School of Information Technology & Computer Science
University of Wollongong

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

CSCI835 Database Systems
Final Project

Published on 24 July 2020

Scope

This project includes the tasks in database normalization, enforcing data consistency, transaction processing, and design and implementation of BSON database.

The outcomes of the project are due by **Friday, 31 July, 2020, 9.00 pm (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** steps and specification of each step 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 PROJECT (Available from 24 July, 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.

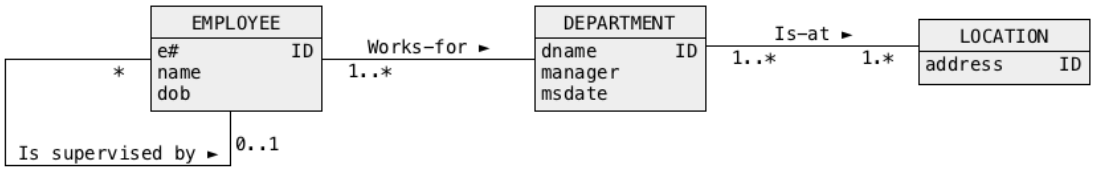
All files left on Moodle in a state "`Draft(not submitted)`" will not be evaluated.

It is expected that all tasks included within **Final 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 (10 marks)
Database normalization

This task requires access to Oracle 19c or 12c database server.

Consider the following conceptual schema of a sample database domain where the employees work at the departments distributed of over many locations The employees are supervised by other employees. Please see a conceptual schema below.



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.

Assume that a conceptual schema above is created in a correct way. Unfortunately, a database designer made few mistakes during a step of logical design, i.e. during a transformation of conceptual schemas into the relational schemas and the implementations of relational schemas in a script file `dbcreate.sql`.

Use information provided in a conceptual schema above and to discover the functional dependencies valid in each relational schema (header of a relational table). Next, use the functional dependencies found to derive the minimal keys and to find the highest normal form valid for each relational schema.

If you find that a relational schema is not in BCNF then transform the relational schema into BCNF. An objective is to minimize the total number of relational schemas in BCNF.

Save the 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.

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. 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

one department located at two addresses and two employees. One employee must supervise the other employee.

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 (6 marks)
Enforcing data consistency

This task requires access to Oracle 19c or 12c database server.

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

For example, an attribute `age` is a **derived attribute** because an `age` can be computed from a `date of birth` and the present date (`sysdate`).

Another example of **derived attribute** is a `total number of students` enrolled in each course offered by a university. A `total number of students` per course can be computed by grouping a relational table `ENROLLMENT` over a course code and by application of an aggregation function `COUNT()` to each group.

Yet another example of **derived attribute** is an attribute `income` computed by deduction of a tax amount from a taxable income.

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

(1) First, the script creates a **derived attribute** in one of the relational tables created and loaded with data in the previous step. A **derived attribute** is up to you.

(2) Next, the script sets the values of a **derived attribute** created in the previous step such that the values are consistent with the contents of the database.

(3) Next the script implements the mechanisms that keep the values of a **derived attribute** consistent with the contents of with the database. A value of **derived attribute** must be automatically changed whenever insertion/deletion/update operation is performed on the contents of the database. You can either use a technique of stored PL/SQL or data base triggers.

(4) Finally, the script comprehensively tests a solution implemented in step (3).

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
```

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.

Step 3 (8 marks)
Transaction processing

This task requires access to Oracle 19c or 12c database server.

Implement a stored PL/SQL procedure that increases one of the values of a **derived attribute** created in the previous step. The procedure must be implemented in a way that requires its execution at an **isolation level serializable**. When processed at an **isolation level read committed** the procedure would corrupt a value of a **derived attribute**.

Explain 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** and such that a value of a derived attribute is corrupted. 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`.

When ready, create a file `solution3.sql` and save implementation of a stored PL/SQL procedure created earlier in the file. Next, 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
```

at the beginning of SQL script and

```
SPOOL OFF
```

at the end of SQL script.

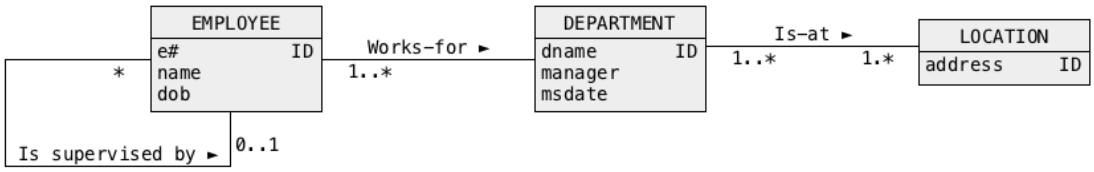
Deliverables
A file `solution3.lst` with a report from processing of a script `solution3.sql`. A report must have no errors and it must list all SQL statements processed.

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, each one implementing the stored procedure and corrupting a value of a **derived attribute** at an **isolation level read committed**.

Step 4 (6 marks)
Design and implementation of BSON database

This task requires access to MongoDB 4.2 database server.

Consider a conceptual schema given in Step 1 and below.



An objective of this task is to create a new collection of documents `company` that contains information represented by a conceptual schema given above and such that the collection is validated with JSON schema validator.

Use a method `createCollection()` to create a collection of documents `company` and use JSON schema validator to enforce appropriate nested structures of the documents to be included in the collection. All data types of valuates associated with the keys in the documents are up to you.

Next, insert into a collection `company` the same information as in Step 1 (information about one department located at two addresses and two employees; one employee must supervise the other employee). Next, insert a document that fails a validation of only one of the constraints listed above. Add a comment with information why a document fails a validation.

Implement the data manipulations listed above in a data manipulation language of MongoDB.

When ready create a report from processing of `createCollection()` method that creates a collection and assigns JSON schema validator to it, insertion of two documents that test validator and save it in file `solution1.lst`. To do so, use `gedit` editor and open a new file `solution1.lst`. Next, select the entire contents of the Terminal window and Copy&Paste it into a file `solution1.lst`. Save a file `solution1.lst`.

Deliverables
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 kind of processing errors except failed validation of the second document scores no marks.

Submission

Submit the files **solution1.pdf**, **solution1.lst**, **solution2.lst**, **solution3.pdf**, **solution3.lst** and **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 (JI20) Database Systems**
(4) Scroll down a bit to a section **PROJECT (Available from 24 July, 2020)**
(5) Click at a link **In this place you can submit the outcomes of the Final 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** 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