

Modelado y Procesamiento de Grandes Volumenes de Datos

MongoDB – Aggregation Pipeline

CPAP, FING, Udelar – 2020

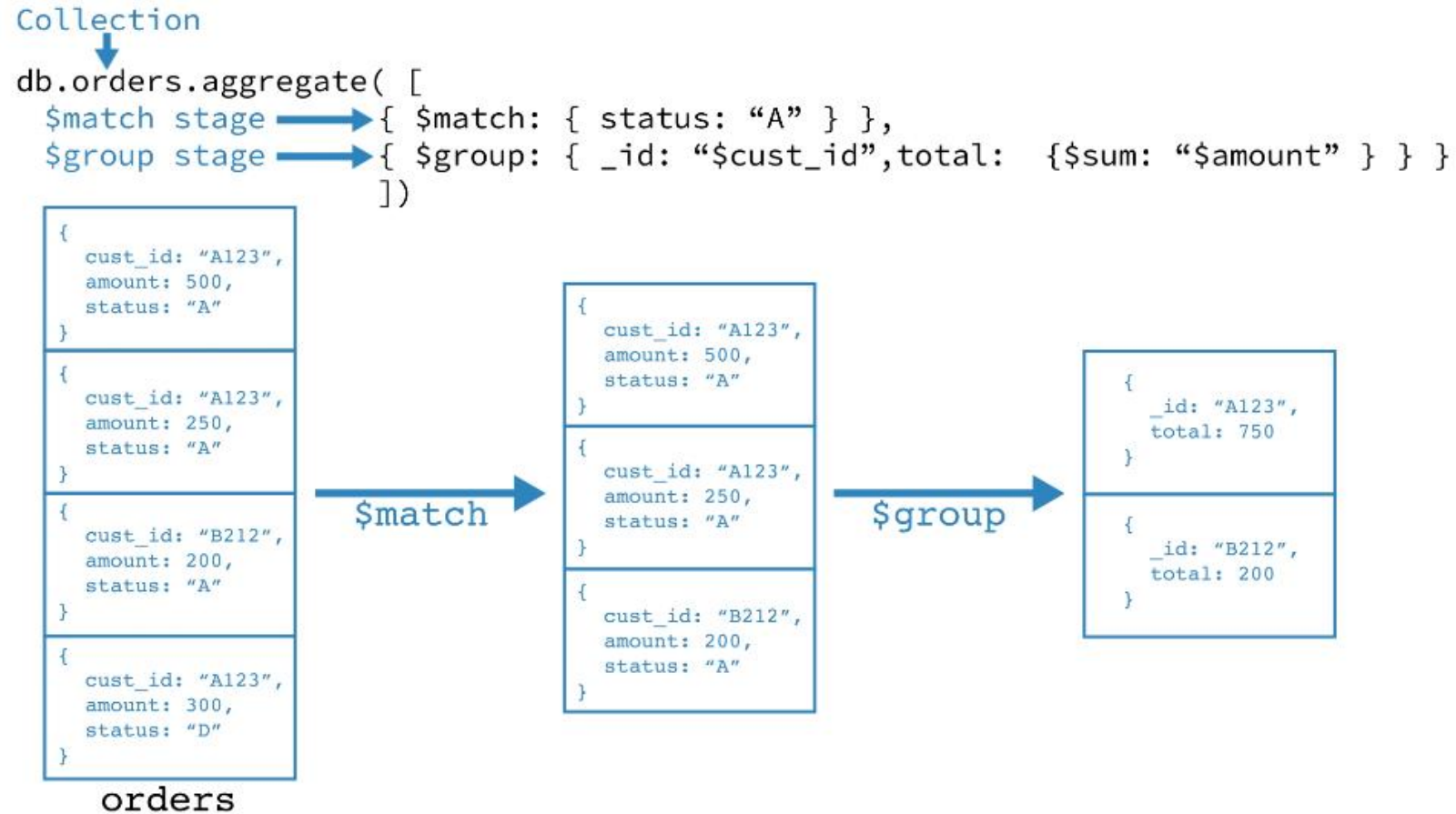
MongoDB Aggregation

- Aggregation operations process data records and return computed results
- MongoDB provides three ways to perform aggregation:
 - aggregation pipeline
 - map-reduce function
 - single purpose aggregation methods

Aggregation Pipeline

- multi-stage pipeline that transforms the documents into an aggregated result
- stages can use operators (arithmetic, boolean, string, etc)
- provides efficient data aggregation using native operations within MongoDB (preferred method)
- can operate on a sharded collection
- can use indexes to improve its performance during some of its stages
- has an internal optimization phase

Aggregation Pipeline - Method



Pipeline Expressions

- specify the transformation to apply to input documents of a stage
- have document structure and can contain other expression
- only operate on the current document in the pipeline and cannot refer to data from other documents
 - expression operations provide in-memory transformation of documents
- generally stateless:
 - exception , accumulators (`$max`, `$min`, `$avg`, `$sum`, etc) used in `$group` stage

Pipeline Expressions – Field Paths

- used to access fields in the input documents
- prefixing the field name or the dotted field name (if the field is in the embedded document) with a \$

- examples:

- `$primary_name`
- `$known_for_titles.tconst`

```
{
  "_id": "nm0000001",
  "primary_name": "Fred Astaire",
  "birth_year": 1899,
  "death_year": 1987,
  "primary_professions": ["soundtrack", "actor", "miscellaneous"],
  "known_for_titles": [{
    "tconst": "tt0031983"
  }, {
    "tconst": "tt0053137"
  }, {
    "tconst": "tt0050419"
  }, {
    "tconst": "tt0072308"
  }
]
```

Pipeline Expressions – Operator Expressions

- similar to functions that take arguments
 - array of arguments and have the following form:
 - { <operator>: [<argument1>, <argument2> ...] }

```
{ $subtract: [ "$death_year", "$birth_year" ] }
```

```
{ $eq: [ "$birth_year", 1950 ] }
```

- single argument:

- { <operator>: <argument> }

```
{ $size: "$primary_professions" }
```

```
{ $gte: 2000 }
```

Pipeline Expressions – Operator Expressions

- Arithmetic:

- `$abs`, `$round`, `$sqrt`, `$trunc`

- Array:

- `$in`, `$first`, `$size`, `$arrayElemAt`

- Boolean:

- `$and`, `$or`, `$not`

- Comparison:

- `$eq`, `$gt`, `$gte`, `$lt`, `$lte`, `$ne`

- Conditional, Date, Custom Aggregation, Data Size, Literal, Object, Set, Text, Trigonometry, Type, Accumulators, Variable

Pipeline Stages – \$match

- filters the documents to pass only the documents that match the specified condition(s) to the next pipeline stage

```
[{
  $match: {
    birth_year: 1971
  }
}]
```





```
[{
  $match: {
    birth_year: 1971,
    death_year: null
  }
}]
```

```
[{
  $match: {
    $or: [{
      birth_year: 1971
    },
    {
      birth_year: null
    }
  ],
    death_year: null
  }
}]
```

- place the `$match` as early in the aggregation pipeline as possible

Pipeline Stages – \$project

- passes along the documents with the requested fields to the next stage in the pipeline
 - specified fields can be existing fields from the input documents or newly computed fields

	Specification Form	Description
	<code><field>: <1 or true></code>	Specifies the inclusion of a field
	<code>_id: <0 or false></code>	Specifies the suppression of the <code>_id</code> field
	<code><field>: <expression></code>	Adds a new field or resets the value of an existing field
	<code><field>:<0 or false></code>	Specifies the exclusion of a field

Pipeline Stages – \$project

- inclusion of fields `birth_year` and `death_year`, and suppression of `_id`

```
[{
  $project: {
    _id: false,
    birth_year: true,
    death_year: true
  }
}]
```

Pipeline Stages – \$project

- new fields `dead_at_age` and `current_age`

```
[{
  $project: {
    dead_at_age: {
      $subtract: ["$death_year", "$birth_year"]
    }
  }
}]
```

```
[{
  $project: {
    current_age: {
      $ifNull: [{
        $subtract: ["$death_year", "$birth_
      },
      {
        $subtract: [{
          $year: "$$NOW"
        }, "$birth_year"]
      }
    ]
  }
}]
```

Pipeline Stages – \$project

- exclude fields `birth_year` and `death_year`:

```
[{
  $project: {
    birth_year: false,
    death_year: false
  }
}]
```

```
[{
  $project: {
    birth_year: false,
    death_year: false,
    dead_at_age: {
      $subtract: ["$death_year", "$birth_year"]
    }
  }
}]
```

If you specify the exclusion of a field other than `_id`, you cannot employ any other `$project` specification forms.

Pipeline Stages – \$group

- groups input documents by the specified `_id` expression and for each distinct grouping, outputs a document that can contain computed fields holding the values of some accumulator expression (`$avg`, `$sum`, etc).

```
{
  $group: {
    _id: "$birth_year",
    average_death_year: {
      $avg: "$death_year"
    }
  }
}
```

Grouping on single field
(birth_year)

```
{
  $group: {
    _id: {
      birth_year: "$birth_year",
      death_year: "$death_year"
    },
    count_names: {
      $sum: 1
    }
  }
}
```

Grouping on multiple field
(birth_year, death_year)

Pipeline Stages - Usefull for lab

- `$unwind`
 - deconstructs an array field from the input documents to output a document for each element
- `$lookup`
 - performs a left outer join to an unsharded collection in the same database to filter in documents from the “joined” collection for processing
- `$graphLookup`
 - performs a recursive search on a collection, with options for restricting the search by recursion depth and query filter

Aggregation Pipeline – MongoDB Shell

- steps:

- open Mongo shell:

```
mongo
```

- switch database:

```
use <db>
```

- execute aggregation pipeline:

```
db.collection.aggregate(<pipeline>, <options>)
```

- example:

```
db.names.aggregate([{$match: {birth_year: 1971}}, {$project: {_id: false, primary_name: true, death_year: true}}], {allowDiskUse: true})
```


Aggregation Pipeline – MongoDB Compass

The screenshot shows the MongoDB Compass interface with the 'Aggregations' tab selected. The pipeline consists of two stages: \$match and \$project.

Stage 1: \$match

- Stage name: `$match` (toggle is on)
- Output after `$match` stage (Sample of 20 documents)
- Code editor content:

```
1 /**
2  * query: The query in MQL.
3  */
4 {
5   birth_year: 1971
6 }
```
- Output preview shows two documents:

```
{ "_id": "nm0000191", "primary_name": "Ewan McGregor", "birth_year": 1971, "primary_professions": Array, "known_for_titles": Array }
{ "_id": "nm0000213", "primary_name": "Winona Ryder", "birth_year": 1971, "primary_professions": Array, "known_for_titles": Array }
```

Stage 2: \$project

- Stage name: `$project` (toggle is on)
- Output after `$project` stage (Sample of 20 documents)
- Code editor content:

```
1 /**
2  * specifications: The fields to
3  * include or exclude.
4  */
5 {
6   _id: false,
7   primary_name: true,
8   death_year: true
9 }
```
- Output preview shows two documents:

```
{ "primary_name": "Ewan McGregor" }
{ "primary_name": "Winona Ryder" }
```

At the bottom of the interface, there is an 'ADD STAGE' button.

References

- Aggregation pipeline
<https://docs.mongodb.com/manual/core/aggregation-pipeline/>
- Aggregation quick reference: Operator expressions
<https://docs.mongodb.com/manual/meta/aggregation-quick-reference/#agg-quick-ref-operator-expressions>
- Aggregation quick reference: Aggregation accumulator expressions
<https://docs.mongodb.com/manual/meta/aggregation-quick-reference/#aggregation-accumulator-operators>
- db.collection.aggregate
<https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/#db.collection.aggregate>