School of Engineering and Computer Science

SWEN 432
Advanced Database Design and Implementation

Assignment 3

Due date: Monday 15 May at 11:59 pm

The objective of this assignment is to test your understanding of MongoDB data modeling and query language.

The Assignment is worth 5.0% of your final grade. The Assignment is marked out of 100.

When doing the assignment you will need to use MongoDB, the cloud database management system. There is a brief Instruction containing all needed commands given in an Appendix to this Assignment. Read it carefully before starting doing Assignment. In the course of answering assignment questions, you will need to issue command prompt instructions and to use a MongoDB shell of: a single sharded MongoDB installation. For deployment of the installations, a script is provided in the Instruction.

The assignment is mostly based on a database named Boat Hire, containing data about marinas, boats, sailors, and boat reservations made by sailors. Sailors may book only those boats in a marina that belong to the marina. Each marina has a unique name. Each boat has a unique number within its marina. Each sailor has a unique sailorId. A boat can have at most one reservation on a given day. A sailor can make at most one reservation on a given day. The database should initially contain data given in a sequence of tables on the next pages under the name “Assignment 3 Data”. In the MongoDB environment, the Boat Hire data base can be implemented either as a collection containing embedded documents, or a set of collections containing documents that rely on referencing.

There are two parts to this assignment. Both are about using `insert()`, `update()`, `find()`, and single purpose aggregations (`count()` and `distinct()`) in the mongo shell of a single sharded deployment. If you think there is an advantage of using single purpose aggregations (`count()` and `distinct()`) over using the `find()` method, use single purpose aggregations

Model Solution

and justify your decision. You are not expected to use (pipeline) `aggregate()` methods in this assignment. The documents in the first and second part contain the same data. Only, in the first one, documents are implemented using embedding, while the documents of the second one use referencing.

Assignment 3 Data

Marina

name	location
Sea View	Petone
Port Nicholson	Wellington
Evans Bay	Wellington

Boats in Sea View marina

name	number	color	driven_by
Flying Dutch	313	blue	sail
Blue Shark	515	yellow	motor
Killer Whale	111	black	row
Dolphin	110	blue	sail, motor

Boats in Port Nicholson marina

name	number	color	driven_by
Red Cod	616	yellow	sail, motor
Tarakihi	717	red	row, motor
Night Breeze	818	black	row
Mermaid	919	white	sail, motor
Dolphin	110	white	
Pretty Lady	515	pink	sail

Boats in Evans Bay marina

Name	number	color	driven_by
Sally Ann	313	white	motor
Charlie	515	blue	sail, motor

Sailors

name	sailorId	skills	address
James	707	row, sail, motor, fish	Wellington
Peter	111	row, sail, motor	Upper Hutt
Milan	818	row, sail, motor, first aid	Wellington
Eileen	919	sail, motor, swim	Lower Hutt
Paul	110	row, swim	Upper Hutt
Charmain	999	row	Upper Hutt
Gwendolyn	777	row, sail, motor, dance	Masterton

Reserves

marina	boat	sailor	res_date
Sea View	313	707	2017-03-15
Sea View	313	111	2017-03-16
Sea View	111	111	2017-03-15
Sea View	110	818	2017-03-15
Sea View	515	919	2017-03-15
Port Nicholson	919	818	2017-03-16
Port Nicholson	919	707	2017-03-17
Port Nicholson	515	111	2017-03-17
Port Nicholson	616	707	2017-03-21
Evans Bay	313	818	2017-03-19
Evans Bay	515	818	2017-03-21
Port Nicholson	818	999	2017-03-20
Port Nicholson	818	999	2017-03-21
Port Nicholson	717	919	2017-03-25
Port Nicholson	717	777	2017-03-28

Important

1. In your answers to the assignment questions, show the commands and methods you issued and used and the answers produced by MongoDB on the standard output.
2. Do not delete your MongoDB collections after finishing the assignment. You will need them in Assignment 4.

Part I

Question 1. Making your `reserves` Collection [32 marks]

To spare you from doing a tedious and repetitive job, Pavle made his `reserves` MongoDB collection and exported it into the file

`reserves_17.txt`.

You will find the file `reserves_17.txt` on the course `Assignments` page. Start a single sharded MongoDB deployment and import the file `reserves_17.txt`. (Import command is given in the Appendix.) Call your collection `reserves`. This is a must. (Note: The `reserves_17.txt` may contain some valid documents that are not shown in Assignment 3 Data.)

Soon after exporting the file `reserves_17.txt`, Pavle realized that it contained a number of errors and that it was not complete. In this question, you need to tidy and complete your `reserves` collection.

a) [2 marks] The name of the Port Nicholson marina is misspelled in a number of ways. Use `multi` option of the `db.collection.update()` method to bring it in order.

ANSWER

```
> db.reserves.update({'marina.name': {$regex: /^Port N/}},
{$set: {'marina.name': "Port Nicholson"}}, {multi: true})
WriteResult({ "nMatched" : 7, "nUpserted" : 0, "nModified"
: 6 })
```

b) [4 marks] In the document `"_id" :`

`ObjectId("54f102de0b54b61a031776ed"),`

the field `number` is misspelled as `numbver`. Rename it.

ANSWER

```
> db.reserves.update({'reserves.boat.numbver': {$exists:
true}}, {$rename: {"reserves.boat.numbver":
"reserves.boat.number"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified"
: 1 })
```

c) [4 marks] A document for the row

Port Nicholson	717	919	2017-03-25

of the Reserves table is missing. Insert it.

ANSWER

```
> var reserves_17 = {marina: {name: "Port Nicholson",
  location: "Wellington"}, reserves: {boat: {name:
  "Tarakihi", number: 717, color: "red", driven_by: ["row",
  "motor"]}, sailor: {name: "Eileen", sailorId: 919, skills:
  ["sail", "motor", "swim"], address: "Lower Hutt"},
  res_date: "2017-03-25" }}

> db.reserves.insert(reserves_17)
WriteResult({ "nInserted" : 1 })
```

d) [5 marks] We need all boats to be in our database, but the boat Dolphin, number 110 from Port Nicholson marina had no reserves yet and its data are missing. Make a document containing Dolphin's data and store it in your collection. The document should follow the structure of other documents in the `reserves` collection to the highest possible (and reasonable) extent. Don't introduce fake data.

Model Solution

ANSWER

```
var reserves_15 = {marina: {name: "Port Nicholson",
location: "Wellington"},
reserves: {boat: {name: "Dolphin", number: 110, color:
"white"}}}
> db.reserves.insert(reserves_15)
WriteResult({ "nInserted" : 1 })
```

e) [5 marks] We need all sailors to be in our database, but the sailor Paul from Upper Hutt did not make any reserves yet and his data are missing. Make a document containing Paul's data and store it in your collection. The document should follow the structure of other documents in the `reserves` collection to the highest possible (and reasonable) extent. Don't introduce fake data.

ANSWER

```
var reserves_17 = {reserves: {sailor: {name: "Paul",
sailorId: 110, skills: ["row", "swim"], address: "Upper
Hutt"}}}
> db.reserves.insert(reserves_17)
WriteResult({ "nInserted" : 1 })
```

f) **[12 marks]** Pavle also realized that he missed to define the following unique constraints:

i. A sailor can make at most one reservation on a given day. [4 marks]

ANSWER

```
> db.reserves.ensureIndex({'reserves.sailor.sailorId': 1,
'reserves.res_date': 1}, {unique: true})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
```

ii. A boat can have at most one reservation on a given day. [4 marks]

ANSWER

```
> db.reserves.ensureIndex({'marina.name': 1,
'reserves.boat.number': 1, 'reserves.res_date': 1},
{unique: true})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
```

iii. Check whether your indexes perform as expected. In your answer, show how did you perform checking, and what messages you received.

[4 marks]

ANSWER

```
> var reserves_18 = {marina: {name: "Port Nicholson",
location: "Wellington"},
... reserves: {boat: {name: "Tarakihi", number: 717, color:
"red", driven_by: ["row", "motor"]},
... sailor: {name: "Paul", sailorId: 110, skills: ["row"],
address: "Upper Hutt"}, res_date: "2017-03-25" }}
> db.reserves.insert(reserves_18)
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 11000,
                "errmsg" : "insertDocument :: caused by ::
11000 E11000 duplicate key error index:
ass2.reserves.$marina.name_1_reserves.boat.number_1_reserve
s.res_date_1  dup key: { : \"Port Nicholson\", : 717.0, :
\"2017-03-25\" }"
        }
})}

> var reserves_19 = {marina: {name: "Port Nicholson",
location: "Wellington"},
... reserves: {boat: {name: "Dolphin", number: 110, color:
"white"},
... sailor: {name: "Eileen", sailorId: 919, skills:
["sail", "motor", "swim"], address: "Lower Hutt"},
res_date: "2017-03-25" }}
> db.reserves.insert(reserves_19)
```

Model Solution

```
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 11000,
                "errmsg" : "insertDocument :: caused by ::
11000 E11000 duplicate key error index:
ass2.reserves.$reserves.sailor.sailorId_1_reserves.res_date
_1  dup key: { : 919.0, : \"2017-03-26\" }"
        }
})
```

Question 2. Simple Queries [26 marks]

a) [2 mark] Find the number of all documents in your `reserves` collection.

ANSWER

```
> db.reserves.count()
17
```

b) [4 marks] Find the number of all documents in your `reserves` collection containing valid reserves made in Port Nicholson marina.

ANSWER

```
>   db.reserves.count({'marina.name':   "Port   Nicholson",
'reserves.res_date': {$exists: true}})
8
```

c) [4 mark] Find unique sailor names.

ANSWER

```
> db.reserves.distinct('reserves.sailor.name')
[ "James", "Peter", "Milan", "Eileen", "Charmaine",
  "Gwendolyn", "Paul" ]
```

Model Solution

d) [5 marks] Find marina names, boat names, and sailor names having a reservation on “2017-03-16”.

ANSWER

```
> db.reserves.find({'reserves.res_date': "2017-03-16"},
{_id: 0, 'marina.name': 1, 'reserves.boat.name': 1,
'reserves.sailor.name': 1})

{ "marina" : { "name" : "Sea View" }, "reserves" : {
"boat" : { "name" : "Flying Dutch" },
"sailor" : { "name" : "Peter" } } }
{ "marina" : { "name" : "Port Nicholson" }, "reserves" : {
"boat" : { "name" : "Mermaid" },
"sailor" : { "name" : "Milan" } } }
```

e) [5 marks] Find sailors having swim skill. Display just sailor names.

ANSWER

```
db.reserves.distinct('reserves.sailor.name',
  {'reserves.sailor.skills': "swim"})
[ "Eileen", "Paul" ]

> db.reserves.find({'reserves.sailor.skills': "swim"},
{'reserves.sailor.name': 1, _id: 0})

{ "reserves" : { "sailor" : { "name" : "Eileen" } } }
{ "reserves" : { "sailor" : { "name" : "Paul" } } }
{ "reserves" : { "sailor" : { "name" : "Eileen" } } }
```

Model Solution

f) [6 marks] Find sailors having exactly row, sail, and motor skills (no more and no less, but in an arbitrary order). Display just sailor names.

ANSWER

```
> db.reserves.distinct('reserves.sailor.name',
  {"reserves.sailor.skills": {$all: ["row", "motor",
  "sail"], $size: 3 } })
[ "Peter" ]

> db.reserves.find({'reserves.sailor.skills': ["row",
  "sail", "motor"]}, {_id:0, 'reserves.sailor.name': 1})
{ "reserves" : { "sailor" : { "name" : "Peter" } } }
{ "reserves" : { "sailor" : { "name" : "Peter" } } }
{ "reserves" : { "sailor" : { "name" : "Peter" } } }
```

Question 3. Time Table Collection [10 marks]

Assume Wellington Tranz Metro has acquired an iPhone application that works as a data recorder for railway vehicles (cars, engines). The application uses mobile data connections to send information to servers in real time. The information includes measurements such as the location and speed of the vehicle. They have selected MongoDB (instead of Cassandra, as in Assignment 1) as the CDBMS to use for this project. They invited you, as a respected database specialist to design an embedded collection of the database.

a) [6 marks] Assume your document will contain the following entity types:

- driver,
- vehicle,
- time table, and
- data point.

Design a MongoDB document by representing relationships between entity types by embedding. Make relationships between entity types clearly visible in your design. Use iPhon database sample data of Assignment 1 to populate your document. Show your design in your answer.

ANSWER

```
{
 _id: {line_name: "Melling Line", service_no: 3, res_date: "2017-05-21"}
 res_date: "2017-05-21",
 line_name: "Melling Line",
 service_no: 3,
 stops:
 [{ stop_name: "Wellington", latitude: -41.2865, longitude: 174.7762, time: 741, distance: 0},
 { stop_name: "Western Hutt", latitude: -41.2118, longitude: 174.89, time: 801, distance: 11.4},
 { stop_name: "Melling", latitude: -41.2036, longitude: 174.9054, time: 807, distance: 13.7}
 ],
 driver: { driver_name: "fred",
           current_position: "FA1122",
           email: "fred@ecs.vuw.ac.nz",
           mobile: 2799797,
           password:  "f00f",
           skill: ["Ganz Mavag", "Guliver"]
          },
 vehicle: { vehicle_id: "FA1122", status: "in use", type: "Ganz Mavag" },
 data_point:
[{sequence: "2017-03-25 07:39:40+0000", latitude: -41.2865, longitude: 174.7762, speed: 0},
 {sequence: "2017-03-25 07:45:10+0000", latitude: -41.2312, longitude: 174.717, speed: 78.5},
 {sequence: "2017-03-25 07:48:10+0000", latitude: -41.2199, longitude: 174.7551, speed: 18.8}
 ]
 }
```

b) [4 marks] How many instances of each entity type may contain your document maximally? How many documents will be produced per day? Make a best guess if you can't give an exact number.

ANSWER

Each document contains:

- One instance of the time table service entity type
- One instance of the driver entity type
- One instance of the vehicle type
- Several instances of the data point type

An instance of the service type contains a number of stations (up to a few tens)

Each document contains several hundreds of instances of the data type (for a service taking an hour from departure to destination, say 600 instances)

There are as many documents per a day as there are services of all lines (assuming documents are not deleted).

Part II

Pavle implemented the referencing version of the Boat Hire database as four collections and exported them into files:

- `marina_17.txt`,
- `sailor_17.txt`,
- `boat_17.txt`,
- `res_ref_17.txt`.

You will find these files at the course `Assignments` page. Start a single sharded MongoDB deployment and import the files. You may do import into the same database as for `reserves` collection. Call the new collections: `marina`, `sailor`, `boat`, and `res_ref`. This is a must. So far, Pavle did not find any errors in the implementation. Use these collections to answer the following questions.

Question 4. Simple Queries [8 marks]

a) [4 marks] Find all unique sailors.

ANSWER

```
db.sailor.find({})
{ "_id" : ObjectId("54f92b017b2b4c977e827da8"), "name" :
"James", "sailorId" : "707", "skills" : [ "row", "sail",
"motor", "fish" ], "address" : "Wellington" }
{ "_id" : ObjectId("54f92b2c7b2b4c977e827da9"), "name" :
"Peter", "sailorId" : "111", "skills" : [ "row", "sail",
"motor" ], "address" : "Upper Hutt" }
{ "_id" : ObjectId("54f92bdd7b2b4c977e827daa"), "name" :
"Milan", "sailorId" : "818", "skills" : [ "row", "sail",
"motor", "first aid" ], "address" : "Wellington" }
{ "_id" : ObjectId("54f92bff7b2b4c977e827dab"), "name" :
"Eileen", "sailorId" : "919", "skills" : [ "sail", "motor",
"swim" ], "address" : "Lower Hutt" }
{ "_id" : ObjectId("54f92cac7b2b4c977e827dac"), "name" :
"Paul", "sailorId" : "110", "skills" : [ "row", "swim" ],
"address" : "Upper Hutt" }
{ "_id" : ObjectId("54f92cdf7b2b4c977e827dad"), "name" :
"Charmain", "sailorId" : "999", "skills" : [ "row" ],
"address" : "Upper Hutt" }
{ "_id" : ObjectId("56g92edf7b2b4c977e827dad"), "name" :
"Gwendolynn", "sailorId" : "777", "skills" : [ "row, sail,
motor, dance" ], "address" : "Masterton" }
```

Model Solution

b) [4 marks] Find sailors having exactly row, sail, and motor skills (no more and no less). Display just sailor names.

ANSWER

```
> db.sailor.find({skills: ["row", "sail", "motor"]},
  {_id:0, name: 1})
{ "name" : "Peter" }
```

Question 5. Complex Queries [24 marks]

a) [8 marks] Find marina names, boat names, and sailor names having a reservation on "2017-03-16". Use manual references.

ANSWER

The following queries return the data requested:

```
> db.res_ref.find({'reserves.res_date': "2017-03-16"},
{_id: 0, marina: 1, 'reserves.boat': 1, 'reserves.sailor':
1})
{ "marina" : "Sea View", "reserves" : { "boat" : 313,
"sailor" : 111 } }
{ "marina" : "Port Nickolson", "reserves" : { "boat" : 919,
"sailor" : 818 } }
> db.boat.find({marina : "Sea View", number: 313},
{_id: 0, name: 1})
{ "name" : "Flying Dutch" }
> db.sailor.find({sailorId: 111}, {_id: 0, name: 1})
{ "name" : "Peter" }
> db.boat.find({marina : "Port Nicholson", number: 919},
{_id: 0, name: 1})
{ "name" : "Mermaid" }
> db.sailor.find({sailorId: 818}, {_id: 0, name: 1})
{ "name" : "Milan" }
```

Model Solution

b) [16 marks] Find marina names, boat names, and sailor names having a reservation on "2017-03-16" using not more than two commands in `mongo` shell.

ANSWER

```
> var curs = db.res_ref.find({"reserves.res_date" : "2017-
  03-16"});
> while (curs.hasNext()) { res = curs.next();
..sailor = db.sailor.findOne({"sailorId":
  res.reserves.sailor});
. boat = db.boat.findOne({"marina" : res.marina, "number":
  res.reserves.boat});
..ret = {"marina": res.marina, "boat": boat.name, "sailor":
  sailor.name};
..print(tojson(ret));
..}
{ "marina" : "Sea View", "boat" : "Flying Dutch", "sailor"
  : "Peter" }
{ "marina" : "Port Nicholson", "boat" : "Mermaid", "sailor"
: "Milan" }
```

Submission Instruction:

Submit your answers to assignment questions via the school electronic submission system and hand-in a printed version in the hand-in box on the second floor of the Cotton Building.

Please do not submit any .odt, .zip, or similar files. Also, do not submit your files in toll directory trees. All files in the same directory is just fine.

Additionally, submit your commands for questions: Q2, Q4, and Q5 as separate .txt files only electronically (Pavle is going to run these commands).

Model Solution

A Short Instruction for Using MongoDB on ECS Workstations

1. MongoDB Scripts

Before you try to use MongoDB on our school workstations, you have to type

```
[~] % need mongodb
```

This command will allow all what is needed to deploy MongoDB configurations. You may want to insert `need mongodb` in your `.cshrc` file to avoid typing `need mongodb` whenever you log_on.

Our programmer Royce Brown produced the following four scripts for deploying different MongoDB configurations:

- `[~] % single-mongo,`
- `[~] % rep-mongo,`
- `[~] % sha-mongo,`
- `[~] % sharep-mongo.`

You can run these scripts at the command line prompt of your home directory. Just run them without any parameters to see what each one does.

In Assignemnt3_17 you will use `single-mongo.` In Assignment4_17 you will use `single-mongo, sha-mongo,` and `sharep-mongo.`

After starting a MongoDB configuration (e.g. `single-mongo start`), you can:

- Import a collection from a file into a database by typing:

[~] % `mongoimport --db <database_name> --collection <collection_name> --file <file_name>`

Connect to a mongo shell by typing:

[~] % `mongo`

```
>
```

2. Useful `mongo` shell commands

To get help:

```
> help
```

To see existing databases, while being in a `mongo(s)` shell:

```
> show dbs
```

To use an existing database, or to define a new one:

```
> use <database_name>
```

To see collections in the current database:

```
> show collections
```

To exit from a `mongo(s)` shell:

CTRL/d

Note: The default database is `test`. If you do not issue a `use <database_name>`, all your commands are going to be executed against the `test` database.

Warning:

- In all deployments the same ports are assigned to servers. After finishing a session you have to **stop** all servers of your deployment to release ports for other uses. Failing to do so, you will make trouble to other people (potentially including yourself) wanting to use the same workstation. Later, if you want to use the same deployment again, you just do `<script_name> start` and your deployment will resume functioning reliably. If you don't plan to use a deployment again, don't forget to do `<script_name> cleanall`.
- You are strongly advised to use MongoDB from school lab workstations. The school does not undertake any guarantees for using MongoDB from school servers. You may install and use MongoDB on your laptop, but the school does not undertake any responsibilities for the results you obtain.