



Full Stack Software Development

Course: Server-Side
Development Using Node.js,
Express.js and MongoDB

Lecture on: MongoDB

Instructor: Rocky Jagtiani

In the previous class, we covered...

- Routing of dynamic web applications

Poll 1 (15 Sec)

What will be the output of the following code?

1. MongoDB
2. MongoDB
MongoDB
3. Error

```
events = require('events');  
  
const myEmitter = new events.EventEmitter();  
  
myEmitter.once('event', () => console.log('MongoDB'));  
  
myEmitter.emit('event');  
myEmitter.emit('event');
```

Poll 1 (Answer)

What will be the output of the following code?

1. **MongoDB**

2. MongoDB
MongoDB

3. Error

```
events = require('events');  
  
const myEmitter = new events.EventEmitter();  
  
myEmitter.once('event', () => console.log('MongoDB'));  
  
myEmitter.emit('event');  
myEmitter.emit('event');
```

Poll 2 (15 Sec)

`rawListeners()` method returns **list** of active listeners.

Is the above statement true or false?

1. True
2. False

Poll 2 (Answer)

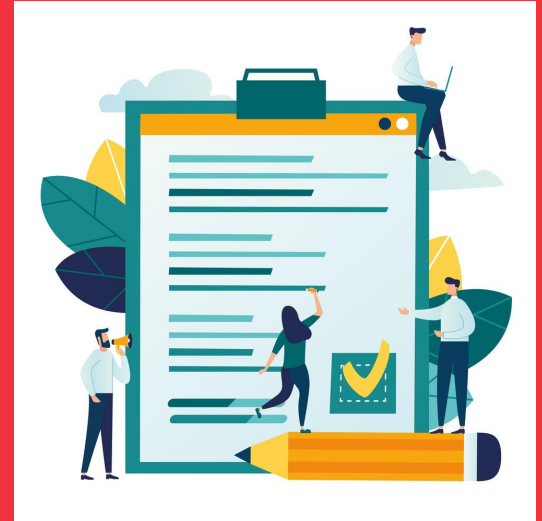
`rawListeners()` method returns **list** of active listeners.

Is the above statement true or false?

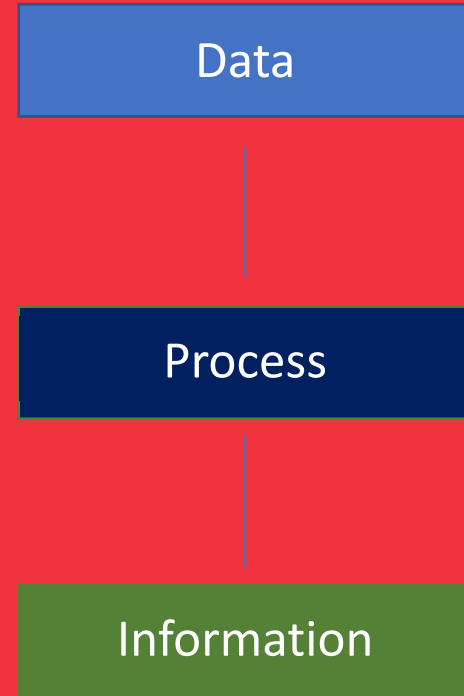
1. **True**
2. False

Today's Agenda

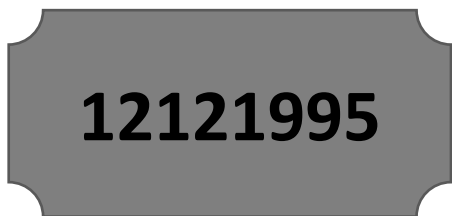
- Data and Its Storage
- File System
- Database
- Relational Database
- Non-Relational Database
- MongoDB
- MongoDB Commands



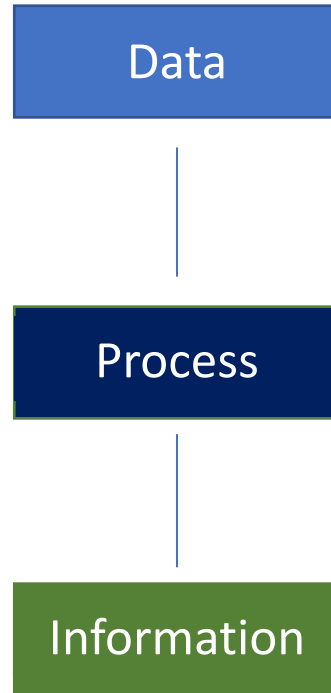
Data and Its Storage

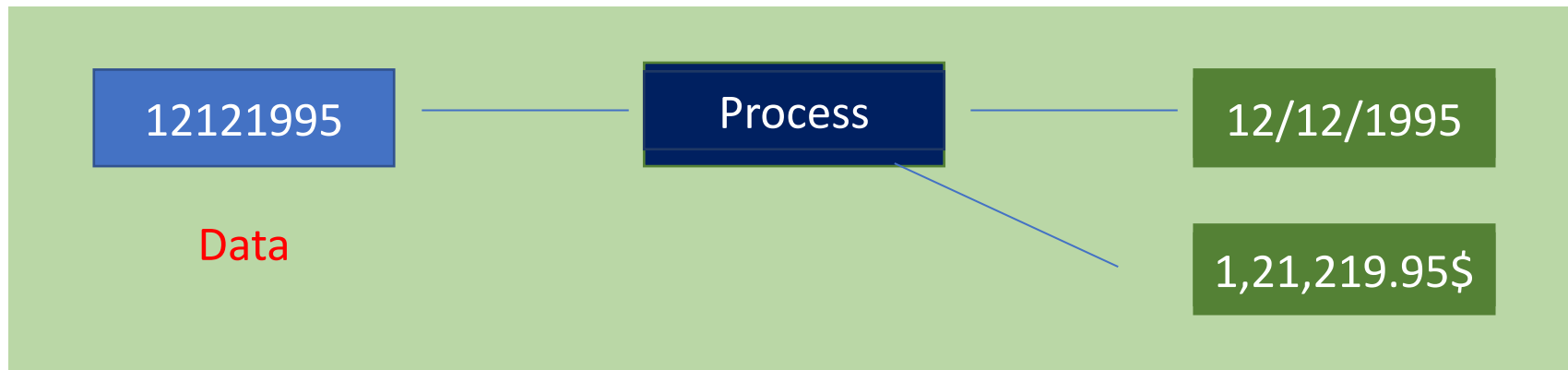


- Data includes raw and unorganised facts that need to be processed
- For example:




What does this number mean?
Is it the price of a product?
Is it a birth date?
Is it a magic number?



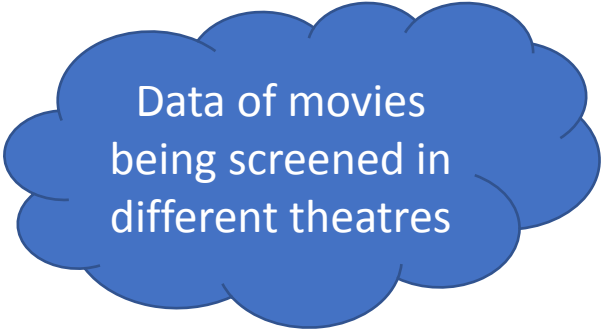


When raw data is processed, it becomes information, which is useful and meaningful

- We are surrounded by data of different types, and this data needs to be stored together at some location
- Consider the example of a movie booking app. In this app, a few different types of data can be as follows:



Data of all the customers



Data of movies being screened in different theatres



Data of all tickets booked by different customers

Poll 3 (15 Sec)

Which of the following can be classified as information?
(Note: More than one option may be correct.)

1. Temperature readings of 3 days
2. Average temperature of the last 10 days
3. Balance of some accounts in a bank
4. Net worth of HDFC bank, XYZ branch

Poll 3 (Answer)

Which of the following can be classified as information?
(Note: More than one option may be correct.)

1. Temperature readings of 3 days
- 2. Average temperature of the last 10 days**
3. Balance of some accounts in a bank
- 4. Net worth of HDFC bank, XYZ branch**

File System

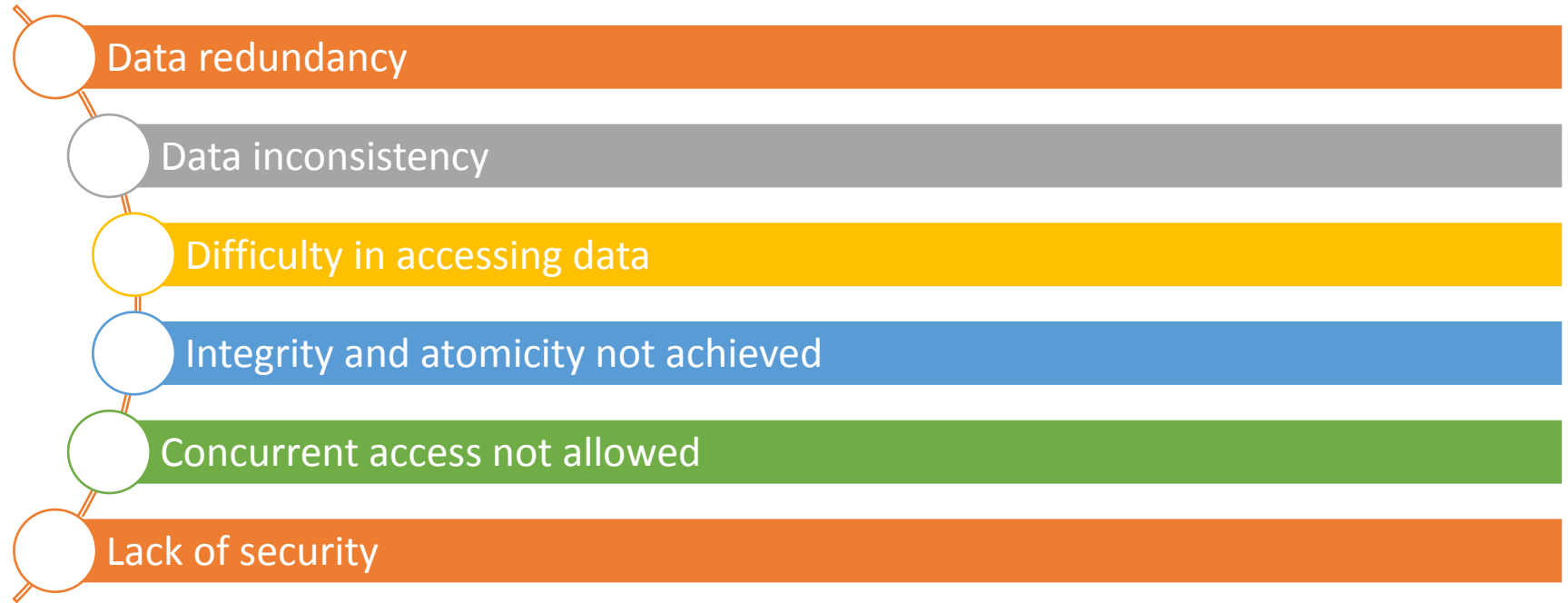


- The file system is a traditional way of storing data
- A few advantages of using a traditional way of storing are as follows:

- ☐ Is universal
- ☐ No need to learn any new language
- ☐ Low maintenance cost

Are these advantages enough?

The answer is **NO!**



Database



- To overcome the shortcomings of file systems, we use databases
- A database is an organised collection of meaningful and useful information that is stored and accessed through computers
- In other words, it is a method that is used widely to store, retrieve and manage data



Characteristics	File System	Database
Data consistency	Data is highly inconsistent	Data is consistent
Data sharing	Data sharing is difficult	Data sharing is easy
Redundancy	High data redundancy	Low data redundancy
Security	Not very secure	Secure by use of roles, passwords, etc.
Backup and recovery	No backup and recovery processes in place	Backup and recovery processes in place

Poll 4 (15 Sec)

What is not an advantage of database?

1. Data is highly redundant
2. Data is more consistent
3. Data sharing is secure and easy
4. Data can be recovered easily

Poll 4 (Answer)

What is not an advantage of database?

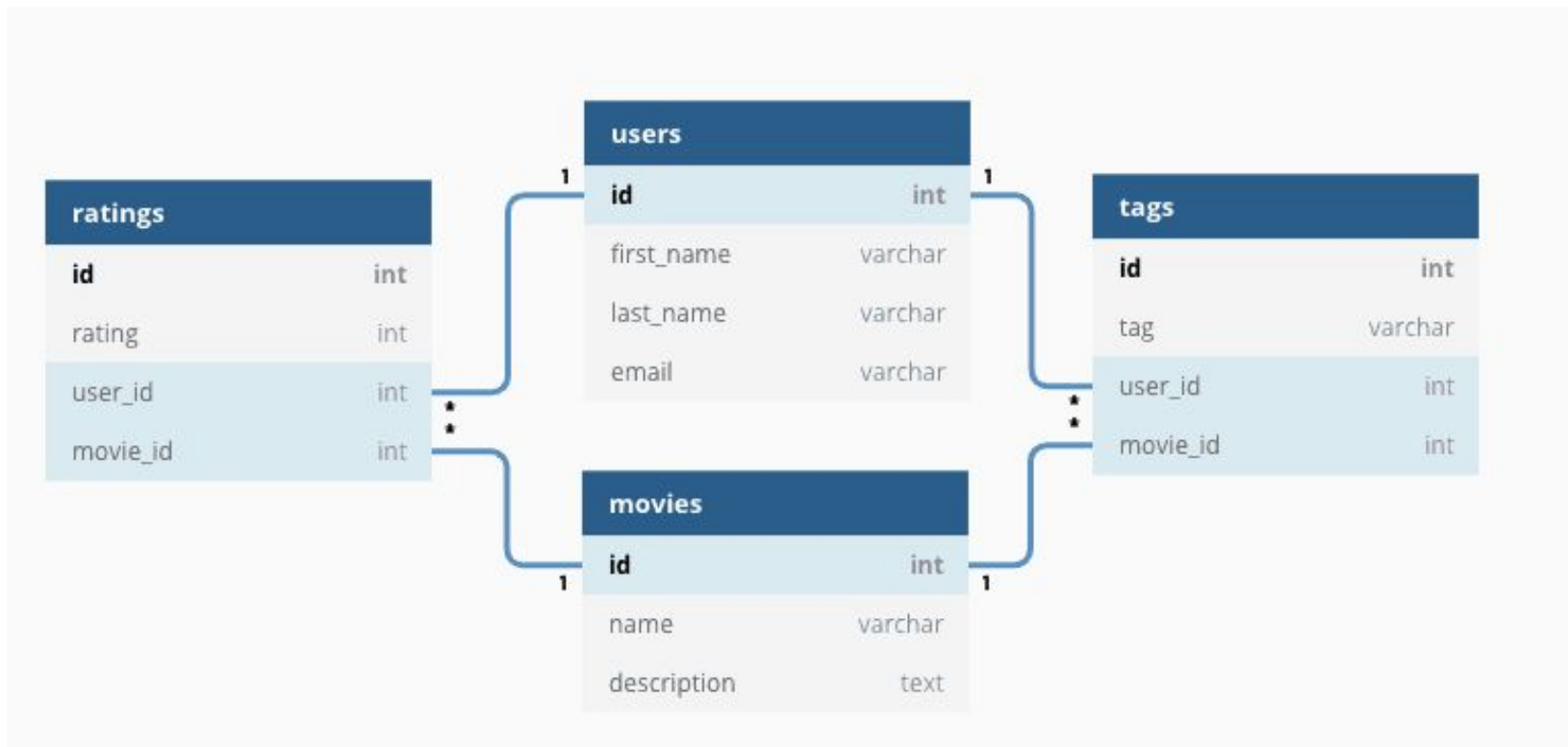
1. **Data is highly redundant**
2. Data is more consistent
3. Data sharing is secure and easy
4. Data can be recovered easily

Relational
Databases

Non-
Relational
Databases

Relational Database

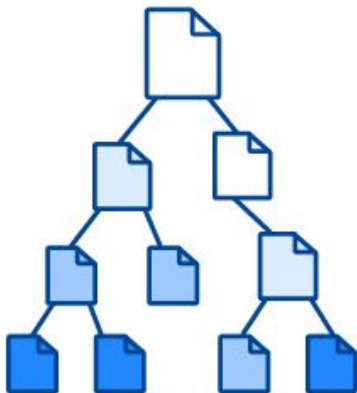
- We have been talking a lot about data, right? So, where are we going to store our data?
Data can be stored in tables
- A relational database means a collection of data items stored in a tabular (table) format with a predefined relationship between them
- These items are organised as sets of tables with rows and columns



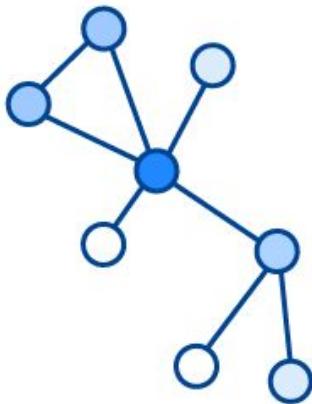
Non-Relational Database

- In a non-relational database, data is stored in an unstructured format
- Unlike relational models, there is no relation between the tables in a non-relational database

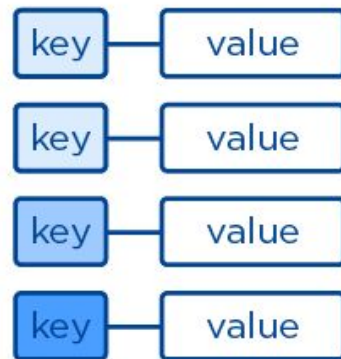
Document



Graph



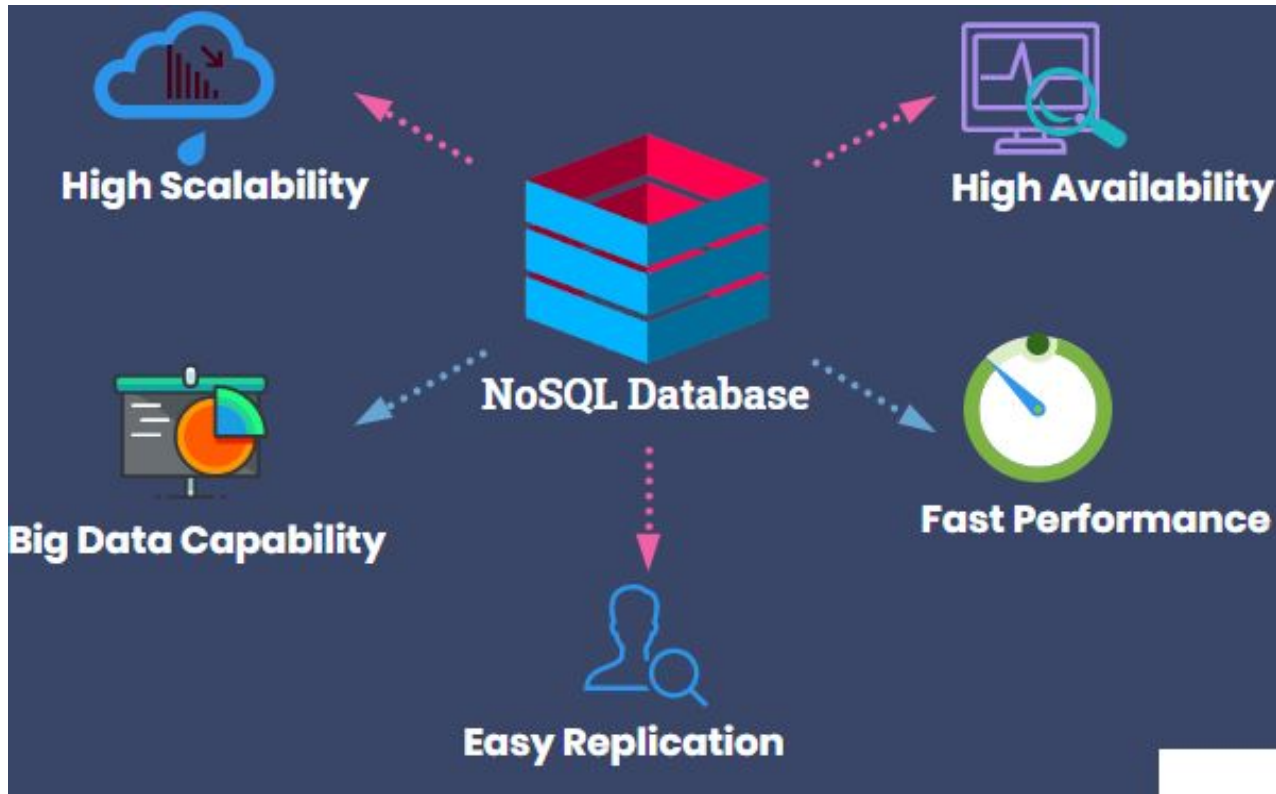
Key-Value



Some popular NoSQL databases are as follows:

- **MongoDB**
- Redis
- Couch DB
- RavenDB
- MemcacheDB
- Riak
- Neo4j: *This is a NoSQL graph database.*
- HBASE
- Perst
- HyperGraphDB
- Cassandra
- Voldemort
- Terrastore
- NeoDatis
- MyOODB

Note: list is shown in descending order of popularity in terms of no. of users



Poll 5 (45 Sec)

Which of the following statements is true?

- I:** SQL databases are suited for vertical scaling, whereas NoSQL databases are ideal for horizontal scaling
- II:** The schema in NoSQL databases is flexible
- III:** NoSQL databases are more suitable than SQL databases for complex querying procedures to obtain the relevant results

1. I and II
2. I and III
3. II and III

Poll 5 (Answer)

Which of the following statements is true?

- I:** SQL databases are suited for vertical scaling, whereas NoSQL databases are ideal for horizontal scaling
- II:** The schema in NoSQL databases is flexible
- III:** NoSQL databases are more suitable than SQL databases for complex querying procedures to obtain the relevant results

1. **I and II**

2. I and III

3. II and III

MongoDB



- Open source, document-based NoSQL database
- Created in 2007, with the idea of building databases that can support humongous amounts of data to ensure scalability
- Provides high performance, scalability and data modelling features



Before we start performing the necessary operations in MongoDB, it is essential for you to learn about the way in which MongoDB stores data and individual records

- Data is stored in the form of collections, and each collection stores information on several records
- These collections act as tables in SQL, and the information in these tables is stored in the form of documents
- Each record in the collection is stored as a document, and the information is recorded in the form of a **key:value** pair

Poll 6 (15 Sec)

Which of the following entities in MongoDB is similar to tables in SQL databases?

1. Documents
2. Database
3. Collections

Poll 6 (Answer)

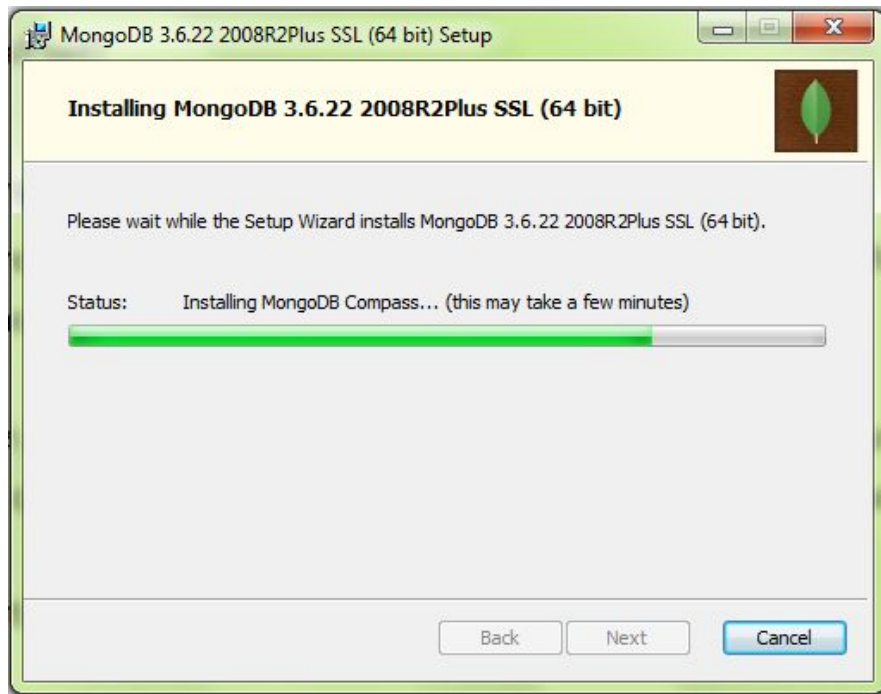
Which of the following entities in MongoDB is similar to tables in SQL databases?

1. Documents
2. Database
3. **Collections**

You can download the MongoDB database from this link: <https://www.mongodb.com>

Installation steps:

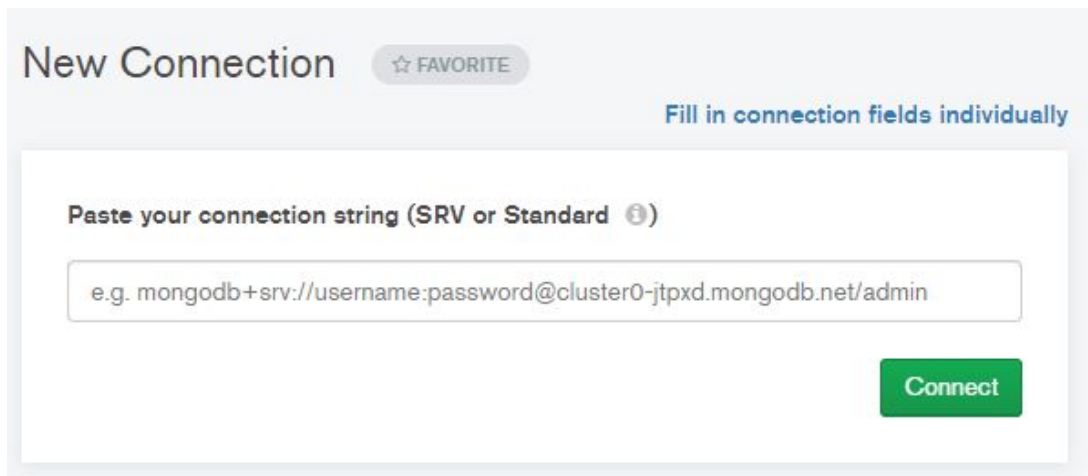
- Install the MongoDB driver: To download and install the official MongoDB driver:
 - Open the command terminal
 - Install the MongoDB package using
npm install mongodb
- Now, Node.js can use this module to manipulate MongoDB databases using
var mongo = require('mongodb');



- MongoDB is a command line-based noSQL database
- We will install the MongoDB Compass. It is an official **GUI** for MongoDB (<https://www.mongodb.com/products/compass>)
- The MongoDB Compass enables us to:
 - Visually explore the data
 - Run ad hoc queries in seconds
 - Interact with data following complete CRUD functionality.

After launching it, it would ask for a connection to MongoDB

Click on Connect



The screenshot shows the 'New Connection' dialog box in MongoDB Compass. At the top left is the title 'New Connection' followed by a '☆ FAVORITE' button. To the right is the instruction 'Fill in connection fields individually'. Below this is a text area with the prompt 'Paste your connection string (SRV or Standard ⓘ)'. Inside the text area is a placeholder example: 'e.g. mongodb+srv://username:password@cluster0-jtpxd.mongodb.net/admin'. At the bottom right of the dialog is a green 'Connect' button.

MongoDB Commands



- To create a database in MongoDB, start by creating a MongoClient object. Then, specify a connection URL with the correct IP address and the name of the database that you want to create
- MongoDB will create the database if it does not exist and make a connection to it
- To select an existing database or to create a new database, use this command *use <database_name>*
- This command is used to switch to the database mentioned in the command or create a new database by the name passed in the command

- In MongoDB, a database is not created until it gets some content.
- MongoDB waits until you have created a collection (table), with at least one document (record) before it actually creates the database (and collection).

- MongoDB stores documents in the form of collections
- Collections in NoSQL are analogous to tables in relational databases
- To create a collection and start storing the data, use this command
db.createCollection('<collection_name>')

- In MongoDB, a record is a document, which is a data structure composed of field and value pairs
- MongoDB documents are similar to JSON objects
- The values of fields may include other documents, arrays and arrays of documents

If you do not specify an ***_id field***, then MongoDB will add one for you and assign a unique id for each document.

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```

← field: value
← field: value
← field: value
← field: value

- The ***db.Collection.insert()*** command is used to insert a document into the collection

Refer: [create-collection.js](#)

- To insert a single record (document) into a collection, you can use the ***insertOne()*** method
- The first parameter of the ***insertOne()*** method is an object containing the name(s) and value(s) of each field in the document that you want to insert
- The ***insertOne()*** method also takes a callback function where you can work with any errors or the result of the insertion

```
var myobj = { name: "ABC", address: "Mumbai", CartValue: 6700 };
db.collection("customers").insertOne(myobj, function(err, res) {
  if (err)
    throw err;
  console.log("1 document inserted");
  db.close();
});
```

- To insert multiple documents into a collection in MongoDB, use the *insertMany()* method
- The first parameter of the *insertMany()* method is an array of objects containing the data that you want to insert
- The second parameter is a callback function where you can work with any errors or the result of the insertion

- To select data from a table in MongoDB, we can also use the **find()** method
- The **find()** method returns all the occurrences in the selection
- The first parameter of the **find()** method is a query object. In the example given below, we are using an empty query object, which selects all the documents in the collection

No parameters in the **find()** method gives us the same result as **SELECT *** in MySQL.

```
db.collection("customers").find({}).toArray(function(err, result) {  
  if (err) throw err;  
  console.log(result);  
  db.close();  
});
```

- To select data from a collection in MongoDB, you can use the ***findOne()*** method
- The ***findOne()*** method returns the first occurrence in the selection
- The first parameter of the ***findOne()*** method is a query object. In the following example, we use an empty query object, which selects all the documents in a collection (but returns only the first document)

```
db.collection("customers").findOne({}, function(err, result) {  
    if (err)  
        throw err;  
    console.log(result.name);  
    db.close();  
});
```

- To suppress a column, mark it as 0 and other columns as 1. Supply this information to the projection attribute of the find function
- `_id: 0` suppresses the field. If the `_id` is not specified, it will not be automatically dropped

```
db.collection('customers', function (err, collection) {  
  collection.find( {}, { projection: { _id: 0, name: 1, address: 1}}).toArray(function(err,  
items)  
  {  
    if(err)  
      throw err;  
    console.log("o/p of find() with some columns");  
    console.log(items);  
    console.log("-----");  
  });  
});
```

Note: Not specifying any other field (other than `_id`) would drop it

- The find method can take regex (regular expression) as the first parameter, i.e., the query object
- For example, in the following code, we will find all the documents whose name ends with the ***character n***
- So, records having names such as **Simran** and **Manan** would be selected

```
db.collection('customers', function (err, collection) {  
  collection.find( {name : /[A-Za-z]+n$/} , { projection: { _id: 0, name: 1, address: 1 }})  
    .toArray(function(err, items)  
    {  
      if(err)  
        throw err;  
      console.log("o/p of find() with some columns");  
      console.log(items);  
      console.log("-----");  
    });  
});
```

- Regex will start and end with the / symbol
- `/^S/` means that you want to search strings that start with an S
Example: Suraj
- `/[A-De-z]/` means that you are searching whether the input character is in range of A–D or e–z
Example: ABCwins
- `/[A-Z]+[1-5]$/` means that you are searching whether the input character starts with any character from A to Z, followed with 1 or more characters and ends with a digit 1–5
Example: ABCDEF3
- Regex rules can be much complex. Learners who are interested in exploring this topic can visit https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

The find method can take a condition like:

```
// displays all records with age < 25
```

```
db.myCollection.find(  
  {  
    age : {$eq : "25"}  
  }  
);
```

Similarly,

\$gt stands for greater than,

\$lte is 'less than or equal to'

\$gte is 'greater than or equal to' and

\$ne is 'not equal'

\$eq is 'equal'

\$in matches any of the values specified in an array

The find method can take a logical condition like this.

```
//We can use the $and operator
```

```
db.myCollection.find({  
  $and : [ {age : {$lt : 25}}, {location: "mumbai"} ]  
});
```

Other logical operators,

\$or stands for the OR operator

\$not stands for the NOT operator

Poll 7 (15 Sec)

Identify the line with an error

1. 1

2. 2

3. 3

4. 4

```
1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://27017:localhost/mykartDatabase";
3. MongoClient.connect(url, function(err, db) {
4.     if (err) throw err;
5.     console.log("Database created!");
6. });
```

Poll 7 (15 Sec)

Identify the line with an error

1. 1

2. 2

3. 3

4. 4

```
1. var MongoClient = require('mongodb').MongoClient;
2. var url = "mongodb://27017:localhost/mykartDatabase";
3. MongoClient.connect(url, function(err, db) {
4.     if (err) throw err;
5.     console.log("Database created!");
6. });
```


- Use the sort() method to sort the result in the ascending or the descending order
- The sort() method takes one parameter, which is an object defining the sorting order

```
// Sort the result alphabetically by name
var mysort = { name: 1 };

db.collection("students").find().sort(mysort).toArray(function(err, result) {
  if (err) throw err;
  console.log(result);
});
```

Use the value -1 in the sort object to sort descending.

```
{ name: 1 } // ascending
{ name: -1 } // descending
```

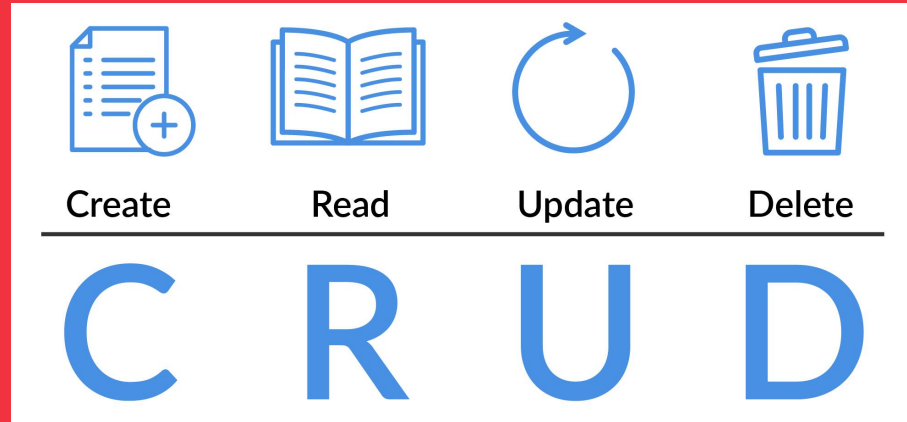
Refer: [sort-documents.js](#)

- You can delete a table or a collection as it is called in MongoDB using the [drop\(\)](#) method.
- The drop() method takes a callback function containing the error object and the result parameter, which returns true if the collection was dropped successfully; otherwise, it returns false.

```
db.collection(" students ").drop(function(err, delOK) {  
  if (err) throw err;  
  if (delOK) console.log("Collection deleted");  
});
```

Refer: [drop-collection.js](#)

CRUD Operations



- Every database management system needs to perform a series of four essential operations. These operations are collectively referred to as **CRUD** operations, or **Create**, **Read**, **Update** and **Delete** operations.
- They help us store, retrieve, update and remove records from any type of database system.

- We can **create** a new document using the **db.collection.insert()**
- Create or select a database, choose the collection and then insert the document by passing the key-value pairs inside the bracket.
- For example, if you're adding a new document to a collection 'blogs', the syntax would be

```
db.blogs.insert({'blog_id': 101, 'likes': 24, 'comments': 12})
```

- **Read** the documents using **db.collection.find()**
- All the documents in the collection will be returned
- Can be used to filter documents based on a specific field and value pair
- Also, additional conditional statements and logical operators can also be added to further filter the documents

- **Update** the values using **db.collection.update()**
- Find the document that you need to update
- Then mention all the details with which you need to update the values of the document
- For Example -

```
db.blogs.insert({'blog_id': 101,'likes':24,'comments':12})
```

In the above case if we want to update the number of likes to 30 and the number of comments to 14 then you need to perform the following operation -

```
db.blogs.update({'blog_id': 101},{'blog_id': 101, 'likes':30,'comments':14})
```

- You can update a single record or a document in MongoDB using the [updateOne\(\)](#) method.
- The first parameter of the **updateOne()** method is a query object defining the document to be updated and the second parameter is an object defining the new values of the document.

Note: *If the query finds more than one record, only the first occurrence is updated.
To update specified fields we need to **\$set operator**, only that are to be updated.*

- To update all the documents that meet the criteria of the query, use the [updateMany\(\)](#) method

Refer: [update-one-many-documents.js](#)

- Delete a document using **db.collection.remove()**
- Match the document that you want to delete and then delete it with **db.collection.remove** command.

- To delete a record or a document as it is called in MongoDB, the [deleteOne\(\)](#) method is used
- The first parameter of the deleteOne() method is a query object defining the document to be deleted

Note: *If the query finds more than one document, only the first occurrence is deleted*

- To delete more than one document, use the [deleteMany\(\)](#) method
- The first parameter of the deleteMany() method is a query object defining the document to be deleted

Refer: delete-one-many.js

Poll 8 (15 Sec)

State whether the following statement is true or false.

The \$in operator is used to match values from a given set.

Consider the following example:

`db.inventory.find({ qty: { $in: [5, 15] } })`

This query selects all the documents in the inventory collection where the qty field value is either 5 or 15. Although you can express this query using the \$or operator, choose the \$in operator instead when performing equality checks on the same field.

1. True
2. False

Poll 8 (15 Sec)

State whether the following statement is true or false.

The \$in operator is used to match values from a given set.

Consider the following example:

`db.inventory.find({ qty: { $in: [5, 15] } })`

This query selects all the documents in the inventory collection where the qty field value is either 5 or 15. Although you can express this query using the \$or operator, choose the \$in operator instead when performing equality checks on the same field.

1. **True**

2. False

Hands on Exercise (20 mins)

- Define an array of JSON data in a file named hands-on-dataset.js
- To code mongo-hands-on.js, follow these steps:
 - Perform necessary coding for creating a MongoClient.
 - Load the data from the [hands-on-dataset.js](#) file.
 - Make a connection and create the database mongo-hands-on.
- Use insertMany() to insert many documents into the collection **greatcontributors**
- Choose a document of your choice and delete it (use findOneAndDelete)
Refer: <https://docs.mongodb.com/manual/reference/method/db.collection.findOneAndDelete/>
- Choose a document of your choice and update it (use findOneAndUpdate)
Refer: <https://docs.mongodb.com/manual/reference/method/db.collection.findOneAndUpdate/>

Refer: hands-on-dataset.js and mongo-hands-on.js

Project Work - Checkpoint 3



- Code the **app/config/db.config.js** file. In db.config.js file, we would define the connection link to MongoDB.
- We will not create any database, collection or add documents using MongoDB commands, as it is prone to errors when it comes to larger databases and is also time consuming.
- In the next session, you will learn about Mongoose, which is an ODM(Object Data Modelling), and use it to define the database schema

Homework

Create a movie collection with 4-5 documents. A sample document would look like this
{ title: 'Spiderman Homecoming', imdb: { rating: 4.3, votes: 525673, id: 778200 } }

Write a code to achieve the following:

- First, create a database movies and a collection called movieratings and insert five documents into it [Hint: use insertMany()]
- Find a single document from the movies collection with the following criteria:
 - A query document that configures the query to return only movies with the title 'The Room' (Hint: Insert two movies with this name but with different ratings)
 - Sort the matched documents in the descending order based on the ratings. So, if your query matches multiple documents, the returned document will be the document with the highest rating
 - A projection that explicitly excludes the _id field from the returned documents and explicitly includes only the title and the imdb object (and its embedded fields)
For example, a sample o/p could look like this
{ title: 'The Room', imdb: { rating: 3.5, votes: 22565, id: 368227 } }

Refer: [mongoDb_homework](#)

Doubt Clearance (5 mins)

Key Takeaways

- We understood what is MongoDB and its advantages.
- You learnt different about MongoDB commands and their applications.

The following tasks are to be completed after today's session:

Homework
MCQs
Coding Questions
Course Project - Checkpoint 4

In the next class, we will discuss...

- The applications of Mongoose



Thank you!