ADVANCED PROJECTIONS: $ELEMMATCH AND $SLICE:

$ELEMMATCH OPERATOR:

The **$elemMatch** operator projects the first element in an array that matches the specified query condition. This is particularly useful when you only need to return specific elements of an array that satisfy certain criteria, rather than the entire array.

Syntax:

db.collection.find(
{ query },
{ arrayField: { $elemMatch: { condition } } }
)

Let's perform the $elemMatch on the new dataset called “movies”.

```
test> use db
switched to db db
db> show collections
candidates
locations
movies
students
students_permission
db> db.movies.find(
... {"cast":{$elemMatch:{"name":"Al Pacino"}}}
... )
[
  {
    _id: '573a1390f29313caabcd4135',
    title: 'The Godfather',
    year: 1972,
    genres: [ 'Crime', 'Drama' ],
    cast: [
      { name: 'Marlon Brando', role: 'Don Vito Corleone' },
      { name: 'Al Pacino', role: 'Michael Corleone' },
      { name: 'James Caan', role: 'Sonny Corleone' }
    ],
    rating: { average: 9.2, reviews: 1000 }
  }
]
db> db.movies.find( { "cast": { $elemMatch: { "role":"Joker" } } } )
[
  {
    _id: '573a1390f29313caabcd42e8',
    title: 'The Dark Knight',
    year: 2008,
    genres: [ 'Action', 'Crime', 'Drama' ],
    cast: [
      { name: 'Christian Bale', role: 'Bruce Wayne / Batman' },
      { name: 'Heath Ledger', role: 'Joker' },
      { name: 'Aaron Eckhart', role: 'Harvey Dent' }
    ],
    rating: { average: 9, reviews: 1500 }
  }
]
db>
```

Query 1: Find movies where Al Pacino is in the cast:

Explanation:

1. db.movies.find(...): This part of the query tells MongoDB to search in the movies collection.
2. "cast": This specifies that we are interested in the cast field of each document. The cast field is an array of objects, each representing a cast member with name and role properties.
3. $elemMatch: This operator is used to specify criteria that at least one element in the array must satisfy.
4. { "name": "Al Pacino" }: This is the condition inside the $elemMatch operator. It specifies that we are looking for an element (i.e., a cast member) within the cast array where the name field is "Al Pacino".

What it does:

- MongoDB scans each document in the movies collection.
- For each document, it looks at the cast array.
- If there is at least one element in the cast array with "name": "Al Pacino", that document is included in the result set.

Result:

This query will return the document for the movie "The Godfather" because Al Pacino is a cast member.

Query 2: Find movies with a cast member playing the role of "Joker"

Explanation:

1. db.movies.find(...): This part of the query tells MongoDB to search in the movies collection.
2. "cast": This specifies that we are interested in the cast field of each document. The cast field is an array of objects, each representing a cast member with name and role properties.
3. $elemMatch: This operator is used to specify criteria that at least one element in the array must satisfy.
4. { "role": "Joker" }: This is the condition inside the $elemMatch operator. It specifies that we are looking for an element (i.e., a cast member) within the cast array where the role field is "Joker".

What it does:

- MongoDB scans each document in the movies collection.
- For each document, it looks at the cast array.
- If there is at least one element in the cast array with "role": "Joker", that document is included in the result set.

Result:

This query will return the document for the movie "The Dark Knight" because Heath Ledger played the role of Joker.

Both queries demonstrate how $elemMatch can be effectively used to filter documents based on criteria within array elements.

$SLICE OPERATOR:

The **$slice** operator is used in MongoDB queries to limit the number of elements returned from an array field. It can be used in the projection part of a query to include only a subset of the array elements in the result.

Syntax of $slice Operator:
There are three ways to use $slice:

1. Return the first N elements of an array:
- **syntax:** { "arrayField": { $slice: N } }
- **N:** The number of elements to return from the start of the array.
2. Return the last N elements of an array:
- **syntax:**{ "arrayField": { $slice: -N } }
- **-N:** The number of elements to return from the end of the array.
3. Return N elements starting from position skip:
- **syntax:** { "arrayField": { $slice: [skip, N] } }
- **skip:** The position to start returning elements from (0-based index).
- **N:** The number of elements to return from the skip position.

This will provide you with a complete view of how the data is structured and stored in your collection, allowing you to verify the data before performing further operations like $slice.

```
db> db.movies.find()
[
  {
    _id: '573a1390f29313caabcd4135',
    title: 'The Godfather',
    year: 1972,
    genres: [ 'Crime', 'Drama' ],
    cast: [
      { name: 'Marlon Brando', role: 'Don Vito Corleone' },
      { name: 'Al Pacino', role: 'Michael Corleone' },
      { name: 'James Caan', role: 'Sonny Corleone' }
    ],
    rating: { average: 9.2, reviews: 1000 }
  },
  {
    _id: '573a1390f29313caabcd42e8',
    title: 'The Dark Knight',
    year: 2008,
    genres: [ 'Action', 'Crime', 'Drama' ],
    cast: [
      { name: 'Christian Bale', role: 'Bruce Wayne / Batman' },
      { name: 'Heath Ledger', role: 'Joker' },
      { name: 'Aaron Eckhart', role: 'Harvey Dent' }
    ],
    rating: { average: 9, reviews: 1500 }
  },
  {
    _id: '573a1390f29313caabcd4136',
    title: 'Pulp Fiction',
    year: 1994,
    genres: [ 'Crime', 'Drama' ],
    cast: [
      { name: 'John Travolta', role: 'Vincent Vega' },
      { name: 'Samuel L. Jackson', role: 'Jules Winnfield' },
      { name: 'Uma Thurman', role: 'Mia Wallace' }
    ],
    rating: { average: 8.9, reviews: 1100 }
  },
```

Example 1: Return the first 2 genres of each movie:

```
db.movies.find(
  {},
  { "title": 1, "cast": { $slice: 2 }, "_id": 0 }
).pretty()
```

Explanation:

- {}: Matches all documents in the collection.
- { "title": 1, "genres": { $slice: 2 }, "_id": 0 }: This projection includes only the title and the first two elements of the genres array in the results, excluding the _id.

```
db> db.movies.find( {}, { "title": 1, "cast": { $slice: 2 }, "_id": 0 } ).pretty()
[
  {
    title: 'The Godfather',
    cast: [
      { name: 'Marlon Brando', role: 'Don Vito Corleone' },
      { name: 'Al Pacino', role: 'Michael Corleone' }
    ]
  },
  {
    title: 'The Dark Knight',
    cast: [
      { name: 'Christian Bale', role: 'Bruce Wayne / Batman' },
      { name: 'Heath Ledger', role: 'Joker' }
    ]
  },
  {
    title: 'Pulp Fiction',
    cast: [
      { name: 'John Travolta', role: 'Vincent Vega' },
      { name: 'Samuel L. Jackson', role: 'Jules Winnfield' }
    ]
  },
  {
    title: 'Inception',
    cast: [
      { name: 'Leonardo DiCaprio', role: 'Dom Cobb' },
      { name: 'Joseph Gordon-Levitt', role: 'Arthur' }
    ]
  },
  {
    title: 'Fight Club',
    cast: [
      { name: 'Brad Pitt', role: 'Tyler Durden' },
      { name: 'Edward Norton', role: 'The Narrator' }
    ]
```

Example 2: Return the last 2 cast members of each movie:

```
db.movies.find(
  {},
  { "title": 1, "cast": { $slice: -2 }, "_id": 0 }
).pretty()
```

Explanation:

- {}: Matches all documents in the collection.
- { "title": 1, "cast": { $slice: -2 }, "_id": 0 }: This projection includes only the title and the last two elements of the cast array in the results, excluding the _id.

```
db> db.movies.find( {}, { "title": 1, "cast": { $slice: -2 }, "_id": 0 } ).pretty()
[
  {
    title: 'The Godfather',
    cast: [
      { name: 'Al Pacino', role: 'Michael Corleone' },
      { name: 'James Caan', role: 'Sonny Corleone' }
    ]
  },
  {
    title: 'The Dark Knight',
    cast: [
      { name: 'Heath Ledger', role: 'Joker' },
      { name: 'Aaron Eckhart', role: 'Harvey Dent' }
    ]
  },
  {
    title: 'Pulp Fiction',
    cast: [
      { name: 'Samuel L. Jackson', role: 'Jules Winnfield' },
      { name: 'Uma Thurman', role: 'Mia Wallace' }
    ]
  },
  {
    title: 'Inception',
    cast: [
      { name: 'Joseph Gordon-Levitt', role: 'Arthur' },
      { name: 'Ellen Page', role: 'Ariadne' }
    ]
  },
  {
    title: 'Fight Club',
    cast: [
      { name: 'Edward Norton', role: 'The Narrator' },
      { name: 'Helena Bonham Carter', role: 'Marla Singer' }
    ]
```

Example 3: Return 2 cast members starting from the second one (index 1):

```
db.movies.find(
  {},
  { "title": 1, "cast": { $slice: [1,2] }, "_id": 0 }
).pretty()
```

Explanation:

- {}: Matches all documents in the collection.
- { "title": 1, "cast": { $slice: [1, 2] }, "_id": 0 }: This projection includes only the title and two elements of the cast array starting from index 1 in the results, excluding the _id.
- The .pretty() method formats the output to be more readable.

By mastering $elemMatch and $slice, you can perform advanced data retrieval operations, ensuring your queries are both powerful and efficient. These operators are essential tools in the MongoDB query arsenal, providing flexibility and precision in data manipulation and retrieval. Whether you're working with complex datasets or simply need to fine-tune your queries, understanding these advanced projections will significantly enhance your ability to interact with MongoDB effectively.