# Web Application Development

Produced by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology

http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE


PART 3

mongoose & mongoDB

# Table of Contents




mongoose

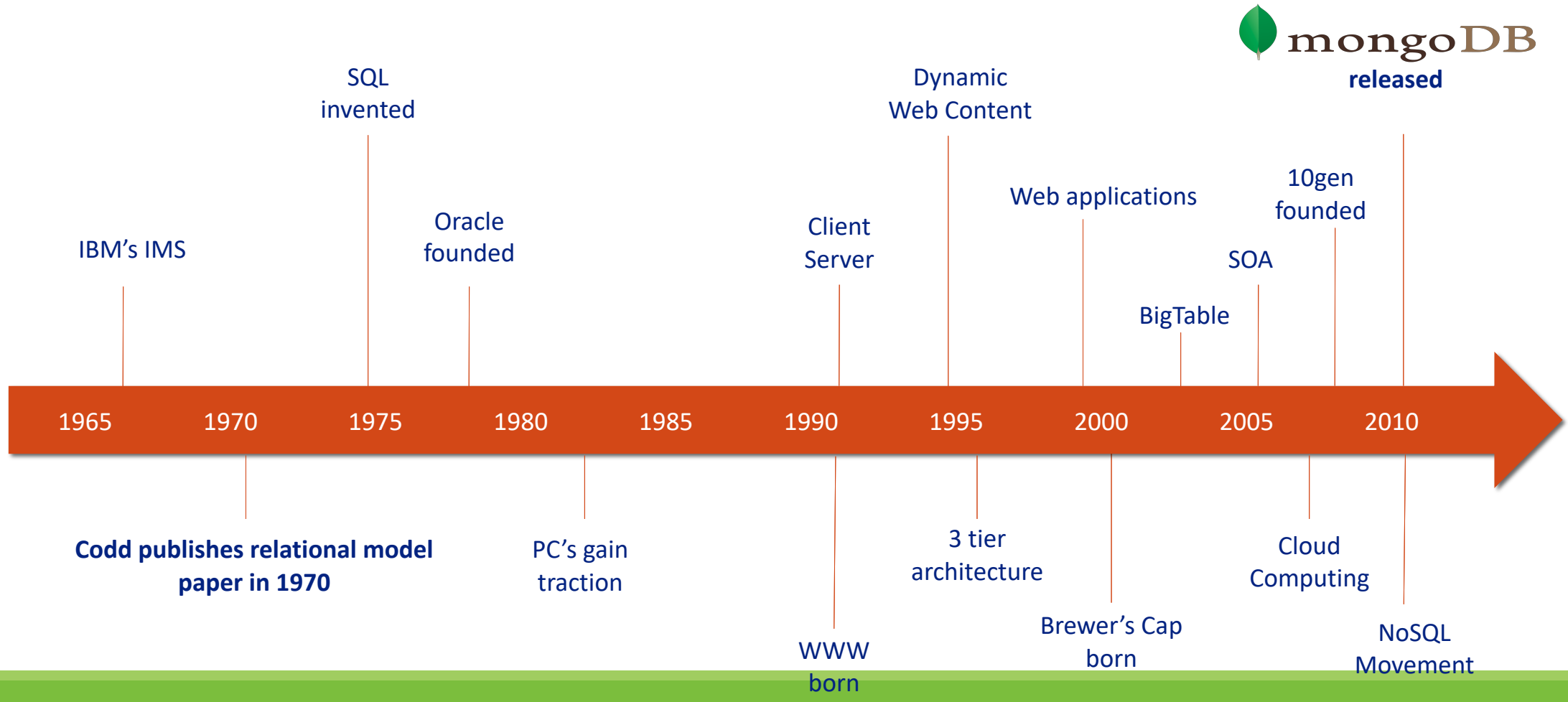

# Databases

HISTORY & BACKGROUND

mongoose

mongoDB

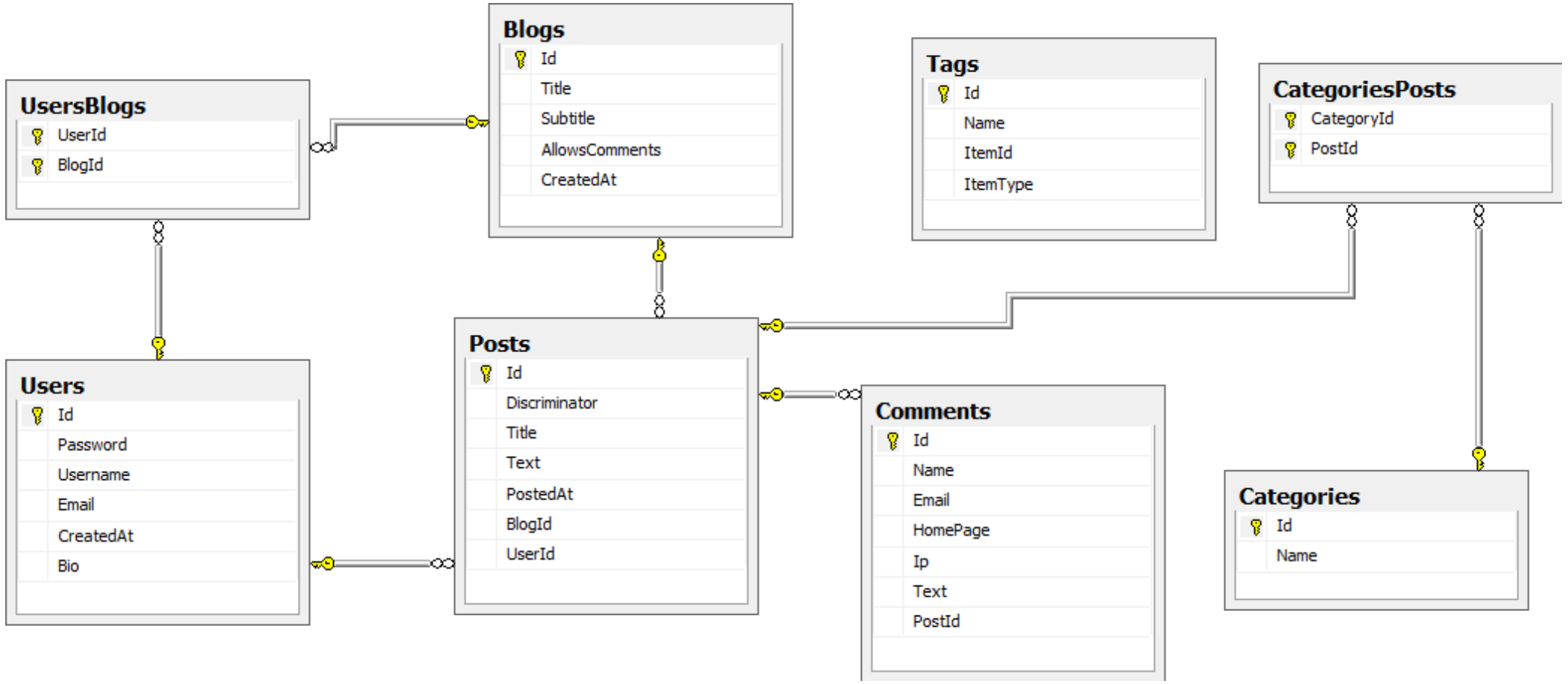
# Dawn of the Databases



mongoose

mongoDB

# Relational Databases



mongoose


# Relational Databases Strengths

Data stored in a RDBMS is very compact (disk was more expensive)

SQL and RDBMS made queries flexible with rigid schemas

Rigid schemas helps optimize joins and storage

Massive ecosystem of tools, libraries and integrations

Been around 40 years!

mongoose

mongoDB

# Enter Big Data

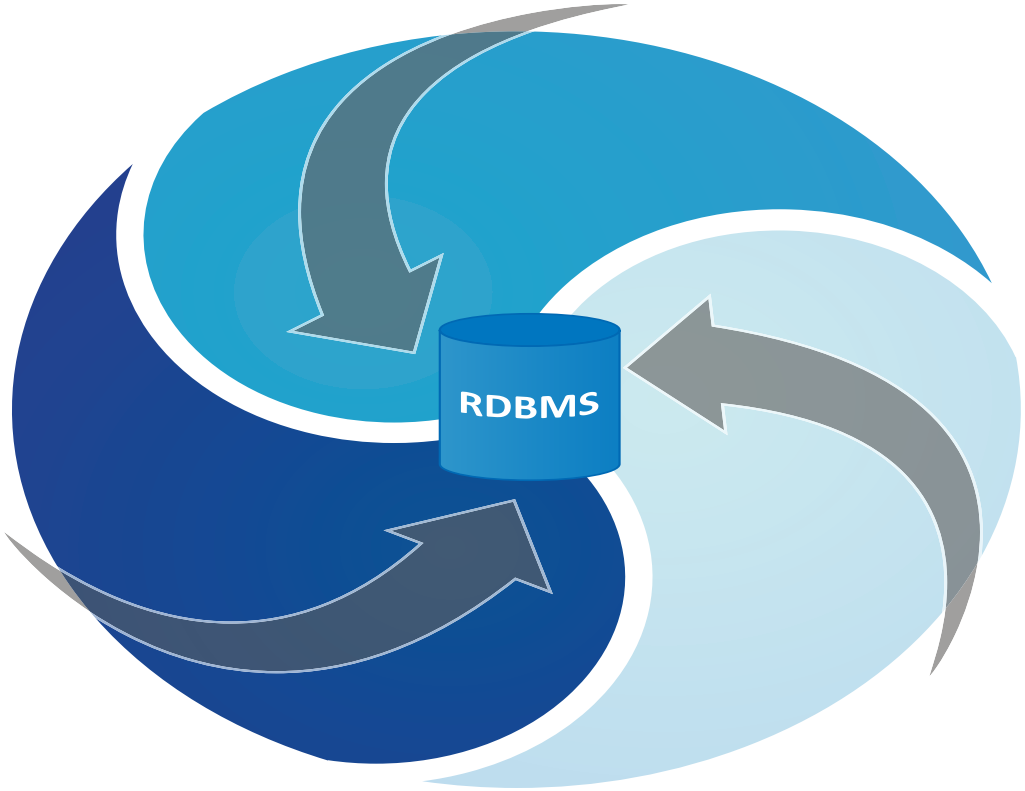
Gartner uses the 3Vs to define

- Volume - Big/difficult/extreme volume is relative
- Variety
  - Changing or evolving data
  - Uncontrolled formats
  - Does not easily adhere to a single schema
  - Unknown at design time
- Velocity
  - High or volatile inbound data
  - High query and read operations
  - Low latency

# RDBMS Challenges

## DATA VARIETY & VOLATILITY

- Extremely difficult to find a single fixed schema
- Don’t know data schema a-priori




## VOLUME & NEW ARCHITECTURES

- Systems scaling horizontally, not vertically
- Commodity servers
- Cloud Computing

## TRANSACTIONAL MODEL

- N x Inserts or updates
- Distributed transactions

mongoose


# Enter NoSQL

Modern NoSQL theory and offerings started in early 2000s
(idea around much longer)

Modern usage of term introduced in 2009

NoSQL = Not Only SQL

A collection of very different products

Alternatives to relational databases when they are a bad fit

Motives

- Horizontally scalable
- Flexibility

mongoose

mongoDB

# NoSQL Databases – Key-Value

## Key-Value Stores

- Key-Value Stores
- Value (Data) mapped to a key (think primary)

## Big Table Descendants

- Looks like a distributed multi-dimension map
- Data stored in a column oriented fashion
- Predominantly hash based indexing

## Document oriented stores

- Data stored as either JSON or XML documents


# 10gen & MongoDB

2007

- Eliot Horowitz & Dwight Merriman tired of reinventing the wheel
- 10gen founded
- MongoDB Development begins

2009

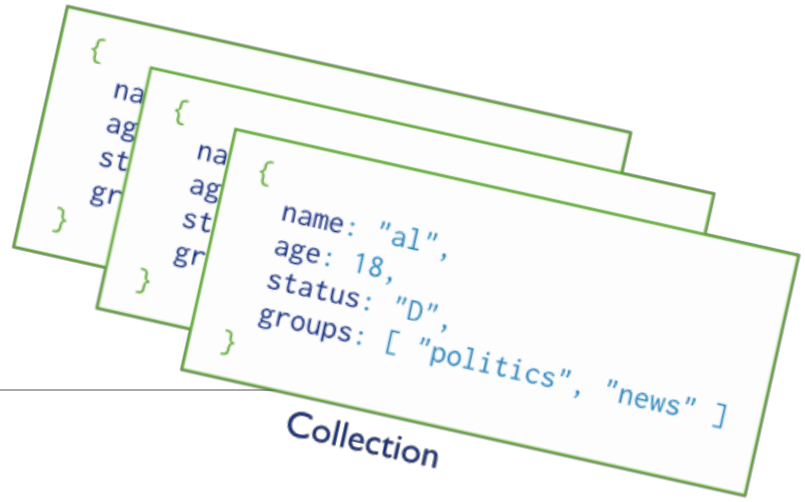
- Initial release of MongoDB

73M+ in funding

- Funded by Sequoia, NEA (New Enterprise Associates), Union Square Ventures, Flybridge Capital


# MongoDB

OBJECT-DOCUMENT MODEL



# Overview: MongoDB – What is it?

MongoDB is an open-source **document store** database

- The leading NoSQL database (see next slide)
- Save JSON-style objects with dynamic schemas

Support many features:

- Support for indices
- Rich, document-based queries (CRUD operations)
- Flexible aggregation and data processing
- Store files of any size without complicating your stack
- Fast in-place updates

https://db-engines.com/en/ranking

mongoose

mongoDB

# DB-Engines Ranking (October 2018)

346 systems in ranking, October 2018

| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Oct 2018 | Sep 2018 | Oct 2017 | | | Oct 2018 | Sep 2018 | Oct 2017 |
| 1. | 1. | 1. | Oracle | Relational DBMS | 1319.27 | +10.15 | -29.54 |
| 2. | 2. | 2. | MySQL | Relational DBMS | 1178.12 | -2.36 | -120.71 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational DBMS | 1058.33 | +7.05 | -151.99 |
| 4. | 4. | 4. | PostgreSQL | Relational DBMS | 419.39 | +12.97 | +46.12 |
| 5. | 5. | 5. | MongoDB | Document store | 363.19 | +4.39 | +33.79 |
| 6. | 6. | 6. | DB2 | Relational DBMS | 179.69 | -1.38 | -14.90 |
| 7. | ↑ 8. | ↑ 9. | Redis | Key-value store | 145.29 | +4.35 | +23.24 |
| 8. | ↓ 7. | ↑ 10. | Elasticsearch | Search engine | 142.33 | -0.28 | +22.09 |
| 9. | 9. | ↓ 7. | Microsoft Access | Relational DBMS | 136.80 | +3.41 | +7.35 |
| 10. | 10. | ↓ 8. | Cassandra | Wide column store | 123.39 | +3.83 | -1.40 |
| 11. | 11. | 11. | SQLite | Relational DBMS | 116.74 | +1.29 | +4.76 |

mongoose

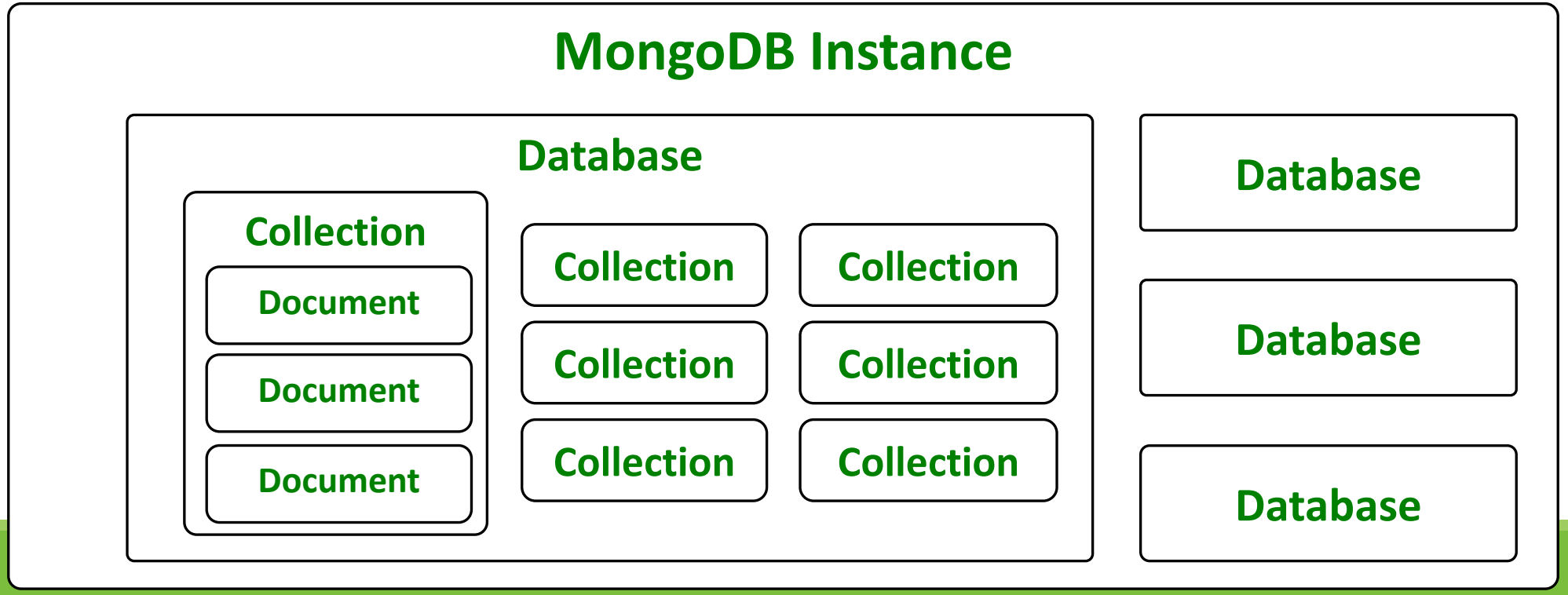

# Overview: MongoDB Key Features

- Data stored as documents (JSON)
- Schema free
- CRUD operations – (Create Read Update Delete)
- Atomic document operations
- Ad hoc Queries like SQL
  - Equality
  - Regular expression searches
  - Ranges
  - Geospatial
- Secondary indexes
- Sharding (sometimes called partitioning) for scalability
- Replication – HA (High Availability) and read scalability

# Overview: MongoDB Data Model

A MongoDB instance can have many databases

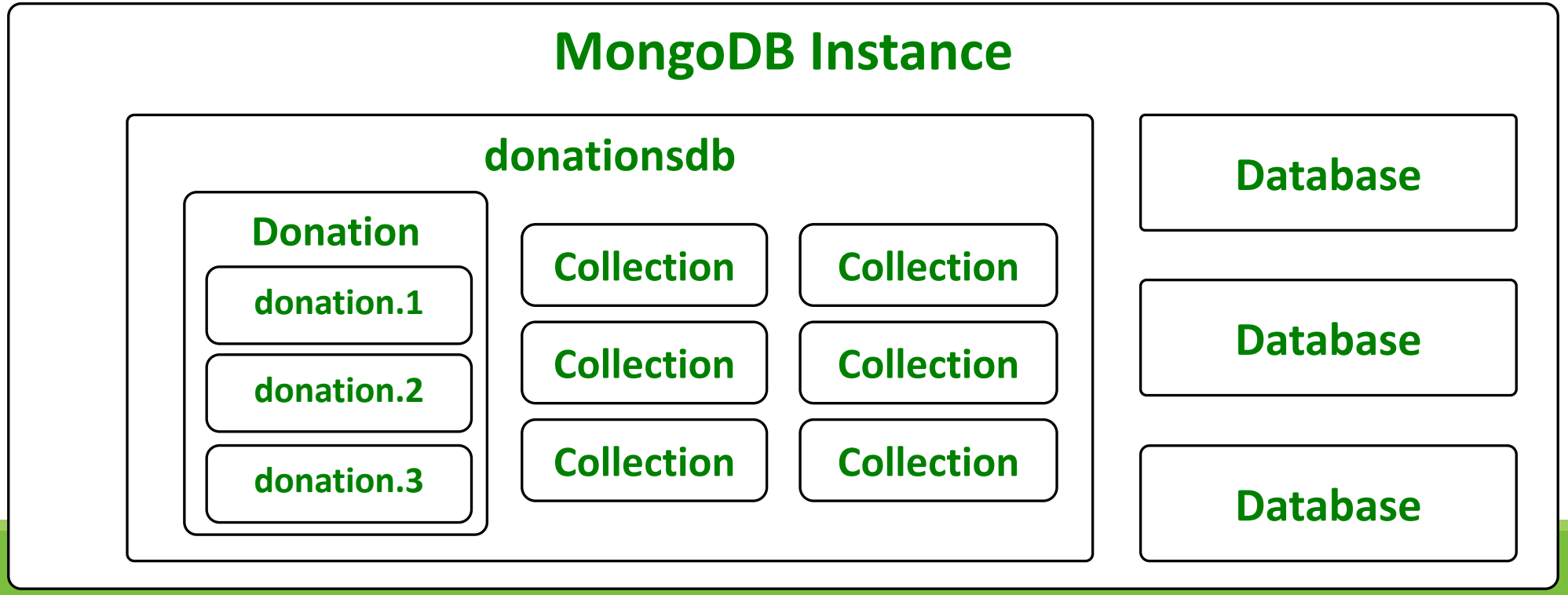
- A database can have many collections
  - A collection can have many documents



# Overview: MongoDB Data Model

A MongoDB instance can have many databases

- A database can have many collections
  - A collection can have many documents



# Overview: MongoDB Documents

## Documents in MongoDB are JSON objects

**Module Name: Databases**

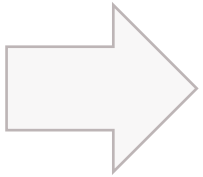
**Students: 250**

**Titles:**

- Intro to SQL
- NoSQL databases
- Entity Framework

**Evaluation:**

- Homework: 10
- Exam: 60
- Teamwork: 20
- Attendance: 10



```
{
 ”module_name": "Databases",
 "students": 250,
 "titles": [
   "Intro to SQL",
   "NoSQL databases",
   "DB Performance"
 ],
 "evaluation": {
   "homework": 10,
   "exam": 60,
   "teamwork": 20,
   "attendance": 10
}
```

# Overview: MongoDB Documents

MongoDB documents can have many types of values

- Numbers, Booleans
- Strings, Object Ids
- Dates
- Objects
  - Also know as nested documents
- Arrays

```
17, 3.14, true/false
```

```
"David Drohan"
```

```
2014-09-01T14:58:48.126Z
```

```
{
 "username": ”daveyd",
 "accessLevel": 2
}
```

```
['bananas', 'oranges']
```


{ name: mongo, type: DB }

# MongoDB Installation

INSTALLING & USING MONGODB

# MongoDB Installation

Download MongoDB from the official web site:

- https://www.mongodb.org/downloads
- Installers for all major platforms

How to start MongoDB?

- Go to installation folder and run **mongod**
- Paths need some configuration

```
$ cd C:\MongoDb\bin
```

  - You should specify databases path
  - or use default path and create folder **C:\data\db**

```
$ mongod --dbpath D:\mongodb\data
```

```
$ mongod
```

# How to use MongoDB

| | |
|---|---|
| Connect to a **mongodb** | `$ mongo` |
| Select a database | `$ use <database name>` |
| Show database name | `$ db` |
| Show all databases | `$ show dbs` |

# Authentication

How to create users in a database?

```
$ use ssd4db
$ db.createUser({
  "user": "daveyd",
  "pwd": "ssd412345",
  "roles": ["readWrite", "dbAdmin"]
})
```

Database user roles:

- **read**, **readWrite**, **dbAdmin**, **dbOwner**, **userAdmin**
- **userAdminAnyDatabase**

Login:

```
db.auth("daveyd", "ssd412345")
```


{ name: mongo, type: DB }

# MongoDB Queries

INSTALLING & USING MONGODB

# MongoDB Queries – Read

Read query

```
$ db.courses.find({ level: { $gt: 2 }}, { module_name: true }).limit(5);
```


Same as (in SQL):

```
SELECT module_name
FROM courses
WHERE level > 2
LIMIT 5
```

# MongoDB Queries – Read (2)

```
$ db.courses.find({
    module_name: ”Databases"
  });
```

```
$ db.courses.find({
    $or: [{ level: { $gt: 1 } }, { level: { $lt: 3 } }]
  });
```

Operators:

- Comparison – **$gt**, **$gte**, **$et**, **$lt**, **$lte**
- Logical – **$and**, **$nor**, **$not**, **$or**
- Element – **$exists**, **$type**

# MongoDB Queries – Insert

Insert query

```
$ db.courses.insert({
    module_name: "Databases",
    description: "Databases description",
    level: 3
  })
```

Same as (in SQL):

```
INSERT INTO courses (module_name, description, level)
VALUES ("Databases", "Databases description", 3)
```

# MongoDB Queries – Bulk Insert

Inserting bulk data in MongoDB

```
var bulk = db.courses.initializeUnorderedBulkOp();
bulk.insert( { name: "ASP.NET MVC", level: 3 } );
bulk.insert( { item: "Web Development", level: 3 } );
bulk.insert( { item: "SPA Applications", level: 2 } );
bulk.execute();
```

# MongoDB Queries – Update

Update query

```
$ db.courses.update(
  { module_name: { $et: "Databases" }},
  { $set: { description: "Concepts, MSSQL, MySQL, Oracle"}},
  { multi: true }
);
```

Same as (in SQL):

```
UPDATE courses
SET description = "Concepts, MSSQL, MySQL, Oracle"
WHERE module_name = "Databases"
```

# MongoDB Queries – Delete

Delete query

```
$ db.courses.remove({ module_name: "Databases" });
```

Same as (in SQL):

```
DELETE FROM courses
WHERE module_name = "Databases"
```

# MongoDB Queries – Aggregation

Aggregation is operation that process data records and return computed results

```
$ db.courses.aggregate([
  { $match: { isActive: true } },
  { $group: { level: "$level", students: { $sum: "$studentsCount" }} }
])
```

Result:

```
[
  { level: 1, students: 460 },
  { level: 2, students: 350 },
  { level: 3, students: 200 }
]
```

# MongoDB Management Tools

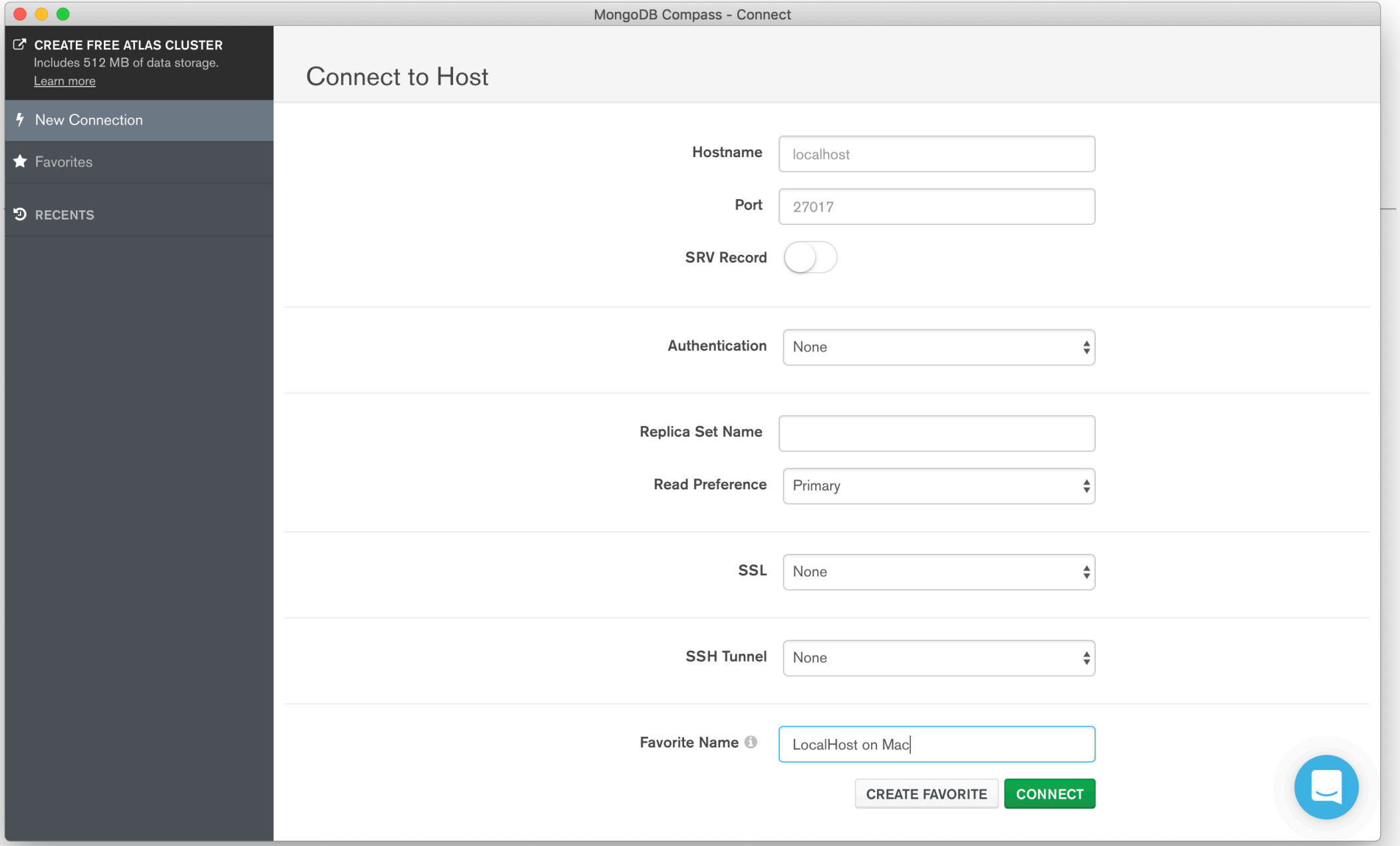
MongoDB is an open-source DB system

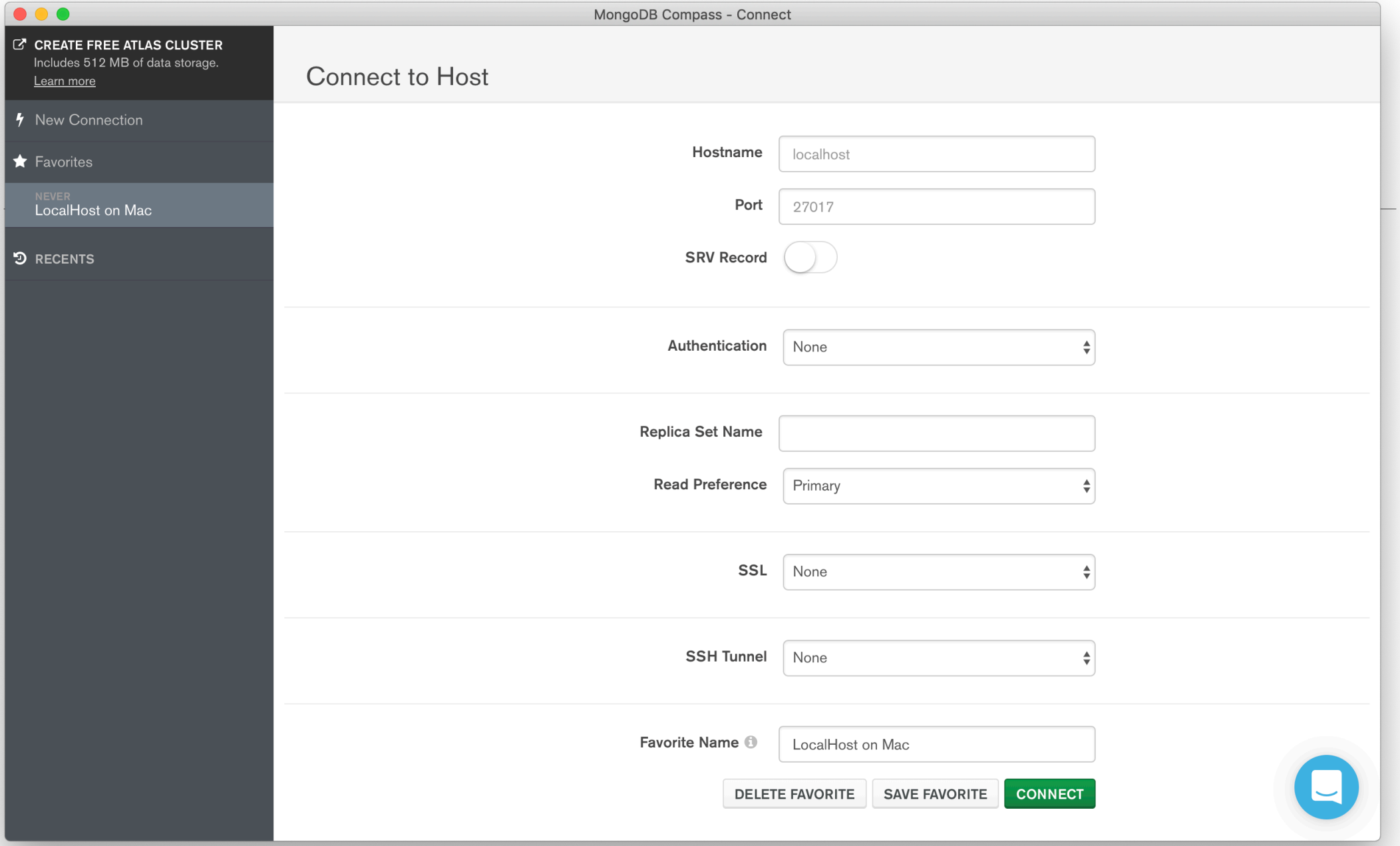
- So there are many available viewers

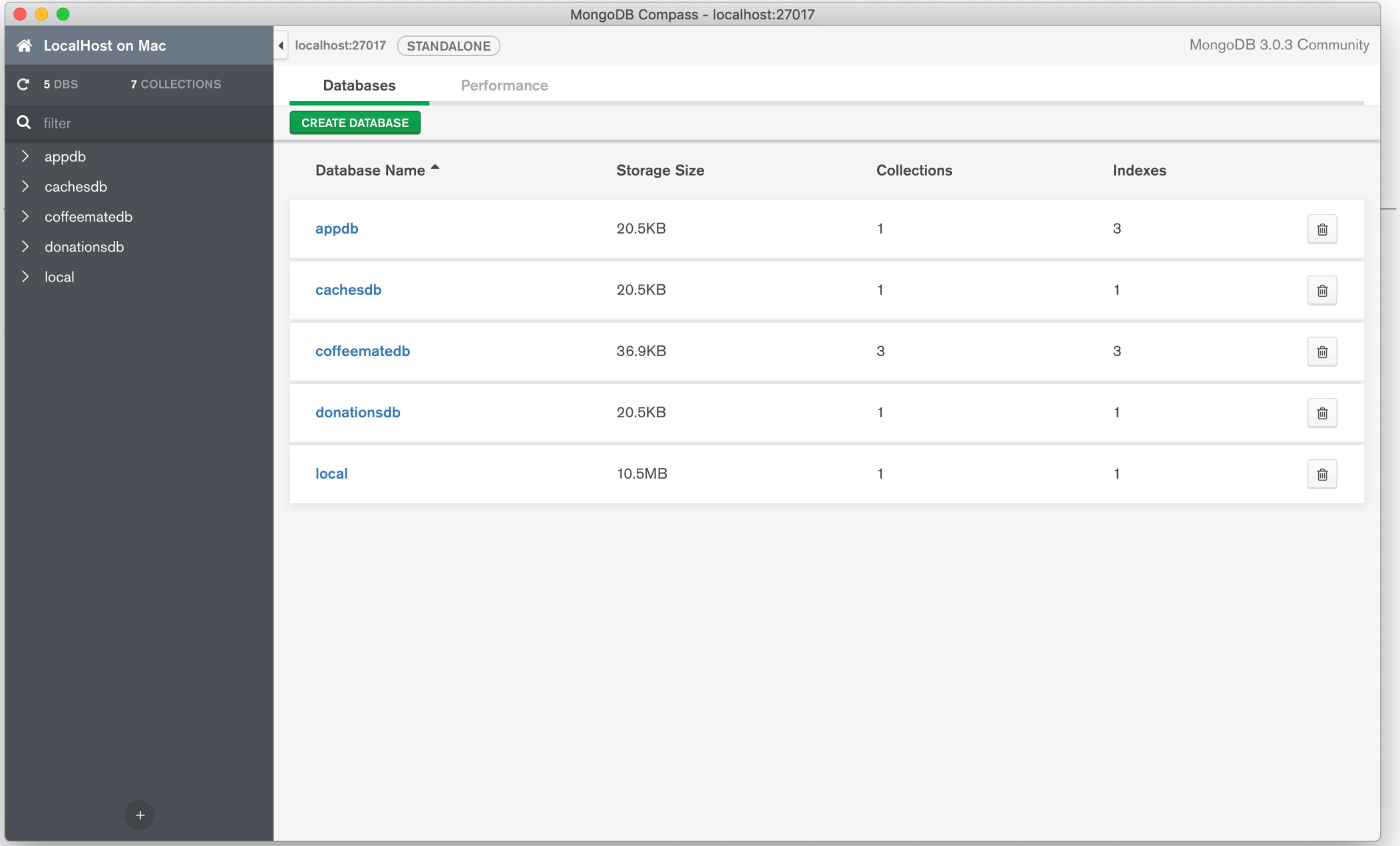
Some, but not all are:

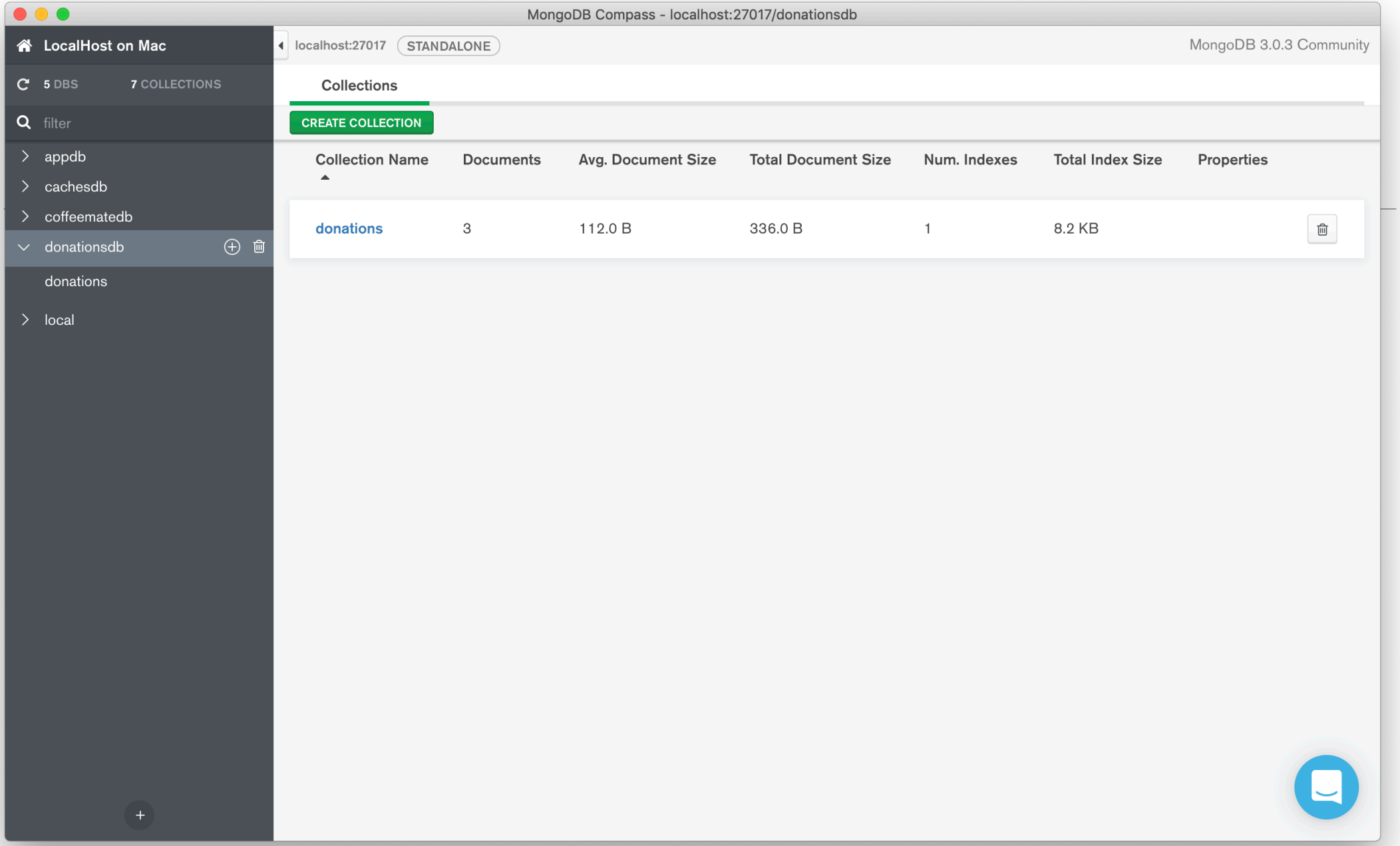
- **MongoDB CLI**
  - Comes with installation of MongoDB
  - Execute queries inside the CMD/Terminal
- **MongoDB Compass, MongoVUE, UMongo & Robomongo**
  - Provide UI to view, edit are remove DB documents
  - Execute queries inside the tool

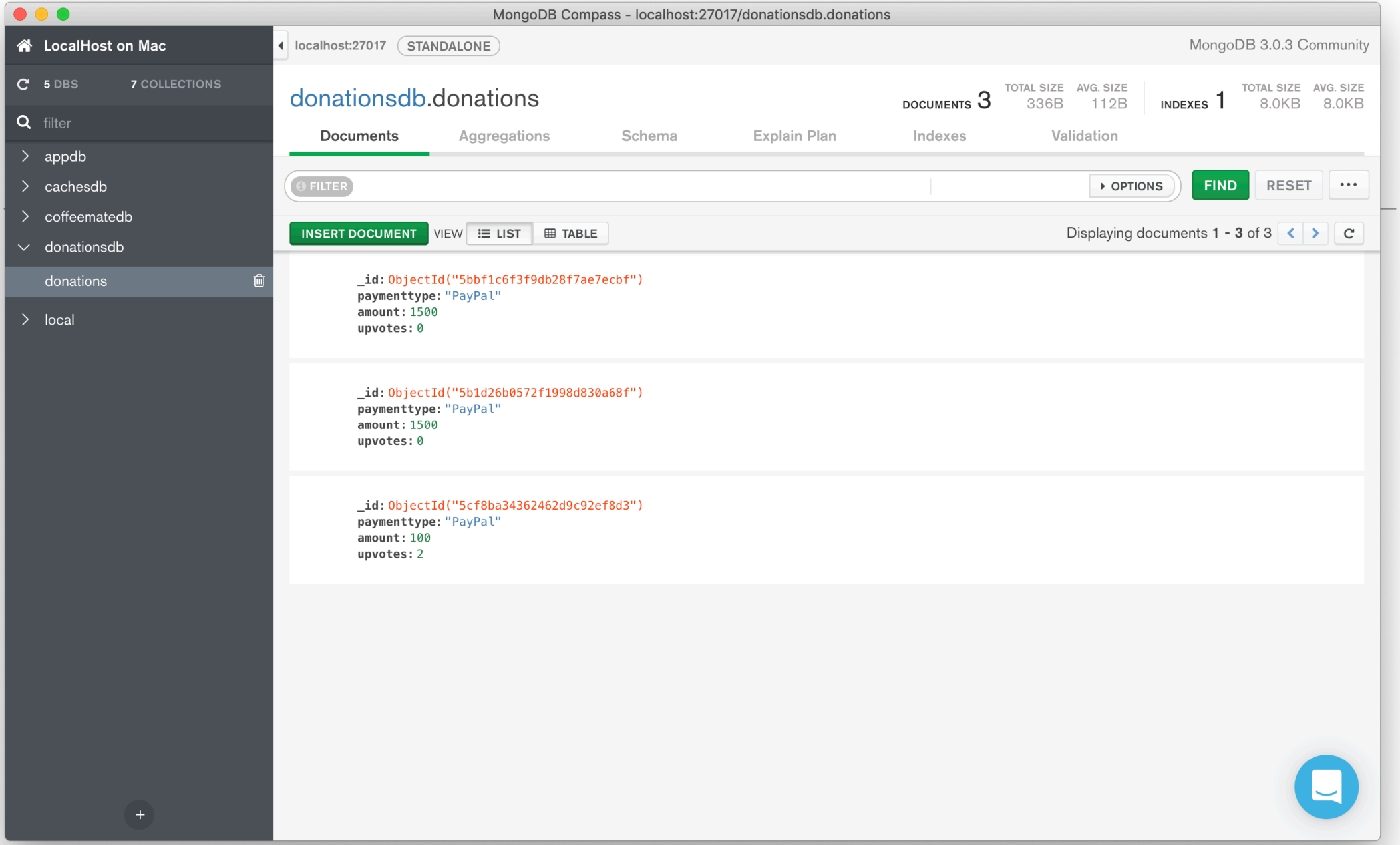

# MongoDB Compass

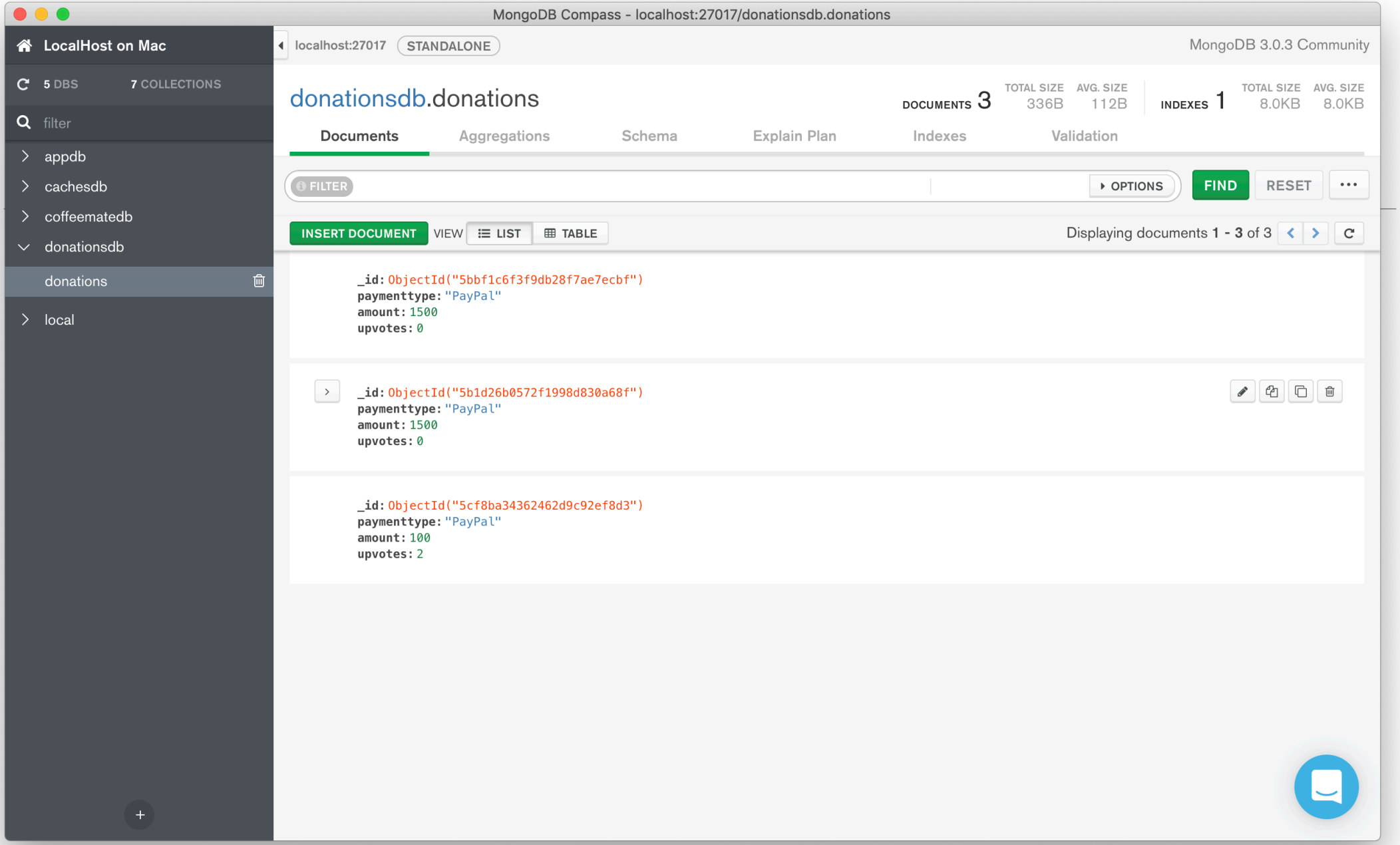

MongoDB Compass - Connect
CREATE FREE ATLAS CLUSTER
Includes 512 MB of data storage.
Learn more
New Connection
Favorites
RECENTS
Connect to Host
Hostname
localhost
Port
27017
SRV Record
Authentication
None
Replica Set Name
Read Preference
Primary
SSL
None
SSH Tunnel
None
Favorite Name
LocalHost on Mac
CREATE FAVORITE
CONNECT

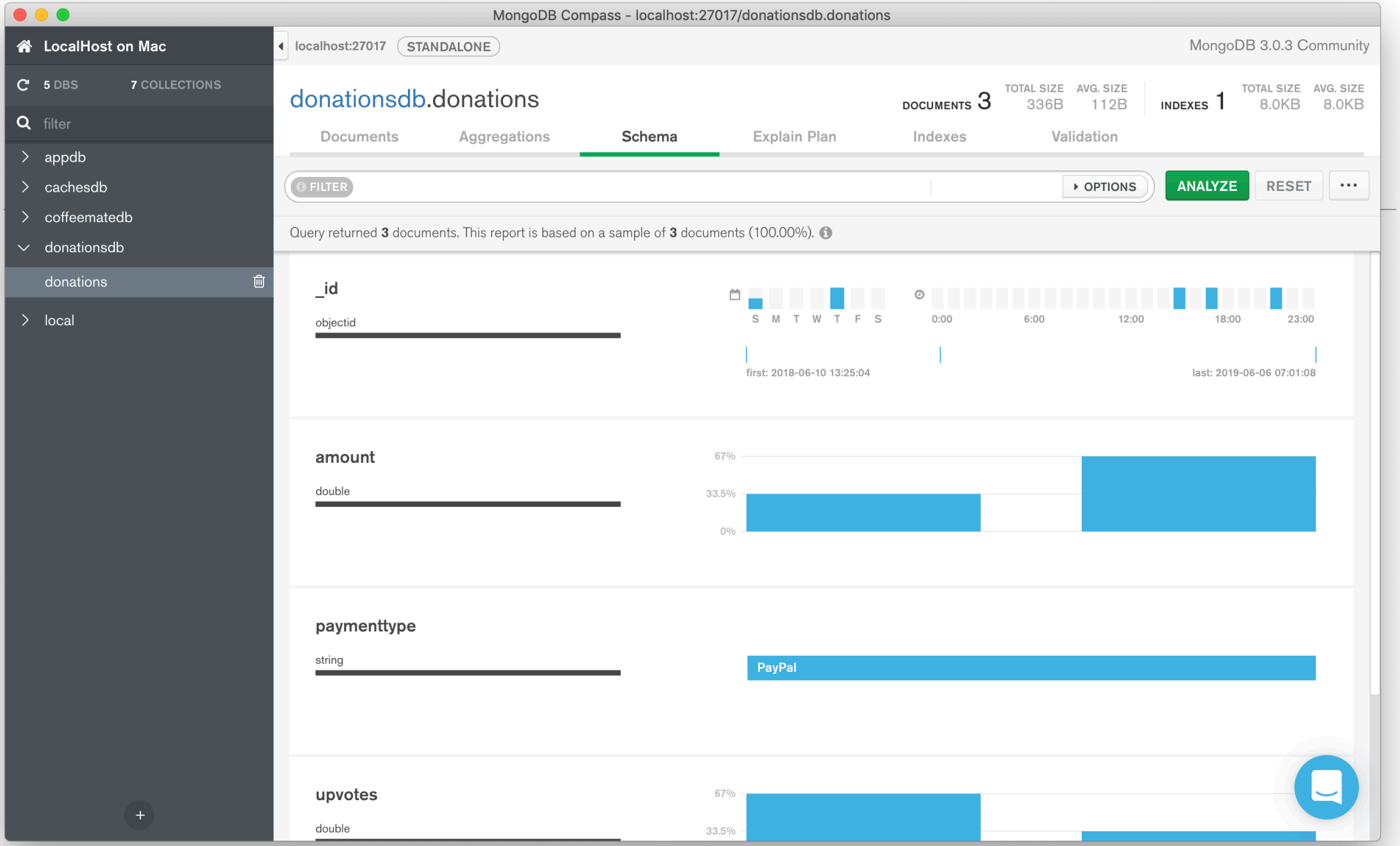

MongoDB Compass - Connect
CREATE FREE ATLAS CLUSTER
Includes 512 MB of data storage.
Learn more
New Connection
Favorites
NEVER
LocalHost on Mac
RECENTS
Connect to Host
Hostname
localhost
Port
27017
SRV Record
Authentication
None
Replica Set Name
Read Preference
Primary
SSL
None
SSH Tunnel
None
Favorite Name
LocalHost on Mac
DELETE FAVORITE
SAVE FAVORITE
CONNECT

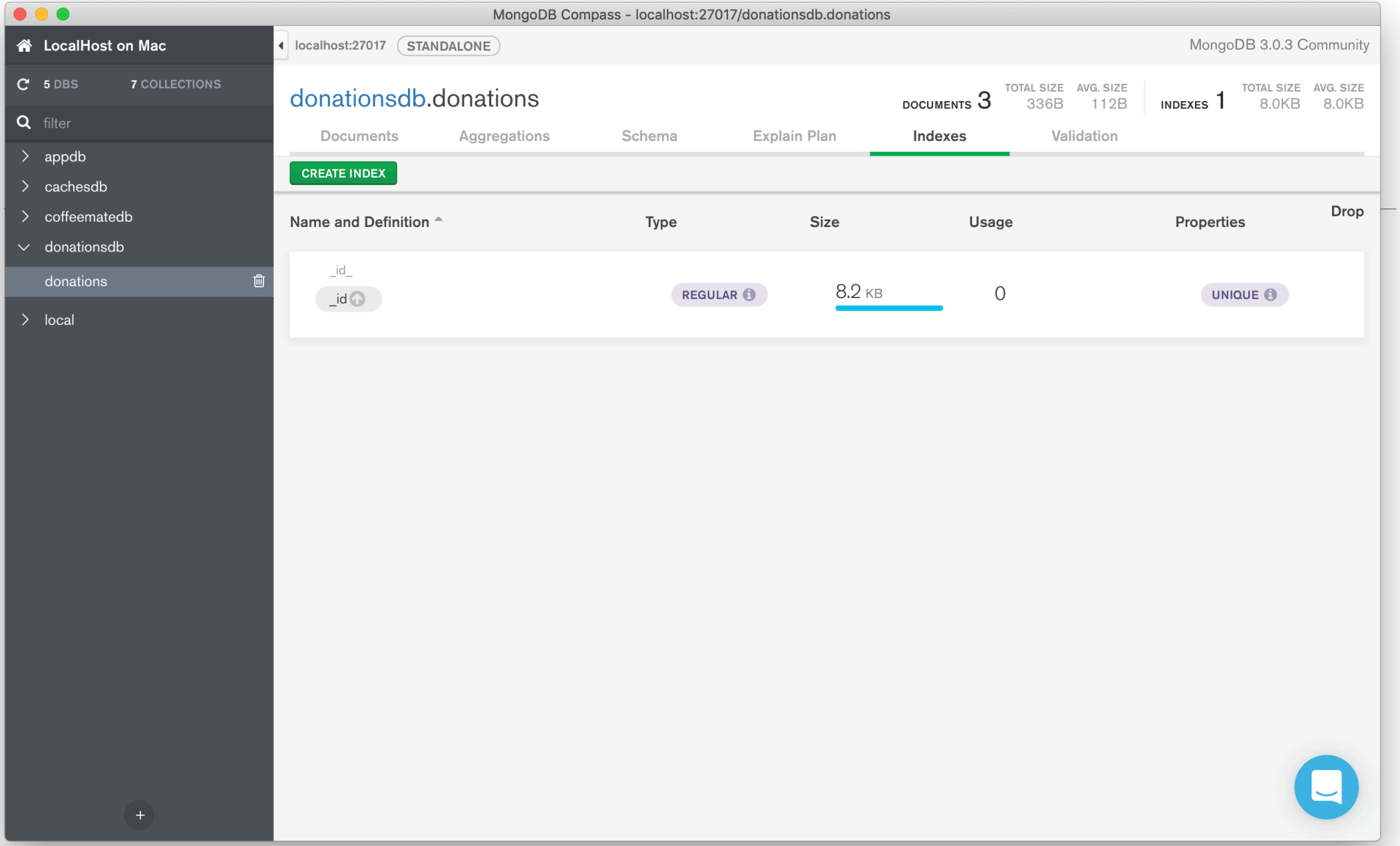

MongoDB Compass - localhost:27017
LocalHost on Mac
5 DBS
7 COLLECTIONS
filter
appdb
cachesdb
coffeematedb
donationsdb
local
localhost:27017
STANDALONE
MongoDB 3.0.3 Community
Databases
Performance
CREATE DATABASE
Database Name
Storage Size
Collections
Indexes
appdb 20.5KB 1 3
cachesdb 20.5KB 1 1
coffeematedb 36.9KB 3 3
donationsdb 20.5KB 1 1
local 10.5MB 1 1


MongoDB Compass - localhost:27017/donationsdb
LocalHost on Mac
5 DBS
7 COLLECTIONS
filter
appdb
cachesdb
coffeematedb
donationsdb
donations
local
localhost:27017
STANDALONE
MongoDB 3.0.3 Community
Collections
CREATE COLLECTION
Collection Name
Documents
Avg. Document Size
Total Document Size
Num. Indexes
Total Index Size
Properties
donations
3
112.0 B
336.0 B
1
8.2 KB

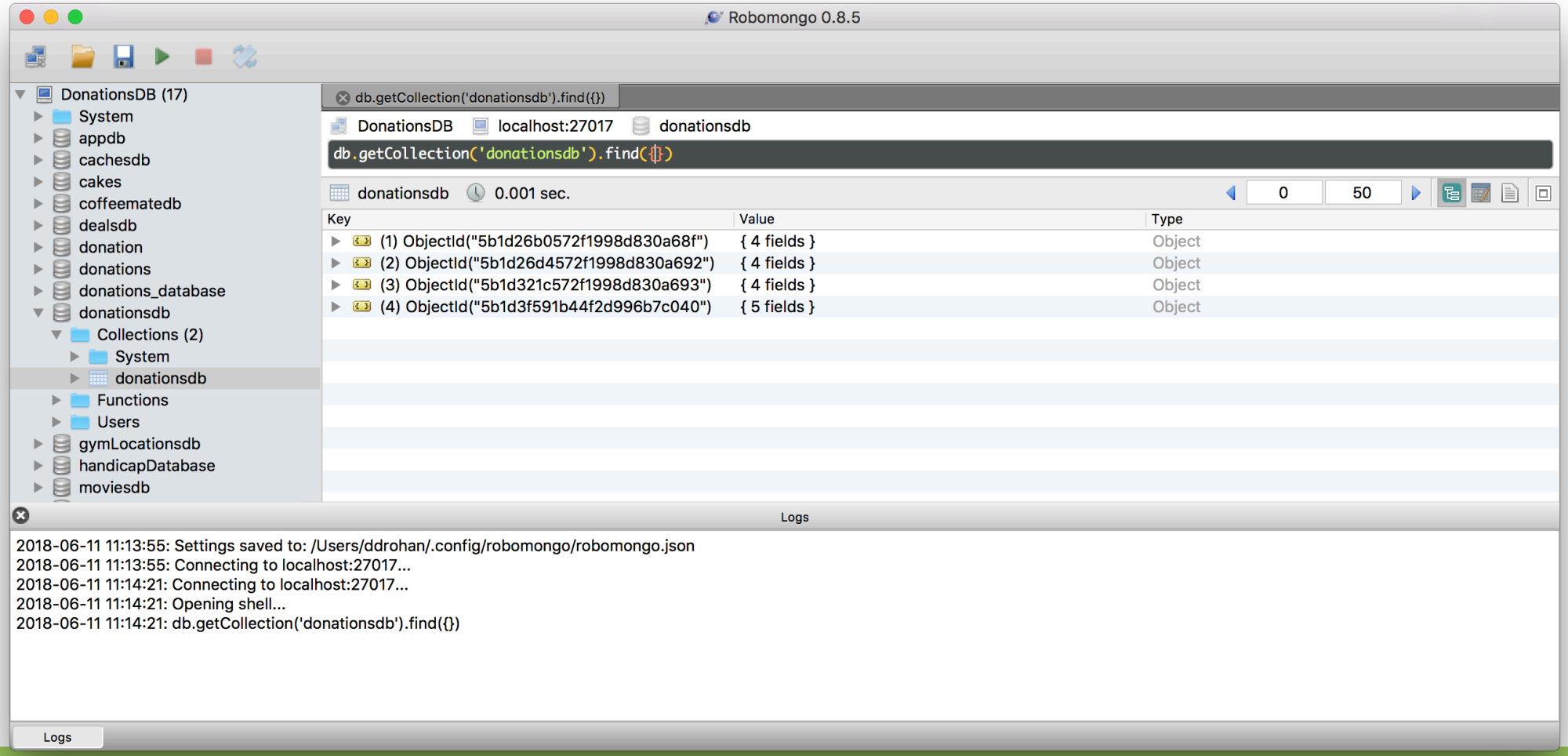

MongoDB Compass - localhost:27017/donationsdb.donations
LocalHost on Mac
5 DBS
7 COLLECTIONS
filter
appdb
cachesdb
coffeematedb
donationsdb
donations
local
localhost:27017
STANDALONE
MongoDB 3.0.3 Community
donationsdb.donations
DOCUMENTS 3
TOTAL SIZE 336B
AVG. SIZE 112B
INDEXES 1
TOTAL SIZE 8.0KB
AVG. SIZE 8.0KB
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
OPTIONS
FIND
RESET
INSERT DOCUMENT
VIEW
LIST
TABLE
Displaying documents 1 - 3 of 3
_id: ObjectId("5bbf1c6f3f9db28f7ae7ecbf")
paymenttype: "PayPal"
amount: 1500
upvotes: 0
_id: ObjectId("5b1d26b0572f1998d830a68f")
paymenttype: "PayPal"
amount: 1500
upvotes: 0
_id: ObjectId("5cf8ba34362462d9c92ef8d3")
paymenttype: "PayPal"
amount: 100
upvotes: 2


MongoDB Compass - localhost:27017/donationsdb.donations
LocalHost on Mac
5 DBS
7 COLLECTIONS
filter
appdb
cachesdb
coffeematedb
donationsdb
donations
local
localhost:27017
STANDALONE
MongoDB 3.0.3 Community
donationsdb.donations
DOCUMENTS 3
TOTAL SIZE 336B
AVG. SIZE 112B
INDEXES 1
TOTAL SIZE 8.0KB
AVG. SIZE 8.0KB
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
OPTIONS
FIND
RESET
INSERT DOCUMENT
VIEW
LIST
TABLE
Displaying documents 1 - 3 of 3
_id: ObjectId("5bbf1c6f3f9db28f7ae7ecbf")
paymenttype: "PayPal"
amount: 1500
upvotes: 0
_id: ObjectId("5b1d26b0572f1998d830a68f")
paymenttype: "PayPal"
amount: 1500
upvotes: 0
_id: ObjectId("5cf8ba34362462d9c92ef8d3")
paymenttype: "PayPal"
amount: 100
upvotes: 2

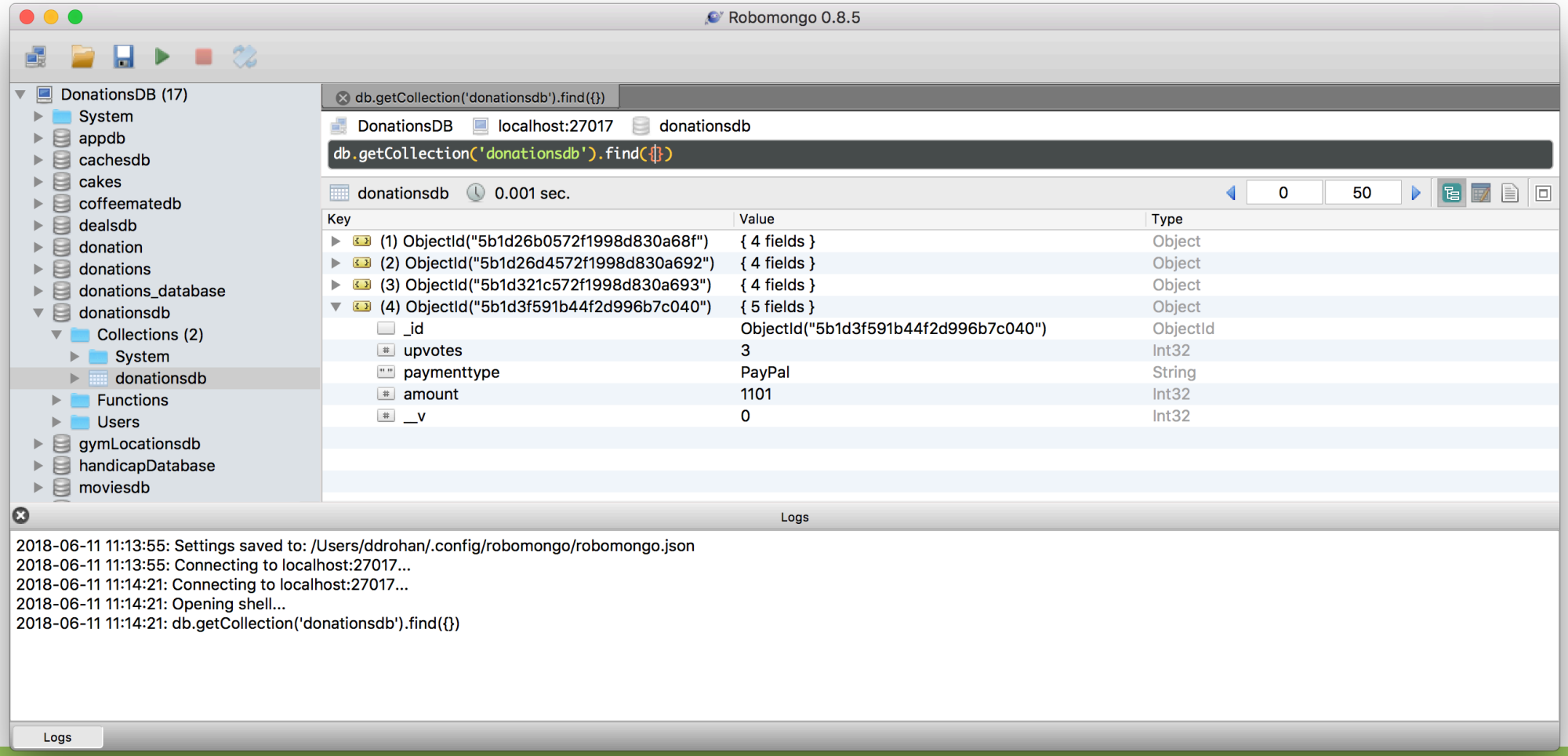

MongoDB Compass - localhost:27017/donationsdb.donations
LocalHost on Mac
5 DBS
7 COLLECTIONS
filter
appdb
cachesdb
coffeematedb
donationsdb
donations
local
localhost:27017
STANDALONE
MongoDB 3.0.3 Community
donationsdb.donations
DOCUMENTS 3
TOTAL SIZE 336B
AVG. SIZE 112B
INDEXES 1
TOTAL SIZE 8.0KB
AVG. SIZE 8.0KB
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER
OPTIONS
ANALYZE
RESET
Query returned 3 documents. This report is based on a sample of 3 documents (100.00%).
_id
objectid
S M T W T F S
0:00
6:00
12:00
18:00
23:00
first: 2018-06-10 13:25:04
last: 2019-06-06 07:01:08
amount
double
67%
33.5%
0%
paymenttype
string
PayPal
upvotes
double
67%
33.5%


MongoDB Compass - localhost:27017/donationsdb.donations
LocalHost on Mac
5 DBS
7 COLLECTIONS
filter
appdb
cachesdb
coffeematedb
donationsdb
donations
local
localhost:27017
STANDALONE
MongoDB 3.0.3 Community
donationsdb.donations
DOCUMENTS 3
TOTAL SIZE 336B
AVG. SIZE 112B
INDEXES 1
TOTAL SIZE 8.0KB
AVG. SIZE 8.0KB
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
CREATE INDEX
Name and Definition
Type
Size
Usage
Properties
Drop
_id_
_id
REGULAR
8.2 KB
0
UNIQUE

# MongoDB - Robomongo



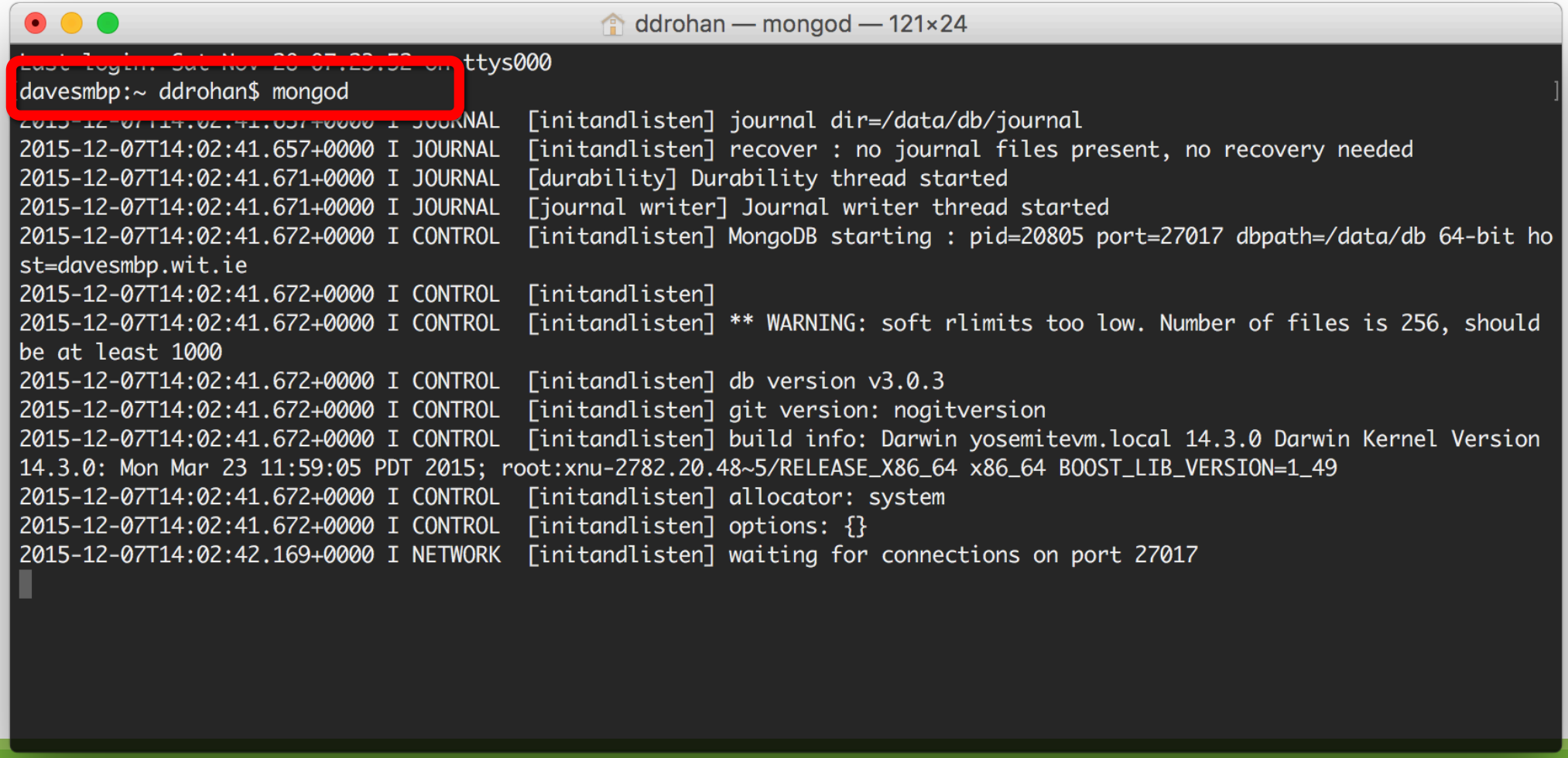
# MongoDB - Robomongo



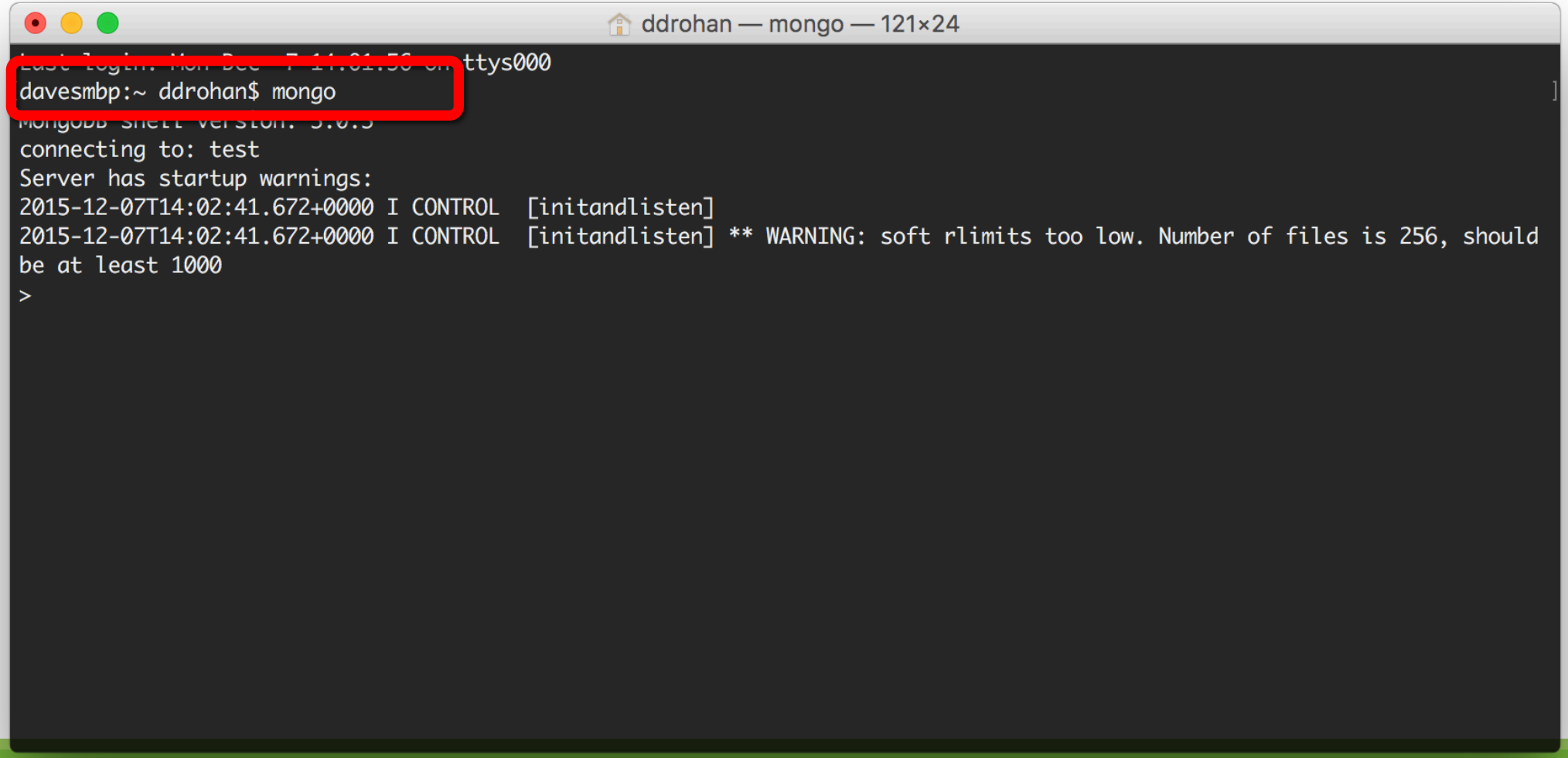

# Donationweb

CASE STUDY EXAMPLE

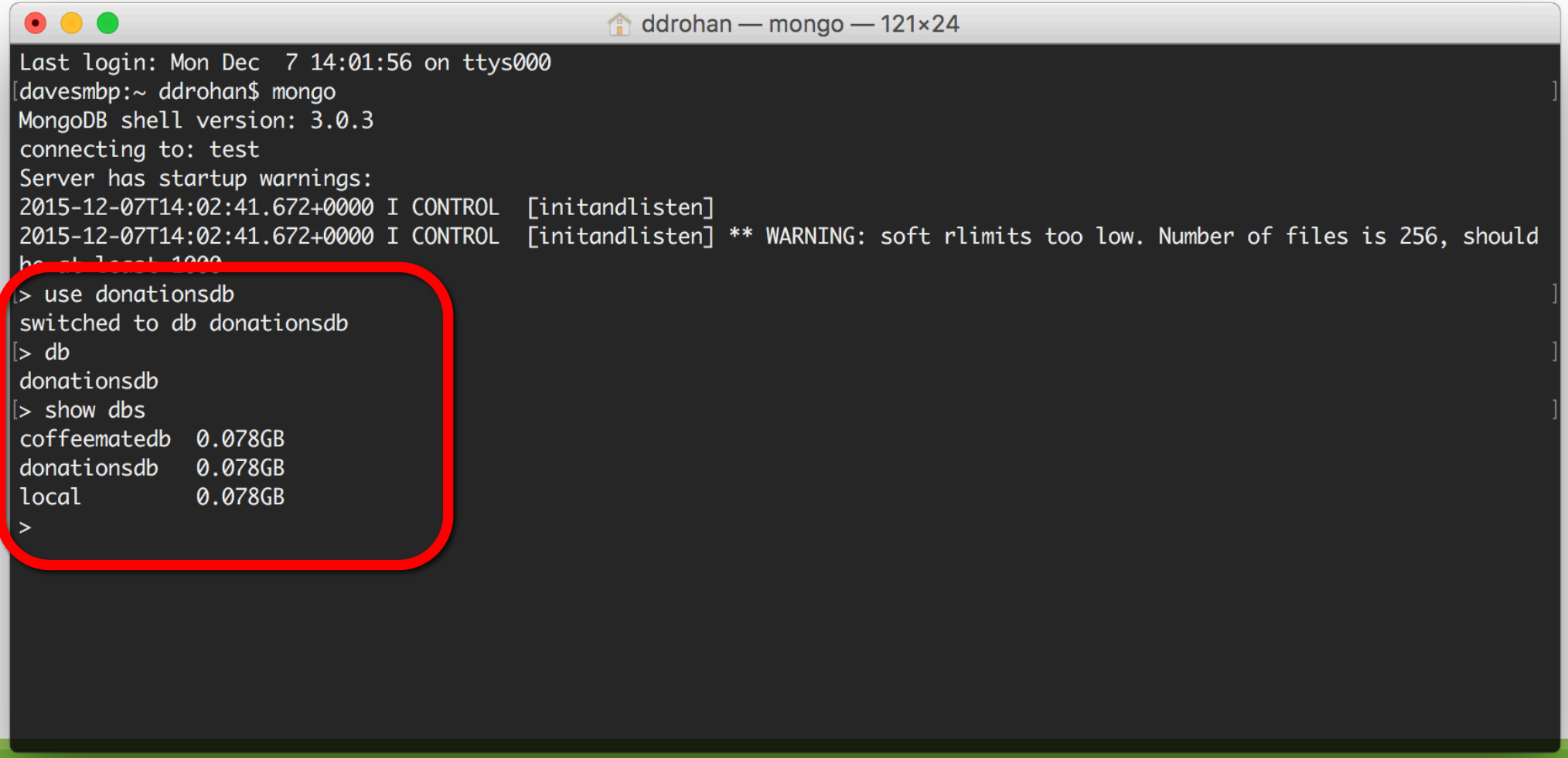
# Launch the MongoDB Server



# Launch the MongoDB Client



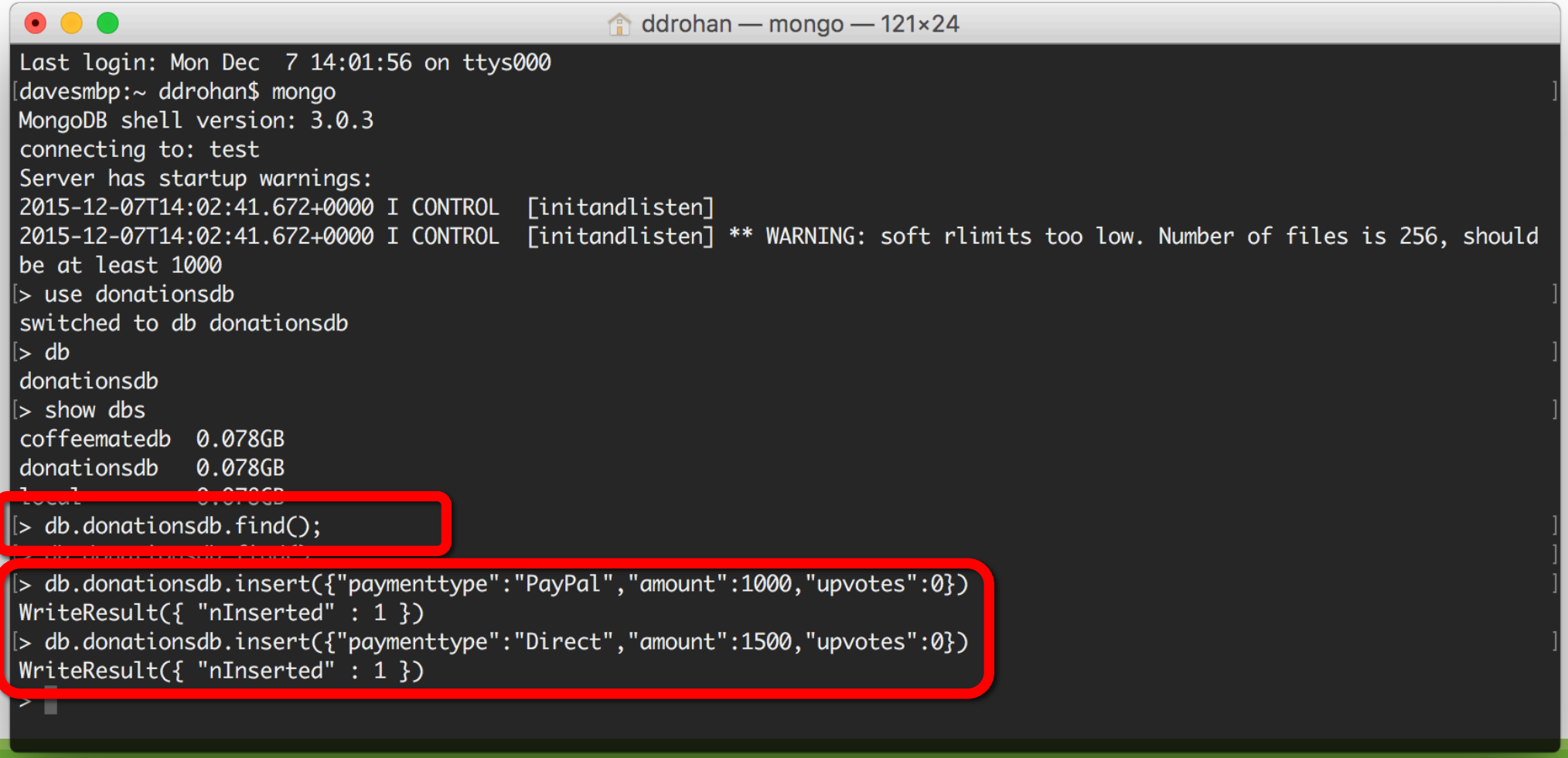
# Create/Switch to chosen Database



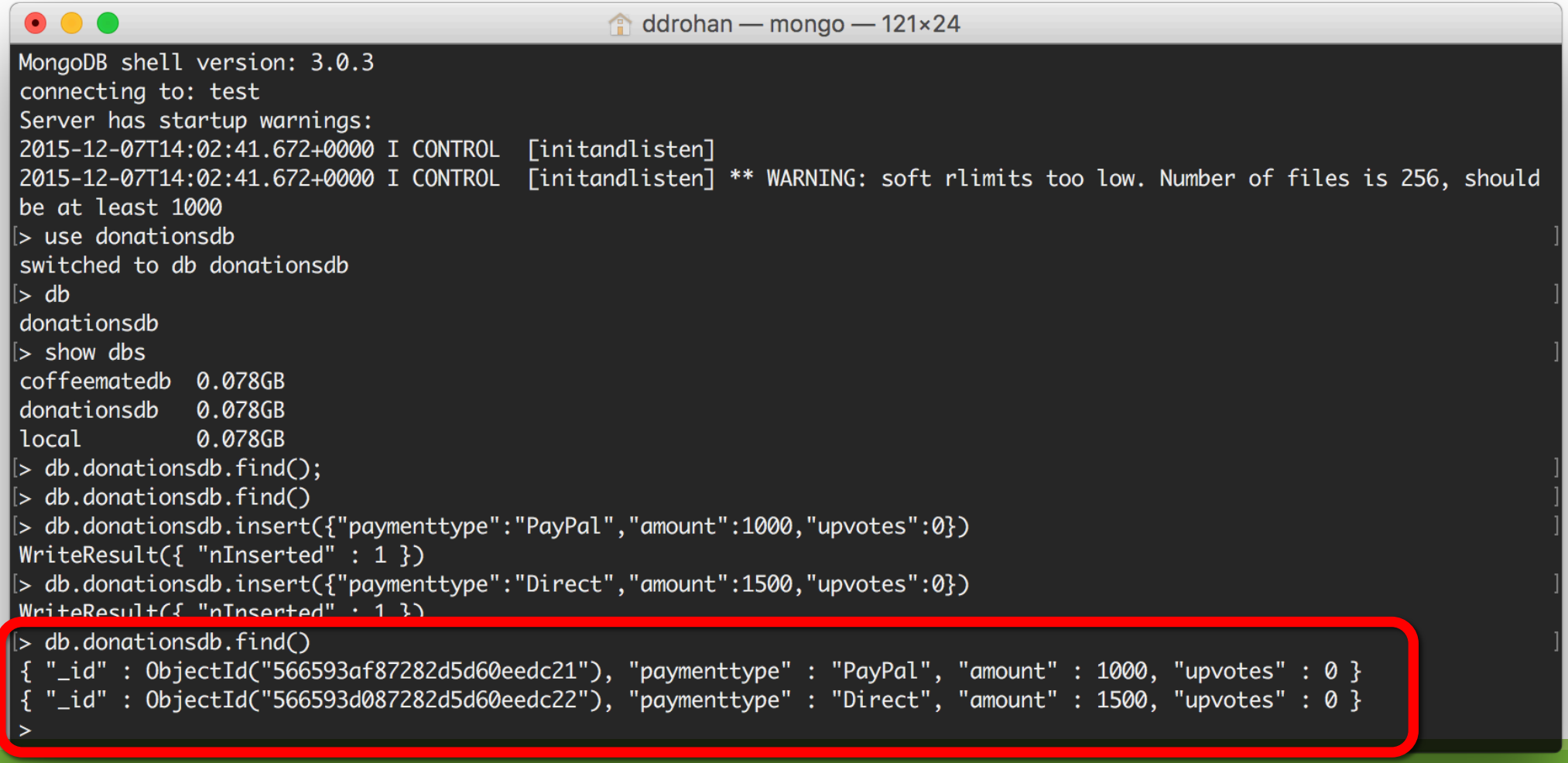
mongoose

mongoDB

# Search / Insert Records



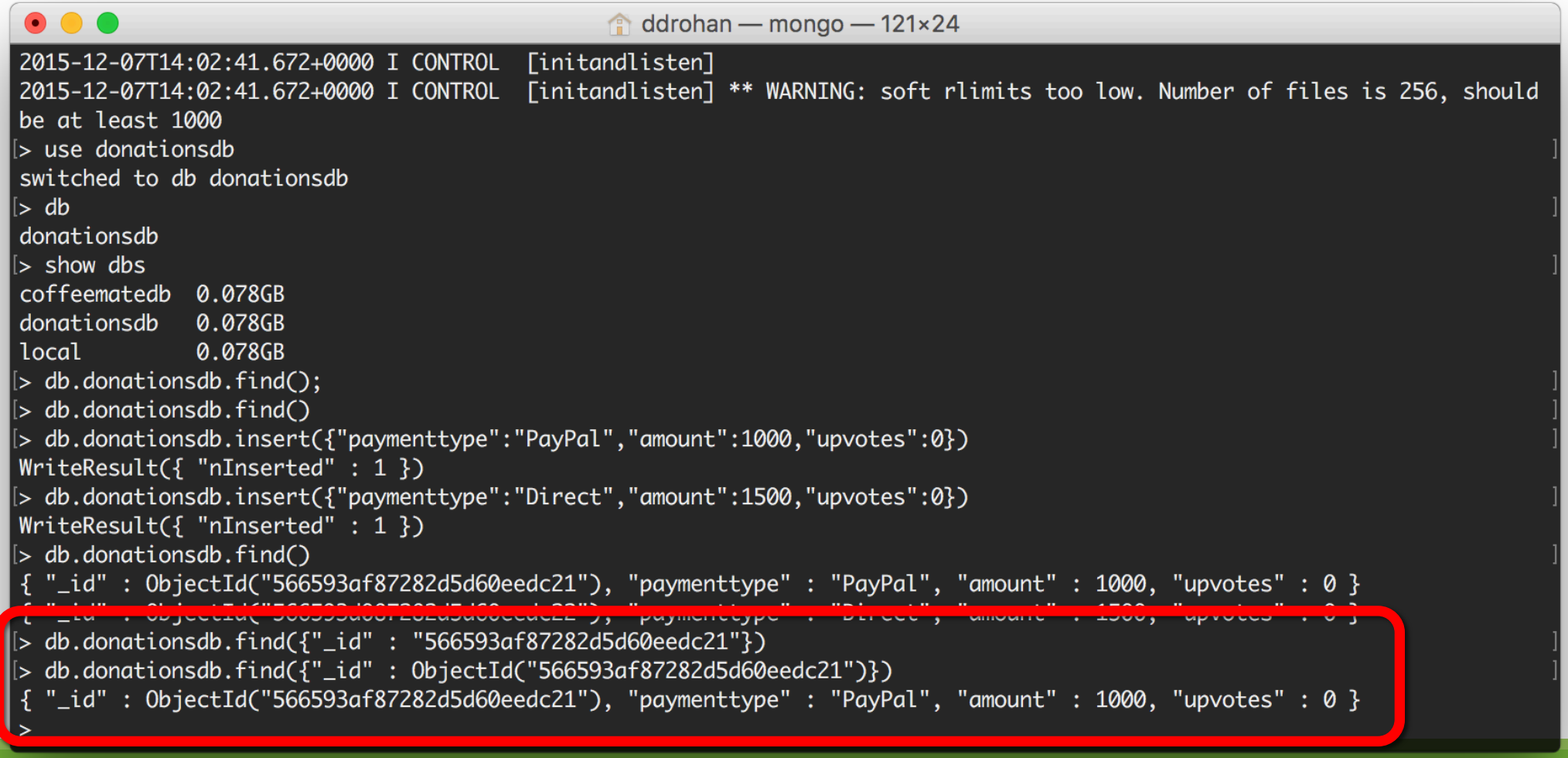
# Find All Records



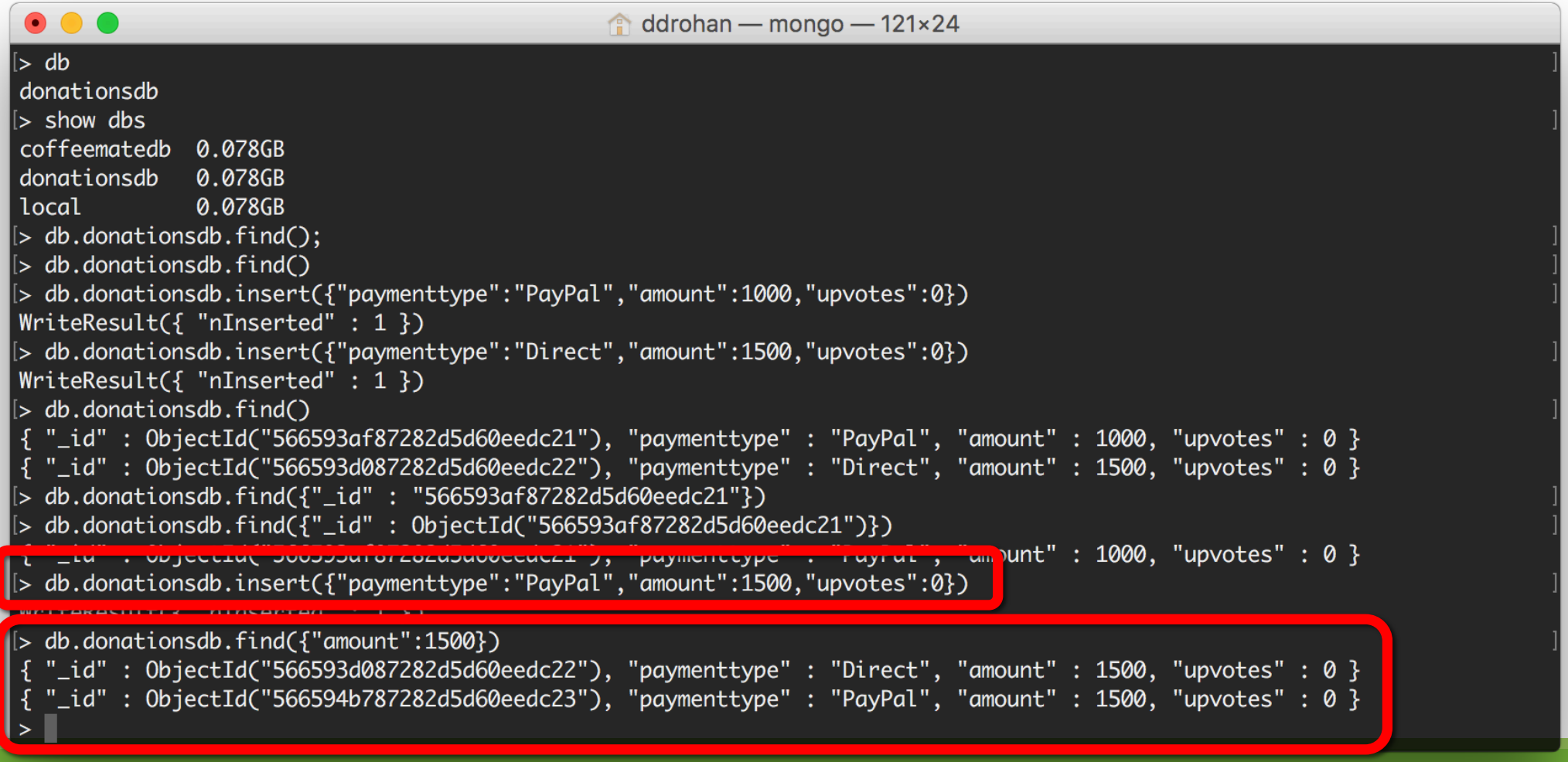
mongoose

mongoDB

# Find Records (with criteria)



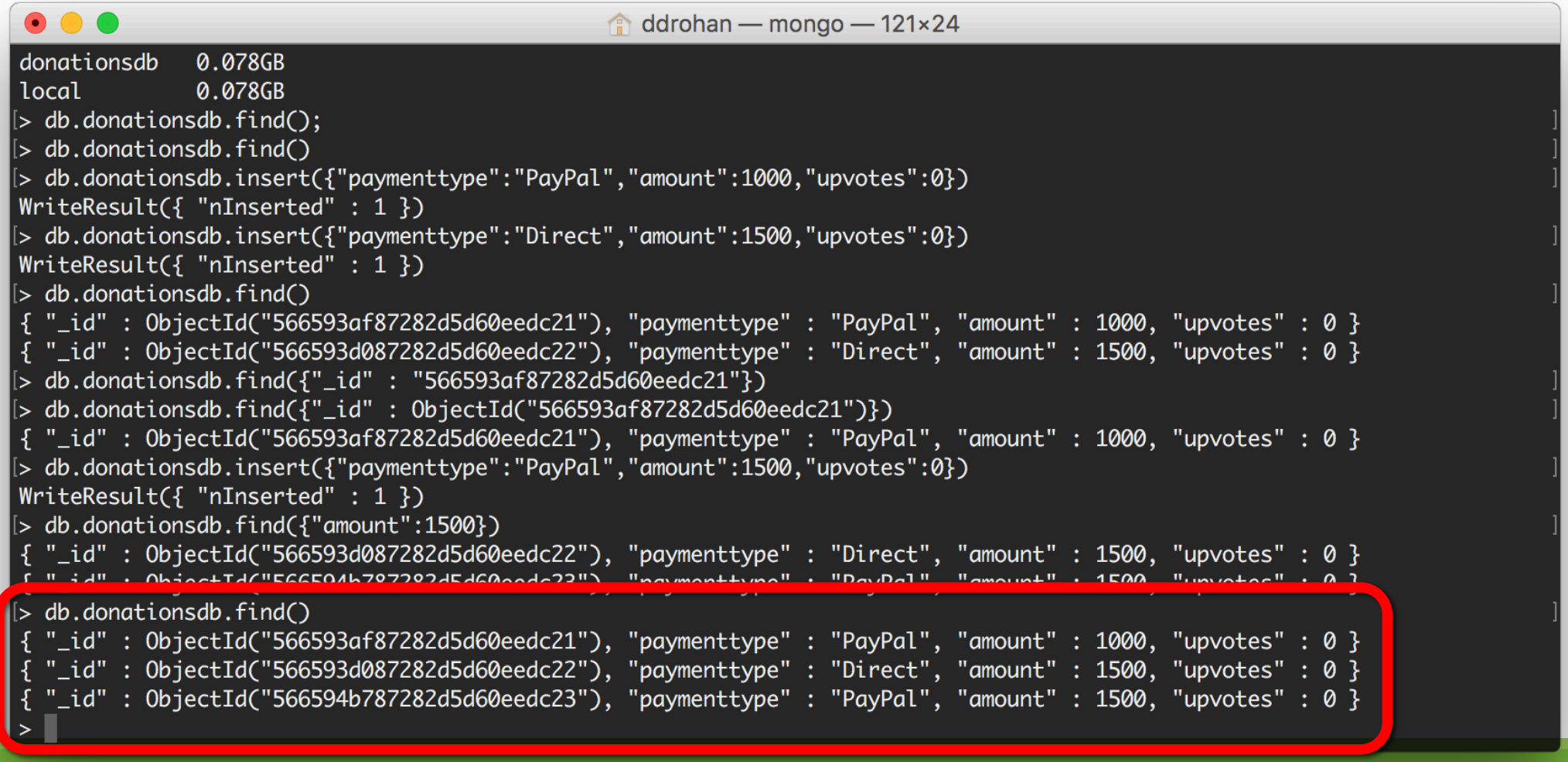
# Insert / Find Records (with criteria)



mongoose

mongoDB

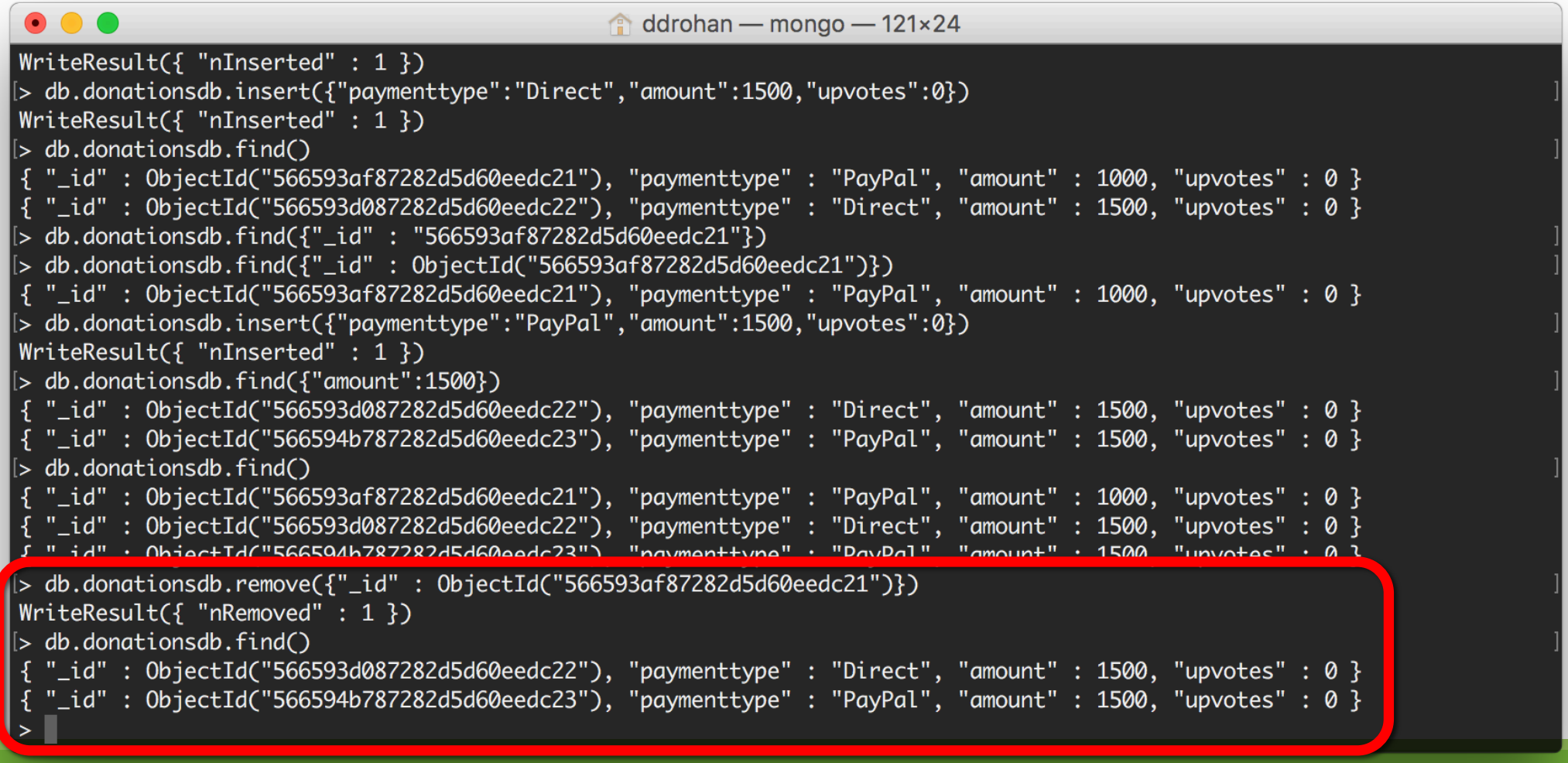
# Find all Records



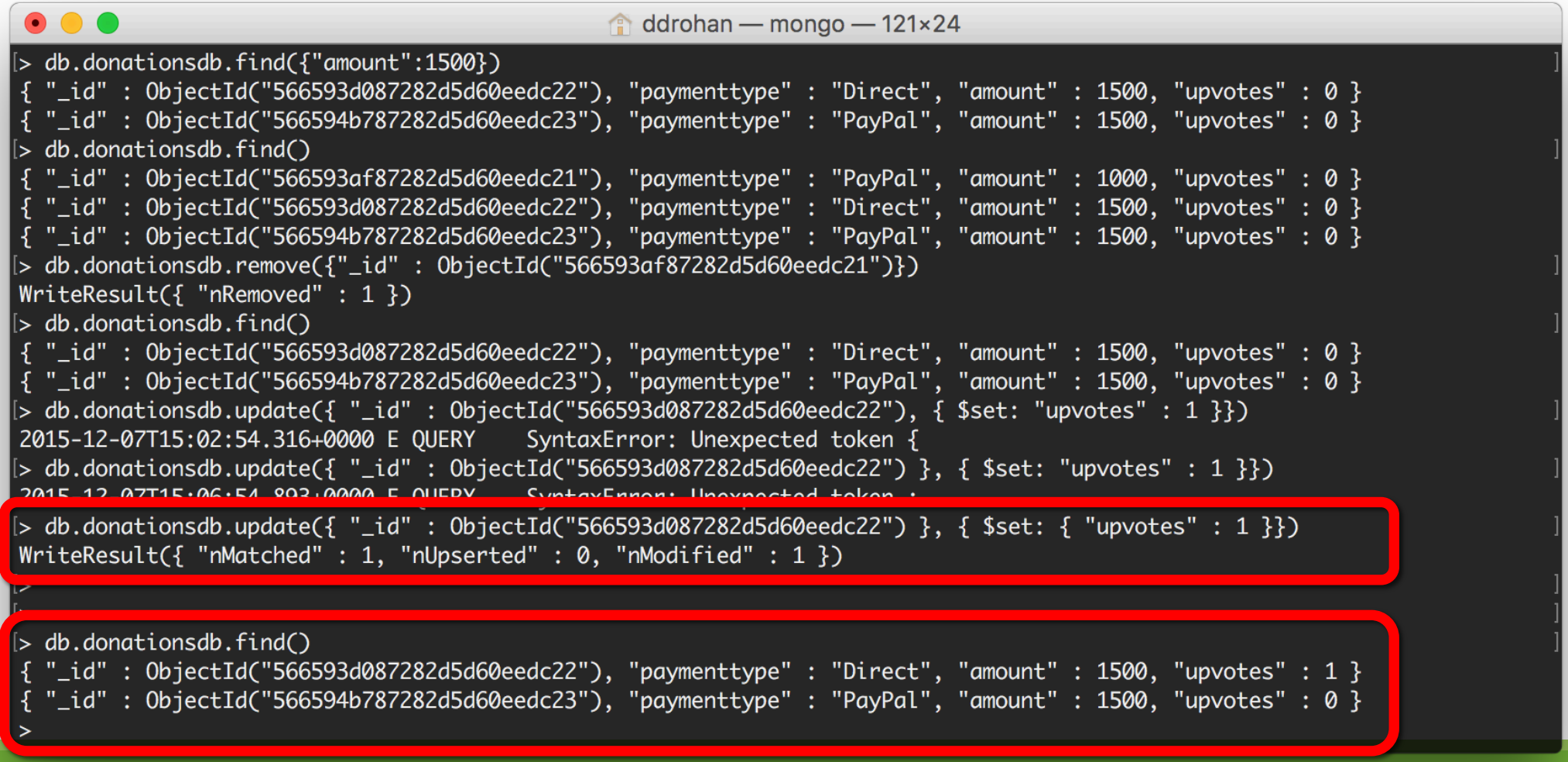
mongoose

mongoDB

# Delete / Remove a Record



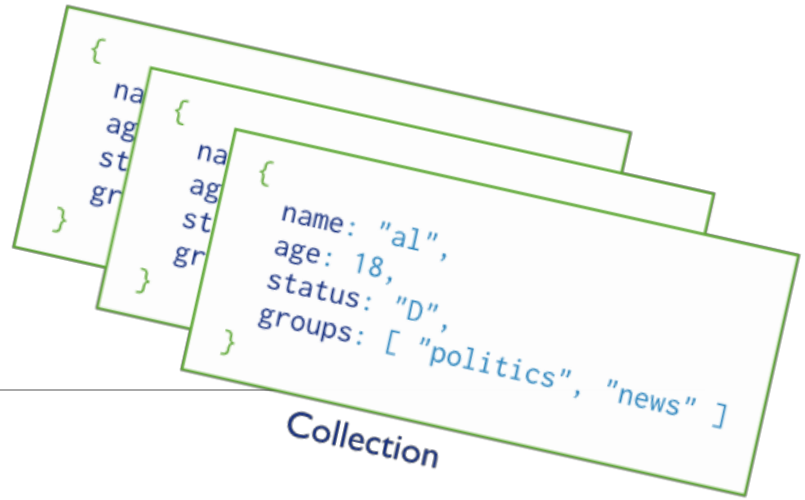
# Update a Record

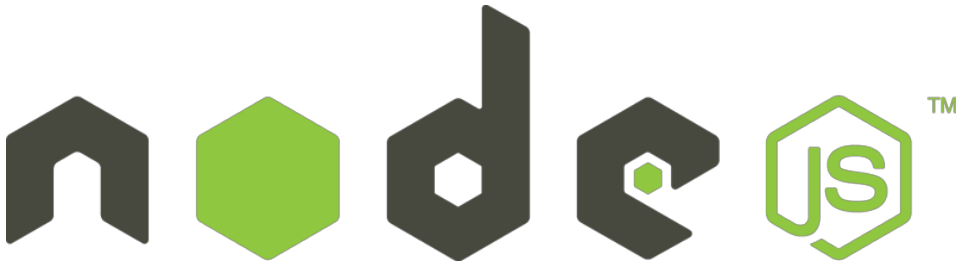



{ name: mongo, type: DB }

# MongoDB & Node.js

OBJECT-DOCUMENT MODEL



# MongoDB Queries

The MongoDB **module** supports all kinds of queries over the data

- Creating new documents
  - And adding records
- Editing existing documents
  - And their records
- Removing documents and records
- Querying whole documents or parts of them

# Using MongoDB with Node.js

Once installed, MongoDB must be started before it can be used

- Go to installation folder and run mongod

```
$ cd path/to/mondodb/installation/folder
$ mongod
```

- Or add mongod.exe to the PATH

When run, MongoDB can be used from **Node.js**

# Using MongoDB with Node.js

Download MongoDB from the official web site:

- https://www.mongodb.org/downloads
- Installers for all major platforms

When installed, MongoDB needs a driver to be usable with a specific platform

- One to use with Node.js, another to use with .NET, etc…

Installing MongoDB dependency module for **Node.js:**

```
$ npm install mongodb -g
```

# MongoDB Module & Node.js

- The 'mongodb' module is required

```
var mongodb = require('mongodb');
```

- Create a server to host the database

```
var server = new mongodb.Server('localhost', 27017);
```

- Create mongodb client that connects to the server

```
var mongoClient = new mongodb.MongoClient(server);
```

- Open connection to the mongodb server

```
mongoClient.open(function(err, client){
  var db = client.db('DATABASE_NAME');
  //queries over the db
});
```

mongoose

elegant mongodb object modeling for node.js

# Mongoose Overview

OBJECT-DOCUMENT MODEL MODULE FOR NODE.JS

# Mongoose Overview

Mongoose is a object-document model module in Node.js for MongoDB

- Wraps the functionality of the native MongoDB driver
- Exposes models to control the records in a doc
- Supports validation on save
- Extends the native queries

# Installing Mongoose

Run the following from the CMD/Terminal

```
$ npm install mongoose
```

Load the Module

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

Connect to the Database

```
mongoose.connect(mongoDbPath);
```

Create Models and persist data

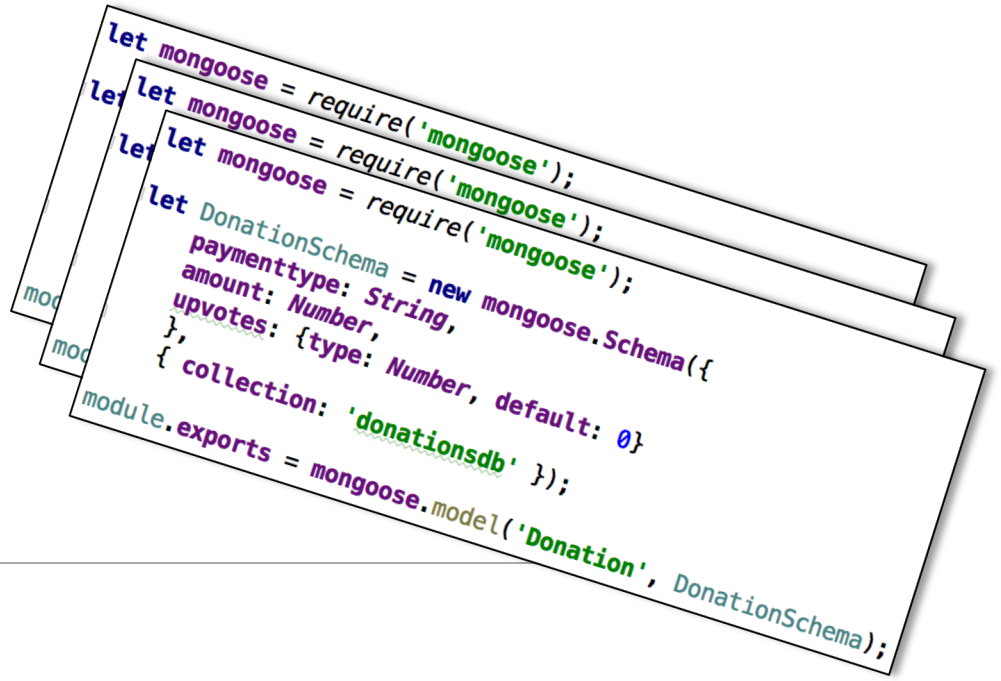
```
var Unit = mongoose.model('Unit', { type: String} );
new Unit({type: 'warrior'}).save(callback); //create
Unit.find({type: 'warrior'}).exec(callback); //fetch
```

# Mongoose Models

OBJECT-DOCUMENT MODEL SCHEMA

mongoose

elegant mongodb object modeling for node.js



# Mongoose Models

- Mongoose supports Models
  - i.e. fixed types of documents
- Used like object Constructors
  - Needs a ‘mongoose.Schema’

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

let DonationSchema = new mongoose.Schema({
    paymenttype: String,
    amount: Number,
    upvotes: {type: Number, default: 0},
    message: String
    },
    { collection: 'donations' });

module.exports = mongoose.model('Donation', DonationSchema);
```

# Mongoose Models

- Each of the properties must have a type
  - Types can be Number, String, Boolean, array, object
  - Even nested objects
  - Good idea to explicitly name your collection

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

let DonationSchema = new mongoose.Schema({
    paymenttype: String,
    amount: Number,
    upvotes: {type: Number, default: 0},
    message: String
    },
    { collection: 'donations' });

module.exports = mongoose.model('Donation', DonationSchema);
```

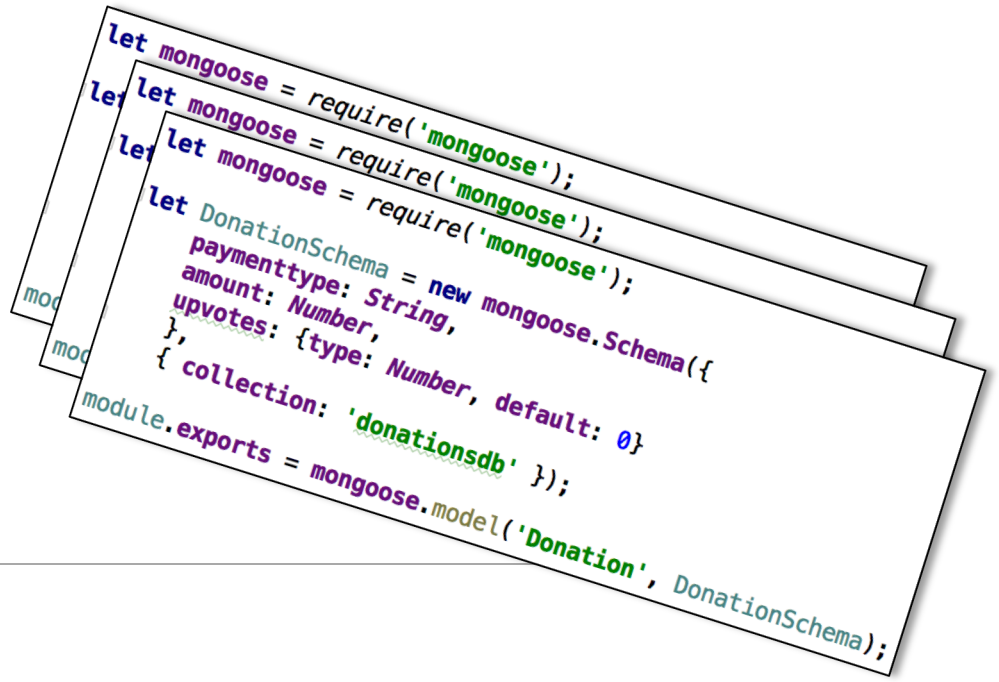
# Mongoose Models

- Each of the properties must have a type
  - Types can be Number, String, Boolean, array, object
  - Even nested objects
  - Good idea to explicitly name your collection

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

let UserSchema = new mongoose.Schema({
    username: String,
    userfname: String,
    usersname: String
    },
    { collection: 'users' });

module.exports = mongoose.model('User', UserSchema);
```

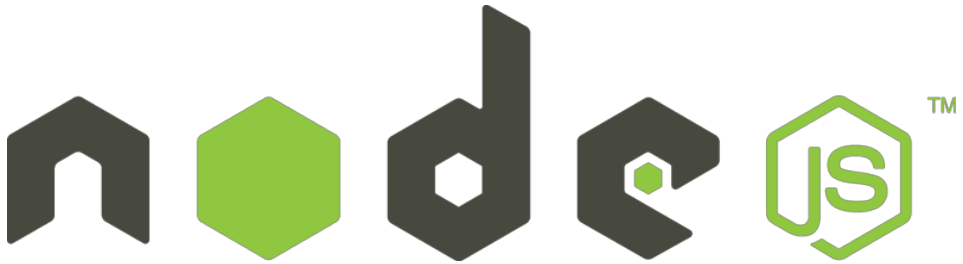
# CRUD with Mongoose

OBJECT-DOCUMENT MODEL SCHEMA



# CRUD with Mongoose

- Mongoose supports all the CRUD operations:
  - Create –> modelObj.save(callback)
  - Read –> Model.find().exec(callback)
  - Update –> modelObj.update(props, callback)
    –> Model.update(condition, props, callback)
  - Remove –> modelObj.remove(callback)
    –> Model.remove(condition, props, callback)



{ name: mongo, type: DB }

# donationweb

CASE STUDY EXAMPLE

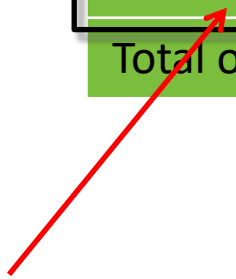

mongoose

elegant mongodb object modeling for node.js

# Donation: Resource, URIs & Methods

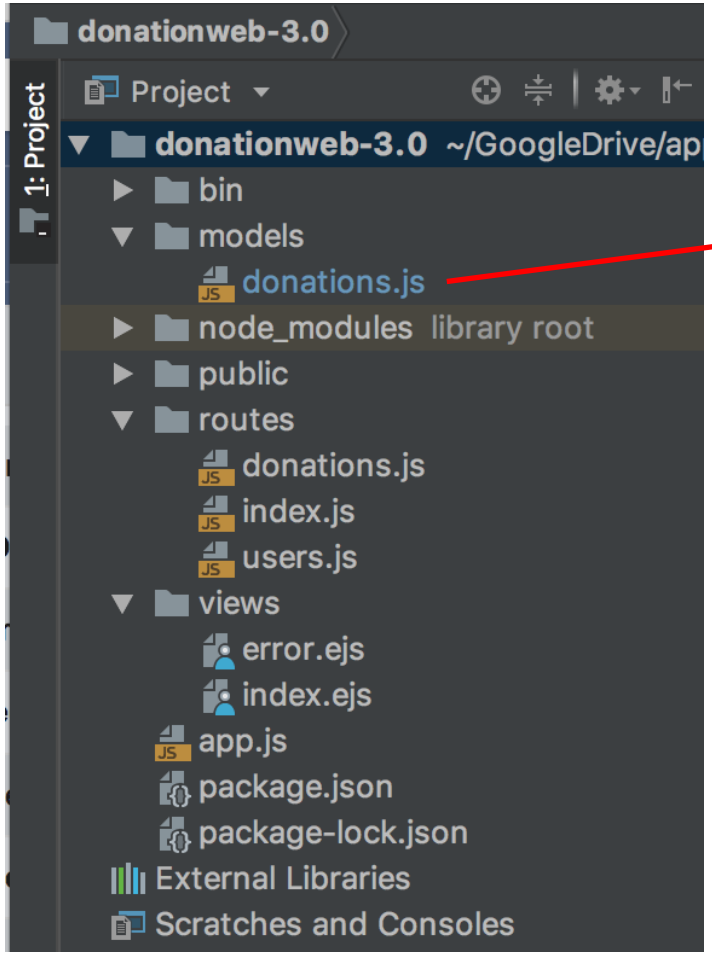
| Resource | URI (structure) | HTTP Request |
|---|---|---|
| List of Donations | /donations | GET |
| Get a Single Donation | /donations/{id} | GET |
| Upvote a Donation | /donations/{id}/vote | PUT |
| Delete a Donation | /donations/{id} | DELETE |
| Update a Donation | /donations/{id} | PUT |
| Add a Donation | /donations/{id} | POST |
| Total of Donation Votes | /donations/votes | GET |

{…} = variable value; changeable by user/application to refer to specific resource



**We'll look at this Use Case as an example…**

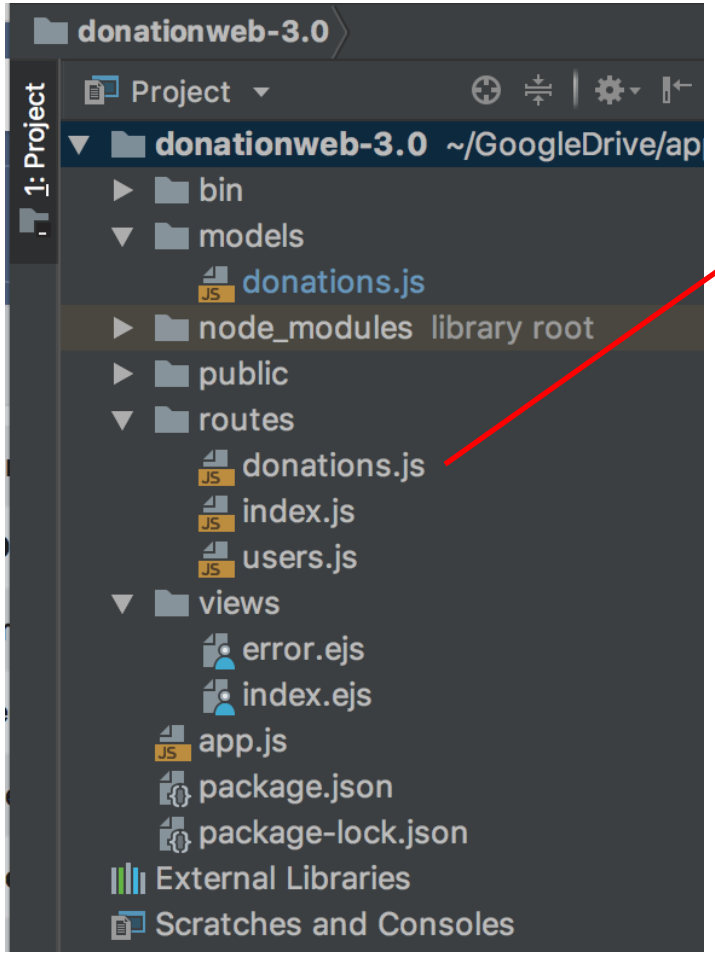
# Creating the Model – *Server Side*




donations.js

```
1
2   let mongoose = require('mongoose');
3
4   let DonationSchema = new mongoose.Schema({
5       paymenttype: String,
6       amount: Number,
7       upvotes: {type: Number, default: 0},
8       message: String
9       },
10      { collection: 'donations' });
11
12  module.exports = mongoose.model('Donation', DonationSchema);
```

# Creating the Routes (1) – *Server Side*

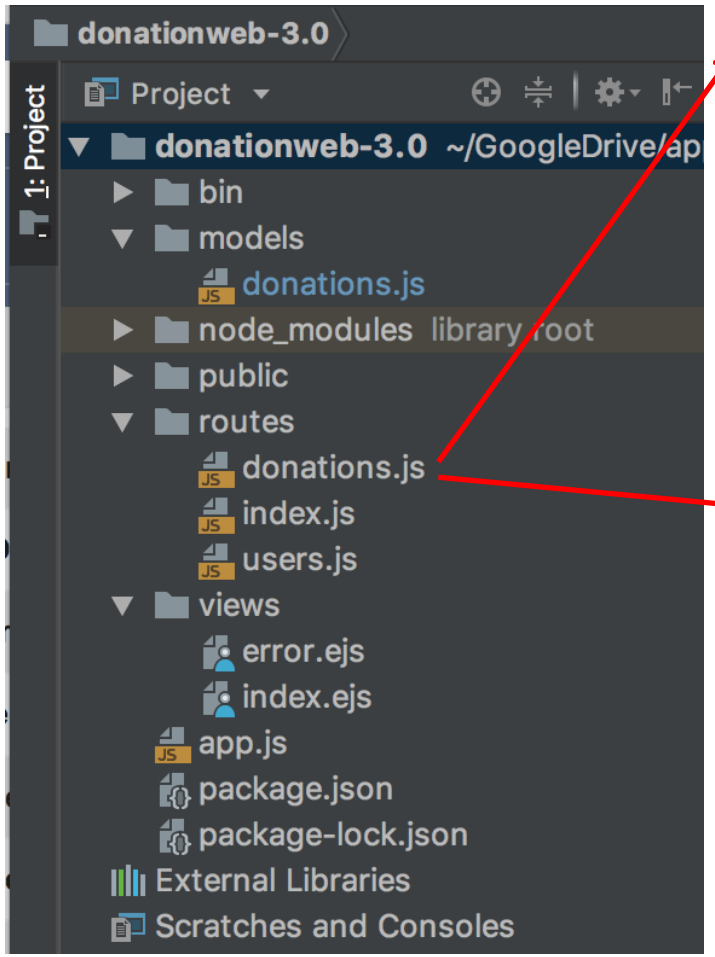



donations.js

```
let Donation = require('../models/donations');
let express = require('express');
let router = express.Router();
let mongoose = require('mongoose');
let uriUtil = require('mongodb-uri');

var mongodbUri ='mongodb://[redacted]@ds255260.mlab.com:55260/donationsdb';

mongoose.connect(mongodbUri);

//mongoose.connect('mongodb://localhost:27017/donationsdb');

let db = mongoose.connection;

db.on('error', function (err) {
    console.log('Unable to Connect to [ ' + db.name + ' ]', err);
});

db.once('open', function () {
    console.log('Successfully Connected to [ ' + db.name + ' ] on mlab.com');
});
```

N.B. on 'imports'

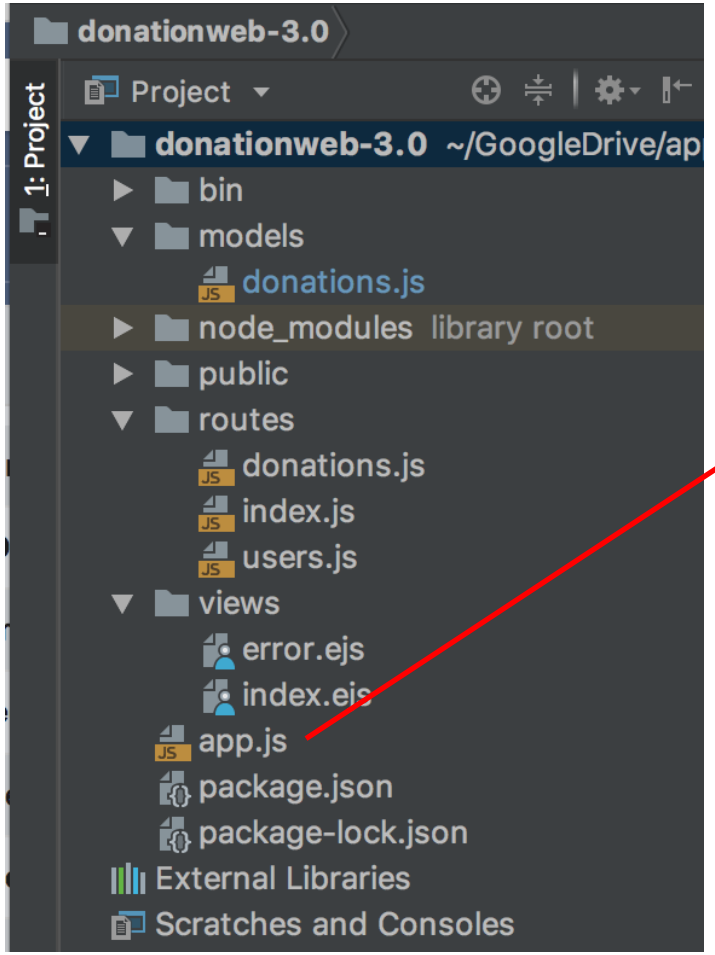
# Creating the Routes (2) – *Server Side*




```
router.addDonation = (req, res) => {

    res.setHeader('Content-Type', 'application/json');

    var donation = new Donation();

    donation.paymenttype = req.body.paymenttype;
    donation.amount = req.body.amount;

    donation.save(function(err) {
        if (err)
            res.json({ message: 'Donation NOT Added!', errmsg : err } );
        else
            res.json({ message: 'Donation Successfully Added!', data: donation });
    });
}
```

```
router.findAll = (req, res) => {
    // Return a JSON representation of our list
    res.setHeader('Content-Type', 'application/json');

    Donation.find(function(err, donations) {
        if (err)
            res.send(err);

        res.send(JSON.stringify(donations,null,5));
    });
}
```

# Creating the Routes (3) – *Server Side*




```
// Our Custom Donation Web App Routes
app.get('/donations', donations.findAll);
app.get('/donations/votes', donations.findTotalVotes);
app.get('/donations/:id', donations.findOne);

app.post('/donations',donations.addDonation);

app.put('/donations/:id/vote', donations.incrementUpvotes);

app.delete('/donations/:id', donations.deleteDonation);
```

mongoose


# Great Resources

Official Tutorial – https://nodejs.org/documentation/tutorials/

Official API – https://nodejs.org/api/

Developer Guide – https://nodejs.org/documentation

Video Tutorials – http://nodetuts.com

Video Introduction – https://www.youtube.com/watch?v=FqMlyTH9wSg

YouTube Channel – https://www.youtube.com/channel/UCvhIsEIBIfWSn_Fod8FuuGg

Articles, explanations, tutorials – https://nodejs.org/community/


# Questions?