Class 5:
Projection Operators

In MongoDB, projection operators are used to control the inclusion, exclusion, and
transformation of fields in the documents returned by a query. Here are the common projection operators available in MongoDB:
Create and demonstrate how projection operators ($,$elements and $slice) would
be used in the MongoDB.

Field Inclusion and Exclusion Operators:

1.Inclusion: Include specific fields in the output.
db.collection.find({}, { field1: 1, field2: 1 })

2.Exclusion: Exclude specific fields from the output.
db.collection.find({}, { field1: 0, field2: 0 })

Positional Operators:

3.$elemMatch: Projects the first matching element from an array based on a specified condition.
db.collection.find({}, { arrayField: { $elemMatch: { field: value } } })

4.$slice: Limits the number of elements projected from an array.
db.collection.find({}, { arrayField: { $slice: [skip, limit] } })

5.$: Projects the first element in an array that matches the query condition.
db.collection.find({ "arrayField.field": value }, { "arrayField.$": 1 })

Computed Fields and Expressions:

6.$meta: Projects the metadata (e.g., text search score) for a document.
db.collection.find({}, { score: { $meta: "textScore" } })

7.$slice: Projects a subset of an array.
db.collection.find({}, { arrayField: { $slice: limit } })

Aggregation Framework Operators:
When using the aggregation framework, you can use the $project stage to include/exclude fields and add computed fields.

8.$project: Used in the aggregation pipeline to reshape each document in the stream.
db.collection.aggregate([
{ $project: { field1: 1, field2: 1, computedField: { $add: ["$fieldA", "$fieldB"] } } }
]);

An example of using projection operators in MongoDB:

Assume we have a MongoDB collection named `users` with the following documents:

```
{
  "_id": 1,
  "name": "Alice",
  "age": 30,
  "email": "alice@example.com",
  "address": "123 Main St"
},
{
  "_id": 2,
  "name": "Bob",
  "age": 25,
  "email": "bob@example.com",
  "address": "456 Oak Ave"
}
```

Example 1: Including Specific Fields

Let's say we want to retrieve only the name and email fields for each user, excluding the rest.

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.

studentgrades

```
[
    {
        "_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.

studentgrades

```
{ "_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({ : ... }, { ".$elemMatch": ($elemMatch operator) })

Example:
Below you can check the "schoolsdData" collection that we will use to demonstrate the $elemmMatch 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 } } } )
```

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.

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(, { : { $slice: } })

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

```
[
   {
     _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. fooey" }, { 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.

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.