CLASS-5

PROJECTION OPERATORS

Projection operators in MongoDB are used to select specific fields from the documents in a collection. They are used in the find() method to specify which fields to include or exclude from the result set.

There are several types of projection operators in MongoDB:

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.

Include and Exclude Fields

You can also use the `include` and `exclude` fields to specify which fields to include or exclude from the result set.

The syntax for including fields:

{ field1: 1, field2: 1, ... }

Example:

To retrieve name, age, and blood group from the candidate database

```
db> db.candidates.find({},{name: 1,age:1, blood_group:1});
[
  {
    _id: ObjectId('66968c49c2639361c5b5f4c6'),
    name: 'Alice Smith',
    age: 20,
    blood_group: 'A+'
  },
  {
    _id: ObjectId('66968c49c2639361c5b5f4c7'),
    name: 'Bob Johnson',
    age: 22,
    blood_group: 'O-'
  },
  {
    _id: ObjectId('66968c49c2639361c5b5f4c8'),
    name: 'Charlie Lee',
    age: 19,
    blood_group: 'B+'
  },
  {
    _id: ObjectId('66968c49c2639361c5b5f4c9'),
    name: 'Emily Jones',
    age: 21,
    blood_group: 'AB-'
  },
```

The syntax for excluding fields is:

{ field1: 0, field2: 0, ... }

Example:

Exclude id and courses from the candidates' database

```
db> db.candidates.find({},{_id:0, courses:0});
[
  {
    name: 'Alice Smith',
    age: 20,
    gpa: 3.4,
    home_city: 'New York City',
    blood_group: 'A+',
    is_hotel_resident: true
  },
  {
    name: 'Bob Johnson',
    age: 22,
    gpa: 3.8,
    home_city: 'Los Angeles',
    blood_group: 'O-',
    is_hotel_resident: false
  },
  {
    name: 'Charlie Lee',
    age: 19,
    gpa: 3.2,
    home_city: 'Chicago',
    blood_group: 'B+',
    is_hotel_resident: true
  },
  {
```

$elemMatch

The $elemMatch operator selects a single element from an array field that matches a specified condition. It is often used to select a specific element from an array of sub-documents.

Syntax:
db.collection.find({}, { field: { $elemMatch: { condition } } })

Example: Consider a `players` collection with the following document:

```
db.players.insertOne( {
   name: "player1",
   games: [ { game: "abc", score: 8 }, { game: "xyz", score: 5 } ],
   joined: new Date("2020-01-01"),
   lastLogin: new Date("2020-05-01")
} )
```

The following projection returns the `games` field after the other existing fields included in the projection even though in the document, the field is listed before `joined` and `lastLogin` fields:

db.players.find({}, { games: { $elemMatch: { score: { $gt: 5 } } }, joined: 1, lastLogin: 1 })

That is, the operation returns the following document:

```
{
   "_id" : ObjectId("5edef64a1c099fff6b033977"),
   "joined" : ISODate("2020-01-01T00:00:00Z"),
   "lastLogin" : ISODate("2020-05-01T00:00:00Z"),
   "games" : [ { "game" : "abc", "score" : 8 } ]
}
```

$meta

The $meta operator is used to access metadata values from the documents in a collection. It can be used to access the text score of a document, which is a value that indicates how well the document matches a search query.

Syntax:

```
db.collection.find({}, { field: { $meta: "textScore" } })
```

$slice

The $slice operator is used to select a specified number of elements from an array field. It can be used to limit the number of elements returned from an array.

Syntax:

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

Or

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

Example:

Database:

```
db.posts.insertMany([
   {
     _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" }, { commen
   }
])
```

Return an Array with Its First 3 Elements

The following operation uses the `$slice` projection operator on the `comments` array to return the array with its first three elements. If the array has less than three elements, all elements in the array are returned.

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

The operation returns the following documents:

```
{
   "_id" : 1,
   "title" : "Bagels are not croissants.",
   "comments" : [ { "comment" : "0. true" }, { "comment" : "1. croissants aren't
}
{
   "_id" : 2,
   "title" : "Coffee please.",
   "comments" : [ { "comment" : "0. fooey" }, { "comment" : "1. tea please" }, {
}
```

Return an Array with Its Last 3 Elements

The following operation uses the `$slice` projection operator on the `comments` array to return the array with its last three elements. If the array has less than three elements, all elements in the array are returned.

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

The operation returns the following documents:

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

Return an Array with 3 Elements After Skipping the First Element

The following operation uses the `$slice` projection operator on the `comments` array to:

- Skip the first element such that the second element is the starting point.
- Then, return three elements from the starting point.

If the array has less than three elements after the skip, all remaining elements are returned.

```
db.posts.find( {}, { comments: { $slice: [ 1, 3 ] } } )
```

The operation returns the following documents:

```
{
   "_id" : 1,
   "title" : "Bagels are not croissants.",
   "comments" : [ { "comment" : "1. croissants aren't bagels." } ]
}

{
   "_id" : 2,
   "title" : "Coffee please.",
   "comments" : [ { "comment" : "1. tea please" }, { "comment" : "2. iced coffee
}
```