Taking an Exam

You are about to take a MongoDB Certification Exam. Our free study guide will be your road map to exam preparation. We give you a summary of the most pertinent information so you can pinpoint the areas where it would be most beneficial to focus your attention. With a good understanding of the subjects covered here, you should be well prepared to pass. Good luck!

Proctoring

MongoDB professional certification exams are delivered entirely online allowing you to conveniently take the exam in the comfort of your home or office while being monitored by an offsite proctor. The proctor will provide step-by-step instructions to ensure a seamless check-in process. After completing authentication, they will continuously monitor both you and your testing environment to ensure the security and integrity of the exam and help you avoid from incurring any unintentional violations.

Computer Specifications

To complete your exam, you need a Windows or Mac computer with a video camera (webcam), a microphone, and speakers. No Tablets, e.g., iPads and Google Chromebooks, and other operating systems, e.g., Linux, are not supported. 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. Webcam, speakers, and microphone must remain on throughout the exam. Headphones/earbuds are not permitted during the exam. To ensure a smooth examination experience it is extremely important that you have a stable connection. Please perform a **system readiness check** prior to scheduling your exam and once again closer to your exam appointment date to verify that your system is still.

Your Test-Taking Environment

You are required to be in a private room without interruption during the exam. Please ask others to refrain from coming into the room where you are taking your exam as it may result in an exam violation. It is not permitted to access reference materials during the exam. Please ensure your desk area is clear of any clutter including electronic devices and notepads. Your proctor will ask you to scan your desk and room area prior to the beginning of the exam and periodically throughout the exam if they believe it to be necessary. 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 is made up of 60 multiple choice and check-all-that-apply questions. All questions are weighted equally. You will find it of advantage to answer all questions, as there is no penalty for an incorrect answer. There is no advantage in leaving a question unanswered.

Translation Software

You are permitted to use translation software during your exam to translate questions. However, please note that MongoDB University does not support any translation software and cannot vouch for the suitability or accuracy of their translation. By using a translation software you are assuming all risks associated with it.

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 searching on the Internet via a search engine (e.g. Google, Bing, etc).

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
- Hexadecimal (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**
  - **What is a Document in MongoDB**
- 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:

- Videos:
  - BSON
  - BSON vs JSON
- Docs:
  - BSON specification

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.

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
  - What is NoSQL 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:

- Docs:
  - Flexible Schema in MongoDB

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 which data types are supported by MongoDB

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.

- Reference:
  - MongoDB CRUD Operations
  - MongoDB Back-to-Basics Presentation
  - MongoDB Query Language (MQL) Demo

Create

For the exam, you should be able to:

- Correctly 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

You should be familiar with the _id field and its special properties:

_id is a field required in every MongoDB document. The _id field must have a unique value for the collection it is used in. You can think of the _id field as the document's Primary Key. If you create a new document without an _id field, MongoDB automatically creates the field and assigns a unique BSON ObjectId. If the document contains an _id field, the _id value must be unique within the collection to avoid duplicate key error.

- Video:
  - Inserting New Documents - ObjectId
  - Compound _id Field
- Docs:
  - ObjectId
  - The _id Field

Document creation can occur through the following commands:

- db.collection.insert()
  - Video:

        - Inserting New Documents - insert() and errors
        - Inserting New Documents - insert() order
    - Docs:
        - Insert Documents
        - db.collection.insertOne()
        - db.collection.insert()
        - db.collection.insertMany()
- db.collection.save() updates an existing document or inserts a new document, depending on its document parameter.
    - Docs:
        - db.collection.save()
- db.collection.update() will insert new documents in the collection if upsert is set to true and no document matches the query criteria.
    - Video:
        - Upsert lesson video
        - Upsert - Update or Insert?
    - Docs:
        - db.collection.update() (will insert only if upsert: true is set)
        - Insert a New Document if No Match Exists (Upsert)
- db.collection.findAndModify() can result in an insert if no matching document exists for the update operation.
    - Docs:
        - db.collection.findAndModify()
        - Upsert
        - Return New Document

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.

Finally, you can bulk insert by passing an array to db.collection.insert() or db.collection.bulkWrite(). You should know the difference between ordered and unordered bulk inserts. You should also know the difference between when to use a bulk operation versus when to use insertMany().

Docs:

- Insert Multiple Documents
- Bulk Write Operations
- db.collection.bulkWrite()
- Bulk Operation Methods

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

Starting in MongoDB 4.4, as part of making find and findAndModify projection consistent with aggregation's $project stage,

- The find and findAndModify projection can accept aggregation expressions and aggregation syntax, including the use of literals and aggregation variables. With the use of aggregation expressions and syntax, you can project new fields or project existing fields with new values.
- The find and findAndModify projection can specify embedded fields using the nested form; e.g. { field: { nestedfield: 1 } } as well as dot notation. In earlier versions, you can only use the dot notation.

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

db.collection.find()

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

Keep in mind, though findOne() method returns a single document, a find() query will return a cursor:

- Videos:
  - Cursor lesson
- Docs:

  Iterate a cursor in mongo Shell

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

- db.collection.count()
  - Videos:
    - count() lesson
  - Docs:
    - count()
- 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()
  - **cursor.count()**
- **sort() and limit()**
- cursor.sort()
  - **Sort documentation**
- cursor.skip()
  - **Skip documentation**
- cursor.limit()
  - **Limit documentation**
- cursor.next()
  - **cursor.next() documentation**
- cursor.allowDiskUse()- New in version 4.4
  - **cursor.allowDiskUse() documentation**
- **More about Cursors**

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

- Videos:
  - **Projection lesson**
  - **Array Operators and Projection**
- Docs:
  - **Project Fields to Return From Query**
  - **Projection in db.collection.find()**

Update

For the exam, you should be able to:

- Correctly 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 modify existing 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**

- db.collection.findAndModify()
  - findAndModify() documentation
- db.collection.findOneAndUpdate()
  - findOneAndUpdate() documentation
- db.collection.update()
  - This will update unless upsert: true is specified and the query matches no documents.
  - update() documentation
  - Update Documents
  - With no operators specified in the update parameter, there will be a wholesale update
    - Updating Documents
    - Wholesale Update Lesson Video
  - Operators
    - Update Operators
    - Update Operators Documentation
    - $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 One
        - db.collection.updateOne()
    - Update Many
        - updateMany()
        - db.collection.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
- Alternative approaches to manage deletion of data
- Video:
    - Deleting Documents and Collection
- Docs:
    - Dropping a Collection: - Starting in MongoDB 4.4, the db.collection.drop() method and drop command abort any in-progress index builds on the target collection before dropping the collection. Prior to MongoDB 4.4, attempting to drop a collection with in-progress index builds results in an error, and the collection is not dropped. - db.collection.drop() - drop command
    - db.collection.remove()
    - db.collection.deleteOne()
    - db.collection.deleteMany()
    - db.collection.findOneAndDelete()
    - Expire Data from Collections by Setting TTL

Indexes

In the certification exam, we will verify that you:

- Understand the types of indexes available in MongoDB
    - Single Field Indexes
    - Compound Indexes
    - Multikey Indexes
    - Geospatial Indexes
    - Text Indexes
    - Hashed Indexes

    - Wildcard Indexes
- Know the index properties:
    - TTL
    - Sparse
    - Unique
    - Partial
    - Hidden
- Know how to improve the efficiency of a query using indexes
- Understand the write performance costs of indexes

Introduction

The following resources provide a basic introduction to indexes.

- Videos:
    - **Introduction to Indexes**
- Docs:
    - **Indexes**
    - **Index Names**

Default _id index

MongoDB creates a unique index on the _id field during the creation of a collection. The _id index prevents clients from inserting two documents with the same value for the _id field. You cannot drop this index on the _id field.

-Docs:

- **Default _id index**

Creating and Dropping Indexes

In the exam, we will ensure that you:

- Know how to create an index
- Know how to drop an index
- Videos:
    - **createIndex(), getIndexes(), dropIndex()**
    - **Creating Indexes**
- Docs:
    - **db.collection.createIndex()**
    - **db.collection.getIndexes()**
    - **db.collection.dropIndex()**
    - **db.collection.dropIndexes()**

    - db.collection.drop() - Removes a collection or view from the database. The method also removes any indexes associated with the dropped collection.

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

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:

- Videos:
    - Single Field Index Part 1
    - Single Filed Index Part 2
- Docs:
    - Single Field Indexes
    - Create an Index on Embedded Field
    - Create an Index on Embedded Document
    - Sort with Single Field Index

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
- Videos:
    - Compound Indexes Part 1

    - Compound Indexes Part 2
- Docs:
    - Compound Indexes
    - Prefixes
    - Sort on Multiple Fields

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.

- Videos:
    - Multikey Indexes
    - Dot Notation and Multikey
- Docs:
    - Multikey Indexes
    - Multikey Index Bounds

Geospatial Indexes

For the exam, you will need to know:

- How to create 2d and 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

To support efficient queries of geospatial coordinate data, MongoDB provides two special indexes: 2d indexes that use planar geometry when returning results and 2dsphere indexes that use spherical geometry to return results.

Resources:

- Video:
    - Geospatial Indexes
- Docs:
    - Geospatial Indexes and Queries

- GeoJSON Object
- 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

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 Examples

Hashed Indexes

For the exam, you will need to know:

- How to create a hashed index
- How to create a compound hashed index

Resources:

- Docs:
  - Hashed Indexes
  - Create a Compound Hashed Index

Wildcard Indexes

For the exam, you should know:

- How to create a Wildcard index

Resources:

- Video:
  - Wildcard Indexes: part 1
  - Wildcard Indexes: part 2
  - Wildcard Index Use Cases
- Docs:
  - Wildcard Indexes
  - Wildcard Indexes Query/Sort Support

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
- Videos:
  - When you can sort with Indexes
- Docs:
  - Use Indexes to Sort Query Results

The .explain() Method

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())

Reference:

- Tips and Tricks for Query Performance: Let us .explain() them

Resources:

- Videos:
  - Understanding Explain Part 1
  - Understanding Explain Part 2
- 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 apply
- When the query optimizer is re-run
- Videos:
  - Choosing an Index
  - Efficiency of Index Use
  - Query plans
- 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:

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

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

Resources:

- Video:
  - Insert Performance
- Docs:
  - Write Operation Performance

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:

- Unique 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:
  - TTL Indexes

Hidden Indexes

For the exam, you should know:

- How to create a hidden index
- How to hide an existing index
- How to unhide an existing index
- When and why you would use a hidden index

Resources:

- Docs:
  - Hidden Indexes

Partial Indexes

Starting in MongoDB 3.2, MongoDB provides the option to create partial indexes. Partial indexes offer a superset of the functionality of sparse indexes. If you are using MongoDB 3.2 or later, partial indexes should be preferred over sparse indexes.

For the exam, you should know:

- How to create a partial index
- Behavior of Partial Index
- Comparison with Sparse Index
- Restrictions applied on Partial Index

Resources:

- Docs:
  - Partial Indexes
  - Sparse Indexes

Hybrid Index Build

For the exam, you will need to know:

- How indexes are built on populated collections
- How index build impacts the Database Performance

Resources:

- Video:
  - Index Builds
  - Index Key Length Limits
- Docs:
  - Index Builds on Populated Collections
  - Build Indexes on Replica Sets
  - Build Indexes on Sharded Clusters

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

Data Modeling (Developer Only)

In the certification exam, we will verify that you:

- Understand the document model
- Given two alternative data models, you can determine which will be more efficient
- Know common patterns for schema design
- Know the benefits of special data types in MongoDB
- Understand the difference between embedding and linking pieces of information

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:

- Video:
  - **Data Modelling in MongoDB**
  - **Schema Design Patterns**
  - **Data Modeling with MongoDB talk**
- Docs:
  - **Data Modeling Introduction**
  - **Working Set**
- 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:

- Video:
  - **The Document Model**
  - **Benefits of Embedding**
  - **When to Denormalize**
- Docs:
  - **Document Structure**

    - Data Model Design

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
- 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 - Part 1
    - One to One Relationships - Part 2

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 - Part 1
    - One-to-Many Relationships - Part 2
- Docs:
    - One-to-Many Relationships with Embedded Documents

    - One-to-Many Relationships with Document References
- Blog Posts:
    - Rules of Thumb Part 1
    - 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 - Part 1
    - Many to Many Relationships - Part 2

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:

- Videos:

    - Advanced Schema Design Patterns talk
    - Schema Design Anti-patterns
- Docs:
    - Model Data to Support Keyword Search
    - Model Monetary Data
- Blogs:
    - Schema Design Anti-Pattern: Massive arrays

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

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
    - Efficiency of Querying Read Only 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
    - Collation Locales and Default Parameters
- 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

MongoDB Blogs on Data Modelling:

- The Polymorphic Pattern
- The Attribute Pattern
- The Bucket Pattern
- The Outlier Pattern
- The Computed Pattern
- The Subset Pattern
- The Extended Reference Pattern
- The Approximation Pattern
- The Tree Pattern
- The Pre-allocation Pattern
- The Document Versioning Pattern
- The Schema Versioning Pattern
- Schema Design Anti-Pattern: Massive arrays

MongoDB Schema Design Best Practices Series:

- Introduction to Data Modelling
- Modelling Relationships
- Building with Patterns - Part 1 Polymorphic & Attribute Patterns
- Building with Patterns Part 2 Bucket & Outlier Patterns
- Building with Patterns Part 3: Computed & Subset Patterns
- Building with Patterns Part 4: Extended Reference and Approximation
- Building with Patterns Part 5: Tree and Preallocation Patterns
- Series_Building with Patterns Part 6: Document Versioning & Schema Versioning Patterns

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 passed from one stage to another, and are returned when the pipeline completes

Introduction

An aggregation pipeline allows you to combine data from multiple documents in a collection and perform expressive filtering, powerful data transformation, statistical utilities and data analysis.

The aggregation framework in MongoDB is based on the idea of UNIX pipelining. 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 Framework
  - The Concept of Pipelines
  - Aggregation Structure and Syntax
- Docs:
  - Aggregation Introduction
  - Aggregation Pipeline

Aggregation Expressions

For the exam, you will need to:

- Identify an aggregation expression
- Determine 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 stages
- The semantics of each stage
- The output of each stage
- How to assemble aggregation pipeline stages to perform specific tasks.

Resources:

- Videos:
  - Using $match
  - Using $project
  - Using $addFields
  - Using $text
  - Using $sort

    - Using $limit and $skip
    - Using $redact
    - Using $group
    - Using $unwind
    - Using $lookup
    - Introduction to $graphLookup
        - Example of $graphLookup
    - $count, $replaceRoot, $addFields
    - Using $merge
    - Using $facet
- Docs:
    - Aggregation Pipeline Quick Reference
    - $match
    - $project
    - $addFields
    - $text
    - $sort
    - $limit
    - $skip
    - $redact
    - $group
    - $unwind
    - $out
    - $sample
    - $geoNear
    - $lookup
    - $graphLookup
    - $count
    - $replaceRoot
    - $merge
    - $facet
    - $bucket
    - $indexStats
    - $collStats

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
- 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 $last
- Docs:
  - Aggregation Pipeline Operators
  - Accumulators($group)
  - Accumulators(in Other Stages)

Aggregation Mechanics

For the exam, you will need to know:

- Memory limits imposed on an aggregation pipeline
- Optimizations that are applied to the aggregation pipeline
- When you need to use indexes for aggregation

Resources:

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

Aggregation Options

For the exam, you will need to know:

- The aggregation options available
- The effect of these aggregation options

Resources:

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

Aggregation Examples

To help you gain familiarity with aggregation, here are some examples to give you an idea about how aggregation works.

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

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

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:
  - What is Replication
  - MongoDB Replica Sets
  - Asynchronous Replication
- Docs:
  - Replication Introduction

- Replica set data synchronization

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
  - Replica Set Configuration
  - Replica Set Arbiter

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 replica set members)
- The initial sync of a secondary node in a replica set

Resources:

- Video:
  - Setting Up a Replica Set
- Docs:
  - rs.initiate()
  - Initial Sync
  - Replica Set Deployment Tutorials

Elections

For the exam, you will need to know:

- Events that can trigger an election
- How priority, votes, optime, and unreachable servers in the set will affect the outcome of the election

- Which node will win the election

Resources:

- Video:
  - Failover and Elections
- Docs:
  - Elections

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
  - Rollback time limit
  - Rollback directory

rs.status()

For the certification exam, you should be able to:

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

Resources:

- Video:
  - Replica Set Status
- Docs:
  - replSetGetStatus

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
  - Replica Set Reconfiguration Changes in 4.4
- Videos:
  - Reconfiguring 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 are 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
  - Minimum Oplog Retention Period
  - Idempotent (glossary)
  - Streaming Replication

Read Preference

For the exam, you should know:

- Which node (or nodes) could be queried for every possible read concern (depending on the state of your nodes, 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 the write concern to a majority or to a fixed number of nodes
- How many nodes will have copies of the data for a given write concern
- How to ensure writes get to the journal before the acknowledgment

Resources:

- Videos:
  - Write Concern Principles
  - Write Concern : Part 1
  - Write Concern : Part 2
  - Examining the 'w' parameter
  - Write Concern Use Cases and Patterns
- Docs:
  - Write Concern
  - Write Concern Reference
  - Default Write Concern
  - Write Concern for Replica Sets
  - Global Default Write Concern
  - Replica Set Write Acknowledgement in 4.4

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 role of the load 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 and Sharding is concerned with horizontal scaling of read and write workloads.

Resources:

- Docs:
  - Sharding Introduction
  - Sharded Cluster Components
  - Shards
  - Deploy a Sharded Cluster
- Video:
  - Introduction to Sharding
  - Sharding Architecture

The Shard Key

For the exam, you should know:

- Starting in MongoDB 4.2, shard key field values are mutable and they can be changed unless the shard key field is the immutable _id field
- 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
  - Missing Shard Keys
  - Shard Key Indexes
  - What changed in MongoDB version 4.4
- Video:

- Shard 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 chunk 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)
  - Manage Sharded Cluster Balancer
  - Chunk Splits in a Sharded Cluster
  - Sharded Collection Balancing
  - Migrate chunks in a sharded cluster
  - Modify Chunk Size in a Sharded Cluster
  - What changed in MongoDB Version 4.4 (jumbo-chunk-migration)
- Videos:
  - Chunks
  - Balancing

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:
  - Cluster Setup Topology
  - Setting up a Sharded Cluster
  - Config DB

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
  - Merge Chunk

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
  - Explain Results
  - Sharded Cluster Query Routing
  - Aggregation Pipeline and Sharded Collections
  - Hedged Reads
- Video:
  - Queries in a Sharded Cluster
  - Targeted vs. Scatter gather query

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**
  - **Ranged Sharding**
- Video:
  - **Choosing a Shard Key**
  - **Picking a Good Shard Key**

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**

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
- Understand Monitoring and Performance Tuning
- 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
- How the journal ensures durability in the event of a crash.
- The fundamentals of how the journal works, e.g., how often data is flushed to disk, for WiredTiger

Resources:

- Docs:
  - **Journaling Mechanics**
  - **Manage Journaling**
- Blog:
  - **How MongoDB's Journaling Works**
  - **MongoDB Storage Engine Journaling**

Compression

For the exam, you should know:

- How MongoDB compresses the data
- Available compression libraries
- Advantages of compression

WiredTiger Compression:

- With WiredTiger, MongoDB supports compressionfor all collections and indexes. Compression minimizes storage use at the expense of additional CPU. - **WiredTiger Storage Engine** - **Compression**

snappy:

- A compression/decompression library designed to balance efficient computation requirements with reasonable compression rates. snappy is the default compression library for MongoDB's use of WiredTiger. - **snappy** - **WiredTiger Compression documentation**

prefix compression:

- Reduces memory and disk consumption by storing any identical index key prefixes only once, per page of memory. - **Compression**

zlib:

- A data compression library that provides higher compression rates at the cost of more CPU, compared to MongoDB's use of snappy. You can configure wiredTiger to use zlib as its compression library. - **zlib**

zstd:

- New in version 4.2
- A data compression library that provides higher compression rates and lower CPU usage when compared to zlib.

Server Logs

For the exam, you should know:

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

Resources:

- Videos:
  - **Introducing Server Logs**
- Docs:
  - **Log Messages**
  - **Rotate Log Files**
  - **Process Logging**

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:

- Videos:

    - Introducing the Profiler
    - Profiler Demo
    - Profiling Overview
    - Examining Profiler Operations
    - Filtering the Profiler by Timestamp
- Docs:
    - Database Profiler
    - Database Profiler Output
- Blog:
    - Performance Best Practices: Query Patterns and Profiling

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
    - Disk I/O is saturated

Resources:

- Docs:
    - Monitoring for MongoDB
    - Analyzing MongoDB Performance
- Blog:
    - Performance Best Practices: MongoDB Data Modeling and Memory Sizing
    - Performance Best Practices: Hardware and OS Configuration

MongoDB Security

For the exam, you should know:

- How to define user roles and permissions in MongoDB
- Which other security best practices are available

Reference:

- Security Best Practices Series: Authn Authz
- Security Best Practices Series: Auditing
- Security Best Practicees Series: Field Level Encryption

Resources:

- Videos:
    - Authentication Mechanisms

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

Cluster Architecture

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

- Replica sets
- Sharded clusters

Resources:

- Docs:
    - Replica Set Deployment Architectures
    - Sharded Cluster Requirements
    - Production Cluster Architecture
    - Config Server Availability
- Blog:
    - Performance Best Practices: Sharding
    - MongoDB Data Platform Architecture Guide

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:

- Videos:
    - Introduction to Diagnostics and Debugging
    - currentOp and killOp
    - Introducing Server Status - 1

    - Introducing Server Status - 2
- Docs:
    - db.currentOp()
    - db.serverStatus()
    - Diagnostics FAQ
    - rs.status()
    - sh.status()
    - db.killOp()

Maintenance

For the exam, you should be able to:

- Rotate log files
- Remove a shard from a sharded cluster

Resources:

- Docs:
    - Run-time Database Configuration
    - Upgrade to the Latest Revision of MongoDB
    - Manage mongod Processes
    - Terminate Running Operations
    - Rotate Log Files

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
    - Backup and Restore with MongoDB Tools
    - Restore a Replica Set from MongoDB Backups
    - Backup and Restore Sharded Clusters
    - Recover a standalone after an Unxpected Shutdown
- Webinar:
    - Introduction to Ops Manager
    - MongoDB Ops Manager Datasheet

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 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**
- **mongofiles**
- **bsondump**

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**
  - **mongoexport**

Basic Server Monitoring

For the exam, you should know:

- How to use mongostat to monitor MongoDB
- 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:

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

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

Note:

> mongodump and mongorestore cannot be part of a backup strategy for 4.2+ sharded clusters that have sharded transactions in progress. - Reference

Advanced MongoDB Diagnostics (DBA Only)

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

- bsondump
- jq tool : In MongoDB 4.4, the logs are changed to structured logs and you need to know how to use the jq tool to parse the log

Manipulating BLOBs (DBA only)

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

Resources:

- mongofiles

Storage Engines

For the exam, we will verify that you know:

- Concurrency levels for WiredTiger
- The compression algorithms available for WiredTiger
- The effects of these features on the performance of MongoDB

Introduction

The storage engine is the component of the database that is responsible for managing how data is stored, both in memory and on disk. MongoDB supports multiple storage engines, as different engines perform better for specific workloads. Choosing the appropriate storage engine for your use case can significantly impact the performance of your applications. Pluggable storage engines were introduced with MongoDB 3.0.

- Docs:
  - WiredTiger
  - In-Memory Storage Engine
  - Storage Engines Documentation

WiredTiger

WiredTiger is the default storage engine starting in MongoDB 3.2. It is well-suited for most workloads and is recommended for new deployments. WiredTiger provides a document-level concurrency model, checkpointing, and compression, among other features.

Features of WiredTiger

- Locks/Concurrency
  - WiredTiger supports document-level concurrency
- Journaling
  - Journaling is recommended for WiredTiger
  - Starting in MongoDB 4.0, you cannot specify --nojournal option or storage.journal.enabled: false for replica set members that use the WiredTiger storage engine.
  - For WT, it ensures that writes make it to disk between checkpoints

    - If a log records less than or equal to 128 bytes (the minimum log record size for WiredTiger), WiredTiger does not compress that record.
- Data Compression
    - By default, WiredTiger uses block compression with the **snappy** compression library for all collections and prefix compression for all indexes.
    - For collections, the following block compression libraries are also available: - **zlib** - **zstd** (Available starting in MongoDB 4.2)

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**
    - **Manage Journaling**
    - **Journaling with WiredTiger Storage Engine**
- Videos:
    - **WiredTiger**

Data Files

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 ./
```

In-Memory Storage Engine

Starting in MongoDB Enterprise version 3.2.6, the in-memory storage engine is part of general availability (GA) in the 64-bit builds. Other than some metadata and diagnostic data, the in-memory storage engine does not maintain any on-disk data, including configuration data, indexes, user credentials, etc. By avoiding disk I/O, the in-memory storage engine allows for more predictable latency of database operations.

- Journaling
  - Starting in version 4.2 (and 4.0.13 and 3.6.14), if a replica set member uses the in-memory storage engine (voting or non-voting) but the replica set has writeConcernMajorityJournalDefault set to true, the replica set member logs a startup warning.
- Concurrency
- Durability
- Transactions
  - You cannot run transactions on a sharded cluster that has a shard with writeConcernMajorityJournalDefault set to false, such as a shard with a voting member that uses the in-memory storage engine.

- Docs:
  - In-Memory Storage Engine
  - Journaling and the In-Memory Storage Engine