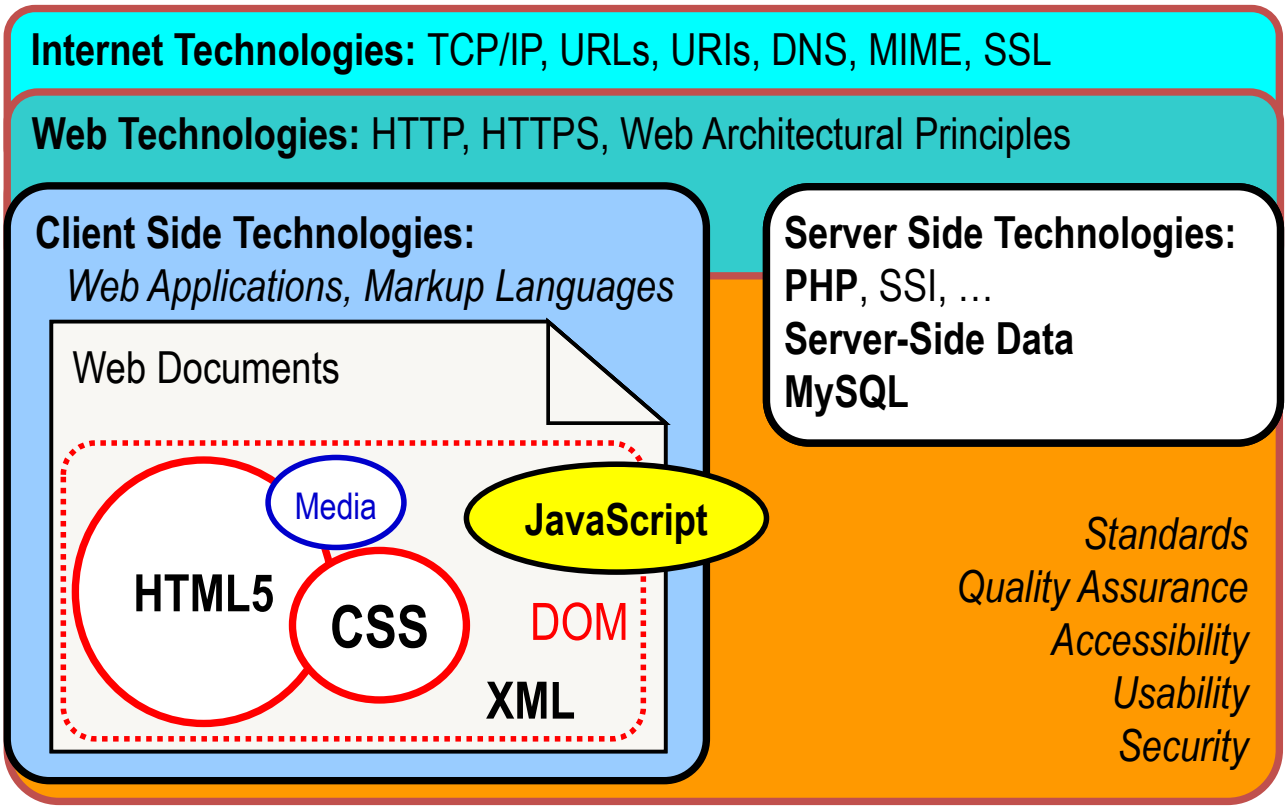
SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10011/60004
Creating Web Applications

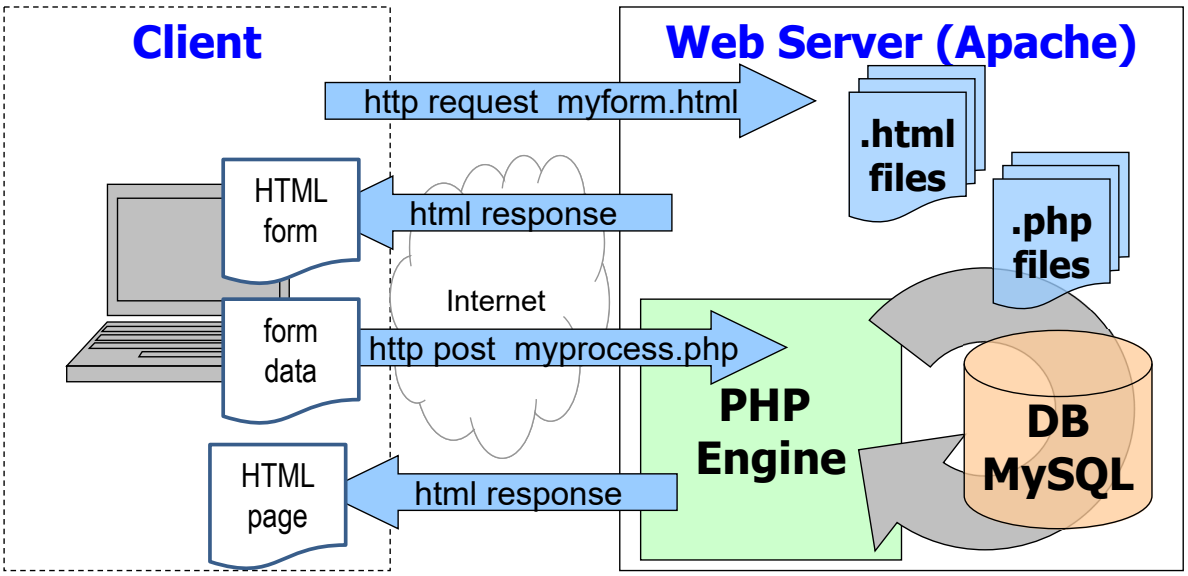
Lecture 10
PHP and MySQL Part 1

Unit of Study Outline



Server-Side Scripting and PHP

Apache/PHP/MySQL example



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

Introduction to Databases

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

- **primary key** - 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
- **foreign key** - 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

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-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-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

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

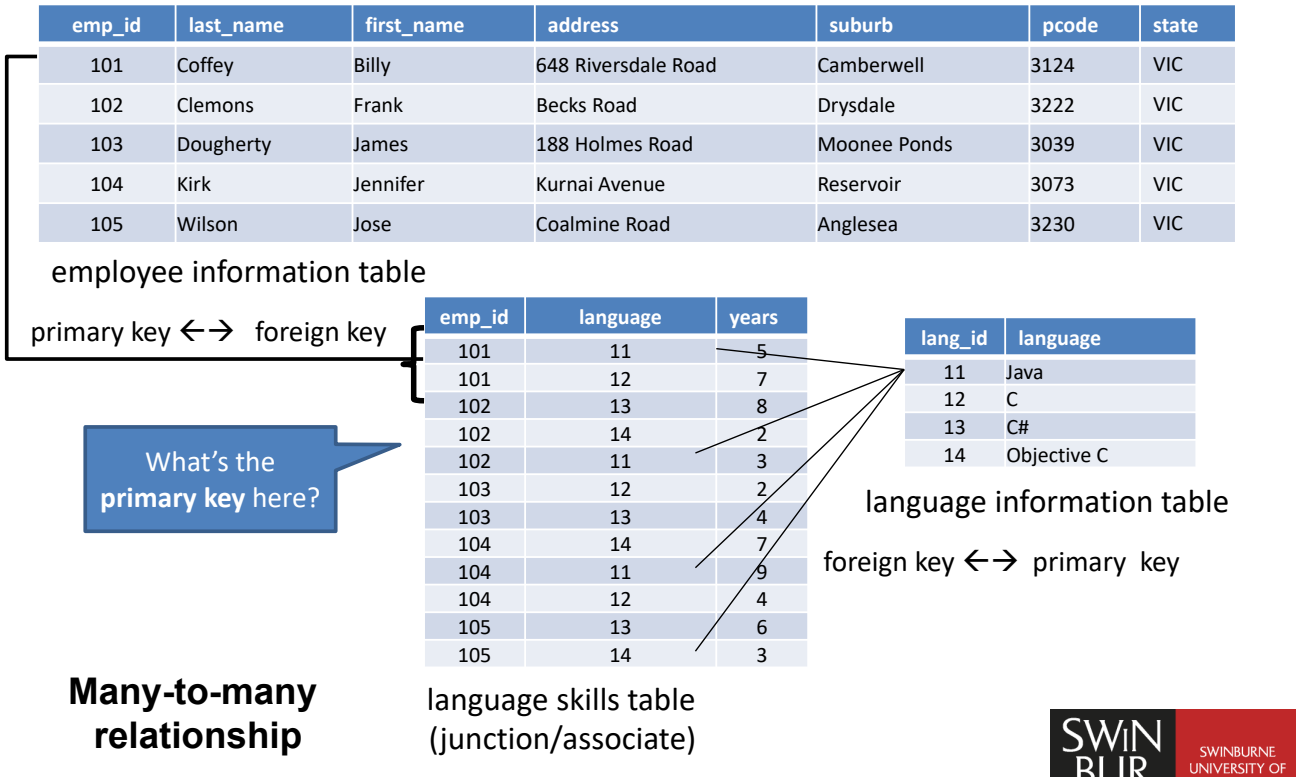
What's the **primary key** here?

One-to-many 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*

Many-to-Many Relationship



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/associate)

What's the **primary key** here?

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

language information table

foreign key ←→ primary key

Many-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.

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

Getting Started with 'MySQL'

- "MySQL" is an open source database server, and it is fast and reliable.
  Acquired by Oracle through the Sun Microsystems acquisition.
- "MariaDB" was developed in 2009 as an alternative open source database server to MySQL.
- There are several ways to interface with a MySQL / MariaDB database server:
  1. Using 'MySQL Monitor', a command-line program
  2. Using 'phpMyAdmin', a web interface program
  3. Using PHP database functions within PHP scripts
- Our "MySQL" database server is now "MariaDB": **feenix-mariadb.swin.edu.au**

For more details see:
https://feenix.swin.edu.au/help/index.php?page=MySQL%20%28MariaDB%29

Logging in to 'MySQL' Monitor

- Our 'MySQL' / 'MariaDB' database server at **feenix-mariadb.swin.edu.au** is already set-up, and your account and database will already have been created.
  See the MySQL Lab notes for more details.

If you want to set up MySQL locally on your own computer, you will need to initialize it, using the following command:

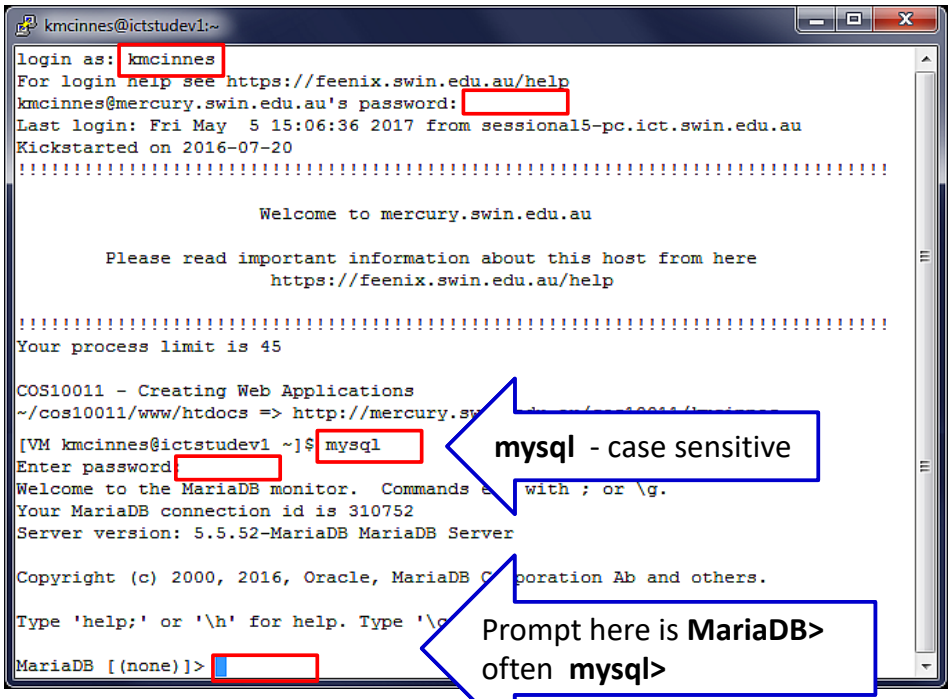
mysql –h host –u user –p
mysql –u user –p

Two accounts would then be created:

1. **Anonymous user account** allows login without specifying a username or password (Note: security issue)
2. **Root account** (the primary administrative account for MySQL) is created without a password mysql –u root

Log out with the **exit** or **quit** commands

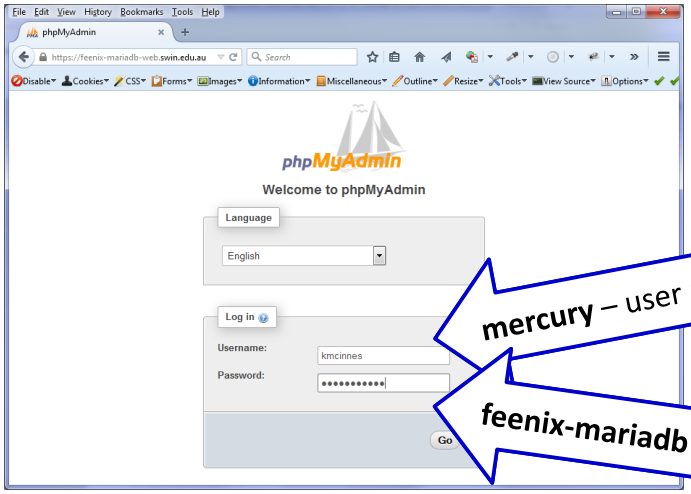
Logging in to MySQL Monitor (continued)



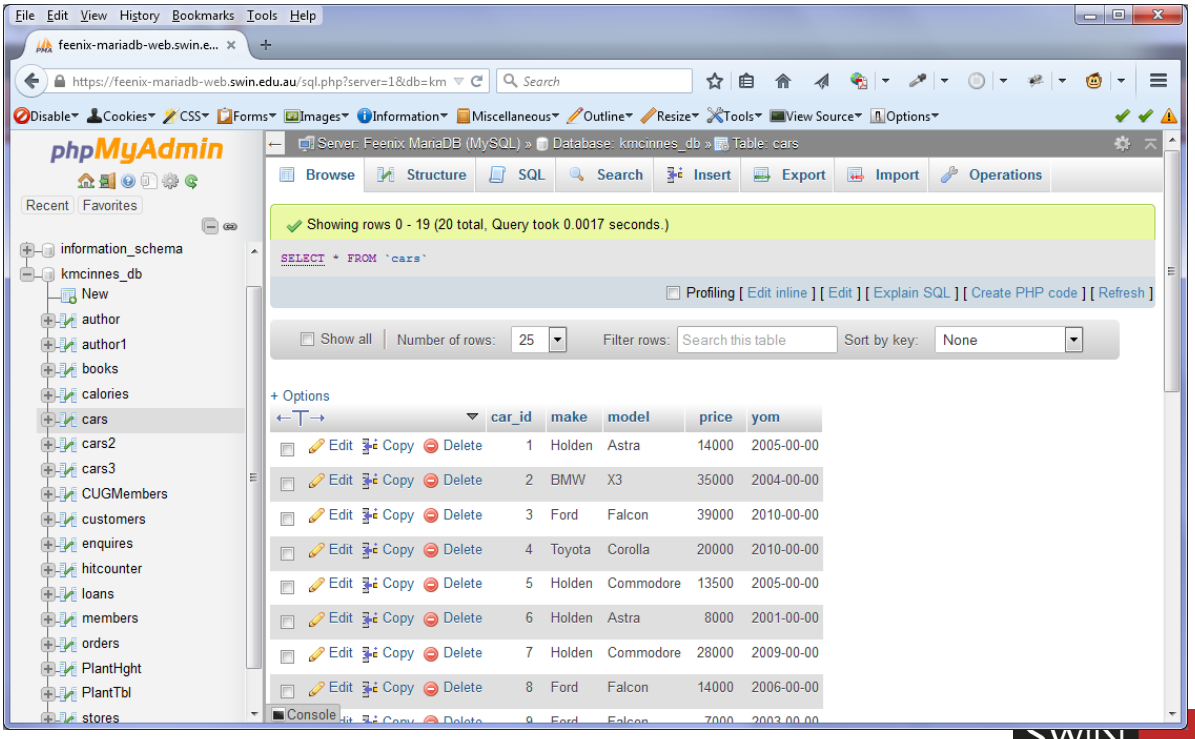
Typical MySQL Monitor on a mercury server

Using phpMyAdmin

- Web User Interface to MySQL / MariaDB
- Log in to **phpMyAdmin** with your MariaDB username and MariaDB password
  http://feenix-mariadb-***web***.swin.edu.au



Using phpMyAdmin



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

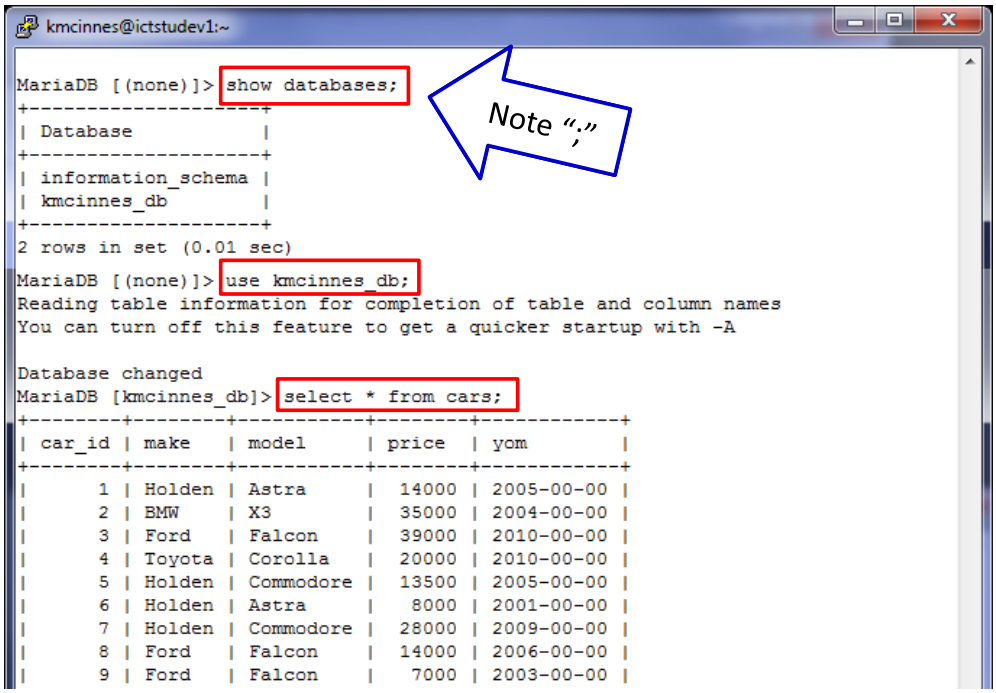
SQL Command Basics

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

- `SHOW DATABASES` statement to view the databases that are available
- 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

You are only given **one** database on MySQL. You can't create or drop your database

Selecting Databases (continued)



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***:

Need to USE database first.

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

Structured Query Language (SQL)

Common SQL keywords

Keyword	Description
INSERT	Inserts a new row into a table
UPDATE	Updates field value in a record
DELETE	Deletes a row from the table
SELECT	Retrieves 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`

SQL queries using MySQL Monitor

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

```
mysql> SELECT * FROM cars;
```

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

```
mysql> SELECT * FROM cars
    -> 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;
```

```
cchua@mercury:~
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://php.net/manual/en/mysqli.summary.php

Hint: Separate file for your login info

Example

```
<?php
        $host = “feenix-mariadb.swin.edu.au";
        $user = "s1234567";
        $pwd  = “password";
        $sql_db  = " s1234567_db";
?>
```

Edit the host name when ported to a production server

Your student id

Initially ddmmyy. Change, but don’t use your SIMs password

ITS has created a predefined database for you

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

HUPD

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("feenix-mariadb.swin.edu.au",
   "s1234567", "yourMySQLpassword", "s1234567_db");
```


Connecting to MySQL (continued)

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

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

HUPD

  - The ***host*** argument specifies the host name where your MySQL/MariaDB database server is installed

    e.g. `feenix-mariadb.swin.edu.au`
  - The ***user*** and ***password*** arguments specify a MySQL/MariaDB account name and password

    e.g. `s1234567 yourMySQLpassword`
  - The ***database*** argument specifies a database

    e.g. `s1234567_db`

Connecting and Selecting

- The **`mysqli_connect`** also allows one to connect and select the database in one step.

```
$dbConnect = mysqli_connect(
  "feenix-mariadb.swin.edu.au",
  "s1234567", "ddmmyy", "s1234567_db");
```

Your MySQL/MariaDB password

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

Database and Table queries:

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

Closing Connection

- Close a connection to a MySQL/MariaDB database server with the `mysqli_close()` function

```
mysqli_close($dbconnect);
```

Next Week

- **PHP and MySQL Part 2**