# MongoDB

## Projection Operators

Projection operators in MongoDB are used to specify or restrict the fields to return in the documents retrieved from a collection. They are primarily used within the $project stage of an aggregation pipeline or in the find method to shape the documents.

Here are some of the key projection operators in MongoDB:

**Basic Projection**

In the find method, you can use projection to include or exclude fields:

- **Include Fields**: Specify the fields to include with a value of 1

```
db.collection.find({}, { name: 1, age: 1 })
```

This will return documents with only the name and age fields (plus the _id field by default).

- **Exclude Fields**: Specify the fields to exclude with a value of 0.

```
db.collection.find({}, { password: 0 })
```

- This will return documents without the password field.

**$project Stage in Aggregation Pipeline**

The $project stage in an aggregation pipeline reshapes each document by including, excluding, or adding new fields.

**Inclusion and Exclusion**

**Inclusion**: Specify fields to include with a value of 1.

```
db.collection.aggregate([
  { $project: { name: 1, age: 1 } }
])
```

This includes only the name and age fields.

- **Exclusion**: Specify fields to exclude with a value of 0

```
db.collection.aggregate([
  { $project: { password: 0 } }
])
```

This excludes the password field.

**Adding New Fields**

- **Adding Constant Fields**: Add a new field with a constant value.

```
db.collection.aggregate([
  { $project: { name: 1, age: 1, active: { $literal: true } } }
])
```

This adds an active field with a constant value true.

- **Adding Computed Fields**: Add a new field with a computed value.

```
db.collection.aggregate([
  { $project: { name: 1, age: 1, fullName: { $concat: ["$firstName", " ", "$lastName"]
])
```

This adds a fullName field that concatenates firstName and lastName.

**Projection Operators**

**$slice**

- Limits the number of elements in an array field

```
db.collection.aggregate([
  { $project: { name: 1, firstTwoItems: { $slice: ["$items", 2] } } }
])
```

This includes only the first two elements of the items array.

**$elemMatch**

- Projects the first element in an array that matches the specified condition

```
db.collection.aggregate([
  { $project: { name: 1, matchedItem: { $elemMatch: { items: { type: "A" } } } } }
])
```

This includes only the first element in the items array where type is "A".

**$filter**

- Selects a subset of an array to return based on the specified condition

```
db.collection.aggregate([
  { $project: { name: 1, filteredItems: { $filter: { input: "$items", as: "item", cond
])
```

This includes only elements in the items array where price is greater than 10.

**$map**

- Applies an expression to each element in an array and returns the resulting array

```
db.collection.aggregate([
  { $project: { name: 1, discountedPrices: { $map: { input: "$prices", as: "price", in
])
```

This includes a discountedPrices field where each element in the prices array is multiplied by 0.9.

**Example**

Here is a more comprehensive example using various projection operators in the aggregation pipeline:

```
db.collection.aggregate([
  { $project: {
      name: 1,
      totalAmount: { $sum: "$items.amount" },
      discountedPrices: { $map: { input: "$prices", as: "price", in: { $multiply: ["$$pric
      matchedItem: { $elemMatch: { items: { type: "A" } } },
      firstTwoItems: { $slice: ["$items", 2] },
      filteredItems: { $filter: { input: "$items", as: "item", cond: { $gt: ["$$item.price
    }
  }
])
```

In this pipeline:

- name is included.
- totalAmount is a new field that sums the amount in items.
- discountedPrices applies a 10% discount to each element in the prices array.
- matchedItem includes the first element in items with type "A".
- firstTwoItems includes the first two elements in items.
- filteredItems includes elements in items with price greater than 10.

Projection operators in MongoDB allow for flexible and powerful document reshaping to meet various application needs.

EXAMPLE

```
test> db.candidate.find({},{name:1,age:1,gpa:1}).count();
12
test>
```

---

VARIATION:EXCLUDE FIELDS

EXAMPLE

```
test> db.candidate.find({},{_id:0,courses:0}).count();
12
test>
```

---

## $elemMatch(projection)

**Example:To find candidates enrolled in"Computer Science"with specific projection**

```
test> db.candidate.find({courses:{$elemMatch:{$eq:"Computer Science"}}},{name:1,"courses.$":1});
[
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae3'),
    name: 'Bob Johnson',
    courses: [ 'Computer Science' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae8'),
    name: 'Gabriel Miller',
    courses: [ 'Computer Science' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064aec'),
    name: 'Kevin Lewis',
    courses: [ 'Computer Science' ]
  }
```

$elemMatch:

```
test> 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")
... } )
{
  acknowledged: true,
  insertedId: ObjectId('6682dbaeec4204f7fecdcdf6')
}
```

```
test> db.players.find({},{games:{$elemMatch:{score:{$gt:5}}},joined:1,lastlogin:1})
;
[
  {
    _id: ObjectId('6682dbaeec4204f7fecdcdf6'),
    joined: ISODate('2020-01-01T00:00:00.000Z'),
    games: [ { game: 'abc', score: 8 } ]
  }
```

$slice(projection):

Example:To retrieve all candidates with first too courses.

```
test> db.candidate.find({},{name:1,courses:{$slice:2}});
[
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae2'),
    name: 'Alice Smith',
    courses: [ 'English', 'Biology' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae3'),
    name: 'Bob Johnson',
    courses: [ 'Computer Science', 'Mathematics' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae4'),
    name: 'Charlie Lee',
    courses: [ 'History', 'English' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae5'),
    name: 'Emily Jones',
    courses: [ 'Mathematics', 'Physics' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae6'),
    name: 'David Williams',
    courses: [ 'English', 'Literature' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae7'),
    name: 'Fatima Brown',
    courses: [ 'Biology', 'Chemistry' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae8'),
    name: 'Gabriel Miller',
    courses: [ 'Computer Science', 'Engineering' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064ae9'),
    name: 'Hannah Garcia',
    courses: [ 'History', 'Political Science' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064aea'),
    name: 'Isaac Clark',
    courses: [ 'English', 'Creative Writing' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064aeb'),
    name: 'Jessica Moore',
    courses: [ 'Biology', 'Ecology' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064aec'),
    name: 'Kevin Lewis',
    courses: [ 'Computer Science', 'Artificial Intelligence' ]
  },
  {
    _id: ObjectId('6682d28ec38b2a00d2064aed'),
    name: 'Lily Robinson',
    courses: [ 'History', 'Art History' ]
  }
]
```