Module 3 – MySQL Database

- Module 3 Contains 2 components
  - Individual Assignment
  - Group Assignment
- BOTH are due on Mon, July 9th
- Read the WIKI before attempting the lab

Database Management System

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

<table>
<thead>
<tr>
<th>Student_ID</th>
<th>Name</th>
<th>Major</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Shannon</td>
<td>CSE</td>
<td>A</td>
</tr>
<tr>
<td>102</td>
<td>Mike</td>
<td>CHEM</td>
<td>A</td>
</tr>
<tr>
<td>103</td>
<td>Wang</td>
<td>BIO</td>
<td>A</td>
</tr>
</tbody>
</table>

- Database Management System (DBMS) is software to maintain and utilize the collections of data (Oracle, DB2, MySQL)
Cells, Rows, Tables and Databases

- Cell -- a single (scalar) value

Letter: July 23, 1842

- Row -- a group of scalar values representing a single instance of an object or event
Cells, Rows, Tables and Databases

- Table -- a series of rows describing separate objects or events

<table>
<thead>
<tr>
<th>ID</th>
<th>METSID</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>12134</td>
<td>1090313313</td>
<td>Letter: November 18, 1838</td>
</tr>
<tr>
<td>12135</td>
<td>1310391314</td>
<td>Letter: July 23, 1842</td>
</tr>
<tr>
<td>12136</td>
<td>1313020414</td>
<td>Waterloo at Sunset</td>
</tr>
</tbody>
</table>

Cells, Rows, Tables and Databases

- Database -- a collection of related tables describing various facets of a group of objects or events

<table>
<thead>
<tr>
<th>OBJECTS</th>
<th>COLS</th>
<th>CLINKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>METSID</td>
<td>METSID</td>
<td></td>
</tr>
<tr>
<td>LABEL</td>
<td>COLID</td>
<td></td>
</tr>
<tr>
<td>NAME</td>
<td>URL</td>
<td></td>
</tr>
<tr>
<td>ABSTRACT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Relationships in Databases

- Databases are great for storing different types of data and managing relationships between them.

- When designing a DB it is important to understand the what types of relationships you need to create:
  - These relationships are defined through referential integrity (keys and/or constraints).

- There are a few common types of DB relationships we will look at:
  - One-to-One
  - One-to-Many
  - Many-to-Many

Relations -- One to One

- Example: Table containing Social Security Number has a one-to-one relationship to table of Wash U Student IDs.

<table>
<thead>
<tr>
<th>Table1</th>
<th>Table2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record</td>
<td>Record</td>
</tr>
<tr>
<td>Record</td>
<td>Record</td>
</tr>
<tr>
<td>Record</td>
<td>Record</td>
</tr>
<tr>
<td>Record</td>
<td>Record</td>
</tr>
<tr>
<td>Record</td>
<td>Record</td>
</tr>
<tr>
<td>Record</td>
<td>Record</td>
</tr>
<tr>
<td>Record</td>
<td>Record</td>
</tr>
</tbody>
</table>

Example: Table containing Social Security Number has a one-to-one relationship to table of Wash U Student IDs.
One to Many

- Example: Table of Academic Advisors has a one to many relationship with a table containing students at WashU

Many to Many

- Example: Table of courses taught in the Fall semester has a many to many relationship with a table containing students at WashU
Which DB to use? Why MySQL?

- Free SQL (Structured Query Language) database server
  - licensed with the GNU General public license
    http://www.gnu.org/

- MySQL is a relational database management system (RDBMS)

- MySQL is Open Source Software
- Officially pronounced “my Ess Que Ell”

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)

Login

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

**Example**

```bash
% mysql -u username -p
Enter password: password
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> create user 'test1'@'localhost' identified by 'mysecretpass';`
  - Create a new database user test1

- `mysql> grant all on *.* to test1@'localhost' with grant option;`
  - Gives administrative privileges to user test1
    - It is common to restrict users to a particular database with limited access, which we are NOT doing here

Create Database

What are the current databases at the server?
`mysql> show databases;`
```
+------------+
| Database   |
+------------+
| mysql      |
| test       |
+------------+
```
- `mysql` is a database (stores users' password ...) used by system.
- `test` is a database

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**

```sql
mysql> CREATE TABLE student
-> {
->   student_ID INT UNSIGNED NOT NULL,
->   name VARCHAR(20) NOT NULL,
->   major VARCHAR(50),
->   email VARCHAR(50)
-> }
Query OK, 0 rows affected (0.00 sec)
```

<table>
<thead>
<tr>
<th>Student_ID</th>
<th>Name</th>
<th>Major</th>
<th>Email</th>
</tr>
</thead>
</table>

Domain Types in SQL

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(n)</td>
<td>Fixed length character string, with specified length n</td>
</tr>
<tr>
<td>VARCHAR(n)</td>
<td>Variable length character string, with specified maximum length n</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Integer (a machine-dependent finite subset of the integers)</td>
</tr>
<tr>
<td>SMALLINT(n)</td>
<td>A small integer (a finite subset of INTEGER)</td>
</tr>
<tr>
<td>FLOAT(M,D)</td>
<td>Floating point number, with total number of digits M and number of digits following the decimal point D</td>
</tr>
<tr>
<td>DOUBLE(M,D)</td>
<td>Double-precision floating point number</td>
</tr>
</tbody>
</table>

- Similar to data types in classical programming languages
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      |
+----------+---------+-------+------+-----+------------+
| student_ID | int(10)  |       |      |        |            |
| name     | varchar(20) |       |      |        |            |
| major    | varchar(50) | YES   | NULL |       |            |
| email    | varchar(50) | YES   | NULL |       |            |
+----------+---------+-------+------+-----+------------+
4 rows in set (0.00 sec)

Demo
Primary and Foreign Keys

- One mechanism to enforce “referential integrity” is through primary and foreign keys.

- Primary Keys are used in MySQL as unique identifiers for each row in a table.
  - Consider a database of students.
    - The student ID could serve as a primary key.

- A foreign key is a field in a table which is also the primary key of another table.
  - Known as referential integrity.
  - Consider a WashU Database with tables for students, courses, and enrollment.
    - What if the student tries to enroll in a course that does not exist?

Modify Table Structure

- **ALTER TABLE** table_name Operations

```
mysql> alter table student add primary key (student_ID);
Query OK, 0 rows affected (0.00 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> describe student
+---------------+---------+-----------------+---------+---------+---------------------+---------+
| Field         | Type    | Null | Key  | Default | Extra               | Extra   |
+---------------+---------+------|------|--------+---------------------+---------|
| student_ID    | int(10) |      | PRI  | 0       |                     |         |
| name          | varchar(20) |    |     |         |                     |         |
| major         | varchar(10) | YES | NULL |         |                     |         |
| email         | varchar(50) | YES | NULL |         |                     |         |
+---------------+---------+------|------|--------+---------------------+---------+
4 rows in set (0.00 sec)
```
Insert Record

• INSERT INTO table_name SET col_name1=value1, col_name2=value2, col_name3=value3, ...

• Example

mysql> INSERT INTO student SET student_ID=101, name='Shannon', major='CSE', email='shannon@yahoo.com';
Query OK, 1 row affected (0.00 sec)

<table>
<thead>
<tr>
<th>Student_ID</th>
<th>Name</th>
<th>Major</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Shannon</td>
<td>CSE</td>
<td><a href="mailto:shannon@yahoo.com">shannon@yahoo.com</a></td>
</tr>
</tbody>
</table>

Retrieve Record

• SELECT what_columns FROM table or tables WHERE condition

• Example

mysql> SELECT major, email FROM student WHERE name='Shannon';
+---------------+-----------------------------+
| major | email                  |
| CSE   | shannon@yahoo.com       |
+---------------+-----------------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM student

<table>
<thead>
<tr>
<th>Student_ID</th>
<th>Name</th>
<th>Major</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Shannon</td>
<td>CSE</td>
<td><a href="mailto:shannon@yahoo.com">shannon@yahoo.com</a></td>
</tr>
<tr>
<td>102</td>
<td>Mike</td>
<td>CHEM</td>
<td><a href="mailto:mike@gmail.com">mike@gmail.com</a></td>
</tr>
<tr>
<td>103</td>
<td>Wang</td>
<td>BIO</td>
<td><a href="mailto:wang@wustl.edu">wang@wustl.edu</a></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Update Record

- **UPDATE** `table_name`
  
  SET **which columns to change**

  WHERE **condition**

- **Example**

  ```
  mysql> UPDATE student SET `email`='shannon@wustl.edu' WHERE `name`='Shannon';
  Query OK, 1 row affected (0.00 sec)
  Rows matched: 1  Changed: 1  Warnings: 0
  ```

  ```
  mysql> SELECT * FROM student WHERE `name`='Shannon';
  +--------------------------+
<table>
<thead>
<tr>
<th>name</th>
<th>student_ID</th>
<th>major</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shannon</td>
<td>101</td>
<td>CSE</td>
<td><a href="mailto:shannon@wustl.edu">shannon@wustl.edu</a></td>
</tr>
</tbody>
</table>
  +--------------------------+
  1 row in set (0.00 sec)
  ```

Delete Record

- **DELETE FROM** `table_name` **WHERE** **condition**

- **Example**

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

  ```
  Mysql> DELETE FROM student;
  Will delete ALL student records!
  ```
More complex queries

- Queries are often interested in data from multiple tables
- INNER JOIN (or just JOIN)
- LEFT JOIN
- RIGHT JOIN
- The wiki has some great examples of using the JOIN operator
PHP and MYSQL

- We will use PHP to communicate with our mySQL database

- MySQL Improved Prepared Statements provide a clean way to issue queries

- Refer to the wiki for additional syntax examples

MySQL and PHP Demo
Create database with user/password

- Never store passwords in a database
- Instead store password hash
  - More importantly store a salted hash.

User accounts and passwords

- The group portion of this module requires user accounts and passwords
  - Never store passwords as plain text in a database
  - Instead use a salted password hash
    - A password hash is a function that takes a password and maps it to a fixed size bit string
    - Hashed passwords are also fixed length, so perfect for a char (not varchar) data type
Hashing

- The MD5 Message Digest Algorithm is a widely used hash function for security applications

- I could store my password as an MD5 hash in the mySQL...
  - But these passwords are trivially reversed
    - So we add additional information to the string (called salt) to make it harder to accomplish
    - Hash(Password + Random characters)

- So can PHP help me with this?
  - Fortunately PHP has a method called `password_hash` which takes most of the work
    - Older versions of PHP used a less insecure function called `crypt`
  - See the wiki for more information