CSCI235 Database Systems

# MongoDB Aggregation Framework

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

# Aggregations

## Outline

# Aggregation framework

Aggregation framework is a query language that that can be used to transform and to combine data from multiple documents in order to generate new information not available in any single document

Aggregation framework in MongoDB is similar to SQL `GROUP BY` clause and `HAVING` clause

Aggregation framework makes a task database search much easier and more efficient through specification of a series of operations in an array and processing it in a single call

Aggregation framework defines an aggregation pipeline where the output from each step in the pipeline provides input to the next step

Every step in a pipeline executes a single operation on the input documents to transform the input and to generate output document

A pipeline processes a stream of documents through several operations like filtering, projecting, grouping, sorting, limiting, and skipping

The same operations can be repeated many times in a pipeline in any order

# Aggregation framework

Some operations that can be used in a pipeline:

- `$project`: Extracts the components of a documents to be placed in an output document (similar to `SELECT` clause)
- `$match`: Filters the documents to be processed, similar to `find()` (and similar to `WHERE` clause)
- `$limit`: Limits the total number of documents to be passed to the next operation (similar to `LIMIT` clause, `rownum` condition)
- `$skip`: Skips a given specified number of documents
- `$unwind`: Expands (unnest) an array, generating one output document for each array entry
- `$group`: Groups documents by a specified key (`GROUP BY` clause)
- `$sort`: Sorts documents (`ORDER BY` clause)
- `$out`: Saves the results from a pipeline to a collection
- `$count`: Counts the total number of documents in a pipeline

# Sample collection

## A sample document

Sample document

```
db.department.insert(
  { "name":"School of Astronomy",
    "code":"SOA",
    "total_staff_number":25,
    "budget":10000,
    "address":{"street":"Franz Josef Str",
               "bldg":4,
               "city":"Vienna",
               "country":"Austria"},
   "courses":[ {"code":"SOA101",
                "title":"Astronomy for Kids",
                "credits":3},
               {"code":"SOA201",
                "title":"Black Holes",
                "credits":6},
               {"code":"SOA301",
                "title":"Dark Matter",
                "credits":12}
             ]
  }
);
```

# Aggregations

## Outline

# $project

$project extracts components of subdocuments, rename components, and performs operations on components

Select name of each department, skip document identifier

aggregate()

```
db.department.aggregate([ {$project:{"name":1,"_id":0}} ])
```

Select name of each department and rename name component

aggregate()

```
db.document.aggregate([{$project:{"Department name":"$name"}}])
```

"$keyname" syntax is used to refer to a value associated with a key "keyname" in the aggregation framework

Select a name of each department and concatenate it with its code

aggregate()

```
db.document.aggregate([{$project:{"Name and code":
                                    {"$concat":["$name","-","$code"]}}}])
```

# $project

Select a name of department and 10% of its budget

aggregate()

```
db.department.aggregate([ {$project:{"name":1,
                                     "10% of budget":{"$multiply":["$budget",0.1]},
                                     "_id":0}
                           }
                         ]
                        )
```

# Aggregations

## Outline

# $match

Operation `$match` selects the documents that satisfy a given condition

Find the departments with `budget > 10000`

aggregate()

```
db.department.aggregate([ {$match:{"budget":{$gt:10000}}} ])
```

Find the `names` and `codes` of departments with `budget > 10000`

aggregate()

```
db.department.aggregate([ {$match:{"budget":{$gt:10000}}},
                          {$project:{"name":1,"code":1,"_id":0}} ])
```

# Aggregations

## Outline

# $limit

Operation `$limit` passes a given number of documents through a pipeline

Find the first 2 documents

aggregate()

```
db.department.aggregate([ {$limit:2} ])
```

Find the first document with `budget > 1000`

aggregate()

```
db.department.aggregate([ {$match:{"budget":{"$gt":10000}}},
                          {$limit:1} ]
```

# Aggregations

## Outline

# $skip

Operation `$skip` eliminates a given number of documents from a pipeline

List all documents in a collection except the first two

aggregate()

```
db.department.aggregate([ {$skip:2} ])
```

List all documents with a `budget` greater than 10000 except the first one

aggregate()

```
db.department.aggregate([ {$match:{"budget":{"$gt":10000}}},
                          {$skip:1}] )
```

# Aggregations

## Outline

# $unwind

Operation `$unwind` creates a separate document for each element of a given array

A document is replicated for each element of an array, i.e. an array is unnested

For each department and for each course offered by a department create a separate document

aggregate()

```
db.department.aggregate([ {$unwind:"$courses"} ])
```

For each department and for each course offered by a department create a separate document, list only the courses

aggregate()

```
db.department.aggregate([ {$unwind:"$courses"},
                          {$project:{"courses":1,"_id":0}} ])
```

# $unwind

Use aggregation framework and `$unwind` operation to list the codes of all courses

aggregate()

```
db.department.aggregate([ {$unwind:"$courses"},
                          {$project:{"code":"$courses.code","_id":0}} ])
```

Results

```
{ "code" : "CSCI835" }
{ "code" : "CSIT115" }
{ "code" : "CSCI317" }
{ "code" : "CSIT321" }
{ "code" : "SOA101" }
{ "code" : "SOA201" }
{ "code" : "SOA301" }
{ "code" : "SOPH101" }
{ "code" : "SOPH102" }
{ "code" : "SOPH103" }
```

# $unwind

Use aggregation framework and `$unwind` operation to find all courses with 12 credits

aggregate()

```
db.department.aggregate([ {$unwind:"$courses"},
                          {$project:{"courses":1,"_id":0}},
                          {$match:{"courses.credits":12}} ])
```

# Aggregations

## Outline

# $group

Operation `$group` groups the documents and applies the aggregation functions to each group

List non-distinct values of `total_staff_number`

aggregate()

```
db.department.aggregate([{$project:{"total_staff_number":1,"_id":0}}])
```

Results

```
{ "total_staff_number" : 25 }
{ "total_staff_number" : 5 }
{ "total_staff_number" : 25 }
```

Group the documents by `total_staff_number` and list the distinct values of `total_staff_number`

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number"}} ])
```

Results

```
{ "_id" : 5 }
{ "_id" : 25 }
```

# $group

Perform groupings as above and rename `_id` to `total_staff_number`

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number"}},
                          {$project:{"total_staff_number":"$_id","_id":0}} ])
```

Results

```
{ "total_staff_number" : 5 }
{ "total_staff_number" : 25 }
```

Find the total number of distinct values of `total_staff_number`

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number"}},
                          {$count:"Total distinct values"}])
```

Find the total number of distinct values of `total_staff_number`

distinct()

```
db.department.distinct("total_staff_number").length
```

# $group

Group the documents by `total_staff_number` and by `budget`

aggregate()

```
db.department.aggregate([ {$group:{"_id":{"total_staff_number":"$total_staff_number",
                                          "budget":"$budget"}}} ])
```

Group the documents by `total_staff_number` and count the total number of departments in each group

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number",
                                   "total_departments":{$sum:1}}} ])
```

Results

```
{ "_id" : 5, "total_departments" : 1 }
{ "_id" : 25, "total_departments" : 2 }
```

# $group

Group the documents by `total_staff_number` and by `budget` and perform summation of budgets in each group

aggregate()

```
db.department.aggregate([ {$group:{"_id":{"total_staff_number":"$total_staff_number",
                                          "budget":"$budget"} ])
```

Results

```
{ "_id" : { "total_staff_number" : 5, "budget" : 10000 } }
{ "_id" : { "total_staff_number" : 25, "budget" : 100000 } }
{ "_id" : { "total_staff_number" : 25, "budget" : 1000000 } }
```

Group the documents by `total_staff_number` and perform summation of budgets in each group

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number",
                                   "total budgets":{$sum:"$budget"}}} ])
```

Results

```
{ "_id" : 5, "total budgets" : 10000 }
{ "_id" : 25, "total budgets" : 1100000 }
```

# $group

Group the documents by `total_staff_number` and find the largest budget in each group

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number",
                                   "largest budget":{$max:"$budget"}}} ])
```

Results

```
{ "_id" : 5, "largest budget" : 10000 }
{ "_id" : 25, "largest budget" : 1000000 }
```

Group the documents by `total_staff_number` and find the smallest budget in each group

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number",
                                   "smallest budget":{$min:"$budget"}}} ])
```

Results

```
{ "_id" : 5, "largest budget" : 10000 }
{ "_id" : 25, "largest budget" : 100000 }
```

# $group

Group the documents by `total_staff_number` and find an average budget in each group

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number",
                                   "largest budget":{$avg:"$budget"}}} ])
```

Other operators like `$first`, `$last` are useful when sorting is applied

What about application of aggregation operation on entire collection of documents ?

Then, (like in SQL) we assume that a collection is a single group

Count the total number of departments in a collection

aggregate()

```
db.department.aggregate([ {$group:{"_id":null,
                                   "total departments":{$sum:1}}} ])
```

Results

```
{ "_id" : null, "total_departments" : 3 }
```

# $group

Find the total and average budget in a collection

aggregate()

```
db.department.aggregate([ {$group:{"_id":null,
                                   "total budget":{$sum:"$budget"},
                                   "average budget":{$avg:"$budget"}}} ])
```

Results

```
{ "_id" : null, "total budget" : 1110000, "average budget" : 370000 }
```

# Aggregations

## Outline

# $sort

Operation `$sort` sorts the documents

Display the names and budgets of departments sorted in ascending order by budget

aggregate()

```
db.department.aggregate([ {$project: {"name":1,"budget":1,"_id":0}},
                          {$sort:{"budget":1}} ])
```

Results

```
{ "name" : "School of Astronomy", "budget" : 10000 }
{ "name" : "School of Physics", "budget" : 100000 }
{ "name" : "School of Computing and Information Technology", "budget" : 1000000 }
```

Display a name of a department with the largest budget, display a name of department and its budget

aggregate()

```
db.department.aggregate([ {$project:{"name":1,"budget":1,"_id":0}},
                          {$sort:{"budget":-1}},
                          {$limit:1} ])
```

# $sort

Group the documents by `total_staff_number`, count the total number of departments in each group display the results sorted in the descending order of the total number of departments

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number",
                                   "total departments":{$sum:1}}},
                          {$sort:{"total departments":-1}} ])
```

Results

```
{ "_id" : 25, "total departments" : 2 }
{ "_id" : 5, "total departments" : 1 }
```

# $sort

Group the documents by `total_staff_number`, count the total number of departments in each group display the results sorted in the descending order of the total number of departments and display only groups where total number of departments is greater than 1

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number",
                                   "total departments":{$sum:1}}},
                          {$sort:{"total departments":-1}},
                          {$match:{"total departments":{$gt:1}}} ])
```

Results

```
{ "_id" : 25, "total departments" : 2 }
```

# Aggregations

## Outline

# $out

Operation `$out` saves the results of processing in a collection

Find the total number of distinct values of `total_staff_number`

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number"}},
                          {$out:"total_distinct"}])
```

count()

```
db.total_distinct.count()
```

drop()

```
db.total_distinct.drop()
```

# Aggregations

## Outline

# $count

Operation `$count` counts the total number of documents in a pipeline

List the codes of all courses

aggregate()

```
db.department.aggregate([ {$unwind:"$courses"},
                          {$project:{"code":"$courses.code","_id":0}},
                          {$count:"Total number of courses"} ])
```

Find the total number of distinct values of `total_staff_number`

aggregate()

```
db.department.aggregate([ {$group:{"_id":"$total_staff_number"}},
                          {$count:"Total distinct values"} ])
```

# References

MongoDB Manual, Aggregation

Banker K., Bakkum P., Verch S., Garret D., Hawkins T., MongoDB in Action, 2nd ed., Manning Publishers, 2016

Chodorow K. MongoDB The Definitive Guide, O'Reilly, 2013