Indexing and aggregation MongoDB

Import json file to MongoDB

https://docs.mongodb.com/guides/server/import/

If you didn't import the city_inspections last class:

- Download the file city_inspections.json from ICON
- Use the mongoimport utility:

```
mongoimport --db moderndb --collection city_inspections
       --drop --file ~\downloads\city_inspections.json
```

Or if you enabled authentication:

```
mongoimport --db moderndb --collection city_inspections
       --authenticationDatabase admin --username <user>
       --password <password>  --drop
       --file ~\downloads\city_inspections.json
```

Execution stats

Let's get the execution statistics for a query by certificate_number in the city_inspections collection:

```
db.city_inspections.find({certificate_number:
10003581}).explain("executionStats").executionStats
```

- You can also set the profiler to record all or long running queries. To disactivate the profiler (default)

```
db.setProfilingLevel(0)
```

- Recall that Mongo automatically creates an index by the _id

```
db.getCollectionNames().forEach(function(collection){
    print("Indexes for the " + collection + " collection:");
    printjson(db[collection].getIndexes());
});
```

Create an Index

Let's create an index on certificate_number
db.city_inspections.createIndex ({"certificate_number": 1}, {unique: false})

Now let's see the improvement on executing the same query:

db.city_inspections.find({certificate_number: 10003581}).explain("*executionStats*").executionStats

Query by id
db.city_inspections.find({id:"108-2015-UNIT"}).explain("*executionStats*").executionStats

Now let's create a hash index over id
db.city_inspections.createIndex ({"id": "hashed"})

Create an index on zip code (from address document)

db.city_inspections.findOne()

Ascending order
db.city_inspections.createIndex ({“address.zip”: 1})

Descending order
db.city_inspections.createIndex ({“address.zip”: -1})

db.city_inspections.getIndexes()

Let’s drop one:

db.city_inspections.dropIndex (“address.zip_1”)

More queries

db.city_inspections.find ({“certificate_number”: {“$lt”: 100000}}).sort({“address.zip”: -1})

Single-purpose aggregators:

Count - number of documents in the result
db.city_inspections.find ({“certificate_number”: {“$lt”: 100000}}).count()
db.city_inspections.count ({“certificate_number”: {“$lt”: 100000}})

Distinct – collect the result set into an array of unique values
db.city_inspections.distinct(“address.zip”,
{“certificate_number”: {“$gt”: 100000}})

Aggregate – returns document according to the logic you provide

Aggregate (a pipeline-style logic)

https://docs.mongodb.com/manual/aggregation/#aggregation-pipeline

Stages (full list https://docs.mongodb.com/manual/reference/operator/aggregation-pipeline/):

$match – filters
$group – group by
$sort – order by
$project – select tags/documents to display in the results set
$limit – limit the number of results

* hint option can be used to force the usage of the specified index

Count the number of passed inspections per city

```
db.city_inspections.aggregate([
{$match: { “result”: {$eq: “No Violation Issued”}}},
{$group: {
        _id: “$address.city”,
        count: {$sum: 1}
        }
}
])
```

Count the number of passed inspections per city order by count

```
db.city_inspections.aggregate([
{$match: { “result”: {$eq: “No Violation Issued”}}},
{$group: {
        _id: “$address.city”,
        count: {$sum: 1}
        }
},
{$sort: {"count": -1}}
])
```

Cities with over 200 passed inspections order by count

```
db.city_inspections.aggregate([
{$match: { “result”: {$eq: “No Violation Issued”}}},
{$group: {
        _id: “$address.city”,
        count: {$sum: 1}
        }
},
{$sort: {"count": -1}},
{$match: {“count”: {$gt: 200}}}
])
```

Server side commands

- Pre-built Mongo commands execute in the server
- Some of them need to be executed under the admin database

- use moderndb
- db.listCommands() –most of the commands execute on the server, not the client

- To have our own functions executed on the server (similar to stored procedures), add it to collection system.js

```
db.system.js.save ({_id: “getLast”, value: function(collection){
        return collection.find({}).sort({‘_id’: 1}).limit(1)[0];
        }
})
use moderndb
db.loadServerScripts()
getLast(db.inventory).display
```

For today..

- https://docs.mongodb.com/manual/tutorial/aggregation-zip-code-data-set/
- Write a query to return Largest and Smallest Cities for these Midwest States: **Illinois**, **Indiana**, Iowa, and **Kansas**
- Submit your query to ICON