Cursor Helper Methods in MongoDB

MongoDB offers several helper methods that allow us to control the cursor's behavior when fetching documents from a collection. Below are the most commonly used cursor methods:

---

1. `limit()`:

The `limit()` method restricts the number of documents returned from a query.

Example:

```
db.learners.find().limit(1);
db.learners.find().limit(2);
```

Output:

```
storedb> db.learners.find().limit(1);
[
  { _id: 6, name: 'Dhanaraju', marks: 60 }
]

storedb> db.learners.find().limit(2);
[
  { _id: 6, name: 'Dhanaraju', marks: 60 },
  { _id: 13, name: 'Pranav', marks: 30 }
]

storedb> db.learners.find().limit(5);
[
  { _id: 6, name: 'Dhanaraju', marks: 60 },
  { _id: 13, name: 'Pranav', marks: 30 },
  { _id: 14, name: 'Meera', marks: 40 },
  { _id: 15, name: 'Vinay', marks: 50 },
  { _id: 16, name: 'Sarika', marks: 60 }
]
```

---

2. `skip()`:

The `skip()` method is used to skip a specified number of documents from the result.

Example:

```
db.learners.find().skip(3);
```

To skip 10 documents and retrieve the next 10:

```
db.learners.find().skip(10).limit(10);
```

Use Case: Pagination in web applications can be achieved using `skip()` and `limit()` methods. For example, if you want to display 10 documents per page:

- **1st page:** `db.learners.find().limit(10)`
- **2nd page:** `db.learners.find().skip(10).limit(10)`
- **3rd page:** `db.learners.find().skip(20).limit(10)`

---

3. `sort()`:

The `sort()` method arranges the documents in ascending or descending order based on the value of a specified field.

Syntax:

```
sort({ field: 1 })    // Ascending
sort({ field: -1 })   // Descending
```

Examples:

- **Q1. Display learners in ascending order of marks:**

  ```
  db.learners.find().sort({marks: 1}).pretty();
  ```

- **Q2. Display learners in descending order of marks:**

  ```
  db.learners.find().sort({marks: -1}).pretty();
  ```

- **Q3. Display learners in alphabetical order of names:**

  ```
  db.learners.find().sort({name: 1}).pretty();
  ```

Sorting Based on Multiple Fields:

You can also sort based on multiple fields. If two documents have the same value for the first field, the second field is used to break the tie.

Example: To sort learners based on marks (ascending) and if two learners have the same marks, sort based on the reverse alphabetical order of names:

```
db.learners.find().sort({marks: 1, name: -1});
```

Combining Cursor Methods:

Cursor methods such as `limit()`, `skip()`, and `sort()` can be chained together.

Example:

```
db.learners.find().sort({name: -1}).skip(100).limit(15);
```

Note: The methods are executed from left to right, so the order of operations matters.

Pagination with Sorting:

For paginated results with sorting, you can sort the data and apply the `skip()` and `limit()` methods.

Example:

If you want to paginate learners in alphabetical order of names, and if two learners have the same name, sort by marks (ascending). Display only 15 documents per page:

- **1st page:**

  ```
  db.learners.find().sort({name: 1, marks: 1}).limit(15).pretty();
  ```

- **2nd page:**

  ```
  db.learners.find().sort({name: 1, marks: 1}).skip(15).limit(15).pretty();
  ```

- **3rd page:**

  ```
  db.learners.find().sort({name: 1, marks: 1}).skip(30).limit(15).pretty();
  ```

This method can be used to implement efficient pagination with sorting for larger datasets.

These methods allow for flexible querying and efficient pagination while controlling the output's order and number of documents.