# **Getting Started with MongoDB**

This tutorial provides an introduction to basic database operations using the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell. **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** is a part of the standard MongoDB distribution and provides a full JavaScript environment with a complete access to the JavaScript language and all standard functions as well as a full database interface for MongoDB. See the [mongo JavaScript API](http://api.mongodb.org/js) documentation and the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell [JavaScript Method Reference](http://docs.mongodb.org/manual/reference/method/).

The tutorial assumes that you’re running MongoDB on a Linux or OS X operating system and that you have a running database server; MongoDB does support Windows and provides a Windows distribution with identical operation. For instructions on installing MongoDB and starting the database server, see the appropriate [installation](http://docs.mongodb.org/manual/installation/) document.

## **Connect to a Database**

In this section, you connect to the database server, which runs as **[mongod](http://docs.mongodb.org/manual/reference/program/mongod/" \l "bin.mongod)**, and begin using the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell to select a logical database within the database instance and access the help text in the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell.

### **Connect to a [mongod](http://docs.mongodb.org/manual/reference/program/mongod/" \l "bin.mongod)**

From a system prompt, start **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** by issuing the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** command, as follows:

mongo

By default, **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** looks for a database server listening on port **27017** on the**localhost** interface. To connect to a server on a different port or interface, use the --port and --host options.

### Select a Database

After starting the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell your session will use the **test** database by default. At any time, issue the following operation at the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** to report the name of the current database:

db

1. From the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell, display the list of databases, with the following operation:

show dbs

1. Switch to a new database named **mydb**, with the following operation:

use mydb

1. Confirm that your session has the **mydb** database as context, by checking the value of the **db** object, which returns the name of the current database, as follows:

db

At this point, if you issue the **show dbs** operation again, it will not include the **mydb** database. MongoDB will not permanently create a database until you insert data into that database. The [Create a Collection and Insert Documents](http://docs.mongodb.org/manual/tutorial/getting-started/" \l "getting-started-create-documents)section describes the process for inserting data.

*New in version 2.4:* **show databases** also returns a list of databases.

### Display mongo Help

At any point, you can access help for the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell using the following operation:

help

Furthermore, you can append the **.help()** method to some JavaScript methods, any cursor object, as well as the **db** and**db.collection** objects to return additional help information.

## **Create a Collection and Insert Documents**

In this section, you insert documents into a new [collection](http://docs.mongodb.org/manual/reference/glossary/" \l "term-collection) named **testData** within the new [database](http://docs.mongodb.org/manual/reference/glossary/" \l "term-database) named **mydb**.

MongoDB will create a collection implicitly upon its first use. You do not need to create a collection before inserting data. Furthermore, because MongoDB uses [dynamic schemas](http://docs.mongodb.org/manual/faq/fundamentals/" \l "faq-schema-free), you also need not specify the structure of your documents before inserting them into the collection.

1. From the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell, confirm you are in the **mydb** database by issuing the following:

db

1. If **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** does not return **mydb** for the previous operation, set the context to the **mydb** database, with the following operation:

use mydb

1. Create two documents named **j** and **k** by using the following sequence of JavaScript operations:

j = { name : "mongo" }

k = { x : 3 }

1. Insert the **j** and **k** documents into the **testData** collection with the following sequence of operations:

db.testData.insert( j )

db.testData.insert( k )

When you insert the first document, the **[mongod](http://docs.mongodb.org/manual/reference/program/mongod/" \l "bin.mongod)** will create both the **mydb** database and the **testData** collection.

1. Confirm that the **testData** collection exists. Issue the following operation:

show collections

The **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell will return the list of the collections in the current (i.e. **mydb**) database. At this point, the only collection is**testData**. All **[mongod](http://docs.mongodb.org/manual/reference/program/mongod/" \l "bin.mongod)** databases also have a **[system.indexes](http://docs.mongodb.org/manual/reference/system-collections/" \l "<database>.system.indexes)** collection.

1. Confirm that the documents exist in the **testData** collection by issuing a query on the collection using the **[find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "db.collection.find)** method:

db.testData.find()

This operation returns the following results. The [ObjectId](http://docs.mongodb.org/manual/reference/object-id/) values will be unique:

{ "\_id" : ObjectId("4c2209f9f3924d31102bd84a"), "name" : "mongo" }

{ "\_id" : ObjectId("4c2209fef3924d31102bd84b"), "x" : 3 }

All MongoDB documents must have an **\_id** field with a unique value. These operations do not explicitly specify a value for the **\_id**field, so **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** creates a unique [ObjectId](http://docs.mongodb.org/manual/reference/object-id/) value for the field before inserting it into the collection.

## **Insert Documents using a For Loops or JavaScript Function**

To perform the remaining procedures in this tutorial, first add more documents to your database using one or both of the procedures described in [Generate Test Data](http://docs.mongodb.org/manual/tutorial/generate-test-data/).

## **Working with the Cursor**

When you query a [collection](http://docs.mongodb.org/manual/reference/glossary/" \l "term-collection), MongoDB returns a “cursor” object that contains the results of the query. The **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell then iterates over the cursor to display the results. Rather than returning all results at once, the shell iterates over the cursor 20 times to display the first 20 results and then waits for a request to iterate over the remaining results. In the shell, use enter **it** to iterate over the next set of results.

The procedures in this section show other ways to work with a cursor. For comprehensive documentation on cursors, see [Iterate the Returned Cursor](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "crud-read-cursor).

### **Iterate over the Cursor with a Loop**

Before using this procedure, make sure to add at least 25 documents to a collection using one of the procedures in [Generate Test Data](http://docs.mongodb.org/manual/tutorial/generate-test-data/). You can name your database and collections anything you choose, but this procedure will assume the database named **test** and a collection named **testData**.

1. In the MongoDB JavaScript shell, query the **testData** collection and assign the resulting cursor object to the **c** variable:

**var** c = db.testData.find()

1. Print the full result set by using a **while** loop to iterate over the **c** variable:

**while** ( c.hasNext() ) printjson( c.next() )

The **hasNext()** function returns true if the cursor has documents. The **next()** method returns the next document. The**printjson()** method renders the document in a JSON-like format.

The operation displays 20 documents. For example, if the documents have a single field named **x**, the operation displays the field as well as each document’s **ObjectId**:

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990be6"), "x" : 1 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990be7"), "x" : 2 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990be8"), "x" : 3 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990be9"), "x" : 4 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bea"), "x" : 5 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990beb"), "x" : 6 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bec"), "x" : 7 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bed"), "x" : 8 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bee"), "x" : 9 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bef"), "x" : 10 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf0"), "x" : 11 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf1"), "x" : 12 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf2"), "x" : 13 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf3"), "x" : 14 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf4"), "x" : 15 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf5"), "x" : 16 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf6"), "x" : 17 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf7"), "x" : 18 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf8"), "x" : 19 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf9"), "x" : 20 }

### Use Array Operations with the Cursor

The following procedure lets you manipulate a cursor object as if it were an array:

1. In the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell, query the **testData** collection and assign the resulting cursor object to the **c** variable:

**var** c = db.testData.find()

1. To find the document at the array index **4**, use the following operation:

printjson( c [ 4 ] )

MongoDB returns the following:

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bea"), "x" : 5 }

When you access documents in a cursor using the array index notation, **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** first calls the **cursor.toArray()**method and loads into RAM all documents returned by the cursor. The index is then applied to the resulting array. This operation iterates the cursor completely and exhausts the cursor.

For very large result sets, **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** may run out of available memory.

For more information on the cursor, see [Iterate the Returned Cursor](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "crud-read-cursor).

### Query for Specific Documents

MongoDB has a rich query system that allows you to select and filter the documents in a collection along specific fields and values. See [Query Documents](http://docs.mongodb.org/manual/tutorial/query-documents/" \l "read-operations-query-document) and [Read Operations](http://docs.mongodb.org/manual/core/read-operations/) for a full account of queries in MongoDB.

In this procedure, you query for specific documents in the **testData** [collection](http://docs.mongodb.org/manual/reference/glossary/" \l "term-collection) by passing a “query document” as a parameter to the **[find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "db.collection.find)** method. A query document specifies the criteria the query must match to return a document.

In the **[mongo](http://docs.mongodb.org/manual/reference/program/mongo/" \l "bin.mongo)** shell, query for all documents where the **x** field has a value of **18** by passing the **{ x : 18 }** query document as a parameter to the **[find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "db.collection.find)** method:

db.testData.find( { x : 18 } )

MongoDB returns one document that fits this criteria:

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990bf7"), "x" : 18 }

### Return a Single Document from a Collection

With the **[findOne()](http://docs.mongodb.org/manual/reference/method/db.collection.findOne/" \l "db.collection.findOne)** method you can return a single document from a MongoDB collection. The **[findOne()](http://docs.mongodb.org/manual/reference/method/db.collection.findOne/" \l "db.collection.findOne)** method takes the same parameters as **[find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "db.collection.find)**, but returns a document rather than a cursor.

To retrieve one document from the **testData** collection, issue the following command:

db.testData.findOne()

For more information on querying for documents, see the [Read Operations](http://docs.mongodb.org/manual/core/read-operations/) and [Read Operations](http://docs.mongodb.org/manual/core/read-operations/) documentation.

### Limit the Number of Documents in the Result Set

To increase performance, you can constrain the size of the result by limiting the amount of data your application must receive over the network.

To specify the maximum number of documents in the result set, call the **[limit()](http://docs.mongodb.org/manual/reference/method/cursor.limit/" \l "cursor.limit)** method on a cursor, as in the following command:

db.testData.find().limit(3)

MongoDB will return the following result, with different [ObjectId](http://docs.mongodb.org/manual/reference/object-id/) values:

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990be6"), "x" : 1 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990be7"), "x" : 2 }

{ "\_id" : ObjectId("51a7dc7b2cacf40b79990be8"), "x" : 3 }

# **Read Operations**

Read operations, or [queries](http://docs.mongodb.org/manual/reference/glossary/" \l "term-query), retrieve data stored in the database. In MongoDB, queries select [documents](http://docs.mongodb.org/manual/reference/glossary/" \l "term-document) from a single [collection](http://docs.mongodb.org/manual/reference/glossary/" \l "term-collection).

Queries specify criteria, or conditions, that identify the documents that MongoDB returns to the clients. A query may include a projectionthat specifies the fields from the matching documents to return. The projection limits the amount of data that MongoDB returns to the client over the network.

## **Query Interface**

For query operations, MongoDB provide a **[db.collection.find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "db.collection.find)** method. The method accepts both the query criteria and projections and returns a [cursor](http://docs.mongodb.org/manual/core/cursors/) to the matching documents. You can optionally modify the query to impose limits, skips, and sort orders.

The following diagram highlights the components of a MongoDB query operation:



MongoDB find() statement annotated with MongoDB terms.

The next diagram shows the same query in SQL:



SQL SELECT statement annotated with corresponding MongoDB terms.

**Example**

db.users.find( { age: { $gt: 18 } }, { name: 1, address: 1 } ).limit(5)

This query selects the documents in the **users** collection that match the condition **age** is greater than **18**. To specify the greater than condition, query criteria uses the greater than (i.e. **[$gt](http://docs.mongodb.org/manual/reference/operator/gt/" \l "op._S_gt)**) [query selection operator](http://docs.mongodb.org/manual/reference/operator/" \l "query-selectors). The query returns at most **5** matching documents (or more precisely, a cursor to those documents). The matching documents will return with only the **\_id**, **name** and**address** fields. See [Projections](http://docs.mongodb.org/manual/core/read-operations/" \l "projections) for details.

## **Query Behavior**

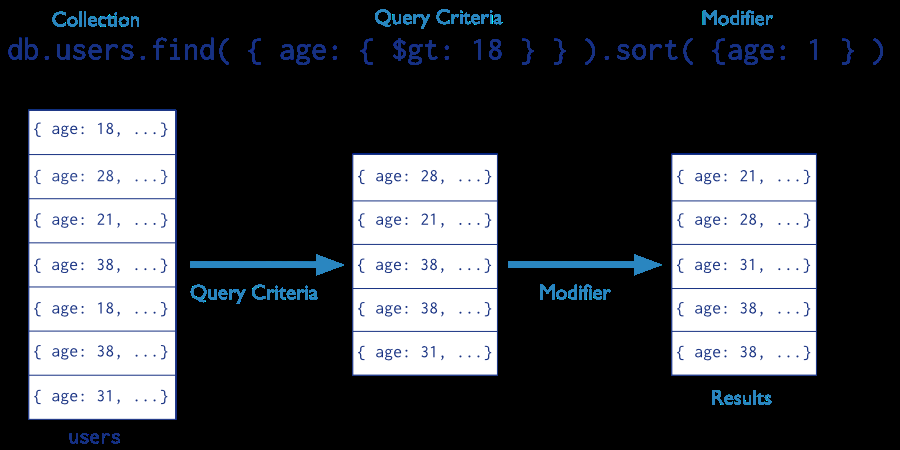
MongoDB queries exhibit the following behavior:

* All queries in MongoDB address a single collection.
* You can modify the query to impose **[limits](http://docs.mongodb.org/manual/reference/method/cursor.limit/" \l "cursor.limit)**, **[skips](http://docs.mongodb.org/manual/reference/method/cursor.skip/" \l "cursor.skip)**, and **[sort orders](http://docs.mongodb.org/manual/reference/method/cursor.sort/" \l "cursor.sort)**.
* The order of documents returned by a query is not defined and is not necessarily consistent unless you specify a **[sort()](http://docs.mongodb.org/manual/reference/method/cursor.sort/" \l "cursor.sort)**.
* Operations that [modify existing documents](http://docs.mongodb.org/manual/tutorial/modify-documents/) (i.e. updates) use the same query syntax as queries to select documents to update.
* In [aggregation](http://docs.mongodb.org/manual/core/aggregation/) pipeline, the **[$match](http://docs.mongodb.org/manual/reference/aggregation/match/" \l "pipe._S_match)** pipeline stage provides access to MongoDB queries.

MongoDB provides a **[db.collection.findOne()](http://docs.mongodb.org/manual/reference/method/db.collection.findOne/" \l "db.collection.findOne)** method as a special case of **[find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "db.collection.find)** that returns a single document.

## **Query Statements**

Consider the following diagram of the query process that specifies a query criteria and a sort modifier:



Stages of a MongoDB query with a query criteria and a sort modifier.

In the diagram, the query selects documents from the **users** collection. Using a [query selection operator](http://docs.mongodb.org/manual/reference/operator/) to define the conditions for matching documents, the query selects documents that have **age** greater than (i.e. **[$gt](http://docs.mongodb.org/manual/reference/operator/gt/" \l "op._S_gt)**) **18**. Then the **[sort()](http://docs.mongodb.org/manual/reference/method/cursor.sort/" \l "cursor.sort)** modifier sorts the results by **age** in ascending order.

For additional examples of queries, see [Query Documents](http://docs.mongodb.org/manual/tutorial/query-documents/).

## **Projections**

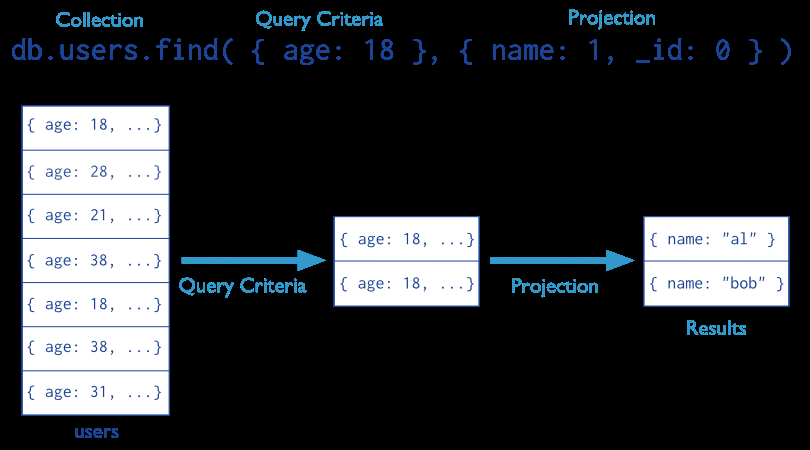
Queries in MongoDB return all fields in all matching documents by default. To limit the amount of data that MongoDB sends to applications, include a [projection](http://docs.mongodb.org/manual/reference/glossary/" \l "term-projection) in the queries. By projecting results with a subset of fields, applications reduce their network overhead and processing requirements.

Projections, which are the the second argument to the **[find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/" \l "db.collection.find)** method, may either specify a list of fields to return or list fields to exclude in the result documents.

**Important**

Except for excluding the **\_id** field in inclusive projections, you cannot mix exclusive and inclusive projections.

Consider the following diagram of the query process that specifies a query criteria and a projection:



Stages of a MongoDB query with a query criteria and projection.

In the diagram, the query selects from the **users** collection. The criteria matches the documents that have **age** equal to **18**. Then the projection specifies that only the **name** field should return in the matching documents.

### **Projection Examples**

#### **Exclude One Field From a Result Set**

db.records.find( { "user\_id": { $lt: 42} }, { history: 0} )

This query selects a number of documents in the **records** collection that match the query **{ "user\_id": {$lt: 42} }**, but excludes the **history** field.

#### Return Two fields and the \_id Field

db.records.find( { "user\_id": { $lt: 42} }, { "name": 1, "email": 1} )

This query selects a number of documents in the **records** collection that match the query **{ "user\_id": {$lt: 42} }**, but returns documents that have the **\_id** field (implicitly included) as well as the **name** and **email**fields.

#### Return Two Fields and Exclude \_id

db.records.find( { "user\_id": { $lt: 42} }, { "\_id": 0, "name": 1 , "email": 1 } )

This query selects a number of documents in the **records** collection that match the query **{ "user\_id": {$lt: 42} }**, but only returns the **name** and **email** fields.

### Projection Behavior

MongoDB projections have the following properties:

* In MongoDB, the **\_id** field is always included in results unless explicitly excluded.
* For fields that contain arrays, MongoDB provides the following projection operators: **[$elemMatch](http://docs.mongodb.org/manual/reference/projection/elemMatch/" \l "proj._S_elemMatch)**, **[$slice](http://docs.mongodb.org/manual/reference/projection/slice/" \l "proj._S_slice)**, **[$](http://docs.mongodb.org/manual/reference/projection/positional/" \l "proj._S_)**.
* For related projection functionality in the [aggregation framework](http://docs.mongodb.org/manual/core/aggregation/) pipeline, use the **[$project](http://docs.mongodb.org/manual/reference/aggregation/project/" \l "pipe._S_project)** pipeline stage