IFB105 Database Management

**Project - Part B**

**Project overview**

This IFB105 project gives you an opportunity to apply the concepts and skills you acquire in the unit to a ‘realistic’ database design scenario and reflect on the data requirements of an organisation.

The submission is divided into two parts due at different times during the semester. These will cover:

1. Design of a database
2. Creation and use of databases

**The tasks for Part B**

For Assessment 2B you will be required to:

1. Create a database for the fictitious University database
2. Query the *Treasure Hunter* Database (including security tasks)
3. Normalise a relation.

**Weighting**

Part B is worth 25 marks, for **25%** of the unit.

**Groups**

You can complete this assignment individually or in pairs. You do not have to work in the same pairs as you did for Part A.

If you choose to work in a pair, only one student should submit the assignment. Please provide the name and student number of the person you worked with in the README.txt file. No consideration will be given to students who claim they did more work in their pair than the other student because this assignment can be done individually.

Doing the assignment in undeclared groups, or groups larger than two students, will be treated as plagiarism. Pairs that work together and then split due to difficulties must not submit any of the same work, or it will be treated as plagiarism.

**Due date**

Tuesday 15 October at 11.59 pm

**Submission**

You must submit 4 files in a ZIP file using the submission link in Blackboard:

1. SQL script (a text file with the file extension changed to sql) containing your solution to task 1.
2. SQL script containing your solutions for tasks 2, 3, 4 and 5 (do not upload the database import script as part of your solution).
3. Word or PDF containing your solution to task 6.
4. README.txt containing your full name, student number (and the name and student number of your partner if working in pairs) and a list of any queries you have attempted but were not able to successfully run in Workbench.

**If you do not follow these submission guidelines you will lose a mark.** Scripts in different file types will not be accepted.

**Late submission**

Assessment work submitted after the due date will be marked only with an approved extension (MOPP E/6.8.2). Assessment work submitted after the due date without an approved extension or, where an extension has been granted, after the extended due date, will not be marked and a grade of 1 or 0% will be awarded against the assessment item. The unit outline sets out the requirements surrounding extensions (including for disabilities), penalties for late submissions and appeals.

Please note: extensions will not be granted for group (2 members) submissions, regardless of the reason.

**Assessment 2B tasks**

For this assessment, you will:

1. Build a script that will create a database for a given relational schema (Week 7)
2. Provide the SQL commands needed to retrieve the required data using the Treasure Hunter schema (Week 8 to 10)
3. Provide the commands to modify (INSERT, UPDATE & DELETE) the data using the Treasure Hunter relational schema (Week 7)
4. Provide the commands needed to create appropriate indexes and views (Weeks 7 to 10)
5. Provide advice on the basic security measures that should be implemented (Week 10 and 11)
6. Normalise a given relation (Week 6).

**Task 1 [4 marks] University Database**

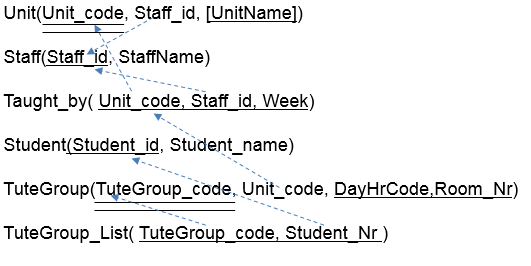
An SQL script is a set of SQL commands saved as an SQL file. If you are already working with MySQL, you can either execute an SQL script file using the source command or import it in Workbench.

Write an SQL script to create a database to match the Rmap provided below. ***Your script MUST execute in MySQL Workbench without errors to receive full marks. You can ensure that by executing your commands in correct order as well as avoiding use of non-MySQL functions such as CHECK.***

Marks will be awarded for the following:

1. Creating the database and successfully creating new tables including all attributes (1 mark)
2. Including constraints (1 mark)
3. Correctly creating Primary Keys (1 mark)
4. Correctly creating Foreign Keys (1 mark)

***Rmap of a given relational schema to create a database for Task 1***

****

**Notes**

* Units are taught between weeks 1 and 13
* Staff ID is a 7 digit auto incrementing number
* Student ID is a 7 digit auto incrementing number
* Unit codes are 3 numbers and then 3 letters
* Student and staff names include both first name and surname
* Day hour code is the full date and time
* Tute group codes are the a combination of numbers and letters
* The default room number for tute groups is 1

**Task 2 [13 marks] using the Treasure Hunt database**

For task 2, we have provided you with the creation script for the Treasure Hunter’s database. Run this script in MySQL Workbench to create the database. You should execute your query solutions to extract the necessary information.

The script is based on the following schematic:

*TREASURE HUNTER’S RELATIONAL MODEL*

**Player** (username, firstName, lastName, gender, DOB, email, streetNo, streetName, suburb, state, postcode,

creationDateTime, totalPoints)

**PhoneNumber** (phoneNumber, username)

**Treasure** (treasureID, description, points, webpage, type, questID)

**Quest** (questID, questName, story, beacon, advancedQuestID)

**Store** (storeID, storeName, openTime, closeTime)

**Badge** (badgeID, badgeName, badgeDescription)

**PlayerProgress** (questID, username, progress)

**PlayerTreasure** (username, treasureID)

**Purchase** (purchaseID, storeID, username, badgeID, purchaseDateTime, cost)

FOREIGN KEYS

* PhoneNumber (username) is dependent on Player(username)
* Quest (advancedQuestID) is dependent on Quest(questID)
* Treasure (questID) is dependent on Quest (questID)
* PlayerProgress (questID) is dependent on Quest (questID)
* PlayerProgress (username) is dependent on Player (username)
* PlayerTreasure (username) is dependent on Player (username)
* PlayerTreasure (treasureID) is dependent on Treasure (treasureID)
* Purchase (storeID) is dependent on Store (storeID)
* Purchase (username) is dependent on Player (username)
* Purchase (badgeID) is dependent on Badge (badgeID)

OTHER CONSTRAINTS

* Player (gender) must be female, male, other, or prefer not to disclose.
* Player (state) domain is [QLD, SA, TAS, NSW, WA, NT or ACT].
* Treasure (type) domain is [common, uncommon, rare, ultra-rare or elite].
* Players may enter up to three phone numbers.
* Players must enter at least one phone number.
* A quest can be advanced (indicated by the questID appearing in advancedQuestID column).
* PlayerProgress (progress) domain is [active, inactive or complete].
* Player (email) is mandatory.

Query 1 (1 mark)

Write a query to list the treasureID, description, points and types of treasures that contain either ‘brick’ or ‘map’ in their description.

Query 2 (1.5 marks)

Write a query to list the total number of treasures for each treasure type. Your output should contain the type and the total of each type in ascending order of the number of types.

Query 3 (1.5 marks)

Write a query that lists the name, badgeID and the cost of the most expensive badge.

Query 4 (3 marks)

Write a query that lists all badge sales. Your output should show the name of the badge together with first name, last name and email address of the player(s) that made the purchase. Sort the list based on the badge name followed by first name then last name in ascending order.

Query 5 (3 marks)

Write a query that provides the players’ name (first and last), username and how many advanced quests they have completed. If a player did not complete any advanced quests, do not include them in your output.

Query 6 (3 marks)

Write a query to produce a report for each store including stores without any sales. Your result-set should include the following information:

* the storeID
* the store name
* the number of players that have purchased a badge from the store
* the number of players that have not purchased a badge from the store;
* the total money spent at the store
* the most expensive badge a player has purchased at the store
* the cheapest badge a player has purchased at the store
* the average price of the items that have been purchased at the store.

**Task 3 [1.5 marks]**

Insert (0.5 marks)

Write an INSERT command to insert a row into badge table. The badge is called ‘Summer Rain’ and the description should be ‘Beach, sun and holidays’.

Delete (0.5 marks)

Write a DELETE command to remove all the rows from the player progress table for which progress is complete.

Update (0.5 marks)

Write an UPDATE comment to change the address of all players with the last name ‘Halpin’ who live at ‘1800 Zelda Street, Linkburb’ to ’72 Evergreen Terrace, Springfield’.

**Task 4 [1.5 marks]**

Create Index (0.5 marks)

Currently the database only contains a small number of records. However, the data contained within it is expected to grow significantly in the future. Creating indexes on commonly searched columns is a way performance issues can be minimized.

Write a command to create an index on story column of the quest table.

Create view – 1 mark

Write a command to create a view to list the firstname, lastname and account creation date of all players that have started a quest but are currently inactive.

**Task 5 [1 mark]**

Working as a Database Administrator for MySQL Treasure Hunter database, write the following commands for two employees namely Lisa and Sri to achieve the following database security requirements:

1. User Lisa is no longer allowed to add data to the Player table (0.25 marks)
2. User Lisa is no longer allowed to delete records from the Player table (0.25 marks)
3. User Sri must be able to add records to the Quest table (0.25 marks)
4. User Sri must be able to remove records from the Quest table (0.25 marks)

Assume usernames of employees namely Lisa and Sri are *lisa* and *sri* respectively.

**Task 6 [4 marks]**

***Hint: Going through the examples of functional dependencies and normalization covered in the Week 6 Lecture will help you to work on this question.***

Using the following table structure, identify all functional dependences and then decompose this table into a set of 3NF relations. Your answer should:

* List the functional dependences
* Contain the relational schema for the relations in 3NF.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CharId | CharName | ActorName | FigId | OwnerId | Pseudonym |
| Vader | Darth Vader | David Prowse | f14 | Bill | Jabba the Hoot |
| Vader | Darth Vader | David Prowse | f22 | Amy | Don’t Blame Me |
| Yoda | Yoda | Frank Oz | f16 | Lucy | Xena Warrior |
| Leia | Princess Leia | Carrie Fisher | f45 | Bill | Jabba the Hoot |
| Leia | Princess Leia | Carrie Fisher | f99 | Amy | Don’t Blame Me |