Submitted By: Angeline Chrisette C. Olegario
Submission Date:

1.2 Tasks

1. Create a database representing the above problem. Your database can contain as many collections as you think necessary.

In your report, briefly describe what mechanisms you used to represent the different facts and information about the company, e.g. justifying why your chose embedding or referencing. Submit a file named create.js along your report containing the mongo shell instructions necessary to create and populate your database.

For my MongoDB schema,I patterned it to the Normalization of Assignment 2 which was through SQL and converted it into NOSQL.

I see that choosing between Embedding and Referencing, it is more of an art than science & in my database tapestry, I implemented Embedding.

In all honesty, I tried both. I have 2 prototypes of me implementing both through Embedding and Referencing. At first I was very pursuant to do Referencing, as my head has always been used to RDBMS and SQL, and that's where I realized that there is a reason that NoSQL was created, not as another platform to make RDBMS but it's a whole new world in itself.

When I realized things were getting complicated for query task 2, I didn't want to do pipelines because why should getting the manager name be so complicated. I assessed my code once more, and I went back to embedding.

I chose embedding because I thought of all of the queries that this assignment needed me to do, and embedding provided the easiest way to project those values without the need of complicated joins. And from there, I realized the beauty of NoSQL, queries were extremely easy to implement.and I am afraid I might not go back to SQL after this.

Sidenote: I actually used your uploaded Assignment 2 Solution as reference to the values you expected to see. hehe

I used MongoDB and Visual Code and tested the codes via Playground and checking if the components were really updated in my cloud.mongodb.com account

----

Database Schema:

Database: employeeRecordsDB
Collections:
- Employee

Connection Instructions

I implemented my MongoDB using Visual Studio Code. The details that I've used to connect it are the following:

MongoDB

Username: admin-mis
Password: mm9G9GpKrbVuaW3k

[PRIMARY] Connection String via Node.js 3.6 or later:
mongodb+srv://admin-mis:mm9G9GpKrbVuaW3k@acolegario-mis.rrg0q.mongodb.net/test

mongodb+srv://admin-mis:mm9G9GpKrbVuaW3k@acolegario-mis.rrg0q.mongodb.net/employeeRecordsDB?retryWrites=true&w=majority

Connection via Mongo Shell:
mongo "mongodb+srv://acolegario-mis.rrg0q.mongodb.net/employeeRecordsDB" --username admin-mis

Create.js

```
use('employeeRecordsDB');

db.employee.drop(); //To Remove if exists

db.employee.insertMany([
   { '_id' : 1, 'employeeName' : 'Deneen Willmon', 'employeeEmail' :
'deneen.willmon@gmail.com', 'employeePhone' : '09087878889',
   worksFor : {
      'departmentName': 'Maintenance',
      'role' : 'Engineer',
   }},
   { '_id' : 2, 'employeeName' : 'Lashay Dann', 'employeeEmail' : 'lashay.dann@gmail.com',
'employeePhone' : '08041971660',
   worksFor : {
      'departmentName': 'Accounting',
      'role' : 'Head Accountant',
    }},
    { '_id' : 3, 'employeeName' : 'Kallie Jolliff', 'employeeEmail' : 'kallie@ymail.com',
'employeePhone' : '09171287291',
    worksFor : {
        'departmentName': 'Maintenance',
        'role' : 'Technician',
    },
    manages : [
        { "_id": 1, "Name": "Deneen Willmon" }
    ]},
    { '_id' : 4, 'employeeName' : 'Starla Priebe', 'employeeEmail' : 'starlap@yahoo.com',
'employeePhone' : '06503382292',
    worksFor : {
        'departmentName': 'Development',
        'role' : 'Coder',
    },
    manages : [
        { "_id": 1, "Name": "Deneen Willmon" }
    ]},

    { '_id' : 5, 'employeeName' : 'Jannette Basnight', 'employeeEmail' :
'jannette012@hotmail.com', 'employeePhone' : '07784902817',
    worksFor : {
        'departmentName': 'Development',
    }},
]);
```

2. Write queries to obtain the following information:

(a) List of employees and the department(s) they work for

(b) List of employees and their manager (show the employee even if they do not have a manager)

(c) List of employees having an ”a” in their name

In your report, briefly describe how you built the queries.Submit a file named queries.js along your report containing the mongo shell queries that give the above Information.

Since I used embedding, it was extremely easy to do the following queries, starting with:

```
//List of employees and the department(s) they work for
db.employee.aggregate(
   [
      {"$unwind" : "$worksFor"},
      {"$project" : {
          _id: 0,
          "Employee ID" : "$_id",
          "Employee Name" : "$employeeName",
          "Department" : "$worksFor.departmentName"
                 }
      }
   ]
).pretty();
```

All I had to do was unwind the worksFor object so that I can get access to the departmentName and then I projected the appropriate values needed.

```
//List of employees and their manager (show the employee even if
//they do not have a manager)
db.employee.aggregate(
   [
    {
      $unwind: {
        path: "$manages",
        preserveNullAndEmptyArrays: true
      }
    },
      {"$project" : {
          _id: 0,
                "Employee ID" : "$_id",
                "Employee Name" : "$employeeName",
                "Manager" : "$manages.Name"
                        }
            }
        ]
    ).pretty();
```

This query is one of the turning points that made me change to embedding. It was incredibly difficult to do this through my previous implementation, but as I have thought, this is NOSQL it shouldn’t be so complicated. So all I did was unwind the manages tuple that holds the Manager ID and Manager Name values then I projected them to show the necessary data.

```
//List of employees having an ”a” in their name
db.employee.aggregate([
  {
    $match: {
      "employeeName": /a/
    }
  },
  {
    $project: {
          _id: 0,
          "Employee Name " : "$employeeName"
    }
  }
]).pretty();
```

I used the match function and then projected the matched employee names.

3. Update your database to account for the following facts:

(a) Deneen has married (congrats!) and has taken a new name: Crawley

(b) People will soon be quitting the firm or retiring so we want to add one piece

of information per employee: whether they are a current employee (true) or not (false). All employees are active for the moment.

(c) Jannette has retired

In your report, briefly describe how you translated these different facts into the database and mongo shell code if an update is required. Submit a file named up-date.js along your report containing the mongo shell code necessary to update the database.

```
// Deneen has married (congrats!) and has taken a new name: Crawley

db.employee.update(
   { "_id": 1  },
   { $set: { "employeeName": "Deneen Crawley" } }
)

db.employee.updateMany(
    { "manages._id": 1 },
    { "$set": { "manages.$.Name": "Deneen Crawley" } }
)
```

What I did was, look for the ID then used the $set function to set Deneen's new surname. I almost forgot to update the other parts where Deneen Crawley Existed (The manages attribute); It was fairly tricky to update an embedded array.

```
//People will soon be quitting the firm or retiring so we want to add one piece
//of information per employee: whether they are a current employee (true) or not
//(false). All employees are active for the moment.

db.employee.updateMany({}, {$set:{"is_active": true}})
```

Here I had a bit of a hard time because I just used .update during my first try, but then I used updateMany and it worked. This was the trickiest of them all.

```
//Jannette has retired
db.employee.update(
   { "_id": 5  },
   { $set: { "is_active": false } }
)
```

Here, I just replicated what I did for when Deneen changed her surname. So it was quite easy.