DATA8005 – Distributed Data Management

# MongoDB Aggregation Lab

## Overview

In the previous lab, we used pymongo to perform some basic database operations. This included one example of aggregation with count:

print(collection.count\_documents({"$expr":{"$gt":[{"$strLenCP":"$name"},5]}}))

In this lab, we will do some more work with aggregation, but on a much larger dataset containing artists and artworks of the Tate Gallery.

The following is an example of an artist document for Josef Albers. There are numerous sub-documents, e.g. activePlaces and birth, and some of these have sub-documents, such as birth having place, and so on. Notice also the movements list, which has two movement sub-documents, Post Painterly Abstraction and Bauhaus.

{

"activePlaceCount":2,

"activePlaces":[

{

"name":"United States",

"placeName":"United States",

"placeType":"nation"

},

{

"name":"München, Deutschland",

"placeName":"München",

"placeType":"inhabited\_place"

}

],

"birth":{

"place":{

"name":"Bottrop, Deutschland",

"placeName":"Bottrop",

"placeType":"inhabited\_place"

},

"time":{

"startYear":1888

}

},

"birthYear":1888,

"date":"1888–1976",

"death":{

"place":{

"name":"New Haven, United States",

"placeName":"New Haven",

"placeType":"inhabited\_place"

},

"time":{

"startYear":1976

}

},

"fc":"Josef Albers",

"gender":"Male",

"id":636,

"mda":"Albers, Josef",

"movements":[

{

"era":{

"id":8,

"name":"20th century 1900-1945"

},

"id":294,

"name":"Bauhaus"

},

{

"era":{

"id":415,

"name":"20th century post-1945"

},

"id":439,

"name":"Post Painterly Abstraction"

}

],

"startLetter":"A",

"totalWorks":22,

"url":"http://www.tate.org.uk/art/artists/josef-albers-636"

}

A lot of data is aggregated here into a single artist document.

Now let’s look at an artwork document:

{

"\_id":ObjectId("5bb13bb0ea002fa5d2e32eb1"),

"acno":"A00019",

"acquisitionYear":1919,

"all\_artists":"William Blake",

"catalogueGroup":{

"accessionRanges":"A00012-A00032; T05845",

"completeStatus":"COMPLETE",

"groupType":null,

"id":65234,

"shortTitle":"Illustrations to 'The Book of Job'"

},

"classification":"on paper, unique",

"contributorCount":1,

"contributors":[

{

"birthYear":1757,

"date":"1757–1827",

"displayOrder":1,

"fc":"William Blake",

"gender":"Male",

"id":39,

"mda":"Blake, William",

"role":"artist",

"startLetter":"B"

}

],

"creditLine":"Purchased with the assistance of a special grant from the National Gallery and donations from the Art Fund, Lord Duveen and others, and presented through the Art Fund 1919",

"dateRange":{

"endYear":1825,

"startYear":1825,

"text":"1825, reprinted 1874"

},

"dateText":"1825, reprinted 1874",

"depth":"",

"dimensions":"image: 199 x 150 mm",

"foreignTitle":null,

"groupTitle":"Illustrations to 'The Book of Job'",

"height":"150",

"id":1053,

"inscription":null,

"medium":"Line engraving on paper",

"movementCount":0,

"subjectCount":7,

"subjects":{

"children":[

{

"children":[

{

"children":[

{

"id":1340,

"name":"Job"

},

{

"id":16603,

"name":"Job, chapter 3"

}

],

"id":135,

"name":"Bible: Old Testament"

}

],

"id":132,

"name":"religion and belief"

},

{

"children":[

{

"children":[

{

"id":727,

"name":"grief"

}

],

"id":31,

"name":"emotions and human qualities"

}

],

"id":29,

"name":"emotions, concepts and ideas"

},

{

"children":[

{

"children":[

{

"id":934,

"name":"hand/hands raised"

},

{

"id":4593,

"name":"head in hand/hands"

},

{

"id":272,

"name":"kneeling"

}

],

"id":92,

"name":"actions: postures and motions"

}

],

"id":91,

"name":"people"

},

{

"children":[

{

"children":[

{

"id":2018,

"name":"religious"

}

],

"id":166,

"name":"inscriptions"

}

],

"id":162,

"name":"symbols & personifications"

}

],

"id":1,

"name":"subject"

},

"thumbnailCopyright":null,

"thumbnailUrl":"http://www.tate.org.uk/art/images/work/A/A00/A00019\_8.jpg",

"title":"Job’s Despair",

"units":"mm",

"url":"http://www.tate.org.uk/art/artworks/blake-jobs-despair-a00019",

"width":"199"

}

Note the contributors field, which contains limited artist data (a contributor might be an artist, but others may have contributed in other ways). So there is some duplication of data, but it makes retrieval of data for common searches (such as by artist or artwork) much easier and summarised artist information is made available for display on a web-page, for example (the user can always click on the artist to get more information, which would lead to a subsequent query by artist).

## How to Import the Tate dataset

Download artists.json and artworks.json from the “Sample Code” folder in Blackboard.

From the command line, CD to your mongodb bin folder and enter the following:

mongoimport --db tate --collection artists --file artists.json

mongoimport --db tate --collection artworks --file artworks.json

Note: either put the full path in from of artists.json and artworks.json or put the file in the bin directory and leave the above commands as is.

## Some Queries to Try

Using the mongo command line client, enter the following commands:

use tate

db.artists.find({"fc":"William Blake"}).pretty()

You should get nicely-formatted JSON for William Blake.

Now try:

db.artists.find(

{"birthYear" : {$gt : 1970}}

).explain("executionStats")

You should get a JSON document as your result, similar to this one:

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "tate.artists",

"indexFilterSet" : false,

"parsedQuery" : {

"birthYear" : {

"$gt" : 1970

}

},

"winningPlan" : {

"stage" : "COLLSCAN",

"filter" : {

"birthYear" : {

"$gt" : 1970

}

},

"direction" : "forward"

},

"rejectedPlans" : [ ]

},

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 143,

"executionTimeMillis" : 43,

"totalKeysExamined" : 0,

"totalDocsExamined" : 3538,

"executionStages" : {

"stage" : "COLLSCAN",

"filter" : {

"birthYear" : {

"$gt" : 1970

}

},

"nReturned" : 143,

"executionTimeMillisEstimate" : 0,

"works" : 3540,

"advanced" : 143,

"needTime" : 3396,

"needYield" : 0,

"saveState" : 27,

"restoreState" : 27,

"isEOF" : 1,

"invalidates" : 0,

"direction" : "forward",

"docsExamined" : 3538

}

},

"serverInfo" : {

"host" : "DESKTOP-NQ1MKNN",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

Copy and paste your result into a document.

You don’t need to understand everything here. If reports, for one thing, on the efficiency of the query. See <https://docs.mongodb.com/manual/reference/explain-results/> for a bit more about explain plans. You’ll notice that the stage is COLLSCAN, which means the entire collection was scanned rather than, say, using an index to speed things up.

Let’s add an ascending index to the birthYear field:

db.artists.createIndex({birthYear: 1})

Rerun the previous explain plan command. You might see something like:

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "tate.artists",

"indexFilterSet" : false,

"parsedQuery" : {

"birthYear" : {

"$gt" : 1970

}

},

"winningPlan" : {

"stage" : "FETCH",

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"birthYear" : 1

},

"indexName" : "birthYear\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"birthYear" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"birthYear" : [

"(1970.0, inf.0]"

]

}

}

},

"rejectedPlans" : [ ]

},

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 143,

"executionTimeMillis" : 21,

"totalKeysExamined" : 143,

"totalDocsExamined" : 143,

"executionStages" : {

"stage" : "FETCH",

"nReturned" : 143,

"executionTimeMillisEstimate" : 0,

"works" : 144,

"advanced" : 143,

"needTime" : 0,

"needYield" : 0,

"saveState" : 1,

"restoreState" : 1,

"isEOF" : 1,

"invalidates" : 0,

"docsExamined" : 143,

"alreadyHasObj" : 0,

"inputStage" : {

"stage" : "IXSCAN",

"nReturned" : 143,

"executionTimeMillisEstimate" : 0,

"works" : 144,

"advanced" : 143,

"needTime" : 0,

"needYield" : 0,

"saveState" : 1,

"restoreState" : 1,

"isEOF" : 1,

"invalidates" : 0,

"keyPattern" : {

"birthYear" : 1

},

"indexName" : "birthYear\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"birthYear" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"birthYear" : [

"(1970.0, inf.0]"

]

},

"keysExamined" : 143,

"seeks" : 1,

"dupsTested" : 0,

"dupsDropped" : 0,

"seenInvalidated" : 0

}

}

},

"serverInfo" : {

"host" : "DESKTOP-NQ1MKNN",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

Compare it to your previous explain plan. Notice how it now used an IXSCAN (index scan) instead of a full collection scan and in this example run the execution time was roughly halved (43ms down to 21ms).

Now let’s try some aggregate functions to see whether female or male artists in the Tate collection are more prolific in terms of artworks (at least, among their artworks owned by or loaned to the Tate).

db.artists.aggregate([

{ $group : {

\_id : "$gender",

total : {$sum : 1}

}

}

])

db.artists.aggregate([

{ $group : {

\_id : "$gender",

total : {$avg : "$totalWorks"}

}

}

])

This will find the average number of total artworks for each artist, grouped by gender. There are also some null genders, for which you get a sum and average. You’ll notice a significant gender imbalance in terms of numbers of artists and artworks.

This is a list of aggregate expressions:

1. **Sum** $sum
2. **Average** $avg
3. **Min** $min
4. **Max** $max
5. **First** $first
6. **Last** $last

## Exercise

**1.**

List the total number of artworks in the Tate collection by place name.

You’ll end up with a list starting like this:

{ "\_id" : "España", "total" : 1 }

{ "\_id" : "Sausalito", "total" : 1 }

{ "\_id" : "Heppenheim an der Bergstrasse", "total" : 14 }

{ "\_id" : "Smolensk", "total" : 6 }

{ "\_id" : "Sayda", "total" : 119 }

{ "\_id" : "Lusaka", "total" : 10 }

{ "\_id" : "Fukuoka", "total" : 6 }

{ "\_id" : "Nagasaki", "total" : 10 }

{ "\_id" : "Kyoto", "total" : 14 }

{ "\_id" : "Hythe", "total" : 11 }

{ "\_id" : "Haarlem", "total" : 4 }

{ "\_id" : "Peterborough", "total" : 10 }

{ "\_id" : "Henley-on-Thames", "total" : 38 }

**2.**

Group artwork titles by acquisition year (using the artworks collection). Limit it to 20 results (by adding .limit(20) to the end of your query).

See <https://docs.mongodb.com/manual/reference/operator/aggregation/group/> for an example that pivots books to have titles grouped by author.

To limit aggregation, see <https://docs.mongodb.com/manual/reference/operator/aggregation/limit/>.

You should see something like this for each acquisition year:

{ "\_id" : 2010, "artwork" : [ "IDYLLS END IN THUNDERSTORMS", "A LAST WORD: RUDDER", "The Music from the Balconies", "No Title (Stacked Plates)", "No Title (Beard Cart)" ] }

## Assessment

There will be another 10 marks (out of 50) for queries like these in project 1, to be handed up in week 7. Instructions to follow shortly. You will need to devise queries to run against the tate database. A little bit of research (or Googling) will be required, but nothing too arduous.