1. **How would you implement a pagination API that needs to retrieve records from a high-volume database?**

Pagination is a method of separating and displaying fetched records into discrete pages. There are different methods we can use to achieve pagination, which method to choose depends on the requirement as each method have its own pros and cons.

As asked, to achieve pagination for high volume data we can use the **Seek pagination** or sort pagination.

We perform this pagination on ID field by sorting the records on ID and querying the DB either on before\_id or after\_id depends on if dataset is sorted in descending order or ascending order respectively.

**Example**: To fetch the result following will the 3 consecutive requests for 3 pages:

**Limit**: number of records needs to be displayed on each page.

GET items?limit=20: will fetch only most recent first 20 rows from the database.

Query: SELECT \* FROM Items WHERE Id<=20 order by Id ASC LIMIT 20

GET /items?limit=20&after\_id=20:

Query: SELECT \* FROM Items WHERE Id>20 order by Id ASC LIMIT 20

GET /items?limit=20&after\_id=40:

Query: SELECT \* FROM Items WHERE Id>40 order by Id ASC LIMIT 20

**Advantages**:

Unlike offset pagination seek pagination take constant time to fetch the dataset hence time does not vary even for the large offset values.

It is faster in response can be used to query large data set.

**Disadvantages**:

1. More logic might be needed at backend side if we need to filter of some other field than ID.
2. Id any records gets deleted from the DB after\_id field might be no longer valid.

**2. Discuss at least two pagination schemes and their pros/cons?**

**Following are the 2 other methods other than the seek pagination we can use:**

1. **Offset Pagination:** It is achieved using the limit and offset commands , works well with sql queries as offset keyword in sql command take care of the functionality hence it requires no backend logic to be implemented .

**Limit:** number of records needs to be displayed on each page.

**Offset**: number of records discards.

**Advantages:**

1. Simple to Implement**.**
2. Works well with small set of data.

**Disadvantages:**

1. Stumble when use it for large set of data, if offset is 1000000 records It might take long time to trace all a million records and discard them.
2. **Cursor pagination**: It is archived by returning a pointer to a specific item in the dataset. On subsequent requests, the server returns results after the given pointer.

**Example:**

**Limit:** number of records needs to be displayed on each page.

**Created:** cursor field records must be ordered on this field either ascending or descending**.**

1. Client requests most recent items

GET /items?limit=20

1. Upon clicking the next page, the query finds the minimum created date then used to create a query filter for the next page.

GET /items?limit=20&created=2019-01-20

**Advantages:**

1. Works well with large dataset.

**Disadvantages:**

1. Must move on pages sequentially cannot skip pages.

**3. Number of records needs to be displayed on each page?**

**Offset Pagination:**

**Request:**

1. **/items?limit=2&offset=0**
2. **/items?limit=2&offset=2**
3. **And so on**

**Response:**

**{**

**“pageNumber”: 1,**

**“Items”: [**

**{**

**“ItemId”: 1,**

**“ItemName”:”test”**

**“price”:100 ,**

**“createDate”: “2020/01/20”**

**},**

**{**

**“ItemId”: 2,**

**“ItemName”:”test1”**

**“price”:2000,**

**“createDate”: “2020/01/19”**

**}**

**]**

**}**

**Cursor Pagination:**

**Request:**

1. **/items?limit=2**
2. **/items?limit=2&createDate=”2020/01/19”**
3. **And so on**

**Response:**

**{**

**“pageNumber”:1 ,**

**“Items”: [**

**{**

**“ItemId”: 1,**

**“ItemName”:”test”**

**“price”:100 ,**

**“createDate”: “2020/01/20”**

**},**

**{**

**“ItemId”: 2,**

**“ItemName”:”test1”**

**“price”:2000,**

**“createDate”: “2020/01/19”**

**}**

**]**

**}**

**In node js mongoose is one of the library which is used most commonly to achieve pagination.**