MongoDB

Mimanshu Maheshwari

August 30, 2024

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Getting Started with MongoDB Atlas

1.1.1 Creating and Deploying on Atlas Cluster

1. Create a Organization
2. Create a Project under Organization
3. Create a cluster → Advanced Settings → Shared M0 cluster → Add username and password → Add you ip as allowed origin.
4. Open Database tab and navigate to your cluster
5. Load default data
6. Navigate to collections tab to see loaded clusters
7. Sample database fetch from analytics accounts table

```
{
  "_id": {
    "$oid": "5ca4bbc7a2dd94ee5816238c"
  },
  "account_id": {
    "$numberInt": "371138"
  },
  "limit": {
    "$numberInt": "9000"
  },
  "products": [
    "Derivatives",
    "InvestmentStock"
  ]
}
```

Listing 1.1: account test data

8. To filter on account id:

```
{
  account_id: 794875
}
```

Listing 1.2: filter on account id

9. To login into atlas login use `atlas auth login`
10. create cluster with user and password:

```
atlas setup --clusterName myAtlasClusterEDU --provider WS --currentIp --skipSampleData --
    ↪ username myAtlasDBUser --password myatlas-001 --projectId 66d19c1f26ef8b512df3b41 |
    ↪ tee atlas_cluster_details.txt
```

Listing 1.3: load cluster

11. Load sample data: `atlas clusters smapleData load myAtlasClusterEDU`

1.2 Introduction to MongoDB

1.2.1 Key Terms

Document

The basic using of data in MongoDB

Collections

A grouping of documents. Documents in collection are typically similar they don't necessarily have same structure.

Database

A container for Collections

1.3 MongoDB and The Document Model

1.3.1 The MongoDB Document Model

- MongoDB stores data in structures known as documents

Document

Docuement are displayed in JavaScript Object Notation (JSON) format but stored in Binary JavaScript Object Notation (BSON) format. BSON also supports additional data types that are unavailable in JSON

It can support All JSON data types as well as Dates, Numbers, Object Id's, and more!

ObjectID is used to create unique identifires. Every Document requires an $_id$ field, which acts as a primary key. If an inserted document doesn't include the $_id$ field, MongoDB automatically generates an ObjectId for the $_id$ field. MongoDB supports flexible schema model and polymorphic data, this allows us to store documents with different structure together. Documents may contain fields. Fields may contain different types.

Optional Schema validation

Set of constraints on the structure of documents.

1.4 Managing Databases, Colections, and Documents in Atlas Data Explorer

Explore Atlas

Chapter 2

Data Modeling

2.1 Introduction to Data Modeling

Data Modeling is process of defining how data is stored and the relationships and the relationships that exists among different entities in your data.
We refer to Organization of data inside a database as a **Schema**.
To develop schema think about your database and you need to ask these questions:

- What does my application do?
- What data will I store?
- How will users access this data?
- What data will be most valuable to me?

Asking these questions will help you to:

- descibe your task as well those of users.
- What you data looks like?
- the relationships among the data.
- The tooling you plan to have.
- The access patterns that might emerge.

A good data model can:

- Make it easier to manage data
- Make queries more efficient
- Use less memory and Computer Processing Unit (CPU)
- Reduce cost

As a general pricipal of MongoDB data that is accessed together should be stored together.
Collections do not enforce any document structure by default. That means document even those in the same collection can have different structures.
Each document at MongoDB can be different which is known as **Polymorphism**.
Embedded document model enables us to build complex relationships among data. You can normalize your data by using database references.
The key pricipal here is how your application will use the data rather than how it's stored in the database.

2.2 Types of Data Relationships

2.2.1 Different types of relationships data can have:

- One-to-one
- One-to-many
- Many-to-Many

One-to-One

A relationship where a data entity in one set is connected to exactly one data entity in another set.

One-to-Many

A relationship where a data entity in one set is connected to any number of entities in another set.

Many-to-Many

A relationship where any number of data entities in one set are connected to any number of data entities in another set.

2.2.2 Two main ways to model these relationships:

- Embeding
- Referencing

Embeding

We take related data and insert it into our document.

Referencing

When we refer to documents in another collection in our document. Reference is made by their respective `ObjectID`.

2.3 Modeling Data Relationships

2.3.1 Embedding Data in Documents

Also known as nested document.

- Avoids application joins.
- Provides better performance for read operatinos.
- Allows developers to update related data in a single write operations.

Issues:

- Embedding data into a single document can create large documents.
- Large documents have to be read into memory in full, which can result in a slow application performance for you end user.
- Continuesly adding data without limit creates **unbounded documents.**
- Unbounded Documents may exceed the BSON document threshold of 16MB. This is schema anit-patterns which you should avoid.

2.3.2 Referencing Data in Documents