**Discussion 5.1 – Aggregate Operations**

Patrick Cuauro

Bellevue University

WEB 335-307O Introduction to NoSQL

Professor Krasso

June 20, 2023

## What does the aggregate $project operation do? Provide at least one example.

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

Prototype form:

|  |
| --- |
| { $project: { <specification(s)> } } |

Considerations:

* Including existing fields
* Suppress the “\_id” field. The “\_id” field is not required, because is included by default. But to suppress it, you must explicitly specify it in $project.
* Excluding Fields. If you specifically exclude a field or field, the others will be returned. Also, we can condition the exclusion of fields.

Example:

|  |
| --- |
| db.users.aggregate ( [ { $project : { field1: 1 , field2: 1 , … } } ] ) |

## Provide an example of when the $project operation could be used.

A screenshot of a computer program

Description automatically generated with medium confidence

In this example I used projects to get the selected fields from my document called users. In other words, the filtering of the field I require, in this case I selected the “firstName” and “lastName” fields from my users document, even though there are many more fields in each user. This is the return of this aggregation. You can note the last three users don’t have any field returned, and the cause is because in those users there are no fields with the “firstName” and/or “lastName” on them.

## What does the aggregate $match operation do? Provide at least one example.

Filters the documents to pass only the documents that match the specified condition(s) to the next pipeline stage.

Prototype form:

|  |
| --- |
| { $match: { <query> } } |

Behavior

* Pipeline Optimization

Place the $match early in the aggregation pipeline as possible. It will limit the total number of documents in the pipeline.

* Restrictions

It does not accept raw aggregation expressions. In case to need an inclusion of an expression, $expr must be used in the query.

* The $where expression cannot be used.
* The $near or $nearSphere expressions cannot be used.
* To use $text expression, the $match stage must be the first stage of the pipeline.

Example:

|  |
| --- |
| db.users.aggregate( [ { $match: { lastName : “Bach” } } ] ) |

## Provide an example of when the $match operation could be used.

A screenshot of a computer

Description automatically generated with medium confidence

This is a simple example of how using the $match operation can return the information of a user that matches the query to the pipeline.

## What does the $sort operation do? Provide at least one example.

Sort all input documents and returns them to the pipeline in sorted order.

Prototype form:

|  |
| --- |
| { $sort: { <field1> : <sort order> , <field2> : <sort order> … } ] |

<sort order> can have one of the following values:

1 – Sort ascending.

-1 – Sort descending.

{ $meta: “textScore” } Sort by the computed “textScore” metadata in descending order.

Example:

|  |
| --- |
| db.users.aggregate(  [  { $sort : { email : 1 } }  ]  ) |

* Provide an example of when the $sort operation could be used.

A picture containing text, screenshot, software, font

Description automatically generated

In this case I decided to sort the list of users in ascendent form, as you can see in this example the first three users are returned first because they don’t contain the fields required to sort.

Sources:

<https://www.mongodb.com/docs/manual/reference/operator/aggregation/project/>

<https://www.mongodb.com/docs/manual/reference/operator/aggregation/match/>

<https://www.mongodb.com/docs/manual/reference/operator/aggregation/sort/>