WEBD 236

Web Information Systems Programming

Week 2

Todd Whittaker and Scott Sharkey

FRANKLIN UNIVERSITY

Agenda

- This week's expected outcomes
- This week's topics
- This week's homework
- Upcoming deadlines
- Questions and answers

Week 2 Outcomes

- Use DDL to create tables and indices in a DBMS.
- Use SQL to extract rows that match given criteria.
- Create server-based scripts to interactively query a data source and display the resulting rows in an HTML page.

Week 2 Topics

- Databases
  - Modeling
  - Normalization
  - Querying
- PHP and databases
  - Querying databases
  - Displaying results
- Logging

Database Review

- Database
  - Tables
    - Rows (records)
    - Columns (fields, attributes)
  - Primary key
    - Column(s) that uniquely identify a row
    - Usually a single column *surrogate key* (i.e. an “id” field) that has no real world significance

Database Review

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_DEGREE
105	ACCT	KLR 229D	8665	Ph.D
114	ACCT	KLR 211	4436	Ph.D
301	ACCT	KLR 244	4683	Ph.D
435	ART	BBG 185	2278	Ph.D
387	BIOL	AAK 230	8665	Ph.D

Database Review

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_DEGREE
105	ACCT	KLR 229D	8665	Ph.D
114	ACCT	KLR 211	4436	Ph.D
301	ACCT	KLR 244	4683	Ph.D
435	ART	BBG 185	2278	Ph.D
387	BIOL	AAK 230	8665	Ph.D

A row (record, entity)

Database Review

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_DEGREE
105	ACCT	KLR 229D	8665	Ph.D
114	ACCT	KLR 211	4436	Ph.D
301	ACCT	KLR 244	4683	Ph.D
435	ART	BBG 185	2278	Ph.D
387	BIOL	AAK 230	8665	Ph.D

A column (attribute, field)

Database Review

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_DEGREE
105	ACCT	KLR 229D	8665	Ph.D
114	ACCT	KLR 211	4436	Ph.D
301	ACCT	KLR 244	4683	Ph.D
435	ART	BBG 185	2278	Ph.D
387	BIOL	AAK 230	8665	Ph.D

Primary key (uniquely identifies a row)

Database Review

- Relationship
  - An association between entities (can be in the same or different tables)
  - Have cardinalities (1:M, 1:1, M:M)
- Foreign key
  - An attribute in one table that is a primary key in another table.
  - Connects the entities in a relationship

Database Review

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	EMP_NUM
10012	ACCT-211	1	MWF 8:00-8:50 a.m.	KLR 225	105
10013	ACCT-211	2	MWF 9:00-9:50 a.m.	KLR 225	105
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	KLR 225	342
10015	ACCT-212	1	MWF 10:00-10:50 a.m.	KLR 240	301
10016	ACCT-212				
10017	ACCT-311				
12001	ART-210				
12002	ART-340				

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_DEGREE
105	ACCT	KLR 229D	8665	Ph.D
114	ACCT	KLR 211	4436	Ph.D
301	ACCT	KLR 244	4683	Ph.D
435	ART	BBG 185	2278	Ph.D
387	BIOL	AAK 230	8665	Ph.D

Database Review

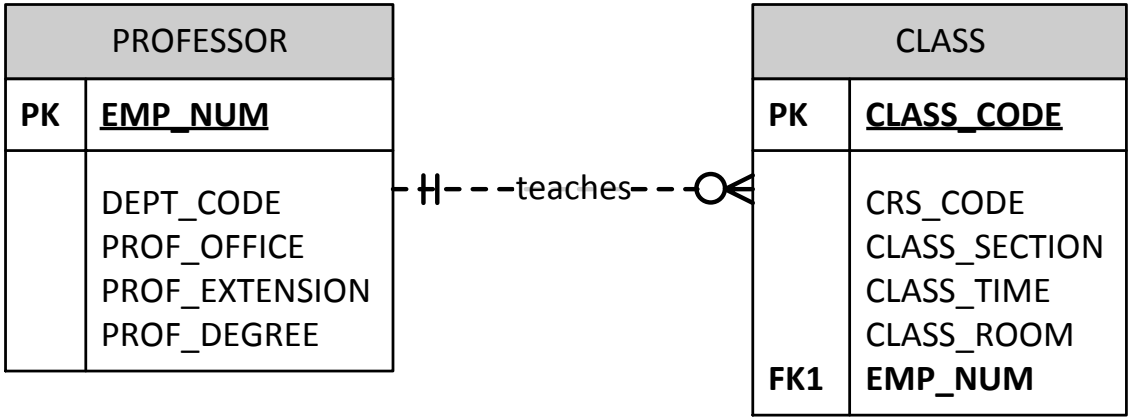
CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	EMP_NUM
10012	ACCT-211	1	MWF 8:00-8:50 a.m.	KLR 225	105
10013	ACCT-211	2	MWF 9:00-9:50 a.m.	KLR 225	105
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	KLR 225	342
10015	ACCT-212	1	MWF 10:00-10:50 a.m.	KLR 240	301
10016	ACCT-212				
10017	ACCT-311				
12001	ART-210				
12002	ART-340				

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_DEGREE
105	ACCT			
114	ACCT			
301	ACCT			
435	ART			
387	BIOL	AAK 230	8665	Ph.D

EMP_NUM in PROFESSOR is a foreign key in CLASS One professor can teach many classes.

Database Review

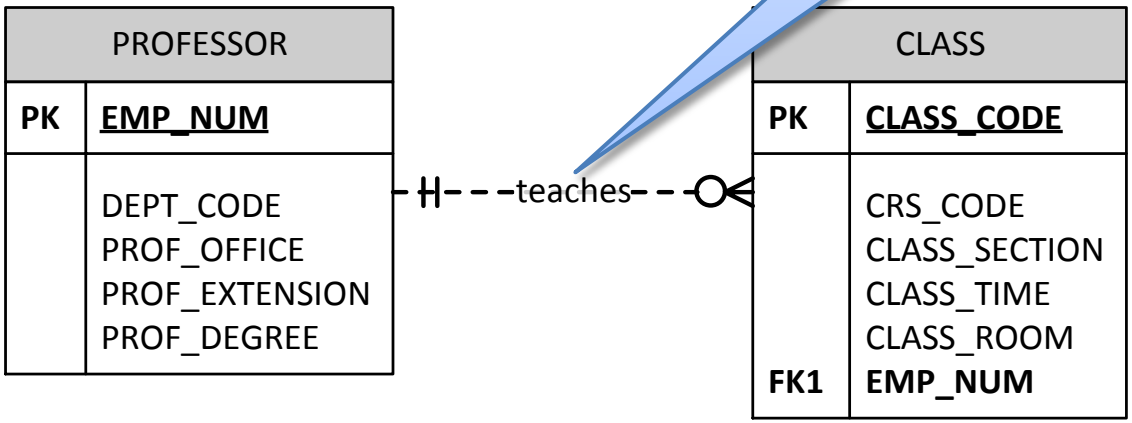
- ERD Modeling
  - Represents relationships graphically
  - Many notations (Crow's foot below)



Database Review

- ERD Modeling
  - Represents relationships graphicall
  - Many notations (Crow's foot belo

Q: how do you represent many-to-many relationships?



Database Review

- Normalization: eliminate redundancy and dependency
  - **First normal form**: tabular format, no repeating groups, primary key identified, non-key attributes are dependent on primary key.
  - **Second normal form**: In 1NF and no partial dependencies (no dependencies on just part of the key).
  - **Third normal form**: 2NF and no transitive dependencies (no nonkey attribute is dependent on another nonkey attribute),

Database Review

- Normalization: eliminate redundancy and dependency
  - **First normal form**: tabular format, no repeating groups, primary key identified, non-key attributes are dependent on primary key.
  - **Second normal form**: In 1NF and no partial dependencies (no dependencies on just part of the key).
  - **Third normal form**: 2NF and no transitive dependencies (no nonkey attribute is dependent on another nonkey attribute),

You should know how to do this.

Database Review

- MySQL vs SQLite

MySQL	SQLite
Runs as a server process	Runs as an embedded library
Data kept in files on the server, many databases per file	Data kept in a single flat file per database
Difficult to set up and manage	Simple to set up and manage
Has separate users, permissions	Has a single user with all permissions
Suitable for production environments	Suitable for development environments
Supports a robust superset of SQL syntax	Supports most, but not all SQL syntax

Database Review

- MySQL vs SQLite

MySQL	SQLite
Runs as a server proces	
Data kept in files on th databases per file	
Difficult to set up and r	
Has separate users, per	
Suitable for production enviro	Suitable for development environments
Supports a robust superset of SQL syntax	Supports most, but not all SQL syntax

If we are careful to use standard SQL, almost everything we write in PHP will be portable between these two database management systems.

Database Review

- SQL – CREATE TABLE

```
CREATE TABLE IF NOT EXISTS CLASS (
  CLASS_CODE INTEGER PRIMARY KEY NOT NULL,
  CRS_CODE VARCHAR(25) NOT NULL,
  CLASS_SECTION INTEGER NOT NULL,
  CLASS_TIME VARCHAR(30) NOT NULL,
  CLASS_ROOM VARCHAR(15) NOT NULL,
  EMP_NUM INTEGER NOT NULL,
  FOREIGN KEY(CRS_CODE) REFERENCES COURSE(CRS_CODE),
  FOREIGN KEY(EMP_NUM) REFERENCES EMPLOYEE(EMP_NUM)
);
```

Database Review

- SQL – CREATE TABLE

```
CREATE TABLE IF NOT EXISTS CLASS (
  CLASS_CODE INTEGER PRIMARY KEY NOT NULL,
  CRS_CODE VARCHAR(25) NOT NULL,
  CLASS_SECTION INTEGER NOT NULL,
  CLASS_TIME VARCHAR(30) NOT NULL,
  CLASS_ROOM VARCHAR(15) NOT NULL,
  EMP_NUM INTEGER NOT NULL,
  FOREIGN KEY(CRS_CODE) REFERENCES COURSE(CRS_CODE),
  FOREIGN KEY(EMP_NUM) REFERENCES EMPLOYEE(EMP_NUM)
);
```

These tables need to be created before CLASS is.

Database Review

- SQL – CREATE INDEX

```
CREATE INDEX IF NOT EXISTS STU_NAME_IDX ON
  STUDENT (STU_LNAME, STU_FNAME);
```

Why create indices?

Database Review

- SQL – INSERT

```
INSERT INTO STUDENT (
  STU_NUM, STU_LNAME, STU_FNAME, STU_INITIAL, STU_DOB,
  STU_HRS, STU_CLASS, STU_GPA, STU_TRANSFER, DEPT_CODE,
  STU_PHONE, EMP_NUM
)
VALUES (
  300245, 'Peppard', 'Randy', 'K', '6/21/1975 0:00:00',
  45, 'So', 2.61, 0, 'BIOL', 2134, 387
);
```

Database Review

- SQL – UPDATE

```
UPDATE STUDENT SET
  STU_INITIAL='L'
WHERE
  STU_NUM = 300245;
```

Affects one row. Why?

```
UPDATE STUDENT SET
  STU_CLASS = 'So'
WHERE
  STU_HRS > 32 AND
  STU_HRS <= 64;
```

Could affect multiple rows.

Database Review

- SQL – DELETE

```
DELETE
FROM
  STUDENT
WHERE
  STU_NUM = 300245;
```

Database Review

- SQL – Simple SELECT

```
SELECT *
FROM
  STUDENT
WHERE
  STU_GPA > 3.5
ORDER BY
  STU_LNAME, STU_FNAME ASC
```

Database Review

- SQL – Complex SELECT

```
SELECT COURSE.CRS_CODE || '.' || CLASS.CLASS_SECTION AS
COURSE,
  COURSE.CRS_DESCRIPTION AS DESCRIPTION
FROM COURSE, STUDENT, CLASS, ENROLL
WHERE
  STUDENT.STU_LNAME='Robertson' AND
  STUDENT.STU_FNAME='Anne' AND
  STUDENT.STU_NUM = ENROLL.STU_NUM AND
  ENROLL.CLASS_CODE = CLASS.CLASS_CODE AND
  CLASS.CRS_CODE = COURSE.CRS_CODE;
```

Database Review

- SQL – Complex SELECT

```
SELECT COURSE.CRS_CODE || '.' || CLASS.CLASS_SECTION AS
COURSE,
  COURSE.CRS_DESCRIPTION AS DESCRIPTION
FROM COURSE, STUDENT, CLASS, ENROLL
WHERE
  STUDENT.STU_LNAME='Robertson' AND
  STUDENT.STU_FNAME='Anne' AND
  STUDENT.STU_NUM = ENROLL.STU_NUM AND
  ENROLL.CLASS_CODE = CLASS.CLASS_CODE AND
  CLASS.CRS_CODE = COURSE.CRS_CODE;
```

Database Review

- SQL – Complex SELECT

```
SELECT COURSE.CRS_CODE || '.' || CLASS.CLASS_SECTION AS
COURSE,
  COURSE.CRS_DESCRIPTION AS DESCRIPTION
FROM COURSE, STUDENT, CLASS, ENROLL
WHERE
  STUDENT.STU_LNAME='Robertson' AND
  STUDENT.STU_FNAME='An
  STUDENT.STU_NUM = ENR
  ENROLL.CLASS_CODE = CL
  CLASS.CRS_CODE = COUR
```

	COURSE	DESCRIPTION
	ACCT-211.1	Accounting I
	CIS-370.1	Intro. to Systems Analysis
	MGT-340.1	Intro. to Management
	MKT-360.2	Intro. to Marketing
	MATH-243.1	Mathematics for Managers
	QM-261.2	Intro. to Statistics

Tools for working with SQLite

- SQLiteExpert Personal
  - http://www.sqliteexpert.com

Tools for working with SQLite

- Documentation:
  - http://www.sqlite.org/docs.html
- Naming convention
  - SQLite 3 (what we're using): file extension is .db3
- Quick demonstration after lecture

Using PHP with SQL

- PDO
  - PHP Data Objects (http://php.net/manual/en/book.pdo.php)
  - Abstracts the database into an object

```
try {
    $db = new PDO('sqlite:MicroUniversity.db3');
} catch (PDOException $e) {
    include 'error.inc';
    errorPage("Database error", $e);
    exit();
}
```

Using PHP with SQL

- PDO
  - PHP Data Objects (http://php.net/manu
  - Abstracts the databas to an object

The "DSN" (data source name) designates which database to use. MySQL requires authentication

```
try {
    $db = new PDO('sqlite:MicroUniversity.db3');
} catch (PDOException $e) {
    include 'error.inc';
    errorPage("Database error", $e);
    exit();
}
```

Using PHP with SQL

- PDO
  - PHP Data Objects (http://php.net/manual/
  - Abstracts the database

```
try {
    $db = new PDO('sqlite:MicroUniversity.db');
} catch (PDOException $e) {
    include 'error.inc';
    errorPage("Database error", $e);
    exit();
}
```

Exception handling. "Try" to connect to the database. If an error is thrown, then "catch" it and handle it properly.

Using PHP with SQL

-

```php
<?php
include_once 'ui.inc';
include_once 'util.inc';

function errorPage($title, $exception) {
    head($title);
    print "<h1>$title</h1>\n";
    print "<p>Problem: {$exception->getMessage()}.</p>";
    foot();
}
?>
```

```
try
} c
    errorPage
    exit();
}
```

Every exception has a message that can be printed.

Using PHP with SQL

- Universal way to execute SQL in PHP
  - Prepare a statement
  - Bind needed parameters
  - Execute
  - Fetch and return rows (if querying)

Using PHP with SQL

- Universal way to execute SQL in PHP
  - Prepare a statement
  - Bind needed parameters
  - Execute
  - Fetch and return rows (if

Prepared statements with bound parameters avoid a category of security vulnerabilities known as “SQL Injection,” attacks

Using PHP with SQL

- Universal way to execute SQL in PHP

```
function findStudentByName($lname, $fname) {
    global $db;
    $lname = "%{$lname}%";
    $fname = "%{$fname}%";
    $st = $db -> prepare('SELECT * FROM STUDENT WHERE ' .
        'STU_LNAME LIKE ? AND STU_FNAME LIKE ? ' .
        'ORDER BY STU_LNAME, STU_FNAME');
    $st -> bindParam(1, $lname);
    $st -> bindParam(2, $fname);
    $st -> execute();
    return $st -> fetchAll(PDO::FETCH_ASSOC);
}
```

Using PHP with SQL

- Universal way to execute SQL in PHP

```
function findStudentByName($lname, $fname) {
    global $db;
    $lname = "%{$lname}%";
    $fname = "%{$fname}%";
    $st = $db -> prepare('SELECT * FROM STU
        'STU_LNAME LIKE ? AND STU_FNAME
        'ORDER BY STU_LNAME, STU_FNAME
    $st -> bindParam(1, $lname);
    $st -> bindParam(2, $fname);
    $st -> execute();
    return $st -> fetchAll(PDO::FETCH_ASSOC);
}
```

This returns the resulting rows as an array of associative arrays whose keys are the database field names.

Using PHP with SQL

```
Array (
   [0] => Array (
         [STU_NUM] => 332345
         [STU_LNAME] => Paulus
         [STU_FNAME] => Annelise
         [STU_INITIAL] => H
         [STU_DOB] => 1/5/1974 0:00:00
         [STU_HRS] => 108
         [STU_CLASS] => Sr
         [STU_GPA] => 3.92
         [STU_TRANSFER] => 0
         [DEPT_CODE] => MATH
         [STU_PHONE] =>
         [EMP_NUM] => 155
      )
   [1] => Array (
         [STU_NUM] => 311198
         [STU_LNAME] => Robertson
         [STU_FNAME] => Anne
         [STU_INITIAL] => B
         [STU_DOB] => 11/15/1970 0:00:00
         [STU_HRS] => 93
         [STU_CLASS] => Jr
         [STU_GPA] => 3.04
         [STU_TRANSFER] => 0
         [DEPT_CODE] => CIS
         [STU_PHONE] => 2215
         [EMP_NUM] => 162
      )
)
```

This returns the resulting rows as an array of associative arrays whose keys are the database field names.

Using PHP with SQL

- Using result rows

```
$rows = findStudentByName($lname, $fname);
print "<table><tr><th>First</th><th>Last</th></tr>\n";
foreach ($rows as $row) {
    print "<tr><td>{$row['STU_FNAME']}</td>" .
        "<td>{$row['STU_LNAME']}</td></tr>\n";
}
print "</table>";
```

Using PHP with SQL

- Using

foreach walks through every row of the result set.

```
$rows = findStud...yName($lname, $fname);
print "<table><tr><th>First</th><th>Last</th></tr>\n";
foreach ($rows as $row) {
    print "<tr><td>{$row['STU_FNAME']}</td>" .
        "<td>{$row['STU_LNAME']}</td></tr>\n";
}
print "</table>";
```

Access data within a row by using the column name as an index.

Using PHP with SQL

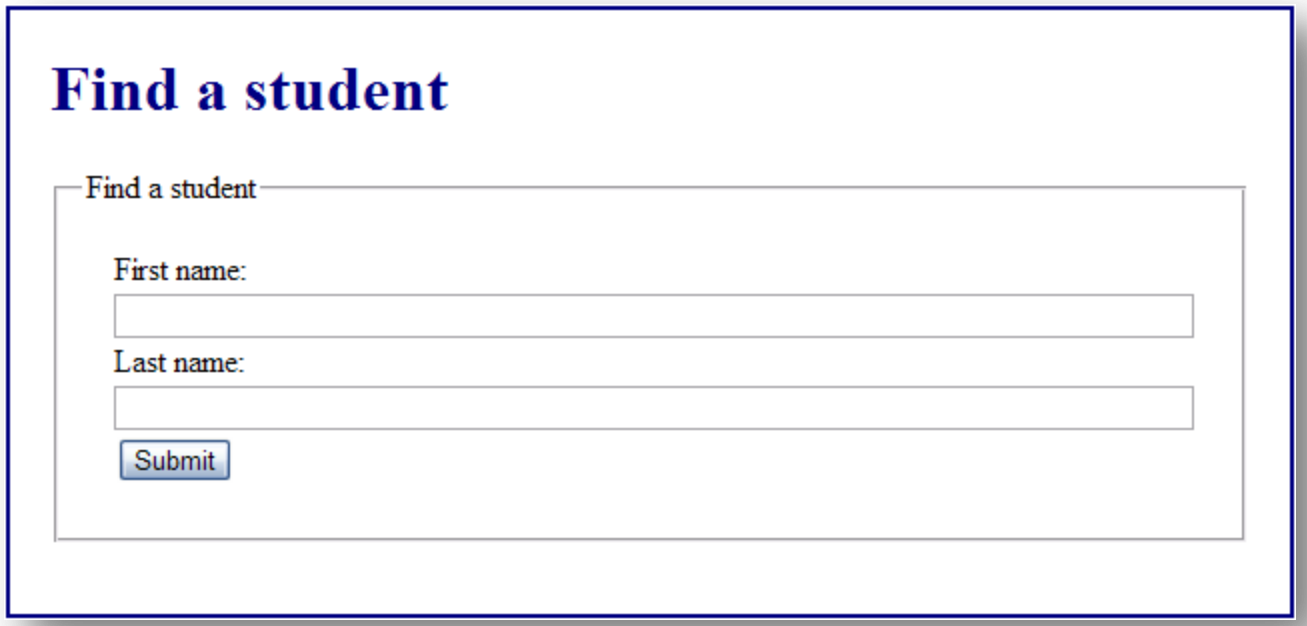
- Using result rows

```
<?php $rows = findStudentByName($lname, $fname) ?>
<table>
   <tr>
      <th>First</th>
      <th>Last</th>
   </tr>;
   <?php foreach ($rows as $row) : ?>
   <tr>
      <td><?php echo $row['STU_FNAME']; ?></td>
      <td><?php echo $row['STU_LNAME']; ?></td>
   </tr>
   <?php endforeach; ?>
</table>
```

A more tag-oriented approach to tables.

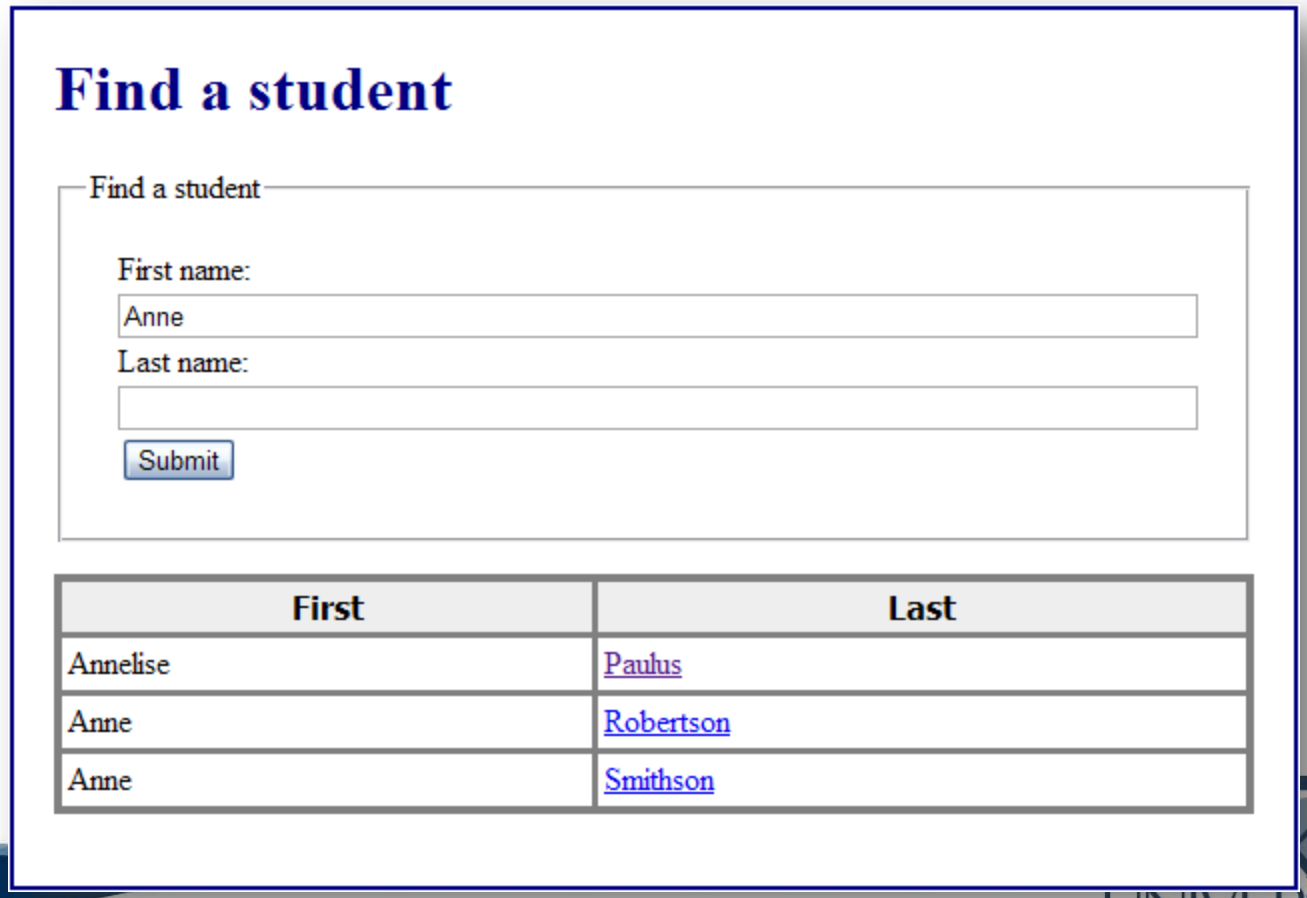
Using PHP with SQL

- Simple walkthrough



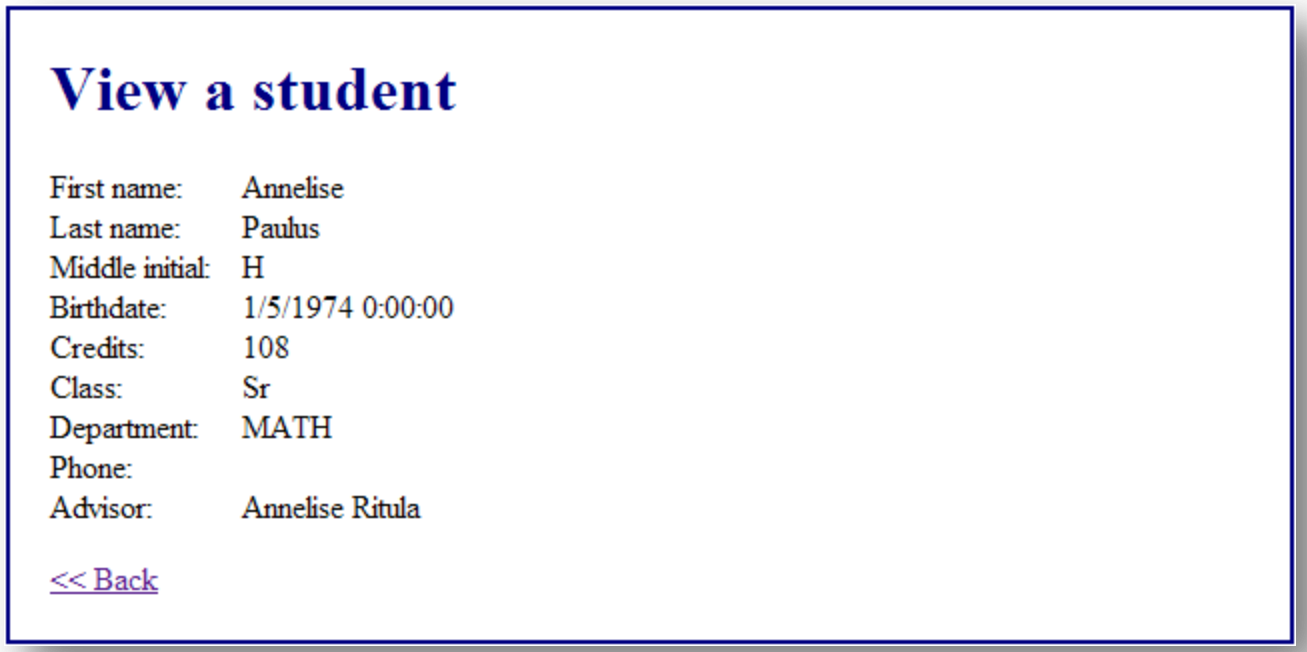
Using PHP with SQL

- Simple walkthrough



Using PHP with SQL

- Simple walkthrough



View a student

First name:	Annelise
Last name:	Paulus
Middle initial:	H
Birthdate:	1/5/1974 0:00:00
Credits:	108
Class:	Sr
Department:	MATH
Phone:	
Advisor:	Annelise Ritula

<< Back

Using PHP with SQL

- Show me the code!
  - Available from http://cs.franklin.edu/~sharkesc/webd236/

Logging

- Logging class Lib/Logger.inc

```
class Logger {
    private static $instance;
    private $filename;
    private $level;

    const DEBUG = 0;
    const INFO = 1;
    const WARN = 2;
    const ERROR = 3;

    private function __construct($level=self::DEBUG, $filename='debug.log') {
        // log everything
        $this -> level = $level;
        $this -> filename = $filename;
    }
```

Logging

- Logging class Lib/Logger.inc

```
public static function instance($level=self::DEBUG, $filename='debug.log) {
    if (!isset(self::$instance)) {
        self::$instance = new Logger($this->level, $this->filename);
    }
    return self::$instance;
}

public function debug($message) {
    return $this -> log(self::DEBUG, $message);
}
```

Logging

- Logging class Lib/Logger.inc

```
public function setLevel($level) {
    $this -> level = $level;
}

public function setFilename($filename) {
    $this -> filename = $filename;
}
```

Logging

- Logging class Lib/Logger.inc

```
public function info($message) {
    return $this -> log(self::INFO, $message);
}

public function warn($message) {
    return $this -> log(self::WARN, $message);
}

public function error($message) {
    return $this -> log(self::ERROR, $message);
}
```

Logging

- Logging class Lib/Logger.inc

```
    private function log($level, $message) {
        if ($level >= $this -> level) {
            $names = array('DEBUG', 'INFO', 'WARN', 'ERROR');
            $timestamp = date("Y-m-d H:i:s", time());
            $fd = fopen($this -> filename, "a");
            fprintf($fd, "%s %s %s\r\n",
                $timestamp, $names[$level], $message);
            fclose($fd);
        }
    }
}
```

Upcoming Deadlines

- Readings for next week
  - Chapters 5 and 6 in *PHP and MySQL*
- Assignments
  - Homework 1 due end of week 2
  - Homework 2 due end of week 3
  - Lab 1 due end of week 4
- Next week:
  - Avoiding ugly URLs, MVC pattern, and testing/debugging

General Q & A

- Questions?
- Comments?
- Concerns?

FRANKLIN UNIVERSITY