INFO2180 - LECTURE 7

SQL

SQL STANDS FOR STRUCTURED QUERY LANGUAGE. IT LETS YOU ACCESS AND MANIPULATE DATABASES.

http://www.w3schools.com/sql/sql_intro.asp

OVERVIEW OF SQL

- is an ANSI (American National Standards Institute) standard.
- While there is a standard set of SQL commands, some database management systems have their own extensions to the standard.
- You can use SQL to retrieve, insert, update and delete data from a database.
- You can also create new databases and tables for those databases.
- You can create views and stored procedures.
- And you can grant permissions to users for tables, views and procedures.

RELATIONAL DATABASES

OVERVIEW OF RELATIONAL DATABASES

- RDBMS stands for Relational Database Management System.
- examples of RDBMS's are MySQL, Microsoft SQL Server, IBM DB2, Oracle, PostgreSQL, SQLite
- Data is stored in a tables.
- A table is collection of related data entries which consist of columns (fields or attributes) and rows (records).

OVERVIEW OF RELATIONAL DATABASES

- Databases have some mechanisms to help with recovery from failures (e.g. transactions)
- They allow for multiple concurrent users to have access to view/edit at the same time.
- Can be made to scale up to large data sizes.

Database

Column/Field/Attribute

Table

Row/Record

id	Name	Email	Gender
1	John Brown	jbrown@example.com	Male
2	Mary Jane	mjane@example.com	Female
3	Peter Parker	spiderman@example.com	Male
4	Tammy Chin	tchin@example.com	Female

SQL QUERIES

SELECT STATEMENTS

- Used to select data from a database table.
- The result is called a result-set.

```
SELECT column_name, column_name FROM table_name;

SELECT * FROM table_name;
```

DISTINCT MODIFIER

- statement is used to return only distinct (different) values.

```
SELECT DISTINCT column_name, column_name FROM
table_name;
```

WHERE CLAUSE

- used to filter (extract) records that match the criteria.
- Operators can be `=, <>, >, <, >=, <=, BETWEEN, LIKE, IN`

```sql
SELECT column_name, column_name FROM table_name
WHERE column_name operator value;
```

MULTIPLE WHERE CLAUSES

- multiple WHERE conditions can be combined using AND and OR

```
SELECT column_name, column_name FROM table_name
WHERE column_name = value AND column_name = value;
```

LIKE OPERATOR

- used in a **`WHERE`** clause to search for a specified *pattern* in a column

```sql
SELECT column_name(s) FROM table_name
WHERE column_name LIKE pattern;
```

LIKE OPERATOR

- The pattern usually takes one of these forms:
  - **`LIKE 'text%'`** searches for text that starts with a given prefix
  - **`LIKE '%text'`** searches for text that ends with a given suffix
  - **`LIKE '%text%'`** searches for text that contains a given substring

LIKE OPERATOR

```sql
SELECT column_name(s) FROM table_name
WHERE column_name LIKE '%mytext%';
```

ORDER BY KEYWORD

- used to sort the result-set by one or more columns

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```

INSERTING ROWS WITH INSERT STATEMENTS

- used to insert new records in a table.

```
INSERT INTO table_name VALUES
(value1,value2,value3,...);
```

```
INSERT INTO table_name (column1,column2,column3,…)
VALUES (value1,value2,value3,...);
```

UPDATING ROWS USING THE UPDATE STATEMENT

- used to update records in a table

```
UPDATE table_name
SET column1=value1, column2=value2,…
WHERE some_column=some_value;
```

DELETING ROWS WITH THE DELETE STATEMENT

- used to delete records in a table.

```
DELETE FROM table_name
WHERE some_column=some_value;
```

CREATING/DELETING DATABASES

```
CREATE DATABASE dbname;
```

```
DROP DATABASE dbname;
```

CREATING TABLES

```
CREATE TABLE table_name (
    column_name1 data_type(size),
    column_name2 data_type(size),
    column_name3 data_type(size),
   ....
);
```

EXAMPLE CREATING TABLES

```
CREATE TABLE users (
    id INT UNSIGNED NOT NULL PRIMARY KEY,
    name VARCHAR(32) DEFAULT NULL,
    email VARCHAR(32) DEFAULT NULL,
    password VARCHAR(16) DEFAULT NULL
);
```

SOME SQL DATA TYPES IN MYSQL

- Here are a few are other data types:
  - `DATE, DATETIME, TIME`
  - `BOOLEAN`
  - `TEXT`
  - `DECIMAL`
  - `BLOB`
- and there are others. (http://dev.mysql.com/doc/refman/5.7/en/data-types.html)

DELETING TABLES

```
DROP TABLE tablename;
```

SQL JOINS

SQL JOINS ARE USED TO COMBINE ROWS FROM TWO OR MORE TABLES.

http://www.w3schools.com/sql/sql_join.asp

TYPES OF JOINS

- There are different types of joins, here are a few:
  - INNER JOIN (same as JOIN)
  - LEFT JOIN
  - RIGHT JOIN
  - NATURAL JOIN
- Depending on the RDMS, there may be others.
- We will focus on the most basic which is the INNER JOIN.

EXAMPLE OF A JOIN

```
SELECT Orders.OrderID, Customers.CustomerName,
Orders.OrderDate
FROM Orders
JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

EXAMPLE OF A JOIN WITH TABLE NAME ALIASES

```
SELECT o.OrderID, c.CustomerName, o.OrderDate
FROM Orders o
JOIN Customers c
ON o.CustomerID=c.CustomerID;
```

DESIGNING A QUERY

- To determine what query you need, start by asking:
  - Which tables contain the data you want (**`FROM`**)
  - Which columns do I need for the result-set? (**`SELECT`**)
  - Do I need to connect more than one table? (**`JOIN`**)
  - How will I filter the values? (**`WHERE`**)
  - Do I need it to be sorted in a particular way? (**`ORDER BY`**)
- Then test the query in the MySQL console.

DATABASES AND PHP

DATABASES AND PHP

- PHP can be used to connect to, query and display the results from a database.
- PHP has extensions for a variety of databases.
- We will be using MySQL as our database management system.
- It is recommended that you use either the **`mysqli`** or **`pdo`** extensions for interacting with MySQL.
- We will be using the **`pdo`** extension.

PDO STANDS FOR PHP DATA OBJECTS. IT IS A LIGHTWEIGHT, CONSISTENT INTERFACE FOR ACCESSING DATABASES IN PHP.

http://php.net/manual/en/intro.pdo.php

PDO EXAMPLE

```php
// PDO + MySQL

$pdo = new
PDO('mysql:host=example.com;dbname=database',
'user', 'password');

$statement = $pdo->query("SELECT some_field FROM
some_table");

$row = $statement->fetch(PDO::FETCH_ASSOC);

echo htmlentities($row['some_field']);
```

PDO EXAMPLE

```php
$pdo->query("SELECT name FROM users WHERE id =
" . $_GET['id']); // <-- NO!
```

Why is this bad?

Remember you should ***NEVER*** trust user input.
Always sanitize and filter user input.

PDO EXAMPLE

```php
$stmt = $pdo->prepare('SELECT name FROM users
WHERE id = :id');

$id = filter_input(INPUT_GET, 'id',
FILTER_SANITIZE_NUMBER_INT); // <-- filter your
data first, especially important for INSERT,
UPDATE, etc.

$stmt->bindParam(':id', $id, PDO::PARAM_INT); //
<-- Automatically sanitized for SQL by PDO

$stmt->execute();
```

PREPARED STATEMENTS

- There are multiple benefits to using prepared statements, both for performance and security reasons.
- Prepared Statements will do some basic filtering of the variables you bind to them by default, which is great for protecting your application against SQL injection attacks.
- From a performance standpoint the benefits are more significant when the same query is being used multiple times in your application. You can assign different values to the same prepared statement, yet MySQL will only have to parse it once.

EXAMPLE PREPARED STATEMENT WITH AN INSERT STATEMENT

```php
<?php
$stmt = $pdo->prepare("INSERT INTO REGISTRY (name, value)
VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);

// insert one row
$name = 'one';
$value = 1;
$stmt->execute();

// insert another row with different values
$name = 'two';
$value = 2;
$stmt->execute();

?>
```

HTML TABLES

HTML TABLES WERE CREATED TO PROVIDE A STRAIGHTFORWARD WAY TO MARK UP STRUCTURED TABULAR DATA AND TO DISPLAY THAT DATA IN A FORM THAT IS EASY FOR USERS TO READ AND DIGEST.

Shay Howe – http://learn.shayhowe.com/html-css/organizing-data-with-tables/

OVERVIEW TABLES

- Back in the old days, tables were used not only for tabular data but also to layout pages.
- But tables were not meant for layouts and this too resulted in a number of associated problems.
- Luckily CSS came along and with it a better way to position and layout elements.
- Tables are for *tabular data*.

DON'T use tables for layouts!

EXAMPLE HTML TABLE

```
<table>
  <tr>
    <th>Title</th>
    <th>Stock</th>
    <th>Quantity</th>
    <th>Price</th>
  </tr>
  <tr>
    <td>Don&#8217;t Make Me Think by Steve Krug</td>
    <td>In Stock</td>
    <td>1</td>
    <td>$30.02</td>
  </tr>
</table>
```

EXAMPLE HTML TABLE CAPTION

```
<table>
  <caption>List of Books</caption>
  <tr>
    <th>Title</th>
    <th>Stock</th>
    <th>Quantity</th>
    <th>Price</th>
  </tr>
  …
</table>
```

EXAMPLE HTML TABLE WITH THEAD

```
<table>
  <thead>
    <tr>
      <th>Title</th>
      <th>Stock</th>
      <th>Quantity</th>
      <th>Price</th>
    </tr>
  </thead>
  …
</table>
```

EXAMPLE HTML TABLE WITH TBODY

```
<table>
  …
  <tbody>
    <tr>
      <td>Don&#8217;t Make Me Think by Steve Krug</td>
      <td>In Stock</td>
      <td>1</td>
      <td>$30.02</td>
    </tr>
    …
  </tbody>
</table>
```

TABLE SPECIFIC ATTRIBUTES

- **`colspan`** allows table cells to span multiple columns

  e.g. `<td colspan="2">My data</td>`

- **`rowspan`** allows table cells to span multiple rows

  e.g. `<td rowspan="3">My data</td>`

USING CSS WITH TABLES

- Standard CSS styles apply to tables also.
- There are some CSS properties specific to tables
  - **`border-collapse`**
  - **`border-spacing`**
  - **`caption-side`**
  - **`empty-cells`**
- You can also use **`text-align`** and **`vertical-align`**.

List of Students

ID	Name	Email
123	Bart	bart@fox.com
404	Ralph	ralph@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com

With **`border-collapse: collapse;`** the spacing between cells is removed. You can then use **`border-spacing: `*`length`*`;`** to set how much spacing.

List of Students

ID	Name	Email
123	Bart	bart@fox.com
404	Ralph	ralph@fox.com
456	Milhouse	milhouse@fox.com
888	Lisa	lisa@fox.com

With **`border-collapse: separate;`** there will be space between the cells.

Did I mention ***NOT*** to use tables for layouts?

I repeat...Tables should ***NOT*** be used for layout! They are to be used for displaying tabular data.

EXAMPLE OF PRINTING DATABASE DATA USING PHP IN A TABLE

```php
<?php
$stmt = $pdo->query("SELECT * FROM STUDENTS");
$students = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
```

EXAMPLE OF PRINTING DATABASE DATA USING PHP IN A TABLE

```
<table>
  <thead>
    <tr>
      <th>Student ID</th>
      <th>Name</th>
      <th>Email</th>
    </tr>
  </thead>
  <tbody>
    <?php foreach ($students as $student): ?>
    <tr>
      <td><?= $student['id']; ?></td>
      <td><?= $student['name']; ?></td>
      <td><?= $student['email']; ?></td>
    </tr>
    <?php endforeach; ?>
 </tbody>
</table>
```

RESOURCES

- W3Schools SQL - http://www.w3schools.com/sql/
- PDO - http://php.net/manual/en/book.pdo.php
- PHP The Right Way: Databases - http://www.phptherightway.com/#databases
- HTML Tables - http://learn.shayhowe.com/html-css/organizing-data-with-tables/