School of Information Technology & Computer Science
University of Wollongong

Session: **Spring 2020**
Lecturer: **Janusz R. Getta**

CSCI235 Database Systems
Project

Published on 21 November 2020

Scope

This project includes the tasks related to the database normalization, implementation of consistency constraints, implementation of derived attributes and programming database transaction, and design of BSON database.

The outcomes of the project are due by **Saturday, 28 November, 2020, 7.00pm (sharp).**

This project is worth 30% of the total evaluation in the subject.

A submission procedure is explained at the end of project specification.

This project consists of **4** tasks and specification of each tasks starts from a new page.

The outcomes of the project must be submitted through Moodle in the same way as all other coursework tasks in the subject. A submission link is available in a section THE FINAL PROJECT (Available from 21 November, 2020) just below a link to specification of the project.

A submission of the outcomes of the project marked by Moodle as "late" is treated as a late submission no matter how many seconds it is late. If you have poor Internet connection then allocate more time for a submission procedure. Please apply a principle saying that "*it is better to submit a project 1 hour too early than 1 second too late*".

A policy regarding late submissions is same as for late submissions of all coursework tasks in the subject and it is explained in the subject outline.

A submission of compressed files (zipped, gzipped, rared, tared, 7-zipped, lhzed, … etc) is not allowed. The compressed files will not be evaluated.

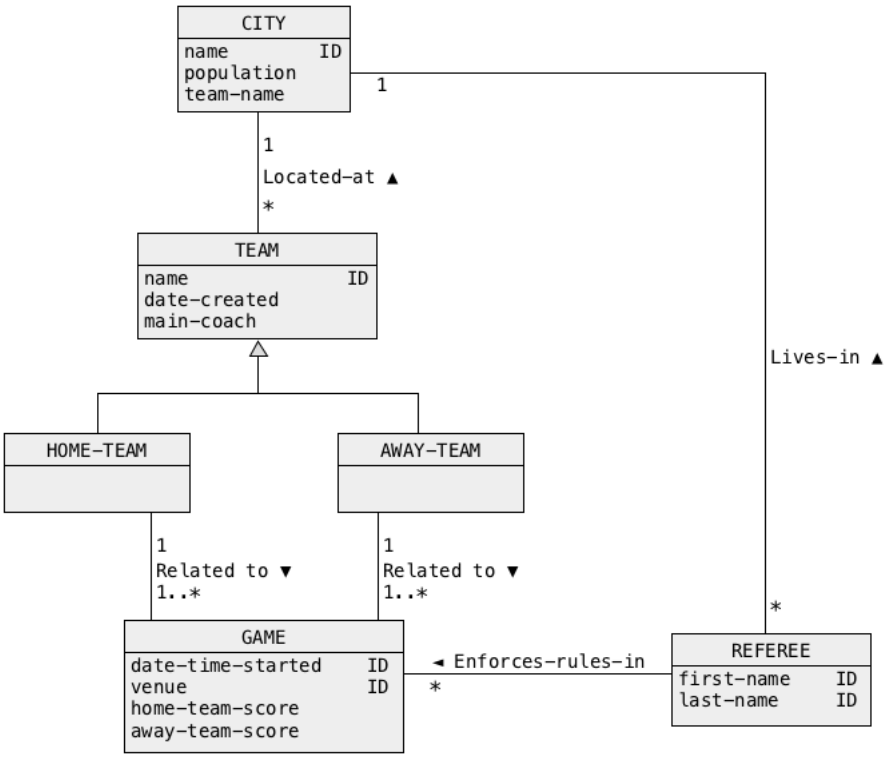
All files left on Moodle in a state "`Draft(not submitted)`" will not be evaluated.

It is expected that all tasks included within **Project** will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or office hours. Plagiarism will result in a **FAIL** grade being recorded for the assessment task.

Tasks
Step 1 (9 marks)
Improving a database design

This task requires access to either `csora` Oracle 19c database server, or one of `data-pc..` Oracle 19c database servers, or Oracle 19c running on a virtual machine.

Consider a conceptual schema given below. The schema describes a domain of a sample database where a number of football teams participate in a competition. Each team is identified by its name. The teams are located at the cities. Name of a city uniquely identifies each city. The teams play games in round-robin system, i.e. each team plays two games with all other teams. One game is played at home and one game is played away. A moment in time when a game starts and a venue of a game uniquely identifies each game. The referees enforce the rules during the games. To simplify a sample database domain, we assume that one game is referred by only one referee (like in U7 competitions). We do not have a lot of referee and because of that we assume that a pair of attributes: first name and last name uniquely identifies each referee.



SQl scripts `dbcreate.sql` and `dbdrop.sql` can be downloaded from Moodle. SQL script `dbcreate.sql` can be used to create the relational tables implementing a sample database. SQL script `dbdrop.sql` can be used to drop the relational tables. The files `dbcreate.lst` and `dbdrop.lst` contain the reports from processing of the scripts.

A database designer made few mistakes at the stages of conceptual modelling and logical design, i.e. during creation of conceptual schema and during transformation of conceptual schema into the relational schemas.

Your first subtask is to remove the mistakes made by a database designer. This subtask is worth 8 marks.

To do so, apply the following procedure for each relational schema.

(1) Use a short description of a database domain and information provided in a conceptual schema above and the contents of SQL script `dbcreate.sql` to discover nontrivial functional dependencies valid a relational schema.
(2) Use the functional dependencies found in the previous step to derive the minimal keys valid in a relational schema.
(3) Find the highest normal form valid in a relational schema.
(4) If a relational schema is not in BCNF then decompose it into the smallest number of relational schemas in BCNF.

And again, please be aware, that both original conceptual schema and the original relational schemas may contain few mistakes.

Save the nontrivial functional dependencies found, derivations of minimal keys, identifications of the normal forms and potential transformations of relational schemas in a file `solution1.pdf`. The file must contain all functional dependencies discovered, all derivations of minimal key and justifications for highest normal form valid in each schema.

Your second subtask is to improve a script `dbcreate.sql` after the modifications of the relational schemas and to insert some data into the relational tables. This subtask is worth 1 mark.

If you find that the transformations of the relational schemas into BCNF are needed then improve a script `dbcreate.sql` such that all relational tables created through processing of the script are in BCNF. Save the final relational schemas in a file `solution1.pdf`. Again, please keep in mind that an objective is to have the smallest number of relational tables in BCNF.

Append at the very end of a script `dbcreate.sql` SQL statements that insert data into the relational tables created in the previous step. Insert in to the database information about at least three cities, three teams, three games played, and two referees. Please do not underestimate this task. Insertion test may reveal design problems much faster than a formal approach based on analysis of functional dependencies.

Process an upgraded script `dbcreate.sql` and save a report from processing in a file `solution1.lst`.

Your report must include a listing of all SQL statements processed. To achieve that put the following SQLcl commands:

```
SPOOL solution1
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 100
SET PAGESIZE 200
```

at the beginning of SQL script and

```
SPOOL OFF
```

at the end of SQL script.

Deliverables
A file `solution1.pdf` with analysis of functional dependencies, derivations of minimal keys, identifications of the valid normal forms and transformations (if any) of the relational schemas into BCNF whenever it is necessary. A file `solution1.lst` with a report from processing of a script `dbcreate.sql`. A report must have no errors and it must list all SQL statements processed.

Step 2 (7 marks)
Automatic verification of a logical consistency constraint

This task requires access to either `csora` Oracle 19c database server, or one of `data-pc..` Oracle 19c database servers, or Oracle 19c running on a virtual machine.

A **logical consistency constraint** is the reflection of the real-world properties enforced on the contents of a database. `CREATE TABLE` statement can be used to enforce the following consistency constraints:
- primary key constraints,
- foreign key constraint,
- column type constraints,
- null/not null constraint,
- domain constraint.

Unfortunately, many of the logical consistency constraints cannot be enforced/verified through implementation of `CREATE TABLE` statements. In all such cases we have to use other programming techniques to do so. For example, we can enforce a more sophisticated logical consistency constraint through implementation of stored PL/SQL procedures/functions or through implementation of database triggers.

Your task is to implement automatic verification of a logical consistency constraint that cannot be enforced by `CREATE TABLE` statement.

Please use a sample database created in Task 1 for this task. A logical consistency constraint is up to you. It is very important, that your logical consistency constraint **CANNOT BE ENFORCED BY** `CREATE TABLE` statement. We shall not be able to grant you any marks for implementation of automatic verification of a logical consistency constraint, that can be enforced by `CREATE TABLE` statement.

Use can use either stored PL/SQL procedures/functions or database triggers to implement automatic verification of a logical consistency constraint.

Please consider verification of a logical consistency constraint when a new row is inserted into a relational table, when a row is updated, and when a row is deleted from a relational table.

And again, please make sure that a logical consistency constraint you implement cannot be enforced by `CREATE TABLE` statement.

Implement SQL script file `solution2.sql` that performs the following actions.

(1) First the script contains a comment that explains what logical consistency constraint is implemented. When processed with `SET ECHO ON` the script will automatically display the comment. For example, see a file `dbcreate.lst`.

(1) Next, the script implements verification of a logical consistency constraint using either stored PL/SQL procedures/functions or database triggers.

(2) Finally, the script comprehensively tests your implementation of automatic verification of a logical consistency constraint. Please, consider the cases when a row is inserted into a relational table, when a row is deleted from a relational table, and when a row is updated in a relational table.

When ready, process a script `solution2.sql` and save a report from processing in a file `solution2.lst`.

Your report must include a listing of all SQL and PL/SQL statements processed. To achieve that put the following SQLcl commands:

```
SPOOL solution2
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 100
SET PAGESIZE 200
SET SERVEROUTPUT ON
```

at the beginning of SQL script and

```
SPOOL OFF
```

at the end of SQL script.

Deliverables
A file `solution2.lst` with a report from processing of a script `solution2.sql`. A report must have no errors and it must list all SQL statements processed.

Task 3 (8 marks)
Implementation of a derived attribute and transaction processing

This task requires access to either `csora` Oracle 19c database server, or one of `data-pc..` Oracle 19c database servers, or Oracle 19c running on a virtual machine.

A **derived attribute** is an attribute in a relational table such that its values can be computed from the contents of one or more relational tables in the same database.

Your first subtask is to add a new derived attribute to a sample database and to implement a stored PL/SQL procedure that updates the values of the new derived attribute after modifications of sample database. The stored procedure must be implemented such that it may corrupt a database when processed at isolation level read committed. This subtask is worth 4 marks.

Implement SQL script file `solution3.sql` that performs the following actions.

(1) First, the script creates a derived attribute in a database obtained from the implementation of Task 1. The derived attribute can be appended to one of the relational tables or it can be located in a new relational table. The values of a derived attribute must be numerical. The values of derived attribute must be computed from the contents of a sample database

(2) Next, create a stored PL/SQL procedure that computes the values of the derived attribute from the present contents of a sample database. The stored procedure must be implemented such that it may corrupt a database when processed at **isolation level read committed**. Assume, that we use the scheduler implemented by Oracle 19c database server.

(3) Next, update a sample database such that a derived attribute must be recomputed after the update.

(4) Next, list the values of a derived attribute, process a stored procedure implemented in a step (2) above to update the values of a derived attribute, and list the values of a derived attribute after the updates.

When ready, process a file `solution3.sql` and save a report in a file `solution3.lst`.

Your report must include a listing of all SQL and PL/SQL statements processed. To achieve that put the following SQLcl commands:

```
SPOOL solution3
SET ECHO ON
SET FEEDBACK ON
SET LINESIZE 100
```

```
SET PAGESIZE 200
SET SERVEROUTPUT ON
```

at the beginning of SQL script and

```
SPOOL OFF
```

at the end of SQL script.

Your second subtask is to prove that a stored PL/SQL procedure implemented in the first task cannot be processed at an isolation level read committed. This subtask is worth 4 marks.

Create a document `solution3.pdf` with the explanations why a stored procedure implemented by you cannot be processed at an **isolation level read committed**. As an explanation provide a sample concurrent processing of two transactions both running at an **isolation level read committed.** One of the transactions processes the stored procedure implemented earlier to update the values of derived attribute. The other transaction is up to you. Both transactions are processed at an **isolation level read committed**. The results of the processing of the stored procedure corrupt one or more values of derived attribute.

Use a two-dimensional visualisation of concurrent execution of database transactions as it has been already used in the lectures slides and Assignment 2. Save your explanations and sample concurrent processing of database transactions that operate on a value of a **derived attribute** in a file `solution3.pdf`.

Deliverables
A file `solution3.lst` with a report from processing of a script `solution3.sql`. A report must have no errors and it must list all SQL statements processed.

A file `solution3.pdf` with the explanations why a stored procedure implemented by you cannot be processed at an **isolation level read committed** and with a sample concurrent execution of two transaction where one is processing the stored procedure and corrupting one or more values of a **derived attribute** at an **isolation level read committed**.

Task 4 (7 marks)
Design and implementation of BSON database

This task requires access to MongoDB 4.2 database server available on a virtual machine.

Consider the improved relational schemas obtained from implementation of Task 1.

An objective of the first subtask is to design a new collection of documents `games` that contains information included in a relational database implemented in Task 1. An important objective of the design is to maximize the size and complexity of the hierarchical structures in the collection and in the same moment eliminate any redundancies from the collection. This subtask is worth 4 marks.

First, create a logical schema of a new collection `games`. Use a graphical notation introduced in a presentation 22 BSON Design, slide 25, slide 31, and in a file `jsontpchr.bmp` available on Moodle. You can use UMLet 14.3 with `CSCI235-835Palette` to draw a logical schema. UMLet 14.3 is available on Moodle in a section `OTHER RESOURSEC`. A readable picture of a neat hand drawing is also acceptable.

Save your logical schema of a collection `games` in a file `solution4.pdf`.

An objective of the second subtask is to create a collection of documents `games` that use a JSON schema for validation of the inserted documents. This subtask is worth 3 marks.

Implement a script `solution4.js` that performs the following actions.

(1) Use a method `createCollection()` to create a collection of documents `games` and use JSON schema validator to enforce the constraints on the collection derived from the types of columns in the relational tables

(2) Next, insert into a collection `games` the same information as in Task 1 (Insert in to the database information about at least three cities, three teams, three games played, and two referees).

(3) Finally, insert a document that fails a validation of only one of the constraints listed above. Add a comment explaining why a document failed a validation.

When ready create a report from processing of `solution4.js` in the following way.

Use `gedit` editor to open a file `solution4.js`.

Select the entire contents of the file and Copy it into a buffer.

Open a new `Terminal` window and start mongo client in the following way.

```
mongo –port 4000
```

Paste the contents of the buffer copied earlier from `gedit` window in front of > prompt of `mongo` client. You may have to press Enter key to process the last query in a case when it is not followed by a newline control character.

Select the entire contents of the `Terminal` window and Copy&Paste it into a file `solution4.lst`. Save a file `solution4.lst`. Examine the contents of a file `solution4.lst` and make sure that it does not contain any errors.

Deliverables
A file `solution4.pdf` with a logical schema of a collection `games`.

A file `solution4.lst` with a report from processing of MongoDB script `solution4.js` with an implementation of the actions listed above.

And again, please remember that:
- a report without the listings of applied methods and feedback messages issued by MongoDB scores no marks,
- a report that contains any sort of processing errors except the failed validation scores no marks.

Submission

Submit the files **solution1.pdf**, **solution1.lst**, **solution2.lst**, **solution3.pdf**, **solution3.lst** and **solution4.pdf**, **solution4.lst** through Moodle in the following way:

(1) Access Moodle at **http://moodle.uowplatform.edu.au/**
(2) To login use a **Login** link located in the right upper corner the Web page or in the middle of the bottom of the Web page
(3) When logged select a site **CSCI835/CSCI235 (S220) Database Systems**
(4) Scroll down a bit to a section **THE FINAL ROJECT (Available from 21 November, 2020)**
(5) Click at a link **In this place you can submit the outcomes of Project**
(6) Click at a button **Add Submission**
(7) Move a file **solution1.pdf** into an area **You can drag and drop files here to add them**. You can also use the link **Add**...
(8) Repeat step (7) for the files **solution1.lst**, **solution2.lst**, **solution3.pdf**, **solution3.lst**, **solution4.pdf**, and **solution4.lst**.
(9) Click at a button **Save changes**
(10) Click at a button **Submit assignment**
(11) Click at the checkbox with a text attached: **By checking this box, I confirm that this submission is my own work,** ... in order to confirm the authorship of your submission.
(12) Click at a button **Continue**

End of specification