Janis Gonzalez

WEB 335 Introduction to NoSQL

Discussion 5.1 Aggregate Operations

April 13,2023

Bellevue University

The aggregate $project operation passes the documents with the requested fields into the next stage of the lineup. These specified fields in the documents can be fields that already exist from the input documents or newly created fields. The exclusion of fields may also be specified. The $project operation should be used as the last stage in the pipeline which would specify what fields to return to a client (MongoDB Manual). The aggregate $project operation would be typed like this: { $project: { <specification(s)> } } (MongoDB Manual).

One example of this with the document for books (MongoDB Manual):

{

"\_id" : 13465,

title: "Plus Ultra Compendium",

isbn: "0001122223334",

author: { last: "Gerard", first: "Alexandre" },

copies: 5

}

With the project aggregate:

db.books.aggregate( [ { $project : { title : 13465, author : 13465} } ] )

Which would result in the document:

{ "\_id" : 13465, "title" : "Plus Ultra Compendium", "author" : { "last" : "Gerard", "first" : "Alexandre" } }

The aggregate $match operation is used to filter certain documents that would match the specific condition(s) to the next stage in the pipeline. The $match stage would be typed like: { $match: { <query> } } (MongoDB Manual). The aggregate $match operation should be placed early in the pipeline as possible since it limits the total amount of documents and will minimize the amount of processing that has to take place (MongoDB Manual).

The following collection named articles includes the following documents (MongoDB Manual):

{ "\_id" : ObjectId("512bc95fe835e68f199c8686"), "author" : "aly", "score" : 70, "views" : 235}

{ "\_id" : ObjectId("512bc962e835e68f199c8687"), "author" : "aly", "score" : 90, "views" : 346}

{ "\_id" : ObjectId("55f5a192d4bede9ac365b257"), "author" : "lero", "score" : 65, "views" : 467}

{ "\_id" : ObjectId("55f5a192d4bede9ac365b258"), "author" : "freO", "score" :98, "views" : 555 }

{ "\_id" : ObjectId("55f5a1d3d4bede9ac365b259"), "author" : "greO", "score" : 53, "views" : 976}

{ "\_id" : ObjectId("55f5a1d3d4bede9ac365b25a"), "author" : "po", "score" : 76, "views" : 657}

{ "\_id" : ObjectId("55f5a1d3d4bede9ac365b25b"), "author" : "ser", "score" : 56, "views" : 156 }

The aggregate $match operation to perform the equality match that selects documents where the author field equals greO:

db.articles.aggregate(

[ { $match : { author : "greO" } } ]

);

This is what the aggregate $match operation returns:

{ "\_id" : ObjectId("55f5a1d3d4bede9ac365b259"), "author" : "greO", "score" : 53, "views" : 976}

The aggregate $sort operation sorts all the input documents and returns them in the sorted order. You can only sort a maximum of 32 keys. This aggregate $sort operation is written like: { $sort: { <field1>: <sort order>, <field2>: <sort order> ... } } (Sort aggregation in mongodb).

An example of this is with a student collection (Sort aggregation in mongodb):

{ "\_id" : 1, "name" : "rommel", "age" : 21, "email" : "rommel@example.com" }  
 { "\_id" : 2, "name" : "joseph", "age" : 24, "email" : "joseph@example.com" }  
 { "\_id" : 3, "name" : "jokoy", "age" : 22, "email" : "jokoy@example.com" }  
 { "\_id" : 4, "name" : "dan", "age" : 22, "email" : "dan@example.com" }  
 { "\_id" : 5, "name" : "steve", "age" : 21, "email" : "steve@example.com" }  
 { "\_id" : 6, "name" : "james", "age" : 24, "email" : "james@example.com" }  
 { "\_id" : 7, "name" : "greg", "age" : 23, "email" : "greg@example.com" }

With the aggregate $sort operation:

db.student.aggregate(  
 [  
 { $sort : { age : 1} }  
 ]  
)

It will return the following ascending order (Sort aggregation in mongodb):

{ "\_id" : 1, "name" : "rommel", "age" : 21, "email" : "rommel@example.com" }  
 { "\_id" : 5, "name" : "steve", "age" : 21, "email" : "steve@example.com" }  
 { "\_id" : 3, "name" : "jokoy", "age" : 22, "email" : "jokoy@example.com" }  
 { "\_id" : 4, "name" : "dan", "age" : 22, "email" : "dan@example.com" }  
 { "\_id" : 7, "name" : "greg", "age" : 23, "email" : "greg@example.com" }  
 { "\_id" : 2, "name" : "joseph", "age" : 24, "email" : "joseph@example.com" }  
 { "\_id" : 6, "name" : "james", "age" : 24, "email" : "james@example.com" }

References:

*MongoDB Manual*. MongoDB Manual. (n.d.). Retrieved from <https://www.mongodb.com/docs/manual/reference/operator/aggregation/sort/>

*Sort aggregation in mongodb*. ObjectRocket Sort Aggregation in MongoDB Comments. (n.d.). Retrieved from <https://kb.objectrocket.com/mongo-db/sort-aggregation-in-mongodb-1349>