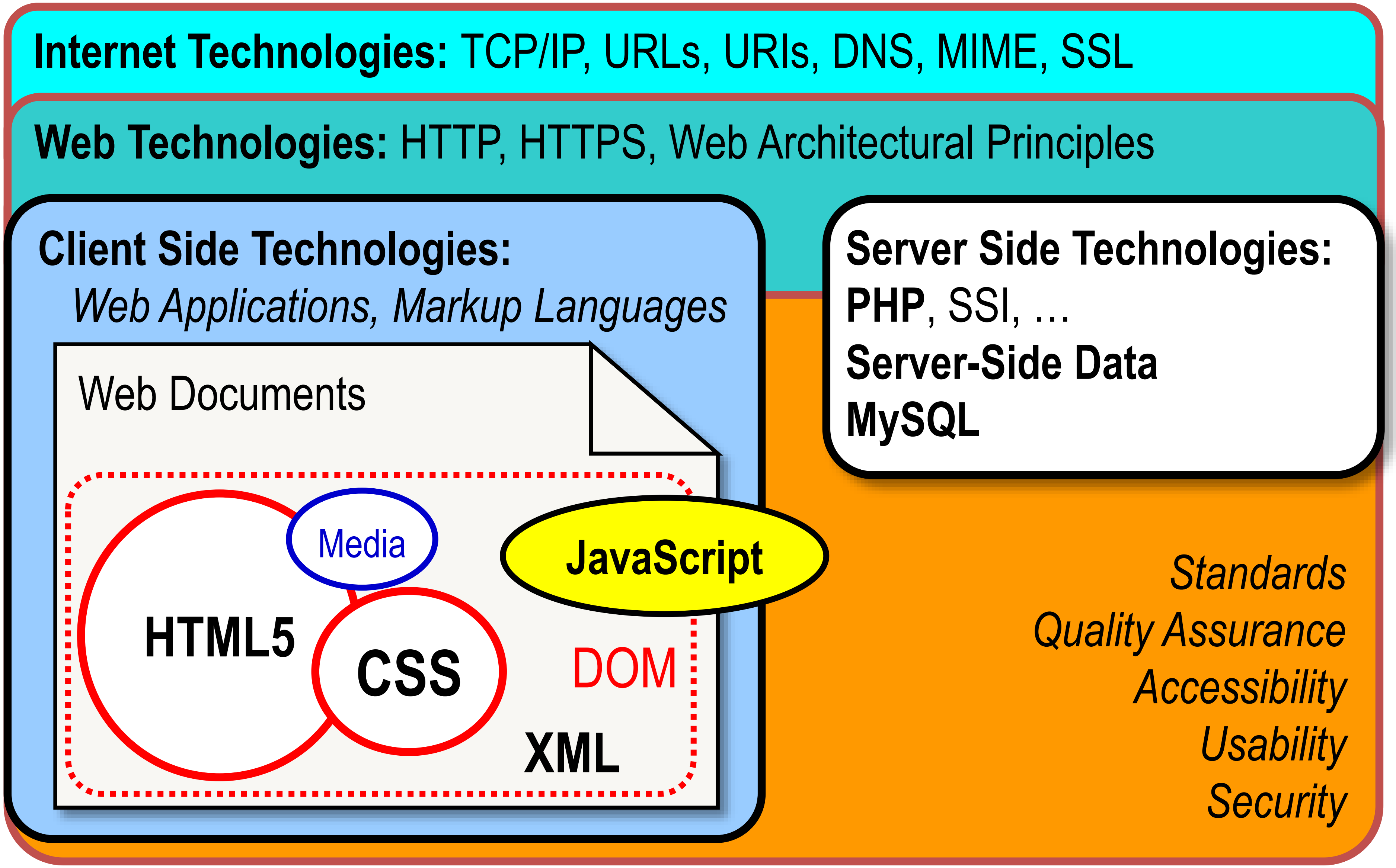
SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10011 Creating Web Applications

Lecture 10 – Server-side Data PHP and MySQL

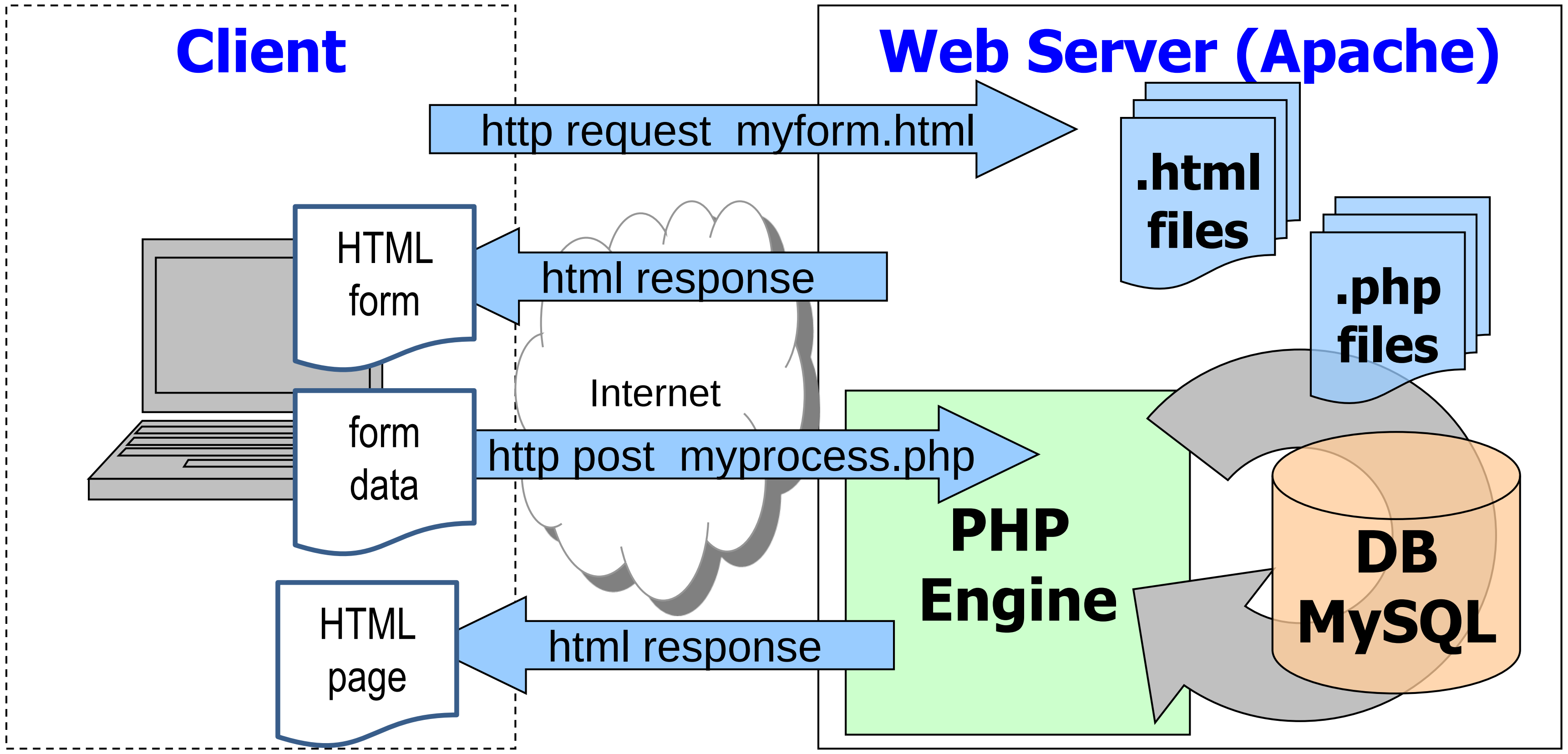
Acknowledgement: The contents in this document was adopted from learning material prepared by Alan Colman (Convener, SUT)

Unit of Study Outline


Internet Technologies: TCP/IP, URLs, URIs, DNS, MIME, SSL
Web Technologies: HTTP, HTTPS, Web Architectural Principles
Client Side Technologies:
Web Applications, Markup Languages
Web Documents
HTML5
Media
CSS
JavaScript
DOM
XML
Server Side Technologies:
PHP, SSI, …
Server-Side Data
MySQL
Standards
Quality Assurance
Accessibility
Usability
Security

Server-Side Scripting and PHP

Apache/PHP/MySQL example



Outline

- **Understanding the Basics of Databases**
- MySQL databases
- Accessing Databases with PHP
  - Creating and Deleting Databases and Tables
  - Selecting, Creating, Updating, and Deleting Records
  - Handling errors

Introduction to Databases

- A **database** is an ordered collection of information from which a computer program can quickly access information
- A **relational database** stores data in **tables**
- A **table** is a set of data expressed in terms of **records,** i.e. a row of a table
- A **record** is a single complete set of related information made up of **fields**
- A **field** is the individual category of information stored in a record

Introduction to Databases (continued)

Fields

Records

last_name	first_name	address	suburb	pcode	state
Coffey	Billy	648 Riversdale Road	Camberwell	3124	VIC
Clemons	Frank	Becks Road	Drysdale	3222	VIC
Dougherty	James	188 Holmes Road	Moonee Ponds	3039	VIC
Kirk	Jennifer	Kurnai Avenue	Reservoir	3073	VIC
Wilson	Jose	Coalmine Road	Anglesea	3230	VIC

employee information table

- A **relational database** stores information across ***multiple*** *related* tables

Understanding Relational Databases

(continued)

- A **primary key** is a field that contains a ***unique*** identifier for each record in a primary table. *It is a type of index that identifies records in a database and makes retrievals and sorting faster*
- A **foreign key** is a field in a related table that refers to the primary key in a primary table
- **Primary** and **foreign** keys link records across multiple tables in a relational database

One-to-One Relationships

- A **one-to-one** relationship exists between two tables when a related table contains exactly one record for each record in the primary table
- Information in the tables in a one-to-one relationship can be placed within a single table
- Creating a one-to-one relationship breaks information into multiple, logical sets
- The information in one of the tables can then be made confidential and accessible only to certain individuals

One-to-One Relationships (continued)

emp_id	last_name	first_name	address	suburb	pcode	state
101	Coffey	Billy	648 Riversdale Road	Camberwell	3124	VIC
102	Clemons	Frank	Becks Road	Drysdale	3222	VIC
103	Dougherty	James	188 Holmes Road	Moonee Ponds	3039	VIC
104	Kirk	Jennifer	Kurnai Avenue	Reservoir	3073	VIC
105	Wilson	Jose	Coalmine Road	Anglesea	3230	VIC

employee information table

primary key ←→ foreign key

emp_id	start_date	pay_rate	health_cover
101	2005	31.50	none
102	2003	29.00	individual
103	2009	33.00	family
104	2007	40.25	indivudal
105	2011	38.50	family

payroll rate table

One-to-one relationship

One-to-Many Relationship

- A **one-to-many** relationship exists in a relational database when one record in a primary table has many related records in a related table
- Breaking tables into multiple related tables to reduce redundant and duplicate information is called **normalization**
- *This provides a **more efficient**, less redundant, and **easier to maintain** method of storing data*

One-to-Many Relationship (continued)

emp_id	last_name	first_name	language	years
101	Coffey	Billy	Java	5
101	Coffey	Billy	C	7
102	Clemons	Frank	C#	8
102	Clemons	Frank	Objective C	2
102	Clemons	Frank	Java	3
103	Dougherty	James	C	2
103	Dougherty	James	C#	4
104	Kirk	Jennifer	Objective C	7
104	Kirk	Jennifer	Java	9
104	Kirk	Jennifer	C	4
105	Wilson	Jose	C#	6
105	Wilson	Jose	Objective C	3

Language Skills table with redundant information

One-to-Many Relationship (continued)

emp_id	last_name	first_name	address	suburb	pcode	state
101	Coffey	Billy	648 Riversdale Road	Camberwell	3124	VIC
102	Clemons	Frank	Becks Road	Drysdale	3222	VIC
103	Dougherty	James	188 Holmes Road	Moonee Ponds	3039	VIC
104	Kirk	Jennifer	Kurnai Avenue	Reservoir	3073	VIC
105	Wilson	Jose	Coalmine Road	Anglesea	3230	VIC

employee information table

primary key ←→ foreign key

emp_id	language	years
101	Java	5
101	C	7
102	C#	8
102	Objective C	2
102	Java	3
103	C	2
103	C#	4
104	Objective C	7
104	Java	9
104	C	4
105	C#	6
105	Objective C	3

language skills table

One-to-many relationship

Many-to-Many Relationship

- A **many-to-many relationship** exists in a relational database when many records in one table are related to many records in another table e.g. relationship between programmers and languages
- Must use a **junction** or **associative table** that creates a one-to-many relationship for each of the two tables in a many-to-many relationship. It contains *foreign keys* from the two tables

Many-to-Many Relationship (continued)

emp_id	last_name	first_name	address	suburb	pcode	state
101	Coffey	Billy	648 Riversdale Road	Camberwell	3124	VIC
102	Clemons	Frank	Becks Road	Drysdale	3222	VIC
103	Dougherty	James	188 Holmes Road	Moonee Ponds	3039	VIC
104	Kirk	Jennifer	Kurnai Avenue	Reservoir	3073	VIC
105	Wilson	Jose	Coalmine Road	Anglesea	3230	VIC

employee information table

primary key ←→ foreign key

emp_id	language	years
101	11	5
101	12	7
102	13	8
102	14	2
102	11	3
103	12	2
103	13	4
104	14	7
104	11	9
104	12	4
105	13	6
105	14	3

language skills table (junction)

lang_id	language
11	Java
12	C
13	C#
14	Objective C

language information table

foreign key ←→ primary key

Many-to-many relationship

Working with Database Management Systems

- A **database management system** (or DBMS) is an application or collection of applications used to access and manage a database
- A **schema** is the structure of a database including its tables, fields, and relationships
- A **relational database management system** (or RDBMS) stores data in a relational format

Functions of a DBMS

- The structuring and preservation of the database file
- Ensuring that data is stored correctly in a database’s tables, regardless of the database format
- Querying capability
- Security

Querying Databases

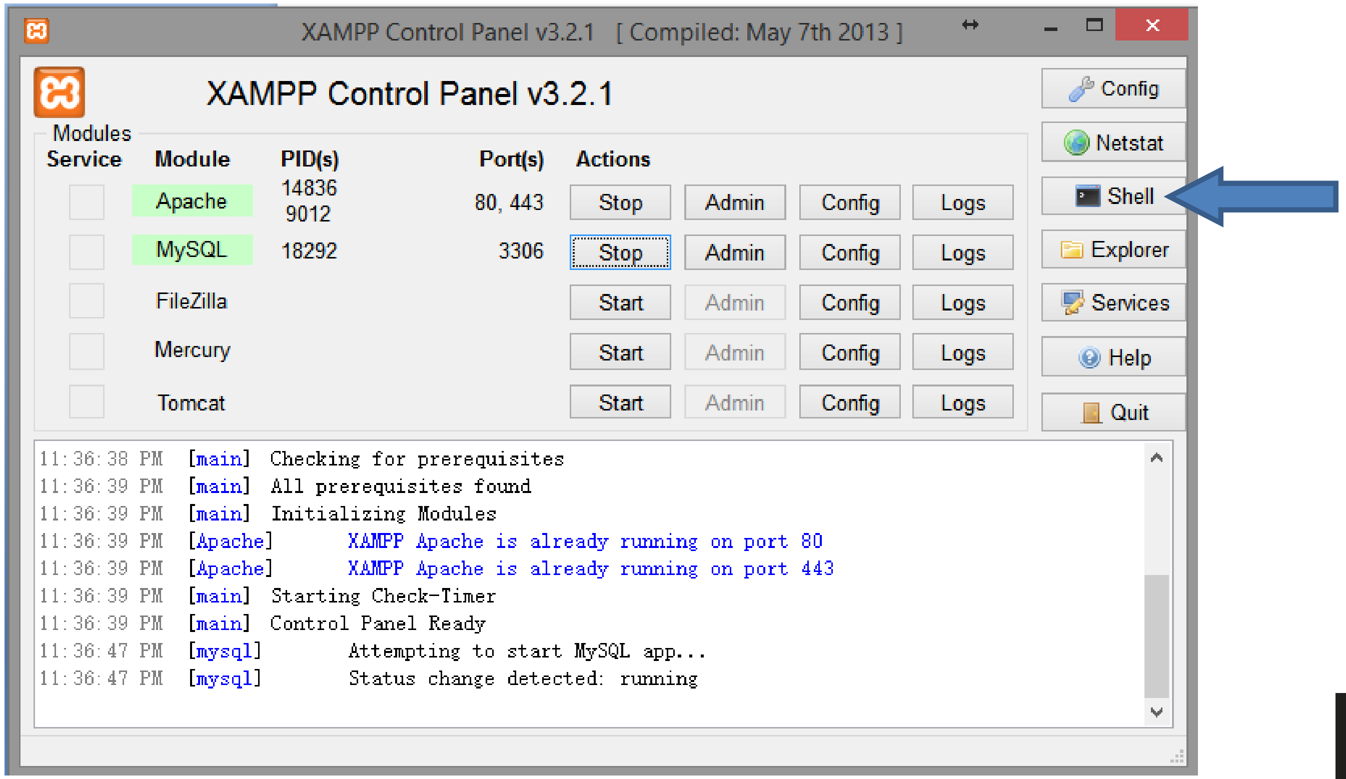
- A **query** is a structured set of instructions and criteria for retrieving, adding, modifying, and deleting database information
- **Structured query language** (or SQL – often pronounced as sequel) is a standard data manipulation language used by most database management systems

Outline

- Understanding the Basics of Databases
- **MySQL databases**
  - Working with MySQL Databases
  - Managing Databases and their Tables
  - Managing Tables and their Records
- Accessing Databases with PHP
  - Creating and Deleting Databases and Tables
  - Selecting, Creating, Updating, and Deleting Records
  - Handling errors

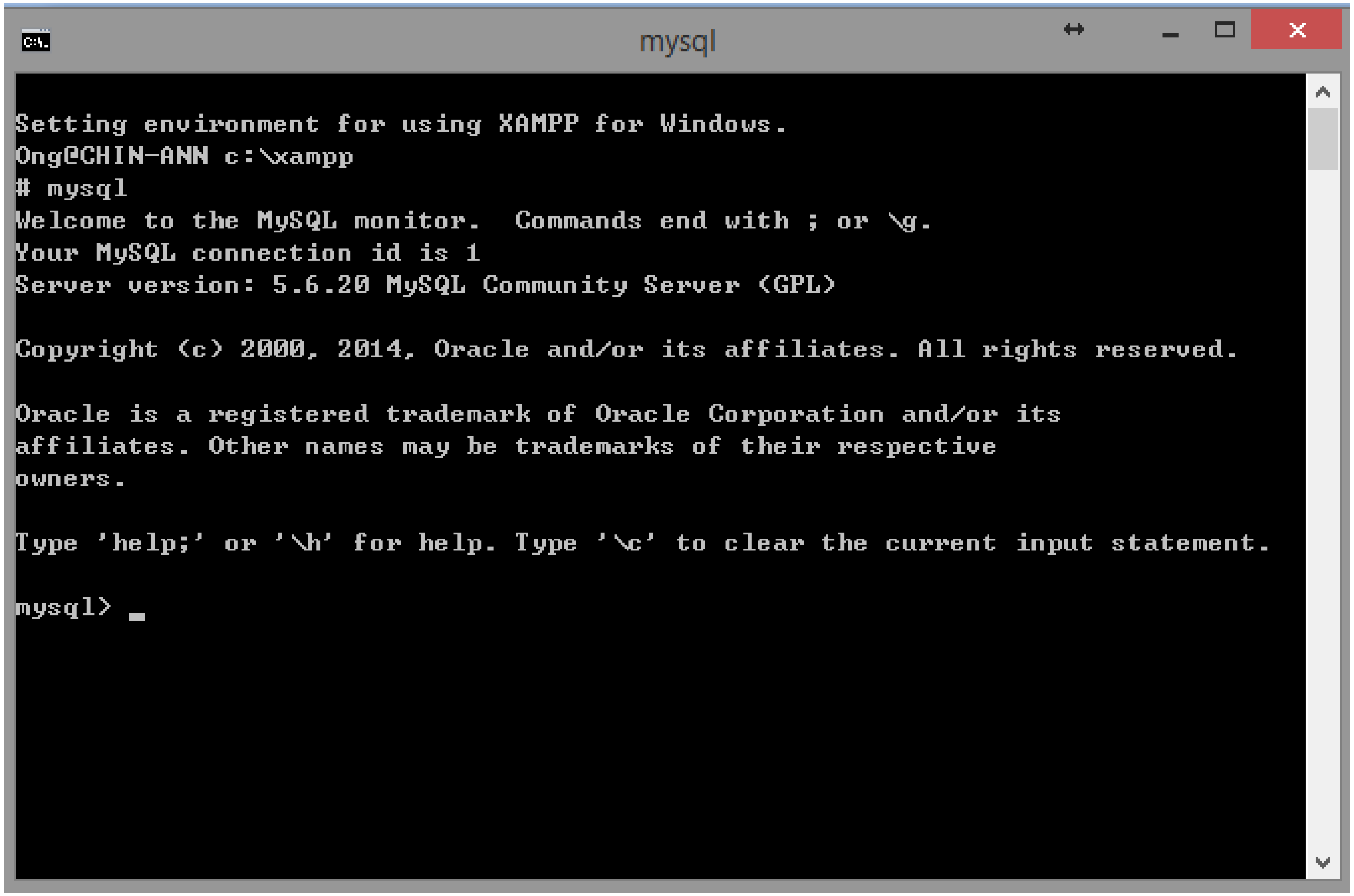
Startup MySQL Monitor

- We uses XAMPP
- Comes with MySQL
- Run “shell” to execute SQL Statements.



```
11:36:38 PM  [main]    Checking for prerequisites
11:36:39 PM  [main]    All prerequisites found
11:36:39 PM  [main]    Initializing Modules
11:36:39 PM  [Apache]      XAMPP Apache is already running on port 80
11:36:39 PM  [Apache]      XAMPP Apache is already running on port 443
11:36:39 PM  [main]    Starting Check-Timer
11:36:39 PM  [main]    Control Panel Ready
11:36:47 PM  [mysql]       Attempting to start MySQL app...
11:36:47 PM  [mysql]       Status change detected: running
```

Startup MySQL Monitor



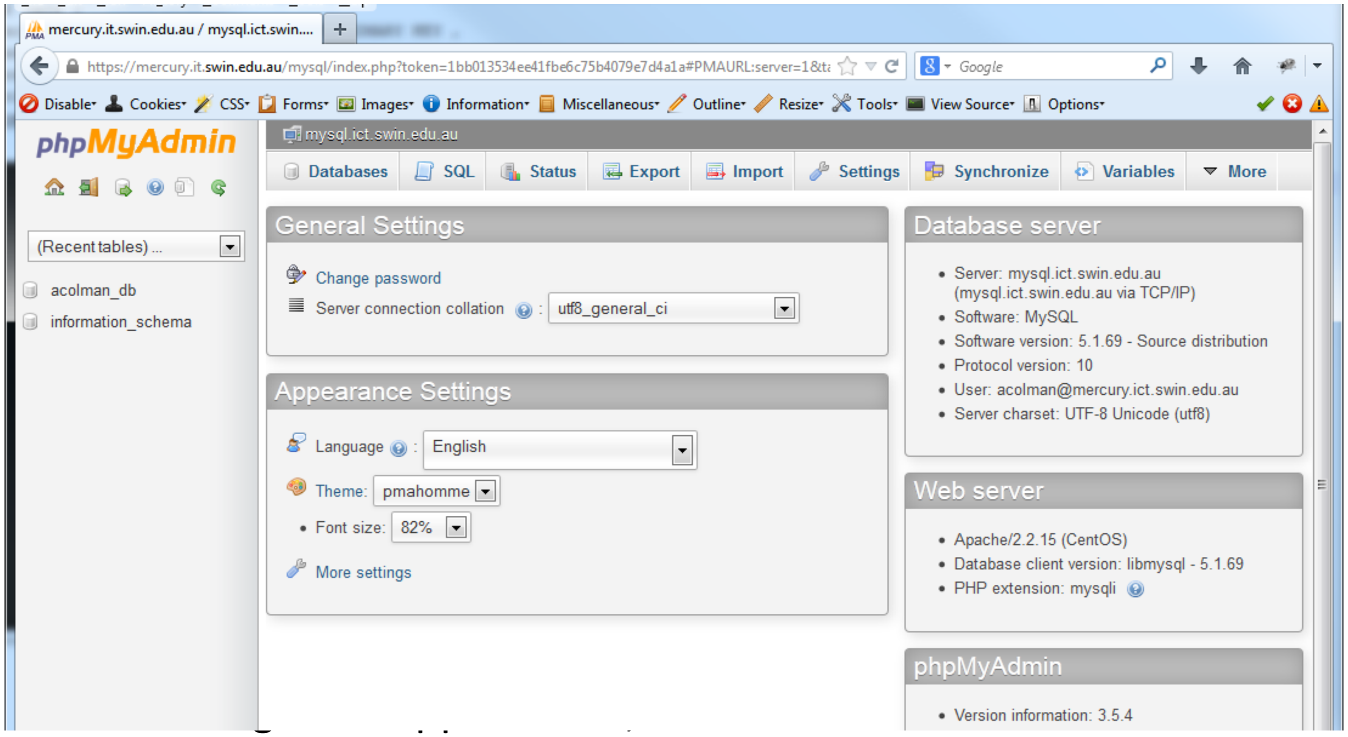
mysql

```
Setting environment for using XAMPP for Windows.
Ong@CHIN-ANN c:\xampp
# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.20 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

Using phpMyAdmin

- Web UI to mySQL
- Log in to phpMyAdmin with your mysql username and mysql password

http://localhost/phpmyadmin/


mercury.it.swin.edu.au / mysql.ict.swin....
https://mercury.it.swin.edu.au/mysql/index.php?token=1bb013534ee41fbe6c75b4079e7d4a1a#PMAURL:server=1&t
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
phpMyAdmin
(Recent tables) ...
acolman_db
information_schema
mysql.ict.swin.edu.au
Databases SQL Status Export Import Settings Synchronize Variables More
General Settings
Change password
Appearance Settings
More settings

Server connection collation: utf8_general_ci

Language: English
Theme: pmahomme
- Font size: 82%

Database server

- Server: mysql.ict.swin.edu.au (mysql.ict.swin.edu.au via TCP/IP)
- Software: MySQL
- Software version: 5.1.69 - Source distribution
- Protocol version: 10
- User: acolman@mercury.ict.swin.edu.au
- Server charset: UTF-8 Unicode (utf8)

Web server

- Apache/2.2.15 (CentOS)
- Database client version: libmysql - 5.1.69
- PHP extension: mysqli

phpMyAdmin

- Version information: 3.5.4

Outline

- Understanding the Basics of Databases
- **MySQL databases**
  - Working with MySQL Databases
  - Managing Databases and their Tables
  - Managing Tables and their Records
- Accessing Databases with PHP
  - Creating and Deleting Databases and Tables
  - Selecting, Creating, Updating, and Deleting Records
  - Handling errors

Selecting Databases

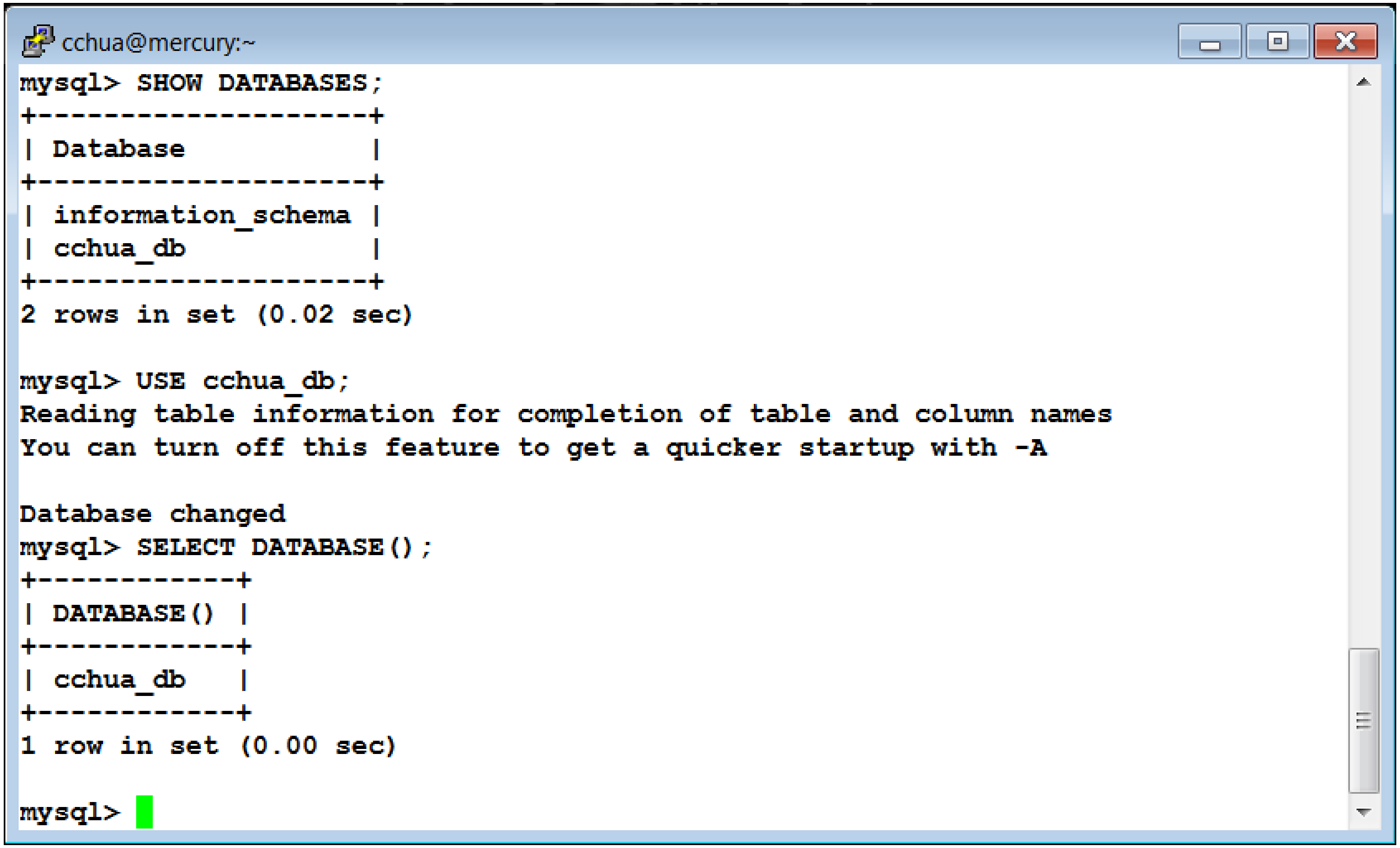
- Use `SHOW DATABASES` statement to view the databases that are available
- Use `USE DATABASE` statement to select the database to work with
- Use `SELECT DATABASE()` statement to display the name of the currently selected database

SQL Command Basics

The four important basic SQL commands for managing databases and tables:

- USE: select a database to use
- CREATE: add a new **database** or
  add **table** to the existing database
- DROP: delete a **database** or
  delete **table** from database

Selecting Databases (continued)



```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cchua_db           |
+--------------------+
2 rows in set (0.02 sec)

mysql> USE cchua_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| cchua_db   |
+------------+
1 row in set (0.00 sec)

mysql> 
```

MySQL Monitor after selecting a database

Outline

- Understanding the Basics of Databases
- **MySQL databases**
  - Working with MySQL Databases
  - Managing Databases and their Tables
  - Managing Tables and their Records
- Accessing Databases with PHP
  - Creating and Deleting Databases and Tables
  - Selecting, Creating, Updating, and Deleting Records
  - Handling errors

SQL Command Basics

The four important basic SQL commands for managing records:

– SELECT:	**ask** for data
– INSERT:	**add** new data
– UPDATE:	**modify** existing data
– DELETE:	**remove** existing data

SQL queries using MySQL Monitor

- At the `mysql>` command prompt terminate the command with a semicolon

```
mysql> SELECT * FROM car;
```

- Without a semicolon, the MySQL Monitor enters a multiple-line command and changes the prompt to ->

```
mysql> SELECT * FROM car
           -> WHERE make = "Holden";
```

- Note that the SQL **keywords** entered in the MySQL Monitor are **not** case sensitive

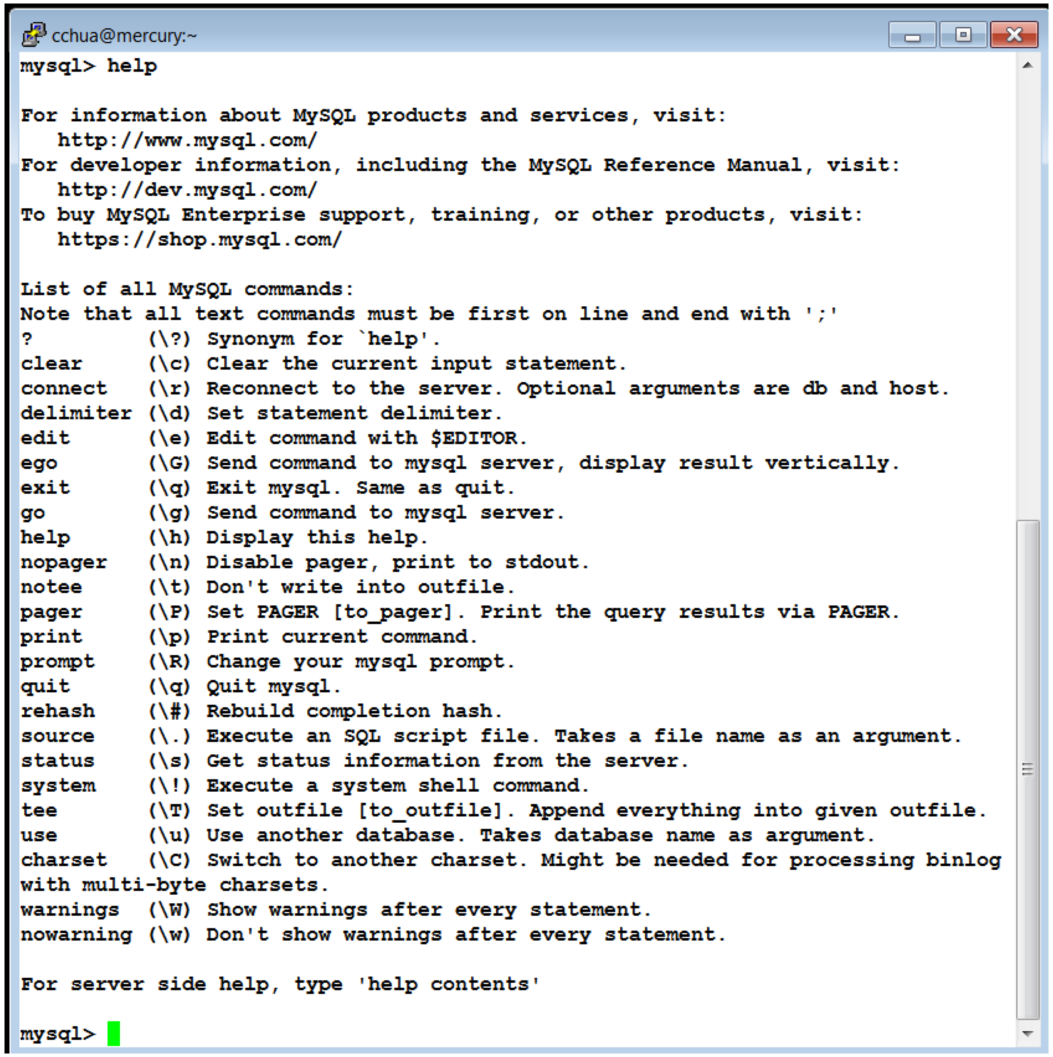
Understanding MySQL Identifiers

Identifiers for databases, tables, fields, indexes, and aliases

- The **case sensitivity** of database and table **identifiers** depends on the operating system
  - Not case sensitive on Windows platforms
  - Case sensitive on UNIX/Linux systems
- MySQL stores each database in a directory of the same name as the database identifier
- Field and index identifiers are case insensitive on all platforms *… but try and be consistent* ☺

Getting Help with MySQL Commands

```
mysql> help;
```



```
mysql> help

For information about MySQL products and services, visit:
   http://www.mysql.com/
For developer information, including the MySQL Reference Manual, visit:
   http://dev.mysql.com/
To buy MySQL Enterprise support, training, or other products, visit:
   https://shop.mysql.com/

List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear the current input statement.
connect   (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
edit      (\e) Edit command with $EDITOR.
ego       (\G) Send command to mysql server, display result vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
nopager   (\n) Disable pager, print to stdout.
notee     (\t) Don't write into outfile.
pager     (\P) Set PAGER [to_pager]. Print the query results via PAGER.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an argument.
status    (\s) Get status information from the server.
system    (\!) Execute a system shell command.
tee       (\T) Set outfile [to_outfile]. Append everything into given outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for processing binlog
with multi-byte charsets.
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'

mysql>
```

MySQL command help

Outline

Understanding the Basics of Databases

- Working with MySQL Databases
- Managing Databases and their Tables
- Managing Tables and their Records

Accessing Databases with PHP

- Creating and Deleting Databases and Tables
- Selecting, Creating, Updating, and Deleting Records
- Handling errors

Accessing Databases with PHP

- There are three main options when considering connecting to a MySQL database server using PHP:
  - PHP's mysql Extension
  - PHP's mysqli Extension
  - PHP Data Objects (PDO)

We will use mysqli

- The mysqli extension features a dual interface, supporting both procedural (functions) and object-oriented interfaces.
- These notes and examples use the procedural interface.

http://www.php.net/manual/en/book.mysqli.php

Hint: Separate file for your login info

Example

```
<?php
	$host = “localhost";
	$user = “root";
	$pwd  = “";
	$sql_db  = " s1234567_db";
?>
```

Can edit the host when goes to production server

Default setting, can change later

By default, there is no password. But good set a password in production site

Your database name

Template 1 – for SQL* queries

* Create and drop tables

* Insert update and delete records

```
<?php
    require_once "settings.php";
    $conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
    if ($conn) {

        $query = "replace with a valid SQL query";
        $result = mysqli_query ($conn, $query);
        if ($result) { …}
        else {…}

        mysqli_close ($conn);
    } else      echo "<p>Unable to connect to the db.</p>";
?>
```

Step 1: Connect to the database

Specify the credentials in setting.php

Step 2: Create your SQL query

Step 3: Execute your SQL query

Step 4: Did it work?

Step 5: Close connection

Connecting to MySQL

- Open a connection to a MySQL database server with the `mysqli_connect()` function
- The `mysqli_connect()` function returns a ***positive integer*** if it connects to the database successfully or `false` if it does not
- Assign the return value from the `mysqli_connect()` function to a variable that you can use to access the database in your script
- Example

```
$yourconn= mysqli_connect("localhost",
   "root", "<yourMySQLpassword>", "<db name>");
```

Connecting to MySQL (continued)

- The syntax for the `mysqli_connect()` function is:

  ```
  $connection = mysqli_connect("host"[,
  "user", "password", "database"])
  ```

  - The ***host*** argument specifies the host name where your MySQL database server is installed
    e.g. `mysql.ict.swin.edu.au / localhost`
  - The ***user*** and ***password*** arguments specify a MySQL account name and password
    e.g. `s1234567 yourMySQLpassword`
  - The ***database*** argument specifies a database
    e.g. `s1234567_db`

Selecting a Database

We can connect() and select_db() in separate steps

- The statement for selecting a database with the MySQL Monitor is **`use`** ***`database`***
- The function for selecting a database with PHP is **`mysqli_select_db(`*`connection, database`*`)`**
- The function returns a value of **`true`** if it successfully selects a database or **`false`** if it does not

Executing SQL Statements

The **`mysqli_query()`** function returns one of three values:

- For SQL statements that *do not* return results (**`CREATE DATABASE`** and **`CREATE TABLE`** statements) they return a value of `true` if the statement executes successfully
- For SQL statements that *do* return results (**`SELECT`** and **`SHOW`** statements) they return a ***result pointer*** that represents the query results
  - A **result pointer** is a special type of variable that refers to the currently selected row in a resultset
- For SQL statements that fail, **`mysqli_query()`** function returns a value of `false`, regardless of whether they return results

Cleaning Up

- When you are finished working with query results retrieved with the `mysqli_query()` function, use the `mysqli_free_result()` function to close the resultset
- To close the resultset, pass to the `mysqli_free_result()` function the variable containing the result pointer from the `mysqli_query()` function

  e.g. `mysqli_free_result(`**`$queryResult`**`);`

Closing Connection

- Close a connection to a MySQL database server with the `mysqli_close()` function
  - `mysqli_close($dbconnect);`

Outline

Understanding the Basics of Databases

- Working with MySQL Databases
- Managing Databases and their Tables
- Managing Tables and their Records

Accessing Databases with PHP

- **Creating and Deleting Databases and Tables**
- Selecting, Creating, Updating, and Deleting Records
- Handling errors

Creating Tables

- The `CREATE TABLE` statement specifies the table and column names and the data type for each column
- The syntax for the `CREATE TABLE` statement is:

```
CREATE TABLE table_name
    (column_name TYPE, ...);
```

- Execute the `USE` statement to select a database before executing the `CREATE TABLE` statement

Creating and Deleting Tables (continued)

```
...
$sqlString = "CREATE TABLE car(
   model        VARCHAR(30),
   make         VARCHAR(25),
   price        INT,
   manufactured    DATE)";

$queryResult = @mysqli_query($dbConnect, $sqlString)
...
```

Use INT if you do not want to store any decimal figures

What does the "@" for?
See later

Creating Tables (continued)

Type	Range	Storage
BOOL	-128 to 127 with 0 considered false	1 byte
INT or INTEGER	-2147483648 to -2147483647	4 bytes
FLOAT	-3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E+38 to 3.402823466E+38	8 bytes
DOUBLE	-1.7976931348623157E+308 to -2.2250738585072014E+308, 0, and 2.2250738585072014E+308 to 1.7976931348623157E+308	8 bytes
DATE	'1000-01-01' to '9999-12-31'	Varies
TIME	'-838:59:59' to '838:59:59'	Varies
CHAR(n)	Fixed length string between 0 to 255 characters	Number of bytes specified by n
VARCHAR(n)	Variable length string between 0 to 65,535 characters	Varies according to the number of bytes specified by n

Common MySQL field data types

Deleting Tables

- The `DROP TABLE` statement removes all data and the table definition
- The syntax for the `DROP TABLE` statement is:

```
DROP TABLE table_name;
```

Outline

Understanding the Basics of Databases

- Working with MySQL Databases
- Managing Databases and their Tables
- Managing Tables and their Records

Accessing Databases with PHP

- Creating and Deleting Databases and Tables
- **Selecting, Creating, Updating, and Deleting Records**
- Handling errors

Structured Query Language (SQL)

Common SQL keywords

Keyword	Description
INSERT	Inserts a new row into a table
UPDATE	Update field value in a record
DELETE	Deletes a row from the table
SELECT	Retrieve records from table(s)
INTO	Specifies the table into which to insert the record(s)
FROM	Specifies the table(s) from which to retrieve or delete record(s)
WHERE	Specifies the condition that must be met
ORDER BY	Sorts the records retrieved (does not affect the table)

e.g. `SELECT * FROM employees`

See also:
http://swinbrain.ict.swin.edu.au/wiki/SQL_Commands_Introduction

Adding Records

- Use the `INSERT` statement to add individual records to a table
- The syntax for the `INSERT` statement is:

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

- The values entered in the `VALUES` list must be in the same order in which you defined the table fields
- Specify `NULL` in any fields for which you do not have a value
- Add multiple records, use the `LOAD DATA` statement

```
LOAD DATA LOCAL INFILE 'file_path_name' INTO TABLE
table_name;
```

Adding Records with INSERT

- Use the **INSERT** and **VALUES** keywords with the **mysqli_query()** function

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

- The values entered in the **VALUES** list must be in the same order that defined in the table fields
- Specify **NULL** in any fields that do not have a value e.g. for **AUTO_INCREMENT** field

Adding record with INSERT: PHP example

```
<?php
    require_once "settings.php";
    $conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
    if ($conn) {
        $query = "INSERT INTO
                `tutors`  (`userid`, `username`,`password`, `datejoined`)
                        VALUES (1,'Alex','8376',curdate())";;
        $result = mysqli_query ($conn, $query);
        if ($result)  { echo "<p>Insert operation successful.</p>";}
        else { echo "<p>Insert operation unsuccessful.</p>";  }
        mysqli_close ($conn);
    } else echo "<p>Unable to connect to the db.</p>";
?>
```

Field names and values must be in the same order

Table name

Updating Records

- To update records in a table, use the `UPDATE` statement
- The syntax for the `UPDATE` statement is:

```
UPDATE table_name
SET column_name=value
WHERE condition;
```

  - The `UPDATE` keyword specifies the name of the table to update
  - The `SET` keyword specifies the value to assign to the fields in the records that match the condition in the `WHERE` keyword

UPDATE record in PHP example

```
<?php
    require_once "settings.php";
    $conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
    if ($conn) {
        $query = "UPDATE `tutors`
                        SET `password`='1234'
                        WHERE userid = 1";
        $result = mysqli_query ($conn, $query);
        if ($result) {echo "<p>Update operation successful.</p>";}
        else { echo "<p>Update operation unsuccessful.</p>"; }
        mysqli_close ($conn);
    } else echo "<p>Unable to connect to the db.</p>";
?>
```

Deleting Records

- Use the `DELETE` statement to delete records in a table
- The syntax for the `DELETE` statement is:

```
DELETE FROM table_name
WHERE condition;
```

- The `DELETE` statement deletes all records that match the condition
- To delete all the records in a table, leave off the `WHERE` keyword

Delete record in PHP example

```
<?php
    require_once "settings.php";
    $conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
    if ($conn) {
        $query = "DELETE FROM `tutors` WHERE userid = 1";
        $result = mysqli_query ($conn, $query);
        if ($result)  { echo "<p>Deleted"
                    . mysqli_affected_rows($dbConnect) . " record(s).</p>";
        }else { echo "<p>Insert operation unsuccessful.</p>";           }
        mysqli_close ($conn);
    } else echo "<p>Unable to connect to the db.</p>";
?>
```

Deleting Records

To Delete records from a table:

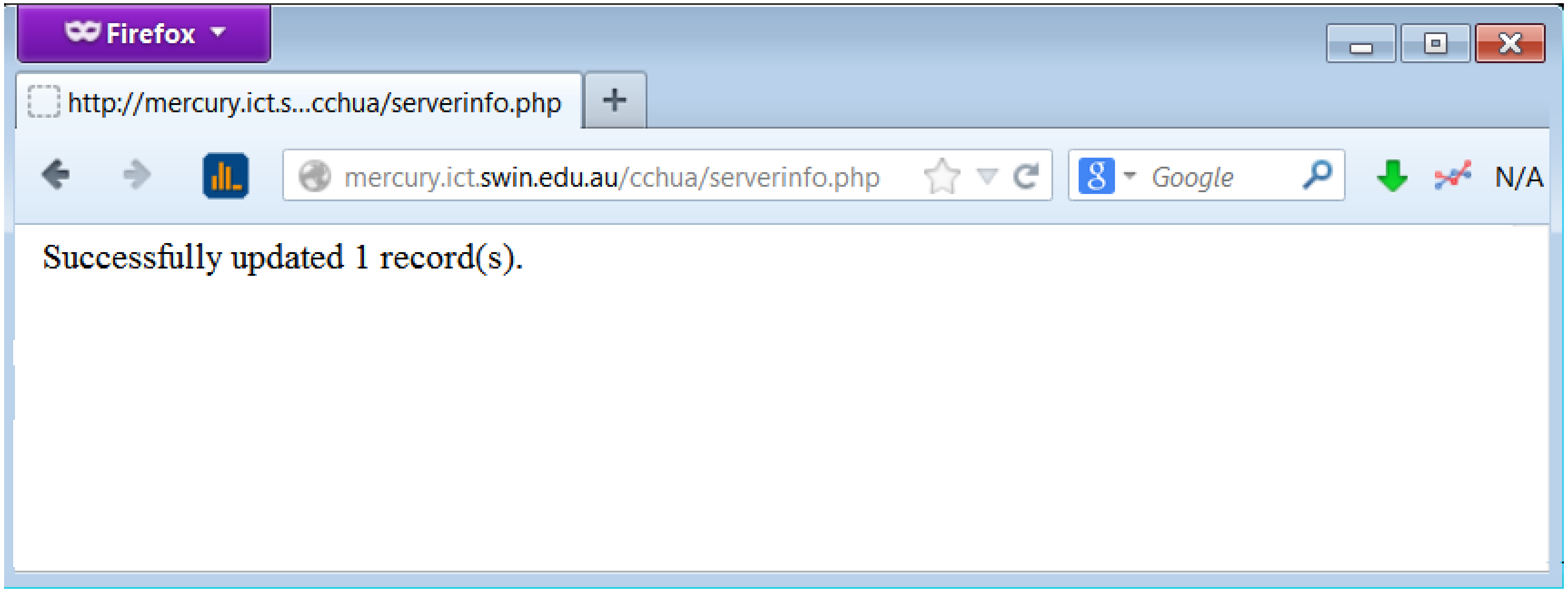
- Use the **`DELETE`** and **`WHERE`** keywords with the **`mysqli_query()`** function
- The **`WHERE`** keyword determines which records to delete in the table
- *Be careful*, if no **`WHERE`** keyword, *all records are deleted !!*

Using the **`mysqli_affected_rows()`** Function

- With queries that modify tables but do not return results (**INSERT**, **UPDATE**, and **DELETE** queries), use the **`mysqli_affected_rows()`** function to determine the *number of affected rows* by the query

```
$sqlString = "UPDATE car SET price=4500
     WHERE make='Fender' AND model='DG7'";
$queryResult = @mysqli_query($dbConnect, $sqlString);
if ($queryResult){
   echo "<p>Successfully updated "
   . mysqli_affected_rows($dbConnect) . "record(s).</p>";
}
```

Using the `mysqli_affected_rows()` Function


Firefox
http://mercury.ict.s...cchua/serverinfo.php
mercury.ict.swin.edu.au/cchua/serverinfo.php
Google
N/A

```
Successfully updated 1 record(s).
```

Output of `mysqli_affected_rows($con)` function for an `UPDATE` query

Selecting and Retrieving Records

- Use the `SELECT` statement to retrieve records from a table:

```
SELECT criteria FROM table_name;
```

- Use the asterisk (*) wildcard with the `SELECT` statement to retrieve all fields from a table
- To return multiple fields, separate field names with a comma

```
mysql> SELECT model, quantity FROM inventory;
```

Retrieving Records – Sorting

- Use the `ORDER BY` keyword with the `SELECT` statement to perform an alphanumeric sort of the results returned from a query

```
mysql> SELECT make, model FROM inventory
     -> ORDER BY make, model;
```

- To perform a reverse sort, add the `DESC` keyword after the name of the field by which you want to perform the sort

```
mysql> SELECT make, model FROM inventory
     -> ORDER BY make DESC, model;
```

Retrieving Records – Filter

- The **criteria** portion of the `SELECT` statement determines which fields to retrieve from a table
- You can also specify which records to return by using the `WHERE` keyword

```
mysql> SELECT * FROM inventory
     -> WHERE make='Martin';
```

- Use the keywords `AND` and `OR` to specify more detailed conditions about the records you want to return

```
mysql> SELECT * FROM inventory
     -> WHERE make='Washburn' AND price<400;
```

Selecting Records in PHP

To select from a table:

- Use the **`SELECT`** and **`WHERE`** keywords with the `mysqli_query()` function
- The `WHERE` keyword determines which records to select in the table
- if no `WHERE` keyword, all records are selected

Selecting Records (continued)

Be careful when constructing query:

```
$make = "Holden";

$sqlString = "SELECT model, quantity FROM
    $dbTable WHERE model = '$make'";
```

Field name
not in ‘quotes’

Variable name
must be in
‘quotes’ if string

Template 2 – for SQL SELECT queries

```
<?php
    require_once "settings.php";
    $conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
    if ($conn) {
        $query = "replace with a MySQL SELECT query";
        $result = mysqli_query ($conn, $query);
        if ($result) {
        $record = mysqli_fetch_assoc ($result);
            if ($record) {
                echo "<p>At least 1 record was retrieved.</p>";
            } else echo "<p>No records retrieved.</p>";
        } else      echo "<p>MySQL operation unsuccessful.</p>";
        mysqli_close ($conn);
    } else  echo "<p>Unable to connect to the db.</p>";
?>
```

Checks if query successful

Check if any records exist

Note: we haven't done anything with the records yet

Selecting Records (continued)

Function	Description
mysqli_data_seek($result, position)	Moves the result pointer to a specific row in the result set
mysqli_fetch_array($result, mysqli_assoc | mysqli_num | mysqli_both)	Returns the fields in the current row of the result set into an associative array, indexed array or both, and moves the result pointer to the next row
mysqli_fetch_assoc($result)	Returns the fields in the current row of the result set into an associative array, and moves the result pointer to the next row
mysqli_fetch_row($result)	Returns the fields in the current row of the result set into an indexed array, and moves the result pointer to the next row
mysqli_fetch_lengths($result)	Returns the field lengths for the current row in a result set into an indexed array

Common PHP functions for accessing database results

Selecting Records (continued)

- The difference between **`mysqli_fetch_assoc()`** and **`mysqli_fetch_row()`** is that instead of returning the fields into an *indexed array*, **`mysqli_fetch_assoc()`** function returns the fields into an *associate array* and uses each *field name* as the *array key*

Selecting Records (continued)

Retrieving Records into an Associative Array

- The `mysqli_fetch_assoc()` function returns the fields in the current row of a result set into an associative array and moves the result pointer to the next row

```
echo "<table border='1'>";
echo "<tr><th>Make</th><th>Model</th>
  <th>Price</th><th>Yr of Manufacture</th></tr>";
$row = mysqli_fetch_assoc($queryResult);
while ($row) {
       echo "<tr><td>{$row['make']}</td>";
       echo "<td>{$row['model']}</td>";
       echo "<td>{$row['price']}</td>";
       echo "<td>{$row['yom']}</td></tr>";
       $row = mysqli_fetch_assoc($queryResult);
}
echo "</table>";
```

Selecting Records (continued)

Retrieving Records into an Indexed Array

- The **`mysqli_fetch_row()`** function returns the fields in the current row of a result set into an indexed array and moves the result pointer to the next row

```
echo "<table border='1'>";
echo "<tr><th>Make</th><th>Model</th>
   <th>Price</th><th>Yr of Manufacture</th></tr>";
$row = mysqli_fetch_row($queryResult);
while ($row) {
       echo "<tr><td>{$row[0]}</td>";
       echo "<td>{$row[1]}</td>";
       echo "<td>{$row[2]}</td>";
       echo "<td>{$row[3]}</td></tr>";
       $row = mysqli_fetch_row($queryResult);
}
echo "</table>";
```

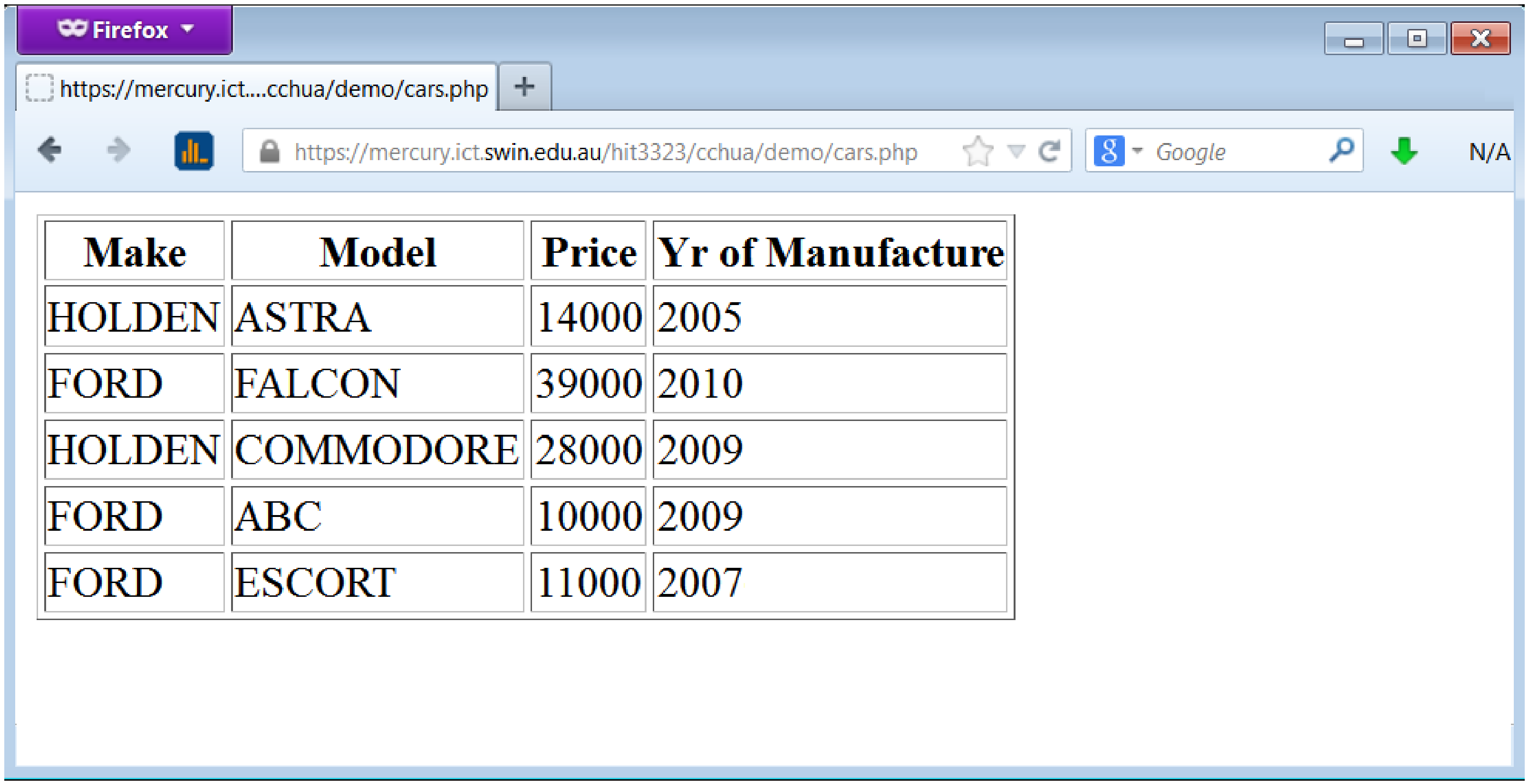
Selecting Records (continued)

- Assignment and comparison can also be combined to reduce the size of the code

```
echo "<table border='1'>";
echo "<tr><th>Make</th><th>Model</th>
  <th>Price</th><th>Yr of Manufacture</th></tr>";

while ($row = mysqli_fetch_assoc($queryResult)) {
      echo "<tr><td>{$row['make']}</td>";
      echo "<td>{$row['model']}</td>";
      echo "<td>{$row['price']}</td>";
      echo "<td>{$row['yom']}</td></tr>";

}
echo "</table>";
```

This is an assignment expression, not a comparison

Selecting Records (continued)


Firefox
https://mercury.ict....cchua/demo/cars.php
https://mercury.ict.swin.edu.au/hit3323/cchua/demo/cars.php
Google
N/A

Make	Model	Price	Yr of Manufacture
HOLDEN	ASTRA	14000	2005
FORD	FALCON	39000	2010
HOLDEN	COMMODORE	28000	2009
FORD	ABC	10000	2009
FORD	ESCORT	11000	2007

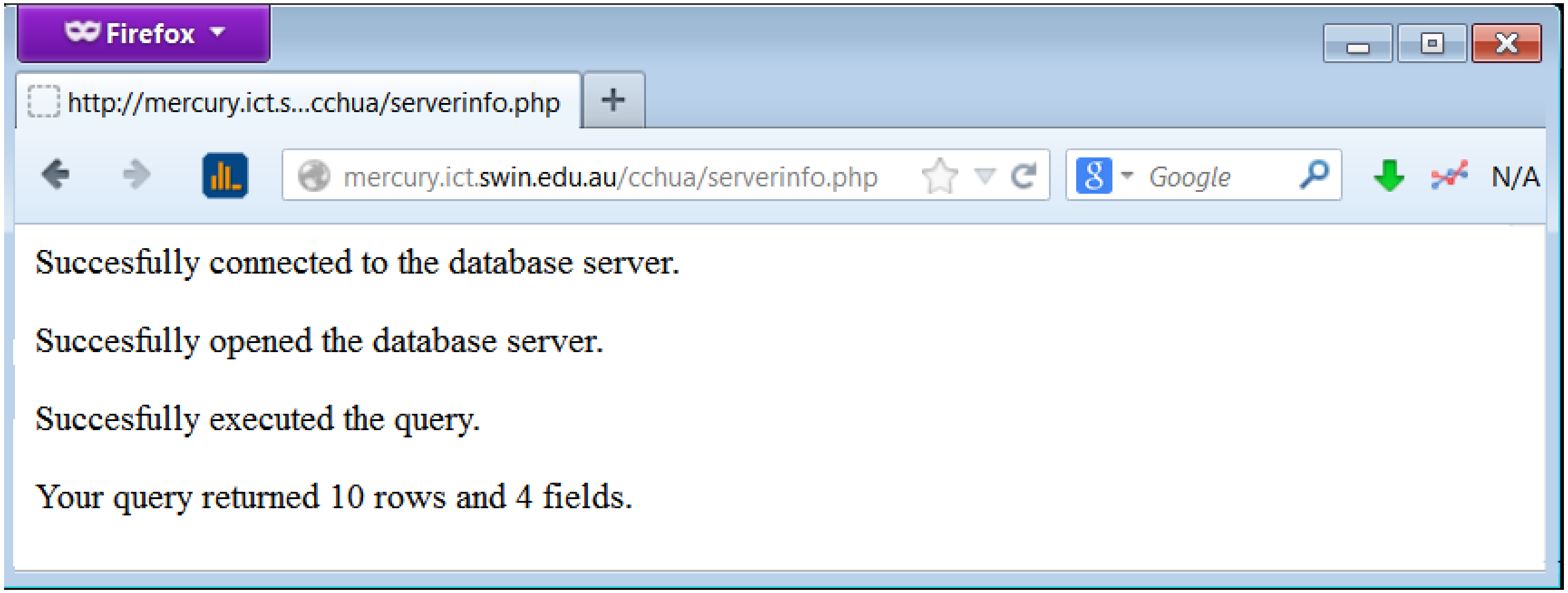
Output of the inventory table in a Web browser

Selecting Records (continued)

Accessing Query Result Information for queries that return result sets:

- The `mysqli_num_rows()` function returns the number of rows in a query result
- The `mysqli_num_fields()` function returns the number of fields in a query result
- Both functions accept a database result variable,
eg.a query result, as an argument

Selecting Records (continued)


Firefox
http://mercury.ict.s...cchua/serverinfo.php
mercury.ict.swin.edu.au/cchua/serverinfo.php
Google
N/A

```
Succesfully connected to the database server.

Succesfully opened the database server.

Succesfully executed the query.

Your query returned 10 rows and 4 fields.
```

Output of the number of rows and fields returned from a query

Outline

Understanding the Basics of Databases

- Working with MySQL Databases
- Managing Databases and their Tables
- Managing Tables and their Records

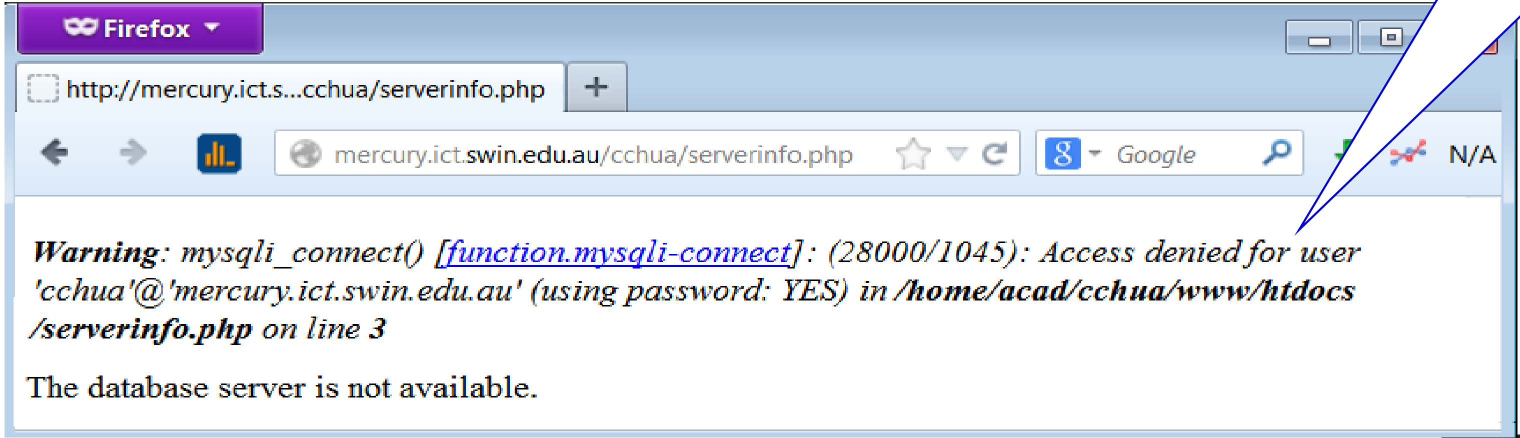
Accessing Databases with PHP

- Creating and Deleting Databases and Tables
- Selecting, Creating, Updating, and Deleting Records
- ❯ **Handling errors**

Handling MySQL Errors

- Reasons for not connecting to a database server include:
  - The database server is not running
  - Insufficient privileges to access the data
  - Invalid username and/or password
- e.g. `if (!$dbConnect) ...`

We do not want users to see any database error messages !


Firefox
http://mercury.ict.s...cchua/serverinfo.php
mercury.ict.swin.edu.au/cchua/serverinfo.php
Google
N/A

```
Warning: mysqli_connect() [function.mysqli-connect]: (28000/1045): Access denied for user
'cchua'@'mercury.ict.swin.edu.au' (using password: YES) in /home/acad/cchua/www/htdocs
/serverinfo.php on line 3

The database server is not available.
```

Database connection error message

Handling MySQL Errors

Suppressing Errors with the Error Control Operator

- Writing code that anticipates and handles potential problems is often called **bulletproofing**
- Bulletproofing techniques include:
  - Checking submitted form data

    e.g. `if (isset($_GET['height']) ...`
  - Using the **error control operator (@)** to suppress error messages

    e.g. `$dbConnect = @mysqli_connect(...);`
    `if (!$dbConnect) ...`

Handling MySQL Errors

Terminating Script Execution

- **die()** and **exit()** terminate script execution
- **die()** version is usually used when attempting to access a data source
- Both functions accept a single string argument
- Invoke the **die()** and **exit()** as separate statements or by appending either function to an expression with the **or** operator

Note: When script is terminated, an *incomplete* html page is sent to the client. This is useful for error diagnostics, but *poor in a production application.*

Handling MySQL Errors (continued)

```
$dbConnect = @mysqli_connect(("mysql.ict.swin.edu.au",
   "s1234567", "ddmmyy")
    or die("<p>The database server is not available.</p>");
// the above is one statement: connected OK or die
echo "<p>Successfully connected to the database server.</p>";

@mysqli_select_db($dbConnect, "s1234567_db")
     or die("<p>The database is not available.</p>");
echo "<p>Successfully opened the database.</p>";
// additional statements that access the database server
mysqli_close($dbConnect);
```

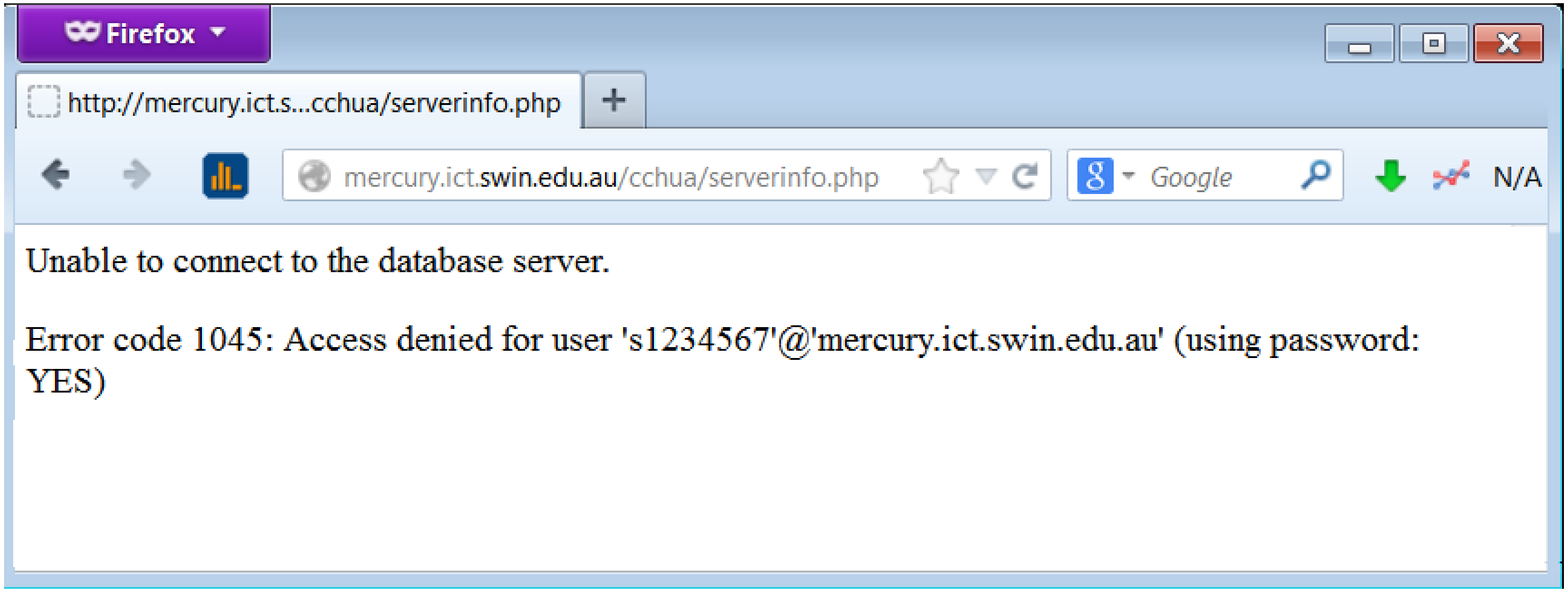
No `if` required here

Handling MySQL Errors (continued)

MySQL error reporting functions

Function	Description
mysqli_connect_errno()	Returns the error code from the last database connection attempt, 0 if no error
mysqli_connect_error()	Returns the error message from the last database connection attempt, empty string if no error
mysqli_errno(connection)	Returns the error code from the last MySQL function call attempted, 0 if no error
mysqli_error(connection)	Returns the error message from the last MySQL function call attempted, empty string if no error
mysqli_sqlstate(connection)	Returns a string of five character error code from the last MySQL operation, '00000' if no error

Handling MySQL Errors (continued)


Firefox
http://mercury.ict.s...cchua/serverinfo.php
mercury.ict.swin.edu.au/cchua/serverinfo.php
Google
N/A

```
Unable to connect to the database server.

Error code 1045: Access denied for user 's1234567'@'mercury.ict.swin.edu.au' (using password:
YES)
```

Error number and message generated by an invalid username and/or password

Reminder: Checking Data Entry

- ***Never trust the user! Never!***
  - **Always** check that input values are of the ***type*** you expect
  - If possible, test that a text value is **within** a **set** of values
  - If showing the content gathered from users, **remove** anything that shouldn’t be there, and **encode** everything else to make sure that nothing is **inserted** into your code! (HTML, JS, CSS or other!)
  - If using information from users as part of a database **query**, **escape** all (string) values, always surround values with **quotes** and log/test whatever you can.

SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10011 Creating Web Applications

What's Next?

- Emerging Internet Technologies
- Web Services
- Cloud
- Internet of Things
- Mobile
- Security Issues

Acknowledgement: The contents in this document was adopted from learning material prepared by Alan Colman (Convener, SUT)