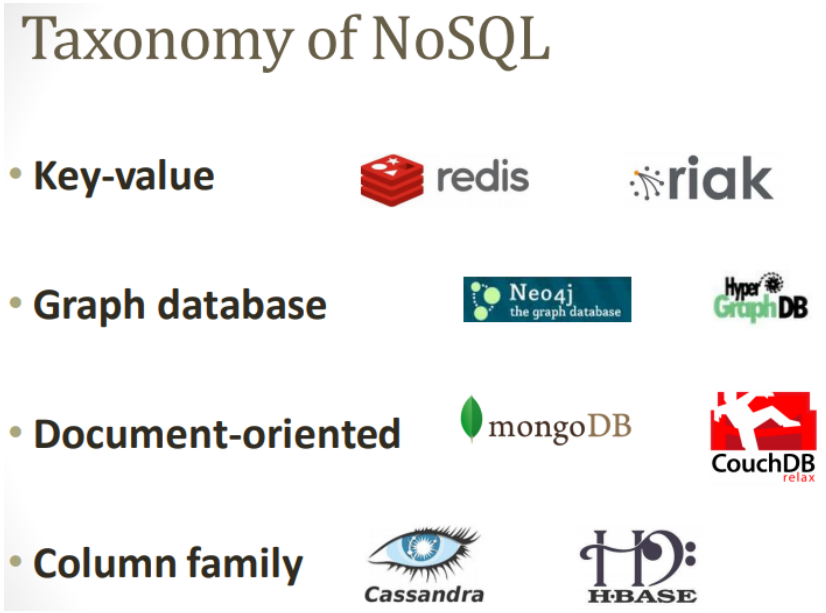
NoSQL – MongoDB - Introduction

Module 7

Dr. L.M. Jenila Livingston
VIT Chennai

MongoDB

- Name comes from "Humongous" & huge data
- Developed by 10gen
- Founded in 2007
- Written in C++, developed in 2009
- One of the most popular **NoSQL database**
  - Not Only SQL
- Document Oriented Database (max 16 MB)
- Full index for High Performance
- MongoDB stores documents or objects
- Document storage in BSON
  - Binary form of JSON
  - Binary-encoded serialization of JSON-like docs
- Each entry consists of a field name, data type, and a value
- Dynamic schema
  - No DDL

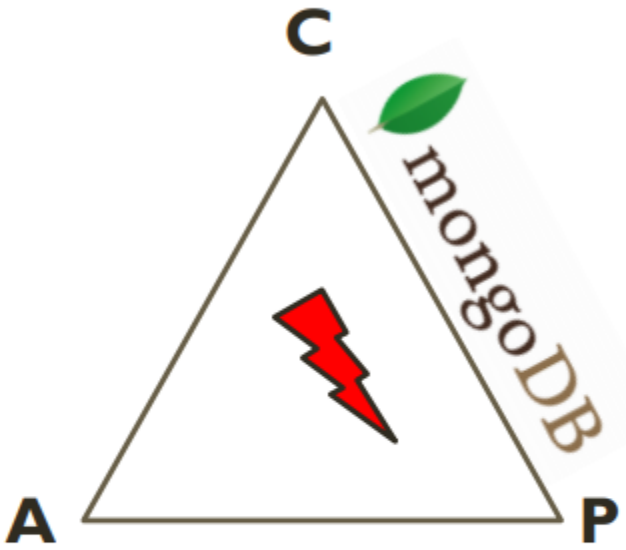


MongoDB: CAP approach

C-P on CAP

Focus on Consistency and Partition tolerance

- **C**onsistency
  - all replicas contain the same version of the data
- **A**vailability
  - system remains operational on failing nodes
- **P**artition tolarence
  - multiple entry points
  - system remains operational on system split

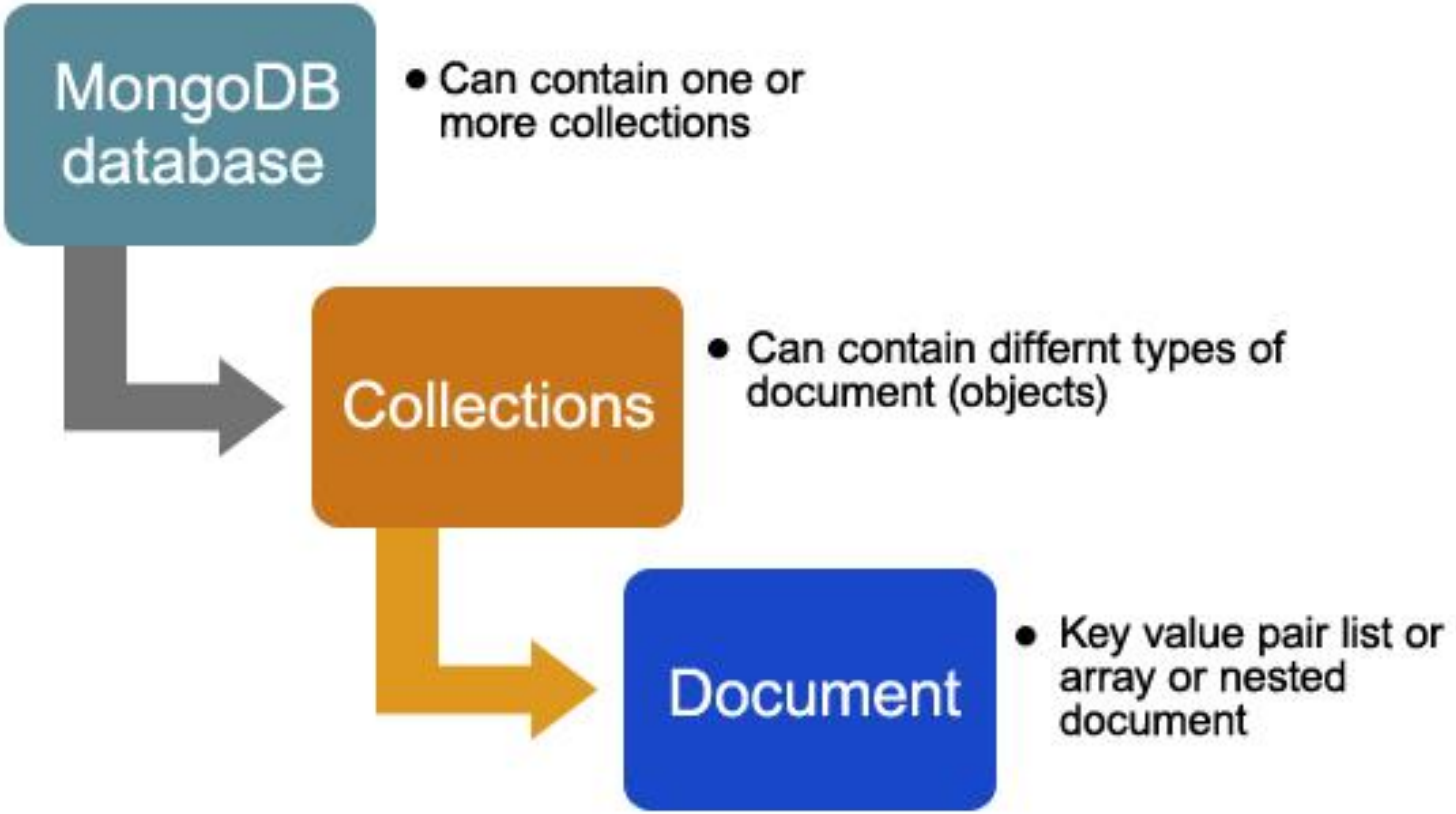


CAP Theorem:
satisfying all three at the same time is impossible

Source: https://www.ccs.neu.edu/home/kathleen/classes/cs3200/20-NoSQLMongoDB.pdf

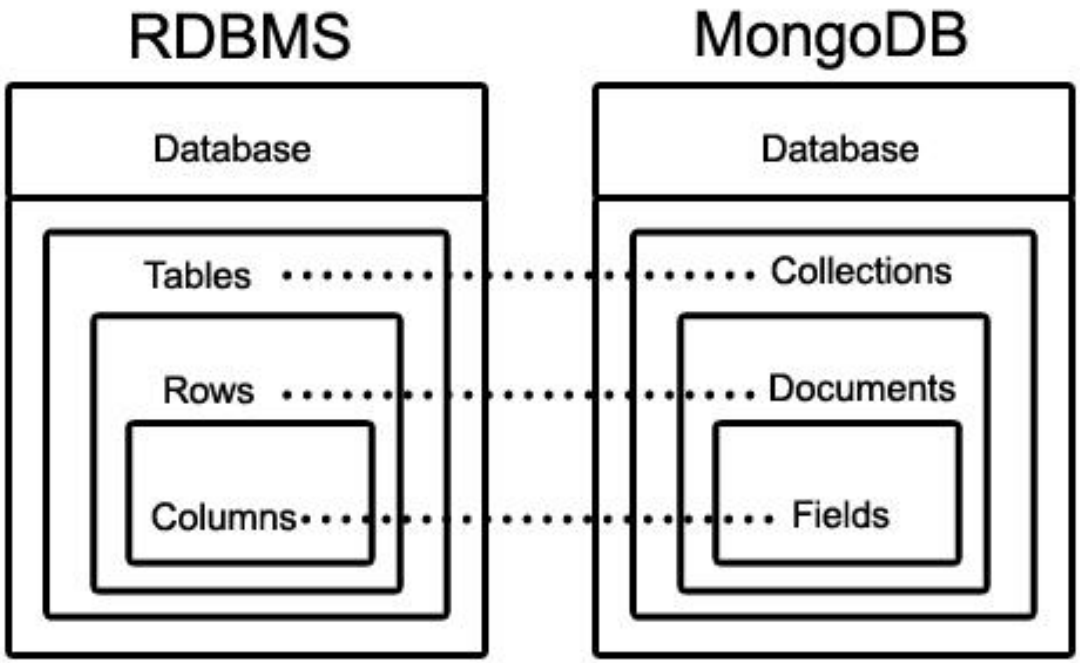
Integration with Others

MongoDB: Hierarchical Objects



Source: https://www.educba.com/what-is-mongodb/

Mapping RDBMS to MongoDB



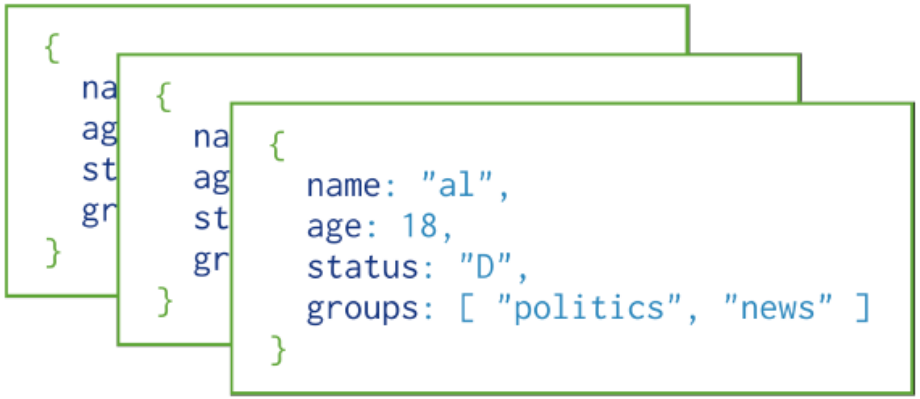
Source: https://www.educba.com/what-is-mongodb/

MongoDB Model

One ***document*** (e.g., one tuple in RDBMS)

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

One ***Collection*** (e.g., one Table in RDBMS)

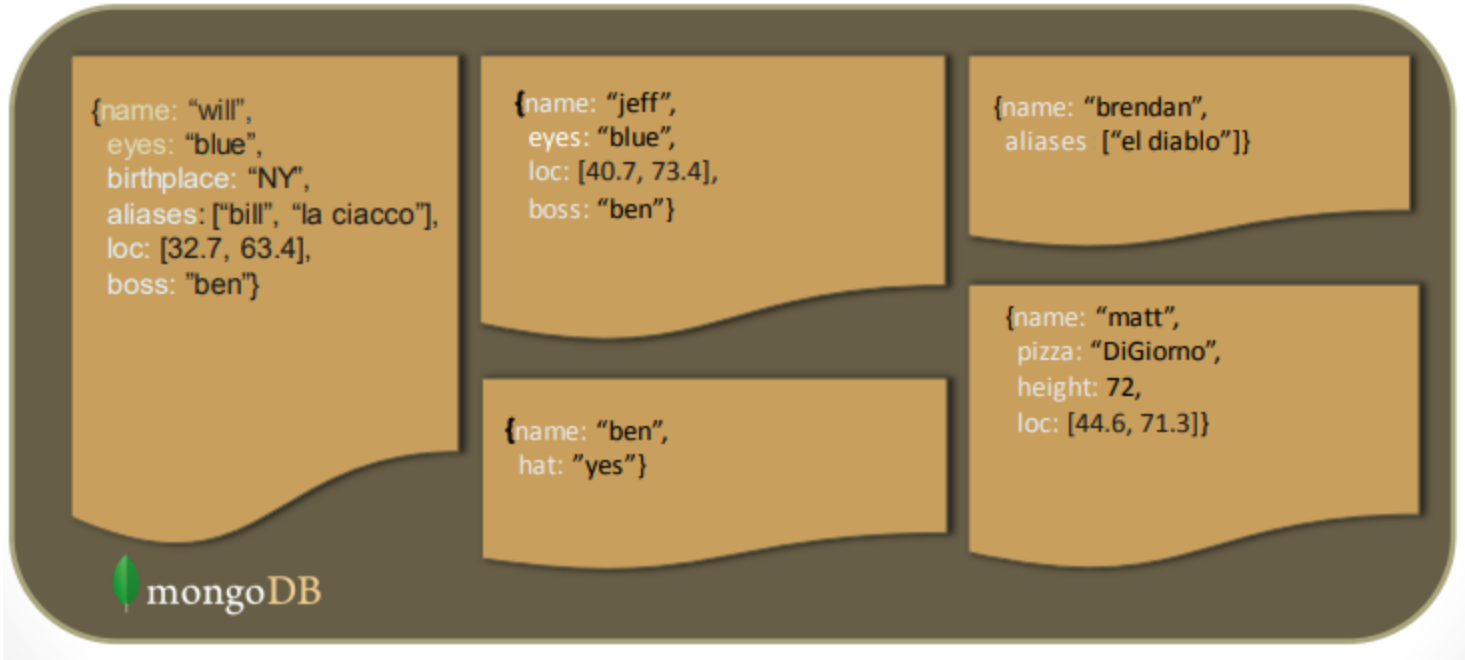


- **Collection** is a group of similar documents
- Within a collection, each document must have a unique Id

Unlike RDBMS:
No Integrity Constraints in MongoDB

Schema Free

- MongoDB does not need any pre-defined data schema
- Every document in a collection could have different data



JSON format

- Data is in name / value pairs
- A name(key)/value pair consists of a field name followed by a colon, followed by a value:
  - Example: “name”**:** “Leni”
- Data is separated by commas
  - Example: “name”: “Leni”**,** Address : “ABABAB”
- Curly braces hold objects
  - Example: **{**“name”: “Leni”, Address : “ABABAB” **}**
- An array is stored in brackets []
  - Example

  [{“name”: “Leni”, Address : “ABABAB” },
  {“name”: “Yoda”, affiliation: “rebels”} **]**

Another Example

```
{ author: 'joe',
  created : new Date('03/28/2009'),
  title : 'Yet another blog post',
  text : 'Here is the text...',
  tags : [ 'example', 'joe' ],
  comments : [
              { author: 'jim',
                comment: 'I disagree'
              },
              { author: 'nancy',
                comment: 'Good post'
              }
  ]
}
```

Remember it is stored in binary formats (BSON)

```
"\x16\x00\x00\x00\x02hello\x00
 \x06\x00\x00\x00world\x00\x00"

"1\x00\x00\x00\x04BSON\x00&\x00
 \x00\x00\x020\x00\x08\x00\x00
 \x00awesome\x00\x011\x00333333
 \x14@\x102\x00\xc2\x07\x00\x00
 \x00\x00"
```

BSON Types

Type	Number
Double	1
String	2
Object	3
Array	4
Binary data	5
Object id	7
Boolean	8
Date	9
Null	10
Regular Expression	11
JavaScript	13
Symbol	14
JavaScript (with scope)	15
32-bit integer	16
Timestamp	17
64-bit integer	18
Min key	255
Max key	127

http://docs.mongodb.org/manual/reference/bson-types/

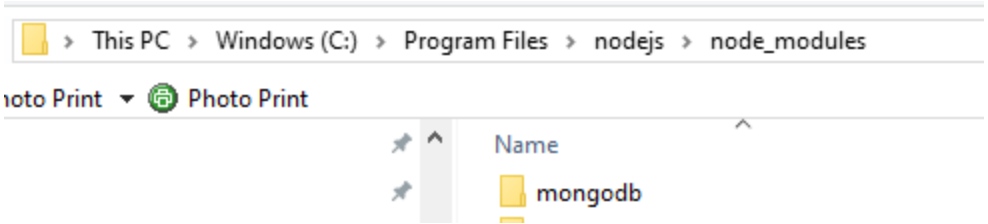
The _id Field

- By default, each document contains an _id field. This field has a number of special characteristics:
  - Value serves as primary key for collection.
  - Value is unique, immutable, and may be any non-array type.
  - Default data type is ObjectId, which is “small, likely unique, fast to generate, and ordered.” Sorting on an ObjectId value is roughly equivalent to sorting on creation time.

http://docs.mongodb.org/manual/reference/bson-types/

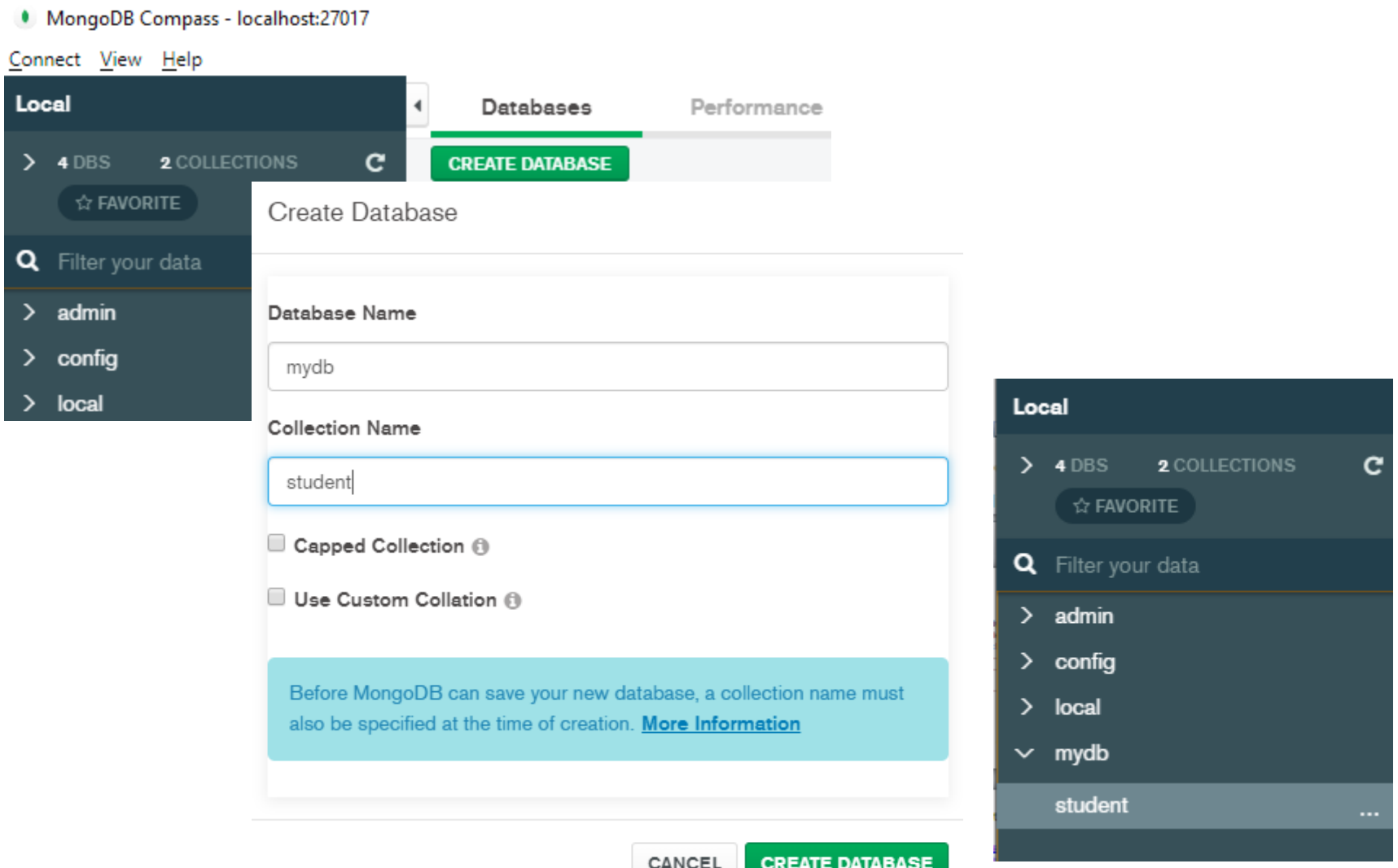
Part 1: Download mongodb Compass

- Download and install from
- https://www.mongodb.com/try/download/community
- Connecting to NodeJS



- Type **npm install mongodb**
  - **Node** Package Manager
  - it is an online repository for the publishing of open-source **Node**. js projects;
  - it is a command-line utility for interacting with said repository that aids in package installation

Create database & Collection



CRUD

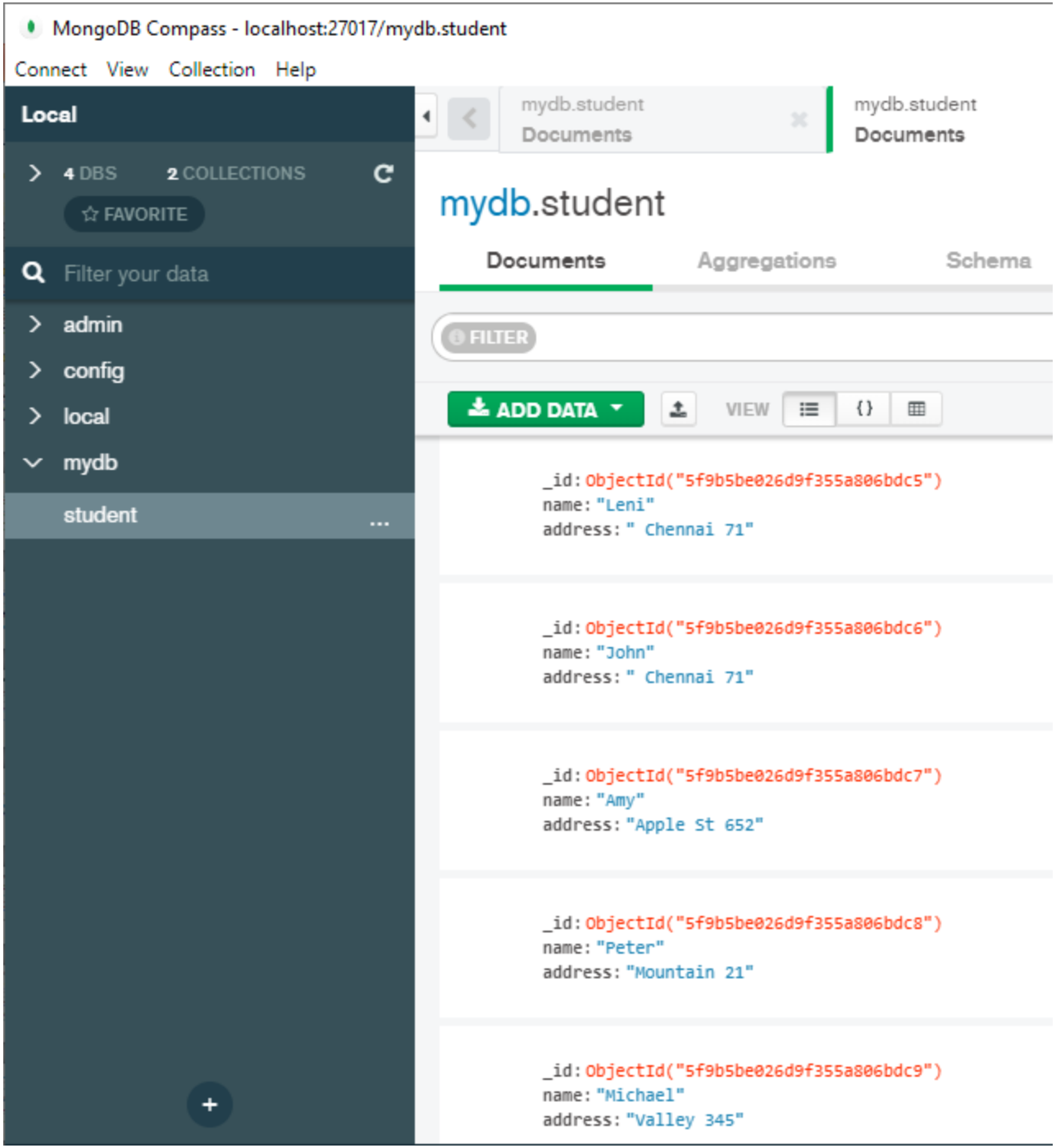
- **C**reate
  - db.collection.**insert**(<document>)
  - db.collection.save(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- **R**ead
  - db.collection.**find**(<query>, <projection>)
  - db.collection.findOne(<query>, <projection>)
- **U**pdate
  - db.collection.update(<query>, <update>, <options>)
- **D**elete
  - db.collection.remove(<query>, <justOne>)

Insert a Single record

```
var dbo = db.db("mydb");
var myobj = [
   { name: 'John', address: Chennai 71'},
];
  dbo.collection(“student").insertOne(myobj);
```

Insert Multiple records

```
var dbo = db.db("mydb");
var myobj = [
   { name: ‘Leni', address: ' Chennai 71'},
   { name: 'John', address: ' Chennai 71'},
   { name: 'Amy', address: 'Apple St 652'},
   { name: ‘Peter', address: 'Mountain 21'},
   { name: 'Michael', address: 'Valley 345'},
   { name: 'Sandy', address: 'Ocean St 2'},
];
  dbo.collection(“student").insertMany(myobj);
```


MongoDB Compass - localhost:27017/mydb.student
Connect View Collection Help
Local
4 DBS 2 COLLECTIONS
FAVORITE
Filter your data
admin
config
local
mydb
student
mydb.student
Documents
mydb.student
Documents
mydb.student
Documents
Aggregations
Schema
FILTER
ADD DATA
VIEW
_id: ObjectId("5f9b5be026d9f355a806bdc5")
name: "Leni"
address: " Chennai 71"
_id: ObjectId("5f9b5be026d9f355a806bdc6")
name: "John"
address: " Chennai 71"
_id: ObjectId("5f9b5be026d9f355a806bdc7")
name: "Amy"
address: "Apple St 652"
_id: ObjectId("5f9b5be026d9f355a806bdc8")
name: "Peter"
address: "Mountain 21"
_id: ObjectId("5f9b5be026d9f355a806bdc9")
name: "Michael"
address: "Valley 345"

Find First Record

//Find the first document in the students collection:

dbo.collection(“student").findOne()

Find All and Display Records

//Find the All document in the students collection:

dbo.collection(“student"). **find()**

If the query finds more than one record, only the first occurrence is updated.

Update One record

```
//Update one record

var myquery = { address: “Chennai 71" };
var newvalues = { $set: {name: “Jaff", address: “Chennai 88" } };
collection.updateOne(myquery, newvalues)
```

Update all documents where the name starts with the letter “L":

Update Multiple records

```
//Update Multiple records
var myquery = { address: /^L/ };
var newvalues = { $set: {name: “Jaff", address: “Chennai 88" } };
collection.updateMany(myquery, newvalues)
```

If the query finds more than one document, only the first occurrence is deleted.

Delete One Record

```
//Delete one record
  var myquery = { address: 'Mountain 21' };
collection.deleteOne(myquery)
```

Delete all documents where the address starts with the letter “C":

Delete Multiple Records

```
//Delete multiple records:
  var myquery = { address: /^C/ };
collection.deleteMany(myquery)
```