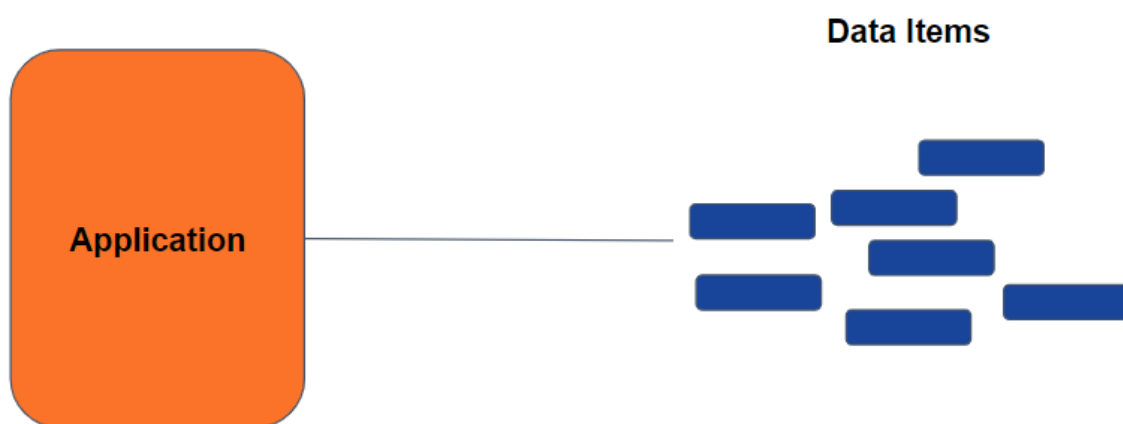# Getting Started with Databases

Understanding Data

Data in Applications

- Data in applications refers to the information the application utilizes to function correctly.

- In an e-commerce backend application, examples of data include users, products, cart items, and orders.

- Applications like Facebook, Twitter, and WhatsApp also handle various types of data, such as user data, posts, comments, pictures, videos, etc.

Importance of Data Management

- Data is the main asset of any application, and without it, the application is rendered useless.

- Currently, the application uses in-memory storage, leading to data loss upon server restarts.

- To avoid data loss, a persistent storage solution is required.

Need for Persistent Storage

- Persistent storage ensures that data remains intact even after the server restarts.

- Persistent storage prevents data loss and maintains the application's functionality and usability.

- File systems can provide persistent storage but may need to be more efficient for handling complex data operations.

- Persistent storage stands as a foundational pillar in data management, guaranteeing the preservation of data integrity even in the face of server restarts or system failures. This resilient storage approach is pivotal in averting data loss, ensuring uninterrupted application functionality, and sustaining the usability of vital information.

Challenges with File Systems

- While file systems offer persistent storage, they are not optimised for data manipulation operations.

- Managing data with file systems can become difficult for applications with large amounts of data (e.g., Facebook, Google, LinkedIn).

Introduction to Databases

- Databases are tools that provide persistent storage and enable efficient data manipulation operations.

- A database allows users to effectively create, read, update, and delete data (CRUD operations).

- Retrieving data from databases efficiently is crucial for populating user profiles or timelines.

Significance of Databases

- Applications like Facebook heavily rely on databases as their primary tool for managing and storing data.

- Losing user data for such companies would mean losing their entire wealth and value.

Understanding Databases
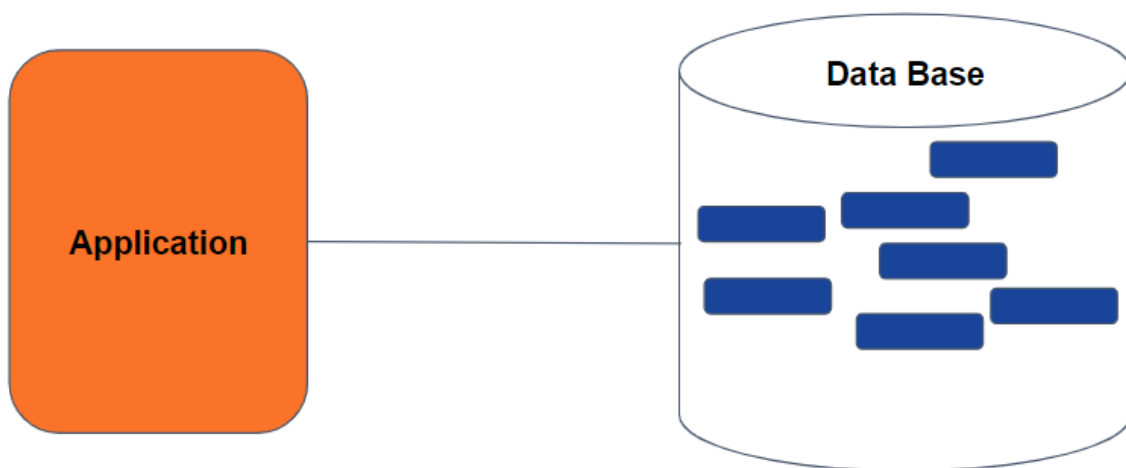
A database is a software tool that stores and manages data for applications. Applications communicate with databases to store, retrieve, and perform operations on data. Different types of databases exist, depending on the nature of the application's data.

Relational Databases

Employee

ID	Name	Address	DeptID
1	John Doe	London, UK	2
2	Samual	Mumbai, IN	1

Department

ID	Name
1	Finance
2	Sales

- Relational databases store data in tabular format (rows and columns), similar to spreadsheets.

- Data is structured with a defined schema, specifying the columns and their data types.

- Primary keys (unique identifiers) and foreign keys establish relationships between tables.

- Relationships allow data from different tables to be linked and related to each other.

- Relational databases are suited for applications with structured and predictable data.

NoSQL Databases

Employee

```
{_id:1, name: 'John Doe' }
{_id:2, name: 'Samual Doe', deptID:1 }
{_id:2, name: 'Samual Doe', dept:{
        name: 'Sales'
}}
```

Table -> Collection
Row -> Document
Column -> Attribute

Department

```
{_id:1, name: 'Finance' }
{_id:2, name: Sales', manager:'Raman' }
```

- NoSQL databases are schemaless, allowing flexibility in storing unstructured or unpredictable data.

- Data is stored in JSON-like format, allowing nested structures within documents.

- NoSQL databases are ideal for applications with varying and unpredictable data attributes.

- They offer more straightforward and efficient data retrieval, especially for nested data.

Differences between Relational and NoSQL Databases

Relational	NoSQL
Has strict schema/structure	Schemaless
Row-column format	Json format
Predictable Data	Difficult to predict data
Performance issues while reading with large set of data and relationships	Better at reading from large data set as it supports complex nested structure.
MySQL, Postgres, SQL Server	MongoDB, DynamicDB, CouchDB

Popular Database Types

- Popular relational databases include MySQL, PostgreSQL, and SQL Server.

- Popular NoSQL databases include MongoDB, DynamoDB, and CosDB.

- Cloud platforms like AWS and Azure also offer databases as a service.

Characteristics that make each of these databases popular choices.

Popular relational databases like MySQL, PostgreSQL, and SQL Server are favored for their structured data model, ACID compliance, and strong consistency, making them ideal for applications requiring strict data integrity.

In contrast, MongoDB, DynamoDB, and CosDB are prominent NoSQL databases celebrated for their flexibility, scalability, and ability to handle unstructured or semi-structured data efficiently, catering to modern applications' diverse data requirements.

Cloud platforms such as AWS and Azure further provide managed database services, streamlining deployment and management tasks for various database systems.
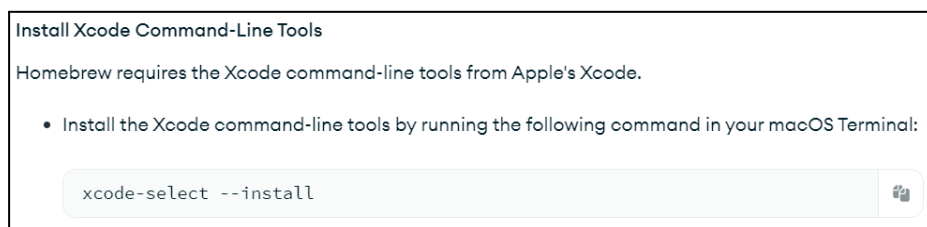
Installing MongoDB and MongoDB Compass

1. MongoDB Installation on macOS:

    a. MongoDB is a popular NoSQL database we are going to learn in depth.

    b. To install MongoDB on macOS, visit the official MongoDB website. **Link**

    c. MongoDB offers both a paid enterprise version and a free community version.

    d. The community version is suitable for learning purposes, and it can be installed using Homebrew.

2. Steps to Install MongoDB Community Edition:

    a. MongoDB may need Xcode and command line tools to be installed, so execute the prerequisite command if necessary.

Install Xcode Command-Line Tools
Homebrew requires the Xcode command-line tools from Apple's Xcode.

- Install the Xcode command-line tools by running the following command in your macOS Terminal:

```
xcode-select --install
```

    b. Ensure Homebrew is installed, a package manager for macOS. **Link**

    c. Tap MongoDB Homebrew to download the official formula for MongoDB.

Installing MongoDB 6.0 Community Edition

Follow these steps to install MongoDB Community Edition using Homebrew's `brew` package manager. Be sure that you have followed the installation prerequisites above before proceeding.

1. Tap the MongoDB Homebrew Tap⧉ to download the official Homebrew formula for MongoDB and the Database Tools, by running the following command in your macOS Terminal:

```
brew tap mongodb/brew
```

If you have already done this for a previous installation of MongoDB, you can skip this step.

2. To update Homebrew and all existing formulae:

```
brew update
```

3. To install MongoDB, run the following command in your macOS Terminal application:

```
brew install mongodb-community@6.0
```

d. Update Homebrew and existing formulas.

e. Install MongoDB Community Edition (version 6.0 or any desired version).

f. MongoDB will start running, and you can verify the installation using the "mongo version" command.

3. Installing MongoDB Compass:

a. MongoDB Compass is a UI tool for MongoDB management provided by MongoDB.

b. Download MongoDB Compass from the official website and install it. **Link**

c. Once installed, open MongoDB Compass to connect to the MongoDB database.

d. MongoDB Compass allows you to visualise the data stored in the database and perform various operations.

4. MongoDB Compass UI:

a. MongoDB Compass provides a GUI to interact with MongoDB databases.

b. After connecting to the MongoDB database, you can view existing databases.

c. By default, there are three prebuilt databases: admin, config, and local.

   d. Users can create their databases, store documents, and manage collections.
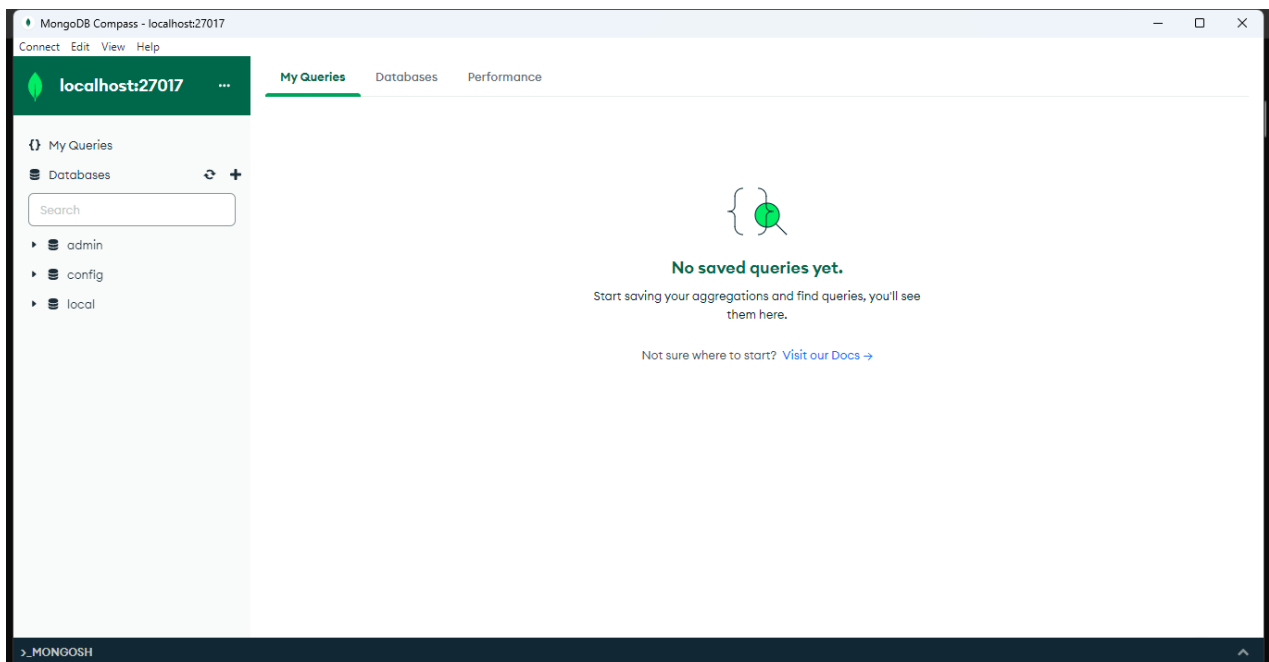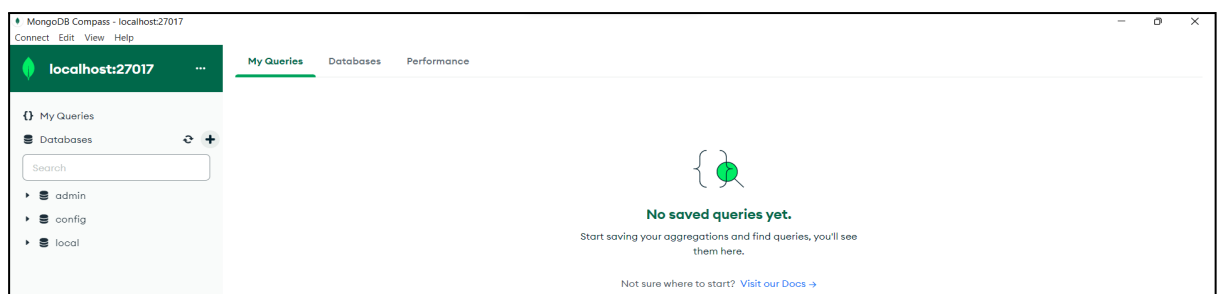

Documents and Collection in MongoDB
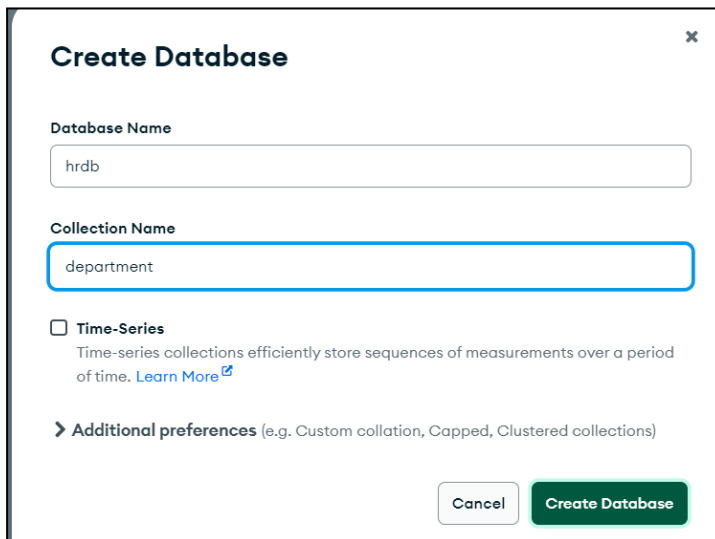
MongoDB Data Structure

- In MongoDB, data is organized in collections and documents.
- A collection is similar to a table in a relational database.
- Each collection contains multiple documents, which are equivalent to rows in a table.
- Documents are stored in JSON format, providing flexibility in data structure.

Creating a Database and Collection

- MongoDB Compass allows the visual management of databases and collections.
- To create a new database, click on the plus button (+) and enter the database name (e.g., "HRDB").

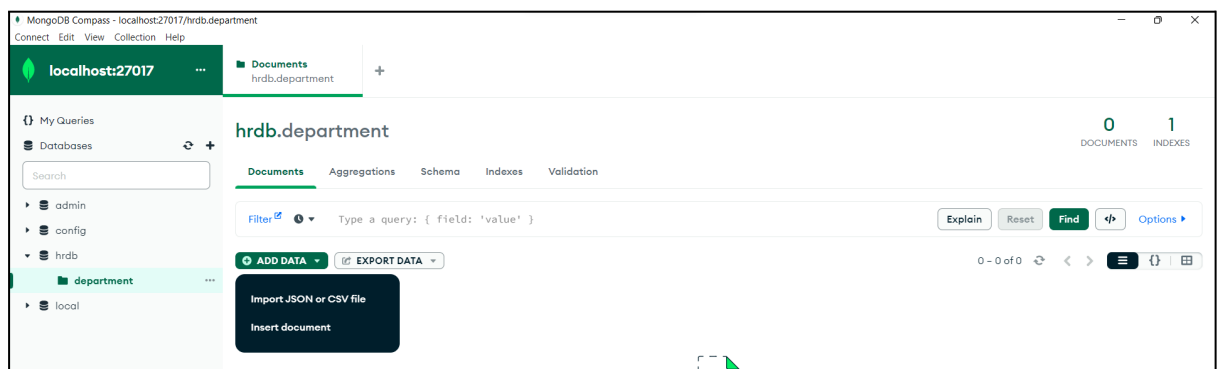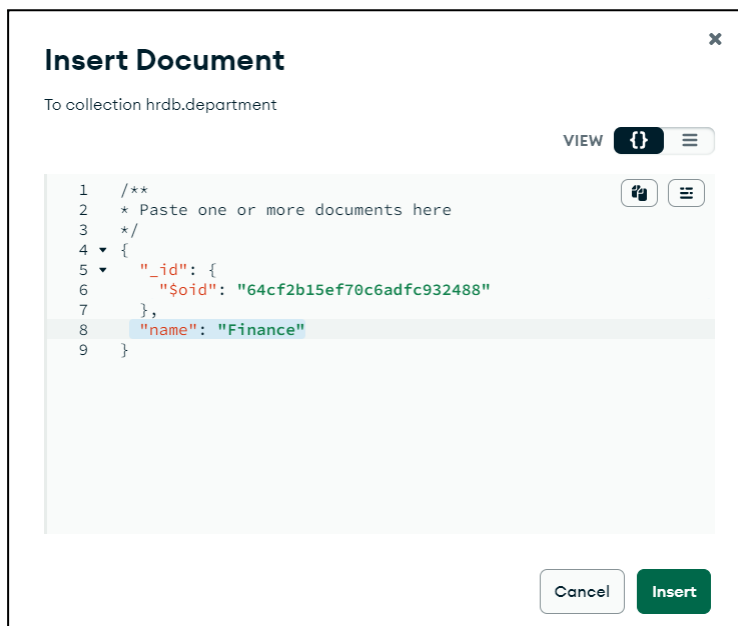- Inside the database, a collection can be created (e.g., "department").

- Collections can store multiple documents with various attributes.

Inserting Documents

- Documents are inserted manually using the "Insert Document" option.

- MongoDB automatically assigns a unique ID to each document.

- Documents can have attributes, and their structure can vary across documents.

- MongoDB provides flexibility to store attributes specific to individual documents.

Nested JSON Documents

- MongoDB allows the creation of nested JSON documents.

- For example, a document can contain a nested object with subattributes.

- This flexibility is suitable for applications requiring varying data structures.

CRUD Operations:

- CRUD stands for Create, Read, Update, and Delete operations.

- These operations are essential for interacting with the database.

- While MongoDB Compass provides a user-friendly UI for CRUD operations, developers need to learn MongoDB commands for programmatic interactions.

Performing CRUD Operations Using Commands

- Understanding MongoDB commands is crucial for Node.js applications to interact with MongoDB databases programmatically.

Application of MongoDB in RealWorld Scenarios

- Real-world applications often use a combination of databases, including both relational and NoSQL databases.

- Each database type serves specific purposes based on the application's needs.

- MongoDB's flexibility makes it suitable for applications where data structures may vary unpredictably.

Documents CRUD in MongoDB 1

Performing CRUD operations using MongoDB commands for programmatic interactions. The MongoSH shell command line tool will be used to execute MongoDB commands.

Connecting to MongoDB

- The `mongosh` command is executed from the MongoDB installation directory (user/local/bin on macOS, Program Files on Windows).

- MongoSH connects to the MongoDB server and displays version details.

Creating a Database and Collection

- The command `use <database_name>` creates a new database or switches to an existing one (e.g., "use bookDB").

- Newly created databases take time to be visible; they require collections or documents to be present.

```
test> show dbs
admin     80.00 KiB
bookdb    72.00 KiB
config   220.00 KiB
hrdb      72.00 KiB
local    420.00 KiB
test> use bookdb
switched to db bookdb
bookdb> show dbs
admin     80.00 KiB
bookdb    72.00 KiB
config   220.00 KiB
hrdb      72.00 KiB
local    420.00 KiB
```

Inserting Documents

- The "insertOne" command inserts a single document into a collection.

- Documents are represented as JSON objects.

- MongoDB automatically generates the unique "_id" field and is globally unique.

```
bookdb> db.books.insertOne({title:"Da Vinci Code", author:"Dan Brown"})
{
  acknowledged: true,
  insertedId: ObjectId("64cf35624ed263dcd47bd150")
}
bookdb> show dbs
admin     40.00 KiB
bookdb    40.00 KiB
config   108.00 KiB
hrdb      72.00 KiB
local     40.00 KiB
bookdb> show collections
books
```

- Multiple documents can be inserted using the "insertMany" command with an array of JSON objects.

```
bookdb> db.books.insertMany([{
...       "author": "Chinua Achebe",
...       "country": "Nigeria",
...       "imageLink": "images/things-fall-apart.jpg",
...       "language": "English",
...       "link": "https://en.wikipedia.org/wiki/Things_Fall_Apart\n",
...       "pages": 209,
...       "title": "Things Fall Apart",
...       "year": 1958
...   },
...   {
...       "author": "Hans Christian Andersen",
...       "country": "Denmark",
...       "imageLink": "images/fairy-tales.jpg",
...       "language": "Danish",
...       "link": "https://en.wikipedia.org/wiki/Fairy_Tales_Told_for_Children._First_Collection.\n",
...       "pages": 784,
...       "title": "Fairy tales",
...       "year": 1836
...   },
...   {
...       "author": "Dante Alighieri",
...       "country": "Italy",
...       "imageLink": "images/the-divine-comedy.jpg",
...       "language": "Italian",
...       "link": "https://en.wikipedia.org/wiki/Divine_Comedy\n",
...       "pages": 928,
...       "title": "The Divine Comedy",
...       "year": 1315
...   },
...   {
...       "author": "Unknown",
...       "country": "Sumer and Akkadian Empire",
...       "imageLink": "images/the-epic-of-gilgamesh.jpg",
bookdb>
...       "link": "https://en.wikipedia.org/wiki/Epic_of_Gilgamesh\n",
...       "pages": 160,
...       "title": "The Epic Of Gilgamesh",
...       "year": -1700
...   },
...   {
...       "author": "Unknown",
...       "country": "Achaemenid Empire",
...       "imageLink": "images/the-book-of-job.jpg",
...       "language": "Hebrew",
...       "link": "https://en.wikipedia.org/wiki/Book_of_Job\n",
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64cf368b4ed263dcd47bd151"),
    '1': ObjectId("64cf368b4ed263dcd47bd152"),
    '2': ObjectId("64cf368b4ed263dcd47bd153"),
    '3': ObjectId("64cf368b4ed263dcd47bd154"),
    '4': ObjectId("64cf368b4ed263dcd47bd155"),
    '5': ObjectId("64cf368b4ed263dcd47bd156"),
    '6': ObjectId("64cf368b4ed263dcd47bd157"),
    '7': ObjectId("64cf368b4ed263dcd47bd158")
  }
}
```

- MongoDB compass view:

bookdb.books

Documents   Aggregations   Schema   Indexes   Validation

Filter   🕐 ▾   Type a query: { field: 'value' }   Explain   Reset   Find   </>   Options ▸

⊕ ADD DATA ▾   ⬈ EXPORT DATA ▾   1–9 of 9

🏠 books

	_id ObjectId	title String	author String	country String	imageLink String	
1	ObjectId('64cf35624ed263dcd47…	"Da Vinci Code"	"Dan Brown"	No field	No field	
2	ObjectId('64cf368b4ed263dcd47…	"Things Fall Apart"	"Chinua Achebe"	"Nigeria"	"images/things-fall-apart.jpg"	
3	ObjectId('64cf368b4ed263dcd47…	"Fairy tales"	"Hans Christian Andersen"	"Denmark"	"images/fairy-tales.jpg"	
4	ObjectId('64cf368b4ed263dcd47…	"The Divine Comedy"	"Dante Alighieri"	"Italy"	"images/the-divine-comedy.jpg"	
5	ObjectId('64cf368b4ed263dcd47…	"The Epic Of Gilgamesh"	"Unknown"	"Sumer and Akkadian Empire"	"images/the-epic-of-gilgamesh…	
6	ObjectId('64cf368b4ed263dcd47…	"The Book Of Job"	"Unknown"	"Achaemenid Empire"	"images/the-book-of-job.jpg"	
7	ObjectId('64cf368b4ed263dcd47…	"One Thousand and One Nights"	"Unknown"	"India/Iran/Iraq/Egypt/Tajiki…	"images/one-thousand-and-one-…	
8	ObjectId('64cf368b4ed263dcd47…	"Njál's Saga"	"Unknown"	"Iceland"	"images/njals-saga.jpg"	
9	ObjectId('64cf368b4ed263dcd47…	"Pride and Prejudice"	"Jane Austen"	"United Kingdom"	"images/pride-and-prejudice.j…	

Viewing Documents

- The "find" command returns all documents in a collection.

```
bookdb> db.books.find()
[
  {
    _id: ObjectId("64cf35624ed263dcd47bd150"),
    title: 'Da Vinci Code',
    author: 'Dan Brown'
  },
  {
    _id: ObjectId("64cf368b4ed263dcd47bd151"),
    author: 'Chinua Achebe',
    country: 'Nigeria',
    imageLink: 'images/things-fall-apart.jpg',
    language: 'English',
    link: 'https://en.wikipedia.org/wiki/Things_Fall_Apart\n',
    pages: 209,
    title: 'Things Fall Apart',
    year: 1958
  },
  {
    _id: ObjectId("64cf368b4ed263dcd47bd152"),
    author: 'Hans Christian Andersen',
    country: 'Denmark',
    imageLink: 'images/fairy-tales.jpg',
    language: 'Danish',
    link: 'https://en.wikipedia.org/wiki/Fairy_Tales_Told_for_Children._First_Collection.\n',
    pages: 784,
    title: 'Fairy tales',
    year: 1836
  },
  {
    _id: ObjectId("64cf368b4ed263dcd47bd153"),
    author: 'Dante Alighieri',
    country: 'Italy',
    imageLink: 'images/the-divine-comedy.jpg',
    language: 'Italian',
    link: 'https://en.wikipedia.org/wiki/Divine_Comedy\n',
    pages: 928,
    title: 'The Divine Comedy',
    year: 1315
  },
  {
    _id: ObjectId("64cf368b4ed263dcd47bd154"),
    author: 'Unknown',
    country: 'Sumer and Akkadian Empire',
    imageLink: 'images/the-epic-of-gilgamesh.jpg',
    language: 'Akkadian',
    link: 'https://en.wikipedia.org/wiki/Epic_of_Gilgamesh\n',
    pages: 160,
    title: 'The Epic Of Gilgamesh',
```

- To find specific documents based on conditions, use "find" with a query (e.g., "find({ year: 1813 })").

```
bookdb> db.books.findOne({"year":1813})
{
  _id: ObjectId("64cf368b4ed263dcd47bd158"),
  author: 'Jane Austen',
  country: 'United Kingdom',
  imageLink: 'images/pride-and-prejudice.jpg',
  language: 'English',
  link: 'https://en.wikipedia.org/wiki/Pride_and_Prejudice\n',
  pages: 226,
  title: 'Pride and Prejudice',
  year: 1813
}
bookdb>
```

- "findOne" returns only one matching document, while "find" returns all matching documents.

```
bookdb> db.books.find({"author": "Unknown"})
[
  {
    _id: ObjectId("64cf368b4ed263dcd47bd154"),
    author: 'Unknown',
    country: 'Sumer and Akkadian Empire',
    imageLink: 'images/the-epic-of-gilgamesh.jpg',
    language: 'Akkadian',
    link: 'https://en.wikipedia.org/wiki/Epic_of_Gilgamesh\n',
    pages: 160,
    title: 'The Epic Of Gilgamesh',
    year: -1700
  },
  {
    _id: ObjectId("64cf368b4ed263dcd47bd155"),
    author: 'Unknown',
    country: 'Achaemenid Empire',
    imageLink: 'images/the-book-of-job.jpg',
    language: 'Hebrew',
    link: 'https://en.wikipedia.org/wiki/Book_of_Job\n',
    pages: 176,
    title: 'The Book Of Job',
    year: -600
  },
  {
    _id: ObjectId("64cf368b4ed263dcd47bd156"),
    author: 'Unknown',
    country: 'India/Iran/Iraq/Egypt/Tajikistan',
    imageLink: 'images/one-thousand-and-one-nights.jpg',
    language: 'Arabic',
    link: 'https://en.wikipedia.org/wiki/One_Thousand_and_One_Nights\n',
    pages: 288,
    title: 'One Thousand and One Nights',
    year: 1200
  },
  {
    _id: ObjectId("64cf368b4ed263dcd47bd157"),
    author: 'Unknown',
    country: 'Iceland',
    imageLink: 'images/njals-saga.jpg',
    language: 'Old Norse',
    link: 'https://en.wikipedia.org/wiki/Nj%C3%A1ls_saga\n',
    pages: 384,
    title: "Njál's Saga",
    year: 1350
  }
]
```

Documents CRUD in MongoDB 1

with the update and delete operations.

Adding More Documents

- Before performing update and delete, additional books are inserted into the "books" collection.

- Multiple books are added using the "insertMany" command with an array of JSON objects.

Update Operations

- Update operations can be performed using "updateOne" and "updateMany" commands.

- The "updateOne" function takes a filter to find the document and an update to specify the attributes to modify.

```
bookdb> db.books.updateOne({title:'Da Vinci Code'}, {$set:{author:'Daniel Brown'}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
bookdb>
```

- The "updateMany" function updates multiple documents based on the provided filter.

```
bookdb> db.books.updateMany({author:'J.K. Rowling'}, {$set:{year:2015}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
bookdb>
```

Delete Operations

- Delete operations can be performed using "deleteOne" and "deleteMany" commands.

- The "deleteOne" function deletes a single document based on the provided filter.

```
bookdb> db.books.deleteOne({title:'To Kill a mocking bird'})
{ acknowledged: true, deletedCount: 1 }
```

- The "deleteMany" function deletes multiple documents based on the filter.

```
bookdb> db.books.deleteMany({year:2015})
{ acknowledged: true, deletedCount: 3 }
```

Summarising it

Let's summarise what we have learned in this module:

- Introduction to MongoDB, a NoSQL database, its key features, and how it differs from traditional relational databases.

- Step-by-step guide on installing MongoDB on macOS and Windows and setting up MongoDB Compass, a UI tool for visualisation.

- Understanding MongoDB's data structure, with documents representing records and collections acting as tables in a relational database.

- Creating a database and collections in MongoDB using MongoDB Compass and learning about the database hierarchy.

- Adding data to MongoDB collections using the UI tool MongoDB Compass and learning about document insertion.

- Performing CRUD operations in MongoDB using the UI tool, understanding filtering and finding documents based on conditions.

- Learning about MongoDB operators, such as limit and other conditional operators for querying data.

- Performing update and delete operations on MongoDB documents.

- Further exploration of CRUD operations with MongoDB commands, understanding update and delete operations in depth.

Some Additional Resources:

- **NoSQL vs. SQL Databases**

- **MongoDB CRUD Operations**