AWD 1111

MongoDB Getting Started Guide

Database Driven Web Development

1. What is MongoDB?

MongoDB is a document-oriented NoSQL database that stores data in flexible, JSON-like documents. Unlike traditional relational databases like SQL Server that store data in tables with rows and columns, MongoDB stores data as documents in collections.

1.1 Why MongoDB?

- Flexible schema: Documents in the same collection can have different fields
- Natural data representation: Data is stored as JSON-like documents, matching how data is used in applications
- Scalability: Built for horizontal scaling across multiple servers
- Developer-friendly: Works naturally with JavaScript and modern web frameworks
- Part of the MERN stack: MongoDB, Express, React, Node.js

1.2 SQL Server vs MongoDB Terminology

Since you have experience with SQL Server, here is how the terminology maps:

SQL Server	MongoDB	Description
Database	Database	Container for collections/tables
Table	Collection	Group of related data
Row	Document	Single data record
Column	Field	Individual data element
Primary Key	_id	Unique identifier (ObjectId)
JOIN	Embedding / $lookup	Relating data across collections
Index	Index	Performance optimization
Schema	Flexible	Structure not enforced by default

1.3 Document Structure

MongoDB stores data as BSON (Binary JSON) documents. Here is an example document from a concerts collection:

```
{
"_id": ObjectId("5f8d04b3b54764421b7156c1"),
"tourName": "The Eras Tour",
"artist": "Taylor Swift",
"genres": ["Pop", "Country"],
"releaseYear": 2023,
"ticketPrice": 250,
"attendance": 70000,
"venue_ids": [
ObjectId("5f8d04b3b54764421b7156d2"),
ObjectId("5f8d04b3b54764421b7156d3"),
ObjectId("5f8d04b3b54764421b7156d4")
],
"setList": ["Cruel Summer", "Love Story", "Bad Blood"]
}
```

Key Differences from SQL
Notice how arrays (genres, venue_ids, setList) are stored directly in the document. In SQL Server, these would require separate tables and JOIN operations.

1.4 Our Demo Database: concert_demo

Throughout this course, we will use a custom concerts database for demonstrations. This database has four collections:

Collection	Description	Key Fields
concerts	Concert tour information	tourName, artist, genres, venue_ids
venues	Venue locations	name, city, state, capacity, location
reviews	Fan reviews of concerts	fan_name, concert_id, text, date
fans	Fan membership info	name, email, membershipLevel, state

2. Setting Up MongoDB Atlas

MongoDB Atlas is a fully managed cloud database service. We will use the free tier (M0) for this course.

2.1 Create an Atlas Account

1. Go to https://www.mongodb.com/cloud/atlas/register
2. Sign up with your email address (or use Google sign-in)
3. Verify your email address
4. Complete the welcome questionnaire (you can skip or answer casually)

2.2 Create a Free Cluster

1. After logging in, click "Build a Database" or "Create"
2. Select the FREE tier (M0 Sandbox) - this is important!
3. Choose a cloud provider (any is fine - AWS is common)
4. Select a region close to you (e.g., us-east-1 for St. Louis)
5. Keep the default cluster name or change it
6. Click "Create Deployment" - this takes 1-3 minutes

> **Important!**
> Make sure you select the FREE M0 tier. The paid tiers will charge your credit card!

2.3 Configure Security

Atlas requires two security configurations before you can connect:

A. Database User

1. Go to Database Access in the left sidebar
2. Click "Add New Database User"
3. Choose "Password" authentication
4. Create a username and password (save these somewhere!)
5. Set privileges to "Read and write to any database"
6. Click "Add User"

Password Tip
Avoid special characters like @, /, or : in your password as they can cause connection string issues. Use letters and numbers only.

B. Network Access (IP Whitelist)

1. Go to Network Access in the left sidebar
2. Click "Add IP Address"
3. "Add Current IP Address" to add your current location
4. Optionally: Click "Allow Access from Anywhere" (0.0.0.0/0) for easier access
5. Click "Confirm"

Network Access Warning
If you connect from school and home, you need both IP addresses whitelisted. Using "Allow Access from Anywhere" is less secure but more convenient for learning.

2.4 Load Sample Data

MongoDB provides sample datasets for learning and testing.

1. Click on "Database" in the left sidebar
2. Click the "..." (three dots) button next to your cluster name
3. Select "Load Sample Dataset"
4. Confirm by clicking "Load Sample Dataset"
5. Wait for the loading to complete (this takes a few minutes)

After loading, you will have several sample databases:

Database	Description
sample_mflix	Movies, comments, theaters, users
sample_airbnb	Airbnb listings and reviews
sample_restaurants	Restaurant data
sample_supplies	Sales data
sample_weatherdata	Weather measurements

2.5 Get Your Connection String

You will need the connection string to connect Compass and the MongoDB Shell.

1. Click "Database" in the left sidebar
2. Click the "Connect" button on your cluster
3. You will see options for different connection methods
4. For Compass: Select "Compass" and copy the connection string
5. For Shell: Select "Shell" and copy the connection string

A connection string looks like this:

```
mongodb+srv://username:password@cluster0.abc123.mongodb.net/
```

Replace Credentials
Replace <username> and <password> with your actual database user credentials you created earlier.

3. MongoDB Compass

MongoDB Compass is a free GUI tool for exploring and managing MongoDB databases. It is excellent for learning because you can see your data visually.

3.1 Download and Install

1. Go to https://www.mongodb.com/try/download/compass
2. Download the version for your operating system
3. Run the installer and follow the prompts
4. Launch Compass after installation

3.2 Connect to Atlas

1. Copy your connection string from Atlas (see Section 2.5)
2. Paste the connection string in Compass
3. Make sure your password is included (replace <password>)
4. Click "Connect"

3.3 Compass Interface Overview

Once connected, you will see:

Area	Purpose
Left Sidebar	Lists all databases and collections
Database View	Shows collections within a selected database
Collection View	Shows documents within a selected collection
Query Bar	Enter filters to search documents
Documents Tab	View and edit individual documents
Schema Tab	Analyze the structure of your documents
Indexes Tab	View and create indexes

3.4 Exploring the concert_demo Database

Let us explore the concert_demo database to understand document structure:

1. Click on "concert_demo" in the left sidebar

2. You will see the collections: concerts, venues, reviews, fans
3. Click on "concerts" to see the concert documents
4. Click on a document to expand and see all its fields

Understanding the Concerts Collection

Expand a concert document and notice these field types:

- _id: The unique identifier (ObjectId)
- tourName, artist: String fields (basic text)
- genres: An ARRAY of genre strings ["Pop", "Country"]
- releaseYear, ticketPrice, attendance: Number fields
- venue_ids: An ARRAY of ObjectIds referencing the venues collection
- setList: An ARRAY of song titles

Understanding the Venues Collection

The venues collection demonstrates nested objects:

```
{
"_id": ObjectId("5f8d04b3b54764421b7156d1"),
"name": "The Pageant",
"city": "St. Louis",
"state": "MO",
"capacity": 2000,
"location": {
"type": "Point",
"coordinates": [-90.2931, 38.6554]
}
}
```

The location field is a NESTED OBJECT containing GeoJSON data for geographic queries.

Try the Query Bar in Compass

In the query bar at the top of the concerts collection, try these filters:

```
// Find concerts from 2024
{ releaseYear: 2024 }

// Find Rock concerts
{ genres: "Rock" }

// Find concerts with tickets under $100
{ ticketPrice: { $lt: 100 } }

// Find concerts at Enterprise Center (by venue ObjectId)
{ venue_ids: ObjectId("5f8d04b3b54764421b7156d2") }
```

Array Matching
When querying an array field like genres or venue_ids, MongoDB checks if ANY element matches. The venue_ids array lets us find all concerts that played at a specific venue.

4. MongoDB Shell (mongosh)

The MongoDB Shell is a command-line interface for interacting with MongoDB. While Compass is great for exploring, the shell is essential for writing and testing queries.

4.1 Connecting via Atlas

The easiest way to use the shell is through Atlas directly:

1. In Atlas, click "Database" then click the "Connect" button
2. Select "Shell"
3. If you have not installed mongosh locally, use the web-based shell option
4. Or copy the connection command for your local mongosh installation

4.2 Essential Navigation Commands

```
// Show all databases
show dbs

// Switch to the concert_demo database
use concert_demo

// Show collections in current database
show collections

// Show current database name
db
```

4.3 Basic Query Commands

```
// Find all documents in concerts collection
db.concerts.find()

// Find one document (first match)
db.concerts.findOne()

// Find with a filter
db.concerts.find({ releaseYear: 2024 })

// Count documents
db.concerts.countDocuments()

// Limit results to 3
db.concerts.find().limit(3)

// Pretty print for readability
db.concerts.find().pretty()
```

Shell Tip
Use the UP arrow key to recall previous commands. Use Ctrl+C to cancel a long-running command.

4.4 Demo: Querying concert_demo

Let us run through some queries together:

Find all concerts

```
db.concerts.find()
```

Find concerts by a specific artist

```
db.concerts.find({ artist: "Taylor Swift" })
```

Find concerts with high attendance (over 50,000)

```
db.concerts.find({ attendance: { $gt: 50000 } })
```

Find all venues in Missouri

```
db.venues.find({ state: "MO" })
```

Find reviews for a specific concert

```
db.reviews.find({ concert_id: ObjectId("5f8d04b3b54764421b7156c1") })
```

Count fans by membership level

```
db.fans.countDocuments({ membershipLevel: "Gold" })
```

5. Query Fundamentals

Understanding query structure is essential for working with MongoDB.

5.1 Query Structure

```
db.collection.find( { }, { } )

// filter: which documents to find (like WHERE in SQL)
// projection: which fields to return (like SELECT in SQL)
```

5.2 Comparison Operators

MongoDB uses special operators prefixed with $ for comparisons:

Operator	Meaning	SQL Equivalent
$eq	Equals	=
$ne	Not equals	<>
$gt	Greater than	>
$gte	Greater than or equal	>=
$lt	Less than	<
$lte	Less than or equal	<=
$in	In array of values	IN (...)
$nin	Not in array of values	NOT IN (...)

5.3 Query Examples with concert_demo

Comparison Operators

```
// Tickets $100 or more
db.concerts.find({ ticketPrice: { $gte: 100 } })

// Concerts NOT in 2024
db.concerts.find({ releaseYear: { $ne: 2024 } })

// Attendance between 5000 and 50000
db.concerts.find({
attendance: { $gte: 5000, $lte: 50000 }
})
```

Working with Arrays

```
// Find Rock concerts (genre array contains "Rock")
db.concerts.find({ genres: "Rock" })

// Find concerts at a specific venue
db.concerts.find({ venue_ids: ObjectId("5f8d04b3b54764421b7156d2") })

// Find venues with capacity in specific values
db.venues.find({ capacity: { $in: [2000, 18000] } })
```

5.4 Projection: Limiting Fields Returned

The second parameter controls which fields are returned:

```
// Return only tourName and artist (plus _id by default)
db.concerts.find({}, { tourName: 1, artist: 1 })

// Exclude the setList field
db.concerts.find({}, { setList: 0 })

// Exclude _id and only show name and city
db.venues.find({}, { _id: 0, name: 1, city: 1 })
```

Projection Rules
Use 1 to include fields, 0 to exclude. You cannot mix inclusion and exclusion (except _id can always be excluded).

5.5 Sorting and Limiting

```
// Sort by ticketPrice ascending (cheapest first)
db.concerts.find().sort({ ticketPrice: 1 })

// Sort by attendance descending (highest first)
db.concerts.find().sort({ attendance: -1 })

// Get the 3 most expensive concerts
db.concerts.find().sort({ ticketPrice: -1 }).limit(3)

// Skip first 2, then get next 2 (pagination)
db.concerts.find().skip(2).limit(2)
```

5.6 SQL to MongoDB Translation

Here is how common SQL queries translate to MongoDB:

SQL Server	MongoDB
SELECT * FROM concerts	db.concerts.find()
SELECT tourName FROM concerts	db.concerts.find({}, {tourName:1})
WHERE releaseYear = 2024	find({releaseYear: 2024})
WHERE ticketPrice >= 100	find({ticketPrice: {$gte: 100}})

ORDER BY ticketPrice	find().sort({ticketPrice: 1})
TOP 5 / LIMIT 5	find().limit(5)
SELECT COUNT(*)	countDocuments()

6. Practice Activities

Complete these activities to verify your setup and practice basic queries.

Activity A: Atlas Setup Verification

- Log into your MongoDB Atlas account
- Verify your cluster is running (green status indicator)
- Confirm sample data is loaded
- Locate and copy your connection string

Activity B: Compass Connection

- Connect Compass to your Atlas cluster
- Browse the sample_mflix database
- Open the movies collection and examine a document
- Identify: arrays, nested objects, and different data types
- Use the Schema tab to see field types and distributions

Activity C: Shell Navigation

Connect to the shell and run these commands:

```
show dbs
use sample_mflix
show collections
db.movies.countDocuments()
db.movies.findOne()
```

Activity D: Basic Queries

Try writing these queries on the sample_mflix database:

- Find all movies (db.movies.find())
- Count how many movies are in the collection
- Find one movie and examine its structure
- Find movies from the year 2000
- Find movies with an IMDB rating of 8 or higher

Hint
The IMDB rating is stored in a nested object. The field path is "imdb.rating" (use quotes around it in your query).

7. Reference Links

Bookmark these resources for future reference:

MongoDB Documentation

https://www.mongodb.com/docs/manual/

Sample Mflix Documentation

https://www.mongodb.com/docs/atlas/sample-data/sample-mflix/

MongoDB Compass Documentation

https://www.mongodb.com/docs/compass/current/

MongoDB Shell (mongosh) Documentation

https://www.mongodb.com/docs/mongodb-shell/

8. What's Next

In upcoming lessons, we will cover:

- Advanced query operators ($and, $or, $not, $exists)
- Querying nested documents and arrays in depth
- Regular expressions for text searching
- Introduction to the Aggregation Pipeline
- CRUD operations: Insert, Update, and Delete

Before Moving On
Make sure your Atlas account, Compass, and shell access are all working. Load the sample datasets if you have not already!