



LESSON

# Deleting Data in MongoDB

Google slide deck available [here](#)

This work is licensed under the [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#)  
(CC BY-NC-SA 3.0)



# Approaches to Deleting Data in MongoDB

- Delete an individual document or subset of documents
- Delete all of the documents in a collection
  - By finding and deleting each or by dropping the collection
- Age the data out using Time To Live (TTL) indexes



Let's look at the approaches for deleting data and within these aspects we will cover the recommendations and best practices for deletion of data in MongoDB

There are a variety of ways to delete data in MongoDB whether you are deleting an individual document or a subset of documents.

There are various performance impacts from the different methods so in this lesson, we'll first introduce the various approaches and discuss the impacts for each.

Recalling from our earlier lesson on "Inserting and updating data in MongoDB" that if you delete data that is indexed you will also have to update the index(es).



# Approaches to Deleting Data in MongoDB

- Delete an individual document or subset of documents
- Delete all of the documents in a collection
  - By finding and deleting each or by dropping the collection
- Age the data out using Time To Live (TTL) indexes



There are two approaches to deleting all of the documents in a collection, either you find and delete every document and then if required you manually delete each associated index for that collection. Alternatively, you can drop the collection which also implicitly deletes the associated indexes.

In terms of performance, it is vastly more performant to perform a drop with the associated indexes drops than to individually delete each document which triggers an update to the index and which then should be deleted if not required. Even if all the indexes are deleted prior to the deletion of all documents, the `_id` index will still present as it is unique and cannot be deleted in this fashion, as it will still be present it will then also still need to be updated to reflect each document deletion.



# Approaches to Deleting Data in MongoDB

- Delete an individual document or subset of documents
- Delete all of the documents in a collection
  - By finding and deleting each or by dropping the collection
- Age the data out using Time To Live (TTL) indexes



Another approach to deleting data in MongoDB is to use Time To Live (TTL) indexes which can be used to expire data on a specific date or after a specific amount of time.



# Deleting Documents and Collections

Let's first look a little more deeply at deleting individual documents or groups of documents and then we'll look at deleting entire collections.

# MQL Delete

**deleteOne()** Deletes one document from a collection.

**deleteMany()** Deletes many documents from a collection.

**writeConcern** Sets the level of acknowledgment requested from MongoDB for write operations.

```
>>> db.cows.deleteOne({milk: 9})
{ "acknowledged" : true, "deletedCount" : 1 }
```

```
>>> db.cows.deleteMany({}, {writeConcern: {w:
"majority"}})
{ "acknowledged" : true, "deletedCount" : 2 }
```

In terms of the D/Delete in CRUD, both functions take a filter document which specifies which documents to be delete, if the filter document is empty the first document in the collection is updated. These functions also take a second document as an argument which contains the various options that can be configured.

Let's again use the data on cows that we have entered and test the deleteOne and deleteMany() functions. We can see in the deletedCount field for both functions how many documents they deleted for the respective operation.



# MQL Delete

Specifically the `db.collection.deleteOne()` and `db.collection.deleteMany()` methods but also `db.collection.drop()`.

To delete all documents from a collection, pass an empty filter document `{}` to the `db.collection.deleteMany()` method.

Additional methods include `db.collection.findOneAndDelete()` and `db.collection.findAndModify()`, both offer a sort option. Deletes are also possible via the `db.collection.bulkWrite()` method.

We covered the `deleteOne` and `deleteMany` methods, these will be used and sufficient for the majority of use. However, if you need to delete all of the documents in a collection then you should look at `db.collection.drop()`. It has a further advantage that it removes all of the associated indexes related to that collection as well as the documents.



# MQL Delete

Specifically the `db.collection.deleteOne()` and `db.collection.deleteMany()` methods but also `db.collection.drop()`.

To delete all documents from a collection, pass an empty filter document `{}` to the **`db.collection.deleteMany()`** method.

Additional methods include `db.collection.findOneAndDelete()` and `db.collection.findAndModify()`, both offer a sort option. Deletes are also possible via the `db.collection.bulkWrite()` method.

It is possible to delete all the documents by simply passing an empty filter document `{}`, however as mentioned it may be more performant to use drop the collection rather than delete all the documents.



# MQL Delete

Specifically the `db.collection.deleteOne()` and `db.collection.deleteMany()` methods but also `db.collection.drop()`.

To delete all documents from a collection, pass an empty filter document `{}` to the `db.collection.deleteMany()` method.

Additional methods include `db.collection.findOneAndDelete()` and `db.collection.findAndModify()`, both offer a sort option. Deletes are also possible via the `db.collection.bulkWrite()` method.

In cases where you want to delete the documents in a sorted order you can use `findOneAndDelete()` or `findAndModify()`.

It is also possible to delete documents using the `bulkWrite()` method, the function takes an array of `bulkWrite` operations where you can control the order of execution if necessary.

The bulk API allows for mixing operations which returns a single result. An example might be a daily processing job where a set of insert, update, and deletions should occur and in that sequence.

# MQL Drop

**drop()** Deletes a collection or a view on a collection.

Recommended approach when deleting all complete documents in a collection.

**writeConcern** Sets the level of acknowledgment requested from MongoDB for write operations.

```
>>> db.cows.deleteOne({milk: 9})  
  
{ "acknowledged" : true, "deletedCount" : 1 }
```

```
>>> db.cows.deleteMany({}, {writeConcern: {w:  
"majority"}})  
  
{ "acknowledged" : true, "deletedCount" : 2 }
```

It is also possible to use the `db.<collection>.drop()` command to drop either a collection or a view on a collection.

The `drop()` command can take an optional `writeConcern` setting.

`drop()` will delete all of the associated indexes for the specific collection.

The number of documents and the number of associated indexes will determine how long the operation will take. The drop operation will take an exclusive lock on the collection for the duration of the operation.

Drop is recommended when you want to delete all of the documents in a collection as it also handles removing the indexes. There is a significant performance difference between dropping the collection and its documents versus individual deleting each and updating each index as a document is deleted. This is why it best practice to chose drop for these scenarios..

# Quiz





# Quiz

Which of the following are true for deleting data in MongoDB?

More than one answer choice can be correct.

- A. Deleting documents has no impact on indexes
- B. Document deletes can be automated with indexes in MongoDB
- C. Dropping a collection does not delete the associated indexes
- D. Bulk API allows multiple CRUD operations on a collection to be combined into a single call to the database



# Quiz

Which of the following are true for deleting data in MongoDB?

More than one answer choice can be correct.

- A. Deleting documents has no impact on indexes
- B. Document deletes can be automated with indexes in MongoDB
- C. Dropping a collection does not delete the associated indexes
- D. Bulk API allows multiple CRUD operations on a collection to be combined into a single call to the database

INCORRECT: Deleting documents has no impact on indexes - This is not correct, if you delete a document or many documents then the corresponding entries in the index or indexes for the collection must also be updated.

CORRECT: Document deletes can be automated with indexes in MongoDB - This is correct, it can be automated in terms of expiring data using MongoDB Time To Live (TTL) indexes.

INCORRECT: Dropping a collection does not delete the associated indexes - This is incorrect, dropping a collection deletes all of the documents and all of the associated indexes for that specific collection.

CORRECT: Bulk API allows multiple CRUD operations on a collection to be combined into a single call to the database - This is correct, the Bulk API allows many different CRUD operations to be combined into one call to the database, further these operations can be ordered or unordered in terms of their execution.



# Quiz

Which of the following are true for deleting data in MongoDB? More than one answer choice can be correct.

- A. Deleting documents has no impact on indexes
- B. Document deletes can be automated with indexes in MongoDB
- C. Dropping a collection does not delete the associated indexes
- D. Bulk API allows multiple CRUD operations on a collection to be combined into a single call to the database

*This is incorrect. If you delete a document or many documents then the corresponding entries in the index or indexes for the collection must also be updated.*

**INCORRECT:** Deleting documents has no impact on indexes - This is not correct, if you delete a document or many documents then the corresponding entries in the index or indexes for the collection must also be updated.



# Quiz

Which of the following are true for deleting data in MongoDB? More than one answer choice can be correct.

- A. Deleting documents has no impact on indexes
- B. Document deletes can be automated with indexes in MongoDB
- C. Dropping a collection does not delete the associated indexes
- D. Bulk API allows multiple CRUD operations on a collection to be combined into a single call to the database

*This is correct. It can be automated in terms of expiring data using MongoDB Time To Live (TTL) indexes.*

**CORRECT:** Document deletes can be automated with indexes in MongoDB - This is correct. It can be automated in terms of expiring data using MongoDB Time To Live (TTL) indexes.



# Quiz

Which of the following are true for deleting data in MongoDB? More than one answer choice can be correct.

- A. Deleting documents has no impact on indexes
- B. Document deletes can be automated with indexes in MongoDB
- C. Dropping a collection does not delete the associated indexes
- D. Bulk API allows multiple CRUD operations on a collection to be combined into a single call to the database

*This is incorrect. Dropping a collection deletes all of the documents and all of the associated indexes for that specific collection.*

**INCORRECT:** Dropping a collection does not delete the associated indexes - This is incorrect. Dropping a collection deletes all of the documents and all of the associated indexes for that specific collection.



# Quiz

Which of the following are true for deleting data in MongoDB? More than one answer choice can be correct.

- A. Deleting documents has no impact on indexes
- B. Document deletes can be automated with indexes in MongoDB
- C. Dropping a collection does not delete the associated indexes
- D. Bulk API allows multiple CRUD operations on a collection to be combined into a single call to the database

*This is correct. The Bulk API allows many different CRUD operations to be combined into one call to the database, further these operations can be ordered or unordered in terms of their execution.*

CORRECT: Bulk API allows multiple CRUD operations on a collection to be combined into a single call to the database - This is correct. The Bulk API allows many different CRUD operations to be combined into one call to the database, further these operations can be ordered or unordered in terms of their execution.



# Deleting Using TTL Indexes

Let's look at using TTL indexes to delete documents in collections, we'll look at both kinds of TTLs, firstly those that expire on a specific date and then you will look at those that expire after a specific period of time.



# Time to Live Index

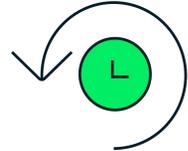
Single field index, only documents with this field will be deleted.

After a certain period of time or at a specific clock time.

A background task that removes expired documents runs every 60 seconds.

In a replica set, deletion occurs on the primary and the deletion operations are replicated to the secondaries.

Expired documents may expire before the background task runs and before the time MongoDB actually removes the document.



TTL indexes are special single field indexes, a document can only be deleted from the collection by a TTL index if it has the specific single field otherwise it will be ignored.



# Time to Live Index

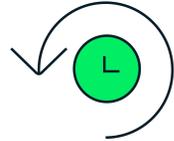
Single field index, only documents with this field will be deleted.

After a certain period of time or at a specific clock time.

A background task that removes expired documents runs every 60 seconds.

In a replica set, deletion occurs on the primary and the deletion operations are replicated to the secondaries.

Expired documents may expire before the background task runs and before the time MongoDB actually removes the document.



TTL indexes have two types of expiration of data, either they can delete the documents after a set period of time or they can be deleted at a specific clock time. The set period of time is set within the index whilst the specific clock time is specified in the document in the field used by the TTL index.



# Time to Live Index

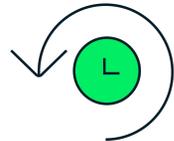
Single field index, only documents with this field will be deleted.

After a certain period of time or at a specific clock time.

A background task that removes expired documents runs every 60 seconds.

In a replica set, deletion occurs on the primary and the deletion operations are replicated to the secondaries.

Expired documents may expire before the background task runs and before the time MongoDB actually removes the document.



Documents are not immediately expired / deleted by a TTL index, rather the database runs a background task every 60 seconds and at that point it will delete the expired documents. This means the deletion time is approximate rather than exact when using TTL indexes.



# Time to Live Index

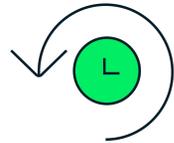
Single field index, only documents with this field will be deleted.

After a certain period of time or at a specific clock time.

A background task that removes expired documents runs every 60 seconds.

In a replica set, deletion occurs on the primary and the deletion operations are replicated to the secondaries.

Expired documents may expire before the background task runs and before the time MongoDB actually removes the document.



In terms of TTL indexes, these perform the operations on the primary with those operations being replicated by the oplog to the secondaries.



# Time to Live Index

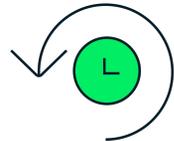
Single field index, only documents with this field will be deleted.

After a certain period of time or at a specific clock time.

A background task that removes expired documents runs every 60 seconds.

In a replica set, deletion occurs on the primary and the deletion operations are replicated to the secondaries.

Expired documents may expire before the background task runs and before the time MongoDB actually removes the document.



As noted earlier there can be a difference between when the document is actually removed and when the document is configured to be removed as there can be a gap between the background task running and the time configured in the TTL for the documents removal.

# Example: TTL Index

Let's look at using a TTL index to expire documents. Let's take an example of coupons for e-commerce and use this with our TTL indexes to show how these can be expired with the index.

You can perform this example either with the MongoDB Web Shell, visit <https://mws.mongodb.com/?version=4.4> in your browser or with a local MongoDB Shell connected to a MongoDB deployment, whether running on your laptop or in the Cloud (maybe on MongoDB's Atlas platform).

EXAMPLE



## TTL Index

```
>>> use test
>>> db.coupons.drop()
```

Firstly, let's add drop the coupons collection to make sure we start with a fresh empty collection.

We drop the collection (`db.coupons.drop()`) to simplify this example as existing data may change the number of documents that could be returned and it's easier for this example to start fresh.

Here is the code block:

```
use test;
db.coupons.drop();
```



## Inserting Data

```
>>> use test
>>> db.coupons.drop()
>>> db.coupons.insertMany([{"_id": 1, "coupon": "10% off apples",
"coupon_value": 10, "discounted_product":"apples","expireAt": new
Date('January 01, 2021 14:00:00')}, {"_id": 2, "coupon": "10% off",
"coupon_value": 10, "expireAt": new Date('December 01, 2021
14:00:00')}])
{ "acknowledged" : true, "insertedIds" : [ 1, 2 ] }
```

Now that we've dropped any old data, let's add two documents, these represent coupons/discounts which have an expiry date.

Here is the code block:

```
use test;
db.coupons.drop();
db.coupons.insertMany([{"_id": 1, "coupon": "10% off apples",
"coupon_value": 10, "discounted_product":"apples","expireAt":
new Date('January 01, 2021 14:00:00')}, {"_id": 2, "coupon":
"10% off", "coupon_value": 10, "expireAt": new Date('December
01, 2021 14:00:00')}]);
```



## Creating the TTL Index

```
>>> db.coupons.createIndex( { "expireAt": 1 }, { expireAfterSeconds:
0 } )
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
```

Let's now create a TTL index on the coupons, specifically on the "expireAt" field. If you note the field "expireAfterSeconds" as being 0/zero then we can identify that the TTL index we are adding will expire on the Date field that has been set in the document in the "expireAt" field.

In this example, we've deliberately set the first document to have a date in the past for the "expireAt" field. This means it will be expired on the next run of the TTL background thread. So let's move to the next slide to see this.

Here is the code block:

```
use test;
db.coupons.drop();
db.coupons.insertMany([{"_id": 1, "coupon": "10% off apples",
"coupon_value": 10, "discounted_product":"apples","expireAt":
new Date('January 01, 2021 14:00:00')}, {"_id": 2, "coupon":
"10% off", "coupon_value": 10, "expireAt": new Date('December
01, 2021 14:00:00')}]);
db.coupons.createIndex( { "expireAt": 1 }, {
expireAfterSeconds: 0 } );
```



## Check the Collections

```
>>> db.coupons.find()
{ "_id" : 1, "coupon" : "10% off apples", "coupon_value" : 10,
  "discounted_product" : "apples", "expireAt" : ISODate("2021-01-01T14:00:00Z") }
{ "_id" : 2, "coupon" : "10% off", "coupon_value" : 10, "expireAt" :
  ISODate("2023-12-01T14:00:00Z") }
```

**(wait 60 seconds or so)**

```
>>> db.coupons.find()
{ "_id" : 2, "coupon" : "10% off", "coupon_value" : 10, "expireAt" :
  ISODate("2023-12-01T14:00:00Z") }
```

In the same MongoDB Shell window, let's run a find on the coupons collection. If we are quick (less than 60 seconds), it will still show two documents.

Let's now wait for about 60 seconds or so, and then we can again re-run the find query.

Running the find query on the coupons collection after this period will only return one document as the other document has been expired by the TTL and deleted from the collection.



## Create a TTL Index for 24 Hours

Using the same window, change `<a>` to the field containing the the only date field in the document (see below for an example). `<b>` should be set to 24 hours in terms of seconds (60 seconds \* 60 minutes \* 24 hours = the value for b).

```
>>> db.coupons.dropIndex({"expireAt": 1})
{ "nIndexesWas" : 2, "ok" : 1 }
...
>>> db.coupons.createIndex( { "<a>": 1 }, { expireAfterSeconds: <b> } )
...
>>> db.coupons.getIndexes()
```

Let's now create a different kind of TTL where we want an explicit expiration time that is controlled not by the field in the document but rather by the TTL index itself. In this case, we also clean up the index we previously created as we want to again use the field name for the TTL index we will create in this exercise.

In this example, you will need to reuse the same Mongo Shell window and change `<a>` to field with the date information we used previously. You should also set `<b>` to represent 24 hours in terms of seconds, this configures the new TTL index to delete documents 24 hours or so after they have been added.

The result in the code block is what will create a TTL with a 24 hour expiration with documents which have the "expireAt" field in the "coupons" collection.

```
db.coupons.dropIndex({"expireAt": 1});
db.coupons.createIndex({"expireAt": 1}, { expireAfterSeconds:
86400 });
db.coupons.getIndexes();
```



## Results from Example

```
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "expireAt" : 1
    },
    "name" : "expireAt_1",
    "expireAfterSeconds" : 86400
  }
]
```

Here is the output from the `getIndexes()` command and we can see that we have created a new TTL index called "expire\_at" which has a 24 hour time-to-live setting for documents with that field as a date/time field in the collection.

It should be noted that documents without that field (and where it is a date/time field) will not be deleted/expired, it's like any document that does not contain the indexed field, it won't be found so it can't be deleted.

# Quiz





## Quiz

Which of the following are true for deleting data in MongoDB? More than one answer choice can be correct.

- A. A Time To Live (TTL) index can be on multiple fields
- B. A TTL index deletes the document at the exact expire time
- C. TTL indexes use a background task to delete documents
- D. TTL indexes are for either a specific elapsed time or at a specific time



# Quiz

Which of the following are true for deleting data in MongoDB? More than one answer choice can be correct.

- A. A Time To Live (TTL) index can be on multiple fields
- B. A TTL index deletes the document at the exact expiry time
- C. TTL indexes use a background task to delete documents
- D. TTL indexes are for either a specific elapsed time or at a specific time

INCORRECT: A Time To Live (TTL) index can be on multiple fields - This is incorrect, a TTL index can only be used with a single field

INCORRECT: A TTL index deletes the document at the exact expiry time - This is incorrect, documents are deleted by a periodic tasks once they have expired but it is not guaranteed to be at the exact expiry time.

CORRECT: TTL indexes use a background task to delete documents - This is correct, a background task that runs every 60 seconds is used to identify and delete documents that have expired.

CORRECT: TTL indexes are for either a specific elapsed time or at a specific time - This is correct, a TTL can either be set for a specific elapsed time or for a specific time (date and time) but not both.



# Quiz

Which of the following are true for deleting data in MongoDB?  
More than one answer choice can be correct.

- A. A Time To Live (TTL) index can be on multiple fields
- B. A TTL index deletes the document at the exact expire time
- C. TTL indexes use a background task to delete documents
- D. TTL indexes are for either a specific elapsed time or at a specific time

*This is incorrect. A TTL index can only be used with a single field.*

**INCORRECT:** A Time To Live (TTL) index can be on multiple fields - This is incorrect, a TTL index can only be used with a single field



# Quiz

Which of the following are true for deleting data in MongoDB?  
More than one answer choice can be correct.

- A. A Time To Live (TTL) index can be on multiple fields
- B. A TTL index deletes the document at the exact expiry time
- C. TTL indexes use a background task to delete documents
- D. TTL indexes are for either a specific elapsed time or at a specific time

*This is incorrect. Documents are deleted by a periodic task once they have expired but it is not guaranteed to be at the exact expiry time.*

**INCORRECT:** A TTL index deletes the document at the exact expiry time - This is incorrect. Documents are deleted by a periodic task once they have expired but it is not guaranteed to be at the exact expiry time.



# Quiz

Which of the following are true for deleting data in MongoDB?  
More than one answer choice can be correct.

- A. A Time To Live (TTL) index can be on multiple fields
- B. A TTL index deletes the document at the exact expire time
- C. TTL indexes use a background task to delete documents
- D. TTL indexes are for either a specific elapsed time or at a specific time

*This is correct. A background task that runs every 60 seconds is used to identify and delete documents that have expired.*

CORRECT: TTL indexes use a background task to delete documents - This is correct. A background task that runs every 60 seconds is used to identify and delete documents that have expired.



# Quiz

Which of the following are true for deleting data in MongoDB?  
More than one answer choice can be correct.

- A. A Time To Live (TTL) index can be on multiple fields
- B. A TTL index deletes the document at the exact expire time
- C. TTL indexes use a background task to delete documents
- D. TTL indexes are for either a specific elapsed time or at a specific time

*This is correct. A TTL can either be set for a specific elapsed time or for a specific time (date and time) but not both.*

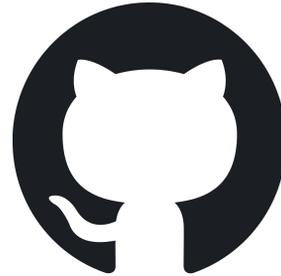
**CORRECT:** TTL indexes are for either a specific elapsed time or at a specific time - This is correct. A TTL can either be set for a specific elapsed time or for a specific time (date and time) but not both.

## Continue Learning!



[MongoDB University](#) has free self-paced courses and labs ranging from beginner to advanced levels.

## GitHub Student Developer Pack



Sign up for the [MongoDB Student Pack](#) to receive \$50 in Atlas credits and free certification!

This concludes the material for this lesson. However, there are many more ways to learn about MongoDB and non-relational databases, and they are all free! Check out [MongoDB's University](#) page to find free courses that go into more depth about everything MongoDB and non-relational. For students and educators alike, MongoDB for Academia is here to offer support in many forms. Check out our [educator resources](#) and join the Educator Community. Students can receive \$50 in Atlas credits and free certification through the [GitHub Student Developer Pack](#).