MongoDB – Introduction

CS477 – Server-side Programming

Maharishi International University

Department of Computer Science

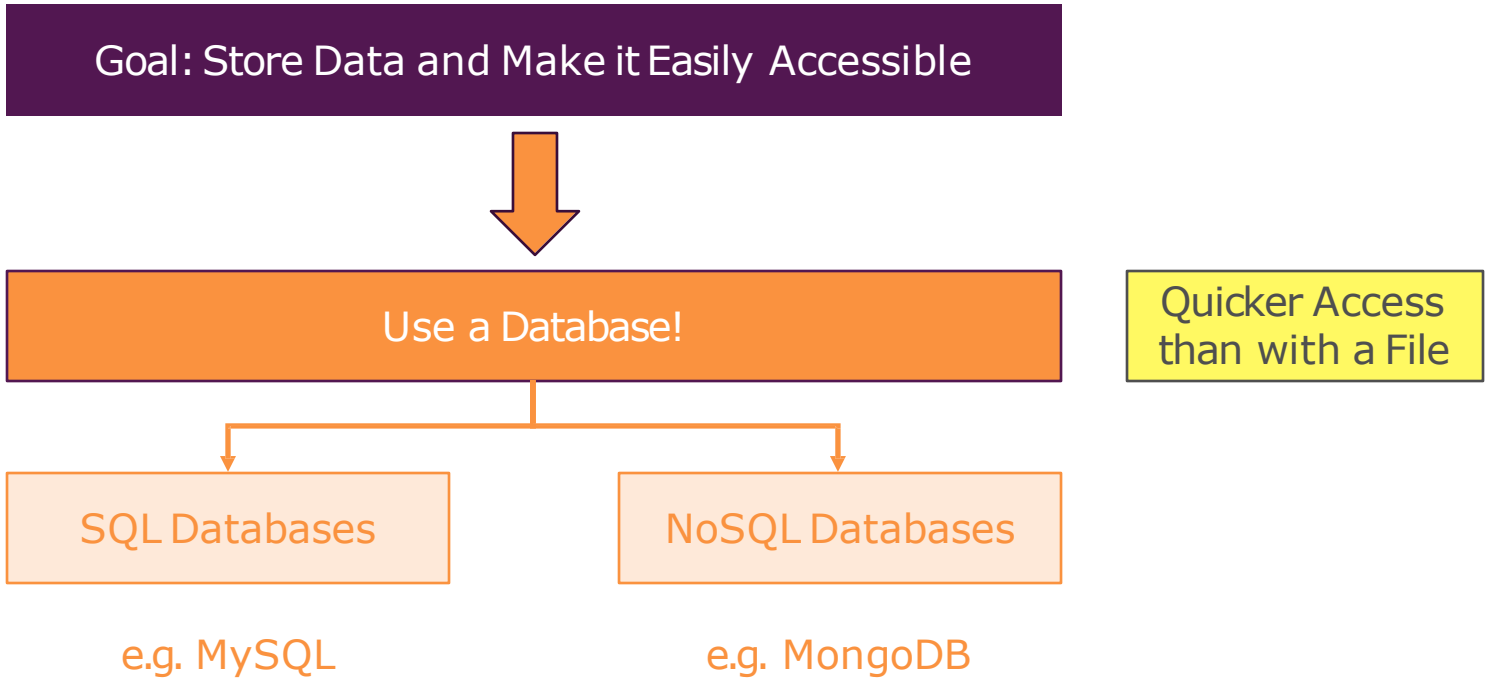
M.S. Thao Huy Vu

Maharishi International University - Fairfield, Iowa

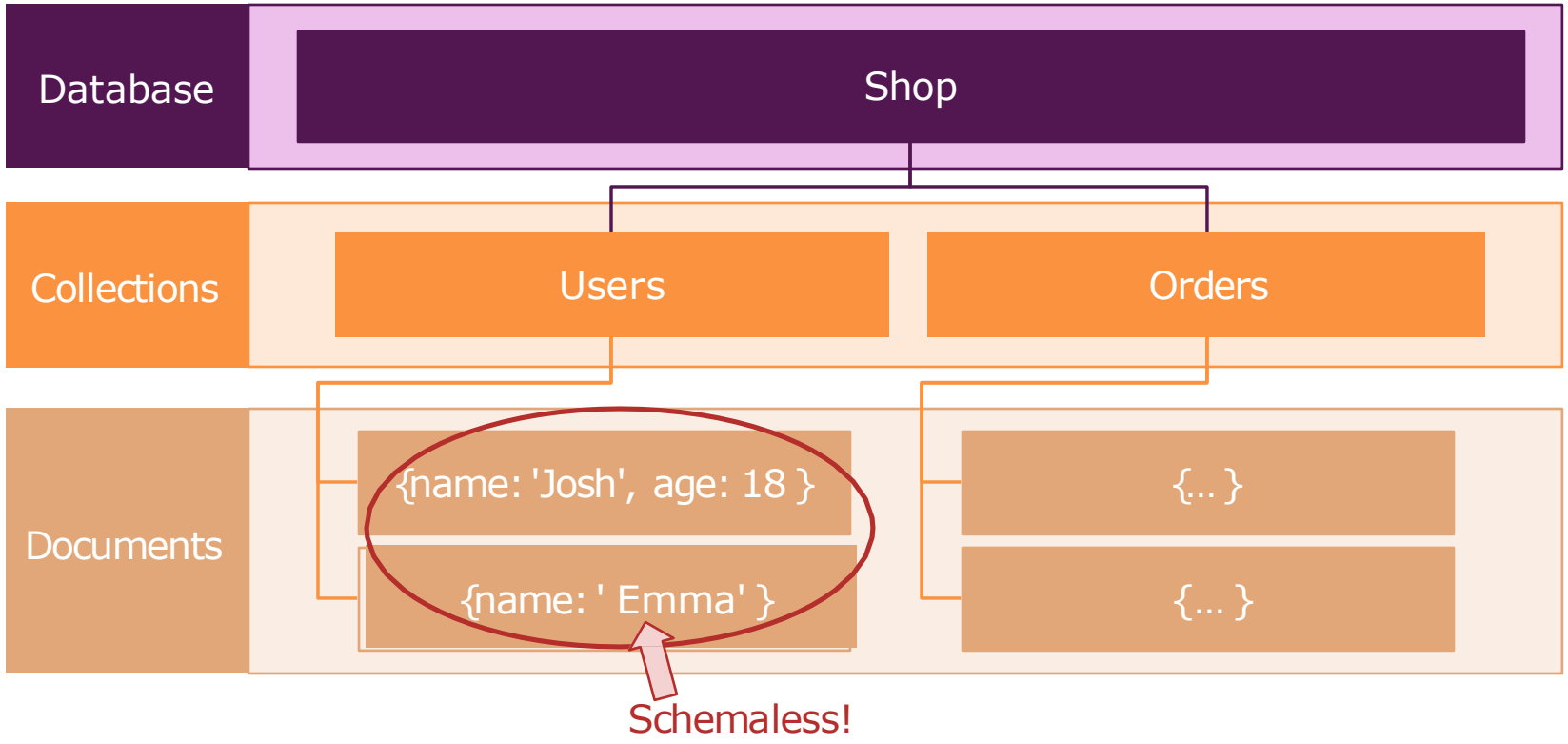
Main Points

- NoSQL: MongoDB
- Mongodb Atlas: Cloud Database
- NodeJS driver for MongoDB
- Operations: Connection; Create, Read, Update, Delete (CRUD)

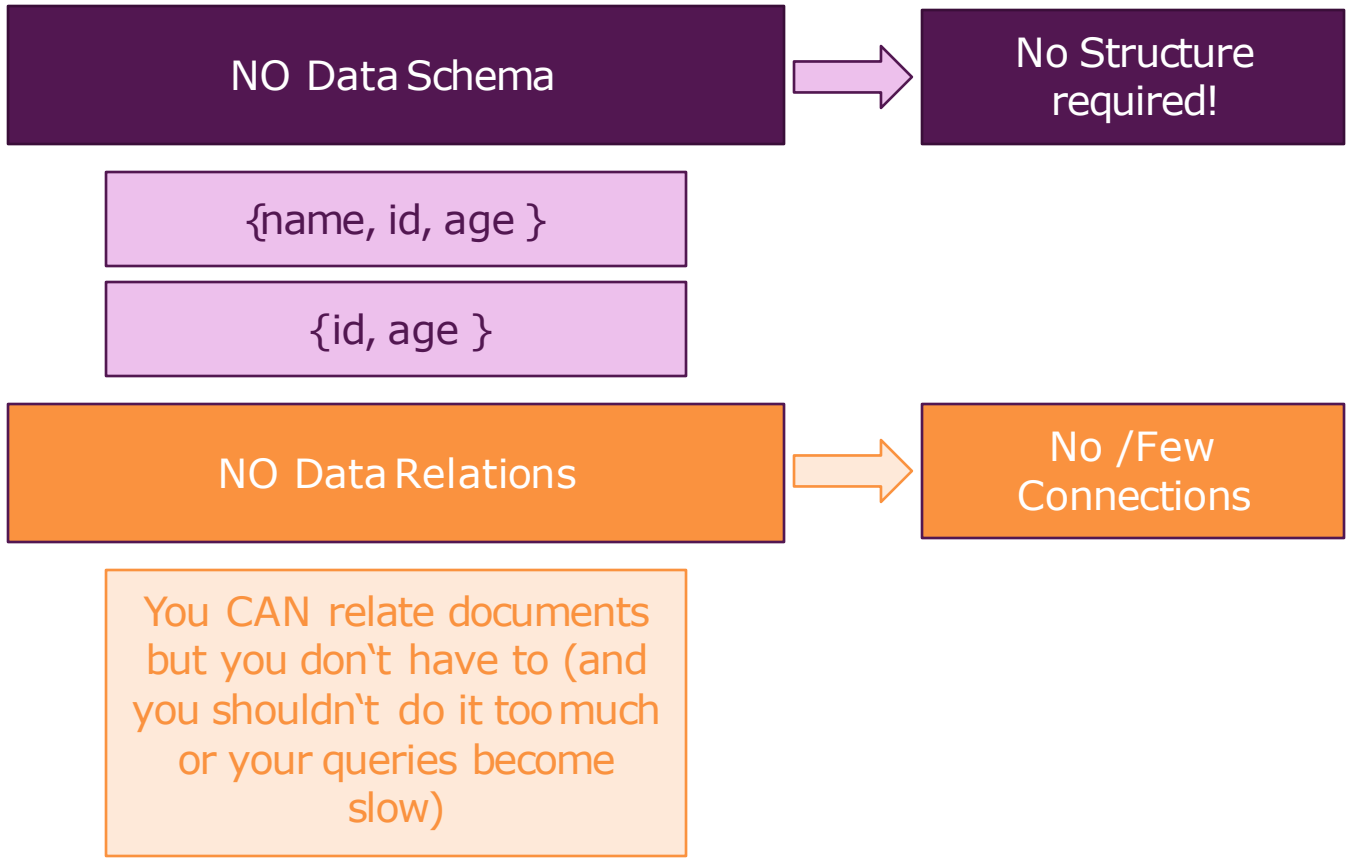
Database


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

NoSQL

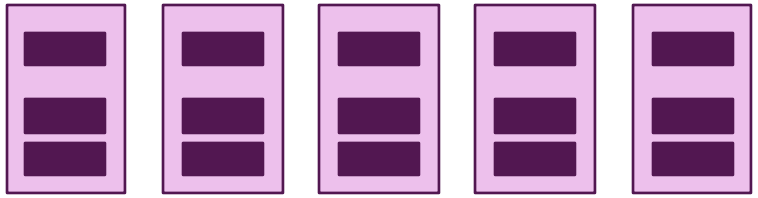
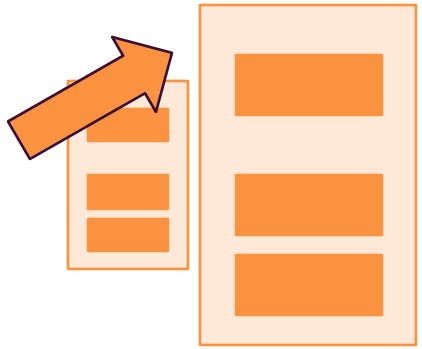

Database
Shop
Collections
Users
Orders
Documents
{name: 'Josh', age: 18 }
{name: ' Emma' }
{... }
{... }
Schemaless!

NoSQL Characteristics



Scaling

While horizontal scaling refers to adding additional nodes, vertical scaling describes adding more power to your current machines. For instance, if your server requires more processing power, vertical scaling would mean upgrading the CPUs. You can also vertically scale the memory, storage, or network speed

Horizontal Scaling	Vertical Scaling
	
Add More Servers (and merge Data into one Database)	Improve Server Capacity /Hardware

NoSQL Revolution

noSQL, horizontal scalling is bettet

- 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	2009	Key-Value	Basho Technologies
SimpleDB	2007	Document	Amazon
BigTable	2015	Column Store	Google
Azure Cosmos DB	2017	Multi-Model	Microsoft

NOSQL Database Types

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

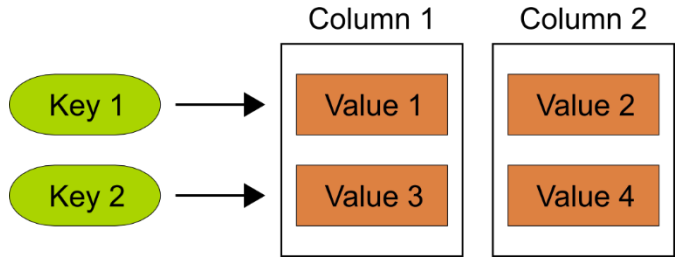
Key-Value Stores

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
             level: 5,
             group: "dev"
           }
}
```

Embedded sub-document

Embedded sub-document

Document Databases



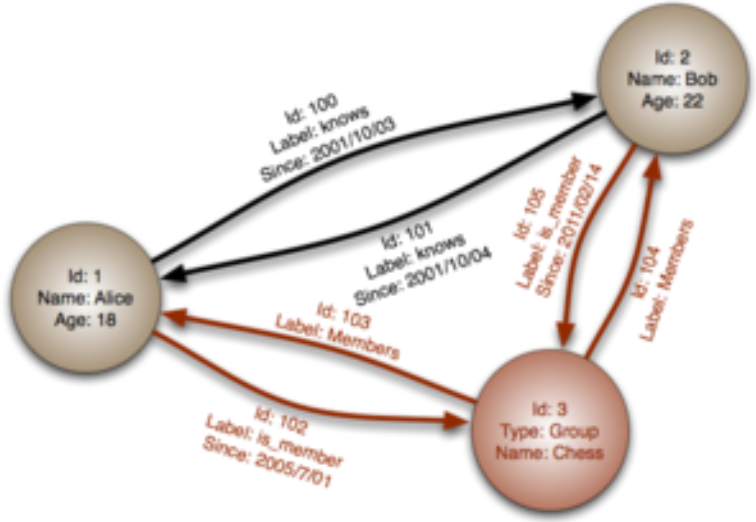
Column Family Stores

Key-Value pairs in hash table, always unique key. Logical group of keys are called: buckets

Document Databases uses Key-Value pairs in a document (JSON, BSON)

Column Stores data is stored in cells that are grouped in columns of data rather than rows (unlimited columns)

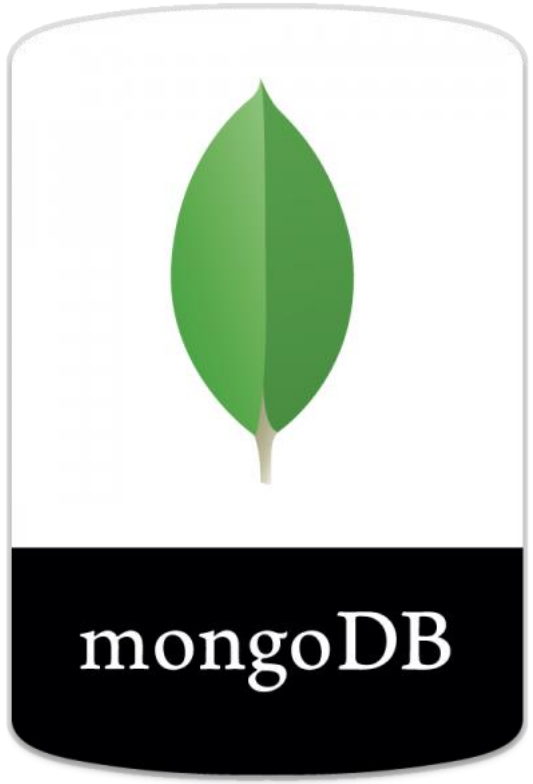
Graph Databases, uses flexible graphical representation (edges and nodes) instead of k/v pairs. Index free. Very fast for associative data sets and maps.



Graph Databases

MongoDB

- Open-source document database
- High performance, high availability, and automatic scaling.
- Non relational DB, stores BSON documents.
- Schema-less: Two documents don't have the same schema.



Document Data Model

- A record in MongoDB is a Document
- Structure of key/value pairs
- A value can be a document, an array, or an array 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 Joins** between two collections *($loopup)*
  - **No relational algebra:** tables/columns/rows *(SQL)*
  - **No Transactions** across multiple collections *(Do it programmatically, documents can be accessed atomically)*

Schema

- Structure of a collection
- Documents can have different schema
- Possible to add validation rules for insert or update (version >= 3.2)

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"]
}
```

Setup

- Download MongoDB Enterprise Version:
  - https://www.mongodb.com/try/download/enterprise
- Install MongoDB Enterprise: Find the right manual based on your OS
  - https://docs.mongodb.com/manual/administration/install-enterprise/

- Two ways to start your MongoDB on Windows. For Mac OS, follow the installation link.
  - 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**).

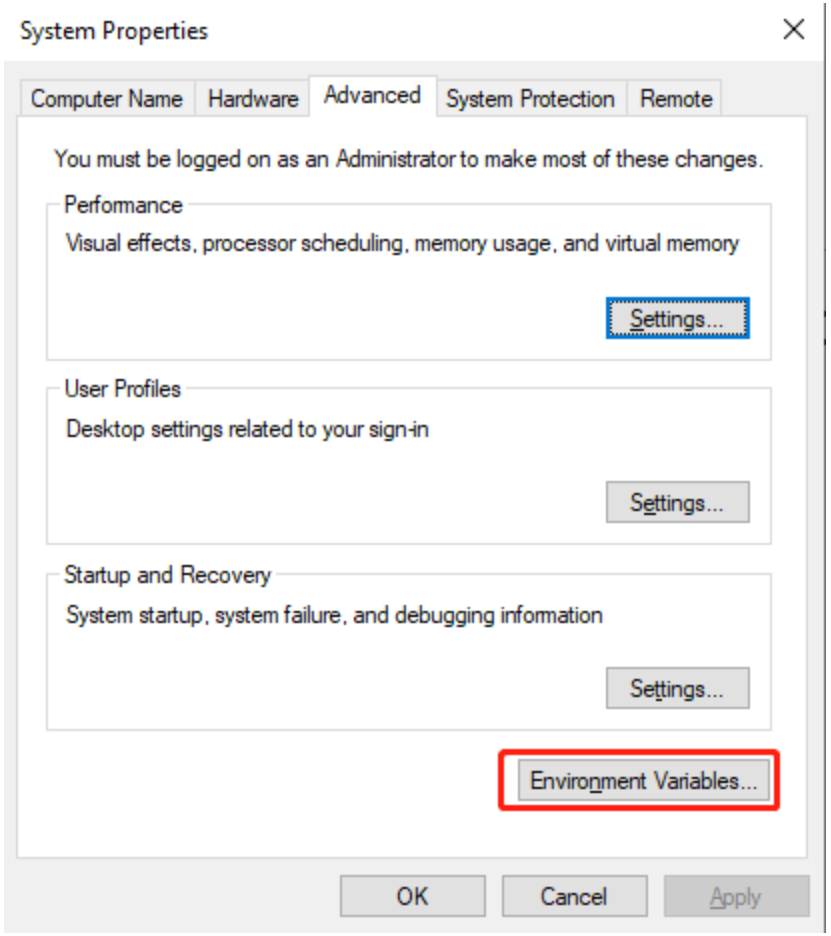
  - from the Command Interpreter
    1. Create database directory - C:/data/db
    2. Start your MongoDB database.
       - "C:\Program Files\MongoDB\Server\4.4\bin\mongod.exe" --dbpath="c:\data\db"
    3. Connect to MongoDB
       - "C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe"

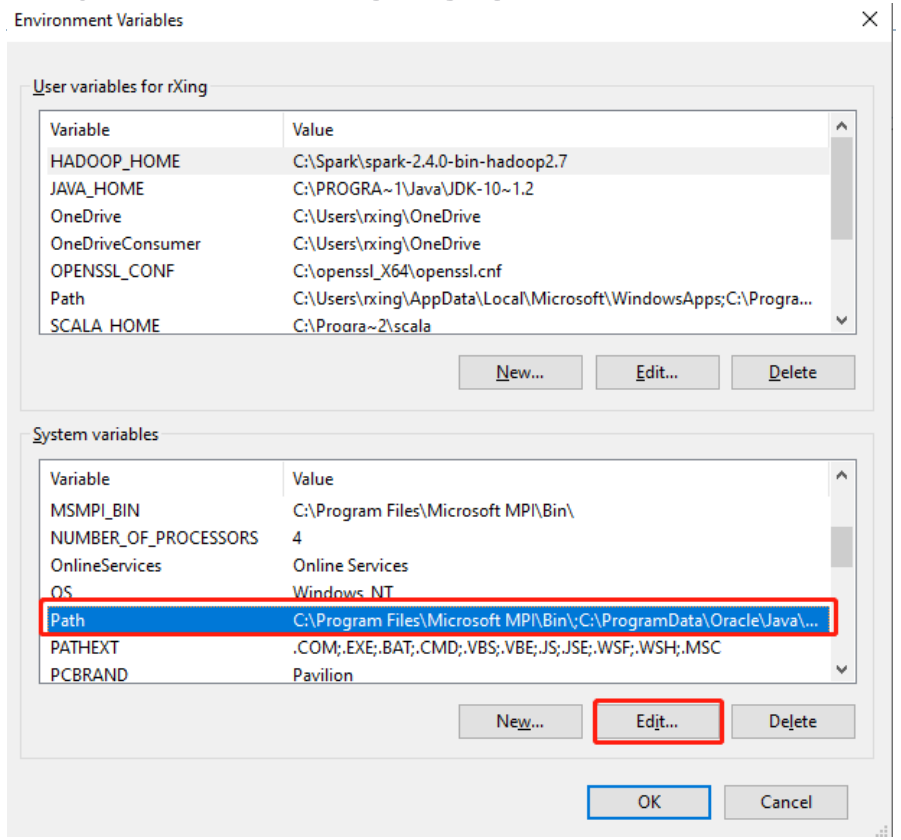
Setup Testing

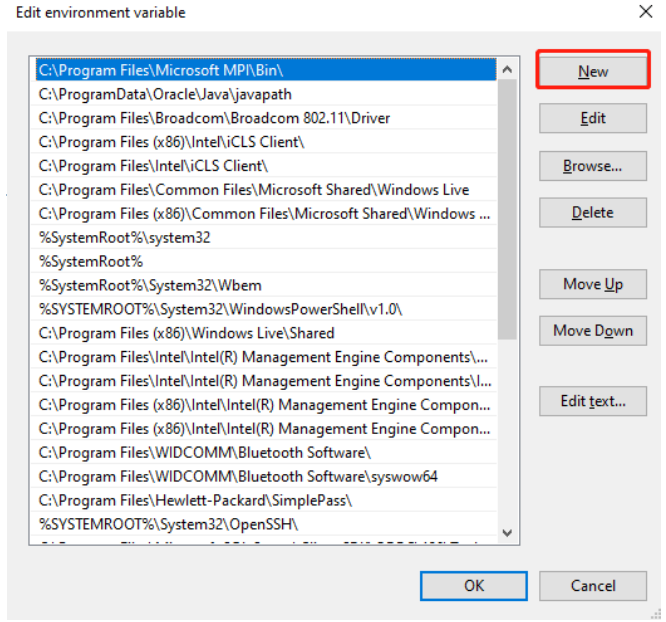
- Test if MongoDB installed and started successfully
  - `mongod –version`
  - `mongo –version`
  - `mongo`

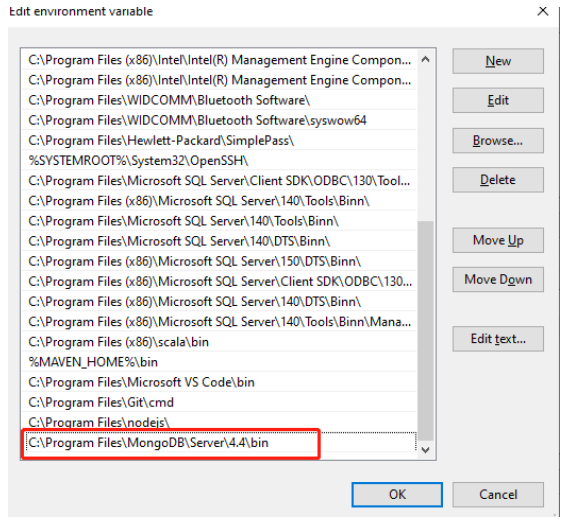
```
C:\WINDOWS\System32>mongod -version
db version v4.2.3
git version: 6874650b362138df74be53d366bbefc321ea32d4
allocator: tcmalloc
modules: enterprise
build environment:
    distmod: windows-64
C:\WINDOWS\System32>mongo -version
MongoDB shell version v4.2.3
git version: 6874650b362138df74be53d366bbefc321ea32d4
allocator: tcmalloc
modules: enterprise
build environment:
    distmod: windows-64
C:\WINDOWS\System32>mongo
MongoDB shell version v4.2.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4000d463-492b-4afb-ad95-2e0e12980a89") }
MongoDB server version: 4.4.6
WARNING: shell and server versions do not match
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
2021-06-18T12:10:35.213-0500 I  STORAGE  [main] In File::open(), CreateFileW for 'Z:\\.mongorc.js'
m cannot find the path specified.
Server has startup warnings:
{"t":{"$date":"2021-06-18T11:50:46.786-05:00"},"s":"W",  "c":"CONTROL",  "id":22120,  "ctx":"init
ss control is not enabled for the database. Read and write access to data and configuration is unr
artupWarnings"]}
MongoDB Enterprise >
```

Set up Environment Variables







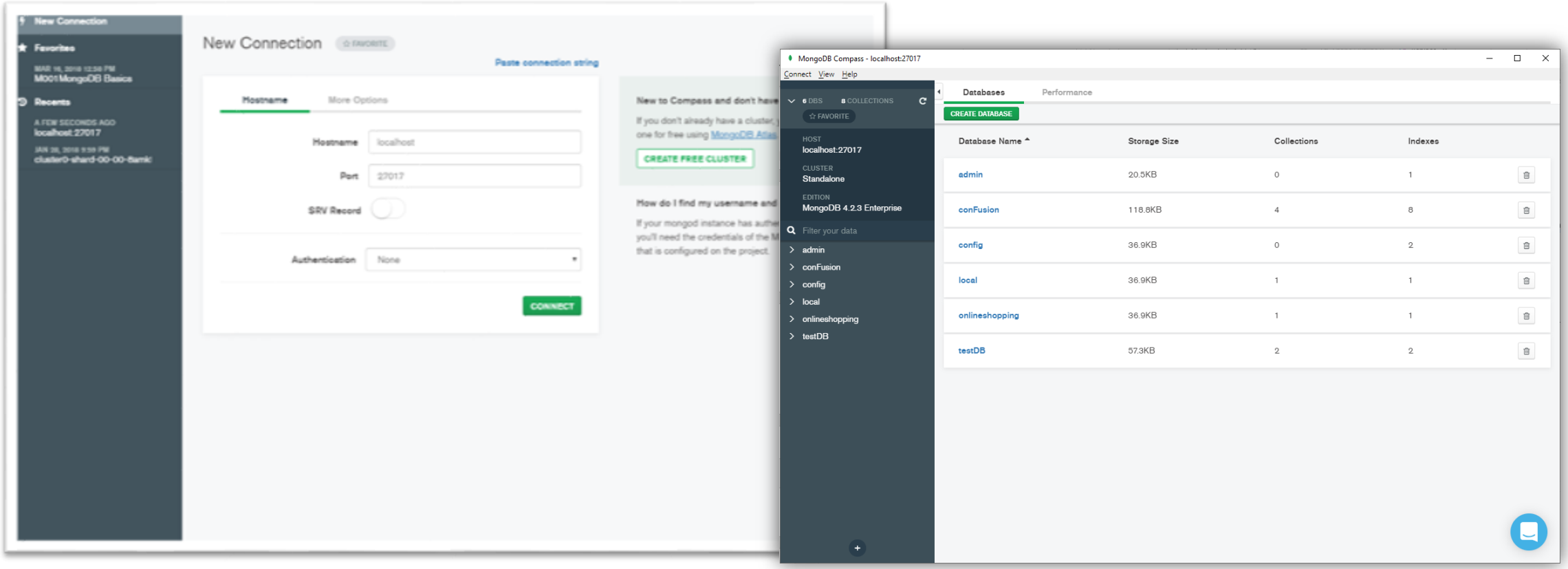


Mongo Atlas

- Create DB at https://www.mongodb.com/
- Init a NodeJS program
- Connect NodeJS program to Mongo Atlas using MongoDB
- Implement a function to insert a document to DB
- Collection: Users, {_id: 1, name: 'Mike'}

MongoDB Compass

- As the GUI for MongoDB, MongoDB Compass allows you to make smarter decisions about document structure, querying, indexing, document validation, and more. Commercial subscriptions include technical support for MongoDB Compass.



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)*

Collections

- MongoDB stores documents in collections. (Collections are similar to tables in relational databases)

```
use myDB
```

- If a database/collection does not exist, MongoDB creates the db/collection when you first store data for that collection

```
use myNewDB
db.myNewCollection.insert( { x: 1 } )
```

- *The insert() operation creates both the database myNewDB and the collection myNewCollection if they do not already exist.*

Shell Demo

```
show dbs
use testDB // switch or create
show collections
db.testCol.insert({"name": "Josh"})// db var refers to the
current database
db.testCol.find() // notice _id
// passing a parameter to find a document that has a property
"name" and value "Josh"
db.testCol.find({"name":"Josh"})
// save() = upsert if _id provided
db.testCol.insert({"name":"Mike"})
// insert 10 documents – Shell is C++ app that uses V8
for (var i=0; i<10; i++){ db.testCol.insert({"x": i}) }
```

Shell Demo

```
db.testCol.insert({a:1, b:2})
db.testCol.insert({a:3, b:4, fruit: ["apple", "orange"] })
db.testCol.insert({name: "Josh", address: {city: "Fairfield",
                                           zip: 52557,
                                           street: "1000 N 4th street"}
})
// show documents in a nice way, it will only work when you
have nested or larger documents:
db.testCol.find().pretty()
```

CRUD in MongoDB

There is no special SQL language to perform CRUD in MongoDB.

Many CRUD operations exist as methods/functions on objects in programming language API, NOT as separate language.

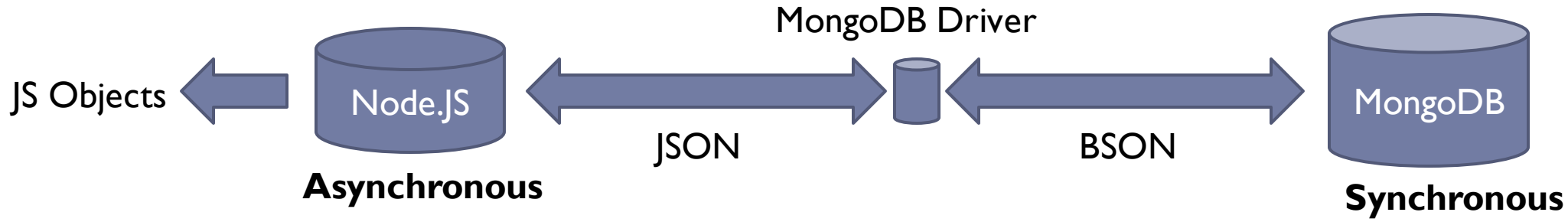
CRUD	MongoDB
Read	`find()`
Create	`Insert`
Update	`update()`
Delete	`Delete()`

MongoDB Driver

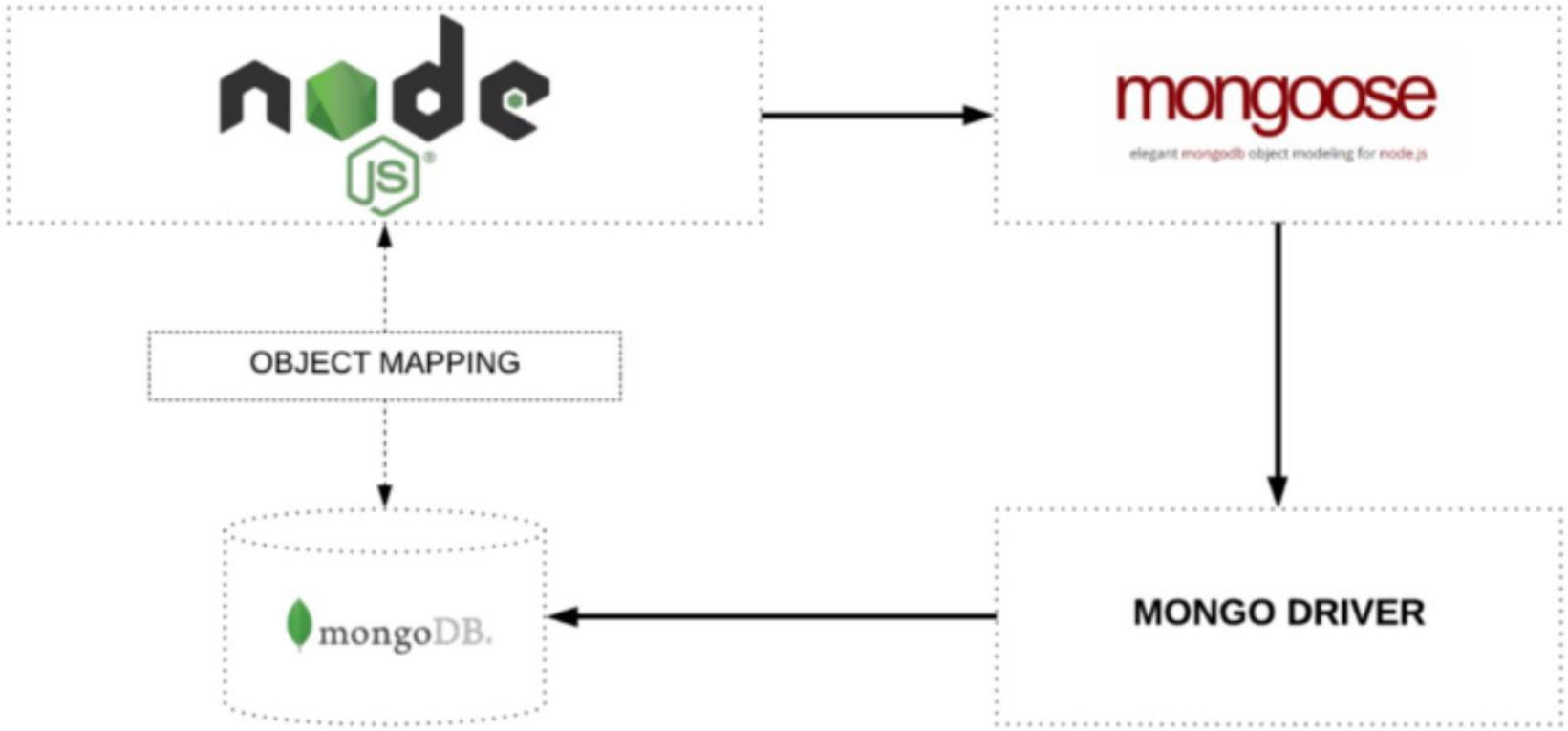
- A library written in JS to handle the communication, open sockets, handle errors and talk with MongoDB Server.

```
npm install mongodb
```

- Note that Mongo Shell is **Synchronous** while Node.JS is **Asynchronous**.



Mongoose: Object Data Modeling (ODM) library



Object Mapping between Node and MongoDB managed via Mongoose

Connect to MongoDB using mongoose

```
const mongoose = require('mongoose');

Let uri  = mongodb+srv://<user>:<password>@cluster0.l9zyoim.mongodb.net/database_name
async function main(){
return mongoose.connect(uri);
}


main().then(() => console.log('DB connected')).catch(err =>
console.log(err));
```

Create

```
async function createUser(user){
try {
const auser = new UserModel(user);
await auser.save();
return auser;
} catch (error) {
return null;
}
}
```

findOne()

```
async function findOne(name){
try {
const auser = await UserModel.findOne({name});
return auser;
} catch (error) {
return null;
}
}
```

updateOne()

```
async function updateOne(name, newAddress){
try {
const user = new UserModel({
name,
address: newAddress
})
await user.save();
return user;
// const user = await UserModel.updateOne({name}, {address: newAddress},
{upsert: false});
// return user;
} catch (error) {
}
}
```

deleteOne()

```
async function deleteOne(name) {
try {
const user = await UserModel.deleteOne({ name });
return user;
} catch (error) {
return null;
}
}
```

FindAll

```
async function findAll(){
try {
const users = await UserModel.find({}).sort('name').skip(3).limit(2);
return users;
} catch (error) {
return null;
}
}
```

Note: These will be implemented in the DB in a very specific order: **1. sort, 2. skip, 3. limit** no matter how we put them in the code *(remember cursor object is being built and sent to the server only , to run call* `.each()`*)*

Resources

- SQL vs NoSQL: https://academind.com/learn/web-dev/sql-vs-nosql/
- Mongo Shell: https://docs.mongodb.com/manual/mongo/
- Mongoose:
  - https://mongoosejs.com/docs/index.html
  - https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57
- MongoDB CRUD Operations: https://docs.mongodb.com/manual/crud/
- Node.js MongoDB Driver API: https://mongodb.github.io/node-mongodb-native/4.0/