

Class 5:

Projection Operators

Projection:

In MongoDB, projection means selecting only the necessary data rather than selecting whole of the data of a document. If a document has 5 fields and you need to show only 3, then select only 3 fields from them. MongoDB projection uses the same find syntax, but we also add a set of parameters to the find function. This set of parameters informs the MongoDB instance of which fields to be returned.

MongoDB projection method positively impacts database performance as it reduces the workload of the find query when trying to retrieve specific data from a document, minimizing resource usage.

To enhance the querying and reduce the workload, multiple operators can be used within a projection query like the ones below:

- \$
- \$elemMatch
- \$slice

Name	Description
\$	Projects the first element in an array that matches the query condition.
\$elemMatch	Projects the first element in an array that matches the specified \$elemMatch condition.
\$meta	Projects the available per-document metadata.
\$slice	Limits the number of elements projected from an array. Supports skip and limit slices.

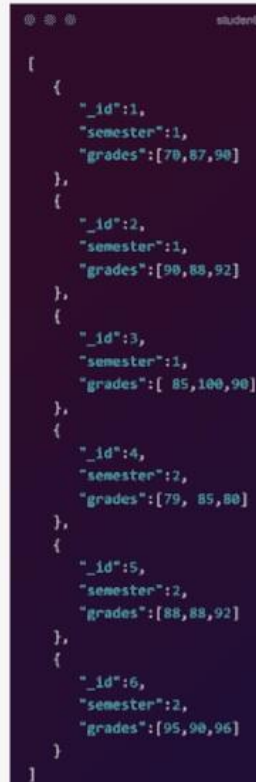
1. \$ Operator:

The \$ operator is utilized in scenarios where there is a need to limit an array to project only the first element that matches the condition of the query, if no query condition is present, the first element is returned in the specified MongoDB projection array, and a sample syntax of the same can be seen below:

```
db.collection.find( { <array>: <condition> ... }, { "<array>.$": 1 } )
```

Example:

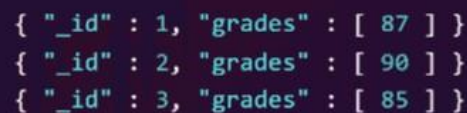
Let us check out the "students" Collection this time to understand the functioning of the \$ operator.



```
{
  {
    "_id":1,
    "semester":1,
    "grades":[70,87,90]
  },
  {
    "_id":2,
    "semester":1,
    "grades":[90,88,92]
  },
  {
    "_id":3,
    "semester":1,
    "grades":[ 85,100,90]
  },
  {
    "_id":4,
    "semester":2,
    "grades":[79, 85,80]
  },
  {
    "_id":5,
    "semester":2,
    "grades":[88,88,92]
  },
  {
    "_id":6,
    "semester":2,
    "grades":[95,90,96]
  }
}
```

```
db.students.find( { semester: 1, grades: { $gte: 85 } }, { "grades.$": 1 } )
```

Using the \$ operator will only return the MongoDB projection first element of array where it is equal to or greater than 85.



```
{ "_id" : 1, "grades" : [ 87 ] }
{ "_id" : 2, "grades" : [ 90 ] }
{ "_id" : 3, "grades" : [ 85 ] }
```

Limitations of the \$ operator:

- In a single MongoDB projection query, only a single \$ operator can be used.

- It will only be applied to a single condition on the array field
- The sort() function in the find() is applied before the \$ operator and this function may cause the sort order to get distorted.

2. \$elemMatch Operator:

Similar to the \$ operator, the \$elemMatch operator also limits the contents of an array to the first element that fits the given constraint. Though, there is a minor difference from the \$ operator because the \$elemMatch projection operator needs an explicit condition argument.

Syntax as follows:

```
db.collection.find( { <array>: <condition> ... }, { "<array>.$elemMatch": ($elemMatch operator) } )
```

Example:

Below you can check the "schoolsData" collection that we will use to demonstrate the \$elemMatch operator.

```
SchoolsData
{
  {
    "_id":1,
    "zipcode":"63109",
    "students":[
      {"name":"john","school":102,"age":10},
      {"name":"jess","school":102,"age":11},
      {"name":"jeff","school":108,"age":15}
    ]
  },
  {
    "_id":2,
    "zipcode":"63110",
    "students":[
      {"name":"ajax","school":100,"age":7},
      {"name":"achilles","school":100,"age":8}
    ]
  },
  {
    "_id":3,
    "zipcode":"63109",
    "students":[
      {"name":"ajax","school":100,"age":7 },
      {"name":"achilles","school":100,"age":8}
    ]
  },
  {
    "_id":4,
    "zipcode":"63109",
    "students":[
      {"name":"barney","school":102, "age":7},
      {"name":"ruth","school":102,"age":16}
    ]
  }
}
```

```
db.schools.find( { zipcode: "63109" }, { students: { $elemMatch: { school: 102 } } } )
```

```
SchoolsDataWithElemMatch
{ "_id" : 1, "students" : [ { "name" : "john", "school" : 102, "age" : 10 } ] }
{ "_id" : 3 }
{ "_id" : 4, "students" : [ { "name" : "barney", "school" : 102, "age" : 7 } ] }
```

Limitations of \$elemMatch operator are as follows:

- The field to which the \$elemMatch projection is applied is returned as the last field of the document, regardless of the order in which the fields are ordered.
- \$elemMatch projection operator is not supported by find() operations on MongoDB views.
- The \$elemMatch operator does not handle \$text query expressions.

3. \$slice Projection Operator:

The \$slice operator bounds the number of elements that should be returned as the output of a MongoDB projection query.

Syntax:

```
db.collection.find( <query>, { <array Field>: { $slice: <number> } } )
```

Now we'll use the "postsData" collection to demonstrate the \$slice operator.

```
PostsData
[
  {
    _id: 1,
    title: "Bagels are not croissants.",
    comments: [ { comment: "0. true" }, { comment: "1. croissants aren't bagels." } ]
  },
  {
    _id: 2,
    title: "Coffee please.",
    comments: [ { comment: "0. foey" }, { comment: "1. tea please" }, { comment: "2. iced coffee" }, { comment: "3. cappuccino" }, { comment: "4. whatever" } ]
  }
]
```

We want to get data from the comments array to return the array with its first three elements. If the array contains less than three elements, all elements of the array are returned.

```
db.posts.find( {}, { comments: { $slice: 3 } } )
```



```
PostsDataWith$lice
{
  {
    "_id" : 1,
    "title" : "Bagels are not croissants.",
    "comments" : [ { "comment" : "0. true" }, { "comment" : "1. croissants aren't bagels." } ]
  }
  {
    "_id" : 2,
    "title" : "Coffee please.",
    "comments" : [ { "comment" : "0. foey" }, { "comment" : "1. tea please" }, { "comment" : "2. iced
coffee" } ]
  }
}
```

Limitations in \$slice operator:

- In a nested array the **\$slice** operator will only return the sliced element and will not return any other item, especially from MongoDB 4.4.
- The **find()** action is not supported by the **\$slice** operator on MongoDB views.
- Due to a constraint imposed by MongoDB, the **\$slice** operator cannot be combined with the **\$** projection operator. This is because top-level fields are not permitted to contain the **\$** symbol as part of the field name.