FANSHAWE

INFO-6076

Web Security

Server-Side Languages

PHP & SQL

Agenda

- PHP Basics
- Databases
- Lab 06 Overview

PHP Basics

PHP

PHP: Hypertext Preprocessor

- Originally called "Personal Home Pages tools"

Server-side scripting language

- Runs on a server, not your local machine
- Removes some of the easier attacks such as using a proxy on a local machine

- PHP is free (open source)
- php.net is a great source of information

PHP

Widely-used general-purpose scripting language

- Has similar syntax to C, but is much simpler

Primarily used for Web Development

- Generating dynamic, interactive web pages
- Interacting with database management systems (DBMS)
  - Relational database management systems (RDBMS)
  - Object-oriented database management systems (OODBMS)

PHP

When it comes to web applications PHP is well suited to interacting with the back-end database

- Validating user input
- Sanitizing user input
- Accepting user input
- Interacting with the contents of the database
  - Adding, modifying, deleting, etc.

PHP handles connection and communication with databases

PHP & MySQL

MySQL is a relational database management system

- Like PHP, MySQL is free (GPL license)
- Lightweight, fast, easy to use RDBMS

PHP can also interact with other databases

- MS SQL Server
- PostgreSQL
- LDAP
- Many More

HTML & PHP

You will usually find HTML and PHP code in a PHP file

- HTML is used for the static content
- PHP is used to generate the dynamic content

PHP code returns HTML that is displayed by the web browser

- The client can’t view the PHP, they only see the HTML returned by the web server

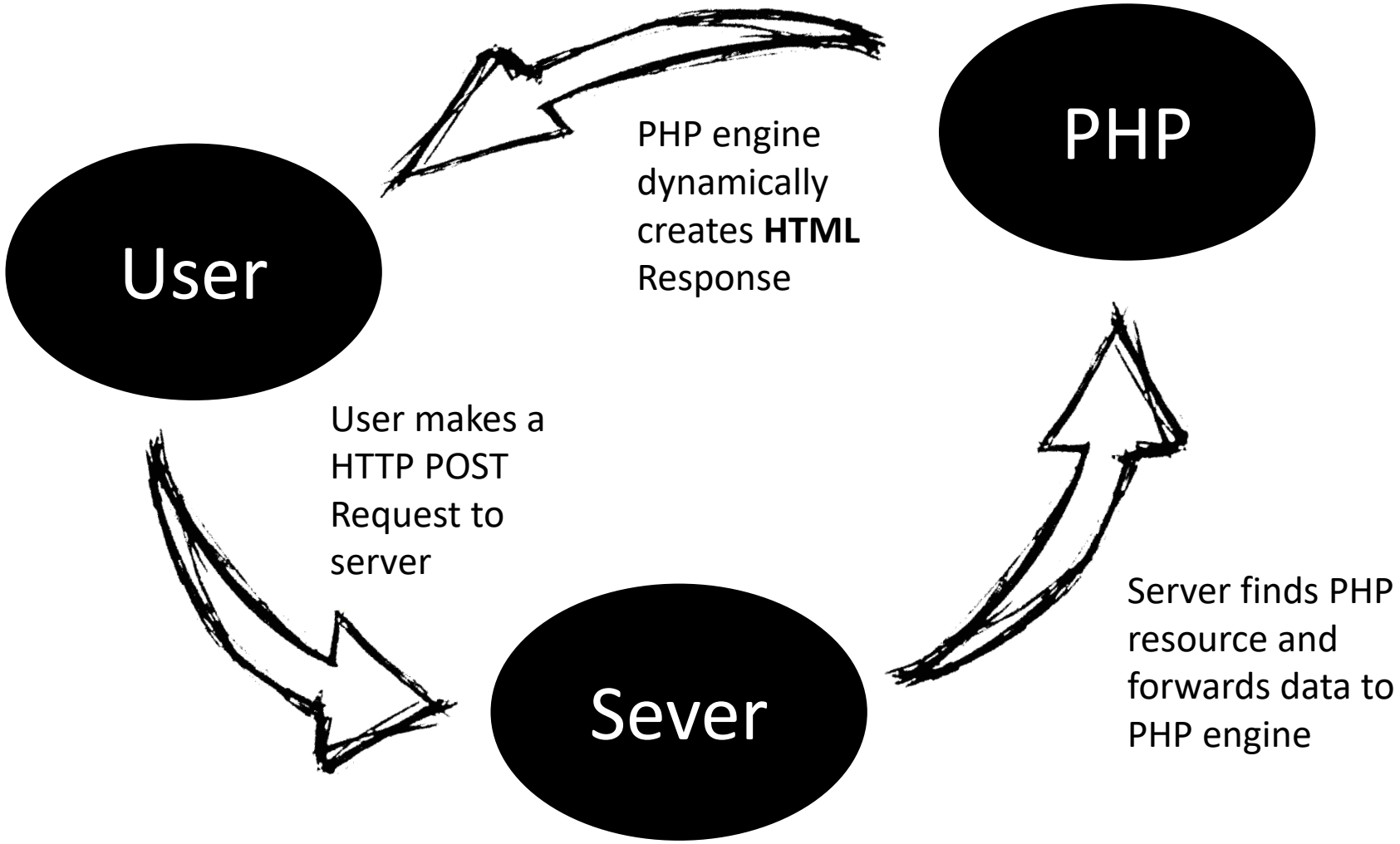
Running PHP

You will need a web server running the PHP software (PHP Engine/Module)

- PHP software runs on all popular web servers
- We are using a LAMP stack on an Ubuntu server
  - Linux operating system
  - Includes Apache, MySQL, and PHP

! It is important to remember that the PHP code is run on the server and the page is returned to the user

Running PHP


PHP
PHP engine dynamically creates **HTML** Response
User
User makes a HTTP POST Request to server
Sever
Server finds PHP resource and forwards data to PHP engine

Basic PHP

- PHP files use a default extension of **.php**
- A PHP file will usually contain HTML and PHP code
- The PHP code can be placed anywhere in the file
- PHP scripts start with **<?php** and end with **?>**
  - Don't make the mistake of trying to use <?php> or <?>
- Everything from the <?php …to the… ?> and the content between is considered a code block

PHP Code Block

You can insert blocks of PHP code right into an HTML document

```
<!DOCTYPE html>
<html lang="en">
    <head>
    <meta charset="UTF-8" />
    <title>INFO-6076</title>
    <style type="text/css">
    h2 {font: 35px Arial; color:#00FF00;}
    </style>
    </head>
    <body>
    <?php
    echo '<h2>This is the easiest class</h2>';
    print '<p>I will get the best grade here</p>';
    ?>
    </body>
    </html>
```

PHP Code Block

The PHP engine returns dynamically created HTML output (Not the PHP source code)

View Page Source:

```html
<!DOCTYPE html>
<html lang="en">
    <head>
    <meta charset="UTF-8" />
    <title>INFO-6076</title>
    <style type="text/css">
    h2 {font: 35px Arial; color:#00FF00;}
    </style>
    </head>
    <body>
    <h2>This is the easiest class</h2>
    <p>I will get the best grade here</p>
    </body>
    </html>
```

Browser Output:

This is the easiest class

I will get the best grade here

Basic PHP

- You will sometimes see php scripts starting with <? and ending with ?>
  - short_open_tag
  - Not recommended because XML files use <?
- To use the short form tags you would need to enable this in your **php.ini** file
- The php.ini file contains your PHP configuration
  - php.ini can be edited with a text editor

Basic PHP

- We are going to use the **echo** and **print** language constructs to output strings
  - They aren't functions, so you don't need to use parentheses ()
- You can do error checking with **print** as it returns an int
  - 1 if it worked
  - 0 if it failed

```
echo "Using echo to output a string";
print  "Using print to output a string";
```

Note that each code line in PHP ends with a semicolon (**;**)
It distinguishes one set of instructions from another.

printf()

We can also use the **printf()** function to output formatted strings

- **printf()** is an actual function so you need to use the parentheses

Allows you to output a formatted string

- Works well with variables
  - Code:

```
printf("I got %0.2f on my last test.", 95.4567);
```

  - Output:

I got **95.46** on my last test.

PHP Comments

- There are three ways you can comment out text or code with PHP
  - Inline Comments:
    - // Use two forward slashes for a single inline comment
  - Block Comments:
    - /* Use a forward slash asterisk, then a closing asterisk and a forward slash */
  - Shell Comments
    - # Using a hash
    - Strongly discouraged

PHP Variables

- As with other languages variables are containers for storing information
- Rules for PHP variables:
  - starts with a dollar sign ($) followed by the variable name: $x $x2 $MyVar
  - Must begin with a letter or an underscore
  - Can only contain alpha-numeric characters and underscores
  - Should not contain spaces
  - Case sensitive

PHP Variables

- With PHP a variable is declared as soon as you assign a value to it
- Examples:
  - $x=5;
  - $hello="Hello World!";
- Variable Notes:
  - You need to use quotes (" or ') around your text
  - You don’t need to declare (define) a data type
    - This makes PHP is a Loosely Type Language

PHP Variables

You can use the concatenation operator (.) to join two string values. (JavaScript uses +)

Example:

```
$myname=“Art Mack";
$hello='Hello World!';
echo $myname . " using variables to say " . $hello;
```

Output: Art Mack using variables to say Hello World!

Note: You can use double quotes (") or single quotes (') in the strings.
Use double quotes if you want to include variables

Example:
```
echo "$hello";  // the output is:   Hello World!
echo '$hello';  // the output is:  $hello
```

phpinfo()

- Commonly used to check configuration settings and environmental variables for a given system

- Can be used on its own or with a number of constants
  - phpinfo(INFO_GENERAL)
  - phpinfo(INFO_CREDITS)
  - phpinfo(INFO_CONFIGURATION)
  - phpinfo(INFO_MODULES)
  - phpinfo(INFO_ENVIRONMENT)
  - phpinfo(INFO_VARIABLES)
  - phpinfo(INFO_LICENSE)
  - phpinfo(INFO_ALL)

Main PHP Security Areas

- Validating and Sanitizing User Input
- Preventing SQL Injection
- Preventing XSS (Cross-Site Scripting)
- Preventing Remote Execution
- Security for Temporary Files
- Preventing Session Hijacking

We will talk about specifics when covering the topics individually

PHP Specific Input Risks

Risk stems from the fact that PHP is a loosely typed language

- No need to declare variables
- Without proper input validation users can supply data of the wrong type, or size, or containing special characters

Controls

- Declare your variables
- Use Input Validation
- Check the type, length and format of inputs
- Sanitize values

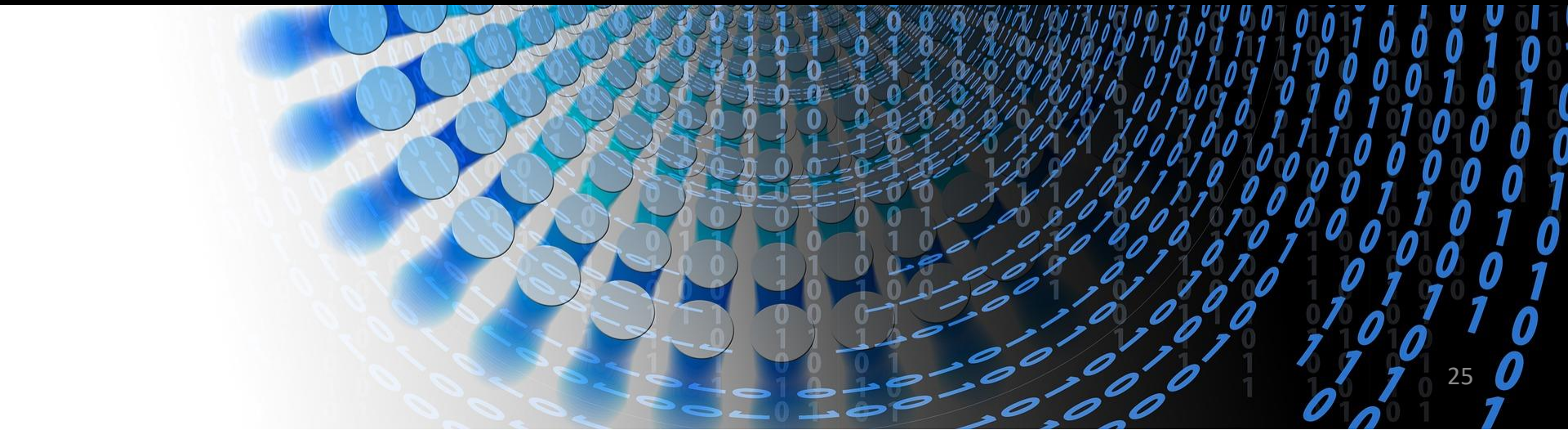
PHP Security Basics

It is important to consider the key principles

- Ensure you are using the current version of PHP
- Balance Risk and Usability
- Control User Input with Input validation
- Track Your Data
  - Control how data enters your application, is stored, and how it exits
- Sanitize the Data (escaping, encoding, etc.)
  - Controlling how data is sent to clients, databases, etc.

FANSHAWE

Databases



Databases

Databases are used for a variety of purposes

- Client Information
- Product and Pricing Information
- Banking Information
- Inventory Information
- Much More

Databases and Database Management Systems

- The database contains the actual data
- The database management system is used to interact with that data

Databases

Database (DB) is simply a collection of data.
In Relational Database (RDB), data is organized into tables.

Student_ID	Name	Major	Grade
10145	Michael	CS	95
10146	Dennis	PHYS	75
10147	Boris	MATH	89
	…	…	…

Student_ID	Citizenship
10145	Canada
10146	Canada
10147	France
	…

Term "Relational" refers to the relationship between columns within a table and also to links between tables.

Database Management System (DBMS) is software that allows you to maintain and utilize the collections of data.

Relational Database Management System (RDBMS) is DBMS that is based on the relational model (Oracle, DB2, MySQL).

Databases

Table

An individual "**relation**" in the relational database

Relates keys to values.

Student_ID	Name	Major	Grade
10145	Michael	CS	95
10146	Dennis	PHYS	75
10147	Boris	MATH	89
	…	…	…

Table Column

An attribute (fields) in the table.

Table Row

An entity (records) in the table.

Typically has a value for each column

Schema

A description of a particular collection of data (a set of table designs that determine a database).
Does not yet include the data – simply shows how it will be structured in the database

Database Relationships

The power of relational databases comes from the tables being related to each other.

These relationships are controlled through keys:

- **Primary Key**
  - An attribute that uniquely identifies an entity in a table.
  - Any table with lots of records should have one.
  - Example: Student ID 10147 identifies student 'Boris'
- **Foreign Key**
  - Used to provide relationships between tables
  - Usually related to the primary key in the related table
  - Example: Student ID 10147 identifies student 'Boris' as a citizen of France

Database Relationships

Primary Key can be the same as **Foreign Key**

Student_ID	Name	Major	Grade
10145	Michael	CS	95
10146	Dennis	PHYS	75
10147	Boris	MATH	89
	…	…	…

Student_ID	Citizenship
10145	Canada
10146	Canada
10147	France
	…

Primary Key can be different from **Foreign Key**

Primary Key	Food	Foreign Key
1	Carrot	2
2	Apple	1
3	Celery	2
4	Turnip	2
5	Orange	1

Primary Key	Classification
1	Fruit
2	Vegetable

Databases: Relationships

Relationships: how does the data in different tables relate?

One-to-One

An entity in a table corresponds to a single entity in another table.
The relationship is typically established using a foreign key for one or both entities.

Example: If we have a table for Student_Info and a table for Academic_History, there is a One-to-One relationship between them.

One-to-Many

An entity in a table corresponds to 1 or more entities in another table.

Example: If the table for Academic_History has an entry for each term, the relationship now becomes one student to many terms.

Databases: Relationships

Many-to-Many

Multiple entities in one table correspond to multiple entities in another table.

Example: Tables Student_Info and Courses_Taken have a many to many relationship, since a student can take many courses and each course can be taken by many students.

Many-to-Many relationship is often defined by a separate table, which in fact changes it into two One-to-Many relationships.

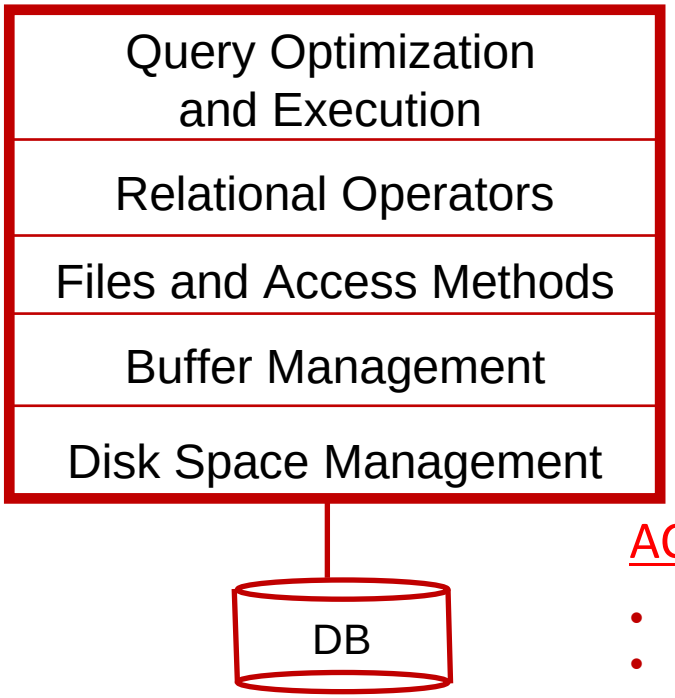
Example: If we create a new table Student_Courses, we can have each entity be a pair: Student_Id, Course_id

Now Student_Info has a One-to-Many relationship with Student_Courses, and so does Courses_Taken
So we created two One-to-Many relationships.

Databases

Database Engine ("Storage Engine")

Is the underlying software component for storing, processing and securing data.



A typical DBMS has a layered architecture.
Each database system has its own variations.

These layers must consider concurrency control and recovery.

Transaction is a sequence of database actions (reads/writes).

ACID Properties of Database Transactions:

- Atomicity (all-or-nothing property)
- Consistency (rows affected by transaction remain consistent)
- Isolation (no transaction interference)
- Durability (committed transactions are protected against crashes)

There are Transactional and Non-Transactional Storage engines

Transaction Example using ACID

A transaction is a sequence of operations performed on a database that work as a unit

Example: Purchasing a ticket to a show

Check availability of tickets

- **SELECT […]**

Subtract a ticket from available tickets

- **UPDATE […]**

Add to sold tickets

- **UPDATE […]**

Mark the transaction a success

- **UPDATE […]**

Transaction Example using ACID

ACID: Atomicity

All-or-Nothing property

If only some of the operations were successful, the transaction is not considered complete

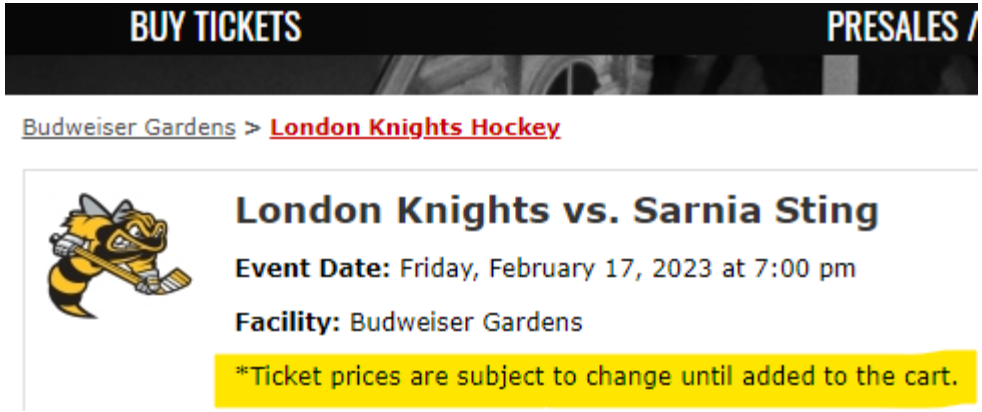
- ✓ Check availability of tickets
  - **SELECT […]**
- ✓ Subtract a ticket from available tickets
  - **UPDATE […]**
- ✗ Add to sold tickets
  - **UPDATE […]**
- ✗ Mark the transaction a success
  - **UPDATE […]**

Transaction Example using ACID

ACID: Consistency

Data affected by transaction is consistent

Constraints – If the transaction is not completed, the data is not affected



Transaction Example using ACID

ACID: Isolation

No transaction interference.

At the highest isolation level, the database management system guarantees that even if multiple simultaneous transactions take place, the integrity of the data will not be affected

A lower isolation level can help speed things up, but lowers the guarantee that multiple transactions will not affect each other

Transaction Example using ACID

ACID: Durability

Committed transactions are protected against crashes

Anything in the buffer/memory has been permanently written to the disk

If there were to be a power outage for example, the transaction has been completed and the data would not be lost

FANSHAWE

Databases: Vendors

Commercial

- Oracle
- IBM (DB2, Informix)
- Microsoft SQL
- Sybase (ASE, IQ)
- Teradata

Open Source

- MySQL (InnoDB and MyISAM storage engines)
- Percona (optimized version of MySQL)
- MariaDB, fork of MySQL (XtraDB storage engine, Aria)
- PostgreSQL
- SQLite

MySQL

MySQL is a Relational Database Management System.

SQL stands for the Structured Query Language.
It defines how to insert, retrieve, modify and delete data.

MySQL is the most popular open source DBMS: http://www.mysql.com

- MySQL has more than 20 million installations.
- Supported by all major programming languages.
- Has powerful security options (access and privilege controls)
- MySQL runs as a server providing multi-user access to a number of databases.
- It is a cross platform (Windows, Linux, Unix, OS X, Solaris, etc.) database server.
  - Multiple **storage engines** (MyISAM, InnoDB, Memory, etc.)
  - **Views** creation and update
  - **Transactions** with the InnoDB Engine
  - Sub Queries / Nested Select
  - Primary and Foreign keys and indexing

MySQL: Storage Engines

CSV

The engine stores data in text files using comma-separated values format.

Memory

The engine creates tables with contents that are stored in memory.

MyISAM

Default in 5.1. Non-transaction Engine.

No transaction overhead: much faster, lower disk space requirements, less memory required to perform updates.

Atomicity (all-or-nothing property)
Consistency (rows affected by transaction remain consistent)
Isolation (no transaction interference)
Durability (committed transactions are protected against crashes)

InnoDB

Default in 5.5. Transaction-safe (ACID compliant) Engine.

Safe, can combine many statements with one **COMMIT**, can execute **ROLLBACK**, row-level locking.

http://dev.mysql.com/doc/refman/5.6/en/storage-engines.html

http://dev.mysql.com/doc/refman/5.1/en/storage-engine-compare-transactions.html

MySQL: Disk Space Management

MySQL server can store several databases.
Databases are usually stored as directories (MyISAM).
Tables are stored as files inside each database (directory).

Each **MyISAM** table has three files:

- table.FRM file containing information about the table structure.
- table.MYD file containing the row data.
- table.MYI containing any indexes belonging with this table, as well as some statistics about the table.

InnoDB:
Data and indexes in tablespaces. Made up of one of more datafiles.
Has a special flag: **innodb_file_per_table**

MySQL: Basic Operations

- Create table
- Insert records
- Retrieve records
- Update records
- Delete records
- Modify table
- Join table
- Drop table

- Count, Like, Order by, Group by, limit
- Optimize table
- Advanced (sub-queries, stored procedures, triggers, views …)

http://dev.mysql.com/doc/refman/5.6/en/differences-from-ansi.html

FANSHAWE

MySQL: Frequently Used Statements

CREATE

create databases and tables

DROP

Delete databases and tables

ALTER

alter the structure of a table

Database & Table Maintenance

INSERT

insert a new row in a table

UPDATE

update rows in a table

SELECT

select table rows based on certain conditions

DELETE

delete one or more rows of a table

Main Data Manipulation SQL Statements

http://dev.mysql.com/doc/refman/5.6/en/sql-syntax-data-manipulation.html

MySQL: Monitor (Console Access)

MySQL allows console access:

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

Typical usage if host is a localhost (example – your Ubuntu VM with the LAMP stack):

```
shell> mysql -u someusername -p
Enter password: *****

Server version: 5.xx.xxxx
Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
```

MySQL: Monitor (Console Access)

CREATE a database (You need to have permissions to create a database !!):

mysql> **CREATE DATABASE** *database_name*;

USE a database:

mysql> **USE** *database_name*;

CREATE a table:

mysql> **CREATE table** *table_name (column definitin, column definition, ….);*

MySQL commands are **NOT** case-sensitive.

Often, command keywords are written in all caps, just to distinguish them from your user-specific values such as table and column names. However, this isn't necessary.

MySQL commands have to end with a semicolon (**;**).

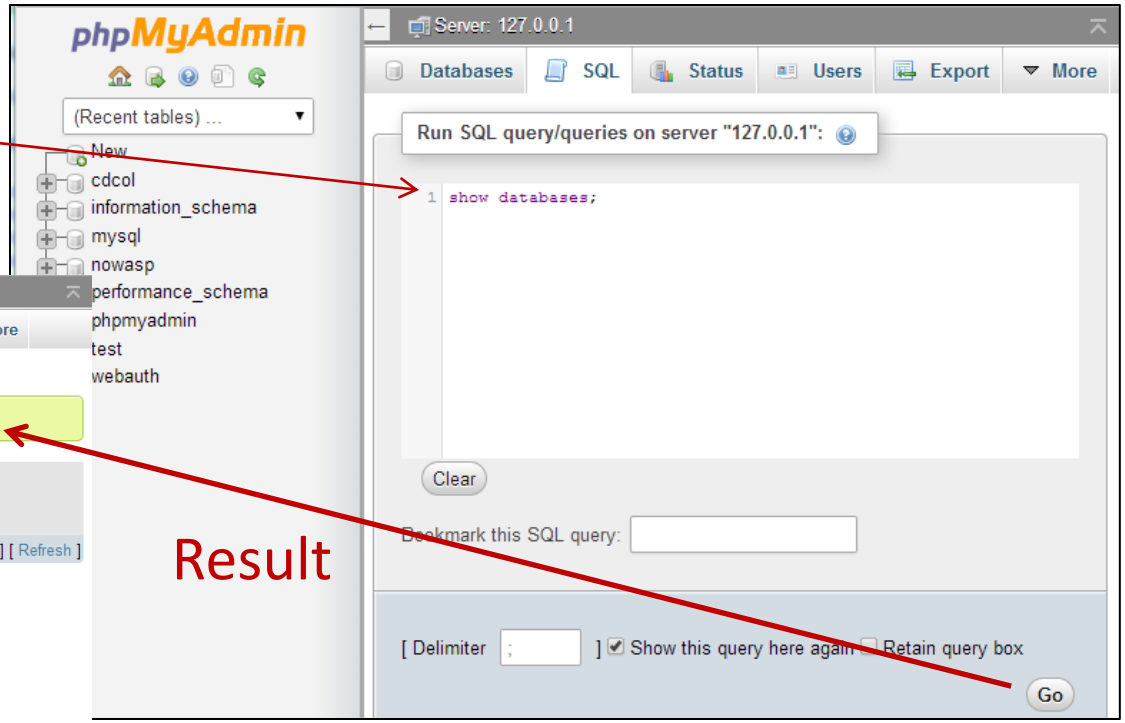
If you forget this, the system will put an arrow on the next line: **->**
which means it is waiting for you to finish the command.

```
mysql> show databases
  ->
```

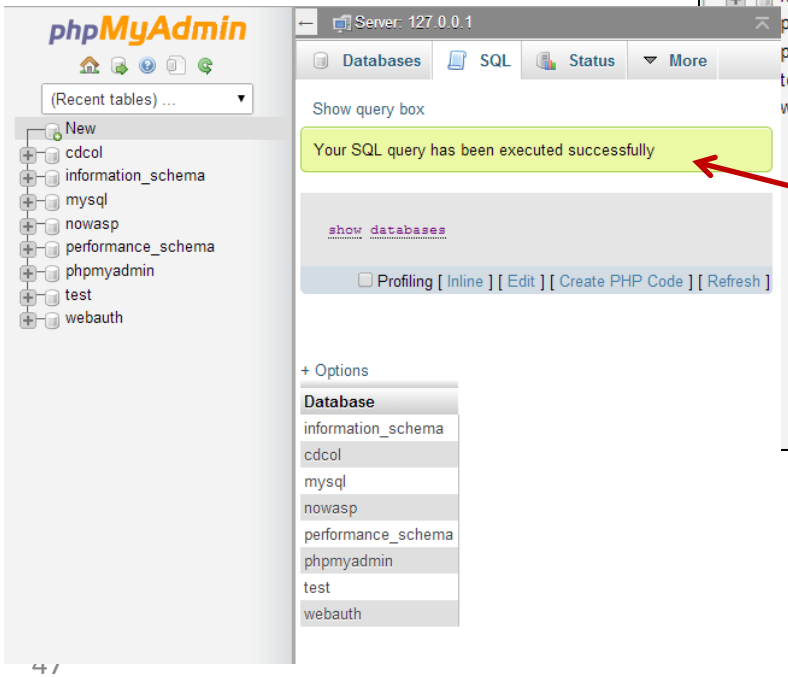
MySQL: PHPMyAdmin

phpMyAdmin is a web interface that allows us to manage MySQL

We execute SQL statement:
SHOW databases;



Result



MySQL Console / PHPMyAdmin SQL

What are the current databases on the server?

mysql> **SHOW** databases;

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
```

Create a database **test_db**:

Can do it as a **privileged** user only!

mysql> **CREATE** database test_db;

Choose **test_db** database to use**:**

mysql> **USE** test_db;

Database changed

What tables are currently stored in the **test_db** database?

mysql> **SHOW** tables;

Empty set (0.00 sec)

SQL

Common components of a SELECT statement

- FROM
  - Indicates the tables from which data is retrieved
- WHERE
  - Restricts the rows returned by the query
- ORDER BY
  - Indicates the columns used to sort the resulting data

Note: We will do more SQL when we look at SQL injection

Database Security Concepts

- Controlling access to the database, tables, etc.
- Controlling access to administrative features
  - Creating Tables
  - Altering Tables
  - Deleting Tables
- Controlling the type of access people have
  - Read-only (Select), write-only (Update), etc.
  - Access to specific tables / columns
- Restricting the ability to manage user accounts

Creating / Removing Users in MySQL

When creating / removing users there are 2 main statements:

- CREATE / DROP – creates / deletes a new user
- GRANT / REVOKE – specifies / revokes user's privileges
  - ON - specifies the databases and tables the user can access
  - TO - specifies the user and local or remote access
  - IDENTIFIED BY ‘password’
    - specifies the password for the user (only for **GRANT**)

MySQL Console / PHPMyAdmin SQL

Grant access to a database:

mysql> **GRANT** all on *databasename*.* to *username@localhost* identified by 'password';

Grant limited access to a database:

mysql> **GRANT** select, insert on *databasename*.* to *username@localhost* identified by 'password';

Delete user/privileges:

mysql> **DROP USER** *'username'*;

Flush privileges!:

mysql> **FLUSH privileges**;

See a list of the privileges granted to a specific user:

mysql> **SHOW GRANTS** for *'username'@'localhost'*;

```
mysql> show grants for 'root'@'localhost';
+-----------------------------------------------------------------------------------------------------------+
| Grants for root@localhost                                                                                 |
+-----------------------------------------------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY PASSWORD '*ED1B38B……' WITH GRANT OPTION  |
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION                                              |
+-----------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

MySQL: Privileges

List of Global privileges:

Global privileges ☐ Check All

Note: MySQL privilege names are expressed in English

Data

- ☐ SELECT
- ☐ INSERT
- ☐ UPDATE
- ☐ DELETE
- ☐ FILE

Structure

- ☐ CREATE
- ☐ ALTER
- ☐ INDEX
- ☐ DROP
- ☐ CREATE TEMPORARY TABLES
- ☐ SHOW VIEW
- ☐ CREATE ROUTINE
- ☐ ALTER ROUTINE
- ☐ EXECUTE
- ☐ CREATE VIEW
- ☐ EVENT
- ☐ TRIGGER

Administration

- ☐ GRANT
- ☐ SUPER
- ☐ PROCESS
- ☐ RELOAD
- ☐ SHUTDOWN
- ☐ SHOW DATABASES
- ☐ LOCK TABLES
- ☐ REFERENCES
- ☐ REPLICATION CLIENT
- ☐ REPLICATION SLAVE
- ☐ CREATE USER

Resource limits

Note: Setting these options to 0 (zero) removes the limit.

MAX QUERIES PER HOUR	0
MAX UPDATES PER HOUR	0
MAX CONNECTIONS PER HOUR	0
MAX USER_CONNECTIONS	0

MySQL: Grant Privileges

- By default, even if you grant a user all privileges at the command prompt, they can't themselves grant privileges to others.
  You need to specifically give a user this permission
  - WITH GRANT OPTION

- Even if a user has the grant permission they can only grant permissions they have themselves
  - They can't create a more powerful user

Specifying Scope with ON

- Global Privileges
  - ON *.*
- Specific Database, all Tables
  - ON specific_db.*
- Specific Database, Specific Table
  - ON specific_db.specific_table
- Specific Columns
  - UPDATE (col1,col2) ON specific_db.specific_table
- Routine
  - Apply to stored routines (functions & procedures)

User, Host & Password

- You use TO to specify the user
  - TO user@host
  - You can use localhost for the host portion to specify that the user can only access the server locally
  - You could also specify a specific IP the connection is allowed from
    - user@192.168.101.11
- You use INDENTIFIED BY to specify a password for the user
  - IDENTIFIED BY 'password'

Securing PHPMyAdmin

We need to make sure **phpMyAdmin** is locked down

- By default there is no password for the root account
- Adding a password for the root account is a two step process:
  - Modify the root account in phpMyAdmin
  - Add the newly created password to the config.inc file inside the phpMyAdmin folder

MySQL: Data Types

CHAR (<size>) (size < 255 bytes)
VARCHAR(<size>) (size < 255 bytes)
BLOB or **TEXT** (size < 65,535 bytes)
MEDIUMBLOB or **MEDIUMTEXT** (size < 16,777,215 bytes)
LONGBLOB or **LONGTEXT** (size < 4 GB)
ENUM (<value1>, <value2>,...<valueN>)
TINYINT, **SMALLINT**, **MEDIUMINT**, **INT**, **BIGINT**
are integers of 1, 2, 3, 4, and 8 bytes, respectively.
DECIMAL or **NUMERIC(M, D)**
FLOAT
DOUBLE
DATE (default format YYYY-MM-DD)

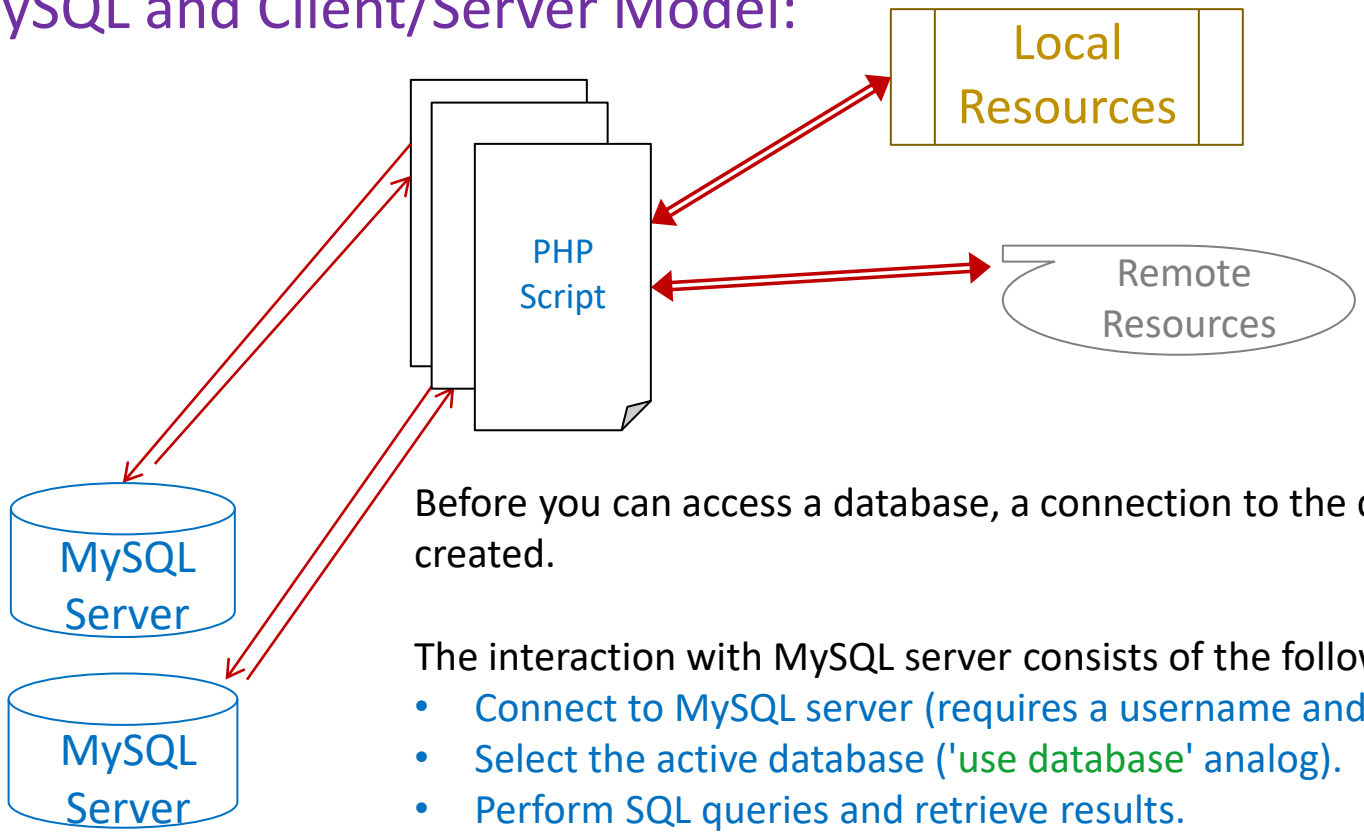
MySQL: Create / Import Tables

```
CREATE TABLE `students` (
  `studentID` int unsigned NOT NULL auto_increment,
  `name` varchar(100) NOT NULL default '',
  `major` varchar(50) NOT NULL default '',
  `grade` tinyint NOT NULL default '0',
  PRIMARY KEY  (`studentID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

StudentID	name	major	grade
1	Michael	CS	95
10146	Dennis	PHYS	75
10147	Boris	MATH	89
	…	…	…

Databases: Relationships

MySQL and Client/Server Model:



Before you can access a database, a connection to the database must be created.

The interaction with MySQL server consists of the following steps:

- Connect to MySQL server (requires a username and password).
- Select the active database ('use database' analog).
- Perform SQL queries and retrieve results.
- Free Results
- Close connection to MySQL

http://www.php.net/manual/en/set.mysqlinfo.php

Lab Details

LAB-06: Overview

Lab-06: PHP & SQL

- Build a basic PHP script
- Add variables and functions
- Working with databases and SQL
- SQL permissions