MongoDB insertMany

Summary: in this tutorial, you'll learn how to use the MongoDB `insertMany()` method to insert multiple documents into a collection.

Introduction to the MongoDB insertMany() method

The `insertMany()` allows you to insert multiple documents into a collection. Here is the syntax of the `insertMany()` method:

```
db.collection.insertMany(
   [document1, document2, ...],
   {
      writeConcern: <document>,
      ordered: <boolean>
   }
)
```

The `insertMany()` method accepts two arguments:

[document1, document2, ...]

The first argument is an array of documents that you want to insert into the collection.

Option

The second argument is a document that contains two optional field-and-value pairs:

```
{
   writeConcern: <document>,
    ordered: <boolean>
  }
```

The `writeConcern` specifies the write concern. If you omit it, the `insertMany()` method will use the default write concern.

The `ordered` is a boolean value that determines whether MongoDB should perform an ordered or unordered insert.

When the `ordered` is set to `true` , the `insertMany()` method inserts documents in order. This is also the default option.

If the `ordered` is set to `false` , MongoDB may reorder the documents before inserts to increase performance. Therefore, you should not depend on the ordering of inserts if the `ordered` is set to `false` . You'll see how the `ordered` affects the behaviors of the insert in the example section.

The `insertMany()` method returns a document that contains:

- The `acknowledged` key sets to `true` if operation executed with a write concern or `false` if the write concern was disabled.
- An array of `_id` values of successfully inserted documents.

Collection creation

If the collection doesn't exist, the `insertMany()` method will create the collection and insert the documents. And it only creates the collection when the insert operation is successful.

_id field

If you don't specify the `_id` field for the document, the MongoDB generates a unique `ObjectId` value, assigns it to the `_id` field, and adds the `_id` field to the document before insert.

If you specify the `_id` fields for the document, it must be unique within the collection or you'll get a duplicate key error.

Error handling

The `insertMany()` throws a `BulkWriteError` exception in case of an error.

If an error occurs, the ordered insert will stop while the unordered insert will continue to process for the remaining documents in the queue.

MongoDB insertMany() method examples

First, launch `mongo` shell from the Terminal on macOS and Linux or Command Prompt on Windows and connect to the `bookdb` database on the local MongoDB server

```
mongosh bookdb
```

1) Using MongoDB insertMany() method to insert multiple documents without specifying _id fields

The following statement uses the `insertMany()` method to insert multiple documents without the `_id` fields:

```
db.books.insertMany([
   { title:  "NoSQL Distilled", isbn: "0-4696-7030-4"},
   { title:  "NoSQL in 7 Days", isbn: "0-4086-6859-8"},
   { title:  "NoSQL Database", isbn: "0-2504-6932-4"},
]);
```

Output:

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("62148d16f514a446bf1a98f1"),
    '1': ObjectId("62148d16f514a446bf1a98f2"),
    '2': ObjectId("62148d16f514a446bf1a98f3")
    }
  }
```

Because we did not specify the `_id` fields for the documents, MongoDB added the `_id` field with unique `ObjectId` to each document.

To retrieve the inserted documents, you use the `find()` method like this:

```
db.books.find()
```

Output:

```
[
  {
    _id: ObjectId("62148d16f514a446bf1a98f1"),
    title: 'NoSQL Distilled',
    isbn: '0-4696-7030-4'
  },
  {
    _id: ObjectId("62148d16f514a446bf1a98f2"),
    title: 'NoSQL in 7 Days',
    isbn: '0-4086-6859-8'
  },
  {
    _id: ObjectId("62148d16f514a446bf1a98f3"),
    title: 'NoSQL Database',
    isbn: '0-2504-6932-4'
  }
]
boo
```

2) Using MongoDB insertMany() method to insert multiple documents with _id fields

The following statement uses the `insertMany()` method to insert multiple documents with the `_id` fields:

```
db.books.insertMany([
   { _id: 1, title:  "SQL Basics", isbn: "0-7925-6962-8"},
   { _id: 2, title:  "SQL Advanced", isbn: "0-1184-7778-1"}
]);
```

Output:

```
{ acknowledged: true, insertedIds: { '0': 1, '1': 2 } }
```

The following statement attempts to insert documents whose `_id` value already exist:

```
db.books.insertMany([
   { _id: 2, title:  "SQL Performance Tuning", isbn: "0-6799-2974-6"},
   { _id: 3, title:  "SQL Index", isbn: "0-5097-1723-3"}
]);
```

Since the `_id: 2` already exists, MongoDB threw the following exception:

```
Uncaught:
MongoBulkWriteError: E11000 duplicate key error collection: bookdb.books index: _
Result: BulkWriteResult {
  result: {
    ok: 1,
    writeErrors: [
      WriteError {
        err: {
          index: 0,
          code: 11000,
          errmsg: 'E11000 duplicate key error collection: bookdb.books index: _ic
          errInfo: undefined,
          op: {
            _id: 2,
            title: 'SQL Performance Tuning',
            isbn: '0-6799-2974-6'
          }
        }
      }
    ],
    writeConcernErrors: [],
    insertedIds: [ { index: 0, _id: 2 }, { index: 1, _id: 3 } ],
    nInserted: 0,
    nUpserted: 0,
    nMatched: 0,
    nModified: 0,
    nRemoved: 0,
    upserted: []
  }
}
```

3) Unordered insert example

The following example uses the `insertMany()` method to perform an unordered insert:

```
db.books.insertMany(
   [{ _id: 3, title:  "SQL Performance Tuning", isbn: "0-6799-2974-6"},
   { _id: 3, title:  "SQL Trees", isbn: "0-6998-1556-8"},
   { _id: 4, title:  "SQL Graph", isbn: "0-6426-4996-0"},
   { _id: 5, title:  "NoSQL Pros", isbn: "0-9602-9886-X"}],
   { ordered: false }
);
```

In this example, the `_id: 3` is duplicated, MongoDB threw an error.

Since this example used the unordered insert, the operation continued to insert the documents with `_id` 4 and 5 into the `books` collection.

The following statement retrieves the inserted documents:

```
db.books.find()
```

Output:

```
[
  {
    _id: ObjectId("62148d16f514a446bf1a98f1"),
    title: 'NoSQL Distilled',
    isbn: '0-4696-7030-4'
  },
  {
    _id: ObjectId("62148d16f514a446bf1a98f2"),
    title: 'NoSQL in 7 Days',
    isbn: '0-4086-6859-8'
  },
  {
    _id: ObjectId("62148d16f514a446bf1a98f3"),
    title: 'NoSQL Database',
    isbn: '0-2504-6932-4'
  },
  { _id: 1, title: 'SQL Basics', isbn: '0-7925-6962-8' },
  { _id: 2, title: 'SQL Advanced', isbn: '0-1184-7778-1' },
  { _id: 3, title: 'SQL Performance Tuning', isbn: '0-6799-2974-6' },
  { _id: 4, title: 'SQL Graph', isbn: '0-6426-4996-0' },
  { _id: 5, title: 'NoSQL Pros', isbn: '0-9602-9886-X' }
]
```

Summary

- Use the `db.collection.insertMany()` method to insert multiple documents into a collection.
- When the `ordered` is `true` , the `insertMany()` performs an unordered insert; otherwise, it performs an unordered insert.