Creating databases**
    - Tables are part of databases, so we need to create a database
    - create database <NAME>;
    - **Example:** create database ourDB;
    - If everything went OK you will see something similar to this:
      Query OK, 1 row affected (0.02 sec)
  - **Changing to a database**
    - Use <DATABASE_NAME>;
    - **Example**: use ourDB;

SQL Commands

- **Creating a table**
  - create table <tableName> (fieldList);
  - fieldList is a comma separated list of fields of the form
    - <FIELDNAME> <TYPE> <ATTRIBUTES>
  - **Example:** create table movies(name varchar(20), year int);
- Showing tables in a database
  - show tables;
- Looking at table structure
  - describe <TABLENAME>;
  - **Example:** describe movies;
- Inserting data
  - insert into <TABLENAME> values (<COMMA_SEPARATED_VALUES>) ;
  - **Example:** insert into movies values (“Jaws Jr”, 1976);
  - Strings in double or single quotes
- Looking at table contents
  - select * from <TABLENAME>;
  - **Example:** select * from movies;

Field Types (Numeric)

- **Table Field Types:**
  - **int** → signed/unsigned integer. Aprox: (-2 billions → 2 billions) or (0 → 4) billions
  - **tinyint** → signed/unsigned integer: (-128 → 128) o (0 → 255)
  - **smallint** →signed/unsigned integer (-32768 → 32767) or (0 → 65535)
  - **mediumint** → signed/unsigned integer: Aprox (-8 millions → 8 millions) o (0 → 16) millions
  - **bigint** → signed/unsigned
  - **float(M,D)** → floating point
    - M → Display length (total number of digits including decimals)
    - D→ number of decimals. M and D are not required and default to 10 and 2
    - Decimal precision → 24 places
  - **double(M,D)** – floating point.
    - M → Display length (total number of digits including decimals)
    - D→ number of decimals. M and D are not required and default to 16 and 4
    - Decimal precision → 53 places
    - real is a synonym for double
- null is possible field value

Field Types (String)

- **Table Field Types:**
  - **char(length)** → fixed-length string between 1 and 255 characters
    - **Example:** state char(2)
  - **varchar(length)** → variable-length string between 1 and 255 characters
    - **Example:** name varchar(20)
  - **blob or text** → (Binary Large Objects) – Use to store binary data (e.g., images). Maximum size is 65535
  - **tinyblob or tynytext** → blob or text with maximum size of 255 characters
  - **mediumblob or mediumtext** → blob or text with maximum size of 16777215
  - **longblob or longtext** →blob or text with maximum size of 4294967295
  - **enum** → enumeration (maximum of 65535 values)
    - **Example:** enum('M','F')
- null is possible field value
- Strings can be specified with single or double quotes

SQL Commands

- Let's create another table
  - create table friends (name varchar(20), met date, salary float);
- Let's insert some values
  - insert into friends values ("Mary", "2007-01-30", 10000);
  - insert into friends (name) values ("Jose");
- Selecting data
  - select * from <TABLE> where <CONDITION>;
    - Example: select * from friends where salary > 5000;
  - Comparison operators for conditions
    - = → equals
    - != → not equals
    - <= → less than or equal to
    - < → less than
    - >= greater than or equal to
    - > greater than
  - Logical Operators: and, or
  - Field specification (projection): select <FIELDLIST> from <TABLE> where <CONDITION>
    - select name,met from friends where salary > 5000;
- **Deleting data from a table**
  - delete from <TABLENAME> where <CONDITION>;
  - DANGER – removes all the entries in the table
    - delete from <TABLENAME>;

SQL Commands

- **Removing a table**
  - drop table <TABLENAME>;
  - Example: drop table movies;
- **Removing a database**
  - drop database <DATABASENAME>;
  - Example: drop database ourDB;

SQL Commands

- **Field Attributes**
  - primary key
  - not null
  - auto_increment
- **Let's create our table again**

  create table friends (name varchar(20) primary key not null, salary float not null);
- **Autoincrement**

  create table items (id int auto_increment primary key, name varchar(20));

  insert into items (name) values ("house");

SQL Commands

- **Update**
  - update friends set salary=20000 where name="Mary";
  - Assuming there was a year field
    - update friends set salary=7778, year=8 where name = "Pat";

- **Replace**
  - If the record you are inserting has a primary key value that matches record in the table the table record will be deleted and new one inserted
  - replace into friends values ("Mary", 5000);

SQL Functions

- You can try the following functions by using **select** at the mysql console
- Functions
  - lcase → returns a lowercase string **Example:** select lcase("TOM");
  - ucase → returns an uppercase string **Example:** select ucase("cat");
  - now() → returns current date/time **Example:** select now();
    - Output: 2006-12-12 12:21:02
  - curdate() → returns current date **Example:** select curdate();
    - Output: 2006-12-12
  - curtime() → returns current time **Example:** select curtime();
    - Output: 12:21:24
  - password() → returns a hash for the string
    - **Example:** select password("hello");
    - Output: 70de51425df9d787

MySQL Commands

- like operator → to compare strings
  - % wildchar character → matches multiple characters
  - _ wildchar character → matches one character
  - **Example:** delete from friends where name like “%Jose%”;
- Order by → to display elements ordered by a field
  - **Example:** *select * from friends order by salary;*
  - Output: elements will be listed in increasing salary order
  - **Example:** select * from friends order by salary desc;
  - Output: elements will be listed in decreasing salary order
- count → allows you to determine number of records satisfying a criteria
  - **Example:** *select count(name) from friends where salary <= 12000;*
  - Output: number of friends satisfying salary restriction
- and, or, between operators

Aggregates Functions

- **having** → to deal with aggregate functions (where clause cannot be used against aggregates). For example, imaging you have a table where you register the name, and amount spent by a person. A person can have multiple entries in the table. The following query will provide a list of those that spent more than 40 dollars:

  select name, sum(amount)
  from persons
  group by name
  having sum(amount) > 40;

- **limit** → controls the number of records returned.

  Example: *select * from allfriends limit 3;* → first three records are displayed

  Example: *select * from allfriends limit 3,2;* → skips the first three records and displays the following two

Joins

- Operation that allow us to combine information from several tables
- Very useful
- Example:
  - friends table with fields name, salary, gender
  - foods table with fields person, food
  - If you want to display the name, salary, and food someone likes you can execute the following query:

    select name,salary,food
    from friends,foods
    where friends.name = foods.person;

  - What happens if we remove the where clause?

Additional Commands

- **Altering table structure**
  - If you want to add a column to a table you can use the alter command

    alter table <TABLENAME> add <FIELDNAME> <FIELDTYPE>
  - **Example:**
    - alter table applicants add password varchar(80);
- **Accessing a remote database**
  - mysql -h <HOST> -u <USER> -p
  - Password must be provided after
- **Granting access via grant command**
  - grant <PRIVILEGE_LIST> on <DATABASE>.* to <USER>@"%" identified by "<PASSWORD>";
  - **Example:**

    grant all on myDB.friends to student@"%" identified by "goodbyeWorld";
  - <PRIVILEGE_LIST> → all, create, delete, drop, insert, update

MySQL (Information about Users)

- If you want to see information about mysql users execute either of the following alternatives:
- **First Alternative**

  *select * from mysql.user;* → difficult to see information

  select host,user from mysql.user;
- **Second Alternative**

  use information_schema;

  *select * from user_privileges;*

MySQL Root Password

- To change any users password (including root) execute the following through the MySQL console (mysql>)

  use mysql;

  update user set password=PASSWORD("NEWPASSWORD") where user='USERNAME';

  flush privileges;

- About root accounts
  - There is a root account associated with localhost and one for external accesses
  - Each can have a different password
  - The above update command handles both passwords
- To reset a root password to no password use
  - *update user set password=''* where user = 'root';
  - Notice '' is an empty string
- You can also change passwords through mysqladmin

MySQL Table Manipulation

- Use single quotes for strings if you want to be ANSI compatible
- Setting a default value for a column after table creation

  alter table <TABLENAME> alter <FIELDNAME> set default <DEFAULTVALUE>;
- Renaming a table

  alter table <TABLENAME> rename as <NEWTABLENAME>;
- Defining a field as a primary key for an existing table

  alter table <TABLENAME> add primary key (<FIELDNAME>);
- Creating table with default values

  Example: *create table toys (id int, name varchar(50) default "No Description");*
- creating table from another table

  create table <NEWTABLENAME> select query;

  Example: *create table nameSalary select name,salary from allfriends;*
- Avoiding error message while creating table that exists. No modification to the table will occur.

  create table if not exists <TABLENAME> …

Database Transactions

- Transaction → group of SQL statements that must be executed as a batch
- Transaction semantics
  - start transaction
  - commit
  - rollback
- MySQL statements usually make use of implicit commit
  - Statements are executed and written directly to the database
- You can disable implicit commit in MySQL by:

  set autocommit = 0;

MySQL Engines

- **MySQL Internal engines**
  - Manage and manipulate data (used to process selects, create tables, etc.)
  - Several engines each capable of performing select, create tables, etc.
- **myisam** engine →default engine (you do not need to specify it). It supports full-text searching but it does not support transactional processing
- **innodb** engine →It provides transactional support. It provides no support for full-text searching
- **memory** engine → equivalent to **myisam** but data is stored in memory (extremely fast)

MySQL Engines

- Creating a table that uses the **innodb** Engine
- Creating table using innodb engine

```
create table tvshows (
     name varchar(50),
    description varchar(80)
) engine=innodb;
```

- You could replace innodb with myisam;
- Transaction example

```
set autocommit = 0
start transaction
// INSERT records
rollback

start transaction
// INSERT records
commit
```

SQLite/PDO

- **SQLite**
  - Lightweight, file-based database
  - Part of PHP 5
  - Database access without running a separate RDBMS process
- **PDO** → PHP Data Objects
  - PDO Database Access Layer → Allows you to use the same PHP functions regardless the database engine you are using

phpMyAdmin

- Interface to database system
- Just type
  - http://localhost/phpmyadmin/

SQL Commands Review

- show databases;
- create database myDB;
- use myDB;
- show tables;
- create table friends (name varchar(20) primary key, gender enum('M','F'), salary float, id int);
- describe friends;
- insert into friends values ("Mary", "F", 10000, 10);
- insert into friends (name) values ("Jose");
- select * from friends where salary > 5000;
- select name,id from friends where salary > 5000;
- update friends set salary=7778, gender="F" where name = "Pat";
- delete from friends where name="Pat";
- show grants;
- drop table friends;
- drop database myDB;