School of Computing and Information Technology
University of Wollongong

Session: **Autumn 2021**
Lecturer: **Janusz R. Getta**

CSCI235/CSCI835 Database Systems
Assignment 3
17 May 2020

Scope

This assignment includes the tasks related to design of a logical schema of MongoDB database, implementation of JSON Schema, implementation of data manipulations in MongoDB, and application of aggregation framework in MongoDB.

The outcomes of the assignment work are due by **Saturday 5 June, 2021, 7.00 pm (sharp).**

Please read very carefully information listed below.

This assignment contributes to 20% of the total evaluation in a subject CSCI235.

A submission procedure is explained at the end of specification.

This assignment consists of 3 tasks and specification of each task starts from a new page.

It is recommended to solve the problems before attending the laboratory classes in order to efficiently use supervised laboratory time.

A submission marked by Moodle as "late" is treated as a late submission no matter how many seconds it is late.

A policy regarding late submissions is included 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.

An implementation that does not compile due to one or more syntactical and/or run time errors scores no marks.

It is expected that all tasks included within **Assignment 3** 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.

Prologue
Install VirtualBox on your system, if it is not installed yet. If you do not remember how you did it in CSIT115 then it is explained in

https://documents.uow.edu.au/~jrg/115/cookbook/e1-1-frame.html

how to do it.

Download from Moodle `ova` image of a virtual machine with Ubuntu and MongoDB. The image is available in a section `OTHER RESOURCES`. You should get a file:

`Ubuntu18.04-64bits-MongoDB-4.2.2-08-JAN-2020.ova`

Start VirtualBox and import `ova` image of a virtual machine with Ubuntu and MongoDB. You should get a new virtual machine `Ubuntu18.04-64bits-MongoDB-4.2.2-08-JAN-2020.`

Start a virtual machine `Ubuntu18.04-64bits-MongoDB-4.2.2-08-JAN-2020.`

A password to login as `CSCI235` user is:

`csci235`

When logged in, start Terminal program (3rd icon from bottom in a column of icons on the left-hand size of a screen).

To start MongoDB server, process the following command in Terminal window.

```
mongod --dbpath DATA --port 4000
```

When MongoDB server is ready then among many, many, … the other messages you should get a message:

```
… waiting for connection on port 4000
```

Minimize Terminal window. Do not close the window, from now, it is used as a console window by MongoDB server.

Open another Terminal window and to start MongDB command line interface, process the following command.

```
mongo –port 4000
```

For a good start, process a command `help`.

Download to your virtual machine a file `customers.zip` from a section `SAMPLE DATABASES` on Moodle to the current folder from where you started `mongo` client.

Unzip a file `customers.zip` to the current folder.

You should get a file `customers.js`.

To create a collection `customers` and to load the documents into the collection, process a script `customers.js` at `>` prompt of `mongo` client in the following way.

```
load("customers.js");
```

A logical schema of a collection `customers` is given in a file `bsonschemap.bmp` located in a section `SAMPLE DATABASES` on Moodle.

Next, you can use the methods:

`db.customers.find().count()` and
`db.customers.find().pretty()`

to count the total number of the documents in a collection `orders` and to list all documents in a pretty format.

Next try few simple queries.

For example, to list in a nice format the contents of a document with an identifier `"CACTU"` process a method:

```
db.customers.find({"_id":"CACTU"}).pretty();
```

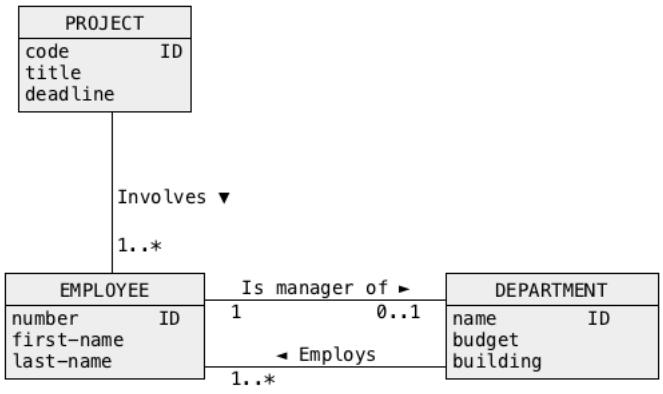
For example, to list information about the customers and the orders submitted by a customer who has a customer code `"FAMIA"` process a method:

```
db.customers.find({"CUSTOMER.customer code":"FAMIA"}).pretty();
```

No report is expected from implementation of the actions performed so far.

Task 1 (10 marks)
Logical design and implementation of BSON documents

Consider the following conceptual schema of a sample database, that contains information about employees working on projects, departments employing employees, and managers of departments.



Transform a conceptual schema given above into a logical schema of BSON documents. To draw a logical schema of BSON documents use a graphical notation presented in the lecture slides `22 BSON DESIGN`. You can use UMLet 14.3 to draw a logical schema. A link to UMlet 14.3 software is available on Moodle in a section OTHER RESOURCES. Use `CSCI235Palette`. Save a logical schema in a file `solution1.bmp`.

An important objective of the design is to maximize the size and complexity of hierarchical structures in a database and in the same moment to eliminate any redundancies from a database.

A correct transformation of a conceptual schema into a logical schema is worth 5 marks.

Next, use a diagram of a logical schema created in the previous step to implement JSON schema, that can be used to validate the documents, that contain information about projects, employees, and departments.

Download and unzip a file `solution1.zip`. You should get a file `solution1.js`. The file contains the comments with the specifications of the actions listed below. Insert into a file `solution1.js` implementation of the actions listed below.

(1) First, the script creates a collection `task1` using a method `db.createCollection()` with the JSON schema implemented in the previous step as a validator (see a presentation `21 Validation with JSON Schema`).

(2) Next, the script inserts into a collection `task1` the documents that contain information about one project, that involves two employees and about one department that employs the same two employees. One of the employees is a manager of a department. The documents must contain meaningful data and the types of values associated with the keys must be consistent with the meanings of the keys. For example, a value associated with a key `"date of birth"` must be of type date. All documents must pass the validation.

(3) Next, the script inserts one more document, that fails the validation against JSON schema used as a validator for a collection `task1`.

(4) Finally, the script prints the explanations on why one of the documents failed the validation. A simple way to print the explanation is to use `print("text')`.

A correct implementation of a script `solution1.js` is worth 5 marks.

To process a script `solution1.js` and to create a report `solution1.lst` from processing of a script, perform the following steps.

(1) Use `gedit` editor to open a file `solution1.js` with the implementations of the actions listed above.

(2) Select the entire contents of `gedit` window and Copy it into a buffer.

(3) Open a new Terminal window and start mongo client in the following way.

```
mongo –port 4000
```

(4) 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 data manipulation in a case when it is not followed by a newline control character.

(5) 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 `solution1.bmp` with a logical schema of BSON documents. A file `solution1.lst` with a report from processing of MongoDB script `solution1.js` that creates a collection task 1 with JSON validator and inserts the documents into the collection. Do not forget about the explanations why one of the documents failed the validation.

Please remember that:
- a report without listings of the processed methods scores no marks,
- a report that contains any kind of processing errors scores no marks.

Task 2 (5 marks)
Data manipulations

Download and unzip a file `solution2.zip`. You should get a file `solution2.js`. The file contains the specifications of the following 5 data manipulation operations on a collection `customers`.

(1) Change a location of a customer who has a customer code `FISSA` from Madrid to Barcelona. Next, display a customer code, city, region, and country of a customer who has a customer code `FISSA`.

(2) Rename a key "submits" to "orders" for all customers from Poland. Next, display customer code, country, and orders of a customers from Poland.

(3) Increase a value of freight by 10% in an order that has order id `274`. Next, display a customer code, and all orders submitted by a customer who submitted and order that has order is equal to `274`.

(4) Append to a customer who has a customer code equal to `LAUGB` the following information about a new order: `"order id" : 999,"freight" : 10.5`. Next list all information about a customer who has a customer code equal to `LAUGB`.

(5) Remove information about a contact title from the description of a customer who submitted an order with an order id `333`. Next list a complete description of a customer who submitted an order with an order id `333`.

Implement the data manipulations listed above in a data manipulation language of MongoDB. Write your solutions into the empty slots following a specification of each data manipulation in a file `solution2.js`. Do not remove the specifications of the data manipulations and semicolons following the specifications.

Implementation of each data manipulation is worth 1 mark.

When ready create a report from processing of the data manipulations in the following way.

Use `gedit` editor to open a file `solution2.js` with the specifications and implementations of the data manipulations.

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 data manipulation 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 `solution2.lst`. Save a file `solution2.lst`.

Deliverables
A file `solution2.lst` with a report from processing of MongoDB script `solution2.js` with the implementation of the data manipulations listed above.

And again, please remember that:
- a report without the specifications of the data manipulations and listings of the processed data manipulations scores no marks,
- a report that contains any kind of processing errors scores no marks.

Task 3 (5 marks)
Query processing and data transformation with aggregation framework

Drop a collection `customers` in the following way.

```
db.customers.drop();
```

To re-create a collection `customers` and to load the documents into the collection, process a scripts `customers.js`. at `>` prompt of `mongo` client in the following way.

```
load("customer.js");
```

Download and unzip a file `solution3.zip`. You should get a file `solution3.js`. The file contains the comments with the specifications of the following 5 queries and data transformations.

(1) Find the total number of orders shipped to the cities Aachen, or London or Madrid. Display the result in a format `{"total orders":integer-value}`.

(2) Find all orders shipped via Federal Shiping. Save the results into a collection `shippedvia` that consists of the documents like:
`{"order id": 111, "employee id": 1}`.
Display in a pretty format without document identifiers all documents in a collection `shippedvia`.

(3) Find the total number of order per each city where the customers are located at. List the results in a format:
```
{"total orders":integer-value,
 "city":city-name}.
```

(4) Save information about a customer code, city, and country of all customers from Aachen or London or Madrid into a collection `aaloma`. Display in a pretty format without document identifiers the contents of a collection `aaloma`.

(5) Find 5 largest values of freight from all orders. List the results in a format:
```
{"order id": integer-value,
 "freight":integer-value}.
```

Use the methods `aggregate()` and `pretty()` to implement all the queries and data transformations and to display the results. Note, that you may need two or more statements to implement a single task.

Implementation of each query/data transformation is worth 1 mark.

When ready create MongoDB script file `solution3.js` with the implementations of your queries and create a report from processing of the data manipulations in the following way.

Use `gedit` editor to open a file `solution3.js` with the specifications and implementations of the data manipulations.

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 data manipulation 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 `solution3.lst`. Save a file `solution3.lst`.

Deliverables
A file `solution3.lst` with a report from processing of MongoDB script `solution3.js` with the implementation of the data manipulations listed above.

Please remember that:
- a report without the specifications of the queries and data manipulations and listings of the processed queries and data manipulations scores no marks,
- a report that contains any kind of processing errors scores no marks.

Submission

Submit the files **solution1.pdf**, **solution1.lst**, **solution2.lst**, and **solution3.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 **CSCI235 (S121) Database Systems**
(4) Scroll down to a section **SUBMISSIONS**
(5) Click at a link **In this place you can submit the outcomes of Assignment 3**
(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 a link **Add**...
(8) Repeat a step (7) for the files **solution1.lst, solution2.lst**, and **solution3.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