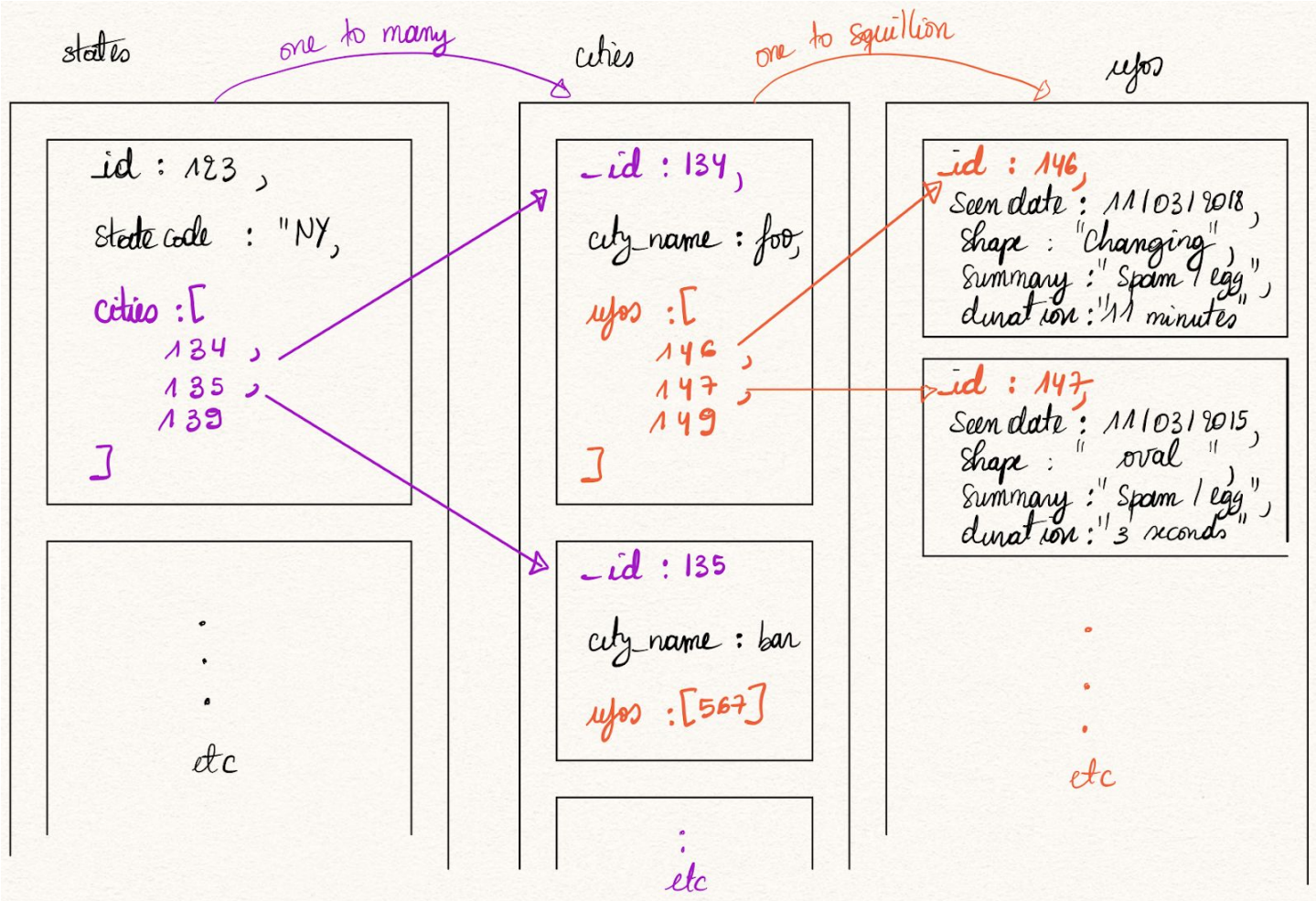
Tatiana Zlhindula
C16339923
Year 3: NoSQL Assignment with MongoDB
Dataset: UFO sightings
14/12/2018

Schema

1. design

Alternatively the arrows could have went backward. keep every state code in the city and every city with the UFO collection.. but what if two cities now have the same name? return all cities in all state matching a city?



2. Schema code Implementation : **index.js**
   a. I use **mongoose** (javascript Node.js library) to model the collections above into concrete collection.
   b. I then **parsed the csv file** using the **fs** and **csv-parser** using Node.js libraries
   c. I proceeded to using **mongoose** to insert the parsed data in bulk. (see index.js) for explanation in comments.

3. queries : **queries.js**

   a. **state name and their city count**

```
//1. get all state names and the number of their cities sorted in ascending order of redistred city count
db.states.aggregate([
   {
      $project: {
         _id: 0,
         state_code:1,
         number_of_cities:{ $size: "$cities" }
      }
   },
   {$sort : { 'number_of_cities' : -1}}
])
```

Aggregate | Aggregate | Aggregate | Aggregate | Aggregate

50 Documents 1 to 50

```
{
    "state_code" : "CA",
    "number_of_cities" : NumberInt(71)
}
{
    "state_code" : "FL",
    "number_of_cities" : NumberInt(51)
}
{
    "state_code" : "NY",
    "number_of_cities" : NumberInt(47)
}
{
    "state_code" : "NONE",
    "number_of_cities" : NumberInt(41)
}
{
    "state_code" : "WA",
    "number_of_cities" : NumberInt(37)
```

2. Cities outside of the USA and their UFO counts

```
//2.  get the name of cities outside of the US and their UFO counts
db.cities.aggregate([
     {
       $project:
         {
           _id:"$_id",
           _id: 0,
           ufo_count: { $size: "$ufos" },
           "city name": "$city_name",
           "not from the US" : {
                $in: [ "$_id", db.states.findOne({state_code: 'NONE'}).cities ]
            }
         }
     },
     { $match: { "not from the US" : true }},
     { $sort : { "city name" : 1} }
])
```

Aggregate | Aggregate | Aggregate | Aggregate | Aggregate | Aggregate | Do

50 Documents 1 to 41

```
{
    "ufo_count" : NumberInt(1),
    "city name" : "Abingdon",
    "not from the US" : true
}
{
    "ufo_count" : NumberInt(1),
    "city name" : "Ahmedabad (India)",
    "not from the US" : true
}
{
    "ufo_count" : NumberInt(1),
    "city name" : "Ballynahinch ((Northern Ireland)",
    "not from the US" : true
}
{
    "ufo_count" : NumberInt(1),
    "city name" : "Bangalore (India)",
    "not from the US" : true
}
```

3. Sightings in ireland

```
*/
//3. get the ufos sightings in Ireland
irish_ufos = db.cities.find( { city_name: { $regex: /ireland/i }}).toArray()

```

Array | Document | Document | Aggregate

```
[
    {
        "_id" : ObjectId("5c12c47e52cf683b1d7d7890"),
        "ufos" : [
            ObjectId("5c12c47e52cf683b1d7d788f")
        ],
        "city_name" : "Ballynahinch ((Northern Ireland)",
        "__v" : 0.0
    },
    {
        "_id" : ObjectId("5c12c47e52cf683b1d7d76b5"),
        "ufos" : [
            ObjectId("5c12c47e52cf683b1d7d76b4")
        ],
        "city_name" : "Dublin (Republic of Ireland)",
        "__v" : 0.0
    }
]

```

4 & 5. Using an index on on ufos directly on the cities.ufos array, return one sighting from ireland

```

//4. create an index on the ufos so that they could be queried directly
db.cities.createIndex({"ufos": 1});

//5. test the index with secific object ID
c = db.cities.findOne( {ufos: ObjectId("5c12c47e52cf683b1d7d788f") } )
```

Array | Document | Document | Aggregate | Document | Document

50 Documents 1 to 1

```
{
    "_id" : ObjectId("5c12c47e52cf683b1d7d7890"),
    "ufos" : [
        ObjectId("5c12c47e52cf683b1d7d788f")
    ],
    "city_name" : "Ballynahinch ((Northern Ireland)",
    "__v" : 0.0
}

```

6. get the ufo data including the irish city name of this specific ufo

```
//5. test the index with secific object ID
c = db.cities.findOne( {ufos: ObjectId("5c12c47e52cf683b1d7d788f") } )
*/
//6. get the ufo data including the irish city name of this specific ufo
db.ufos.aggregate(
    [
        { $match : { _id : {$in: c.ufos }}},
        { $project:
          { shape:1,
            seen_date:1,
            city: c.city_name,
            seen_duration:1
          },
        },
    ]
);
```

Array | Aggregate

50 | Documents 1 to 1

```
{
    "_id" : ObjectId("5c12c47e52cf683b1d7d788f"),
    "shape" : "",
    "seen_date" : ISODate("2013-09-15T21:30:00.000+0000"),
    "seen_duration" : "20 seconds",
    "city" : "Ballynahinch ((Northern Ireland)"
}
```

7. return the city that has had at least 5 ufos sightings

```
//7. return the city that has had at least 5 ufos sightings
db.cities.aggregate([
   {
      $project: {
         _id: 0,
         city_name:1,
         ufo_count: { $cond: { if: { $gte: [ { $size: "$ufos" }, 5 ] }, then: { $size: "$ufos" }, else: "we don't see UFOs here much"} }
      }
   }
])
```

Array | Aggregate | Aggregate

50 | Documents 1 to 50 | JSO

```
{
    "city_name" : "Milwaukee",
    "ufo_count" : NumberInt(5)
}
{
    "city_name" : "Chicago",
    "ufo_count" : NumberInt(10)
}
{
    "city_name" : "Salinas",
    "ufo_count" : "we don't see UFOs here much"
}
{
    "city_name" : "Merrick",
    "ufo_count" : "we don't see UFOs here much"
```

8. return UFO That were seen in New York in 2015

part 1, get the city first

```
//8. return UFO That were seen in New York in 2015
NY = db.cities.findOne({city_name: 'New York'})
db.ufos.aggregate([
     {
       $project:
         {
           _id:"$_id",
           _id: 0,
           seen_date: "$seen_date",
           year: { $year: "$seen_date"},
           shape:1,
           "seen in New York" : { $in: [ "$_id", NY.ufos ]}
         }
     },
     { $match: { $and: [ { "seen in New York" : true },  { "year" : 2015 } ] } },
])
/*
```

Document | Aggregate

50 Documents 1 to 1

```
{
    "_id" : ObjectId("5c12c47e52cf683b1d7d73b0"),
    "ufos" : [
        ObjectId("5c12c47e52cf683b1d7d73af"),
        ObjectId("5c12c47e52cf683b1d7d7629"),
        ObjectId("5c12c47e52cf683b1d7d76de"),
        ObjectId("5c12c47e52cf683b1d7d7736"),
        ObjectId("5c12c47e52cf683b1d7d7999"),
        ObjectId("5c12c47e52cf683b1d7d799c"),
        ObjectId("5c12c47e52cf683b1d7d7a16"),
        ObjectId("5c12c47e52cf683b1d7d7a66")
    ],
    "city_name" : "New York",
    "__v" : 0.0
}
```

part2: get the ufo data

as you can see, out of 8 UFOs only 5 match both 2015 and New York

```
//8. return UFO That were seen in New York in 2015
NY = db.cities.findOne({city_name: 'New York'})
db.ufos.aggregate([
    {
      $project:
        {
          _id:"$_id",
          _id: 0,
          seen_date: "$seen_date",
          year: { $year: "$seen_date"},
          shape:1,
          "seen in New York" : { $in: [ "$_id", NY.ufos ]}
        }
    },
    { $match: { $and: [ { "seen in New York" : true },  { "year" : 2015 } ] } }
])
```

Document | Aggregate

50 | Documents 1 to 5

```
{
    "shape" : "fireball",
    "seen_date" : ISODate("2015-07-26T00:45:00.000+0000"),
    "year" : NumberInt(2015),
    "seen in New York" : true
}
{
    "shape" : "circle",
    "seen_date" : ISODate("2015-07-12T19:35:00.000+0000"),
    "year" : NumberInt(2015),
    "seen in New York" : true
}
{
    "shape" : "light",
    "seen_date" : ISODate("2015-07-04T21:30:00.000+0000"),
    "year" : NumberInt(2015),
    "seen in New York" : true
}
{
    "shape" : "circle",
    "seen_date" : ISODate("2015-07-04T21:30:00.000+0000")
```

9. return the state with the most UFO sights ever

what I wanted to do : for every doc in states**(top level collection)** count the number of ufo sights**(lowest level collection)** and return total as **one number**

please read comments in queries.js for further explanation.

```
// ATTEMPT 2: using map reduce with $lookup to join the results
//            accumulated in the cities collection to the states collection.
// DID IT WORK?: YES.
// WHAT WENT RIGHT?: [..seat back or grab some popcorn..]
// first of all, I reduced every row in the cities collection to the lenght
// of its cities.ufos subcollection, using map reduce below
var mapCityUfos = function() {
    emit(this._id, this.ufos.length);
};
var reduceCities = function(city_id, values) {
        return values;
};
db.cities.mapReduce(
    mapCityUfos,
    reduceCities,
    { out:"at_last"} // the map reduce output will be saved in a collection called 'at_last'
)
db.at_last.find() // view the reduced result..
```

for every city._id, return len(ufos)

a bit redundant as value is computed in map in (o)1 but we need to put something here so..

Document | Find | Aggregate

50 | Documents 1 to 1

```
{
    "result" : "at_last",
    "timeMillis" : 276.0,
    "counts" : {
        "input" : 814.0,
        "emit" : 814.0,
        "reduce" : 0.0,
        "output" : 814.0
    },
    "ok" : 1.0
}
```

reduce: 0.0 is intended, because I shrunk cities based on nested sub-collection cities.ufos and not cities itself (see next output)

view the result found
(each city reduced to the number of UFOs recorded)
this is why reduce had 0.0 because the reduce was done on nested document 'ufos'

```
// ATTEMPT 2: using map reduce with $lookup to join the results
//            accumulated in the cities collection to the states collection.
// DID IT WORK?: YES.
// WHAT WENT RIGHT?: [..seat back or grab some popcorn..]
// first of all, I reduced every row in the cities collection to the lenght
// of its cities.ufos subcollection, using map reduce below
var mapCityUfos = function() {
    emit(this._id, this.ufos.length);
};
var reduceCities = function(city_id, values) {
         return values;
};
db.cities.mapReduce(
    mapCityUfos,
    reduceCities,
    { out:"at_last"} // the map reduce output will be saved in a collection called 'at_last'
)
db.at_last.find() // view the reduced result..
```

Document | Find | Aggregate

50 Documents 1 to 50

```
{
    "_id" : ObjectId("5c12c47e52cf683b1d7d734b"),
    "value" : 2.0
}
{
    "_id" : ObjectId("5c12c47e52cf683b1d7d734e"),
    "value" : 1.0
}
{
    "_id" : ObjectId("5c12c47e52cf683b1d7d7351"),
    "value" : 1.0
}
{
    "_id" : ObjectId("5c12c47e52cf683b1d7d7354"),
    "value" : 2.0
```

Finally the award to the state with the most sighting ever goes to

.

.

are you ready?

.

.

.

.

CA: Carolina(Western U.S)

PS: this ranking is only applicable to the local CSV dataset parsed; which might not reflect the accuracy of the fact.

```
183     {
184         $group : // group by state ID and by state code
185         {
186            '_id' : {
187              '_id': '$_id',
188              'state_code':'$state_code'
189            },
190            total_sights: { $sum: '$count'} // get the sum of all sights
191         }
192      },
193      { // sort in the total number of signts in descending order of sig
194        // change -1 to 1 to view the city with the least sights
195         $sort : { 'total_sights' : -1, 'state_code': 1}
196      },
197      {  // uncomment to leave the city with the most sights
198     // or change this value to view N of them
199         $limit : 1
200      }
201   ]
202 )
203
204 // FINAL THOUGHTS ON USING MAP-REDUCE AFTER BEING CLOSE TO GIVING UP
205
206 // after successfully mapping and reducing the ciities collection then
207 // I started wondering if creating a view like in typical relational DB
208
209 // the collections I designed were fully normalised with 1 to many(stat
210 // this saves on updates/delete as the state name is not replicated mar
```

Document | Find | Aggregate

50 | Documents 1 to 1

```
1 {
2     "_id" : {
3         "_id" : ObjectId("5c12c47e52cf683b1d7d737a"),
4         "state_code" : "CA"
5     },
6     "total_sights" : 86.0
7 }
```

🏆 👽 🥇