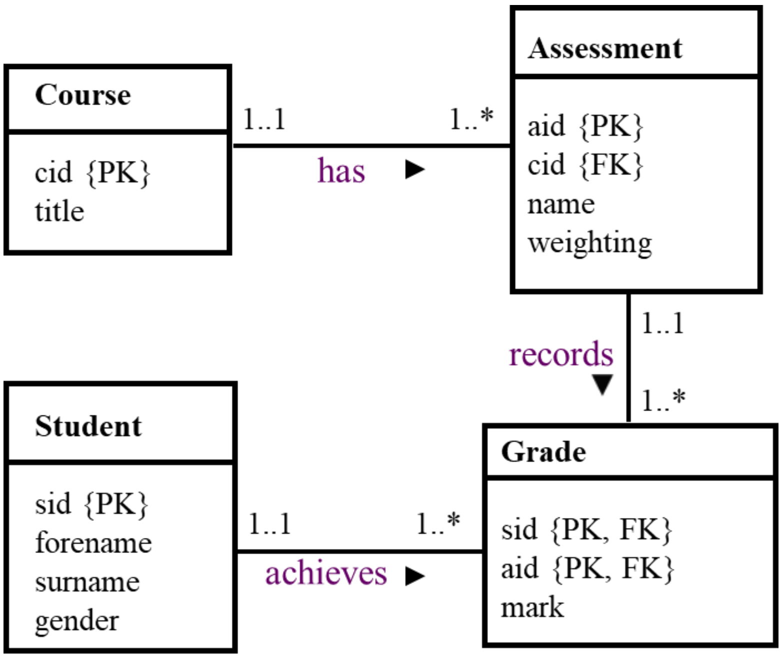
CO323 Assessment 2 – SQL, PHP & MySQL

Introduction

A relational database has been created using MySQL to store details about students' performance. There are four tables in the database: **Student**, **Course**, **Assessment**, and **Grade**. A student takes one or more courses. A course may be assessed by coursework assessments and exam. The name and weighting of each assessment component are stored in the Assessment table. Students' marks on their assessments are recorded in the Grade table.

The database itself is stored on dragon, and the tables can be viewed there. An Entity-Relationship model of the database is shown below:



where PK and FK refer to primary key and foreign key, respectively.

The credentials you will need to access the database are given below:

Server Name:	`dragon.kent.ac.uk`
Database name:	`co323`
Username:	`co323`
Password:	`h@v3fun`

This assignment consists of two parts:

- **Part A** – Writing SQL queries; and
- **Part B** – Writing PHP scripts that access the given MySQL database and display the results of the SQL queries on the web.

You will need to complete and submit part A before starting Part B. For submission mechanisms and deadlines, please see the module moodle page.

Marking scheme

In total 40 marks:

- 16 marks for Part A - SQL
- 22 marks for Part B – PHP & MySQL
- 1 mark for good presentation of outputs using CSS (appropriate colour, font, text alignment etc.)
- 1 mark for good coding style (appropriate comments & indentation)

The mark will be converted to a percentage for recording in SDS.

Part A – SQL

Write an SQL statement for each of the following tasks:

1. [4 marks] List the names and weightings of all assessments in the course titled 'Database systems'. The list should be ordered by the assessment name.
2. [4 marks] Find the course ID, the name and average mark of each assessment. Results should be ordered by the course ID and then the assessment name.
3. (Challenging) Show the detailed results achieved by a specific student: sid= 'S0001'.
   a) [3 marks] For each course taken by the student, show the course ID, title, name, weighting and mark achieved for each assessment.
   b) [5 marks] Show the course ID and the student's final result (weighted average) for each of the courses they have taken.

You should record your SQL statements for Tasks 1-3 in a plain text (.txt) file. Please include your name and login and label your answers clearly.

You should submit your work before the deadline using the 'Assessment 2 Part A submission link' on the module Moodle page.

Note: A model solution to the tasks in Part A will be available to all students immediately after the Part A deadline so that everyone can have a fair attempt at part B. Consequently, **no extensions can be given**. If you have a good reason for not being able to submit on time, you will need to claim a concession.

Part B – PHP & MySQL

Submission: Part B involves creating a number of PHP files. All these PHP files should be placed *before the deadline* in your web-enabled submission area on raptor. This folder is called:

- `\\raptor.kent.ac.uk\exports\proj\co323c\a2\`*`your-login`*`\public_html` when you are accessing it from Windows, or
- `/proj/co323c/a2/`*`your-login`*`/public_html` when you are accessing it from raptor directly, or
- `http://raptor.kent.ac.uk/proj/co323c/a2/`*`your-login`* from a web browser.

Note that this is **not** the folder you have been using for classwork.

You will need to put your work directly into this public_html folder, **not into a sub-folder**.

You are **strongly** recommended to develop your solution in its final submission location, rather than copying it across later.

We will mark your work by using the URL above, so you **must** ensure that your site works at that location (check permissions, relative links, etc.). Work not submitted to the correct location will get a mark of 0. No alternative submission mechanisms (e.g. email) will be accepted, so give yourself plenty of time to deal with network and access issues before the deadline.

To help you start working on the tasks in Part B a simple PHP script is given below. It connects to the database, retrieves details of all courses, and then generates a web page displaying the results.

```
 <?php  // Connect to database, and print error message if it fails
   try {
      $dbhandle = new PDO('mysql:host=dragon.kent.ac.uk; dbname=co323',
        'co323', 'h@v3fun');
   }
   catch (PDOException $e) { die('DB connect error: ' . $e->getMessage()); }

   $sql = "SELECT * FROM Course";   // The SQL query itself

   $query = $dbhandle->prepare($sql);     // Prepare and ...
   if ( $query->execute() === FALSE ) {  // ... execute the query
      die('Query exec error: ' . implode($query->errorInfo(),' '));
   }

   $results = $query->fetchAll();   // Put all the results in an array
?>
   <h2>Details of all courses</h2> <!-- static HTML heading -->
<?php // Generate HTML from the contents of the results array
   foreach ($results as $row) {
      echo "<p>".$row['cid'].": ".$row['title']."</p>";
   }
?>
```

If you create a file called `course.php` from the above script, and save it in your own submission folder on raptor, you should be able to load the generated page using the URL below:

```
http://raptor.kent.ac.uk/proj/co323c/a2/xyz/course.php
```

(where `xyz` should be replaced by your own username.)

Now complete the following tasks. You may, if you like, make use of the script as a starting point.

4. [4 marks] Write a PHP script `task4.php` that connects to the database and displays the results of the query in Task 1 in an HTML table.
5. [4 marks] Write another PHP script named `task5.php` to show the results of the query in Task 2 in an HTML ordered list.
6. [7 marks] Create a PHP script named `task6.php` that retrieves details of each student, i.e. the student ID, full name and gender. The output page should contain an HTML form which has a drop-down list input and a submit button. The drop-down list input should contain the results retrieved from the database. When a user selects a specific student from the drop-down list, the student ID is submitted via the GET method to another PHP script named `task7.php` (to be created in Task 7).
7. [7 marks] (Challenging) Create a PHP script named `task7.php` that accepts the student ID submitted from `task6.php` and then displays the detailed results achieved by the given student in an appropriate format. For each course taken by the given student, it should show the course ID and title, the name and weighting of each assessment and the mark achieved by the student. It should also show the student’s final result on each course as well (the weighted average of their marks). You should

make use of the queries in Tasks 3(a)-(b). You should use a parameterised query to prevent SQL injection.

8. (Optional) Now turn the scripts you have created so far into a simple website.
   (a) Create a PHP script file named `menu.php` that contains a link to each of the pages you've created so far.
   (b) Create an HTML page, named `loginform.html`, that contains a simple HTML login form with a text input for a username and a password input for a password, along with a submit button. The form should submit via the POST method to `login.php` which will check user login details (to be created in (c)).
   (c) Create a PHP script file named `login.php.` It retrieves the parameters passed from `loginform.html` and checks whether the username and password are correct. It only accepts the username `'abc'` and password `'Only4Testing'.` If both username and password are correct, it starts a session, sets a session variable called `loggedin` to true and redirects the user to the page `menu.php`. Otherwise it displays an appropriate message and a link back to `loginform.html` instead.
   (d) Create another PHP script named `logout.php` that destroys the session if the user is logged in, otherwise it redirects the user back to `loginform.html`.
   (e) Modify your scripts for Tasks 4~7 & 8(a) such that they should also start a session and check whether the session variable `loggedin` is set. If a user is logged in your scripts should operate as before, otherwise the user should be redirected back to `loginform.html.`

   Once this task is done, you should have a small web system in which users need to log in and then access its functions and log out when done. Take a look at the lecture material on sessions, and the PHP `header()` function to do the redirect.

Yang He/Ian Utting 2020