

PHP and MySQL

INFO/CS 2300:
Intermediate Web Design and
Programming

HW2

- Sometimes the English is harder than the SQL...even for native speakers
- In the real world you would often have less clear questions but probably more context

Sections

- Section activities help with the assignments

JOIN SA TECH

**We are recruiting
product designers!**

Designers would work on community impacting projects ranging from a campus safety app to a course review website.

Check out our projects at: github.com/Cornell-SA-Tech

If interested, please send a resume to
cornellsatech@gmail.com

SATech is an up-and-coming project team affiliated with Student Assembly looking for students passionate about making a change on our community through design and technology. Apply now by emailing CornellSATech@gmail.com!

Predictable Periods?



Female volunteers (18 – 38y) with regular menstrual cycles **who consider themselves overweight** are needed for research studies to understand associations between:

- Lifestyle habits
- Ovarian function

*Financial compensation will be provided.



Contact Professor Marla Lujan at
womensimaging@cornell.edu

JOINS Revisited

INNER JOIN

movie_id	title	year
1	Gladiator	2000
2	A Beautiful Mind	2001
3	Chicago	2002

name	movie_id
Russell Crowe	1
Russell Crowe	2
Tom Hanks	4

```
SELECT title, year, name
FROM movies
INNER JOIN stars_in
ON movies.movie_id = stars_in.movie_id
```

Order of tables
makes no
difference

Only matching
rows

title	year	name
A Beautiful Mind	2001	Russell Crowe
Gladiator	2000	Russell Crowe

LEFT JOIN

movie_id	title	year
1	Gladiator	2000
2	A Beautiful Mind	2001
3	Chicago	2002

name	movie_id
Russell Crowe	1
Russell Crowe	2
Tom Hanks	4

```
SELECT title, year, name
FROM movies
LEFT JOIN stars_in
ON movies.movie_id = stars_in.movie_id
```

Order of tables
is important

All rows from the
left table and
only matching
from the right

title	year	name
A Beautiful Mind	2001	Russell Crowe
Chicago	2002	(null)
Gladiator	2000	Russell Crowe

RIGHT JOIN

movie_id	title	year
1	Gladiator	2000
2	A Beautiful Mind	2001
3	Chicago	2002

name	movie_id
Russell Crowe	1
Russell Crowe	2
Tom Hanks	4

```
SELECT title, year, name
FROM movies
  RIGHT JOIN stars_in
    ON movies.movie_id = stars_in.movie_id
```

Reading left to
right, stars_in is
on the right

All rows from the
right table and
only matching
from the left

title	year	name
A Beautiful Mind	2001	Russell Crowe
Gladiator	2000	Russell Crowe
(null)	(null)	Tom Hanks

LEFT JOIN

movie_id	title	year
1	Gladiator	2000
2	A Beautiful Mind	2001
3	Chicago	2002

name	movie_id
Russell Crowe	1
Russell Crowe	2
Tom Hanks	4

```
SELECT title, year, name
FROM stars_in
LEFT JOIN movies
ON movies.movie_id = stars_in.movie_id
```

Reading left to
right, stars_in is
on the left

All rows from the
left table and
only matching
from the right

title	year	name
A Beautiful Mind	2001	Russell Crowe
Gladiator	2000	Russell Crowe
(null)	(null)	Tom Hanks

RIGHT JOIN

movie_id	title	year
1	Gladiator	2000
2	A Beautiful Mind	2001
3	Chicago	2002

name	movie_id
Russell Crowe	1
Russell Crowe	2
Tom Hanks	4

```
SELECT title, year, name
FROM stars_in
  RIGHT JOIN movies
    ON movies.movie_id = stars_in.movie_id
```

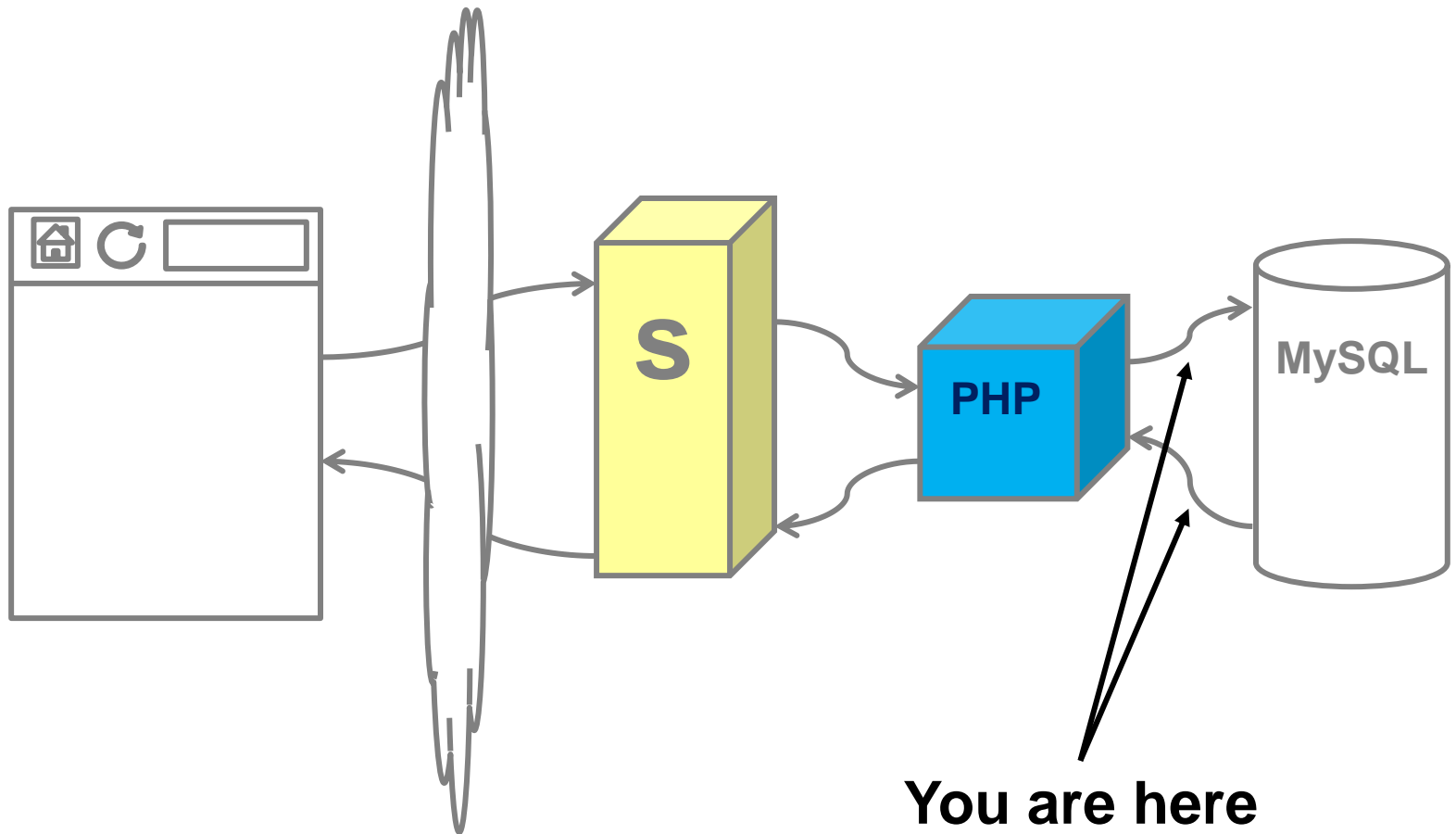
Reading left to
right, movies is
on the right

All rows from the
right table and
only matching
from the left

title	year	name
A Beautiful Mind	2001	Russell Crowe
Chicago	2002	(null)
Gladiator	2000	Russell Crowe

Where are we?

PHP – MySQL connection



Using MySQL from PHP



Ways to connect

`mysql_connect(...)`

**Deprecated.
Do not use.**

`mysqli_connect(...)`

**mysqli
procedural
style**

`new mysqli(...)`

**mysqli
object style**

`new PDO(...)`

**PDO
in 2 weeks**

Remember PHP objects?

I'll be using the object form of mysqli



From
lecture 4

- We can make *instances* of an object:
`$movie = new Movie();`
- *Fields* are data associated with an instance of an object:
`$movie->title = "some value"`
- *Methods* are functions associated with the object.
`print $movie->the_question();`

Connecting

To use MySQL from PHP you first need to **create an instance of a *mysqli* object**. We call the constructor with information on how to connect to the database.

```
$mysqli = new mysqli(hostname, username, password,  
database);
```

Returns a *mysqli* instance.

Click In!

Click In!

In the movies example, where does it make sense to put the host, username, password and db values?

- A. In the 'new mysqli' statement each time I use mysqli
- B. At the beginning of the 2 files where I use mysqli
- C. In a separate file

Click In!

In the movies example, where does it make sense to put the host, username, password and db values?

- A. In the 'new mysqli' statement each time I use mysqli
- B. At the beginning of the 2 files where I use mysqli
- C. In a separate file

config.php

```
<?php // ** MySQL connection settings ** //  
    // database host  
    define( 'DB_HOST', 'localhost' );  
  
    // database name - info230_SP17_username  
    define( 'DB_NAME', 'info230_SP17_sm68sp17' );  
  
    // Your MySQL / Course Server username  
    define('DB_USER', 'sm68sp17');  
  
    // ...and password  
    define('DB_PASSWORD', 'your_password');  
?>
```



**Your course server
credentials**

movies.php

```
require_once 'config.php';  
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
```

Issuing SQL commands

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
```

```
$result = $mysqli->query("SELECT * FROM Movies");
```

The mysqli object method `query(sqlquery)` issues *sqlquery* to the MySQL DB.

For INSERT, UPDATE, DELETE, **returns true if successful**, false if not

For SELECT, **returns instance of *result object* if successful**, false if not.

Getting results

```
$result = $mysqli->query("SELECT * FROM Movies");
```

Given the result object *result*, we can fetch the associated data using the result object.

```
$row = $result->fetch_row();
```

returns a numeric indexed array (e.g. title in \$row[0], year in \$row[1], etc.)

```
$row = $result->fetch_assoc();
```

returns an associative array (e.g. \$row['title'] has the value for title, etc.)

Both return false if no more rows left in the result.

}

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132


```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print( '<table><thead><tr><th>Title</th>...</thead><tbody>' )
while ( $row = $result->fetch_row() ) {

}
```

First time through loop:

```
$row = array('Chicago', 2002,
            113)
```

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

}

Second time through loop:

```
$row = array('The Return of the  
King', 2003, 201)
```

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print( '<table><thead><tr><th>Title</th>...</thead><tbody>' )
while ( $row = $result->fetch_row() ) {

}
```

Third time through loop:

```
$row = array('Million Dollar Baby',
2004, 132)
```

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```

$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print( '<table><thead><tr><th>Title</th>...</thead><tbody>' )
while ( $row = $result->fetch_row() ) {

}

```

Last time through loop:

`$row = false`

`print('</tbody></table>');`

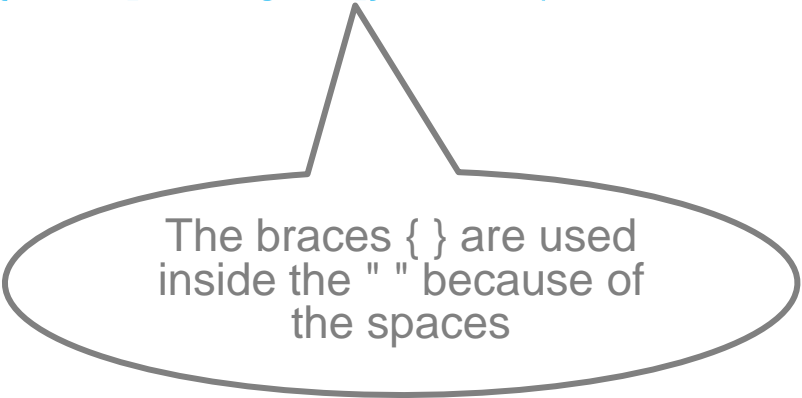
Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>...</thead><tbody>')
while ( $row = $result->fetch_row() ) {
    print( '<tr>' );
    foreach( $row as $value ) {
        print( "<td>$value</td>" );
    }
    print( '</tr>' );
}
```

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Using fetch_assoc()

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );  
$result = $mysqli->query("SELECT * FROM Movies");  
print('<table><thead><tr><th>Title</th>...</thead><tbody>')  
while ( $row = $result->fetch_assoc() ) {  
    print( '<tr>' );  
    print( "<td>{$row[ 'Title' ]}</td>" );  
    print( "<td>{$row[ 'Year' ]}</td>" );  
    print( "<td>{$row[ 'Length' ]}</td>" );  
    print( '</tr>' );  
}
```



The braces { } are used
inside the " " because of
the spaces

Alternatively

```
$mysqli = new mysqli( DB_HOST, DB_USER, DB_PASSWORD, DB_NAME );
$result = $mysqli->query("SELECT * FROM Movies");
print('<table><thead><tr><th>Title</th>...</thead><tbody>')
while ( $row = $result->fetch_assoc() ) {
    print( '<tr>' );
    $title = $row[ 'Title' ];
    $year = $row[ 'Year' ];
    $length = $row[ 'Length' ];

    print( "<td>$title</td>" );
    print( "<td>$year</td>" );
    print( "<td>$length</td>" );
    print( '</tr>' );
}
```

Number of rows returned

`$result->num_rows`

Contains the number of rows in the table given by *result*.

E.g.

```
$result = $mysqli->query("SELECT * FROM Movies");
```

```
$row_count = $result->num_rows;
```

Title	Year	Length
Chicago	2002	113
The Return of the King	2003	201
Million Dollar Baby	2004	132

Close the db

```
$mysqli->close();
```

Closes connection to DB

Not necessary most of the time since
PHP does it eventually on its own

Using other parts of SQL

Searching

Suppose we want to add search functionality. What should we do?

- HTML / PHP form
- SQL: LIKE, REGEXP



Click In!

Click In!

When building a page with a form that submits to the same page, which usually makes sense to do first in the code?

- A. Process (read, save, update) the form input data
- B. Display the form
- C. Doesn't matter

Click In!

When building a page with a form that submits to the same page, which usually makes sense to do first in the code?

- A. Process (read, save, update) the form input data
- B. Display the form
- C. Doesn't matter

Click In!

If I add a second criteria to my search,
which makes more sense in this
context?

- A. More results
- B. Fewer results

Searching - HTML

No action specified so
form self-submits to
the same file it is in

In the demo, these
inputs are wrapped in
table tags

```
<form action="" method="post">  
  <input type="text" name="title">  
  <input type="text" name="year">  
  <input type="text" name="length">  
  <input type="submit" name="search" value="Search" >  
</form>
```


settings.php

```
//Array of fields used
$fields = array(
    array(
        'term' => 'title',
        'heading' => 'Title',
        'filter' => FILTER_SANITIZE_STRING,
    ),
    array(
        'term' => 'year',
        'heading' => 'Year',
        'filter' => FILTER_SANITIZE_NUMBER_INT,
    ),
    array(
        'term' => 'length',
        'heading' => 'Length (min)',
        'filter' => FILTER_SANITIZE_NUMBER_INT,
    ),
);
```

Searching - PHP

//Build an array of search clauses

```
$searches = array();  
foreach( $fields as $field ) {  
    $search_term = $field[ 'term' ];  
    $filter = $field[ 'filter' ];
```

//Does this term exist in the POST data submitted by the search form?

```
if( !empty( $_POST[ $search_term ] ) ) {  
    //Get the value for this term from the POST data  
    $search_value = filter_input(INPUT_POST, $search_term, $filter);
```

//Add the search clause

```
$searches[] = "$search_term REGEXP '$search_value'";
```

```
}
```

```
}
```



title REGEXP 'ago'
year REGEXP '2002'

Building SQL + Searching

index.php

```
//Starting SQL
$sql = 'SELECT * FROM Movies';

//Were there search terms?
if( !empty( $searches ) ) {
    //Build the WHERE clause
    $sql .= ' WHERE ';

    //Add the searches by joining any elements together with AND
    $sql .= implode(' AND ', $searches );
}
```

// \$sql becomes "WHERE title REGEXP 'ago' AND year REGEXP '2002' "

Sorting

Suppose we want to allow the user to sort the entries by the various fields. How can we do that?

- HTML: links
- PHP: `$_GET`
- SQL: ORDER BY



index.php



Sort form

```
<thead>
  <th>
    <a href="?sort=title">Title</a>
  </th>
  <th>
    <a href="?sort=year">Year</a>
  </th>
  <th>
    <a href="?sort=length">Length</a>
  </th>
</thead>
```



`$_GET['sort']`

Sort PHP

```
//Try to get the 'sort' parameter from the URL
//and filter out bad stuff
//Better security would make sure it is one of our expected $fields
$sort = filter_input( INPUT_GET, 'sort', FILTER_SANITIZE_STRING );

//Is this sorted? $sort will be empty if the parameter was not set in the URL
if ( !empty( $sort ) ) {
    $sql .= " ORDER BY $sort";
}
```

Adding items

Suppose we want to be able to add items to our list. What should we do?

- HTML Form
- PHP processing and data checking
- SQL INSERT



[add-edit.php](#)



Adding - HTML

```
<input type="text" name="title">
```

```
<input type="text" name="year">
```

```
<input type="text" name="length">
```


Adding - PHP

Assume the POST data is processed into an array that looks like this

```
$field_values = array (  
    'title' => 'Into the Woods',  
    'year' => '2014',  
    'length' => '124'  
)
```

```
//Get an array of the field names that have data  
$field_name_array = array_keys( $field_values );
```

```
//Comma delimited list of fields  
//equivalent to $field_list = "title, year, length";  
$field_list = implode( ',', $field_name_array );
```

```
//comma delimited values - need quotes around values  
$value_list = implode( "','", $field_values );
```

```
//Build the SQL for adding a movie  
//later we'll improve security and quoting  
$sql = "INSERT INTO movies ( $field_list ) VALUES (  
    '$value_list' );";
```

Autonumber INSERT value

```
$sql = "INSERT INTO movies ( title,year,length )  
VALUES ( 'Into the Woods', '2014', '124' );";  
if( $mysqli->query( $sql ) ) {  
    $new_id = $mysqli->insert_id;  
}
```

Contains the value of the new autonumber id created by MySQL

Modifying

Suppose we want to let the user edit the various entries. How can we do that?

- HTML: link and additional form
- PHP: additional form processing
- SQL: UPDATE

Modifying - HTML

```
<input type="hidden" name="movie_id" value='17'>
```

```
<input type="text" name="title" value="Into the  
Woods">
```

```
<input type="text" name="year" value="2014">
```

```
<input type="text" name="length" value="124">
```

Modifying - PHP

Assume the POST data is processed into an array that looks like this

```
$field_values = array (  
    'movie_id' = 17,  
    'title' => 'Into the Woods',  
    'year' => '2014',  
    'length' => '124'  
)
```

Modifying – PHP & SQL

```
$update_fields = array();  
foreach( $field_values as $field_name => $field_value ) {  
    $update_fields[] = "$field_name = '$field_value'";  
}  
$sets = implode( ' , ', $update_fields );  
  
//Build the SQL for adding a movie  
//later we'll improve security and quoting  
$sql = "UPDATE movies SET $sets  
        WHERE movie_id=$movie_id";
```

Debugging

Getting MySQL Errors

Various fields in the mysqli object *mysqli* contain error information: *mysqli->errno* contains an error code (or 0 if no error), and *mysqli->error* contains a string with the error.

E.g.

```
if ( $mysqli->errno ) {  
    print($mysqli->error);  
    exit();  
}
```

Project 3: Image Album

Now you can practice your skills by writing your own image album website!

- Part 1: Due 3/14
 - DB Schema, Set up tables in your 2300 server DB, draft basic navigation of pages, initial code to display images
- Part 2: Due 3/21
 - Add code to display/add albums, add images to albums
- Part 3: Due 3/28
 - Final working site, with a secure login for image uploading.

Details on Piazza

Review

- We can now use the MySQL DBMS from within PHP by using the mysqli object and its methods (e.g. query, fetch_row, fetch_assoc).
- We can use all our favorite SQL queries (SELECT, INSERT, UPDATE, etc.) to generate results for our page.