School of Computing and Information Technology
University of Wollongong

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

CSCI235/CSCI835 Database Systems
Laboratory 6

19 October 2020

Scope

This laboratory includes two tasks related to the design and implementation of BSON documents and query processing in MongoDB database system.

The outcomes of the laboratory work are due by **Saturday 31 October 2020, 7.00 pm (sharp).**

Please read very carefully information listed below.

This laboratory contributes to 2% of the total evaluation in the subject.

A submission procedure is explained at the end of specification.

This laboratory consists of 2 tasks and specification of each task starts from a new page.

It is recommended to solve the problems before attending a laboratory class 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 errors scores no marks.

It is expected that all tasks included within **Laboratory 6** 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 1

Install VirtualBox on your system. 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
```

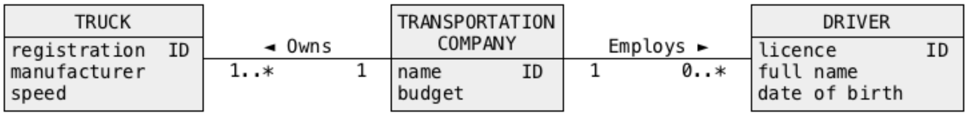
For a good start, process a command `help`.

Tasks

Task 1 (1 mark)

Logical design and implementation of BSON documents

Consider the following conceptual schema of a sample database that contains information about the transportation companies that own the trucks and employ the drivers.



Transform a conceptual schema given above into a logical schema of BSON document. To draw a logical schema of BSON document you can use a graphical notation presented in the lecture slides `22 BSON DESIGN`.

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.

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 transportation companies, drivers, and trucks.

Next, create a MongoDB script `solution1.js` that performs the following actions.

(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 contains information about one transportation company that owns two trucks and employs two employees. The documents must validate well against JSON schema used as a validator for a collection `task1`. 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.

(3) Next, the script inserts into a collection `task1` the documents that contains information about one transportation company that owns two trucks and employs two employees. One of the documents must fail validation against JSON schema used as a validator for a collection `task1`. 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.

(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')`.

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.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.

Prologue 2

Start MongoDB server in a way explained in **Prologue 1**.

Next, open a new Terminal window and use the following command to start a command line client `mongo`.

```
mongo -port 4000
```

Download to your virtual machine the files: `bsontpchr.bmp`, `customer.zip`, `part.zip`, and `supplier.zip` from a section `SAMPLE DATABASES` on Moodle.

Unzip the files: `customer.zip`, `part.zip`, and `supplier.zip`.

You should get the files: `customer.js`, `part.js`, and `supplier.js`.

To create a collection `tpchr` and to load the documents into the collection, process the scripts `customer.js`, `part.js`, and `supplier.js`. at > prompt of `mongo` client in the following way.

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

A logical schema of a collection `tpchr` is available in a file `bsontpchr.bmp`. It is strongly recommended to make yourself familiar with a logical schema of a sample database.

Next, you can use the methods

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

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

Next try few simple queries.

For example, to list information about a hierarchy of parts process a method:

```
db.tpchr.find({"PART":{$exists:true}}).pretty();
```

For example, to list information about a customer who has a customer key equal to `7` process a method:

```
db.tpchr.find({"CUSTOMER.customer key":7}).pretty();
```

For example, to list information about a customer who submitted an order that has an order key equal to `7` process the following method:

```
db.tpchr.find({"CUSTOMER.submits.ORDER.order key":7}).pretty();
```

For example, to list information about the parts of type `LARGE BRUSHED BRASS` process the following method.

```
db.tpchr.find({"PART.type":"LARGE BRUSHED BRASS"}).pretty();
```

No report is expected from implementation of the actions included in **Prologue 2** section.

Task 2 (1 mark)
Implementation of simple queries in MongoDB

Download and unzip a file `solution2.zip`. You should get a file `solution2.js`. The file contains the comments with the specifications of the following 5 queries.

Use the methods `find()` and `pretty()` to implement the following queries.

(1) Display in a pretty format information about the total number of customers who submitted empty orders, i.e. orders with no lines.

(2) Display in a pretty format information about the available quantities of parts (`availqty`), that have retail price greater than `908`. List only information about the available quantities, and retail prices.

(3) Display in a pretty format information about the customers from the nations of `JORDAN` or `MALAWI`. Do not list information about the submitted orders.

(4) Display in a pretty format information about the customers whose account balance (`acctbal`) is less then `122` and about the parts that have size less than `3`. In a relation to customers, list only information about the customer keys and account balances. In a relation to parts, list only information about part key and part size.

(5) Display in a pretty format information about the part keys and the supplier keys of all suppliers who supplied at least one part that has a retail price equal to `909` and a size equal to `12`.

Implement the queries in a query language of MongoDB, i.e. use the methods `find()` and `pretty()` to implement the queries. Write your solutions into a file `solution2.js` into the empty slots following a specification of each query. Do not remove the specifications of the queries and semicolons following the specifications !

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

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

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 `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 queries listed above.

And again, please remember that:

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

Submission

Note, that you have only one submission. So, make it absolutely sure that you submit correct files with the correct contents. No other submission is possible !

Submit the files **`solution1.lst`** and **`solution2.lst`** to 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 to a section **`Laboratory submissions`**
(5) Click at a link **`In this place you can submit the outcomes of Laboratory 6`**
(6) Click at a button **`Add Submission`**
(7) Move a file **`solution1.lst`** into an area **`You can drag and drop files here to add them`**. You can also use a link **`Add`**...
(8) Repeat step (7) for a file **`solution2.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`**

A policy regarding late submissions is included in the subject outline.

End of specification