Taking an Exam

You are about to take a MongoDB Certification Exam. We want you to be successful so we've prepared this guide to help you study. With a good understanding of the subjects covered here, you should be well prepared to pass. Good luck!

Proctoring

You will take the exam at home, at your computer. When you take the exam, there will be a live proctor from our partner, Examity, observing. He or she will take you through the steps, help you begin your test, and will let you know if anything is out of order so you don't violate exam rules without knowing it.

Your Computer

Your computer must be a Windows PC or Mac (Sorry, no Linux yet!) with a video camera (webcam), a microphone, and speakers. The webcam can be built-in or it can be a peripheral, but you need to be able to position it so that the proctor can see your face throughout the exam. You must be able to pivot the camera (possibly by picking up your laptop) so that the proctor can see everything in the room including your desk. You can't use headphones, but do need to use speakers so that you can hear the proctor.

Before you even schedule the exam, please go through Examity's Computer Readiness Check. This will tell you if there are any problems with your webcam, your internet connection, your microphone, etc. If you are going to have problems, it's best to find out early.

Your Test-Taking Environment

You need to be alone when taking the exam. No one else should be in the room. You may not have any reference materials present and your desk area must be free of clutter. This will be verified by the proctor before the exam begins. You may take the exam at any time during the exam session for which you are registered, day or night. Exam sessions run for one week. They typically begin and end at 17:00 UTC on a Tuesday, but be sure to check the about page for your exam (DBA or Developer) for the exact times.

Scoring and Test-Taking Strategies

The exam currently has two types of problems: multiple choice and check-all-that-apply. Most questions will have 5 choices for multiple choice or 3 for check-all-that-apply.

There is no penalty for guessing, so you should have an answer for every problem. If you find that you are getting close to running out of time, be sure to run through and provide an

answer for each question. Leaving questions blank will guarantee that you don't get credit.

Translation Software

You may use translation software in order to translate the questions from English to another language. We do not support any particular translation software. We do not vouch for the translation accuracy of any software. However, you are welcome to use the translation software of your choice during the test if you are willing to assume any risks associated with the software.

General Computing Knowledge

Introduction

This section will describe the foundational computing concepts with which you should be familiar. You should consider these to be prerequisites to understanding MongoDB. We do not teach these directly in our curriculum, but some questions on the exam assume knowledge of these concepts. You can find detailed information on these subjects in computer science textbooks or through a little Googling.

Fundamental Database Concepts

It is expected that you will understand broadly what a database is, specifically:

- Records (i.e. rows/documents)
- Tables/collections
- Databases
- Authentication and authorization concepts
- Joins and how they work in relational databases
- Transactions and a basic understanding related ideas such as commits and rollback.

Memory Management and Data Representation

For the exam, you should have a working understanding of the following:

- Physical memory
- Virtual memory
- Memory mapping
- Hexidecimal (base-16) representations of data

Basics of JavaScript Programming

For the exam, you should know:

- How to assign values to variables in JavaScript
- How to iterate (e.g., using *for* or *while*)

Philosophy and Features

In the Philosophy and Features section on the exam, we will verify that you understand the following:

- The key features of MongoDB and what functionality they provide
- Fundamentals of JSON and BSON
- The MongoDB data model at a high level
- The MongoDB query model at a high level
- Data consistency and availability in MongoDB

The MongoDB Architecture Guide and this video lecture provide a concise overview of these subjects. We expand on this material below.

JSON

For the exam you should know:

- What data types JSON supports, e.g., objects and arrays
- The structure of JSON objects and arrays
- How to nest data in JSON objects and arrays
- How to read JSON

Resources:

- Videos:
  - Introduction to JSON
  - JSON revisited
  - JSON spec
- Docs:
  - Intro to JSON
- External links:
  - json.org

BSON

For the exam you should know:

- That BSON is a binary JSON data format
- What we mean in describing BSON as lightweight, traversable, and efficient
- How BSON encodes a value using explicit type information, a binary representation of the value, and explicit byte length.

Resource:

BSON specification BSON lecture

The Mongo Shell

For the exam you should know:

- How to list available databases/collections in the mongo shell
- How to switch to a particular database context
- How to write JavaScript code to create sample data and for other simple tasks
- How to print output using the "print" function
- Administrative commands available in the Mongo shell

Resources:

- Video:
  - Lesson video
- Docs:
  - Getting Started with the mongo Shell
  - mongo Shell Quick Reference

You should also know how to work with data types in the shell.

Shell Examples

To assign a value to a variable, you use the '=' sign. Let's assign the value of 5 to the 'x' variable:

```
> var x = 5;
> print(x)
5
```

JavaScript is lax about syntax so the following will also work. Note the lack of the *var* keyword and semicolons.

```
> x = 5
5
> print(x)
5
```

JavaScript and the shell include a number of built-in objects and functions that support a variety of uses. For example, *Math* is a built-in class with methods supporting a number of

common mathematical operations. *ISODate* is another class commonly used when date objects are required.

```
> Math.floor(3.5)
3
> Math.PI
3.141592653589793
> ISODate()
ISODate("2015-07-22T19:04:34.219Z")
> print("hello")
hello
>
```

Iteration in the shell (and in JavaScript in general) has the following syntax. Here we are using the most compact form. You may choose to write loops such as this with each statement and the curly braces on separate lines.

```
> var x = 0
> while ( x < 5 ) { x++; print(x) }
1
2
3
4
5
> for (i=0; i<=3; i++) { print(i) }
0
1
2
3
> x = { a : 4, b : 3, c : 2 }
{ "a" : 4, "b" : 3, "c" : 2 }
> for (key in x) { print(key + ': ' + x[key]) }
a: 4
b: 3
c: 2
>
```

In the mongo shell (and drivers), you will work with database handles and collection objects.

```
> db
test
```

```
> db.currentOP()
{ "inprog" : [ ] }
> db.people.find()
{ "_id" : ObjectId("55afeb6a6daadd6baf25c63b"), "name"
: "Dwight", "title" : "Chair of the Board" }
{ "_id" : ObjectId("55afeb7b6daadd6baf25c63c"), "name"
: "Andrew", "title" : "VP of Developer Experience" }
```

Shell Exercises

The following are exercises you may find useful in ensuring you understand the above.

- Write a for loop that inserts one million documents in the mongo shell.
- Create one collection with a field that contains the _id of documents in second collection. Perform a series of find operations that use the results of a query against the first collection to identify the _ids of documents to retrieve from the second collection.

Vertical and Horizontal Scaling

For the exam you should know:

- The difference between vertical and horizontal scaling
- That sharding is MongoDB's approach to horizontal scaling

Resources:

- Webinar:
  - Scaling MongoDB Webinar Series
- Docs:
  - Sharding Introduction

MongoDB and Relational Databases

For the exam you should know:

- The features typically found in relational database management systems that MongoDB does not include for scalability reasons
- How relational data models typically differ from data models in MongoDB

Resources:

- Video:
  - MongoDB and Relational Databases
- Docs:

    - SQL to MongoDB Mapping Chart
- MongoDB.com:
    - Compare MongoDB and MySQL

Flexible Schema in MongoDB

For the exam you should understand:

- Why we say MongoDB has a flexible schema (sometimes called "dynamic schema")
- How this is different from relational databases
- Atomicity concerns with regard to write operations in MongoDB

Resources:

- Video:
    - MongoDB Flexible Schema
- Docs:
    - Data Modeling Introduction

Sample Problems

1. In order to ensure that you can maintain high availability in the face of server failure, you should configure which of the following?
    a. Replication
    b. Sharding
    c. The proper storage engine
    d. Properly defined user roles
    e. Put indexes on all of your documents
2. Which of the following are common uses of the mongo shell? Check all that apply.
    a. Perform queries against MongoDB collections
    b. Perform administrative tasks
    c. Use as a JavaScript framework for your production system

Answers to Sample Problems

1. a
2. a, b

Suggested Exercises for the Reader

- Consider a simple JSON document and determine how it will be encoded as BSON.
- Write a simple routine in JavaScript that creates and inserts some sample data into a collection.
- Experiment with a few find operations to ensure the data was inserted correctly.

CRUD

In the CRUD section of the certification exam, we will verify that you:

- Understand all create, read, update, and delete (CRUD) operations in the MongoDB query language
- Are familiar with commonly used CRUD operators and how to use them
- Know what data types MongoDB supports

In this section, we are not testing that you have the MongoDB query language syntax memorized. However, you should be able to distinguish correctly written queries from those that are not. You should also know which query parameters are necessary versus those that are optional and how to use query parameters. We will not expect you to have query operators memorized, but you should be able to recognize the correct operator for a task from a set of choices.

Create

For the exam, you should be able to:

- Properly use the insert, save, update, and findAndModify commands to create new documents
- Match insert commands with descriptions of what they do
- Know how to perform bulk inserts
- Understand the uniqueness constraint of _id fields and its implications for CRUD operations
- Understand how ObjectIds are created and used in MongoDB

See the ObjectId documentation for details on these values and their use in MongoDB.

Document creation can occur through a few commands:

- db.collection.insert()
  - Docs:
    - db.collection.insert()
    - Inserting Documents Tutorial
  - Video:
    - insertOne()
    - insertMany()
- db.collection.save() will cause a document to be inserted if the _id is not included, or the _id doesn't match any documents.
  - Docs:
    - db.collection.save()
  - Video:

        - The Save Command
- db.collection.update() will insert documents in some circumstances ("upserts").
    - Docs:
        - Upsert (only an insert if upsert: true is set)
    - Video:
        - Upsert lesson video
- db.collection.findAndModify() can result in an insert if updating and "upsert" is set to true.
    - Docs:
        - db.collection.findAndModify()

Most questions about document creation will involve the db.collection.insert() command. Inserts are typically straightforward.

Upserts can be more complicated. In the example below, assume the foo collection does not already contain a document with a=5 and b<=7.

```
> db.foo.update( { a : 5, b : { $lte : 7 } }, { $set :
{ c : 8 } }, { upsert : true } )
WriteResult({
    "nMatched" : 0,
    "nUpserted" : 1,
    "nModified" : 0,
    "_id" : ObjectId("55b0200e5ef34083de46367e")
})
> db.foo.find()
{ "_id" : ObjectId("55b0200e5ef34083de46367e"), "a" :
5, "c" : 8 }
```

In cases such as this, a new document is inserted. In this specific case, the new document contains the value, c : 8, because the upsert instructed that it be set. The query document also contributes fields to the document that gets created. In this case, a : 5 was also be set. The value of b could not be determined from the query, so it will not be set. Finally, an ObjectId was assigned to the _id field.

You should also be familiar with the special properties of the _id field:

- Docs:
    - The _id Field in documents
    - The _id Field
- Video:
    - Compound _id Fields

Finally, you can bulk insert by passing an array to db.collection.insert(). You should know the difference between ordered and unordered bulk inserts.

Docs:

- Insert Multiple Documents

Read

In MongoDB, you read documents using either the db.collection.find() method, or the db.collection.findAndModify() method. You should be familiar with both commands, but the .find() method will receive greater coverage on the exam.

For the exam, you should be able to:

- Correctly use the query, projection, and options parameters
- Sort query results
- Demonstrate an understanding of all match and projection operators
- Read and interpret regular expressions as used in MongoDB queries
- Demonstrate an understanding of how find() queries work with arrays

In MongoDB, reading data is carried out with the following methods:

db.collection.find()

- Docs:
  - db.collection.find()
  - find() tutorial
- Videos:
  - Introduction to find()
  - Reading Scalar Fields
  - Reading Array Fields
  - Projections
  - Operators for db.collection.find():
    - Comparison Operators
    - Element Operators
    - Regex operator
    - Logical Operators
  - Array Operators:
    - Array Operators: $all
    - Array Operator: $size
    - Array Operators: $elemMatch

Keep in mind, a find() query will return a cursor:

- Docs:

Cursor tutorial

- Videos:
    - Cursor lesson

There are other collection read methods that will not return cursors, but with which you should be familiar.

- db.collection.count()
    - Docs:
        - count()
    - Videos:
        - count() lesson
- db.collection.distinct()
    - Docs:
        - distinct()

There are other methods that can be applied to cursors themselves. These can return a number (e.g., count), or they can modify the result set (i.e., sort, skip, and limit). You may also be expected to know how to manually iterate a cursor. See the MongoDB documentation for a list of cursor methods

- cursor.count()
    - Works like collection.count()
    - Docs:
        - cursor.count()
- Lesson on sort, skip, and limit
- cursor.sort()
    - Sort documentation
    - Second lesson on sorting
- cursor.skip()
    - Skip documentation
- cursor.limit()
    - Limit documentation
- cursor.next()
    - Next documentation
    - The lesson on cursors, above, reproduced here, also shows how to do this

You can also project your results in order to limit the fields you get back.

- Projection documentation
- .find() documentation
- Projection lesson

Update

Updates modify existing documents. For the exam, you should be able to:

- Properly use the save, update, and findAndModify commands to mutate existing documents
- Distinguish which parameter finds the documents to change, which mutates them
- Explain the behavior of any update operator with which you are presented
- Recognize when upserts and db.collection.save() will insert documents

Updates can occur with a few collection methods, some of which were in the insert section:

- db.collection.save()
    - This will update if the _id is specified and it matches an existing document.
    - save() documentation
    - save() lesson video
- db.collection.findAndModify()
    - findAndModify() documentation
- db.collection.update()
    - This will update unless upsert: true is specified and the query matches no documents.
    - update() documentation
    - update() tutorial
    - With no operators specified in the update parameter, there will be a wholesale update
        - Wholesale Update Lesson Video
    - Operators
        - Update Operators
        - $set (modify a field)
            - $set lesson video
            - $set documentation
        - $unset (remove a field)
            - $unset lesson video
            - $unset documentation
        - $rename
            - $rename documentation
        - $setOnInsert
            - $setOnInsert documentation
        - $inc
            - $inc documentation
        - $mul
            - $mul documentation
        - $min and $max
            - $min documentation

        - $max documentation
    - Array operators
        - Array operators lesson video
        - Positional $ documentation
        - $addToSet documentation
        - $pop documentation
        - $pull documentation
        - $pullAll documentation
        - $push documentation
            - There are additional modifiers if you use $push (and sometimes $addToSet)
            - $each documentation
            - $slice documentation
            - $sort documentation
            - $position documentation
  - Multi updates
    - Multi update lesson video
  - Update Many
    - updateMany()

Delete

For the exam, you should be able to:

- Drop a collection
- Build a query that deletes only the documents you want it to

Docs:

- Dropping a collection
- db.collection.remove()

Video:

- deleteOne(), deleteMany()

Sample Problems

1. Consider the following documents:

```
{ "_id" : 1, "a" : 1, "b" : 1 }
{ "_id" : 2, "a" : 2, "b" : 3 }
{ "_id" : 3, "a" : 3, "b" : 6 }
```

```
{ "_id" : 4, "a" : 4, "b" : 10 }
{ "_id" : 5, "a" : 5, "b" : 15 }
```

You perform the following query:

```
db.stuff.updateMany( { b : { $gte : 10 } },
                     { $set : { b : 15 } } )
```

How many documents will be modified?

a. 0
b. 1
c. 2
d. 3
e. 5

2. Consider the following document:

```
> db.c.find()
{ "_id" : 12, b : [ 3, 5, 7, 2, 1, -4, 3, 12 ] }
```

Which of the following queries on the "c" collection will return only the first five elements of the array in the "b" field? E.g.,

Document you want returned by your query:

```
{ "_id" : 12, "b" : [ 3, 5, 7, 2, 1 ] }
```

a. db.c.find({ } , { b : [0, 1, 2, 3, 4, 5] })
b. db.c.find({ } , { b : [0 , 5] })
c. db.c.find({ } , { b : { $slice : [0 , 5] } })
d. db.c.find({ } , { b : { $substr[0 , 5] } })
e. db.c.find({ b : [0 , 5] })

3. Consider the following example document from the sample collection. All documents in this collection have the same schema.

```
{
  "_id" : 3,
  "a" : 7,
```

```
  "b" : 4
}
```

Which of the following queries will replace this with the document,

```
{
  "_id" : 7,
  "c" : 4,
  "b" : 4
}
```

a. db.sample.update({ "_id" : 3 } , { "_id" : 7 , "c" : 4 })
b. db.sample.update({ "_id" : 3 } , { "$set" : { "_id" : 7 , "c" : 4 } })
c. db.sample.update({ "_id" : 3 } , { "_id" : 7 , "c" : 4 , { "$unset" : ["a" , "b"] } })
d. db.sample.update({ "_id" : 3 } , { "_id" : 7 , "c" : 4 } , { "justOne" : true })
e. This operation cannot be done with a single query.

4. Which of the documents below will be retrieved by the following query? Assume the documents are stored in a collection called "sample". Check all that apply.

```
  db.sample.find( { "$or" : [ { "a" : { "$in" : [
3, 10] } }, { "b" : { "$lt" : 2 } } ] } )

a. { "_id" : 1, "a" : 0, "c" : 0, "b" : 2 }
b. { "_id" : 2, "a" : 2, "c" : 0, "b" : 1 }
c. { "_id" : 3, "a" : 4, "c" : 0, "b" : 14 }
d. { "_id" : 4, "a" : 5, "c" : 0, "b" : 17 }
e. { "_id" : 5, "a" : 3, "c" : 0, "b" : 12 }
f. { "_id" : 6, "a" : 1, "c" : 1, "b" : 5 }
g. { "_id" : 7, "a" : 8, "c" : 1, "b" : 7 }
h. { "_id" : 8, "a" : 11, "c" : 1, "b" : 0 }
i. { "_id" : 9, "a" : 17, "c" : 1, "b" : 1 }
j. { "_id" : 10, "a" : 3, "c" : 1, "b" : 1 }
```

5. You perform the following operation in the shell:

```
db.foo.insert( { } );
```

What gets inserted?

a. An empty document

b. A document with an _id assigned to be an ObjectId
c. A document that matches the collection's existing schema, but with null fields
d. No document will be inserted; an error will be raised
e. A document will be inserted with the same _id as the last document inserted

Answers to Sample Problems

1. b
2. c
3. e
4. b, e, h, i, j
5. b

Exercises

Here are some exercises you can complete on your own:

- Insert some sample data. Write a for loop that bulk inserts 1,000 documents at a time, 1000 times, for a total of 1 million documents inserted. Do it in such a way that each of its _id fields is unique.
- Perform the same operation, but use some (but not all) non-unique _id field values. See the difference between an ordered vs. unordered bulk insert.
- Perform queries on the sample set. Find documents with fields greater than certain values.
- Perform updates on the sample set. Set new fields without deleting old fields, increment fields, etc. using the update command.
- Perform array operations, pushing to and popping from arrays.
- Insert documents with nested subdocuments. Query on the subdocument fields.

Indexes

On the certification exam, we will verify that you:

- Understand the types of indexes available in MongoDB
  - Single Field Indexes
  - Compound Indexes
  - Multikey Indexes
  - Special Indexes
    - Geo Indexes
    - Text Indexes
- Know the options you can have for an index
  - TTL
  - Sparse

    - Unique
- Know how to improve the efficiency of a query using indexes
- Understand the write performance costs of indexes

Introduction

The following provide a basic introduction to indexes.

- Videos:
    - Introduction to Indexes
- Docs:
    - Index Concepts

Creating Indexes

On the exam, we will ensure that you:

- Know how to create an index
- Know that the _id field is created implicitly if you do not supply one
- Videos:
    - createIndex(), getIndexes(), dropIndex()
    - Creating Indexes
- Docs:
    - db.collection.createIndex()
    - Create an Index

Single Field Indexes

For the exam, you should be able to:

- Recognize single-field indexes
- Know when a single-field index is used (and when it is not)
    - for .find() queries
    - for .update() queries
    - for .remove() queries
- Know how to create a single-field index on a field in a subdocument

Here are some resources to help:

- Docs:
    - Single Field Indexes
    - Embedded Fields
    - Embedded Documents

Collection Scans

For the exam, you should know:

- That a "collection scan" happens when every document in the collection must be checked in order to determine the result set of a query
- Whether a collection scan will occur, given a query and list of available indexes
- Why a collection scan is undesirable
- Videos:
    - Collection scans

Compound Indexes

On the exam, you should know:

- What a compound index is
- How to use a prefix of a compound index to satisfy queries

Docs:

- Compound Indexes

Multikey Indexes

On the exam, you should know:

- How to distinguish multikey indexes from other index types
- Restrictions on multikey indexes
- How many index keys will be created for a particular document in a multikey index

A multikey index is an index on an array field. The index will contain one key per array element.

- Docs:
    - Multikey Indexes
- Videos:
    - Multikey Indexes (video)
    - Dot Notation and Multikey

Sorting with Indexes

For the exam, you will need to know:

- How to sort with an index
- How to use a compound index to both filter and sort
- How to perform a compound sort using a compound index
- When an index will or will not work for a compound sort

Docs:

- Use Indexes to Sort Query Results

The .explain() Methods

For the exam, you should know:

- How to create an Explainable object with db.collection.explain() and use it to explain a cursor
- How to explain a cursor with cursor.explain()
- The three verbosity settings of explain plans, and what they offer.
- How to read each type of explain plan to determine things such as:
    - How many documents were returned by the query
    - How many documents were read by the query
    - How many index entries were viewed by the query
    - Which index was used by the query
    - When a collection scan occurs
    - How many index entries were viewed during the query
    - Which shards were involved in the query for a sharded collection
    - How to recognize that a query is covered
    - Whether or not an index was used to sort the query
    - How long the query took (or was estimated to take)
    - Which types of queries can use an index (.find(), .update(), .remove())

Resources:

- Videos:
    - Explain Plans
    - Using Explain
    - Explain Verbosity
- Docs:
    - cursor.explain()
    - db.collection.explain()
    - Explain Results

Selecting an Index

For the exam, you should know:

- How an index is chosen when multiple indexes may work
- When the query optimizer is re-run
- Videos:
    - Choosing an Index
    - Efficiency of Index Use

- Docs:
  - Query Plans

Covered Queries

For the exam, you will need to understand:

- Covered queries
- Why covered queries are good for performance
- How to recognize that a covered query has happened in an explain plan
- Video:
  - Covered Queries
- Docs:
  - Covered Query

Indexing Strategies

For the exam, you should know:

- How to create an index that supports a query that sorts on one field, queries for an exact match on a second field, and does a range query on a third field
- When an index can be used to sort for a particular query
- How selective queries are and how much they are likely to benefit from an index

Resources:

- Docs:
  - Indexing Strategies
  - Optimize Query Performance
  - Indexes FAQ
  - Query Optimization
- Videos:
  - Efficiency of Index Use 2
  - Index Notes

Effect of Indexes on Write Performance

Indexes generally speed up read performance, but can slow down write performance. Hybrid actions (e.g., with a find and a write) such as update and remove, may depend on the use case (though they usually are faster with indexes).

For the exam, you will want to know:

- Why indexes can slow down write operations
- Why document growth is a problem in MMAPv1 for indexes

- Why update and delete operations can either benefit or suffer in performance due to indexes (but usually benefit)

Resources:

- Docs:
  - Write Operation Performance
- Video:
  - Read and Write Impact of Indexes

Unique Indexes

For the exam, you should know:

- How to create a unique index
- How to recognize a unique index from the db.collection.getIndexes() command
- What happens when you try to insert a document with a value that matches an existing document for a unique indexed field
- How unique compound indexes work
- What happens if you try to create a unique index on a collection that already contains documents with non-unique values for the unique field(s)

Resources:

- Video:
  - Index Creation Options, Unique
- Docs:
  - Creating a Unique Index
  - Unique Indexes

Sparse Indexes

For the exam, you should know:

- How to create a sparse index
- How many index keys you will have for a sparse index on a small collection
- When a sparse index would not automatically be used (ie, for sorting)

Resources:

- Video:
  - Sparse Indexes Video
- Docs:
  - Sparse Indexes

TTL Indexes

For the exam, you should know:

- How to create a TTL index
- How to recognize a TTL index in the output of db.collection.getIndexes()
    - And recognize the time before deletion occurs
- Know when deletion of documents will definitely not occur, and when it may occur

Resources:

- Video:
    - TTL Indexes
- Docs:
    - Indexes, TTL

Background Index Creation

For the exam, you will need to know:

- How to create background indexes
- What operations can occur if an index is created in the background rather than foreground
- How the time required to build a background index differs from the time required to build an index in the foreground
- How using a background index will differ from using an index built in the foreground (hint: you use them the same way)

Resources:

- Video:
    - Background Index Creation
- Docs:
    - Build Indexes in the Background
    - Background Index Creation

Regex on String Fields and Indexes

For the exam, you will need to know:

- How standard indexes behave with strings fields (as compared to text indexes)
- How to use indexes most efficiently by anchoring the regex on the left

Resources:

- Index Use in Regex

Text Indexes

For the exam, you will need to know:

- How to build a text index
- How to use a text index to query
- How to sort results by text score

Resources:

- Video:
  - Text Indexes in MongoDB
- Docs:
  - Text Indexes
  - Create a Text Index
  - Text Search Tutorials

2d and 2dSphere Indexes

For the exam, you will need to know:

- How to create 2dsphere indexes
- How to create geoJSON points for a 2dsphere indexed field in MongoDB
- How to query for geoJSON points:
  - Within a circle
  - Near a point
  - Within a polygon

Resources:

- Video:
  - Geospatial Indexes
- Docs:
  - Geospatial Indexes and Queries
  - 2d Indexes
  - Create a 2d Index
  - Query a 2d Index
  - 2dsphere Indexes
  - Create a 2dsphere Index
  - Query a 2dsphere Index
  - Calculate Distance using Spherical Geometry

Sample Problems

1. Consider the following example document:

```
{
  "_id": ObjectId("5360c0a0a655a60674680bbe"),
  "user": {
            "login": "ir0n",
            "description": "Made of steel"
            "date": ISODate("2014-04-
30T09:16:45.836Z"),
          }
}
```

and index creation command:

```
db.users.createIndex( { "user.login": 1,
"user.date": -1 }, "myIndex" )
```

When performing the following query:

```
db.users.find( { "user.login": /^ir.*/ },
               { "user":1, "_id":0 } ).sort( {
"user.date":1 } )
```

which of the following statements correctly describe how MongoDB will handle the query? Check all that apply.

a. As a covered query using "myIndex" because we are filtering out "_id" and only returning "user.login"
b. As an indexed query using "myIndex" because field "user.login" is indexed
c. As an optimized sort query (scanAndOrder = false) using "myIndex" because we are sorting on an indexed field
d. MongoDB will need to do a table/collection scan to find matching documents
e. None of the above

2. You perform the following query on the sayings collection, which has the index

```
{ quote : "text" }:
```

Assuming the documents below are in the collection, which ones will the following query return? Check all that apply.

```
db.sayings.find( { $text : { $search : "fact find"
} } )
```

a. { _id : 1, quote : “That’s a fact, Jack.” }
b. { _id : 2, quote : “Find out if that fact is correct.” }
c. { _id : 3, quote : “Nobody will ever catch me.” }

3. You have the following index on the toys collection:

```
{
  "manufacturer" : 1,
  "name" : 1,
  "date" : -1
}
```

Which of the following is able to use the index for the query? Check all that apply.

a. db.toys.find({ manufacturer : "Matteo", name : "Barbara", date : "2014-07-02" })
b. db.toys.find({ name : "Big Rig Truck", date : "2013-02-01", manufacturer : "Tanko" })
c. db.toys.find({ date : "2015-03-01", manufacturer : "Loggo", name : "Brick Set" })

4. Adding the index { a : 1 } can potentially decrease the speed of which of the following operations? Check all that apply.

a. db.collection.find({ a : 232 })
b. db.collection.update({ b : 456 }, { $inc : { a : 1 } })
c. db.collection.insert({ a : 341 })

5. You have the following indexes on the things collection:

```
[
  {
    "v" : 1,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "test.things"
  },
  {
    "v" : 1,
    "key" : {
```

```
        "a" : 1
      },
      "name" : "a_1",
      "ns" : "test.things"
    },
    {
      "v" : 1,
      "key" : {
        "c" : 1,
        "b" : 1,
        "a" : 1
      },
      "name" : "c_1_b_1_a_1",
      "ns" : "test.things"
    }
]
```

Which of the following queries will use the index to implicitly sort the query? Check all that apply.

a. db.things.find({ b : 1 }).sort({ c : 1, b : 1 })
b. db.things.find({ c : 1 }).sort({ a : 1, b : 1 })
c. db.things.find({ a : 1 }).sort({ b : 1, c : 1 })

Answers to Sample Problems

1. b
2. a, b
3. a, b, c
4. b, c
5. a

Exercises

On your own, do the following:

- Create:
  - A single-field index
  - A compound index
  - A text index
  - A single-field index on a string field
  - A 2dsphere index
  - A ttl index

    - A unique index
    - A background index
- For each of the above, notice how they are distinguished in db.collection.getIndexes() output.
- Insert some data for the indexes you created. Perform queries for each of the indexes.
- For a TTL index, write a script to insert 1 document per second. Look at when documents get deleted.
- Get explain plans for these queries. Use all 3 verbosities.
- Explain a covered query & verify that it's working (you'll see 0 documents viewed, though some index entries will be viewed).
- Perform some geospatial queries and look at the results.

Data Modeling (Developer Only)

On the certification exam, we will verify that you:

- Understand the overriding principles of data modeling
- Given two alternative data models, can determine which will be more efficient
- Know common patterns for schema design
- Know the benefits of special data types in MongoDB
- Understand the implications of the storage engine for data modeling

Introduction

For the exam, you should know:

- Fundamental data modeling considerations such as consideration for common data access patterns
- How we define the term "working set"
- Why considerations of the working set and working set size are important for efficient read and write operations
- Features used to model data in various ways, including:
    - GridFS
    - Read-only views
    - Collations
    - Special-case data types, such as `NumberDecimal`.

Resources:

- Docs:
    - Data Modeling Introduction
    - Working Set
- Video:
    - Schema Design Introduction

- White Paper:
  - Performance Best Practices for MongoDB
    - Note: This is a fairly comprehensive overview, touching on many of the sections of the exam.

Document Structure

For the exam, you should know:

- The difference between embedding documents and creating references
- What it means to denormalize data
- How each of these practices are used to model data in a collection

Resources:

- Docs:
  - Document Structure
  - Data Model Design
- Video:
  - Benefits of Embedding
  - When to Denormalize

Relational Features and MongoDB Patterns

For the exam, you should know:

- You don't actually need to know this, but it can help for people who come from the relational world to know the differences.
- Videos:
  - Living without Constraints
  - Living without Transactions
- Docs:
  - Database References
  - Atomicity and Transactions
  - Atomicity of Write Operations

One-to-One Relationships

For the exam, you should know:

- How to model one-to-one relationships
- Advantages and disadvantages of embedding vs. referencing for one-to-one relationships

Resources:

- Docs:
  - Model One-to-One Relationships with Embedded Documents
- Video:
  - One to One Relationships

One-to-Many Relationships

For the exam, you should know:

- Your options for modeling one-to-many relationships
- Advantages and disadvantages of each options
- Common patterns for modeling one-to-many relationships

Resources:

- Video:
  - One-to-Many Relationships
- Docs:
  - One-to-Many Relationships with Embedded Documents
  - One-to-Many Relationships with Document References
- Blog Posts:
  - Rules of Thumb
  - Rules of Thumb Part 2
  - Rules of Thumb Part 3

Many-to-Many Relationships

For the exam, you should know:

- How to model many-to-many relationships

Resources:

- Video:
  - Many to Many Relationships

Modeling Tree Structures

For the exam, you should know:

- Common tree structure modeling patterns
- Advantages & disadvantages of each for reading & writing

Resources:

- Video:

    - Trees
- Docs:
    - Model Tree Structures
    - Model Tree Structures with Parent References
    - Model Tree Structures with Child References
    - Model Tree Structures with an Array of Ancestors
    - Model Tree Structures with Materialized Paths
    - Model Tree Structures with Nested Sets

Schema Design Patterns

For the exam, you should know:

- How to model data for keyword search
- How to model monetary data

Resources:

- Docs:
    - Model Data to Support Keyword Search
    - Model Monetary Data

MongoDB BLOB Options

For the exam, you should know:

- That GridFS can store large binary files in a queryable format
- Approximately how large the documents in GridFS are
- How to store data in GridFS

Resources:

- Video:
    - GridFS
- Docs:
    - GridFS Storage
    - GridFS Reference

Views

For the exam, you should know:

- What a view does
- How a view uses indexes
- How to create a view
- What you can build a view on

Resources:

- Video:
  - Views - Introduction
  - Creating and Destroying Views
  - Views, Indexes, and Views of Views
- Docs:
  - Views

Collations and Case Insensitive Indexes

For the exam, you should know:

- How to use collations:
  - For a query in the `mongo` shell
  - For an index
  - For a collection
  - Which collations take precedence over others
- When queries can and cannot use an index with a collation specified
- How to create, use, and identify case insensitive indexes
  - By `strength`, in particular
- For a collation:
  - What the `locale` field specifies
  - Which values of `strength` use diacritics (2+)
  - The default `strength` of a collation (3) and what it uses

Resources:

- Docs:
  - Case Insensitive Indexes
  - Collations
- Video:
  - Collations - Introduction
  - Using collations
  - Case insensitive indexes
  - Collations and index selection
  - Collations on find and sort
  - Collations on Indexes

The NumberDecimal Type

For the exam, you should know:

- Why the `NumberDecimal` type exists

- The advantages of `NumberDecimal` over `NumberLong` and `double` (floating point), including:
  - Provides exact precision when working with base 10 floating-point numbers
  - Less vulnerable to systematic rounding errors than `double`
- How to insert/store decimal type numbers

Resources:

- Docs:
  - NumberDecimal
  - Model Monetary Data
- Video:
  - Decimal Type Demo

Exercise

- Think of an online marketplace, analogous to, perhaps, eBay or Amazon.
- Create a schema for handling the data associated with your producs.
- Optimize the schema for fast writes.
  - Then consider what changes you would make to optimize the schema for fast reads.

Aggregation (Developer Only)

For the exam, you should understand:

- The analogy between the aggregation pipeline and UNIX pipes
- Each aggregation stage operator and its semantics
- How documents enter the pipeline, are pass from one stage to another, and returned when the pipeline completes

Introduction

An aggregation pipeline allows you to combine data from multiple documents in a collection and perform basic grouping, arithmetic, and statistical operations on them.

The aggregation framework in MongoDB is based on the idea of UNIX pipelining. Each stage done job. A stage accepts a list of documents as input, manipulates the data in some way, and emits output documents, passing them to the next stage.

The Aggregation section of the exam is emphasized much more heavily in Developer exams than in DBA exams, but you should be familiar with the basic concepts and format of aggregation queries even for the DBA exam.

- Videos:

        - The Aggregation Pipeline
- Docs:
    - Aggregation Introduction
    - Aggregation Pipeline

Aggregation Expressions

For the exam, you will need to:

- Identify an aggregation expression
- Find what an expression will resolve to

Resources:

- Videos:
    - Aggregation Expressions
- Docs
    - Expressions Reference

Aggregation Stages

For the exam, you will need to know:

- All aggregation stage operators
- The semantics of each stage operator
- The output of each stage of operator
- How to assemble aggregation pipelines to perform specific tasks. Note: the best way to prepare for these types of questions is practice.

Resources:

- Videos:
    - Using $project
    - Using $match
    - Using $text
    - Using $sort
    - Using $limit and $skip
    - Using $unwind
    - Using $out
    - Introduction to $graphLookup
        - Example of $graphLookup
    - $count, $replaceRoot, $addFields
- Docs:
    - Aggregation Pipeline Quick Reference
    - $collStats

    - $project
    - $match
    - $redact
    - $limit
    - $skip
    - $unwind
    - $group
    - $sample
    - $sort
    - $geoNear
    - $lookup
    - $out
    - $indexStats
    - $facet
    - $bucket

Aggregation Operators

For the exam, you will need to know:

- All operators that are used by each stage
    - Note that $match operators are by and large those used for querying ($lt, $in, etc.)
    - Other stages may have unique operators
- How to use $project to change your document schema and rename keys
- Which operators to use to perform typical tasks with the aggregation pipeline

Resources:

- Videos:
    - Using $sum
    - Using $avg
    - Using $addToSet
    - Using $push
    - Using $max and $min
    - Double $group Stages
    - Revisiting $first and $last
- Docs:
    - $match
    - $group
    - $project
    - Aggregation Pipeline Operators

Aggregation Mechanics

For the exam, you will need to know:

- Memory limits imposed on the aggregation pipeline's data, and for use in sorting
- Optimizations that are applied to the aggregation pipeline
- When you are able to use indexes for aggregation

Resources:

- Video:
  - Aggregation Limits
- Docs:
  - Optimizing the Aggregation Pipeline
  - Aggregation Limits
  - Aggregation Pipeline Behavior

Aggregation Options

For the exam, you will need to know:

- The aggregation options available
- What the effect of these options will be

Resources:

- Docs:
  - db.collection.aggregate()
- Video:
  - Aggregation Options

Aggregation Examples

More than any other section, Aggregation is about practice. Here are some examples to give you some ideas and help you to get started.

For the exam, you will need to know:

- How to construct an aggregation query that will perform the operations you require
- How to use multiple $group and $unwind stages to accomplish more than you could without them

Resources:

- Videos:
  - Simple Aggregation Example
  - Simple Aggregation Expanded
  - $unwind Example
  - Compound Grouping

    - Double Unwind
- Docs:
    - Single Purpose Aggregation Operations
    - Aggregation with Zip Code Data
    - Aggregation with User Preference Data

Sample Aggregation Problems

1. Which of the following statements are true about the $match pipeline operator? Check all that apply.

    a. You should use it as early as possible in the pipeline.
    b. It can be used as many times as needed.
    c. It has a syntax similar to find() commands.

2. Suppose you have the following collection with only 2 documents:

```
> db.people.find()
{ "_id" : "apples", "traits" : [ "sweet" , "crispy"
 ] }
{ "_id" : "oranges", "traits" : [ "sweet" ,
"orange" , "juicy" ] }
```

    If you run an aggregation query and use { $unwind : "$traits" } as the first stage, how many documents will be passed to the next stage of the aggregation pipeline?

    - 1
    - 2
    - 3
    - 4
    - 5

Answers

1. a, b, c
2. 5

Aggregation Exercises for the Reader

- Find or build a data set. Perform an aggregation query to see how many documents contain a field within a given range.
- Count all of the elements in an array field, summed across all documents in the collection.

- Count only the elements in an array field above a certain value.
- Count the elements in an array field, but only for documents where another field has a certain value.

Replication

On the certification exam, we will attempt to verify that you:

- Understand the benefits of replication
- Understand tradeoffs between speed and durability
- Know the basics of how the oplog works, including concepts like idempotence and statement-based replication
- Know what happens when a node (primary or not) fails

Introduction

Replication is about availability and durability. It is, generally speaking, not for scaling. That would be the purpose of Sharding.

- Videos:
  - Replication Concepts
  - Replication Overview
  - Asynchronous Replication
- Docs:
  - Replication Introduction

Nodes

In the exam, you will be expected to know:

- The options to use when creating a node, such as:
  - Arbiter
  - Delayed
  - votes
  - priority

Resources:

- Docs:
  - Replica Set Members
  - Delayed Members
  - Hidden Members
  - Non-Voting Members
- Videos:

    - Arbiters
    - Hidden and Slave Delay

Initiating a Replica Set

For the exam, you should be familiar with:

- How to initiate a replica set (or initiate a single server and add members)
- The initial sync of a secondary
- Video:
    - Initiating a Replica Set
- Docs:
    - rs.initiate()
    - Initial Sync

Elections

For the exam, you will need to know:

- Everything that can trigger an election
- How priority, votes, optime, and unreachable servers in the set will affect the election
- Which server will win the election

Resources:

- Docs:
    - Elections
    - Non-voting members

Failover

For the exam, you will need to know:

- What triggers failover
- That failover triggers an election

Resources:

- Video:
    - Automatic Failover
    - Failover Example
- Docs:
    - Replica Set High Availability

Rollback

For the exam, you will need to know:

- What series of events will or won't trigger rollback
- What happens to data that gets rolled back

Resources:

- Video:
  - Recovery
- Docs:
  - Rollbacks

rs.status()

For the certification exam, you should be able to:

- Read the output of rs.status()
- Know what data is in rs.status()

Resources:

- Video:
  - Replica Set Status
- Docs:
  - rs.status() Example

Replica Set Reconfiguration

For the certification exam, you will need to be able to:

- Add and remove replica set members
- Reconfigure a replica set

Resources:

- Docs:
  - rs.add()
  - rs.remove()
  - rs.reconfig()
  - Reconfigure a Replica Set

Oplog

For the certification exam, you will need to:

- Understand the nature of MongoDB's statement-based replication
- Understand why the oplog operations must be idempotent

- Know what operations will be stored in the oplog
- Know that the oplog stores the _id of the document for writes
- Calculate how many oplog entries there may be for a particular write operation (one per document affected)

Resources:

- Video:
  - Statement Based vs Binary Replication
- Docs
  - Replica Set Oplog
  - Capped Collections
  - Idempotent (glossary)

Read Preference

For the exam, you should know:

- Which server (or servers) could be queried for every possible read concern (depending on the state of your servers, as well)
- When your read preference allows you to read stale data

Resources:

- Docs:
  - Read Preference Reference
- Videos:
  - Introduction to Read Preference
  - Read Preference Options

Write Concern

For the exam, you should know:

- The default write concern
- How to set write concern to majority or a fixed number of servers
- How many servers will have copies of the data for a given write concern
- How to ensure writes get to the journal before acknowledgment
- How to amortize write concern on insert using a bulk insert

Resources:

- Videos:
  - Write Concern Principles
  - Examining the 'w' parameter
  - Write Concern Use Cases and Patterns

- Docs:
  - Write Concern
  - Write Concern Reference

Sample Problems

1. Given a replica set with five data-bearing members, suppose the primary goes down with operations in its oplog that have been copied from the primary to only one secondary. Assuming no other problems occur, which of the following describes what is most likely to happen?
   a. The primary may roll back the operations once it recovers.
   b. The secondary with the most current oplog will be elected primary.
   c. Missing operations will need to be manually re-performed.
   d. The most current secondary will roll back the operations following the election.
   e. Reads will be stale until the primary comes back up.
2. Which of the following is true of the mechanics of replication in MongoDB? Check all that apply.
   a. Operations on the primary are recorded in a capped collection called the oplog.
   b. Members of a replica set may replicate data from any other data-bearing member of the set by default.
   c. Clients read from the nearest member of a replica set by default.
3. What read preference should your application use if you want to read from the primary under normal circumstances but allow reads from secondaries when a primary is unavailable?
   a. Nearest
   b. primary
   c. primaryPreferred
   d. secondaryPreferred
   e. Secondary
4. Using an arbiter allows one to easily ensure an odd number of voters in replica sets. Why is this important?
   a. To add greater redundancy
   b. For more efficient backup operations
   c. To help in disaster recovery
   d. To protect against network partitions
   e. To enable certain read preference settings

Answers to Sample Problems

1. b
2. a, b
3. c
4. d

Exercises for the Reader

- Set up a replica set on your laptop.
- Write data to the primary and see it propagate to the secondary.
- Connect to the mongod on your server with a driver, and look at how throughput changes as you increase your write concern.
- Compare standard vs. bulk insert speed for w = 1, 2, or 3
- Step down your primary
- Create rollback using a primary, a secondary, and an arbiter

Sharding

On the certification exam, we will verify that you:

- Understand horizontal scaling and how sharding provides this capability in MongoDB
- Know how to construct a good shard key, and what can go wrong with selecting a shard key
- Understand the balancer
- Know the role of the config servers and how they work.

Introduction

Sharding is about scaling. With sharding, you can distribute your data across several replica sets, each of which is a logical "node" in the sharded cluster.

Note that sharding and replication solve different problems. Replication is concerned with data durability and high availability.

Resources:

- Docs:
  - Sharding Introduction
  - Sharded Cluster Components
  - Shards
- Video:
  - Introduction to Sharding
  - Sharding and Data Distribution

The Shard Key

For the exam, you should know:

- That shard keys are immutable
- What makes a good shard key

- What makes a bad shard key
- How the shard key implements ranged-based sharding in MongoDB

Resources:

- Docs:
  - Shard Keys
  - Shard Key Indexes
  - Unique Keys

Chunks and the Balancer

For the exam, you should know:

- How to define a chunk by shard key range
- How to determine whether a check range includes a specific document
- When chunk splits occur automatically
- How the balancer uses chunks to keep the cluster balanced

Resources:

- Docs:
  - Chunk (glossary entry)
  - Chunk Splits in a Sharded Cluster
  - Sharded Collection Balancing
- Videos:
  - Chunks and Operations

Config Servers and Cluster Metadata

For the exam, you should know:

- What data config servers contain
- How to access data in config servers
- What happens when a config server is unavailable
- What types of servers constitute the config servers
- What happens when your config servers cannot elect a Primary

Resources:

- Docs:
  - Config Servers
  - Replica Set Config Servers
  - Config Server Availability
  - Sharded Cluster Metadata
- Videos:

    - Config DB
    - Cluster Setup Topology
    - Sharding Processes

Pre-Splitting Data (DBA Only)

For the exam, you should know:

- How to pre-split chunks
- Why you would want to pre-split chunks
- How to split chunks manually
- How to merge chunks manually

Resources:

- Docs:
    - Create Chunks
    - Split Chunks

Queries in a Sharded Cluster

For the exam, you should know:

- Performance implications for targeted vs. scatter-gather queries
- Given a query and description of a sharding configuration, Whether the query will be targeted or scatter gather
- How to read .explain() output to determine which shards were affected by a query
- How sorting and aggregation work in a sharded cluster
- What mongos nodes are and their role in a sharded cluster

Resources:

- Docs:
    - Broadcast Operations
    - Targeted Operations
    - Sharded Cluster Query Routing
    - Aggregation Pipeline and Sharded Collections
- Video:
    - Implications of Sharding

Choosing a Shard Key

For the exam, you should know:

- What makes a good shard key:
    - High cardinality

    - High selectivity
    - Non-monotonically increasing/decreasing values
    - What these mean

Resources:

- Docs:
    - Choose a Shard Key
    - Shard a Collection using a Hashed Shard Key
- Video:
    - Choosing a Shard Key
    - Shard Key Selection Example

Primary Shard

For the exam, you should know:

- What data the primary shard contains
- What read and write operations occur on the primary shard
- How aggregation queries use the primary shard

Resources:

- Docs:
    - Primary shard

Exercises

Try to perform the following on your own:

- Spin up a sharded cluster
- Add a shard
- Shard a collection in the cluster.
- Add data to a chunk, and watch as it:
    - Splits
    - Migrates elsewhere (if enough chunks are split)
- View the status of your cluster in the "config" database.
- Manually split a chunk.
- Drain a shard and remove it from the cluster.

Application and Server Administration (DBA Only)

For the certification exam, we will verify that you:

- Understand the mechanics of the MongoDB journal and server logs
- Understand MongoDB security
- Can identify the advantages and disadvantages of different cluster architectures
- Be able to evaluate options about basic server diagnostics, maintenance, backup, and disaster recovery.

Introduction

While the definitions are somewhat fluid, Application Administration deals with MongoDB's relationship to applications. The features we consider here include: the wire protocol, over-the-wire encryption, and security.

Server administration deals with architecting, maintaining, and debugging a deployment.

Journal

For the exam, you should know:

- The purpose of the journal
- That the journal is implemented as a binary write-ahead log
- Why the journal ensures consistent data in the event of server failure for MMAPv1
- Why the journal is unnecessary for WiredTiger
- The fundamentals of how the journal works, e.g., how often data is flushed to disk, for both MMAPv1 and WiredTiger

Resources:

- Videos:
  - Journal's impact on resident memory
- Docs:
  - Journaling Mechanics
  - Manage Journaling
- Blog:
  - How MongoDB's Journaling Works

Server Logs

For the exam, you should to know:

- What queries get captured in the server logs
- How to rotate log files
- Common events that get captured:
  - Creating/dropping databases

    - Connections
- Docs:
    - Log Messages
    - Rotate Log Files
    - Process Logging
- Videos:
    - Examining Log Files

The Profiler

For the exam, you should know:

- How to turn the profiler on and off or change its settings
- What the profiler captures
- Where that information is stored

Resources:

- Docs:
    - Database Profiling
- Videos:
    - Profiler Demo
    - Profiling Overview
    - Profiler Helper Function
    - Examining Profiler Operations
    - Filtering the Profiler by Timestamp

Monitoring and Performance Tuning

For the exam, you will need to know:

- What tools are available to monitor and tune MongoDB
- How to interpret their output for simple scenarios, such as:
    - Working set has outgrown RAM
    - Contention is limiting write speed (MMAPv1)
    - Disk I/O is saturated

Resources:

- Monitoring for MongoDB
- Analyzing MongoDB Performance

MongoDB Security

For the exam, you should know:

- How to define user roles and permissions in MongoDB Challenge and Response (MongoDB-CR)
- Which other security best practices are available

Resources:

- Docs:
  - Security Introduction
  - Authentication
  - Authorization
  - Hardening Network Infrastructure
  - Collection-Level Access Control
  - Auditing
  - Security Tutorials

Cluster Architecture

For the exam, you should know common deployment patterns for:

- Replica sets
- Sharded clusters

Resources:

- Replica Set Deployment Architectures
- Sharded Cluster Requirements
- Production Cluster Architecture
- Config Server Availability

Diagnostics and Debugging

For the exam, you should know:

- Basic commands to look at server, replica set, and sharded cluster status
- How to interpret those commands
- Solutions to simple problems that may arise, such as:
  - A server is down
  - A config server is down
  - A long-running query is grabbing too many resources
  - All queries are confined to one server in a sharded cluster

Resources:

- Docs:
  - db.currentOp()
  - db.serverStatus()

    - Diagnostics
    - rs.status()
    - sh.status()
    - db.killOp()
- Webinar:
    - Diagnostics and Debugging

Maintenance

For the exam, you should be able to:

- Rotate log files
- Create new users in MongoDB-CR
- Remove a shard from a sharded cluster

Resources:

- Docs:
    - Rotate Log Files
    - Upgrade to the Latest Version of MongoDB
- Videos:
    - Removing a Shard
    - Upgrading a Sharded Cluster

Backup and Recovery

For the exam, you will need to know:

- Backup options for individual servers and clusters
    - Filesystem snapshots
    - mongodump
- How to restore data from these backups

Resources:

- Docs:
    - Backup and Restore with Filesystem Snapshots
    - Restore a Replica Set from MongoDB Backups
    - Backup and Restore with MongoDB Tools
- Videos:
    - Overview of Backing Up
    - Mongodump
    - Filesystem Snapshot
    - Backing Up a Sharded Cluster
    - Backup Strategies

Exercises for the Reader

- Find long-running events:
  - Create a replica set or sharded cluster.
  - Create a database, and then drop it.
  - Put the server under a simulated load and then insert an array of large documents (~15 MB), preferably as a bulk insert.
  - Check the server logs and try to find the long-running events.
- Create a user admin
  - Run a mongod with the --auth flag.
  - Log in and create a User Admin
  - Use that login to create a user with Read permissions in one database and ReadWrite permissions in another.
  - Verify that you are able to login and read/write as appropriate, and are locked out of other databases.
- Backup some data
  - Use a working MongoDB deployment.
  - Backup your data with a filesystem snapshot.
  - Drop your database, and restore from the backup.
  - Repeat for mongodump/mongorestore.
- Long-running queries
  - Turn on the profiler set to level 2.
  - Perform some reads and some write queries.
  - Find out how long they took.
  - Don't forget to set the profiler back to level 0.
- Kill a long-running process
  - Turn on the profiler set to level 1.
  - Create some long-running queries.
  - Start a long-running process.
  - Use db.currentOp() and db.killOp() to find and kill the long running process.

Server Tools

For the certification exam, will will verify that you understand:

- How to export and import data using server tools
- How to monitor basic operations on the server using server tools
- How to backup and restore data and examine backed-up data using server tools (DBA only)
- Which tools to use to sniff network data, manipulate GridFS files, and analyze disk I/O (DBA only)

Introduction

Most of the information tested can be found by running the tools with the --help option.

For the Developer exam, you will need to know:

- mongoimport
- mongoexport
- mongostat
- mongotop

For the DBA exam, you will need to know:

- mongoimport
- mongoexport
- mongostat
- mongotop
- mongodump
- mongorestore
- mongosniff
- mongofiles
- bsondump
- mongoperf

Importing and Exporting Data

For the exam, you should know how to import/export data between MongoDB and:

- JSON files
- CSV files

Resources:

- Docs:
  - mongoimport docs
  - mongoexport docs
  - mongostat docs
  - mongotop docs
- Videos:
  - Importing from Reddit

Basic Server Monitoring

For the exam, you should know:

- How to use mongostat to monitor MongoDB for both the MMAPv1 and WiredTiger storage engines
- How to use mongotop to look at server activity
- What fields are of particular interest when diagnosing certain types of performance problems in both mongostat and mongotop

Resources:

- Docs:
  - mongostat
  - mongotop
- Videos:
  - mongostat video
  - mongotop video

Backing up and Restoring Data (DBA Only)

For the exam, you should know:

- How to use mongodump and mongorestore to save and restore data
- How to include your oplog in a mongodump or mongorestore

Resources:

- Docs:
  - mongodump
  - mongorestore

Advanced MongoDB Diagnostics (DBA Only)

For the exam, you should know how to use the following tools:

- mongosniff
- bsondump
- mongoperf

Resources:

- Docs:
  - mongosniff
  - bsondump
  - mongoperf

Manipulating BLOBs (DBA only)

For the exam, you will need to be able to use mongofiles to put data into GridFS.

Resources:

- mongofiles

Exercises for the Reader

- For each of the server tools listed above, run the tool with --help.
- Run mongostat.
  - Write a little code that bulk inserts 1,000 small documents per batch as fast as possible for at least a minute.
  - Run the code and look at the output of mongostat, and compare it with your expectations. Pay attention to each field.
- Run mongostat.
  - Then, write a little code that individually inserts one large document (100+ kb per insert) at a time as fast as possible for at least a minute.
  - Run it and look at the output of mongostat, comparing the output with your expectations.
  - Pay attention to each field. This time, try to use a different storage engine than you did last time.
- Run mongotop.
  - Using a script from either mongostat exercise, begin inserting data, and also run some queries in another database.
  - Look at the output of mongotop.
- For mongoperf, run a similar experiment as you did for mongotop.
- Import a JSON file into a MongoDB collection. You might find one at Reddit's technology subreddit, the Enron email data set, or find something else on the web.
  - Technology Subreddit
  - Enron email data set
  - You may find that the format isn't quite what you want. Transform it, either with a script or with the aggregation pipeline, using $out, into a more usable form.
- Find a CSV data set from US public data, and import it. If you don't like its initial format, you can transform it and put it in a new collection using the aggregation pipeline's $out function.
  - US public data
- Perform a mongodump.
  - Look at the output using bsondump.
  - Drop the database and then use mongorestore to get it back from your backup.
- Run a mongodump with the --oplog option and compare the mongorestore with the --oplog replay option and without.
- If you are running a Linux machine, run mongosniff and run some queries; find them using mongosniff.

Storage Engines

For the exam, we will verify that you know:

- Concurrency levels for MMAPv1 and WiredTiger
- The compression algorithms available for WiredTiger
- For MMAPv1, the causes and effects of document movement
- The effects of these features on the performance of MongoDB

Introduction

A database storage engine is the underlying software component that a database management system uses to create, read, update, and delete data from a database. In the case of MongoDB, the storage engine organizes BSON data in memory and on disk to support read and write operations. Pluggable storage engines were introduced with MongoDB 3.0. Prior to that, MongoDB's storage engine was what we now call MMAPv1.

- Docs:
  - MMAPv1
  - WiredTiger
- Videos:
  - Storage Engines Introduction

Summary of Major Differences between MMAPv1 and WiredTiger

- Locks/Concurrency
  - MMAPv1 uses collection level locking in MongoDB 3.0
  - WiredTiger supports document-level concurrency
- Journaling
  - Journaling is recommended for both MMAPv1 and WiredTiger
  - For MMAPv1, it ensures that writes are atomic
  - For WT, it ensures that writes make it to disk between checkpoints
- Data Compression
  - WiredTiger supports both the snappy and zlib compression algorithms
  - MMAPv1 does not support data compression
- Other Considerations
  - Due to the way MMAPv1 organizes data, if a BSON document outgrows its allotted space, it must be moved. Indexes that point to this document point to its file offset and will need to be updated. As a consequence, writes that force a document to move on disk come at a relatively high performance cost. Document movement also leads to fragmentation.
  - To minimize document movement, MMAPv1 uses a power-of-two size allocation strategy, allocating a "record space" to each document that is larger than the document. If the document outgrows its record space, the newly allocated space

for the document will be twice as large as the previous allocation. Power-of-two sizing also enables the storage engine to more easily reuse space vacated by a moved document, because all document allocations will be a size that is a power of two.

MMAPv1

For the exam, you should know:

- The mechanics of memory mapping in the MMAPv1 storage engine
- That MMAPv1 supports collection-level concurrency (locking)
- Why Power-of-two allocation is used in MMAPv1
  - Why it is usually a good idea
  - How to turn it off
- Docs:
  - Concurrency FAQ
  - Power of 2 Sized Allocations
  - No Padding
- Videos:
  - MMAPv1
  - MMAPv1 Documents and Data Files

WiredTiger

For the exam, you should know:

- Compression options in WiredTiger
- That WiredTiger supports document-level concurrency (locking)
  - Default settings
- How the WiredTiger cache works
- Docs:
  - Concurrency FAQ
  - Storage FAQ WiredTiger Section
  - Index Prefix Compression
  - Compression Options:
    - snappy
    - zlib
- Videos:
  - WiredTiger

Data Files

For the exam, you should know:

- Whether a dbpath directory is for MMAPv1 or WiredTiger based on the files present
- For MMAPv1, the maximum size of database files and when MMAPv1 will create new database files

Following is an example of an MMAPv1 data directory:

```
$ ls -la
total 1179664
-rw-------   1 will  staff    16M Sep 14 08:56 test.ns
-rw-------   1 will  staff    64M Sep 14 08:56 test.0
-rw-r--r--   1 will  staff    69B Sep 10 15:52
storage.bson
-rwxr-xr-x   1 will  staff     5B Sep 10 15:52
mongod.lock*
-rw-------   1 will  staff    16M Sep 16 00:38 local.ns
-rw-------   1 will  staff   256M Sep 16 00:38 local.1
-rw-------   1 will  staff    64M Sep 10 15:54 local.0
drwxr-xr-x   4 will  staff   136B Sep 14 08:55 journal/
-rw-------   1 will  staff    16M Sep 16 00:38 foo.ns
-rw-------   1 will  staff    64M Sep 16 00:38 foo.0
drwxr-xr-x   2 will  staff    68B Sep 16 00:34 _tmp/
drwxr-xr-x   4 will  staff   136B Sep 10 15:52 ../
drwxr-xr-x  15 will  staff   510B Sep 16 00:34 ./
```

and here is an example of WiredTiger data directory:

```
$ ls -la
total 360
-rw-r--r--   1 will  staff    95B Sep 16 15:43
storage.bson
-rw-r--r--   1 will  staff    16K Sep 16 15:43
sizeStorer.wt
-rwxr-xr-x   1 will  staff     6B Sep 16 15:43
mongod.lock*
drwxr-xr-x   5 will  staff   170B Sep 16 15:43 journal/
-rw-r--r--   1 will  staff    16K Sep 16 15:43 index-5-
5307542050812875631.wt
-rw-r--r--   1 will  staff    16K Sep 16 15:43 index-3-
5307542050812875631.wt
-rw-r--r--   1 will  staff    16K Sep 16 15:43 index-1-
5307542050812875631.wt
drwxr-xr-x   4 will  staff   136B Sep 16 15:43
```

```
diagnostic.data/
-rw-r--r--   1 will  staff   4.0K Sep 16 15:43
collection-6-5307542050812875631.wt
-rw-r--r--   1 will  staff    16K Sep 16 15:43
collection-4-5307542050812875631.wt
-rw-r--r--   1 will  staff    16K Sep 16 15:43
collection-2-5307542050812875631.wt
-rw-r--r--   1 will  staff    16K Sep 16 15:43
collection-0-5307542050812875631.wt
-rw-r--r--   1 will  staff    16K Sep 16 15:43
_mdb_catalog.wt
-rw-r--r--   1 will  staff   4.0K Sep 16 15:43
WiredTigerLAS.wt
-rw-r--r--   1 will  staff    24K Sep 16 15:43
WiredTiger.wt
-rw-r--r--   1 will  staff   907B Sep 16 15:43
WiredTiger.turtle
-rw-r--r--   1 will  staff    21B Sep 16 15:43
WiredTiger.lock
-rw-r--r--   1 will  staff    45B Sep 16 15:43
WiredTiger
drwxr-xr-x   4 will  staff   136B Sep 16 15:43 ../
drwxr-xr-x  20 will  staff   680B Sep 16 15:43 ./
```

Exercises for the Reader

- Use mongostat and mongotop to compare the throughput of inserts, updates, and reads for WiredTiger and MMAPv1. Do so for a variety of loads and different types of read an write operations.
- Perform many inserts for both WiredTiger and MMAPv1. Compare the size of the data files. Repeat with each compression option for WiredTiger.
- Set up a replica set with different storage engines for different nodes in that replica sets.