Advanced Database Design
Mongo Aggregation

Gregory S DeLozier, Ph.D.

gdelozie@kent.edu

NoSQL Survey - review these pages

Survey - https://en.wikipedia.org/wiki/NoSQL

CouchDB - http://couchdb.apache.org/

BigTable - https://en.wikipedia.org/wiki/BigTable

Cassandra - https://en.wikipedia.org/wiki/Apache_Cassandra
http://cassandra.apache.org/

Redis - https://en.wikipedia.org/wiki/Redis
http://redis.io/

Aggregation

- We want to combine results for a large number of records
- Grouping - creating groups of records to be processed together
- Matching - processing a group of records
- These are called "stages" of "aggregation"
- A sequence of stages is called an "aggregation pipeline"

Aggregation Resources

- Based on material here:
  - https://docs.mongodb.org/manual/aggregation/
- Zip Code examples here:
  - https://docs.mongodb.org/manual/tutorial/aggregation-zip-code-data-set/

Zip Code Dataset

- JSON data
- Import from:
  - media.mongodb.org/zips.json
- Use *wget* or *curl*


```
{
  "_id": "10280",
  "city": "NEW YORK",
  "state": "NY",
  "pop": 5574,
  "loc": [
    -74.016323,
    40.710537
  ]
}
```

Using MongoImport

- *mongoimport* will import JSON, CSV, and TSV files.
- More data here:
  - https://docs.mongodb.org/v3.0/reference/program/mongoimport
- On Codio, start mongo:
  - $ parts start mongodb
- We need this line:
  - `$ mongoimport --db examples --collection zipcodes --file zips.json`

The *aggregate()* method

- Use *mongo shell.*
- There are analogs in various language drivers
- Takes an array of stages:

```
db.zipcodes.aggregate( [
   { $group: { _id: "$state", totalPop: { $sum: "$pop" } } },
   { $match: { totalPop: { $gte: 10*1000*1000 } } }
] )
```

Aggregation Example

$ mongo

> show dbs

> use examples

```
> db.zipcodes.aggregate([
    { $group: { _id: "$state", totalPop: { $sum: "$pop" } } },
    { $match: { totalPop: { $gte: 10*1000*1000 } } }
] )
```

Intermediate Documents

Lots of intermediate items that look like this:

```
{
  "_id" : "AK",
  "totalPop" : 550043
}
```

Equivalent SQL

```
SELECT state, SUM(pop) AS totalPop
FROM zipcodes
GROUP BY state
HAVING totalPop >= (10*1000*1000)
```

MultiStage Pipeline

Pipelines can have many stages.

```
db.zipcodes.aggregate( [
  { $group: { _id: { state: "$state", city: "$city" }, pop: { $sum: "$pop" } } },
  { $group: { _id: "$_id.state", avgCityPop: { $avg: "$pop" } } }
] )
```

Complicated Pipeline

```
db.zipcodes.aggregate( [
  { $group:
    {
      _id: { state: "$state", city: "$city" },
      pop: { $sum: "$pop" }
    }
  },
  { $sort: { pop: 1 } },
  { $group:
    {
      _id : "$_id.state",
      biggestCity:  { $last: "$_id.city" },
      biggestPop:   { $last: "$pop" },
      smallestCity: { $first: "$_id.city" },
      smallestPop:  { $first: "$pop" }
    }
  }
] )
```

Projection Stage

We can add stages that reformat

```
// the following $project is optional, and
// modifies the output format.

{ $project:
  { _id: 0,
    state: "$_id",
    biggestCity:  { name: "$biggestCity",  pop: "$biggestPop" },
    smallestCity: { name: "$smallestCity", pop: "$smallestPop" }
  }
}
```

Python Aggregation API

- http://api.mongodb.org/python/current/examples/aggregation.html
- Setup:

```
>>> from pymongo import MongoClient
>>> db = MongoClient().aggregation_example
>>> result = db.things.insert_many([{"x": 1, "tags": ["dog", "cat"]},
...                                 {"x": 2, "tags": ["cat"]},
...                                 {"x": 2, "tags": ["mouse", "cat", "dog"]},
...                                 {"x": 3, "tags": []}])
>>> result.inserted_ids
[ObjectId('...'), ObjectId('...'), ObjectId('...'), ObjectId('...')]
```

SON -- Ordered Dictionary

- Works like a dictionary
- Maintains order of keys
- Convert with "dict(...)" or recursively with "<son_object>.to_dict()"

```
>>> from bson.son import SON
```

Explaining Aggregation Plans

- You can get the plan of data activity for an aggregation.

```
>>> db.command('aggregate', 'things', pipeline=pipeline, explain=True)
```

Python Aggregation Example

```
>>> from bson.son import SON
>>> pipeline = [
...     {"$unwind": "$tags"},
...     {"$group": {"_id": "$tags", "count": {"$sum": 1}}},
...     {"$sort": SON([("count", -1), ("_id", -1)])}
... ]
>>> list(db.things.aggregate(pipeline))
```