Introduction to Aggregation Framework

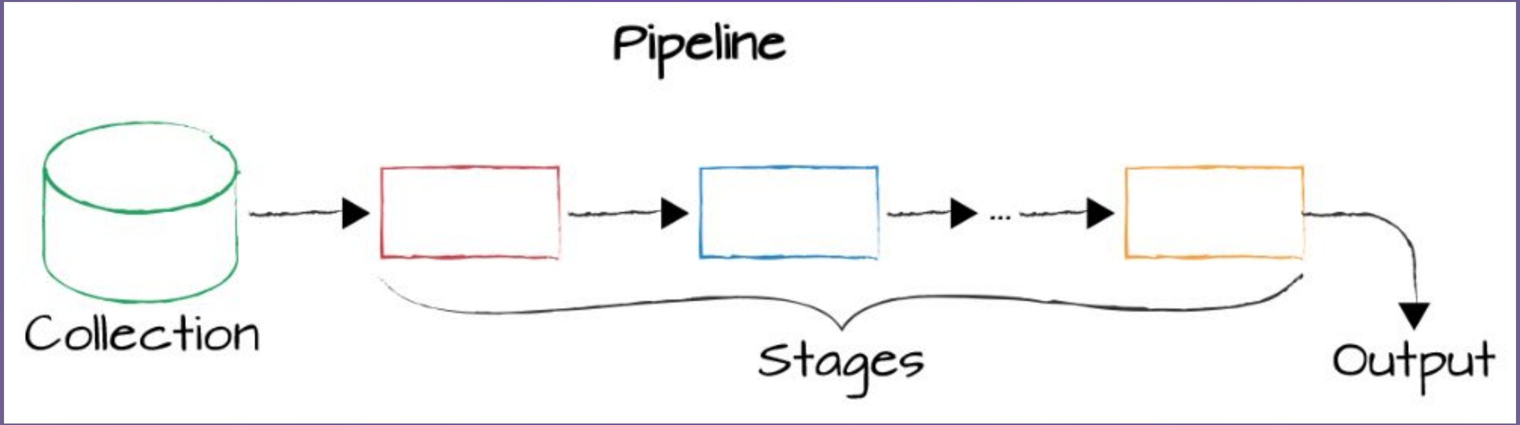
Paige Schaefer

Aggregation Framework

- What is Aggregation Framework?
  - A set of analytics tools within MongoDB that allow you to do analytics on documents in one or more collections
- Based off the idea of a Pipeline

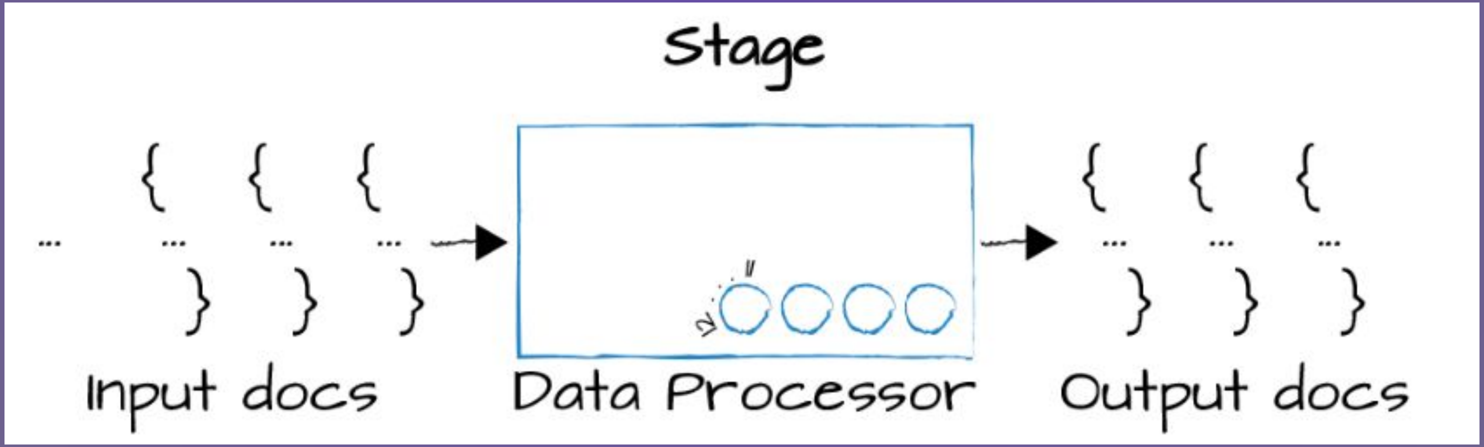
Pipelines

When input is taken from a MongoDB collection and pass the documents from that collection through one or more stages, each of which performs a different option on its inputs



Stages of Pipelines

Each stage provides a set of knobs or tunables that we can control to parameterize the stage to perform whatever task we’re interested in



```
{
  "id" : 2197,
  "round_code" : "c",
  "raised_amount" : 15000000,
  "raised_currency_code" : "USD",
  "funded_year" : 2008,
  "investments" : [
     {
     "company" : null,
     "financial_org" : {
       "name" : "European Founders Fund",
     "permalink" : "european-founders-fund"
      },
     "person" : null
     }
     ]
     }],
    "ipo" : {
        "valuation_amount" :NumberLong("104000000000"),
        "valuation_currency_code" : "USD",
        "pub_year" : 2012,
        "pub_month" : 5,
        "pub_day" : 18,
        "stock_symbol" : "NASDAQ:FB"
}
```

Example Part 1: Simple Filter with Project

```
db.companies.aggregate([
        {$match: {founded_year: 2004}},
])
```

```
db.companies.find({founded_year: 2004})
```

```
db.companies.aggregate([
      {$match: {founded_year: 2004}},
      {$project: {

        _id: 0,
         name: 1,
         founded_year: 1
        }}

])
```

```
{"name": "Digg", "founded_year": 2004 }
{"name": "Facebook", "founded_year": 2004 }
{"name": "AddThis", "founded_year": 2004 }
{"name": "Veoh", "founded_year": 2004 }
{"name": "Pando Networks", "founded_year": 2004 }
{"name": "Jobster", "founded_year": 2004 }
{"name": "AllPeers", "founded_year": 2004 }
{"name": "blinkx", "founded_year": 2004 }
...
```

Example Part 2: Limit Stage

```
db.companies.aggregate([
      {$match: {founded_year: 2004}},
      {$limit: 5},
      {$project: {
           _id: 0,
           name: 1}}
])
```

```
db.companies.aggregate([
      {$match: {founded_year: 2004}},
      {$project: {
           _id: 0,
           name: 1}},
      {$limit: 5}
])
```

```
{"name": "Digg"}
{"name": "Facebook"}
{"name": "AddThis"}
{"name": "Veoh"}
{"name": "Pando Networks"}
```

Example Part 3: Sort and Skip

```
db.companies.aggregate([
        { $match: { founded_year: 2004 } },
        { $sort: { name: 1} },
        { $limit: 5 },
        { $project: {
           _id: 0,
           name: 1}},
])
```

```
{"name": "1915 Studios"}
{"name": "1Scan"}
{"name": "2GeeksinaLab"}
{"name": "2GeeksinaLab"}
{"name": "2threads"}
```

```
db.companies.aggregate([
      {$match: {founded_year: 2004}},
      {$sort: {name: 1}},
      {$skip: 10},
      {$limit: 5},
      {$project: {
           _id: 0,
           name: 1}},
])
```

Expressions

- Boolean
- Set: allow us to work with arrays as sets
- Comparison: allow us to express different types of range filters
- Arithmetic
- String
- Array: allow us to manipulate arrays
- Variable: allow us to work with literals, expressions for parsing dates and conditional expressions
- Accumulators: allow us to calculate sums, descriptive statistics and many other types of values

$project

Types of reshaping operations that should be most common in the applications that you develop

```
db.companies.aggregate([
     {$match: {"funding_rounds.investments.financial_org.permalink": "greylock" }},
          {$project: {
               _id: 0,
               name: 1,
               ipo: "$ipo.pub_year",
               valuation: "$ipo.valuation_amount",
               funders: "$funding_rounds.investments.financial_org.permalink"
      }}
    ])
```

```
{
     "name" : "Digg",
          "funders" : [
     [
          "greylock",
          "Omidyar-network"
     ],
     [
          "greylock",
          "omidyar-network",
          "floodgate",
          "Sv-angel"
     ],
     [
          "highland-capital-partners",
          "greylock",
          "omidyar-network",
          "Svb-financial-group"
     ]

   ]

}
```

```
{
      "name" : "Facebook",
      "ipo" : 2012,
      "valuation" : NumberLong("104000000000"),
      "funders" : [
              [
                 "Accel-partners"
              ],
              [
                 "greylock",
                 "meritech-capital-partners",
                 "founders-fund",
                 "sv-angel"

              ],

              ...

              [

                 "goldman-sachs",

                 "Digital-sky-technologies-fo"

                 ]]}
```

Introduction to the Aggregation Framework
SLIDE 2

- What is Aggregation Framework?
    - A set of analytics tools within MongoDB that allow you to do analytics on documents in one or more collections
    - Based on the concept of a pipeline
        - Pipeline: set of data processing elements connected in series, where the output of one element is the input of the next one

SLIDE 3

- Take input from a MongoDB collection and pass the documents from that collection through one or more stages, each of which performs a different option on its inputs
- Each stage takes as input whatever the stage before it produced as output
- The inputs and outputs for all stages are documents specifically a stream of documents
- Each stage has a specific job it does
    - It expects a specific form of document and produces a specific output

SLIDE 4

- Takes in a stream of input documents one at a time, processes each document one at a time, and produces an output stream of documents on at a time
- Each stage provides a set of knobs or tunables, that we can control to parameterize the stage to perform whatever task we're interested in doing
- Tunables typically take the form of operators that will modify fields, perform arithmetic operations, reshape documents or do some sort of accumulation task

SLIDE 6

- Getting Started with Stages: Familiar Operators
    - Match, Project, Sort, Skip and Limit Stages
- Example
    - Simple Filter looking for all companies that were founded in 2004 **(PIC 1)**
        - Equivalent to find **(PIC 2)**
    - Add a project stage to our pipeline to reduce the output to just a few fields per document **(PIC 3)**
    - Aggregate method: method we call when we want to run an aggregation query
        - To aggregate we pass in an aggregation pipeline
            - Pipeline is an array with documents as elements
            - Each of the documents must stipulate a particular stage operator
    - In example we have a pipeline that has two stages: a match stage for filtering and a project stage with which we're limiting the out to just two fields per document
        - The match stage filters against the collection and passes the resulting documents to the project stage one at a time
        - The project stage then performs its operation, reshaping the documents and passes the output out of the pipeline and back to us

SLIDE 7

- Extending a limit Stage
    - Limit our result set to five and then project out fields we want **(PIC 1)**
    - Limit our output to just the names of each company
    - Constructed pipeline to limit before the project stage

        - If we ran the project stage first and then the limit we would get same result but have to pass hundreds of documents through the project stage before finally limiting the results to 5
- Make sure to always consider the efficiency of your aggregation pipeline
    - Ensure that you are limiting the number of documents that need to be passed on from one stage to another as you build your pipeline

SLIDE 8

- In last example we're only interested in the first five documents that match our query, regardless of how their sorted so its perfectly fine to limit second stage
- If order matters we'll need to sort before the limit stage
    - In aggregate framework we specify sort as a stage within a pipeline as follows (name in ascending order)
- Adding a skip stage **(PIC 2)**
    - Sort first, then skip the first 10 documents and again limit our result set to 5 documents
- Pipeline Review **(PIC 2)**
    - Five Stages
        - Filtering the companies collection, looking only for documents where the "founded_year" is 2004
        - Sorting based on the name in ascending order
        - Skipping the first 10 matches
        - Limiting our end results to 5
        - Pass those five documents on to the project stage where we reshape the documents such that our output documents contain just the company name

SLIDE 9

- *Boolean* expressions allow us to use AND, OR, and NOT expressions.
- *Set* expressions allow us to work with arrays as sets.
    - In particular, we can get the intersection or union of two or more sets.
    - We can also take the difference of two sets and perform a number of other set operations.
- *Comparison* expressions enable us to express many different types of range filters.
- *Arithmetic* expressions enable us to calculate the ceiling, floor, natural log, and log, as well as perform simple arithmetic operations like multiplication, division, addition, and subtraction.
    - We can even do more complex operations, such as calculating the square root of a value.
- *String* expressions allow us to concatenate, find substrings, and perform operations having to do with case and text search operations.
- *Array* expressions provide a lot of power for manipulating arrays, including the ability to filter array elements, slice an array, or just take a range of values from a specific array.
- *Variable* expressions, which we won't dive into too deeply, allow us to work with literals, expressions for parsing date values, and conditional expressions.
- *Accumulators* provide the ability to calculate sums, descriptive statistics, and many other types of values.

SLIDE 10

- $project
    - Types of reshaping operations that should be most common in the applications that you develop
- Example
    - Doing a match
    - Filtering for all companies that has a funding round in which Greylock Partners participated
        - “greylock” is unique identified
    - Uses dot notation to express field paths that reach into the “ipo” field and the “funding_rounds” field to select values from those nested documents and arrays
    - In output each document has a “name” field and a “funders” field
    - The $ character used to specify the values for ipo, valuation and funders in our project stage indicates that the values should be interpreted as field paths and used to select the value that should be projected for each field