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");
- insert into friends values ("Pat", "M", 9000, 12);
- 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";
- Grant command example:
  - *grant all on myDB.friends to student@localhost identified by "goodbyeWorld";*
- Grant privilege list includes all, create, delete, drop, insert, update, select
- show grants;
- show grants for 'student'@'localhost';
- drop table friends;
- drop database myDB;

MySQL Views

- **View** → Virtual table generated from the result of a select statement
  - Contains row and columns
  - Fields in a view are from one or more tables
  - The database does not store the view data
- **Example:**
  - Assume you have a table with friends' information. The table has four fields (name, gender, salary, id)
  - We can create a view (name and salary) as follows:

    create view namesalary as

    select name, salary from friends;
  - You can list the view contents using a select

    select * from namesalary;
- You can drop the view using drop view

  drop view namesalary;

MySQL Indices

- Why indices?
- To see indices for a table use:

  show index from <TABLENAME>;

  show index from friends;
- To add index

  alter table <TABLENAME> add index (<FIELDNAME>);

  alter table friends add index (name);
- To drop an index

  alter table <TABLENAME> drop index <FIELDNAME>;

  alter table friends drop index name;

MySQL show

- To see user permissions

  show grants for <USERNAME>;
- To see table info (equivalent to describe <TABLENAME>)

  show fields from <TABLENAME>;
- To see system variables

  show variables;

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

```
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 Extension
  - Defines a consistent interface for accessing databases in PHP
  - Regardless of which database you're using, you use the same functions to issue queries and retrieve data
  - http://php.net/manual/en/intro.pdo.php

Internet Media Types

- Indicates the type of data associated with a file
- Examples:
  - text/html
  - application/javascript
  - application/pdf
  - text/css
- Media type is composed of type, subtype and optional parameters
  - Example: *text/html; charset=UTF-8*
    - Type → type
    - Subtype → html
    - Optional parameter → charset=UTF-8
- Initially called MIME (Multipurpose Internet Mail Extensions) types
- Sometimes referred to as **Content-types**
- List of types
  - http://www.iana.org/assignments/media-types/media-types.xhtml
- Reference
  - https://en.wikipedia.org/wiki/Internet_media_type

Database Access Through PHP

- Accessing databases from PHP by using the mysqli_* family of functions
- Do not confuse them with mysql
- mysqli documentation can be found at

  http://us3.php.net/mysqli
- **Functions for non-procedural approach**

  - mysqli_connect → to connect to the database (returns a database handle)
  - mysqli_correct_errno → to check for errors
  - mysqli_close → to close a connection
  - mysqli_query → to submit a query
  - mysqli_fetch_array → to generate array with record data

Accessing the Database (Functional)

- **For the following examples we are assuming:**
  - We have created the database ***myDB*** and table ***friend****s as* we saw in the slide "SQL Commands Review"
  - We have inserted one data record

    insert into friends values ("Mary", "F", 10000, 10);
  - A user *student* with password ***goodbyeWorld*** has been granted the following privileges (sql command):

    grant all on myDB.friends to student@localhost identified by "goodbyeWorld";
- **Example:** FunctionalAccess/readingDB.php
  - Let's run the example with empty table
  - Let's run the example after stopping the database system
  - Let's run the example providing a wrong user password
- **Example:** FunctionalAccess/insertDB.php
- **Example:** FunctionalAccess/updateDB.php (relies on result of insertDB.php)

Retrieving/Inserting Images in MySQL

- Before running following example make you drop the table “docs” (in case it exists)
  - drop table docs;
  - grant all on myDB.docs to student@localhost identified by “goodbyeWorld”;
- **Example:** (FunctionalAccess/ImagesEx1 Folder)
  - creatingDocumentsTable.php, insertingDocument.php, retrievingDocument.php
- **Example:** (FunctionalAccess/ImagesEx2 Folder)
  - Retrieves an image and displays the image within an HTML document
  - retrievingDocumentForm.html

Accessing the Database (Object-Oriented)

- **For the following examples we are assuming:**
  - We have created the database ***myDB*** and table ***friend****s as* we saw in the slide "SQL Commands Review"
  - A user *student* with password ***goodbyeWorld*** has been granted the following privileges (sql command):

    grant all on myDB.friends to student@localhost identified by "goodbyeWorld";
- **Example:** ObjectOrientedAccess/insertObjectOriented.php
- **Example:** ObjectOrientedAccess/retrieveObjectOriented.php

MySQL Injection

- MySQL Injection
  - When an SQL query is inserted (without your knowledge) to be run on your database
- Make sure you create a correct query string
- Look at the implications of using %s vs. %d in the sql query of the following example
  - Using same friends table defined in slide "SQL Commands Review"
- Using %s (instead of %d) causes the injection
- Make sure you have at least two records in the table (friends)
- First, run the example with one of the ids in the table (expected result will appear)
- Now try with *999 or true* as id
- Repeat above experiment with %d
- You should clean up your data (e.g., turn into an integer the value provided via post)
- Comic: http://xkcd.com/327/
- **Example:** SQLInjection.php

mysqli_real_escape_string

- To avoid MySQL injections you should have magic quotes on (magic_quotes_gpc = on) in php.ini
- If you do not have magic_quotes on then use *mysqli_real_escape_string* to escape characters in input provided through GET and POST
- To show injection
  - Make sure you have at least two records in the table (friends)
  - Using same friends table defined in slide "SQL Commands Review"
  - Update php.ini with magic quotes disabled
  - Remove mysql_real_escape_string in the SQLInjectionTwo.php
  - Try this input Lynn' or 'a' = 'a
- **Example:** SQLInjectionTwo.php

Preventing both XSS and SQL Injections Attacks

- XSS (Cross-site scripting) Injection
  - Takes place when HTML/JavaScript to be provided by the user is displayed back by the web site
  - Can be prevented by using the **htmlentities** function
- We can use real_escape_string function provided via the database connection to present SQL injection
- **Example: xssSqlPrevention.php**