**Inserting Documents**

Introduction to MongoDB insertOne() method

The insertOne() method allows you to insert a single document into a collection.

The insertOne() method has the following syntax:

```
db.collection.insertOne(
  <document>,
  { writeConcern: <document>}
)
```

Code language: CSS (css)

The insertOne() method accepts two arguments:

- document is a document that you want to insert into the collection.
  The document argument is required.
- writeConcern is an optional argument that describes the level of acknowledgment requested from MongoDB for insert operation to a standalone MongoDB server or to shared clusters. We'll discuss the writeConcern another tutorial.

The insertOne() method returns a document that contains the following fields:

- acknowledged is a boolean value. It is set to true if the insert executed with write concern or false if the write concern was disabled.
- insertedId stores the value of _id field of the inserted document.

Note that if the collection does not exist, the insertOne() method will also create the collection and insert the document into it.

If you don't specify the _id field in the document, MongoDB will add the _id field and generate a unique ObjectId for it before insert.

If you explicitly specify a value for the _id field, you need to ensure that it is unique in the collection. Otherwise, you will get a duplicate key error.

To insert multiple documents into a collection, you use the insertMany() method.

MongoDB insertOne() method examples

First, you need to launch the mongo shell and connect it to the bookdb database:

```
mongosh bookdb
```

1) Insert a document without an _id field example

The following example uses the insertOne() method to insert a new document into the books collection:

```
db.books.insertOne({
   title: 'MongoDB insertOne',
   isbn: '0-7617-6154-3'
});
```

Code language: CSS (css)

Output:

```
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f31cf00902f22de3464ddc4")
}
```

Code language: JSON / JSON with Comments (json)

In this example, we passed a document to the insertOne() method without specifying the _id field. Therefore, MongoDB automatically added the _id field and assigned it a unique ObjectId value.

Note that you will see a different ObjectId value from this example because ObjectId values are specific to machine and time when the insertOne() method executes.

To select the document that you have inserted, you can use the find() method like this:

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

Code language: CSS (css)

Output:

```
[
  {
   _id: ObjectId("621489fcf514a446bf1a98ea"),
   title: 'MongoDB insertOne',
   isbn: '0-7617-6154-3'
  }
]
```

Code language: JavaScript (javascript)

2) Insert a document with an _id field example

The following example uses the insertOne() method to insert a document that has an _id field into the books collection:

```
db.books.insertOne({
  _id: 1,
  title: "Mastering Big Data",
  isbn: "0-9270-4986-4"
});
```

Code language: CSS (css)

Output:

```
{ "acknowledged" : true, "insertedId" : 1 }
```

Code language: JSON / JSON with Comments (json)

The following example attempts to insert another document whose _id field already exists into the books collection:

```
db.books.insertOne({
  _id: 1,
  title: "MongoDB for JS Developers",
  isbn: "0-4925-3790-9"
});
```

Code language: CSS (css)

Since the _id: 1 already exists, MongoDB threw the following exception:

```
WriteError({
	"index" : 0,
	"code" : 11000,
	"errmsg" : "E11000 duplicate key error collection: bookstore.books index: _id_ dup key: { _id: 1.0 }",
	"op" : {
		"_id" : 1,
            "title" : "MongoDB for JS Developers",
            "isbn" : "0-4925-3790-9"
        }
})
```

Code language: JavaScript (javascript)

Summary

- Use db.collection.insertOne() method to insert a single document into a collection.
- If you explicitly provide a value for the _id field, you must ensure that the value is unique within the collection or you will get a duplicate key error.

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>
  }
)
```

Code language: CSS (css)

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>
}
```

Code language: CSS (css)

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"},
]);
```

Code language: JavaScript (javascript)

Output:

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

Code language: CSS (css)

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()
```

Code language: CSS (css)

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
```

Code language: JavaScript (javascript)

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"}
]);
```

Code language: JavaScript (javascript)

Output:

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

Code language: CSS (css)

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"}
]);
```

Code language: JavaScript (javascript)

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

```
Uncaught:
MongoBulkWriteError: E11000 duplicate key error collection: bookdb.books index: _id_ dup
key: { _id: 2 }
Result: BulkWriteResult {
 result: {
   ok: 1,
   writeErrors: [
     WriteError {
       err: {
         index: 0,
         code: 11000,
         errmsg: 'E11000 duplicate key error collection: bookdb.books index: _id_ dup key: {
_id: 2 }',
         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: []
 }
}
```

Code language: CSS (css)

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 }
);
```

Code language: JavaScript (javascript)

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()
```

Code language: CSS (css)

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' }
]
```

Code language: JavaScript (javascript)

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.

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>
  }
)
```

Code language: CSS (css)

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>
}
```

Code language: CSS (css)

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"},
]);
```

Code language: JavaScript (javascript)

Output:

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

Code language: CSS (css)

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()
```

Code language: CSS (css)

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
```

Code language: JavaScript (javascript)

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"}
]);
```

Code language: JavaScript (javascript)

Output:

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

Code language: CSS (css)

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"}
]);
```

Code language: JavaScript (javascript)

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

```
Uncaught:
MongoBulkWriteError: E11000 duplicate key error collection: bookdb.books index: _id_ dup key: { _id: 2 }
Result: BulkWriteResult {
  result: {
    ok: 1,
    writeErrors: [
      WriteError {
        err: {
          index: 0,
          code: 11000,
          errmsg: 'E11000 duplicate key error collection: bookdb.books index: _id_ dup key: {
_id: 2 }',
          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: []
  }
}
```

Code language: CSS (css)

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 }
);
```

Code language: JavaScript (javascript)

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()
```

Code language: CSS (css)

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' }
]
```

Code language: JavaScript (javascript)

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.