SQL Databases

Day #5

Wholeness

- In this lesson, we will learn the concepts, principles and techniques for how to create suitable Database design and how to implement it for an enterprise application software solution.
- A Database is an organized, structured collection of data, useful to an enterprise.
- Science of Consciousness: *Order is present everywhere.*

1.1 Overview

Meaning of essentials terms:

- Data: is facts or figures collected together through observation or measurement, for the purpose of referencing or analysis.
- Information: is what is derived, when data has been made to become meaningful or useful through processing, interpreting or presenting.
- Database: is a structured collection of related data.
- Database Management system (DBMS): is software that manages and controls access to and operation of, a database.

1.1 Overview

- Database Application: is any computer program that interacts with a database at some point during its execution.
- Database system: is the collection of application programs that interact with the database, together with the DBMS and the database itself.

Note: All access to the Database is through the DBMS.

1.1 Overview

Why study Databases?

- The Database system is perhaps the most important development in the field of software engineering.
- A Database typically forms the underlying framework of the information system on which most organizations operate.
- The importance of database systems keeps increasing, driven by significant developments in hardware capability/capacity and data communication and the emergence of the Internet, eCommerce, Business Intelligence and network/grid computing.

1.1 Overview

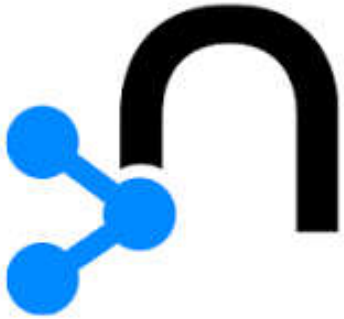
- Some areas of application of Database systems:
  - Purchases from the supermarket.
  - Online purchases using your credit card.
  - Booking a vacation with a travel agency
  - Using the local/school library
  - Taking out insurance
  - Finding and watch a movie
  - Conducting online or traditional banking
  - Studying at a college or university
  - Using the Internet/World-wide websites

Main Point 1

- It is important that we understand what Databases are, their usefulness within the context of an overall enterprise information system and the related terminologies surrounding the study of Database systems. *Knowledge is different in different states of consciousness.*

Relational Database

Non-Relational Database



1.10 Introduction to MySQL

- MySQL is an open-source relational database management system (RDBMS)
- In this course we will be studying and implementing the various Database Management system concepts and techniques using a MySQL instance
- Therefore, you are required to download and install a MySQL Database server on your local machine (if you do not already have it).*
- MySQL can be obtained from https://dev.mysql.com/

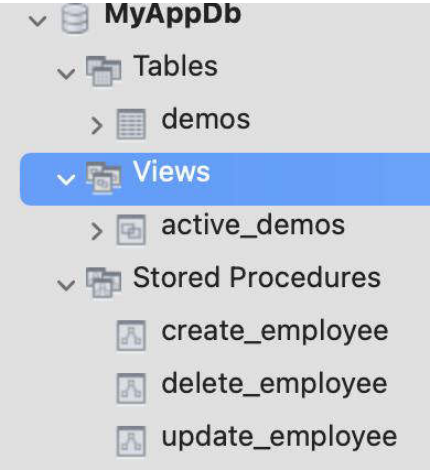
***Note: A demonstration of how to obtain and install MySQL will be given.**

Data Modeling & Database Design

- Conceptual database model
  - Entity-Relationship model (E-R diagram)
  - Not tied to any specific RDBMS, yet
- Physical data model
  - Implementation of the E-R model on to a physical RDBMS using DB tables, views, stored procedures etc.
  - Includes RDBMS-specific data types and constructs

How to create a view

```
CREATE VIEW active_demos AS SELECT * FROM MyAppDB.demos  WHERE status = 'active';
```



Implementing Relational Database

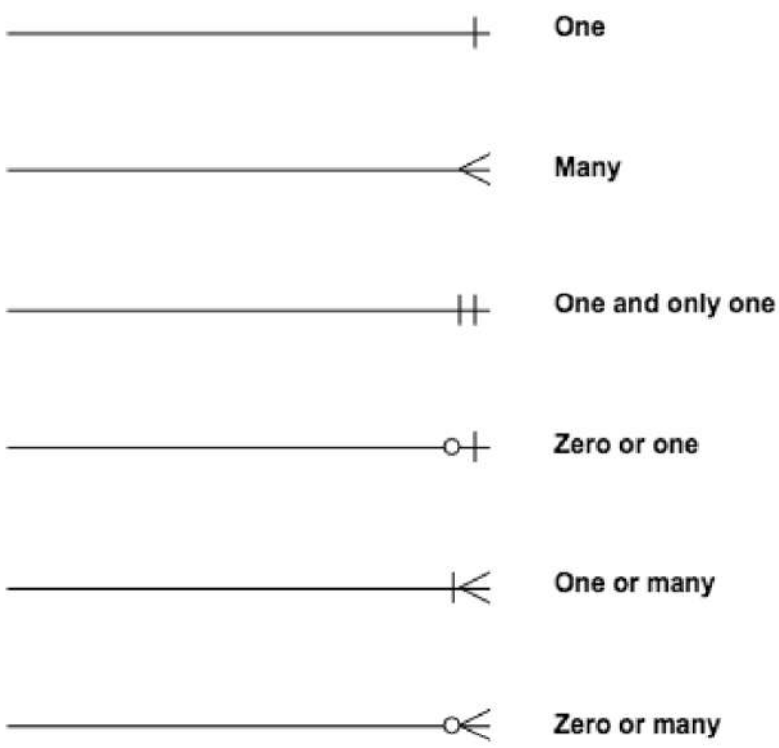
- Two possible approaches:
  - Bottom-up (a.k.a Database-First approach)
    - The Database objects (tables, views etc.) are created first, before the application code is written
    - This can be down either Interactively, using a Database client tool, such as MySQL Workbench or SQL Script
  - Top-down (a.k.a Code-First approach)
    - The Domain Entity classes are first coded in the application
    - And then an Object-Relational Mapping framework/tool is used to generate the corresponding Database objects.

The database design process aims to create database structures that will efficiently store and manage data (Rob & Coronel, 2004)
Ref: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=317584283d177f704b41c0256847888b1b95f47b

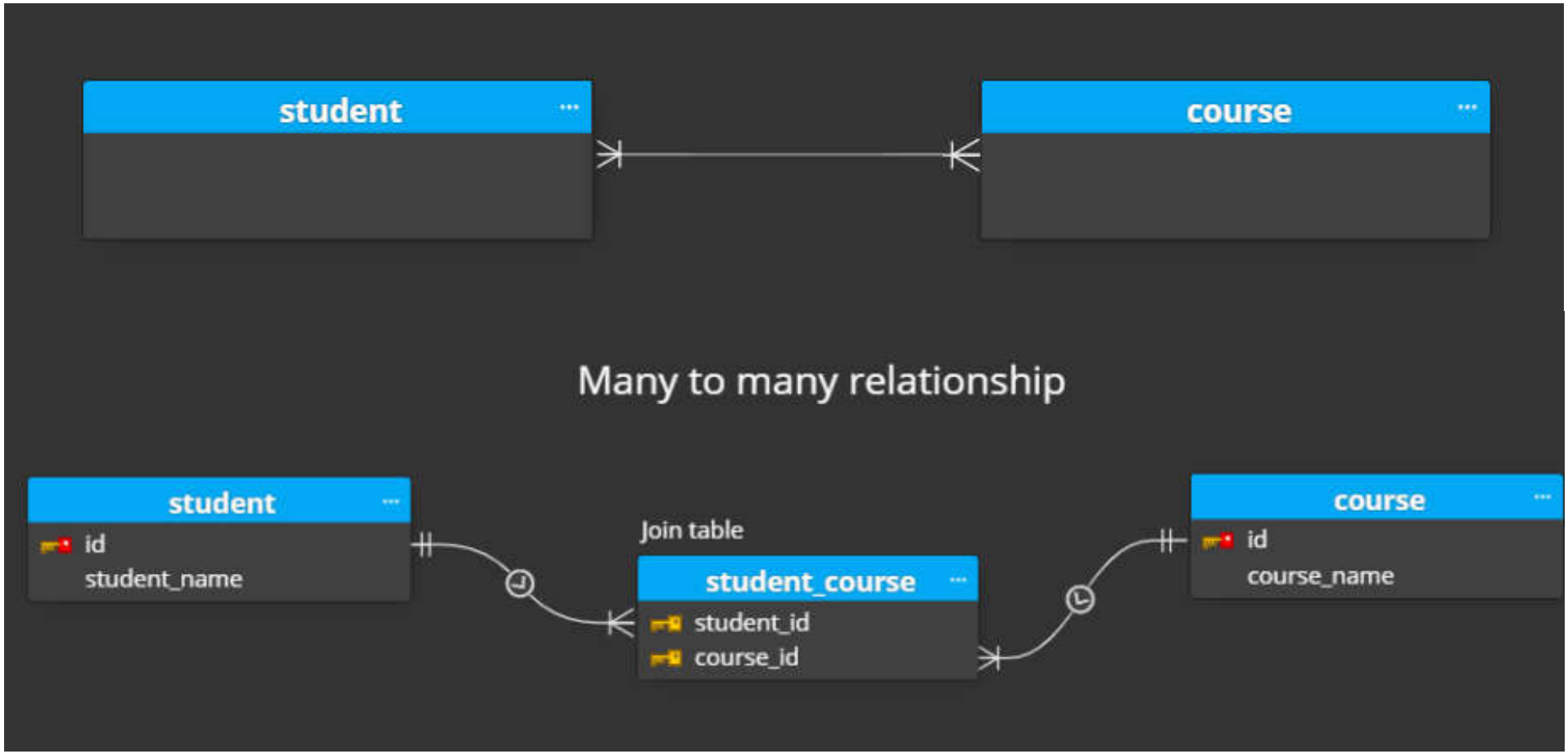
Programming Language	ORM Tools	Description
Java	Hibernate	Mature and widely used, offering powerful and flexible object-relational mapping.
	EclipseLink	Open-source Java persistence platform supporting various data services.
	JPA (Java Persistence API)	Standard specification for accessing relational databases from Java applications.
	DataNucleus	Open-source JDO and JPA implementation for persistence across various data stores.
	jOOQ	Fluent API for building type-safe SQL queries, alternative to string-based SQL.
Python	Django ORM	Integrated ORM in the Django web framework, offering high-level database access.
	SQLAlchemy	Powerful and versatile ORM, supporting various database backends and features like object relationships and unit of work patterns.
	SQLObject	Lightweight and easy-to-use ORM for Python, suitable for simpler projects.
C# and .NET	Entity Framework (EF)	Popular Microsoft-developed ORM, providing a powerful and mature solution for data access.
	Dapper	Lightweight micro-ORM for .NET, offering a simpler approach to database interactions.
	NHibernate	.NET port of the popular Hibernate ORM tool, providing similar features for C#.
Ruby on Rails	ActiveRecord	Powerful and well-integrated ORM included in the Rails framework for Ruby applications.
Node.js	TypeORM, Sequelize	Popular choices for relational database access in Node.js environment.

Database Design and Implementation Demo

- **Exercise**:
  - Create an E-R model for the City Library system. You may draw the ER diagram using a graphical drawing tool on computer or draw by hand.
  - Implement the model on a physical Relational database. Use any RDBMS of your choice e.g. MySQL

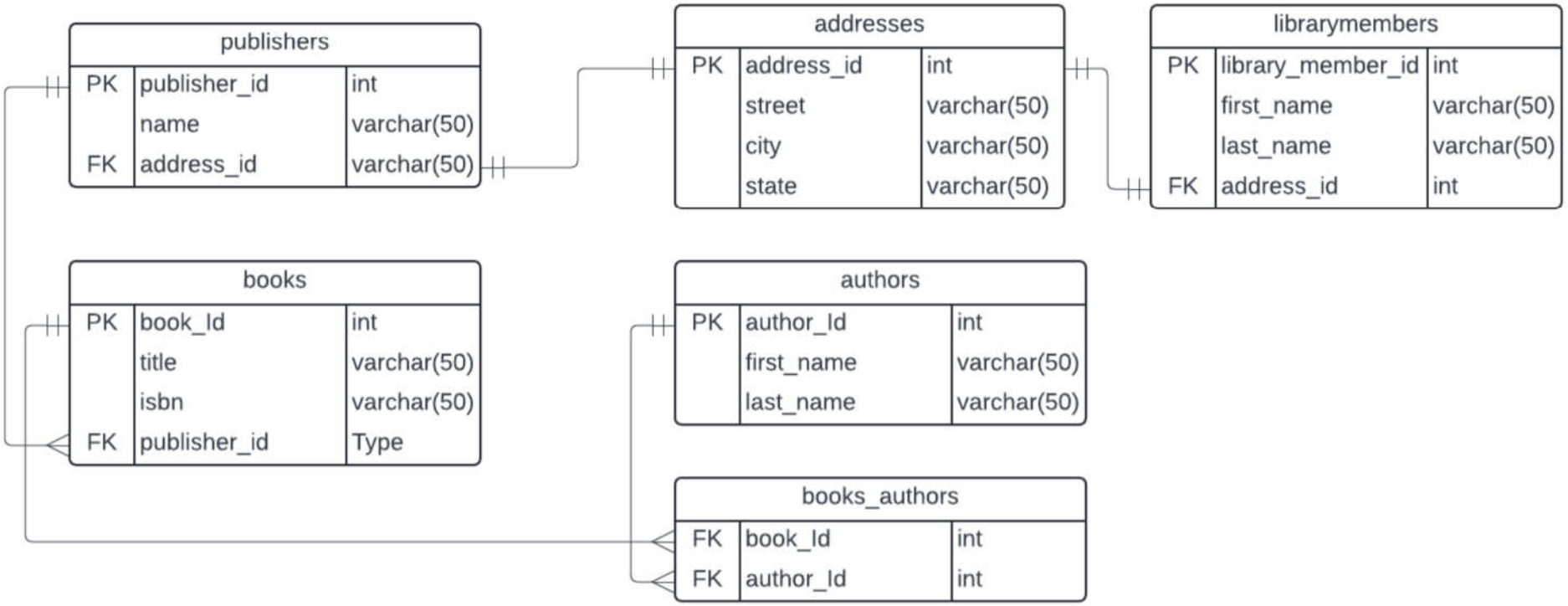

One
Many
One and only one
Zero or one
One or many
Zero or many

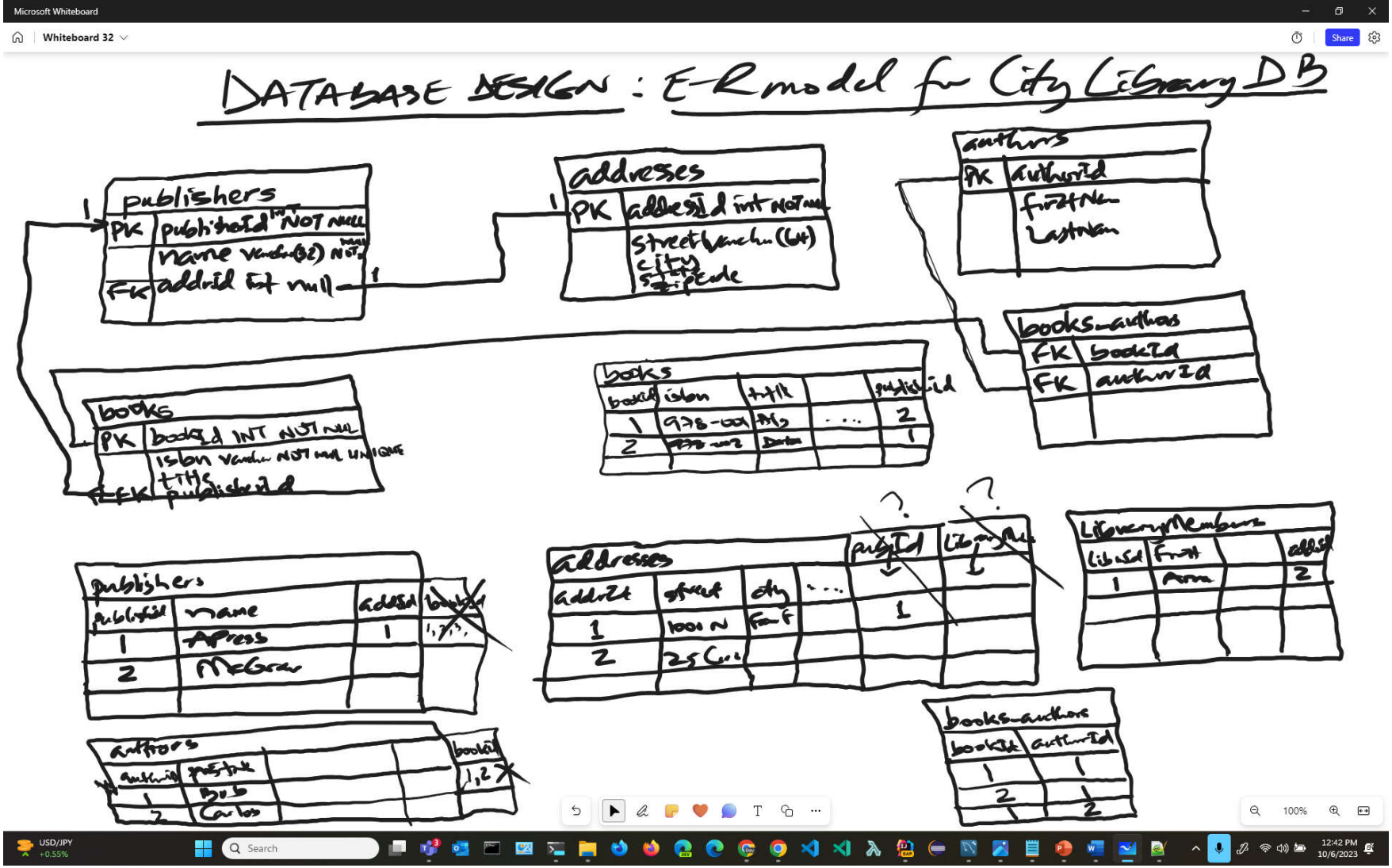
Many to Many



Ref: https://www.datensen.com/blog/er-diagram/many-to-many-relationships/

Partial Solution




Microsoft Whiteboard
Whiteboard 32
Share
DATABASE DESIGN : E-R model for City Library DB
publishers
addresses
authors
books_authors
books
books
addresses
publishers
authors
books_authors
LibraryMembers
100%
USD/JPY
+0.55%
Search
12:42 PM
10/6/2023

Validations

Annotation	Applies To	Condition
@NotNull	Any field type	Field cannot be null
@NotEmpty	String, Collection, Map, Array	Not null and size greater than zero
@NotBlank	String	Not null, not empty, and trimmed (no whitespace only)

JPA Query Method and JPQL

```
public interface PublisherRepository extends JpaRepository<Publisher, Integer> {
    @Query("select p from Publisher p")
    List<Publisher> findAllData();

    Optional<Publisher> findByPublisherName(String publisherName);

    @Query("SELECT p FROM Publisher p WHERE p.publisherName = :publisherName")
    Optional<Publisher> findDataUsingPublisherName(@Param("publisherName") String publisherName);
}
```

```
@Entity    @Table(name = "publish
public class Publisher {
    @Id
    @GeneratedValue(strategy = Ge
    @Column(name = "publisher_id"
    private Integer publisherId;
    @Column(name = "publisher_na
    private String publisherName;
    @Embedded
    private Address address;

    public Publisher(String publisher
        this.publisherName = publisher
    }
    //...
}
```

@Column and @JoinColumn

@Column annotation:

•Used on fields within an entity class.

•Defines how a specific field in your entity class maps to a column in the database table.

@JoinColumn annotation:

•Used within a one-to-one or many-to-one relationships between entities.

•Specifies how a foreign key column in the owning entity table is mapped to the primary key of the referenced entity table.

NoSQL Databases – Introduction to MongoDB

Day #6

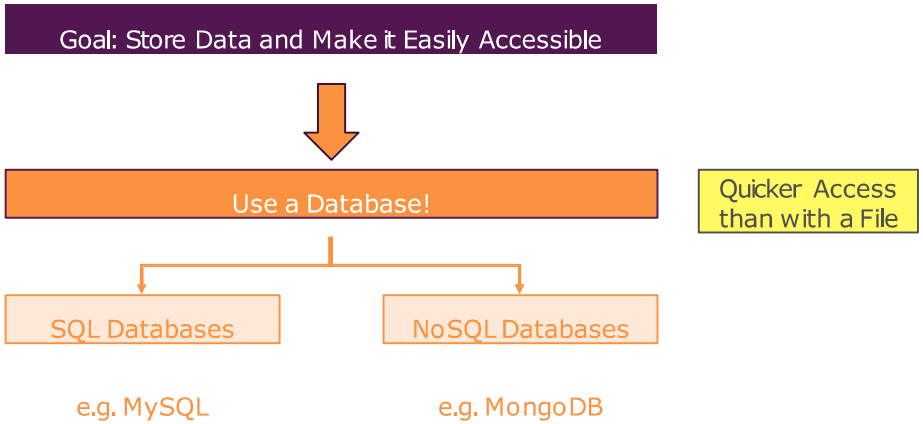
Wholeness

- This lecture aims at giving an overview and introduction to Non-relational (NoSQL) Databases and specifically focus on, a Document-oriented database named, MongoDB.
- Science of Consciousness: *Order is present everywhere; it is only our lack of understanding of the natural order of life that causes problems to arise.*

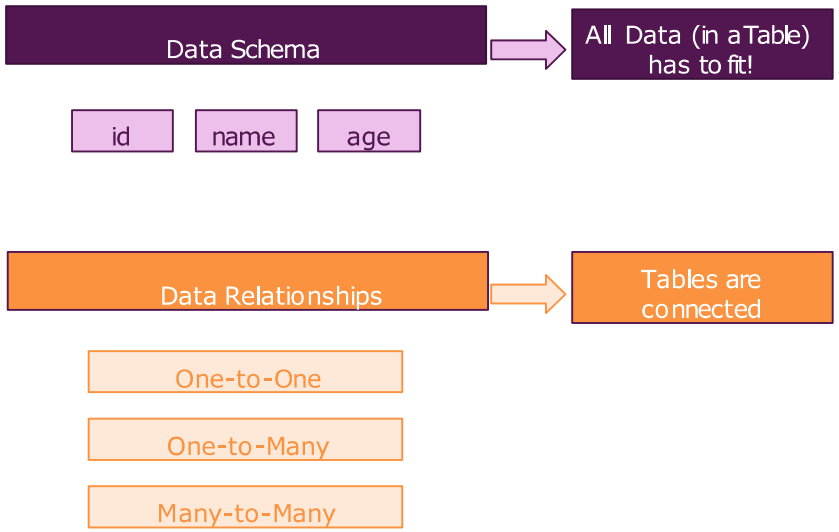
Objectives

- Know the difference between Relational (SQL) Databases and Non-relational (NoSQL) databases.
- Understand how to setup and work with MongoDB.
- Perform CRUD operations on a MongoDB database.
- Develop a MongoDB Data-driven Spring Boot CLI Application

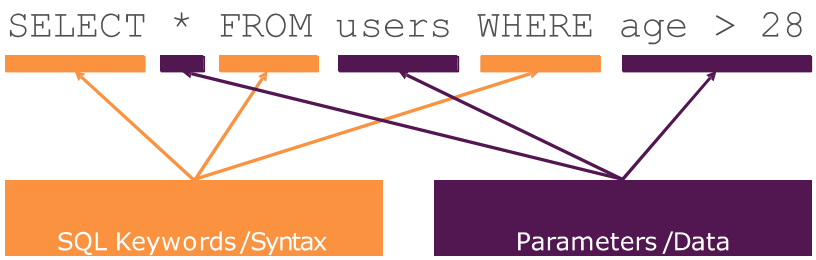
SQL vs NoSQL


Goal: Store Data and Make it Easily Accessible
Use a Database!
Quicker Access than with a File
SQL Databases
NoSQL Databases
e.g. MySQL
e.g. MongoDB

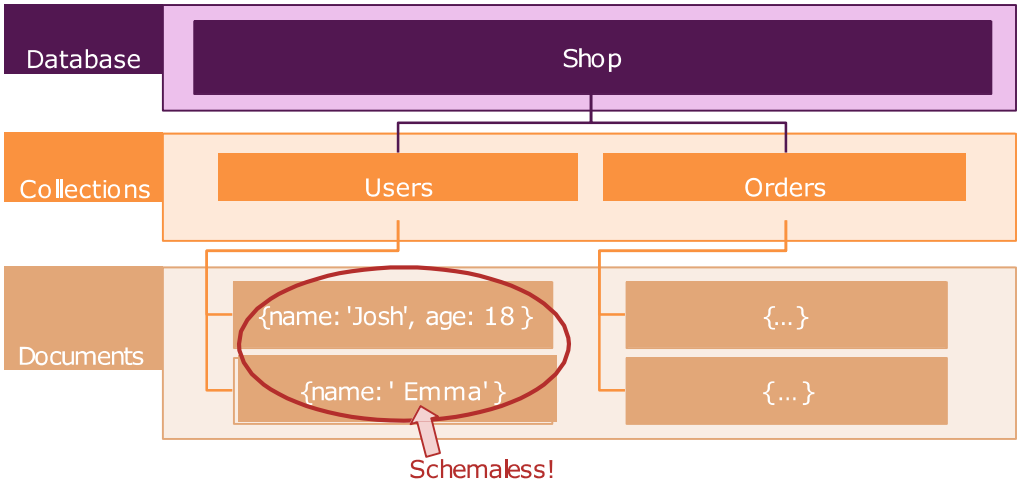
Core SQL Database Characteristics



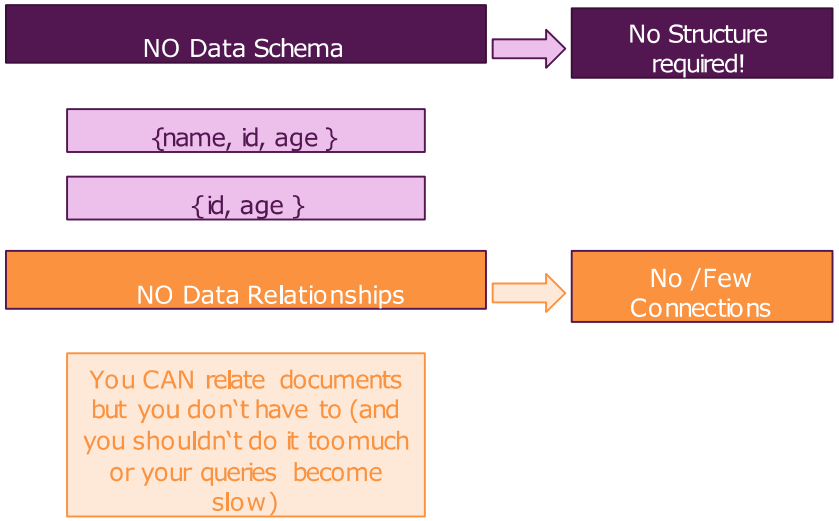
SQL Queries


SELECT * FROM users WHERE age > 28
SQL Keywords /Syntax
Parameters /Data

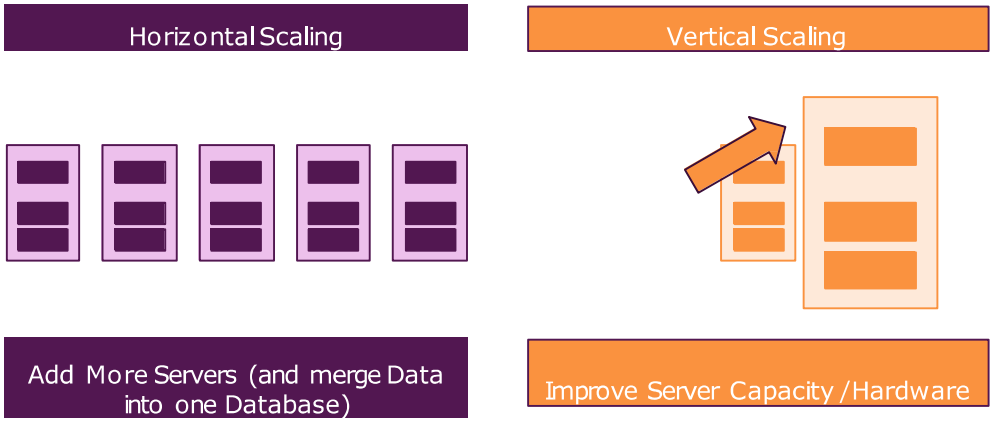
NoSQL


Database
Shop
Collections
Users
Orders
Documents
{name: 'Josh', age: 18 }
{name: ' Emma' }
{...}
{...}
Schemaless!

NoSQL Characteristics



Horizontal vs Vertical Scaling



SQL vs NoSQL

SQL	NoSQL
Data uses Schemas	Schema-less
Relationships (normalized with fk)!	No (or very few) Relationships
Data is distributed across multiple tables (normalized)	Data is typically merged /nested in a few collections (denormalized)
Horizontal scaling is difficult / impossible; Vertical scaling is possible	Both horizontal and vertical scaling is possible
Limitations for lots of (thousands) read & write queries per second	Great performance for mass read & write requests

NoSQL Revolution

- NoSQL (originally referring to "non SQL" or "non relational") databases were created for "Big Data" and Real-Time Web Applications, it provides new data architectures that can handle the ever-growing velocity and volume of data.

Name	Year	Type	Developer
MongoDB	**2008**	**Document**	**10Gen**
CouchDB	2005	Document	Apache
Cassandra	2008	Column Store	Apache
CouchBase	2011	Document	Couchbase
Riak/ redis	2009	Key-Value	Basho Technologies
SimpleDB	2007	Document	Amazon
BigTable	2015	Column Store	Google
Azure Cosmos DB	2017	Multi-Model	Microsoft

What is MongoDB?

- MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.
- Non relational DB, stores BSON documents.
- Schemaless: Two documents don't have the same schema.

Document Data Model

- A record in MongoDB is a Document
- Structure of key/value pairs
- Values may contain other documents, arrays and arrays of documents.

```
{
    _id: 1,
    firstname: "Josh",
    lastname: "Edward",
    email: "test@mim.edu",
    phones: ["6414511111", "6414512222"]
}
```

BSON

- BSON, short for Binary JSON, is a binary-encoded serialization of JSON-like documents.
- Both JSON and BSON support Rich Documents (embedding documents and arrays within other documents and arrays).
- BSON also contains extensions that allow representation of data types that are not part of the JSON spec. (For example, BSON has a BinData ObjectId, 64 bits Integers and Date type… etc)

BSON characteristics

- Lightweight
  - Keeping spatial overhead to a minimum is important for any data representation format, especially when used over the network.
- Traversable
  - BSON is designed to be traversed easily. This is a vital property in its role as the primary data representation for MongoDB.
- Efficient
  - Encoding data to BSON and decoding from BSON can be performed very quickly in most languages. For example, integers are stored as 32 (or 64) bit integers and they don't need to be parsed to and from text.

Non-Relational

- Scalability and Performance *(embedded data models reduces I/O activity on database system)*
- Depth of Functionality *(Aggregation framework, Text Search, Geospatial Queries)*
- To retains scalability
  - MongoDB **does not ~~support~~ favor Joins** between two collections *($lookup)*
  - **No relational algebra:** tables/columns/rows *(SQL)*
  - **No Transactions** across multiple collections *(Do it programmatically, documents can be accessed atomically) – Newer versions of MongoDB now support ACID transaction to an extent*

Schema

- By default, a collection does not require its documents to have the same schema, the documents in a single collection do not need to have the same set of fields and the data type for a field can differ across documents within a collection.
- Starting of MongoDB 3.2, you can enforce document validation rules for a collection during update and insert operations

Document Structure

- The value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents.

```
const doc = {
    _id: new ObjectID('5e44ab7638d4f738f05c57a8'),
    name: { first: "Josh", last: "Edward" },
    birth: new Date('Oct 31, 1979'),
    email: "test@mim.edu",
    phones: ["6414511111", "6414512222"]
}
```

```
{
  // This field holds a unique identifier for the document.
  // Otherwise, the MongoDB driver will generate a new one during insertion.
  _id: ObjectId("637bdef876543210fedcba"),
  // This field stores the person's name as an object with separate properties for first and last name.
  name: { "first": "Alice", "last": "Smith"},
  age: 30, // This field holds the person's age as a number.
  city: "New York", // This field stores the person's city of residence as a string.
  hobbies: ["reading", "hiking"],  //an array containing strings representing the person's hobbies.
  // This field is an array of embedded documents, each representing a product review.
  reviews: [
    {
      product: "Headphones",
      rating: 4.5,
      comment: "Great sound quality"
    },
    {
      product: "Laptop",
      rating: 3.8,
      comment: "Good performance, average battery life"
    }
  ]
};
```

General Rules

- Field names are strings.
- The field name `_id` is reserved for use as a primary key. It is immutable and always the first field in the document. It may contain values of any BSON data type, other than an array.
- The field names cannot start with the dollar sign ($) character and cannot contain the dot (.) character or `null`. Field names cannot be duplicated.
- The maximum BSON document size is 16 megabytes. *(To store documents larger than the maximum size, MongoDB provides the GridFS API)*

Setup

- Follow the link to install MongoDB
  - https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/
- Two ways to start your MongoDB
  - as a Windows Service
    1. From the Services console, locate the MongoDB service.
    2. Right-click on the MongoDB service and click **Stop** (or **Pause**).

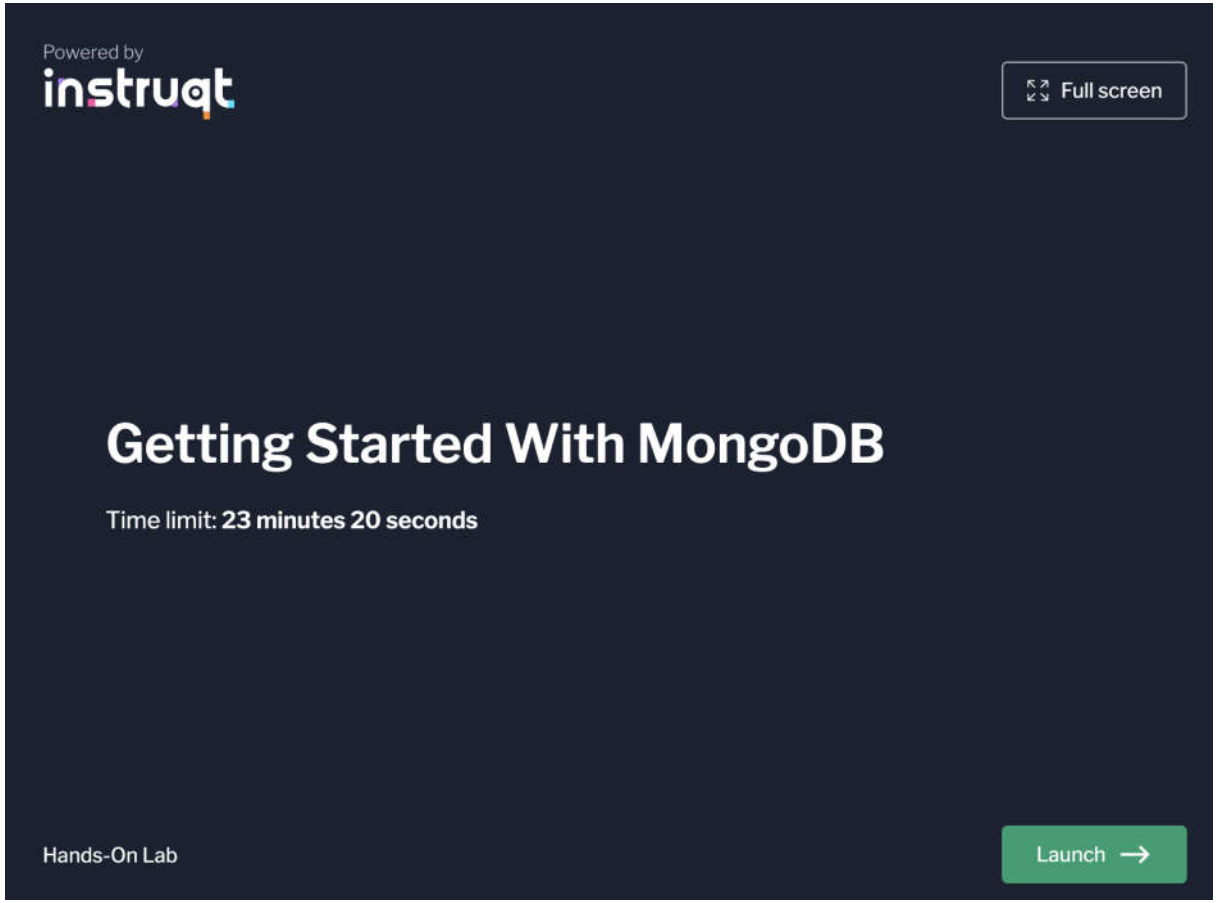
  - from the Command Interpreter
    1. Create database directory - C:/data/db
    2. Start your MongoDB database.
       - "C:\Program Files\MongoDB\Server\4.2\bin\mongod.exe" --dbpath="c:\data\db"
    3. Connect to MongoDB
       - "C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe"

https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-os-x/

https://www.mongodb.com/docs/manual/tutorial/getting-started/

Interactive Tutorial

You do not need to install anything. Click the Launch button of the in-browser Integrated Development Environment to start the tutorial.



To find MongoDB version

```
bright~$mongod --version
db version v7.0.2
Build Info: {
   "version": "7.0.2",
   "gitVersion": "02b3c655e1302209ef046da6ba3ef6749dd0b62a",
   "modules": [],
   "allocator": "system",
   "environment": {
      "distarch": "x86_64",
      "target_arch": "x86_64"
   }
}
```

To run MongoDB (i.e. the mongod process) **as a macOS service**, run:

```
bright~$brew services start mongodb-community@7.0
==> Successfully started `mongodb-community` (label: homebrew.mxcl.mongodb-commu
bright~$
```

To stop a mongod running as a macOS service:

```
bright~$sudo brew services stop mongodb-community
Stopping `mongodb-community`... (might take a while)
==> Successfully stopped `mongodb-community` (label: homebrew.mxcl.mongodb-commu
bright~$
```

bright~$mongosh
Current Mongosh Log ID: 66414d45a3647e65c6e8832e
Connecting to:
mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.5
Using MongoDB: 7.0.8
Using Mongosh: 2.2.5

For mongosh info see: **https://docs.mongodb.com/mongodb-shell/**

The server generated these startup warnings when booting
2024-05-12T18:05:36.952-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2024-05-12T18:05:36.952-05:00: You are running this process as the root user, which is not recommended

---

test>

Show all databases

```
test> show dbs
admin     40.00 KiB
books     40.00 KiB
config    60.00 KiB
elibrary  72.00 KiB
local    112.00 KiB
test      40.00 KiB
test>
```

Create a new Database

```
test> use blog
switched to db blog
```

```
blog> show dbs
admin     40.00 KiB
books     40.00 KiB
config    96.00 KiB
elibrary  72.00 KiB
local    112.00 KiB
test      40.00 KiB
```

blog database is not created

```
blog> db.createCollection("posts")
{ ok: 1 }
blog> show dbs
admin     40.00 KiB
blog       8.00 KiB
books     40.00 KiB
config    96.00 KiB
elibrary  72.00 KiB
local    112.00 KiB
test      40.00 KiB
blog>
```

blog database is created

Drop a Database

```
blog> db.dropDatabase()
{ ok: 1, dropped: 'blog' }
```

```
blog> show dbs
admin     40.00 KiB
books     40.00 KiB
config    96.00 KiB
elibrary  72.00 KiB
local    112.00 KiB
test      40.00 KiB
blog>
```

Create a new Database

```
blog> show dbs
admin     40.00 KiB
books     40.00 KiB
config    96.00 KiB
elibrary  72.00 KiB
local    112.00 KiB
test      40.00 KiB
blog>
```

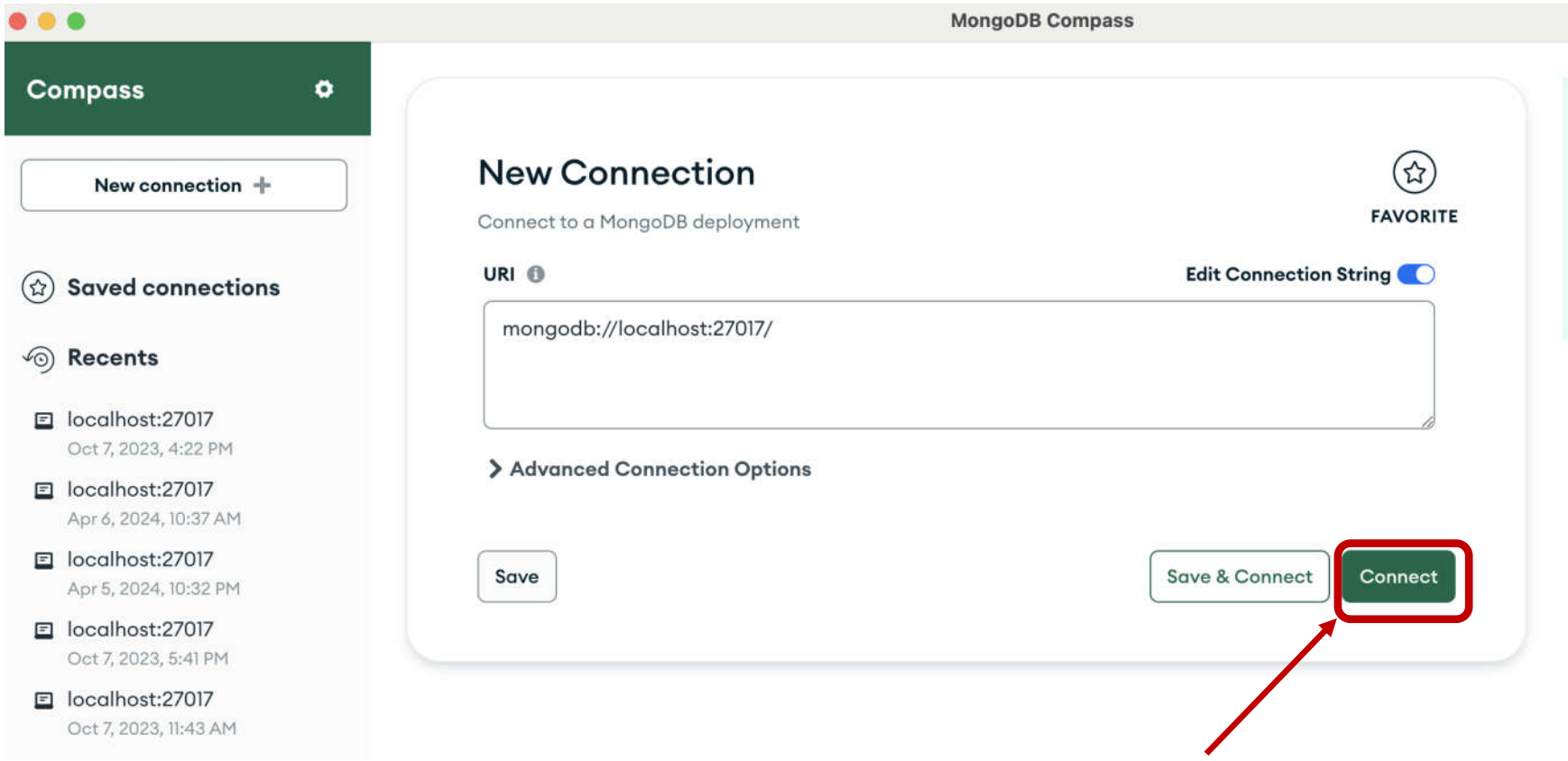
```
blog> db.posts.insertOne({
... title: "Post Title#1",
... body: "Body of post#1",
... category: "News",
... likes: 1,
... tags: ["news", "events"],
... date: Date()
... })
{
  acknowledged: true,
  insertedId: ObjectId('664181faa3647e65c6e88330')
}
blog>
```

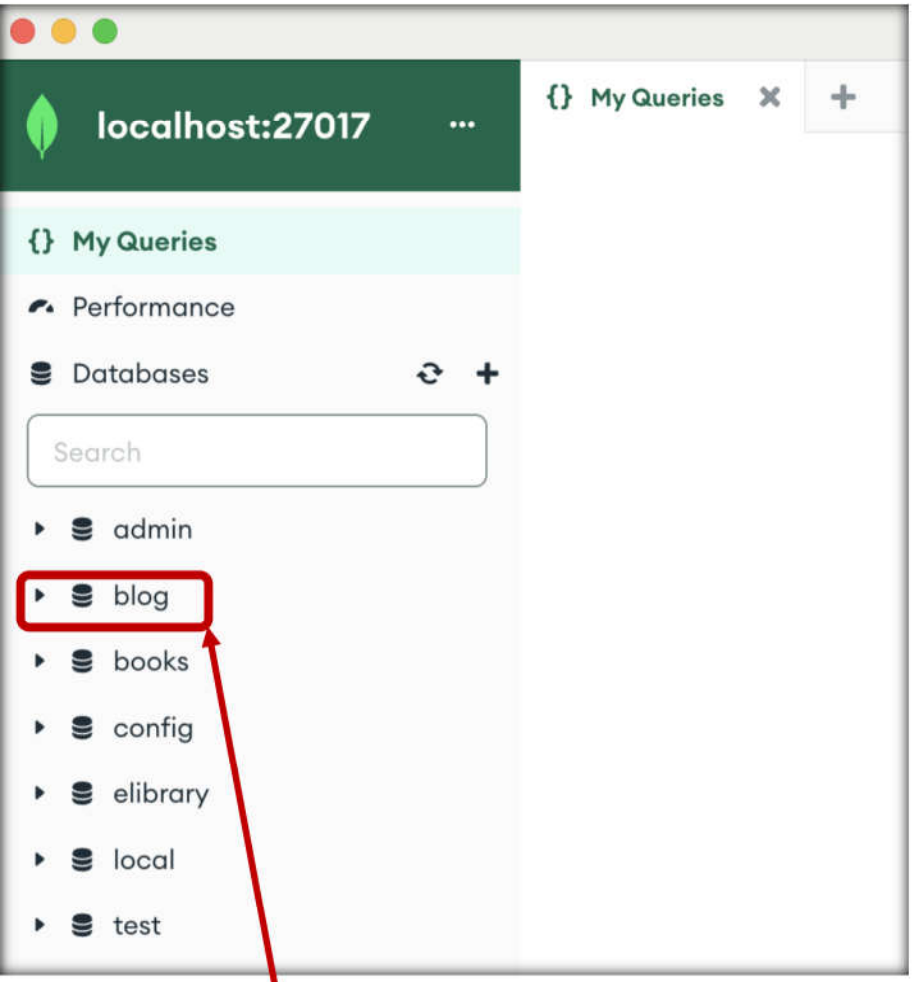
The insertOne() operation creates both the database blog and the collection posts if they do not already exist.

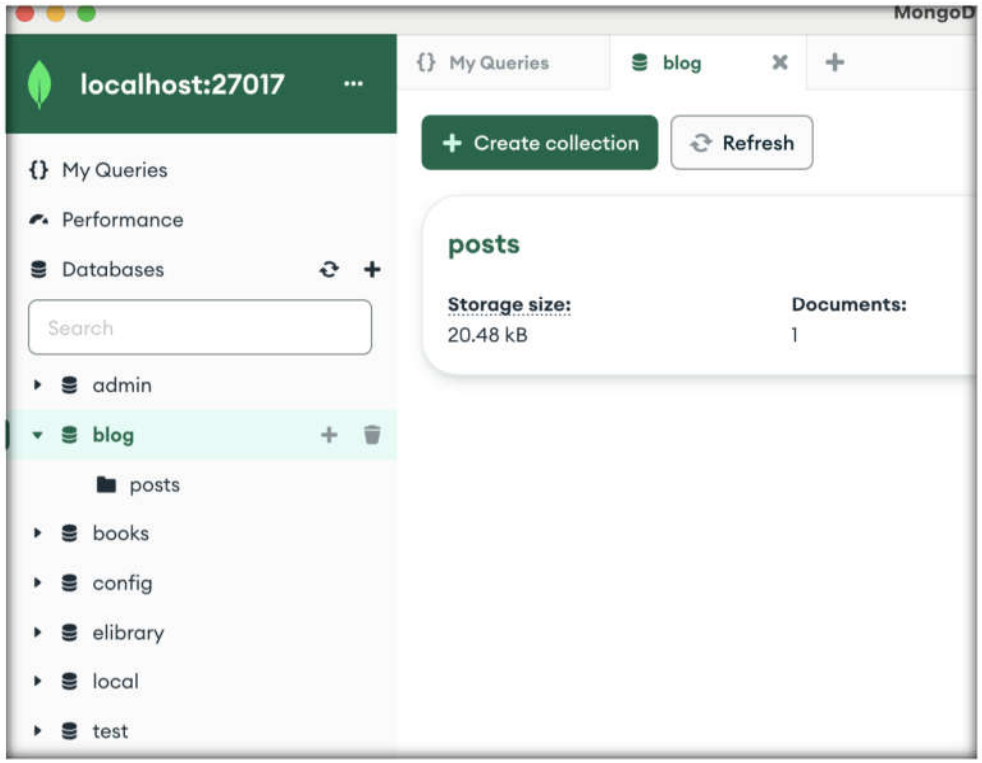
MongoDB Compass (GUI for MongoDB)

https://downloads.mongodb.com/compass/mongodb-compass-1.43.0-darwin-arm64.dmg
https://downloads.mongodb.com/compass/mongodb-compass-1.43.0-darwin-x64.dmg
https://downloads.mongodb.com/compass/mongodb-compass-1.43.0-win32-x64.msi

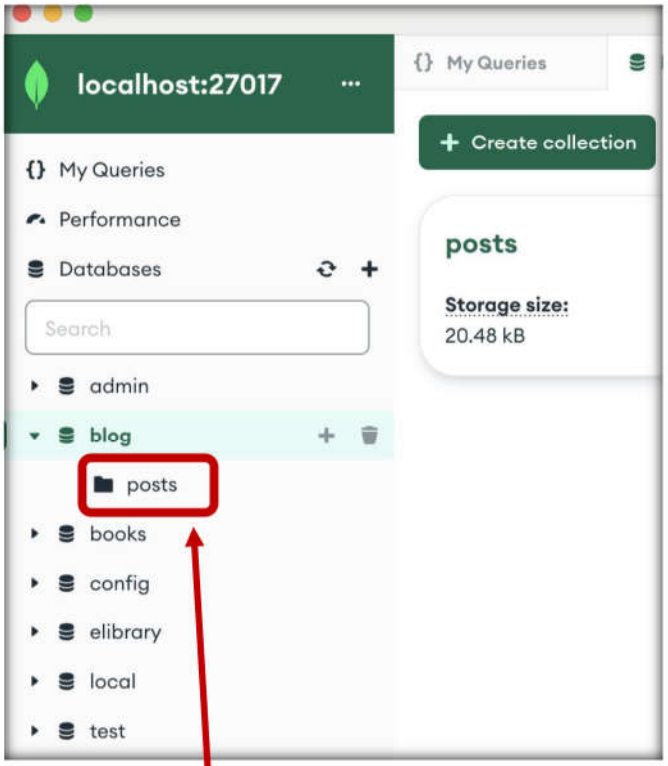
Open MongoDB Compass



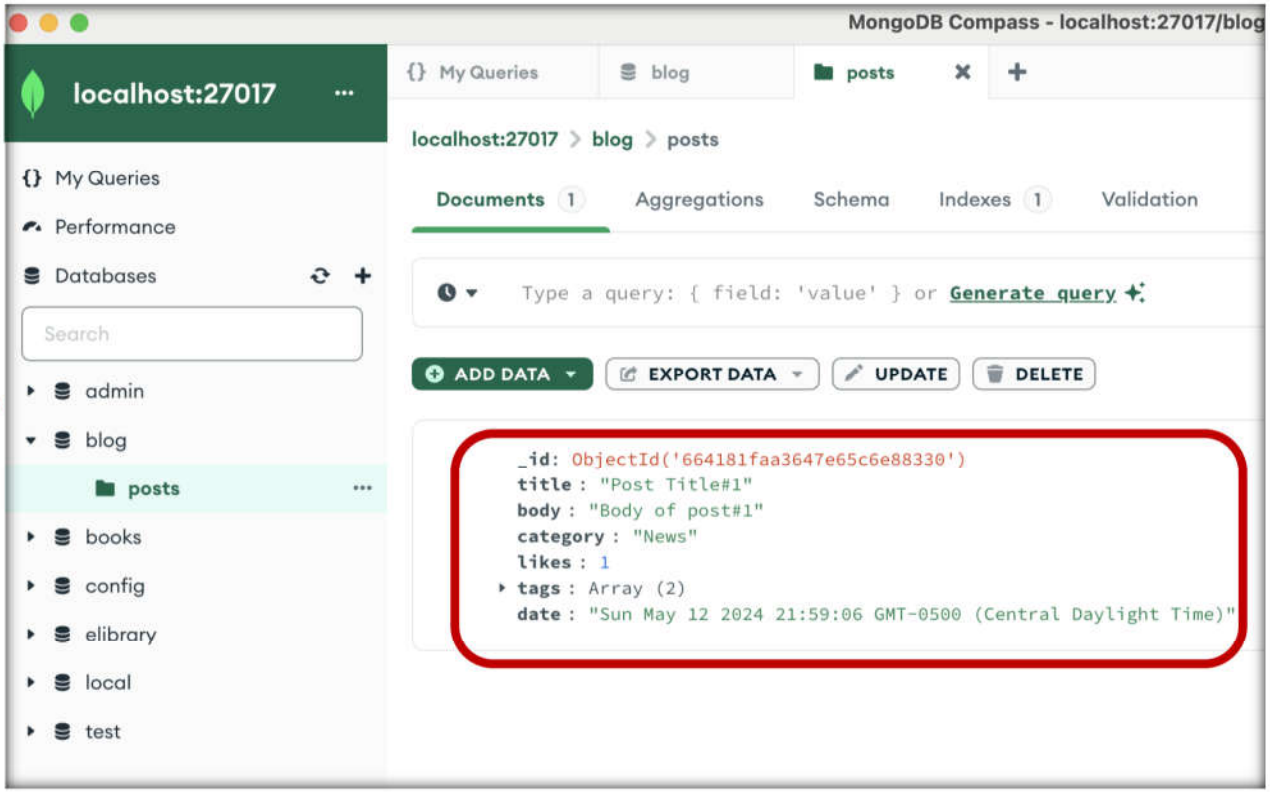

localhost:27017
My Queries
Performance
Databases
Search
admin
blog
books
config
elibrary
local
test
My Queries


localhost:27017
My Queries
Performance
Databases
Search
admin
blog
posts
books
config
elibrary
local
test
My Queries
blog
Create collection
Refresh
posts
Storage size:
20.48 kB
Documents:
1

Click **blog**


localhost:27017
My Queries
Performance
Databases
Search
admin
blog
posts
books
config
elibrary
local
test
My Queries
Create collection
posts
Storage size:
20.48 kB

Click **posts**


MongoDB Compass - localhost:27017/blog
localhost:27017
My Queries
Performance
Databases
Search
admin
blog
posts
books
config
elibrary
local
test
My Queries
blog
posts
localhost:27017 > blog > posts
Documents 1
Aggregations
Schema
Indexes 1
Validation
Type a query: { field: 'value' } or Generate query
ADD DATA
EXPORT DATA
UPDATE
DELETE
_id: ObjectId('664181faa3647e65c6e88330')
title : "Post Title#1"
body : "Body of post#1"
category : "News"
likes : 1
tags : Array (2)
date : "Sun May 12 2024 21:59:06 GMT-0500 (Central Daylight Time)"

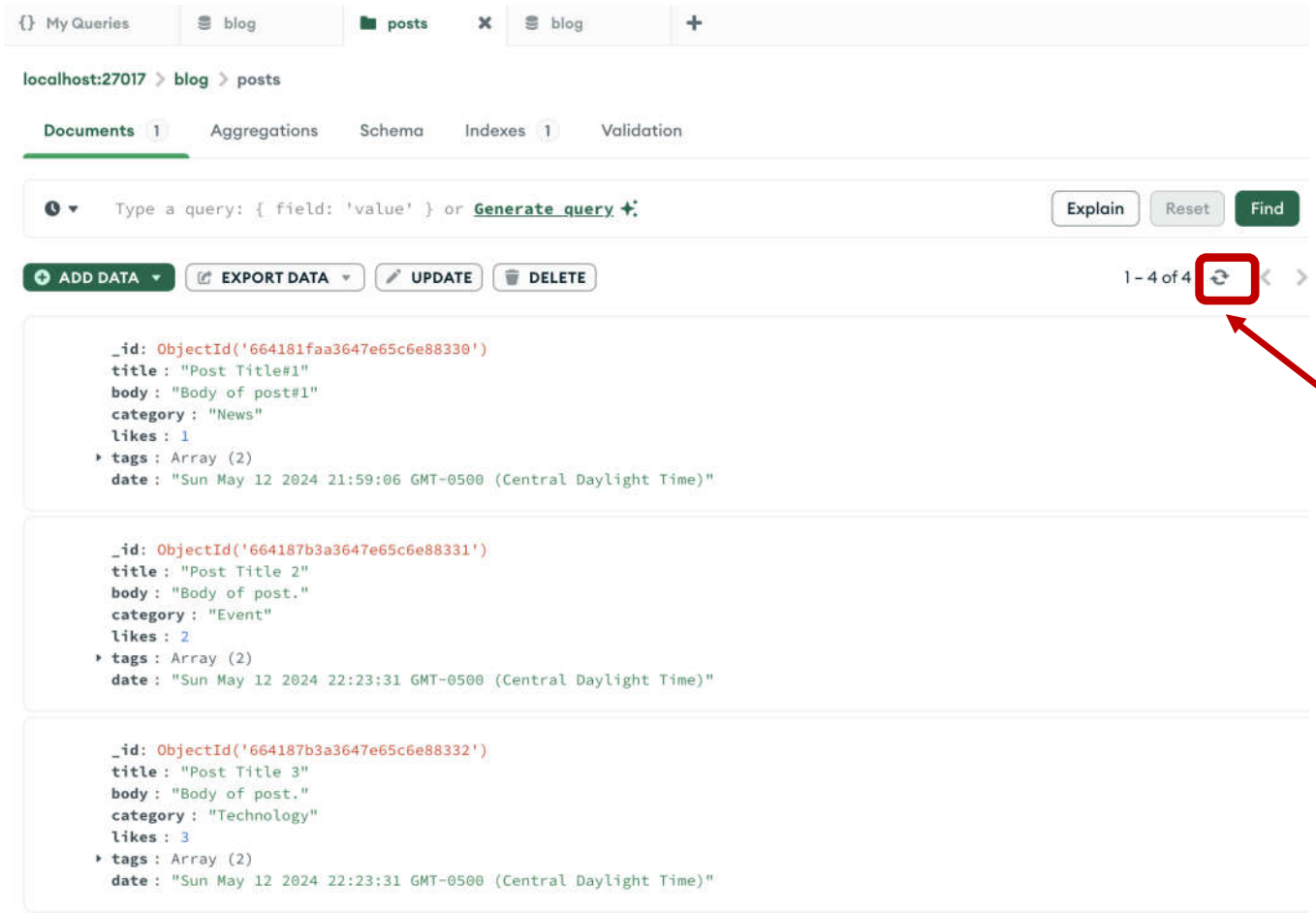
Command to find all documents in a collection

```
blog> db.getCollection("posts").find()
[
  {
    _id: ObjectId('664181faa3647e65c6e88330'),
    title: 'Post Title#1',
    body: 'Body of post#1',
    category: 'News',
    likes: 1,
    tags: [ 'news', 'events' ],
    date: 'Sun May 12 2024 21:59:06 GMT-0500 (Central Daylight Time)'
  }
]
blog>
```

Alternative way

```
blog> db.posts.find()
```

```
blog> db.posts.insertMany([
...   {
...     title: "Post Title 2",
...     body: "Body of post.",
...     category: "Event",
...     likes: 2,
...     tags: ["news", "events"],
...     date: Date()
...   },
...   {
...     title: "Post Title 3",
...     body: "Body of post.",
...     category: "Technology",
...     likes: 3,
...     tags: ["news", "events"],
...     date: Date()
...   },
...   {
...     title: "Post Title 4",
...     body: "Body of post.",
...     category: "Event",
...     likes: 4,
...     tags: ["news", "events"],
...     date: Date()
...   }])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('664187b3a3647e65c6e88331'),
    '1': ObjectId('664187b3a3647e65c6e88332'),
    '2': ObjectId('664187b3a3647e65c6e88333')
  }
}
blog>
```



Click Refresh icon

db.collection.find(<query>, <projection>, <options>)

```
blog> db.posts.find(
... {likes: {$gt: 0}},
... { category: 1, title: 1 },
... { sort: { category: -1 } }
... )
[
 {
  _id: ObjectId('664187b3a3647e65c6e88332'),
  title: 'Post Title 3',
  category: 'Technology'
 },
 {
  _id: ObjectId('664181faa3647e65c6e88330'),
  title: 'Post Title#1',
  category: 'News'
 },
 {
  _id: ObjectId('664187b3a3647e65c6e88331'),
  title: 'Post Title 2',
  category: 'Event'
 },
 {
  _id: ObjectId('664187b3a3647e65c6e88333'),
  title: 'Post Title 4',
  category: 'Event'
 }
]
blog>
```

searches for documents where the likes field is greater than ($gt) 0

indicate which fields you want to include in the retrieved documents

-1 for descending order and 1 for ascending order

To find documents where "likes" is greater than 0 and the "tags" array contains "news":

```
blog> db.posts.find(
... { likes: { $gt: 0 }, tags: { $in: ["news"]} },
... {},
... {}
... )
[
 {
  _id: ObjectId('664181faa3647e65c6e88330'),
  title: 'Post Title#1',
  body: 'Body of post#1',
  category: 'News',
  likes: 1,
  tags: [ 'news', 'events' ],
  date: 'Sun May 12 2024 21:59:06 GMT-0500 (Central Daylight Time)'
 },
 {
  _id: ObjectId('664187b3a3647e65c6e88331'),
  title: 'Post Title 2',
  body: 'Body of post.',
  category: 'Event',
  likes: 2,
  tags: [ 'news', 'events' ],
  date: 'Sun May 12 2024 22:23:31 GMT-0500 (Central Daylight Time)'
 },
 ...
]
blog>
```

findOne()

```
blog> db.getCollection("posts").findOne()
{
  _id: ObjectId('664181faa3647e65c6e88330'),
  title: 'Post Title#1',
  body: 'Body of post#1',
  category: 'News',
  likes: 1,
  tags: [ 'news', 'events' ],
  date: 'Sun May 12 2024 21:59:06 GMT-0500 (Central Daylight Time)'
}
```

db.collection.findOne(query, projection, options)

```
blog> db.posts.findOne(
... {likes: {$gt: 0}},
... {category: 0},
... { sort: { category: -1 } }
... )
{
  _id: ObjectId('664187b3a3647e65c6e88332'),
  title: 'Post Title 3',
  body: 'Body of post.',
  likes: 3,
  tags: [ 'news', 'events' ],
  date: 'Sun May 12 2024 22:23:31 GMT-0500 (Central Daylight
Time)'
}
blog>
```

Ref: https://www.mongodb.com/docs/manual/reference/method/js-collection/

Exit from mongosh

blog> exit

Stop Mongodb Community

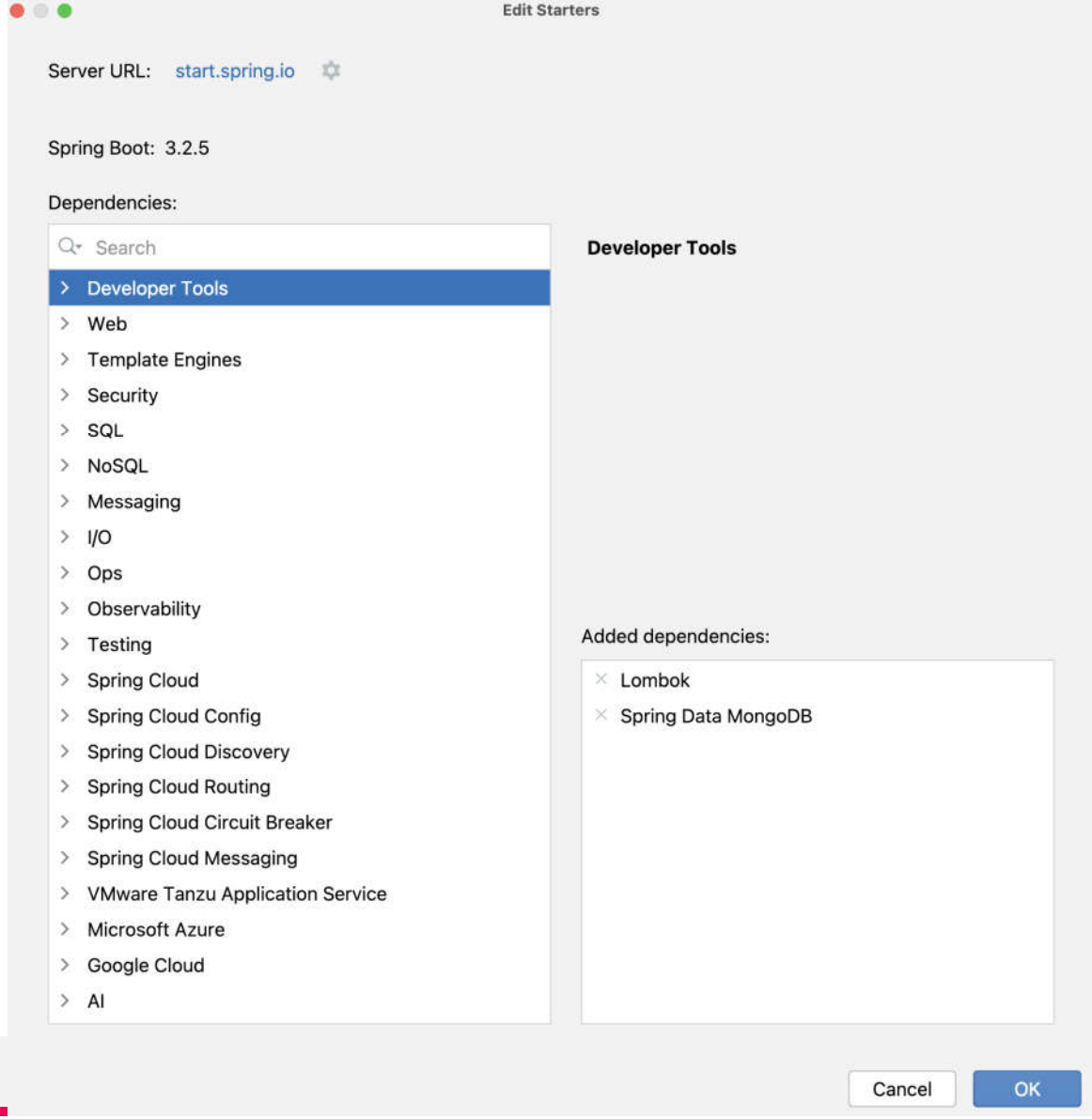
bright~$brew services stop mongodb-community
Stopping `mongodb-community`... (might take a while)
==> **Successfully stopped `mongodb-community` (label: homebrew.mxcl.mongodb-com**

bright~$

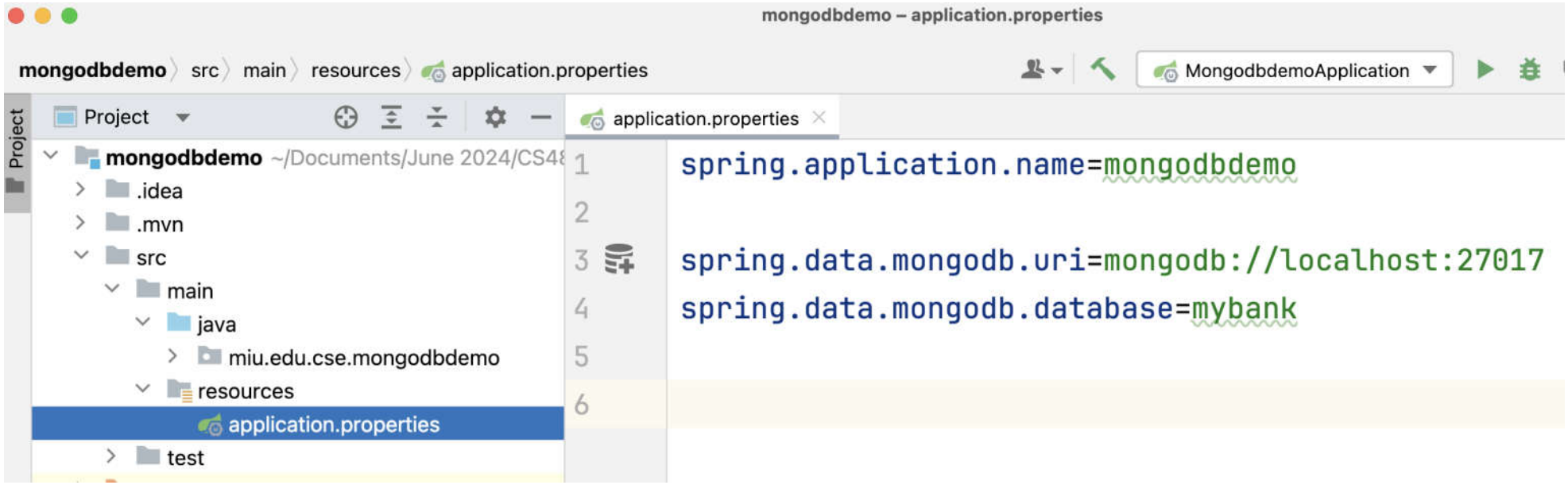
MongoDB Application Development

- Demo/Exercise:
- Using Spring Boot, create a new CLI Application project, adding as dependencies – Spring Data MongoDB
- Implement Code to perform Data Access operations with MongoDB database

Add dependencies



Application.properties



Create Model classes

```java
@Data
@AllArgsConstructor
@Document(collection = "accounts")
public class Account {
    private String id;
    @Field(value = "account_name")
    private String name;
    // Optional: Account type (e.g., bo
    @Field(value = "account_type")
    private String type;
}
```

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
@Document(collection = "users")
public class User {
    private String id;
    private String username;
    private String password;
    @DBRef
    private List<Account> accounts;  // List of referenced accounts
    no usages
    public User(String id, String username, String password) {
        this.id = id;
        this.username = username;
        this.password = password;
    }
}
```

A User has many Accounts.

Create Repositories for User and Account

```
8  public interface UserRepository extends MongoRepository<User, String> {
9  //    Optional<User> findUserByUsername(String name);
       1 usage
10     Optional<User> findByUsername(String name);
       1 usage
11     @Query(value = "{username: '?0'}")
12     Optional<User> findUserBasedOnName(String name);
13 }
```

```
11 public interface AccountRepository extends MongoRepository<Account, String> {
       1 usage
12     @Query(value = "{account_name: '?0', account_type:  '?1'}")
13     List<Account> findUserAccountBasedOnNameType(String name, String type);
14 }
```


Ref: https://www.devglan.com/spring-boot/spring-data-mongodb-queries

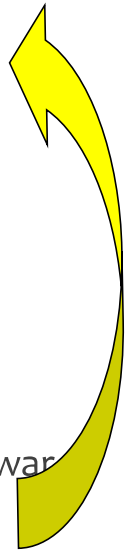
```
CommandLineRunner commandLineRunner() {
    return args -> {
        User user1 = new User( id: "usr_1",  username: "john.doe",  password: "password123");
        Account account1 = new Account( id: "acc_1",  name: "Savings Account",  type: "bank");
        Account account2 = new Account( id: "acc_2",  name: "Social Media Profile",  type: "social");
        user1.setAccounts(List.of(account1, account2));
        accountRepository.save(account1);
        accountRepository.save(account2);
        System.out.println(userRepository.save(user1));
        User user2 = new User( id: "usr_2",  username: "brightlsom",  password: "password456");
        Account account3 = new Account( id: "acc_3",  name: "Savings Account",  type: "bank");
        Account account4 = new Account( id: "acc_4",  name: "Current Account",  type: "current");
        user2.setAccounts(List.of(account3, account4));
        accountRepository.save(account3);
        accountRepository.save(account4);
        System.out.println(userRepository.save(user2));
        userRepository.findAll() List<User>
                .stream() Stream<User>
                .forEach(System.out::println);
        System.out.println(userRepository.findByUsername("john.doe"));
        System.out.println(userRepository.findUserBasedOnName("john.doe"));
        System.out.println(accountRepository.findUserAccountBasedOnNameType( name: "Savings Account",  type: "bank"));
    };
}
```

Connecting the Parts of Knowledge With the Wholeness of Knowledge

Overview of NoSQL Databases and MongoDB

1. NoSQL Databases offer relatively better performance than Relational databases, but they trade-off some Data integrity features found in relational (SQL) databases.

2. MongoDB is arguably the predominant document-oriented NoSQL database currently in use in many real-world, high-performant applications.



---

3. **Transcendental consciousness** is the underlying basis of all levels of creation.

4. **Impulses within the Transcendental Field:** The performance benefit of NoSQL databases such as MongoDB, forms the basis of several high-throughput softwar applications, and this arises as an impulse of the Transcendental Field.

5. **Wholeness moving within itself**: In Unity Consciousness, one directly perceives that all expressions and levels of creation are nothing more than one's own Self – pure consciousness.