G51DBS – Database Systems 2014

Coursework 2 Requirements, v1.0

Overview of this document

This document includes information on how to set up the coursework 2 files and get your original database. It then gives an overview of the database structure. Finally it discusses the PHP file which you need to edit and the .sql file which you will fill in with most of your answers.

This coursework is worth 20% of the module. Each of the questions is worth 10% of the coursework mark (i.e. 2% of the module mark). However, they are also designed to help you to practice and to be more ready for the exam, so their value is more than that.

Using a database and setting up PHP

These instructions are virtually the same as for Exercise 4 so should be easy for you. **Please ensure that you have done Lab Exercise 4 before you continue.**

Login to avon.cs.nott.ac.uk and ensure that you are in your home directory (see the instructions in Exercise 4). You should then see the following command line:
username@avon:~>

Now you are in your home directory, you need to create the necessary php and other files ready for your web page. To make this process easier, we will download a compressed website from elsewhere, and extract it into your home directory. The steps we will follow are shown below:

1. Ensure that you are logged into avon, have a command prompt visible AND check that you are in your home directory.

2. Download the required file from the school servers using the following command:

```
wget http://www.cs.nott.ac.uk/~jaa/dbs/dbscw2014.tar.gz
```

This command downloads the file straight to your home directory. This file contains all the files we need to start a website with PHP. If it fails, use:

```
cp /lhome/jaa/public_html/dbs/dbscw2014.tar.gz .
```

The **cp** will copy the file from Jason's home directory to the current directory. If the proxy is not working correctly, the first command MAKE SURE THAT YOU ARE IN YOUR HOME DIRECTORY WHEN YOU RUN THESE

3. The file is a tar archive, that has also been zipped. When you extract it, it will create the complete website you need to get started, with all of the correct file permissions.

Use the following command to do this:
`tar -pxzvf dbscw2014.tar.gz`
(Make sure that you are still in your home directory when you run this command since it will create a directory called **`public_html/dbscw2014`**. Files need to be inside your public_html directory to be available via the web server.)

This command will extract all the files we need. (See Exercise 4 for more details of how this works.) Importantly, it will also keep the file permissions so that nobody else can read/copy your answers.

4. Now we have downloaded and extracted the file, we should be able to view our test website. If you open your web browser and go to:
http://avon.cs.nott.ac.uk/~*yourusername*/dbscw2014/dbscw2014.php

You should get a message like this: "Failed to connect to MySQL server: Access denied for user 'username'@'avon.cs.nott.ac.uk' (using password: YES)"

If you want to check the PHP part, then you can instead open this file:

http://avon.cs.nott.ac.uk/~*yourusername*/dbscw2014/dbscw2014test.php

IMPORTANT: You can only use PHP pages from within the university. These pages will not be visible from off campus for security reasons (PHP pages let you potentially change the server or database). So please make sure that you are using a PC within the school (e.g. the A32 lab) when you test your PHP files.

5. Navigate to the public_html/dbscw2014 directory, e.g.:

```
username@severn:~> cd public_html/dbscw2014
username@severn:~/public_html/dbscw2014>
```

Open the dbconnect.php file from the public_html/dbscw2013 directory in a text editor and enter your mysql username and password details into the dbconnect.php file in the appropriate places. You can use windows to change it, or use a Linux text editor if you are familiar with these. Note: If you check the permissions (e.g. by using **ls –al**) you should see that it is readable only by you, so nobody can read these details.

6. Create the database files:

Login to MySQL as your user, from the directory which you are in (the one with the php files in it).

Run '**source init2014cw2.sql**' to execute the file called init2014cw2.sql which is in that directory. This will create the database files for you.

7. Test your initial coursework files by opening the following URLs in a web browser:
http://avon.cs.nott.ac.uk/~*yourusername*/dbscw2014/dbscw2014.php
you should now see your initial coursework PHP page.

Overview of the coursework

For this coursework you have been provided with a database which stores the details for part of a very simple test-based exploration game. The details for the game are stored in a database and you will write some queries to produce reports of the information. You will integrate ONE of these reports into the PHP file in question 2. Many of the other reports and updates could later be integrated into the game to provide various facilities, but that is beyond the scope of this coursework.

How to get started:

I suggest starting by looking at the script (**`init2014cw2.sql`**) which creates and populates the database tables, and compare it against the descriptions on the next page. Ensure that you understand what each table is for and how they fit together to meet the requirements. Also experiment with the initial PHP file (the main code is in **`internaldbscw2014.php`**) that you have, since it will allow you to move around the virtual house and pick things up.

Overview of tables:

The database contains details of a number of locations (places within a house, in this case, such as the living room, kitchen or bedroom).
Each location can have one or more exits to other locations (LocationExit table).
Each location can also have one of more items (e.g. a key or a TV) located in/at it (Item table).

The items (rather than MobileObjects) can be picked up, carried and dropped by the player. Each item has a physical location if it is lying in a room. Some items will be being carried, in which case they have no location (their location is NULL). Ideally we would want to include information somewhere about who is carrying the item, but to simplify the coursework we are assuming that there is only one person wandering around and that any item without a location (i.e. where location is NULL) is being carried by that person.

There can also be various MobileObjects. In the supplied data, all of these are people. These MobileObjects/People can be in locations and in theory they can move around on their own. (I intend to keep adding features each year to form a new coursework, so these will probably have more features next year.)

IMPORTANT NOTE: I have deliberately made some of the column/attribute names less useful than they could be. This avoids the case where some questions have trivial answers and avoids some issues people had last year where they thought that NATURAL JOIN could be used everywhere and never really understood the other joins. Don't use these as examples: please use sensible names in your own database designs.

Details of the database tables which have been created for you:

The following tables exist in the example database:

Table	Purpose
Description	This table contains all of the descriptive text that will be used. Putting all of the information into one table makes it easy to change it, to correct spelling errors or to translate the language.
Location	This contains a list of all of the locations in the house (and garden). Each has a unique id number, and a description id number. locID : a unique ID for this location locDescID : the ID for the textual description of the location. A foreign key into Description.
LocationExit	Each location may have one or more exits. Each exit is described by an entry in this table. locIDCurrent : the ID of the location that this is an exit FROM. A foreign key into Location. exitDirectionDescID : the ID of a description of the exit direction, e.g. “North”, “South”, “East”, “West”, “Up”, “Down”. A foreign key into Description. locIDNew : the ID of the location which will be reached if you take this exit. A foreign key into Location. exitDescID : the ID of a description of the exit type, e.g. “Door”, “Passageway”, etc. A foreign key into Description.
Item	A number of items are scattered around the house and garden. Each item has a unique ID, a name and a location. itemID : the unique ID for the item itemLocID : the ID of the location where the item is currently. A foreign key into Location. *This value is permitted to be NULL and if so then the item should be assumed to be being carried (i.e. at no location).* itemInitialLocID : The initial location of the item. itemDescID: the ID for the description of the item. A foreign key into Description.
PlayerData	Certain data will need to be maintained to allow you to wander around this virtual house. This table contains the various data items, each of which is identified by a unique name. You only need to consider the ‘Position’ value for the moment. dataID : the unique id (a string NOT an int!) for this piece of data. dataValue : the value of this piece of data. E.g. the value of the row with dataID of 1 is the location number to show at the moment. dataInitial : the initial value for this item.
MobileObject	These are people who can move around the virtual house. They are similar to items except that each has an additional (unused at present) mobHealth value.
ReverseExits	You will not need to use this table. It is used by an update at the end of the init2014cw2.sql file to fill in the reverse paths. You may find it interesting to investigate how this works but it is not necessary.

Important requirements/information:

- All code should be written in the **`internaldbscw2014.php`** or **`cw2_2014_SQL.sql`** files, which have already been started for you. Questions 1 to 9 test your ability to do SQL queries. Q10 tests your ability to incorporate this into the PHP file.

- You will submit two files (the **`internaldbscw2014.php`** and **`cw2_2014_SQL.sql`**) through the link on the moodle page.

- You can test your cw2_2014_SQL.sql file by using the command '**`source cw2_2014_SQL.sql`**' from within mysql. (Make sure that you were in the directory with your files in before you start mysql.)

- We will test the .sql file by running it within MYSQL using the 'source' command. Please ensure that this works correctly and do not submit a file which will not run. We will look at both the output and the select statements when marking.

- Each question (Q1 to Q9) should be completed using a single SQL statement. You must not use PHP to remove table rows, concatenate tables, order the result rows or execute multiple queries for any of the questions. There is at least one answer for each question (and probably many answers) which requires only a single SQL query.

- We will test the PHP question (Q10) by running the PHP page on our copy of the database and looking at the query code and the resulting output on the page.

- After submission I suggest that you leave your files where they were on the H: drive and then do NOT modify them. Every year there are some issues where people submit the wrong files or forget to submit one of the files and this at least means you will not have lost your work if you do this and could resubmit if there was an error (and we could check the file dates to ensure that they were unchanged). ***To be clear though: you need to submit the (correct) files if you want the marks and if you do make an error submitting you WILL lost marks, so please get it right!***

- In your answers you may only use information from the question. E.g. if it says 'North' in the question then your query should use the string 'North', not the ID for the north direction (i.e. do not look up IDs manually in the tables). You could, however, do a sub-query to find the id of 'North' and then use it within the main query if you wished.

The ten questions:

For questions 1 to 9, add your SQL statement to the correct place in the **cw2_2014_SQL.sql** file.

For question 10, add your code to the **internaldbscw2014.php**.

You should submit both of these files as your coursework answer.

Question 1:

Add a query to present a list of MobileObjects, showing their descriptions, unique id numbers, and the description of the location at which they are currently located. Name the columns as "Description", "ID" and "Location". Sort the resulting list into alphabetical order of the mobile object description.

Example output:

```
+------------------+----+-------------------------------------+
| Description      | ID | Location                            |
+------------------+----+-------------------------------------+
| The gardener     |  1 | In the garden                       |
| Your best friend | 10 | In the SW corner of the Living Room |
| Your brother     |  2 | At the south end of the Hallway     |
| Your father      |  6 | In the bathroom                     |
```

Question 2:

Provide a single query which will display a list of all items which are in the same locations as mobile objects, as a list of pairs of item ids and object ids. Show the description of the mobile object, the description of the item, the id of the mobile object and the id of the item.

Example output:

```
+------------------+----------------+-------+--------+
| description      | description    | mobID | itemID |
+------------------+----------------+-------+--------+
| Your brother     | Front Door Key |     2 |     20 |
| Your brother     | Back Door Key  |     2 |     21 |
| Your brother     | Cup            |     2 |     33 |
```

Question 3:

Provide a report of the mobile objects and all of the possible exits from their current location. Provide the following columns: The id of the mobile object, the name of the mobile object (called 'Name'), the name of the exit direction (called 'Exit'), the exit direction number and the new location id that is in that direction.

Example output:

```
+-------+-----------------+-------+--------------------+----------+
| mobID | Name            | Exit  | exitDirectionDescID | locIDNew |
+-------+-----------------+-------+--------------------+----------+
```

```
|     1 | The gardener     | South |                  2 |      116 |
|     1 | The gardener     | East  |                  3 |      109 |
|     2 | Your brother     | North |                  1 |      107 |
```

Question 4:

Provide a single SQL statement which will find all mobile objects which could move 'North' (you may check for direction id 1) from their current location and change their current location to be the location to the North (i.e. in direction id 1). You MAY use the ID rather than the string 'North' in your query.

e.g. if 'Your brother' (mobID=2) is in a location with an exit in direction 1 (north, to location 107), then after this statement has been executed the location id of mobID 2 would have been set to 107.

Example output:

```
Query OK, 6 rows affected (0.00 sec)
Rows matched: 6  Changed: 6  Warnings: 0
```

Hints: Consider how to change the location of something: look at how the player location is changed in the PHP file. Then consider your answer to the previous question. The target of an update can be a join rather than a single table.

Question 5:

Using an appropriate outer join, produce a report which shows the name and ID (mobID) of every person (mobile object) in the database and all items which are in the same location as the person. If there is no item in the same location, display the value 'NULL' in the Item field. If there are multiple items in the location then multiple rows should be shown for the same person.

Example output:

```
+-----------------+----+----------------+
| Person          | ID | Item           |
+-----------------+----+----------------+
| The gardener    |  1 | NULL           |
| Your brother    |  2 | Front Door Key |
| Your brother    |  2 | Back Door Key  |
| Your brother    |  2 | Cup            |
```

Question 6:

You would like a report of what items are lying around the house, how many there are of each type, and the sort of locations that they are in. Assume that each item description determines the type, so any that items with the same description are the same type of item. Display a report of the number and range of location ids that each item type (i.e. description) is in, showing the minimum location id, the maximum location id, and number of items of that type.
Do NOT include items which are currently carried.
Do NOT include items where there are exactly two of that type of item.

Example output:

```
+----------------+--------+--------+--------+
| Item           | MinLoc | MaxLoc | Number |
+----------------+--------+--------+--------+
| Back Door Key  |    106 |    106 |      1 |
| Blanket Duvet  |    111 |    111 |      1 |
| Cup            |    106 |    106 |      4 |
```

Question 7:

Using a sub-query to identify the locations with Dinner Plates in them, produce a report listing the names and ids of all mobile objects which are in the same locations as Dinner Plates.

Example output:

```
+-----------------+-------+
| description     | mobID |
+-----------------+-------+
| Your friend     |     7 |
| Your best friend |    10 |
```

Hint: Test this by picking up some dinner plates and dropping them in the same locations as some people.

Question 8:

You need a report to see who your friends could move to go and talk to.
Display a report of all of your friends and the other people (mobile objects) who are exactly one move away from them (i.e. for whom there is an exit from your friend's location to the other person's location).
Display the name of your friend (in the first column), the name of the person they could move to join, the id of your friend and the id of the person they could join.

Example output:

```
+-----------------+-----------------+-------+-------+
| description     | description     | mobID | mobID |
+-----------------+-----------------+-------+-------+
| Your friend     | Your brother    |     8 |     2 |
| Your friend     | Your mother     |     9 |     5 |
```

Question 9:

In a single SQL statement, provide a query which will display a list of both items and mobile objects, showing their id, description, location id and the description of their current location. Include items which are carried but show the location as 'Carried'. Name your output columns as 'ID', 'Description', 'Location' and 'LocID'. Sort your results into alphabetical order of Location then Description.

Example output:

```
+----+-----------------+-----------------------------------+-------+
| ID | Description     | Location                          | LocID |
```

```
+----+-----------------+-------------------------------------+-------+
| 11 | Magazine        | Carried                             |  NULL |
| 12 | Magazine        | Carried                             |  NULL |
|  5 | Mobile Phone    | Carried                             |  NULL |
| 30 | Shampoo         | In the bathroom                     |   112 |
| 29 | Soap            | In the bathroom                     |   112 |
```

Hints: You may need to join the results of multiple queries together (probably three queries, to cover items on the ground, items carried and mobile objects). Consider the answer for Q1 since it does part of this for you (with a little modification). You may also find it useful to refer to the slides for lecture 9.

Question 10:

Add a new table to the PHP report, by inserting the correct code into the **`internaldbscw2014.php`** file to add a list of the mobile objects at your current location and their IDs. You can either display this information as two columns, or as a single column with the IDs in brackets (whichever you prefer).

You can see an example implementation here:
http://severn.cs.nott.ac.uk/~jaa/dbscw2014_demo/dbscw2014j.php