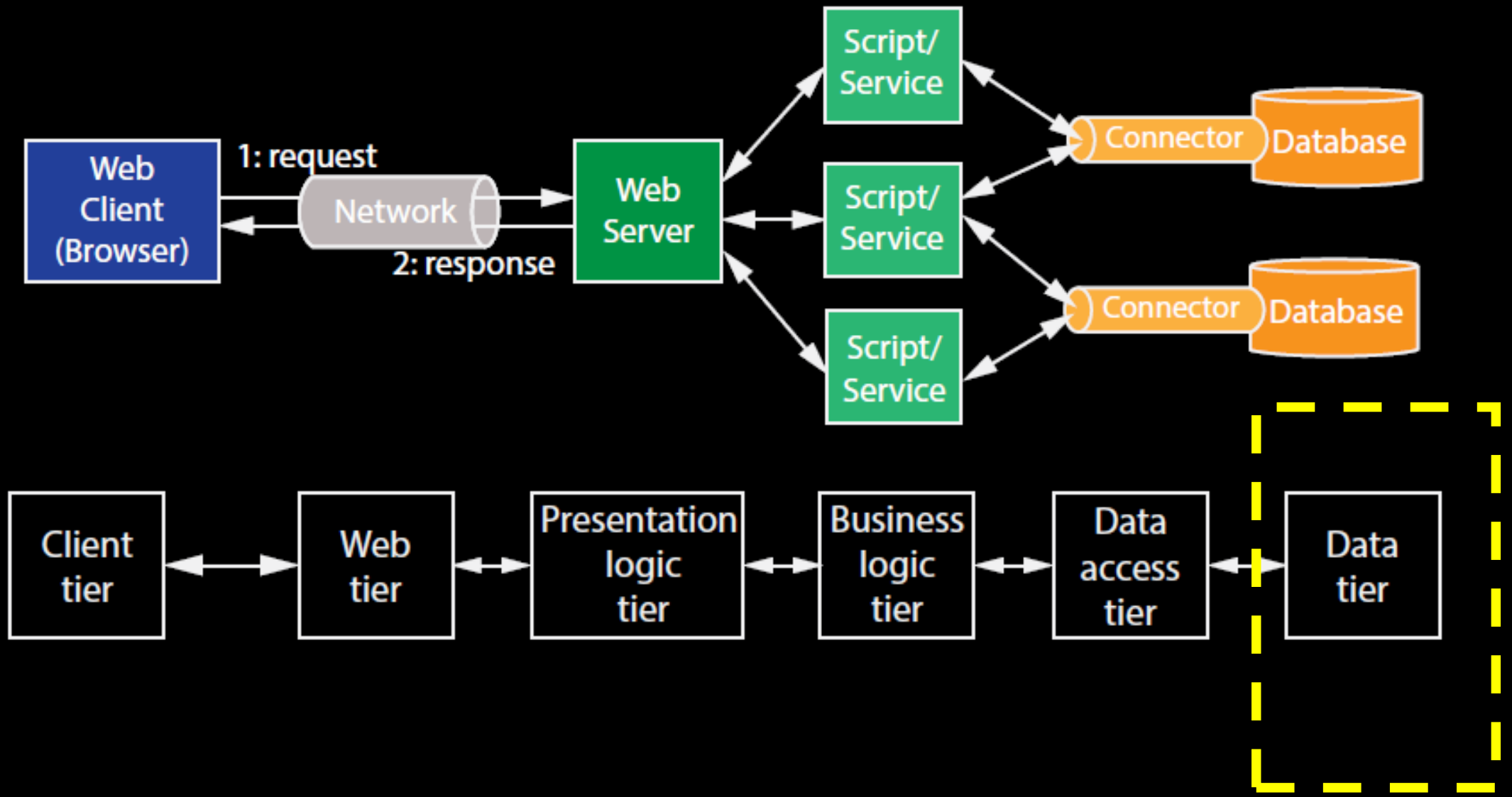
Database – Introduction & MongoDB

INF1802

Profa. Melissa Lemos


Web Client (Browser)
1: request
Network
2: response
Web Server
Script/ Service
Script/ Service
Script/ Service
Connector
Database
Connector
Database
Client tier
Web tier
Presentation logic tier
Business logic tier
Data access tier
Data tier
Our Focus: Database MongoDB

Database Introduction

Reference

- Jennifer Widom, Introdução a Sistemas de Banco de Dados. Coursera, Disponível online 2016, https://pt.coursera.org/course/db
- Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems (7th Edition)Jun 18, 2015.
- Pramod J. Sadalage, Martin Fowler. “NoSQL Distilled - A Brief Guide to the Emerging World of Polyglot Persistence”, Addison-Wesley, 2013.

DB: Database

coleção de dados relacionados.

DBMS: Database Management System

coleção de programas que permitem usuários definir, construir e manipular um BD

DB System = DB + DBMS

Database Management System (DBMS) provides….

… efficient, reliable, convenient, and safe multi-user storage of and access to massive amounts of persistent data.

Database systems are extremely prevalent in the world today.

- Massive
- Persistent
- Safe
- Multi-user
- Convenient
- Efficient
- Reliable

- **Massive**

  terabytes of data

  much larger than can fit in the memory of a typical computing system

- Persistent

  the data in the database outlives the programs that execute on

  that data

  very often multiple programs will be operating on the same data.

- Safe

  guarantees that the data managed by the system will stay in a consistent state, it won't be lost or overwritten when there are failures,

  hardware failures, software failures, malicious users.

- Multi-user

  multiple programs

  manu different users or applications to access the data

  concurrently

- Convenient

  the way that data is actually stored and laid out on disk is

    independent of the way that programs think about the

    structure of the data

  high level query languages

    in the query, you describe what you want out of the database

      but you don't need to describe the algorithm to get the data

      out

- Efficient

  database systems have to do really thousands of queries or updates per second.

- Reliable

  critically important that systems are up all the time

- Database applications may be programmed via “frameworks”
  Meteor, Django, Ruby on Rails
- DBMS may run in conjunction with “middleware”
  Web servers, Application servers
- Data-intensive applications may not use DBMS at all

Key concepts

- Data model
  Description of, in general, how the data is structured
  Relational model, XML documents, Graph data model

- Schema versus data
  The schema sets up the structure of the database

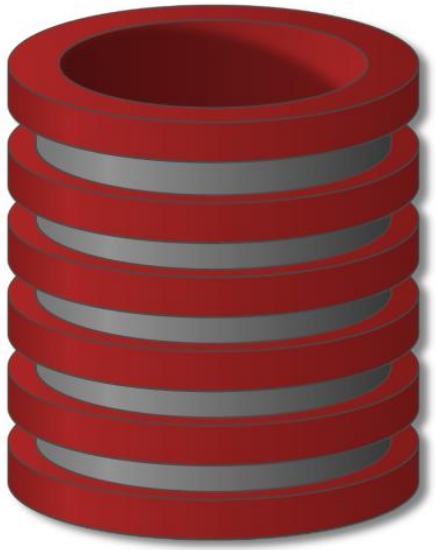
- Data definition language (DDL)

- Data manipulation or query language (DML)

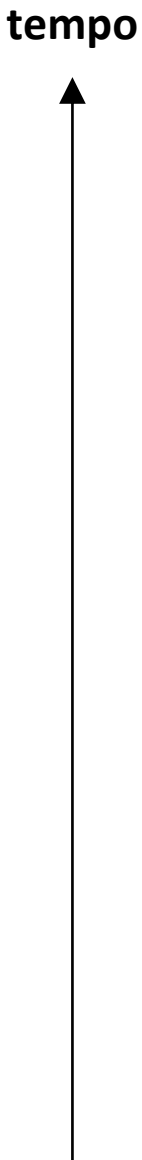
Key people

- DBMS implementer
  the person who builds the system
- Database designer
  the person who establishes the schema for a database
- Database application developer
  the person who builds the applications or programs that are going to run on the database
- Database administrator
  the person who loads the data, sort of gets the whole thing running and keeps it running smoothly

Whether you know it or not,
you're using a database every day



Evolução



Programas e dados
na mesma memória
Código e dados compondo um único objeto.

Programa com dados armazenados

Evolução

tempo

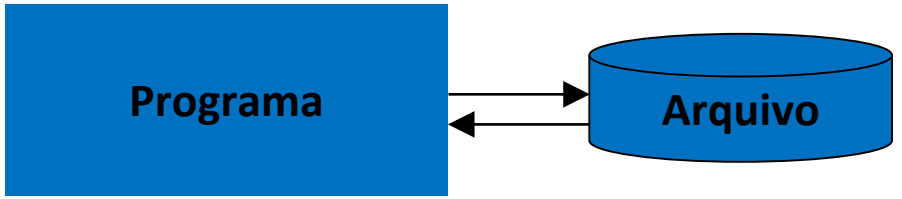
Sistemas de Arquivos

Surgimento do Disco.

Código e dados armazenados de forma distinta.

Dados organizados como coleções de itens relacionados, compondo arquivos.

Código com rotinas para gerência de dados (localização, recuperação e Armazenamento de dados



Programas e dados na mesma memória

Programa com dados armazenados

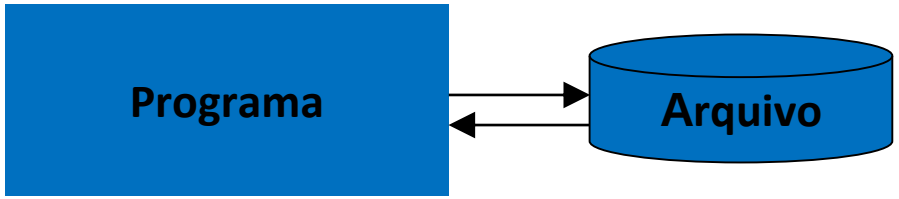
Evolução

tempo

Sistemas de Bancos de Dados

Independência dos programas em relação aos dados armazenados
SGBD possui todas as funções necessárias para localização e manipulação
dos dados

Sistemas de Arquivos

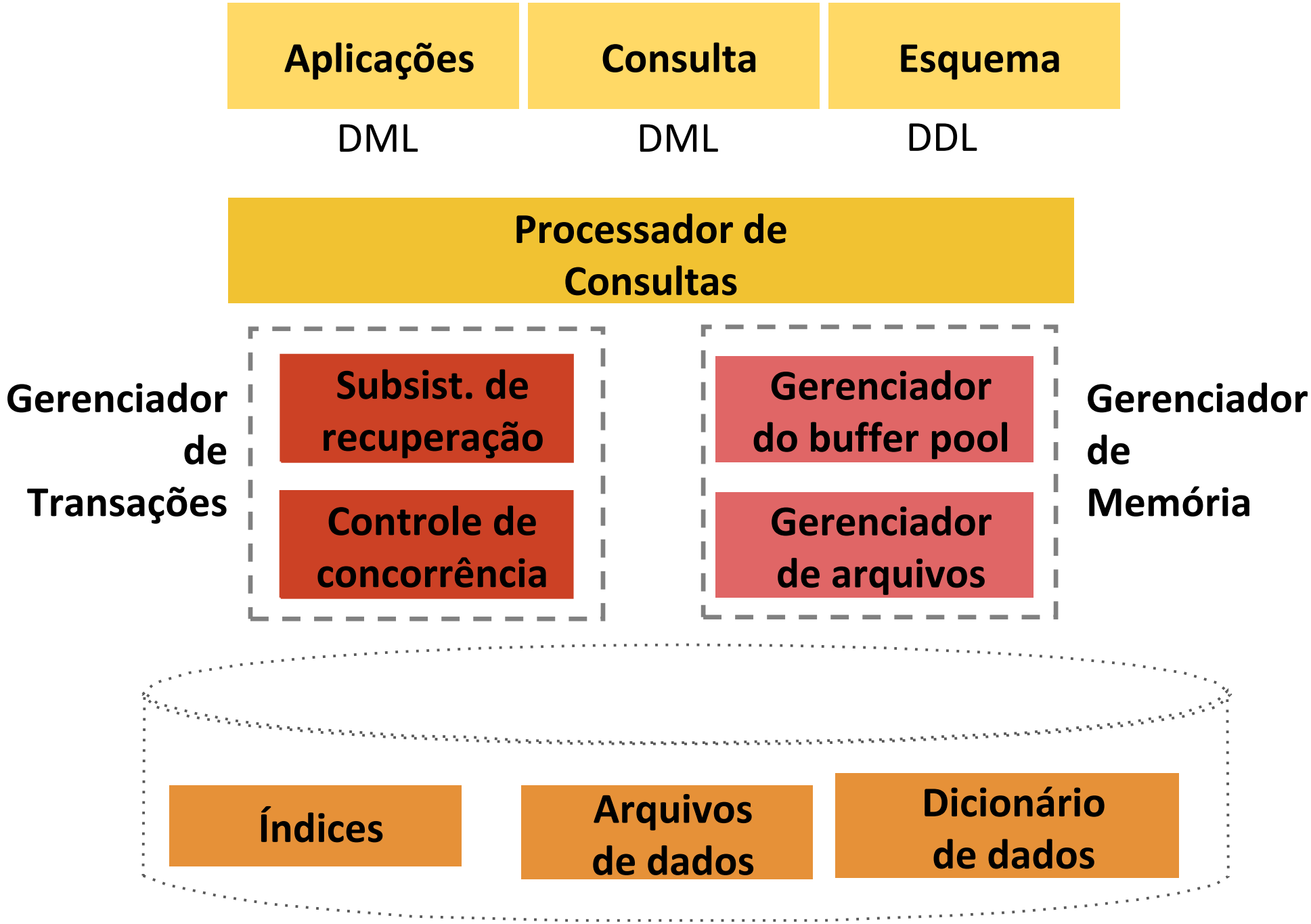


Programas e dados na mesma memória

Programa com dados armazenados

Sistemas de BDs x Sistemas de Arquivos

- Separação entre programas e dados
- Suporte para múltipla visões de usuário
- Compartilhamento de dados e processamento multi usuário de transações
- Armazenamento no BD da sua própria descrição ou *esquema*
- Controle para não haver redundância de dados


Aplicações
Consulta
Esquema
DML
DML
DDL
Processador de Consultas
Gerenciador de Transações
Subsist. de recuperação
Controle de concorrência
Gerenciador do buffer pool
Gerenciador de arquivos
Gerenciador de Memória
Índices
Arquivos de dados
Dicionário de dados

Arquivos de Dados

- Persistência
- Armazenamento dos objetos do banco de dados, que vão sofrendo modificações com o tempo

Dicionário de Dados

- Armazena os metadados relativos à estrutura do banco de dados.

Índices

- Proporcionam acesso rápido aos dados

Processador de Consultas

- Definição e manipulação de dados
- Funções oferecidas pelas linguagens

- DDL
  - processa os comandos de descrição das estruturas do esquema, e a armazena em ***dicionário de dados***

- DML
  - Realizar consultas e atualizações, inserções e remoções

- SQL é a linguagem típica de SGBD relacional que engloba uma LDD e uma LMD.

Gerenciador de Transações/ Subsistema de Recuperação

- Recuperação de dados
- Garantir que falhas durante o processamento de transações não sejam propagadas aos objetos persistentes.
  - Os objetos armazenados devem sobreviver a falhas das transações e mesmo algumas falhas de hardware
  - Mecanismo de recuperação/ *recovery*

Gerenciador de Transações/ Controle de Concorrência de Transações

- Garante que transações concorrentes serão executadas sem conflitos em seus procedimentos.
- Técnicas de controle de concorrência

Gerenciador de Memória/ Gerenciador de Arquivos

- Gerencia a alocação de espaço no armazenamento em disco e as estruturas de dados usadas para representar estas informações armazenadas em disco

Gerenciador de Memória/ Gerenciador de Buffer

- Responsável pela intermediação de dados do disco para a memória principal e pela decisão de quais dados colocar em memória cache

Gerenciador de Autorizações e Integridade

- Mantém o BD em estado consistente, satisfazendo algumas condições, chamadas de restrições de integridade.
- Implementa mecanismos de segurança de acesso para consulta, remoção, atualização e inserção de dados
  - Comandos de concessão e revogação (*grant* e *revoke*) de acesso a usuários individuais ou grupos de usuários

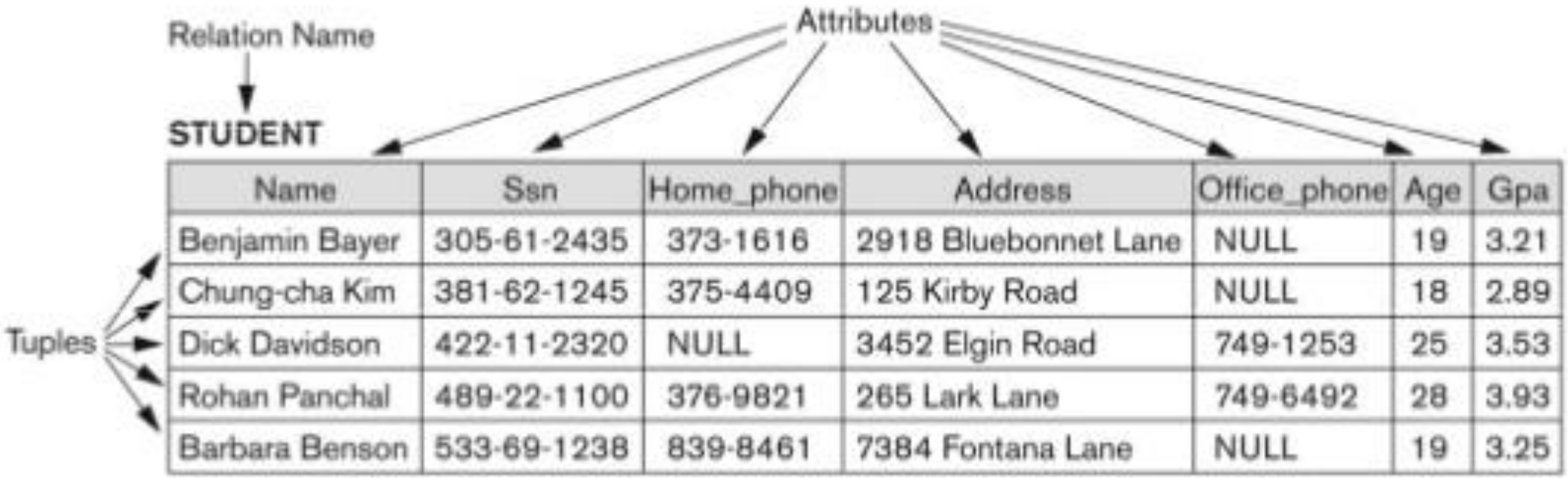
Desempenho

- O SGBD deve executar as funções de forma eficaz e eficiente
  - Estruturas de dados
  - Métodos de acesso
  - Técnicas de otimização

Carga, descarga, cópia, restauração

- Facilidades para
  - Carregar e descarregar o banco de dados ou parte deles
  - Fazer cópia de segurança = backup
  - Restaurar o banco de dados a partir do backup

Relational Database



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1
The attributes and tuples of a relation STUDENT.

From 2007 Ramez Elmasri and Shamkant B. Navathe,
http://pt.slideshare.net/sontumax/5-chapter5

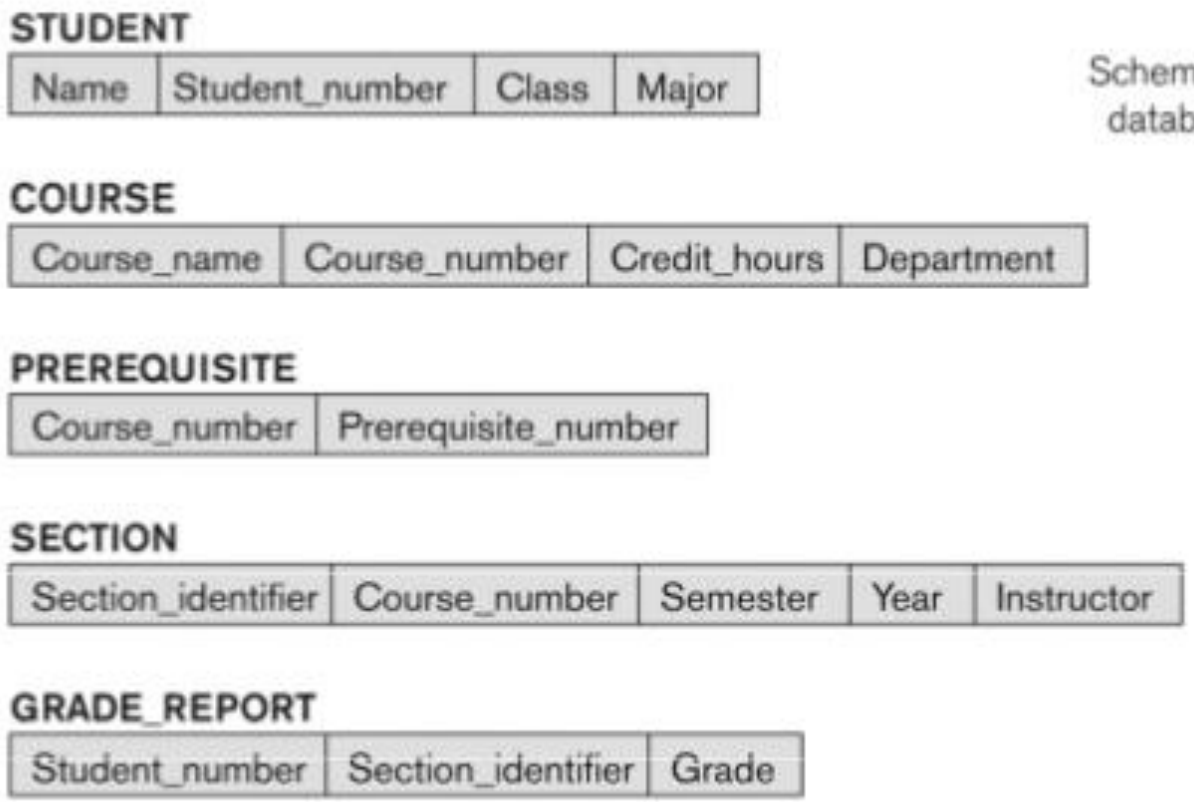


Figure 2.1
Schema diagram for the database in Figure 1.2.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

Copyright © 2007 Ramez Elmasri and Shamkant B. Navathe

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

SELECT

Query: Retrieve the birthdate and address of the employee whose name is 'John'.

```
SELECT    BDATE, ADDRESS
FROM      EMPLOYEE
WHERE     FNAME='John'
```

INSERT

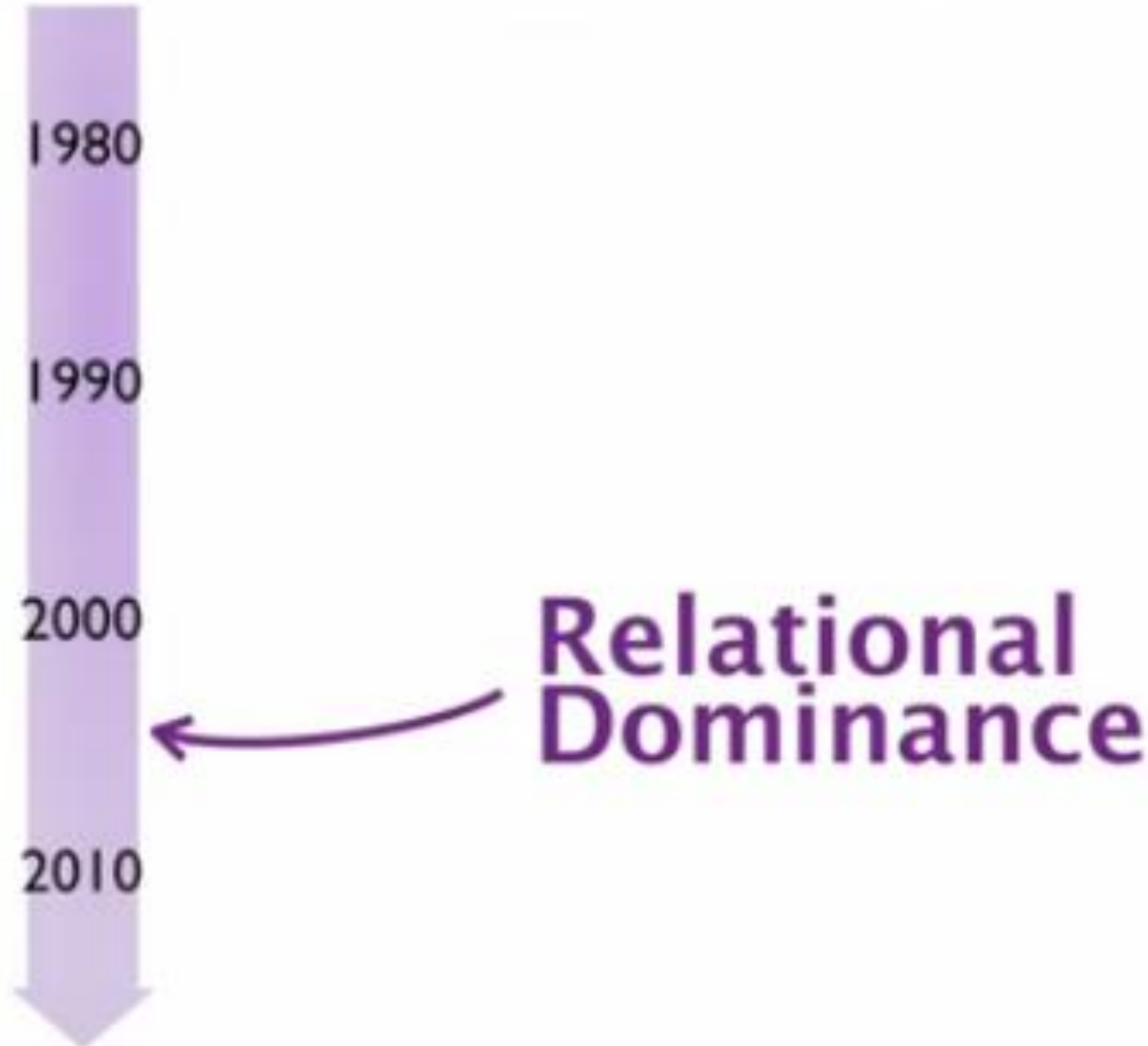
```
INSERT INTO     EMPLOYEE
VALUES
('Richard','K','Marini',
    '653298653', '30-DEC-52',
   '98 Oak Forest,Katy,TX',
   'M',
   37000,
   '987654321',
   4 )
```

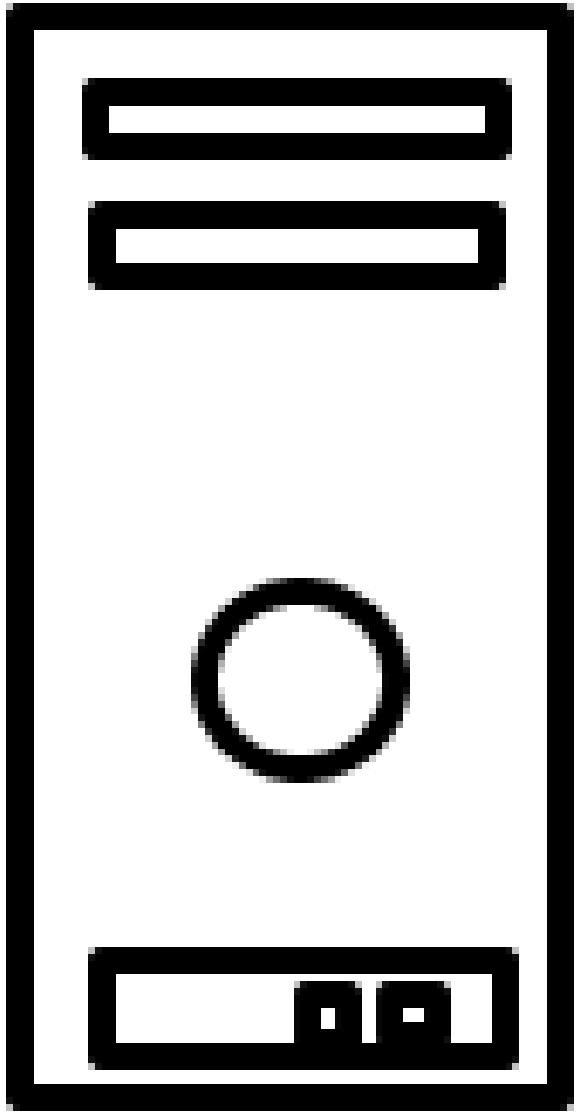
DELETE

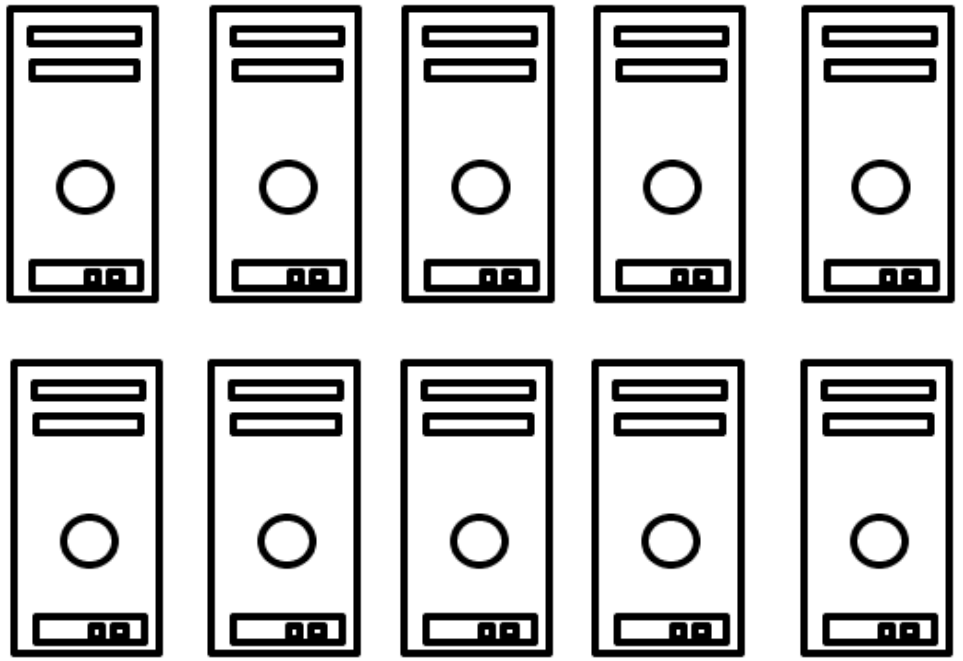
DELETE FROM EMPLOYEE
WHERE LNAME='Brown’

UPDATE

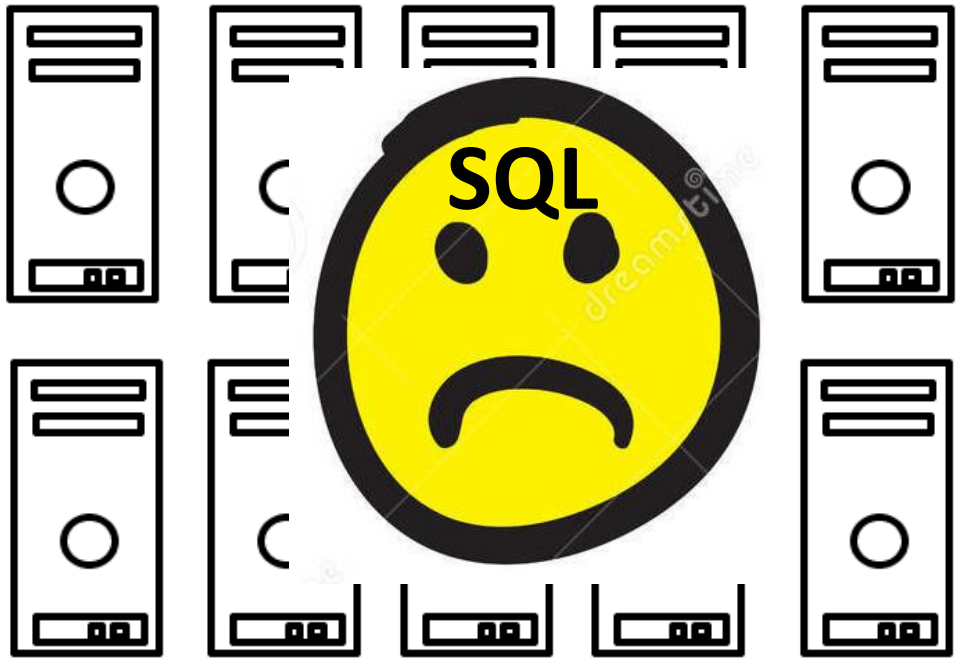
```
UPDATE    PROJECT
SET       PLOCATION = 'Bellaire',        DNUM = 5
WHERE     PNUMBER=10
```

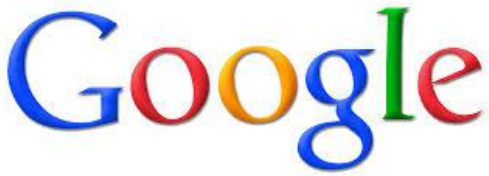

1980
1990
2000
2010
Relational
Dominance






SQL


SQL



⟹ BigTable

⟹ Dynamo

http://nosql-database.org/

noSQL Database

NoSQL is an accidental neologism.

There is no prescriptive definition—all you can make is an observation of common characteristics.

- Flexible schema
- Quicker/cheaper to set up
- Massive scalability
- Relaxed consistency → higher performance & availability
- Cluster friendly

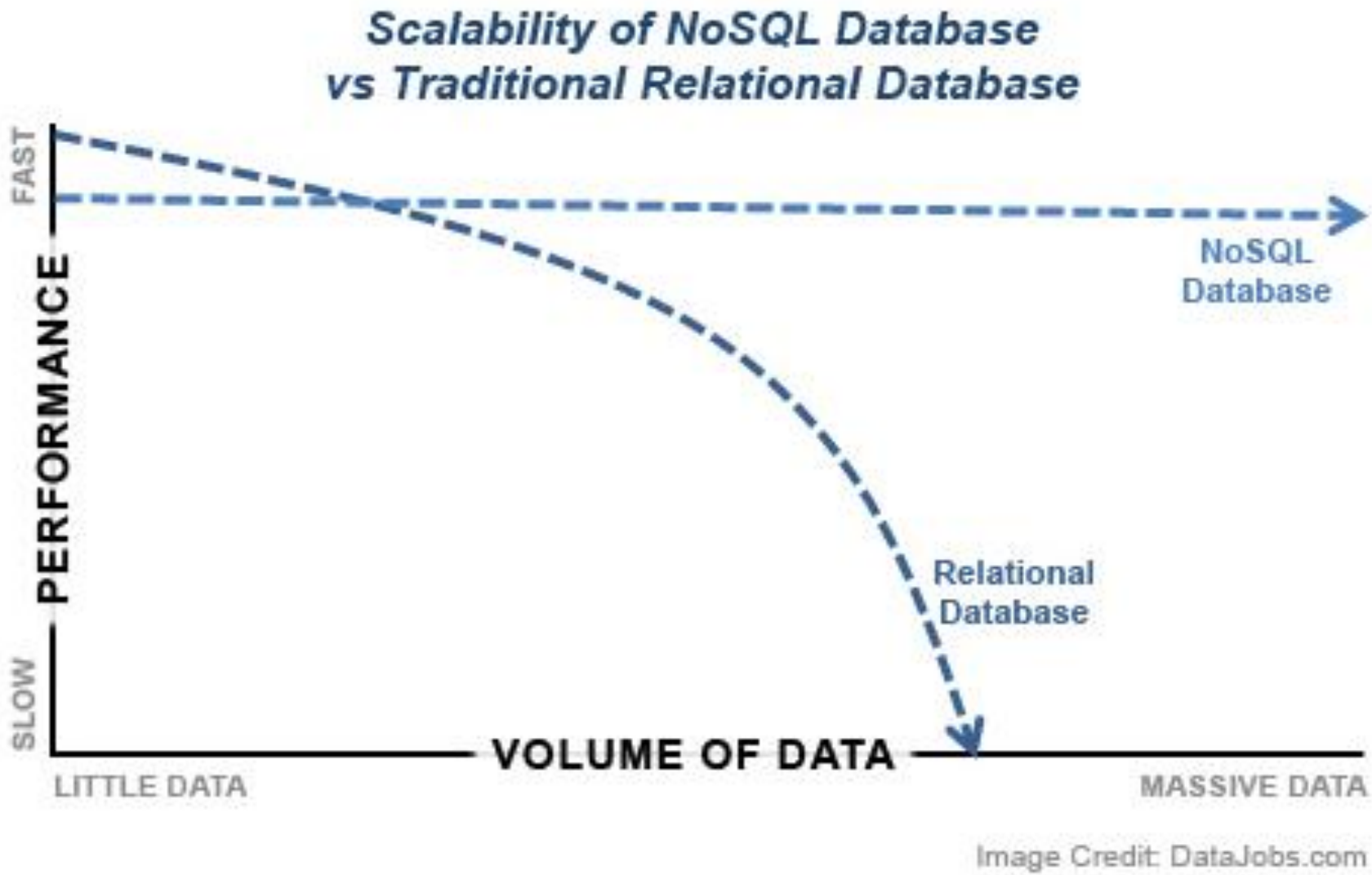
Not every data management/analysis problem is best solved exclusively using a traditional relational DBMS

“NoSQL” = “Not Only SQL”

- “SQL” = Traditional Relational DBMS
- Recognition over past decade

Not every data management/analysis problem is best solved using a traditional relational DBMS

- “No SQL” = Not using traditional relational DBMS
- “No SQL” ≠ Don’t use SQL language


Scalability of NoSQL Database
vs Traditional Relational Database
FAST
PERFORMANCE
SLOW
NoSQL
Database
Relational
Database
VOLUME OF DATA
LITTLE DATA
MASSIVE DATA
Image Credit: DataJobs.com

NoSQL
Data Models

KEY-VALUE

COLUMN

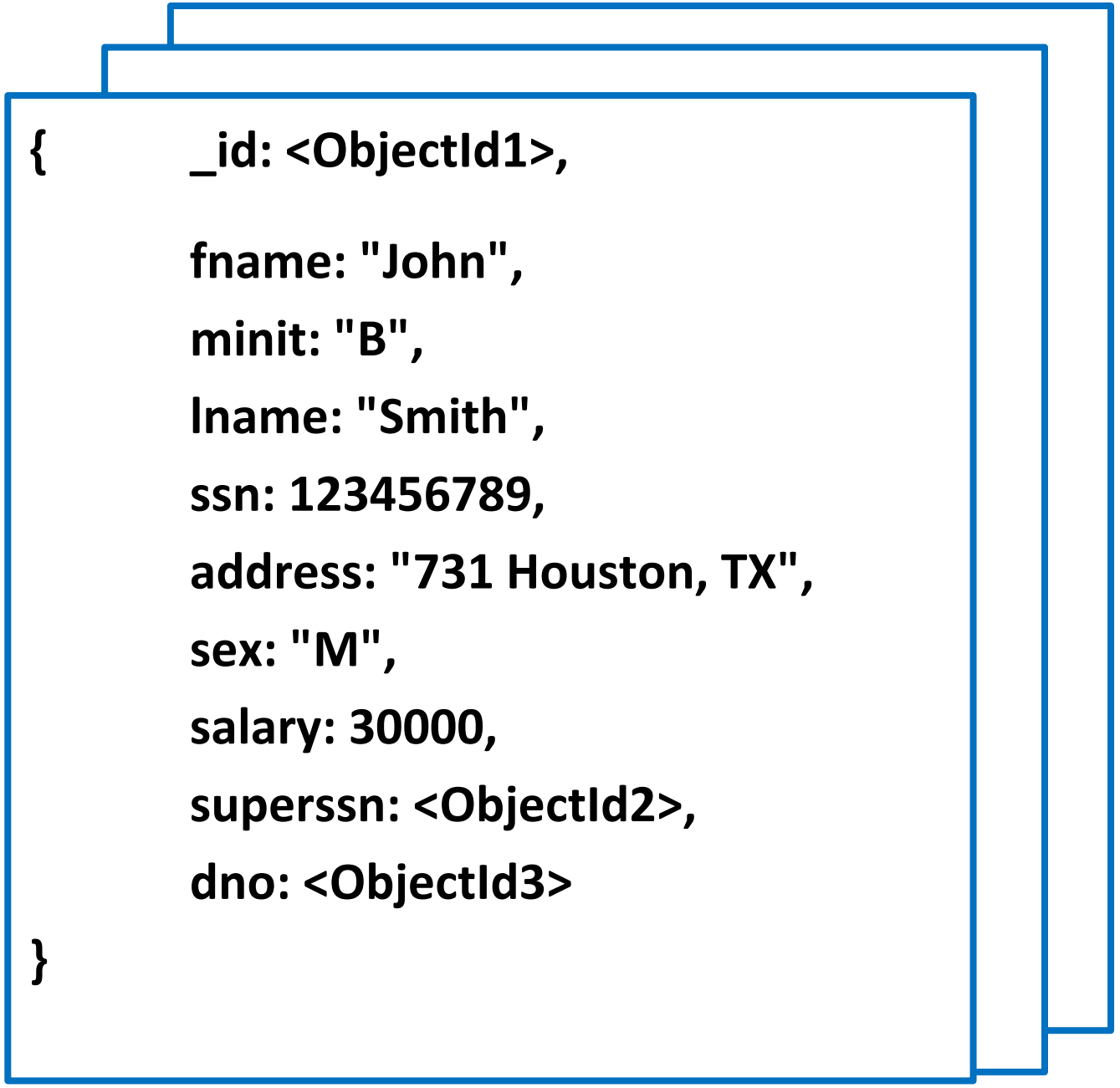
DOCUMENT

GRAPH

mongoDB

Relational	mongoDB
Database	Database
Table	Collection
Row	Document
Index	Index
Join	Embedded Document or Reference

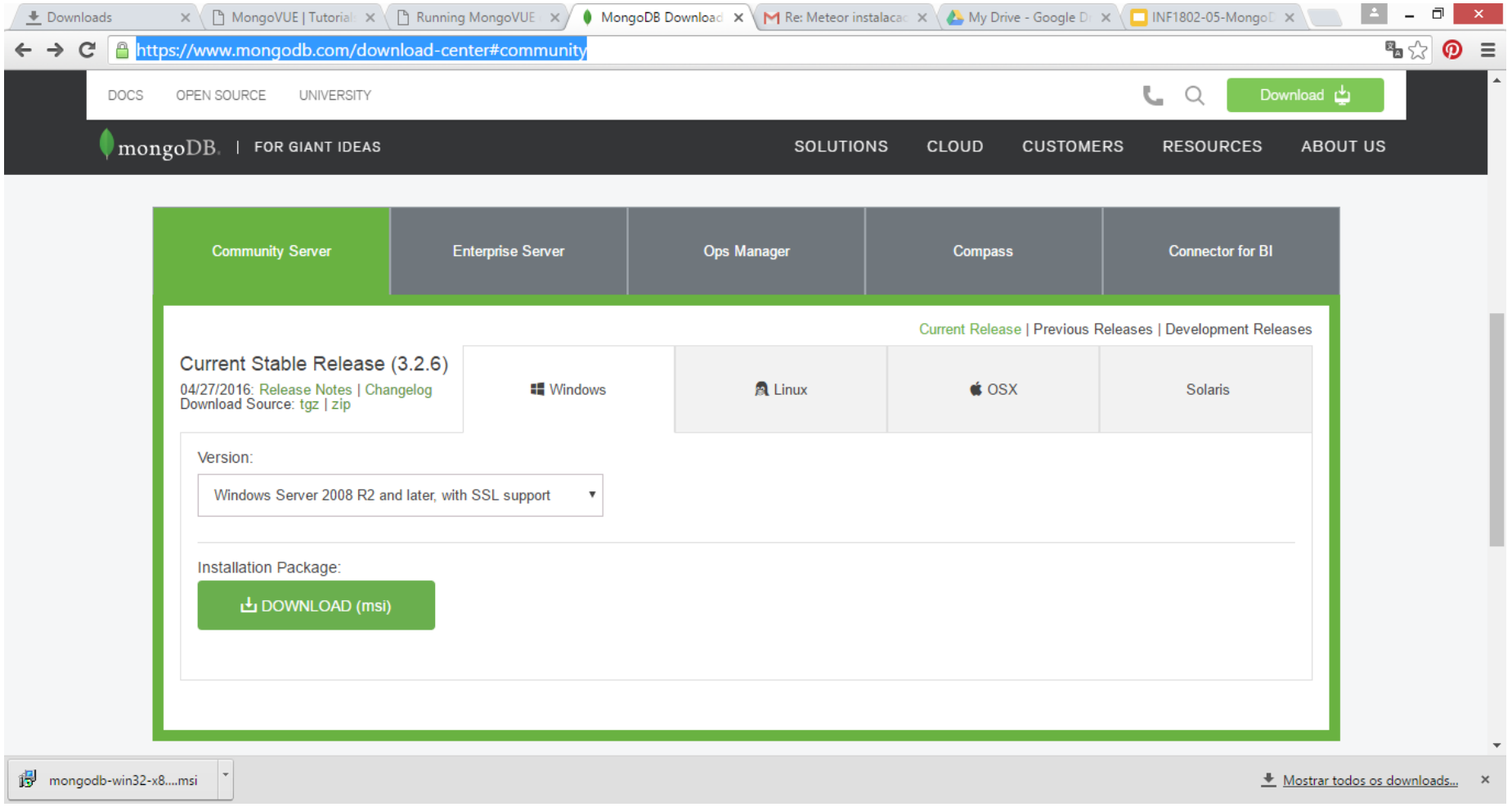
Collection


{
_id: <ObjectId1>,
fname: "John",
minit: "B",
lname: "Smith",
ssn: 123456789,
address: "731 Houston, TX",
sex: "M",
salary: 30000,
superssn: <ObjectId2>,
dno: <ObjectId3>
}

Database Installation

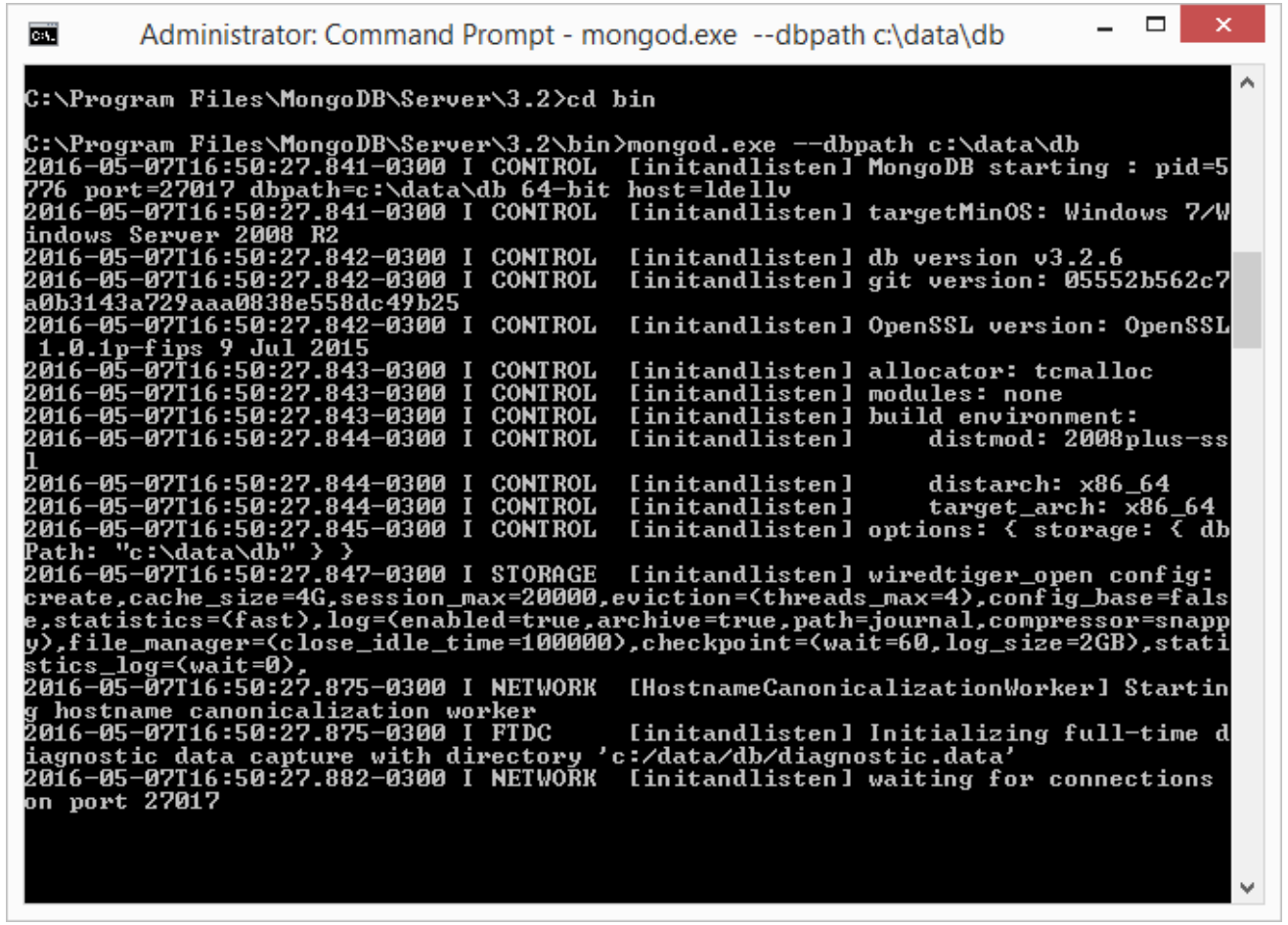
Step 1 - MongoDB Download and Installation (msi file, double click to install)

https://www.mongodb.com/download-center#community



Step 2 - Run the MongoDB server

```
C:\mongodb\bin\mongod.exe --dbpath c:\data\db
```



From
https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-windows/
http://www.mkyong.com/mongodb/how-to-install-mongodb-on-windows/

Step 3 - Connect to MongoDB Server

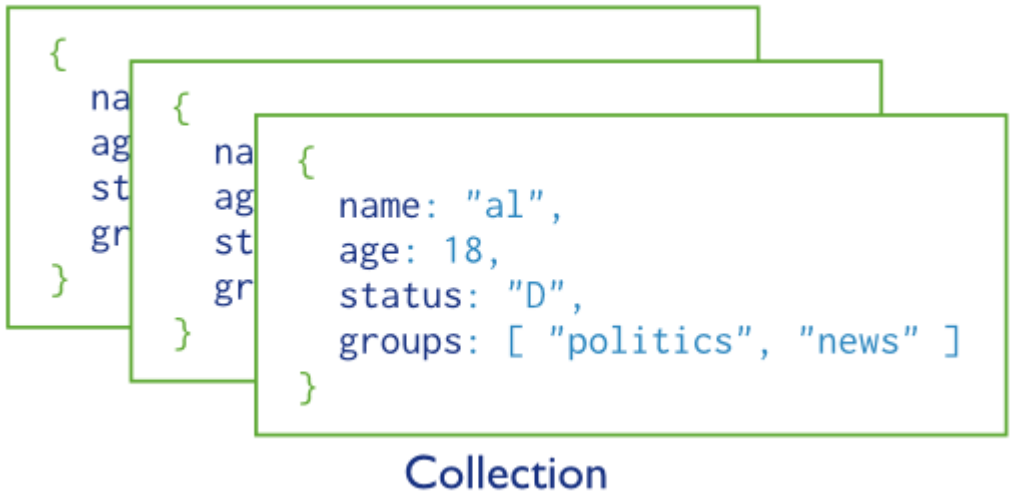
through the mongo.exe shell, open another Command Prompt.

```
C:\mongodb\bin\mongo.exe
```

From https://docs.mongodb.com/manual/tutorial/install-mongodb-enterprise-on-windows/

Data Model

MongoDB stores data in the form of *documents*, which are JSON-like field and value pairs. Documents are analogous to structures in programming languages that associate keys with values (e.g. dictionaries, hashes, maps, and associative arrays).



MongoDB stores all documents in collections. A collection is a group of related documents that have a set of shared common indexes. Collections are analogous to a table in relational databases.

Database Operations

https://docs.mongodb.com/manual/core/crud-introduction/

Query

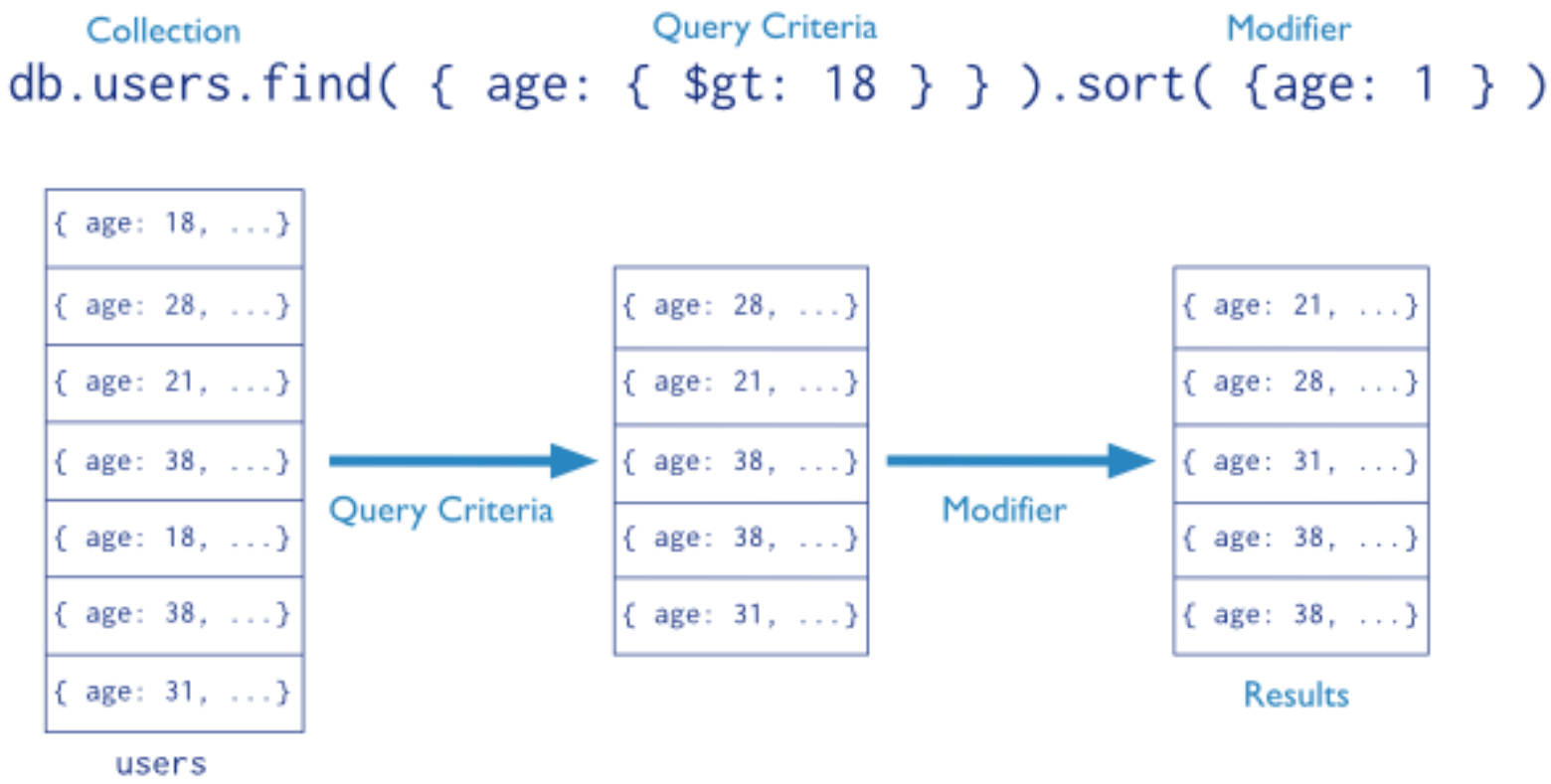
Query

In MongoDB a query targets a specific collection of documents. Queries specify criteria, or conditions, that identify the documents that MongoDB returns to the clients.

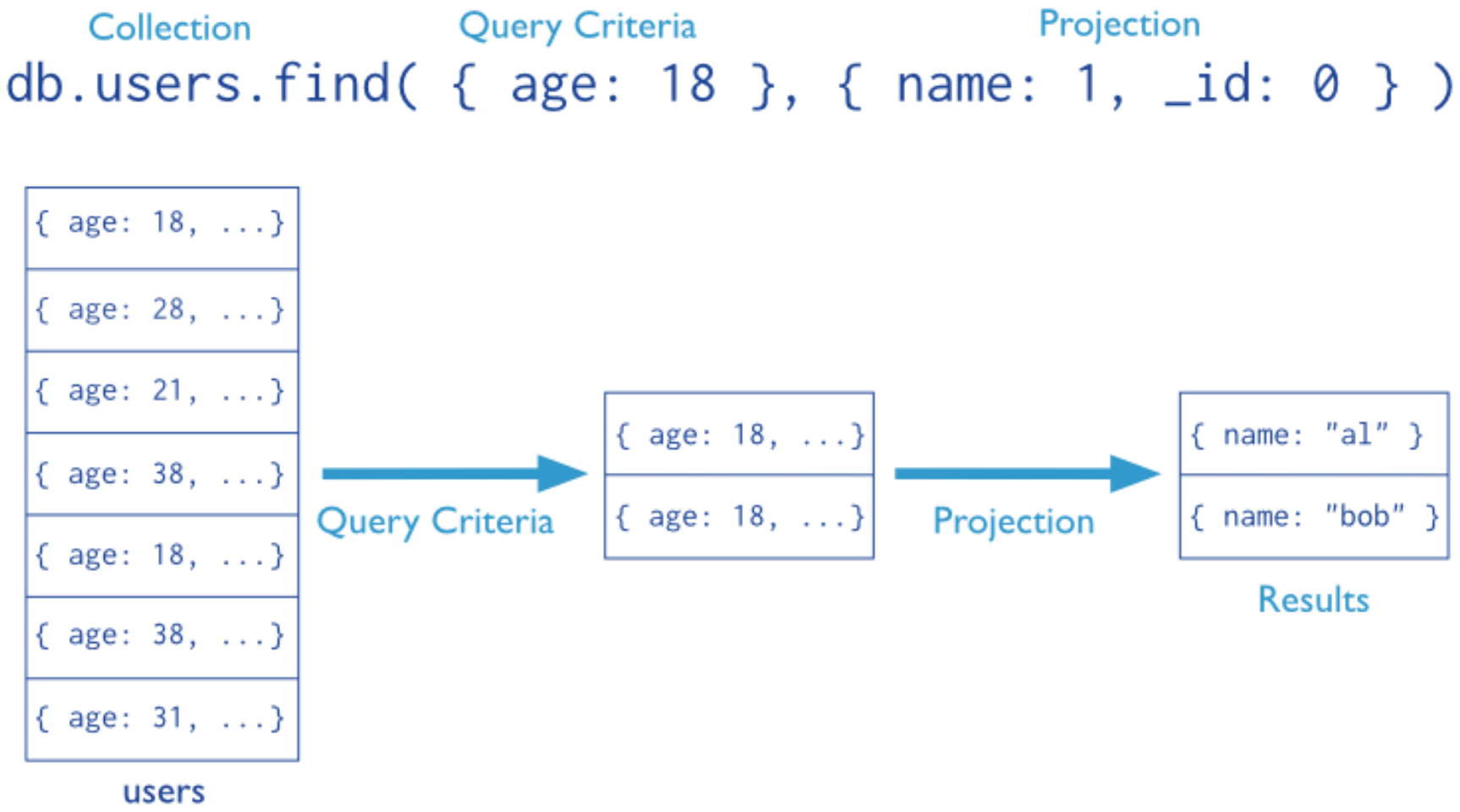
```
db.users.find()
```

```
select *
from users
```

Query - You can optionally modify queries to impose limits, skips, and sort orders.



Query - A query may include a *projection* that specifies the fields from the matching documents to return.



```
db.users.find(
    { age: { $gt: 18 } },
    { name: 1, address: 1 }
).limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

```
SELECT _id, name, address
FROM   users
WHERE  age > 18
LIMIT  5
```

← projection
← table
← select criteria
← cursor modifier

Data Modification

Operations that Create, Update or Delete Data

https://docs.mongodb.com/manual/core/write-operations-introduction/

Collection

Document

```
db.users.insert(
                {
                    name: "sue",
                     age: 26,
                  status: "A",
                  groups: [ "news", "sports" ]
                }
              )
```

Document

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

insert

Collection

{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }

users

```
INSERT INTO users                    <-- table
            ( name, age, status )    <-- columns
VALUES      ( "sue", 26, "A" )       <-- values/row
```

```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

collection

field: value

field: value

field: value

} document

```
INSERT INTO users
            ( name, age, status )
VALUES      ( "sue", 26, "A" )
```

table

columns

values/row

```
db.users.insertMany(
   [
      {
         name: "sue",
         age: 26,
         status: "pending"
      },
      {
         name: "bob",
         age: 25,
         status: "enrolled"
      },
      {
         name: "ann",
         age: 28,
         status: "enrolled"
      }
   ]
)
```

collection

field: value

field: value

field: value

document

document

document

```
INSERT INTO user
            ( name, age, status)
VALUES      ( "sue", 26, "pending" ),
            ( "bob", 25, "enrolled" ),
            ( "ann", 28, "enrolled" )
```

table

columns

values/row

```
db.users.update(
   { age: { $gt: 18 } },
   { $set: { status: "A" } },
   { multi: true }
)
```

collection
update criteria
update action
update option

```
UPDATE users
SET    status = 'A'
WHERE  age > 18
```

table
update action
update criteria

```
db.users.updateOne(
   { age : { $lt : 18 } } ,
   { $set: { status : "reject" } }
)
```

collection

update filter

update action

```
UPDATE users
SET    status = 'reject'
WHERE  age < 18
LIMIT  1
```

table

update action

update filter

update limit

```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

collection

update filter

update action

```
UPDATE users
SET    status = 'reject'
WHERE  age < 18
```

table

update action

update filter

```
db.users.remove(
   { status: "D" }
)
```

collection

remove criteria

```
DELETE FROM users
WHERE  status = 'D'
```

table

delete criteria

```
db.users.deleteOne(
   { status: "Rejected" }
)
```

collection

delete filter

```
DELETE FROM users
WHERE       status = 'reject'
LIMIT       1
```

table

delete filter

delete limit

```
db.users.deleteMany(
   { status: "Rejected" }
)
```

collection

delete filter

```
DELETE FROM users
WHERE       status = 'reject'
```

table

delete filter

Step 4 - Insert a document into a collection named restaurants. The operation will create the collection if the collection does not currently exist.

```
db.restaurants.insert
```

From docs.mongodb.com/getting-started/shell/

Step 5 - Query for All Documents in a Collection. To return all documents in a collection, call the find() method without a criteria document.

```
db.restaurants.find()
```

https://docs.mongodb.com/getting-started/shell/query/

Step 6 - Specify Equality Conditions.

```
db.restaurants.find({<field1>: <value1>, <field2>: <value2>, ...})
```

```
> db.restaurants.insert(
... {
... "name":"Dominos",
... "tipo":"pizzaria"
... }
... )
WriteResult({ "nInserted" : 1 })
>
>
> db.restaurants.insert(
... {
... "name":"Gula Gula",
... "tipo":"variada"
... }
... )
WriteResult({ "nInserted" : 1 })
>
>
>
>
> db.restaurants.find()
{ "_id" : ObjectId("572e4c431d569630bfd9890e"), "name" : "Botequim Joao", "tipo"
 : "comida caseira" }
{ "_id" : ObjectId("572e50401d569630bfd9890f"), "name" : "Dominos", "tipo" : "pi
zzaria" }
{ "_id" : ObjectId("572e506c1d569630bfd98910"), "name" : "Gula Gula", "tipo" : "
variada" }
>
>
> db.restaurants.find({"name":"Dominos"})
{ "_id" : ObjectId("572e50401d569630bfd9890f"), "name" : "Dominos", "tipo" : "pi
zzaria" }
>
>
>
>
```

https://docs.mongodb.com/getting-started/shell/query/

Relationships, Joins

https://docs.mongodb.com/manual/core/write-operations-introduction/

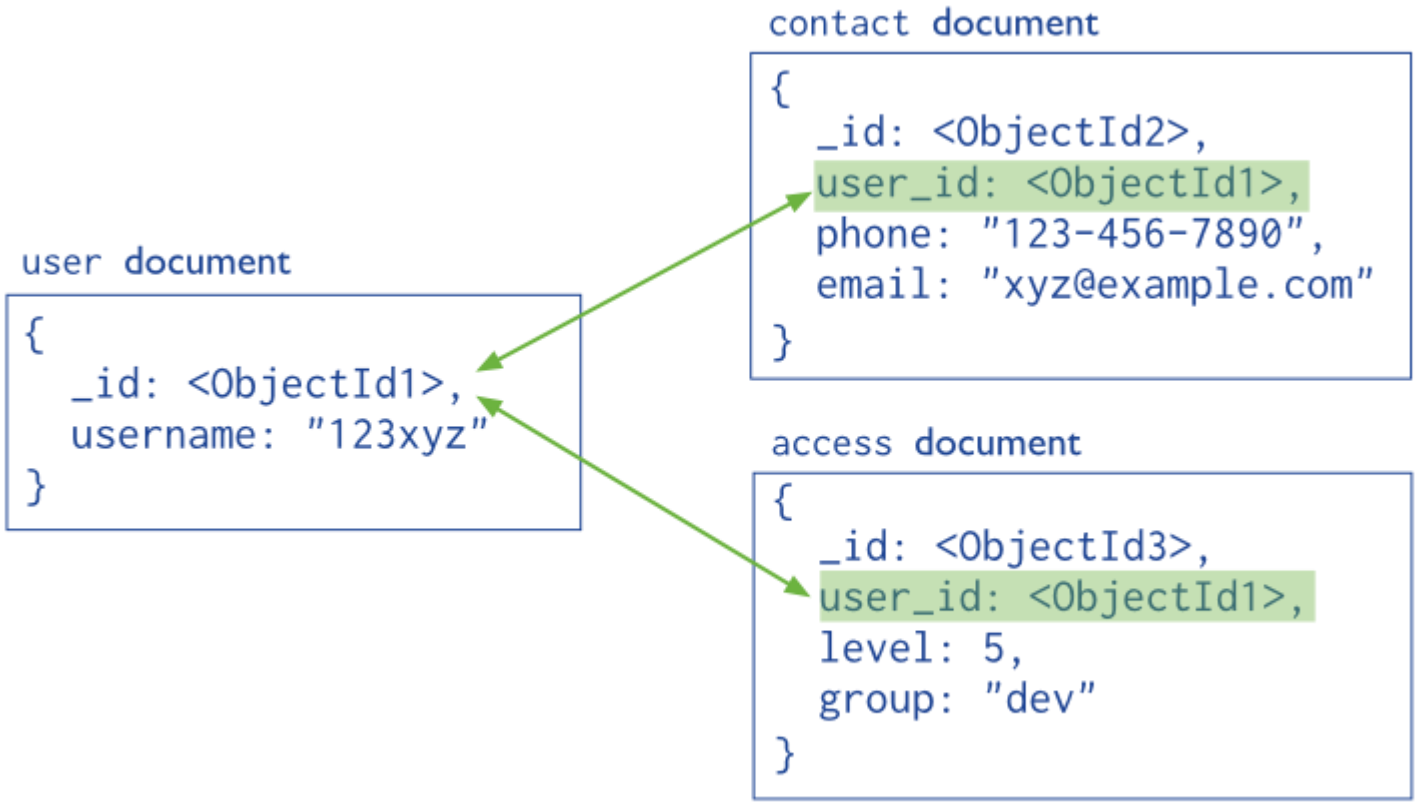
No Joins

The first and most fundamental difference that you'll need to get comfortable with is MongoDB's lack of joins.

Normalized Data Models

In general, use normalized data models:

- when embedding would result in duplication of data but would not provide sufficient read performance advantages to outweigh the implications of the duplication.
- to represent more complex many-to-many relationships.
- to model large hierarchical data sets.



Embedded Document - Relationship 1:1

https://docs.mongodb.com/manual/core/data-model-design/#embedded-data-models

Embedded Document - Relationship 1:n

```
{
    _id: "joe",
    name: "Joe Bookreader",
    addresses: [
                    {
                      street: "123 Fake Street",
                      city: "Faketon",
                      state: "MA",
                      zip: "12345"
                    },
                    {
                      street: "1 Some Other Street",
                      city: "Boston",
                      state: "MA",
                      zip: "12345"
                    }
                ]
 }
```

Document Reference - Relationship 1:n

```
{
   title: "MongoDB: The Definitive Guide",
   author: [ "Kristina Chodorow", "Mike Dirolf" ],
   published_date: ISODate("2010-09-24"),
   pages: 216,
   language: "English",
   publisher: {
              name: "O'Reilly Media",
              founded: 1980,
              location: "CA"
            }
}
```

```
{
   name: "O'Reilly Media",
   founded: 1980,
   location: "CA",
   books: [12346789, 234567890, ...]
}

{
    _id: 123456789,
    title: "MongoDB: The Definitive Guide",
    author: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English"
}

{
   _id: 234567890,
   title: "50 Tips and Tricks for MongoDB Developer",
   author: "Kristina Chodorow",
   published_date: ISODate("2011-05-06"),
   pages: 68,
   language: "English"
}
```

Model One-to-One Relationships with Embedded Documents

https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-one-relationships-between-documents/#data-modeling-example-one-to-one

Model One-to-Many Relationships with Embedded Documents

https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-many-relationships-between-documents/#data-modeling-example-one-to-many

Embedding

- For 1:1 or 1:many
- Document limit to 16MB, consider document growth

Referencing

- _id field is referenced in the related document
- Application runs 2nd query to retrieve the data
- Data duplication vs performance gain
- Object referenced by many different sources
- Models complex Many:Many & hierarchical structures

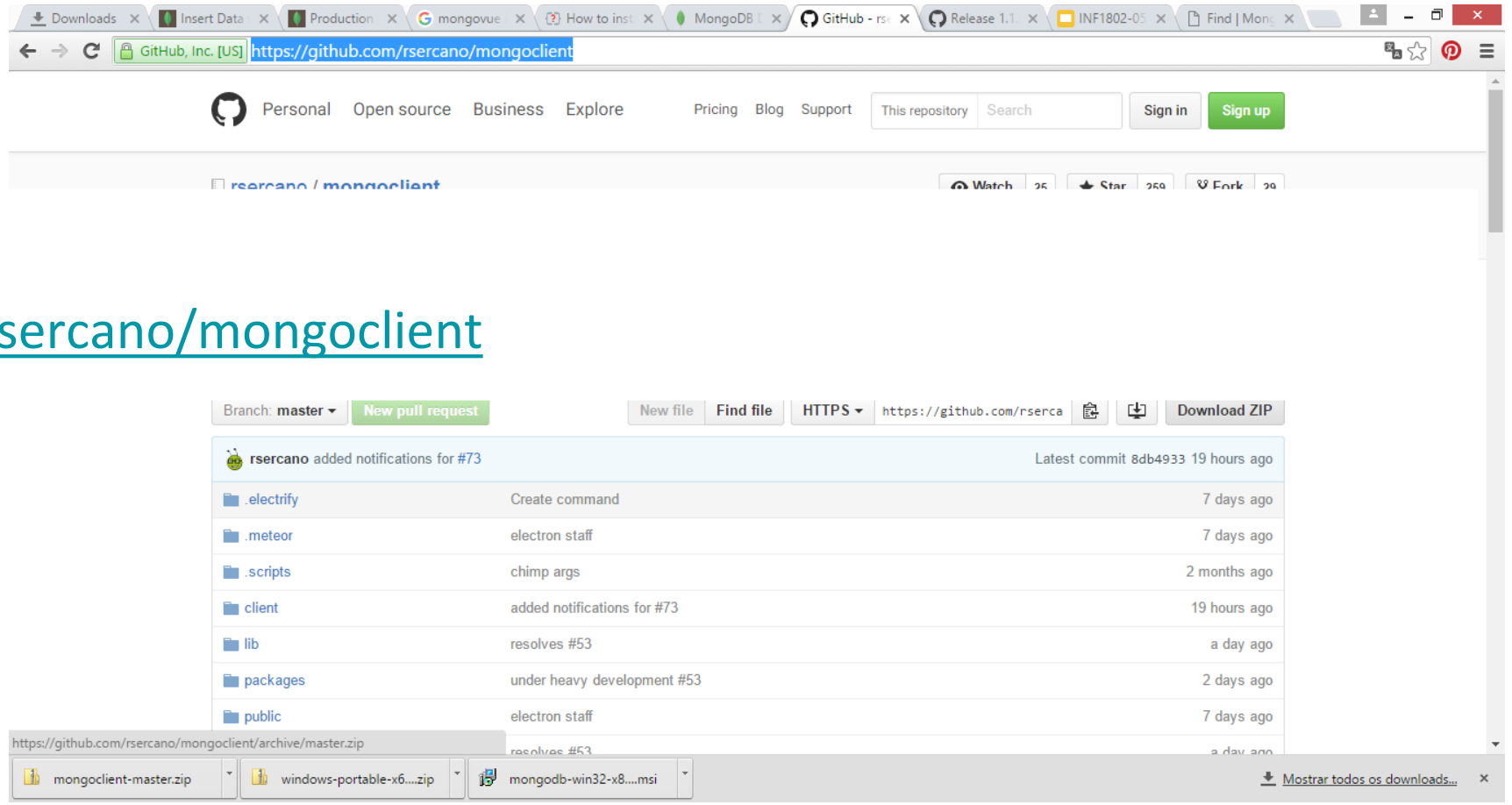
Using a mongoDB Client

Suggestion: http://www.mongoclient.com/

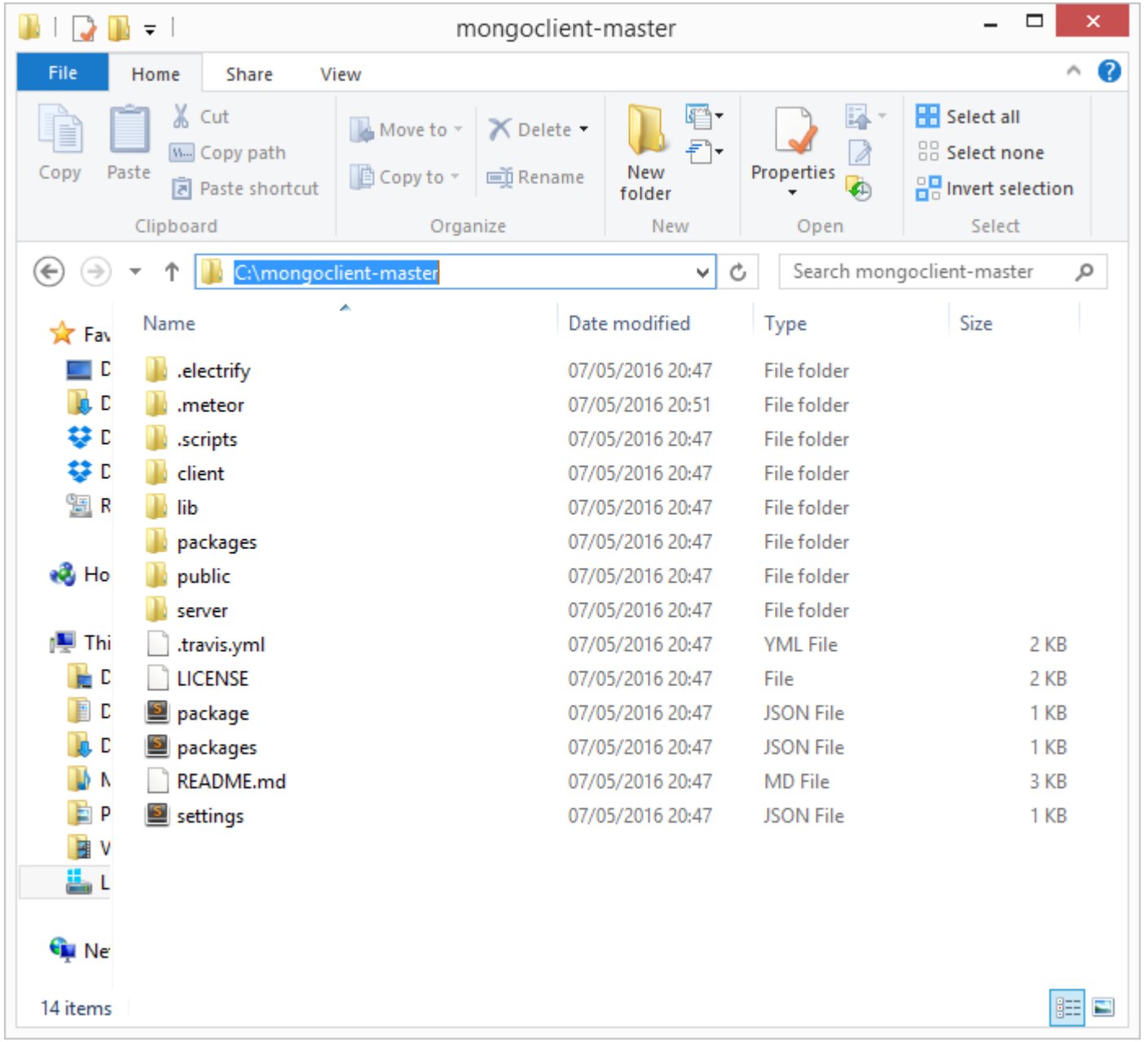
MongoClient

Step 1 - Download zip file

from https://github.com/rsercano/mongoclient



Step 2 - Unzip it. Example C:\mongoclient-master.



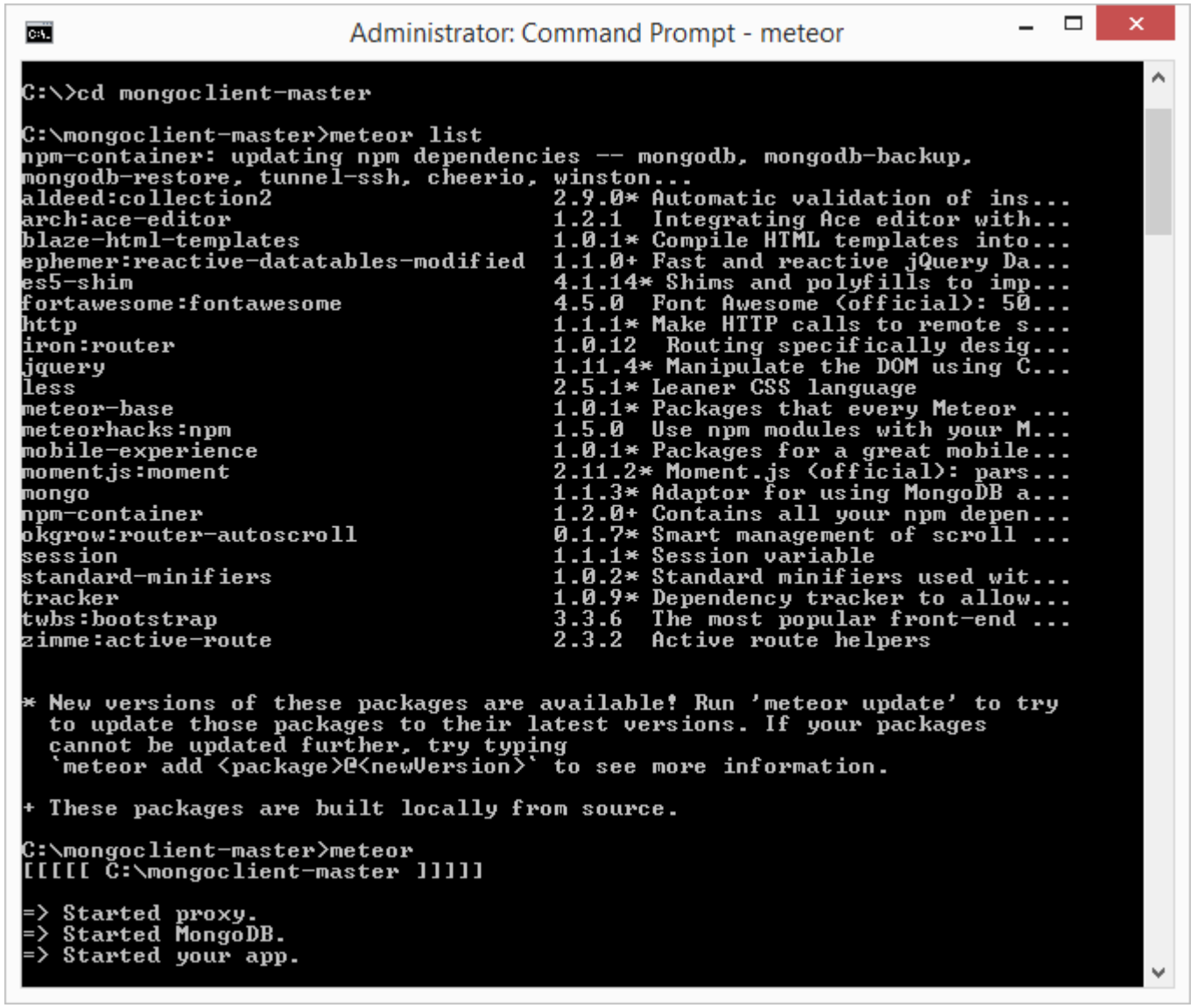
Step 3 - Pre-requisite:
Meteor installed.

Open Command Prompt.

In the mongoclient-master folder:

1. Run the command *meteor list* to install the application mongoclient.

2. Run the command *meteor* to start the application



Step 4 - Open the browser.

http://localhost:3000/

1.Click Connection

2.Add Connection

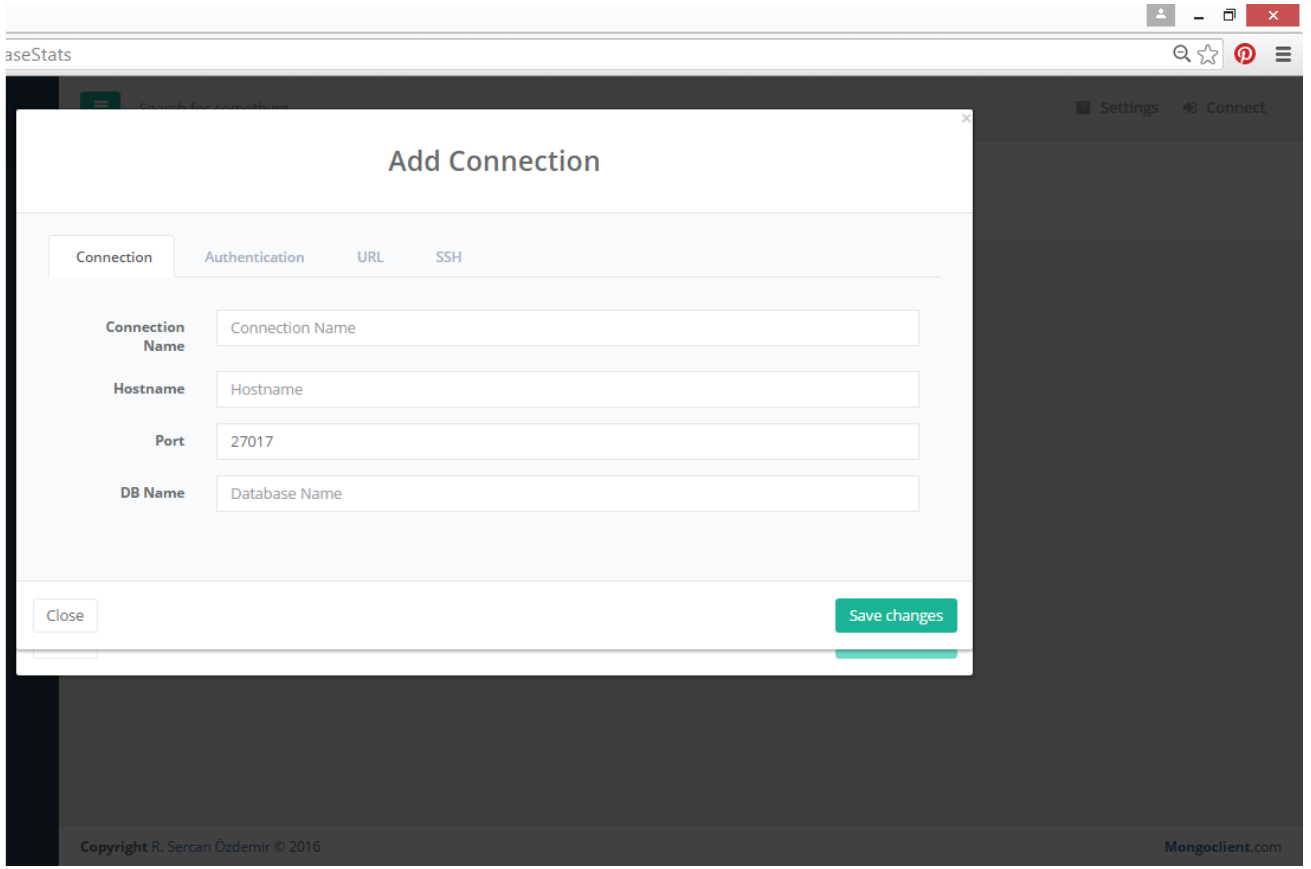
Connection Name: inf1802
(you can choose)

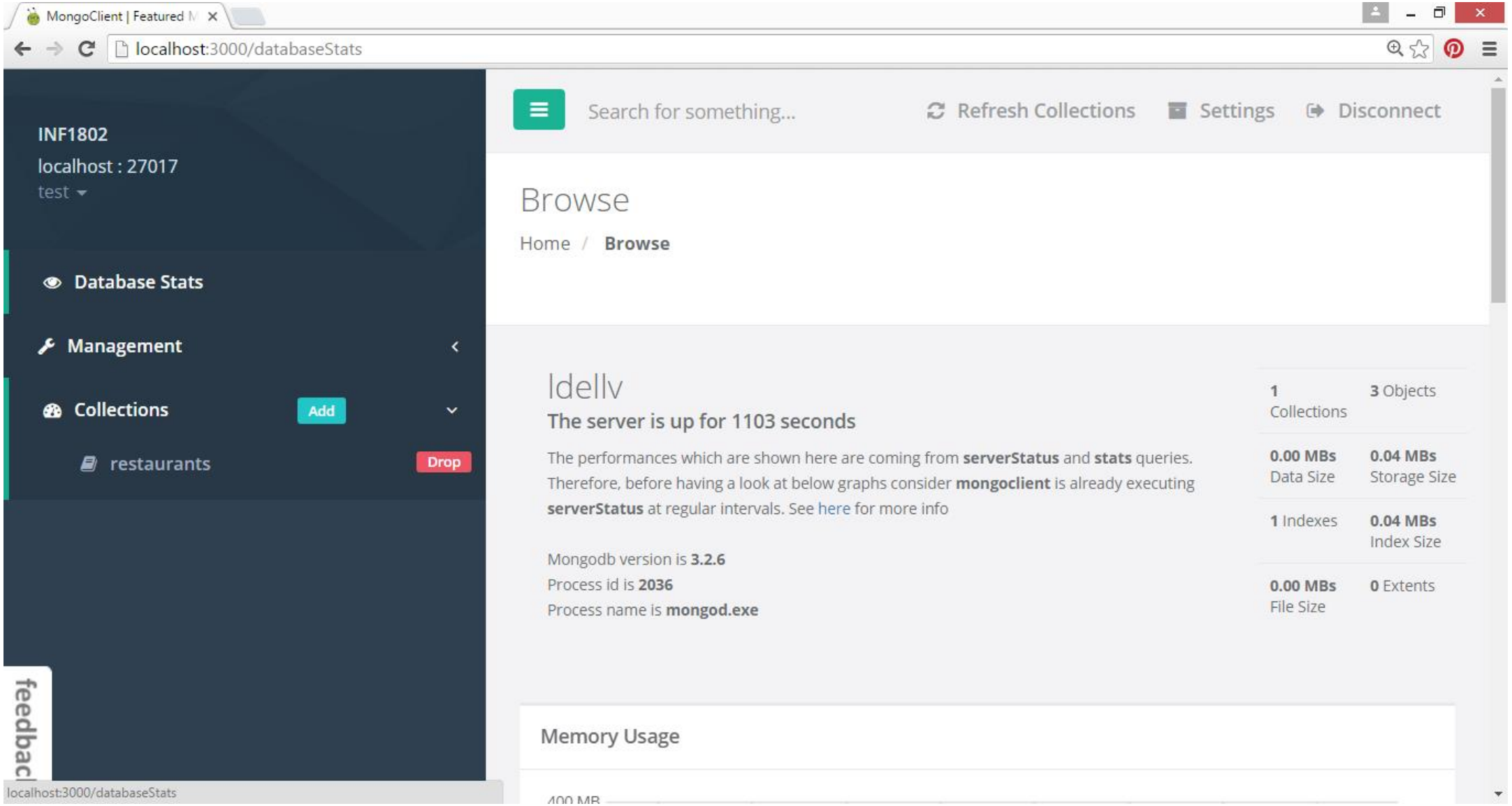
Hostname: localhost

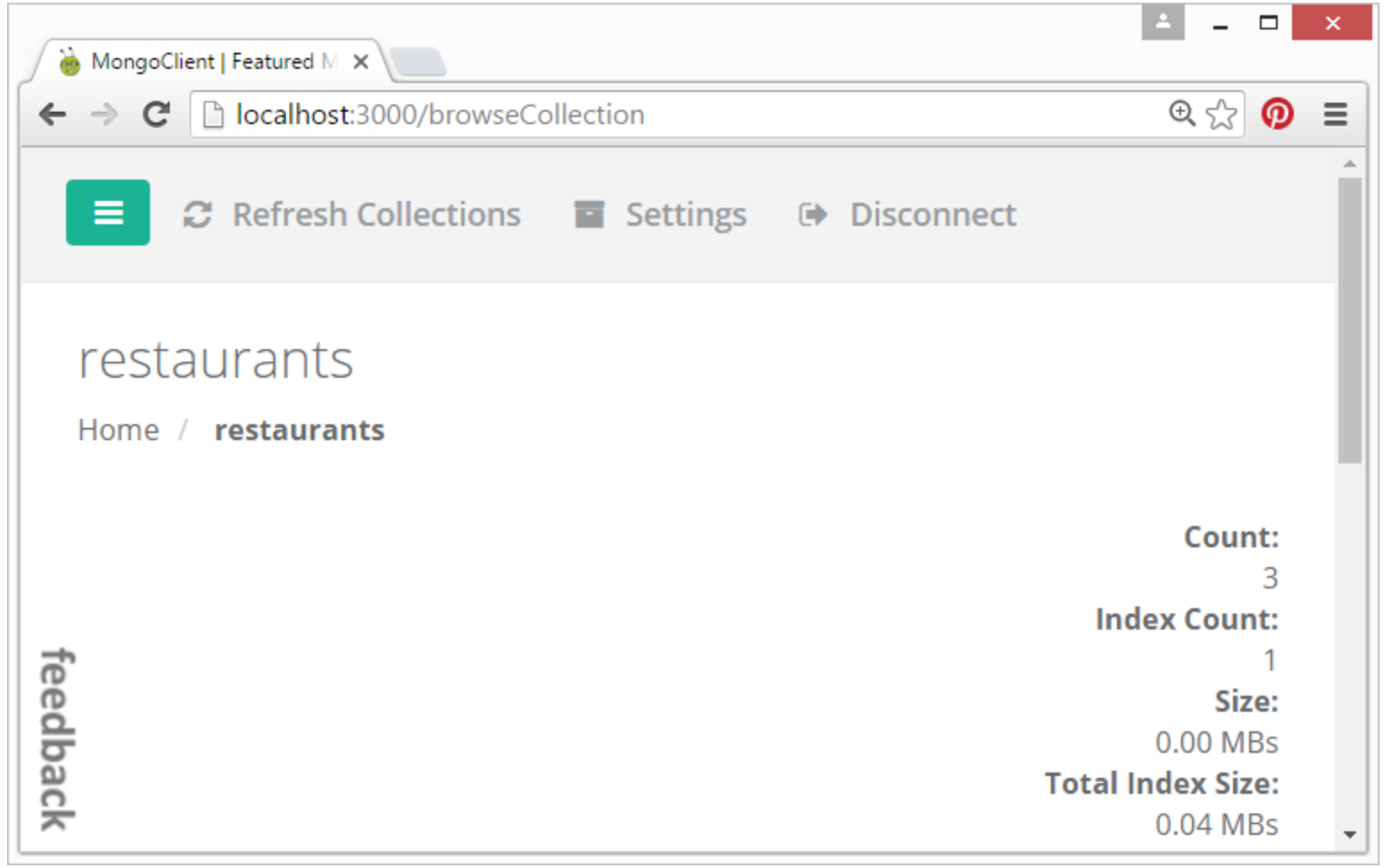
Port: 27017

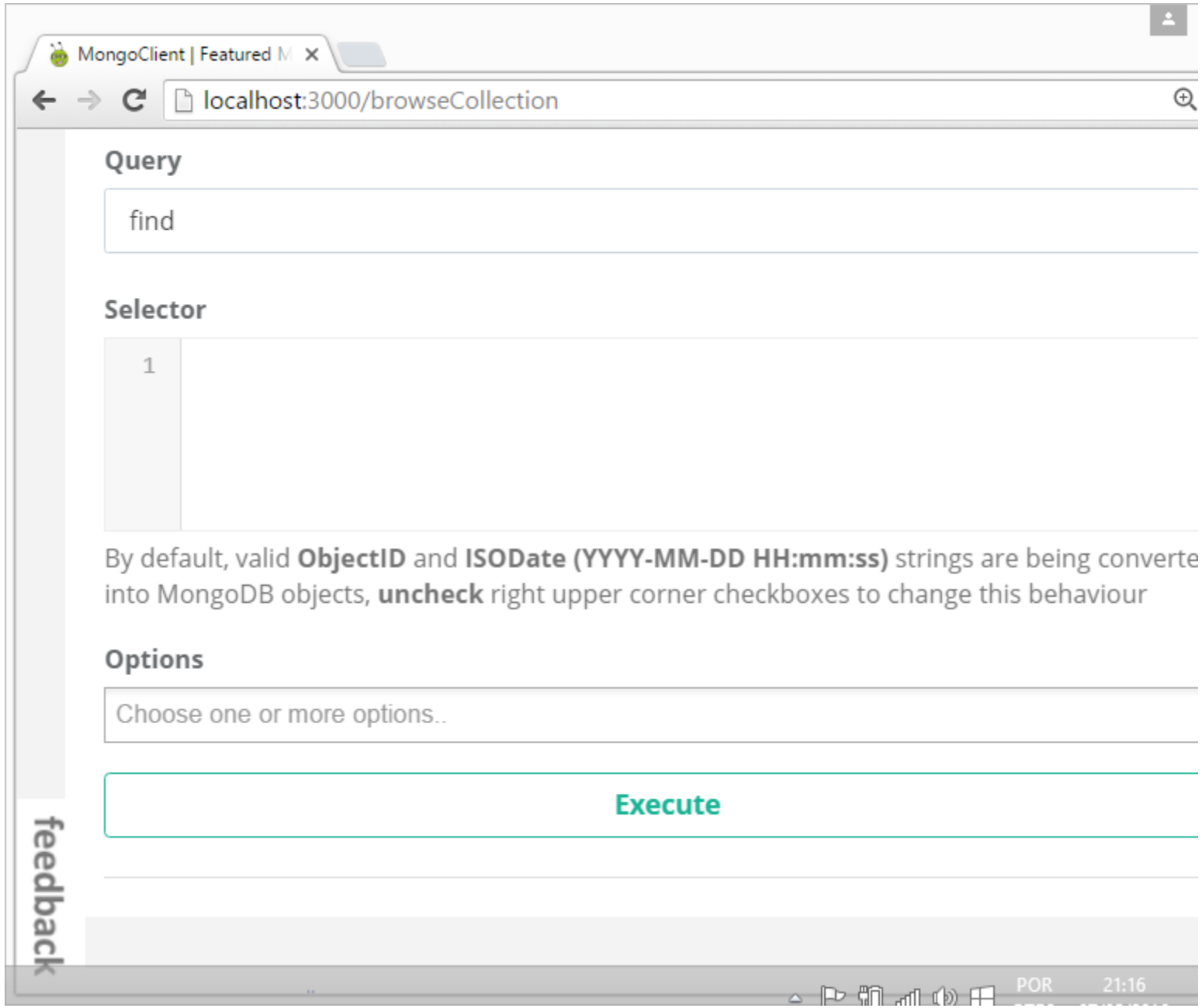
DB Name: test

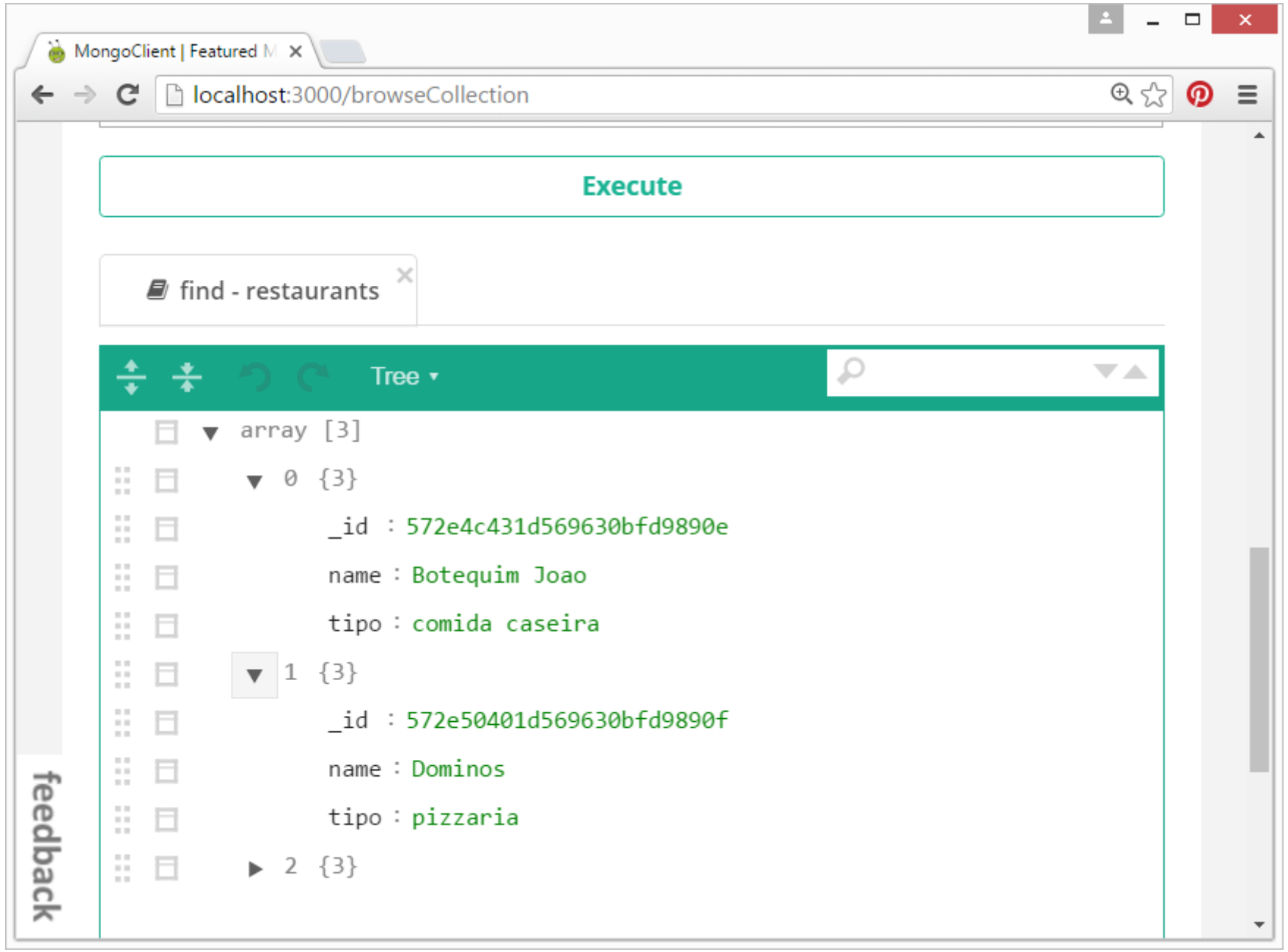
1.Connect

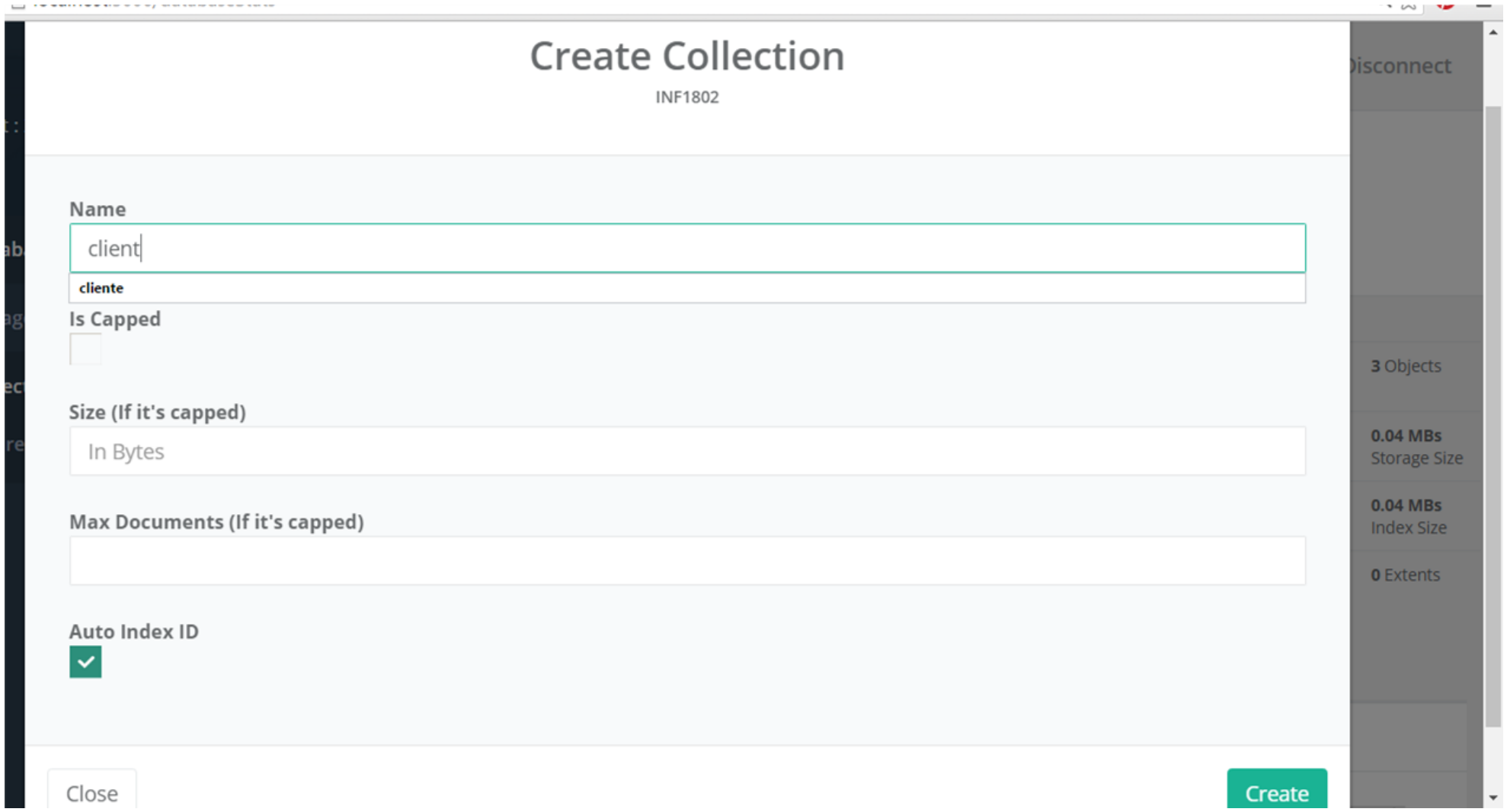


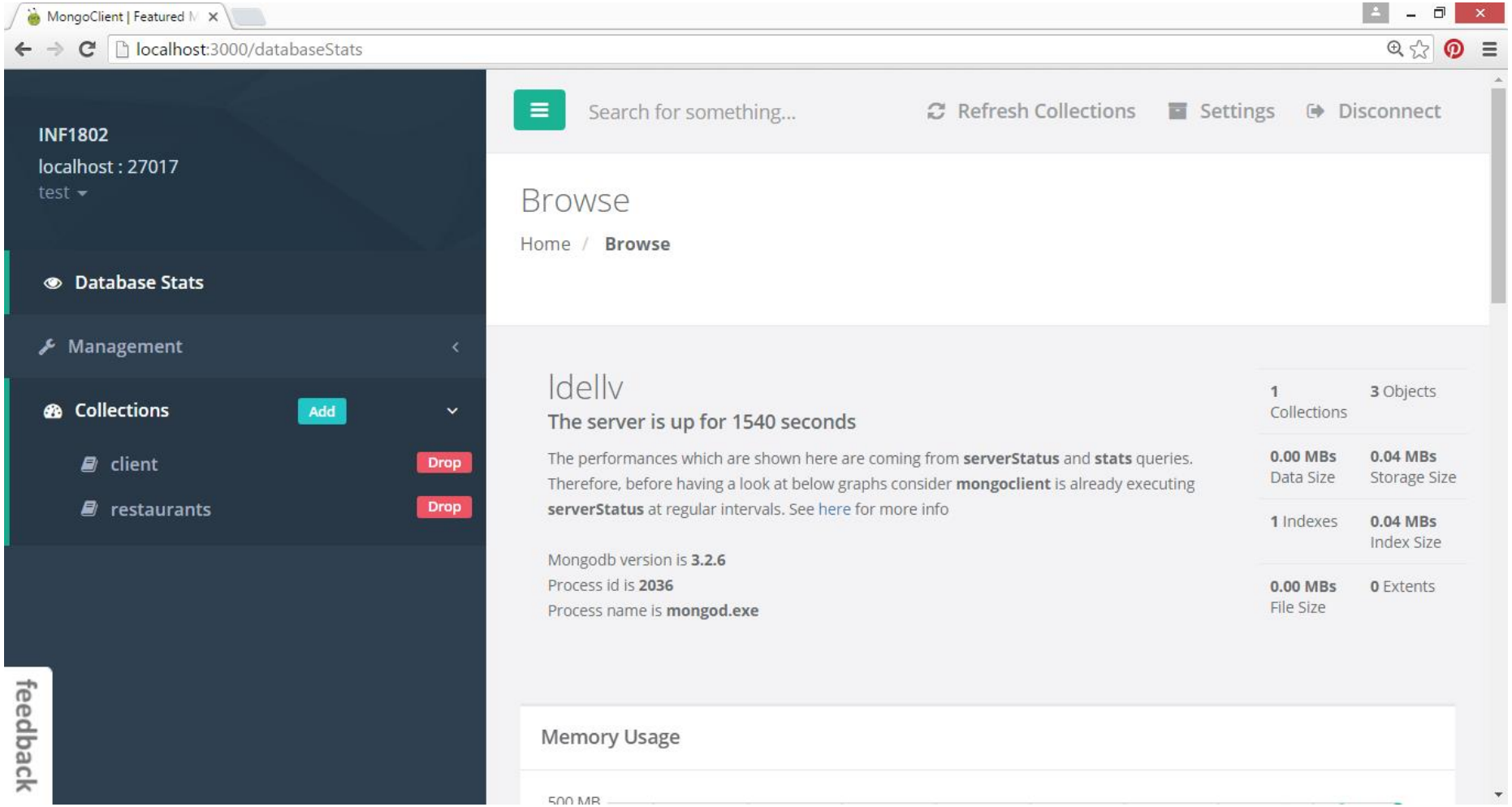

MongoClient | Featured M
localhost:3000/databaseStats
INF1802
localhost : 27017
test
Database Stats
Management
Collections
Add
restaurants
Drop
Search for something...
Refresh Collections
Settings
Disconnect
Browse
Home / Browse
Idellv
The server is up for 1103 seconds
The performances which are shown here are coming from serverStatus and stats queries. Therefore, before having a look at below graphs consider mongoclient is already executing serverStatus at regular intervals. See here for more info
Mongodb version is 3.2.6
Process id is 2036
Process name is mongod.exe
1 Collections
3 Objects
0.00 MBs Data Size
0.04 MBs Storage Size
1 Indexes
0.04 MBs Index Size
0.00 MBs File Size
0 Extents
Memory Usage
400 MB
feedback
localhost:3000/databaseStats

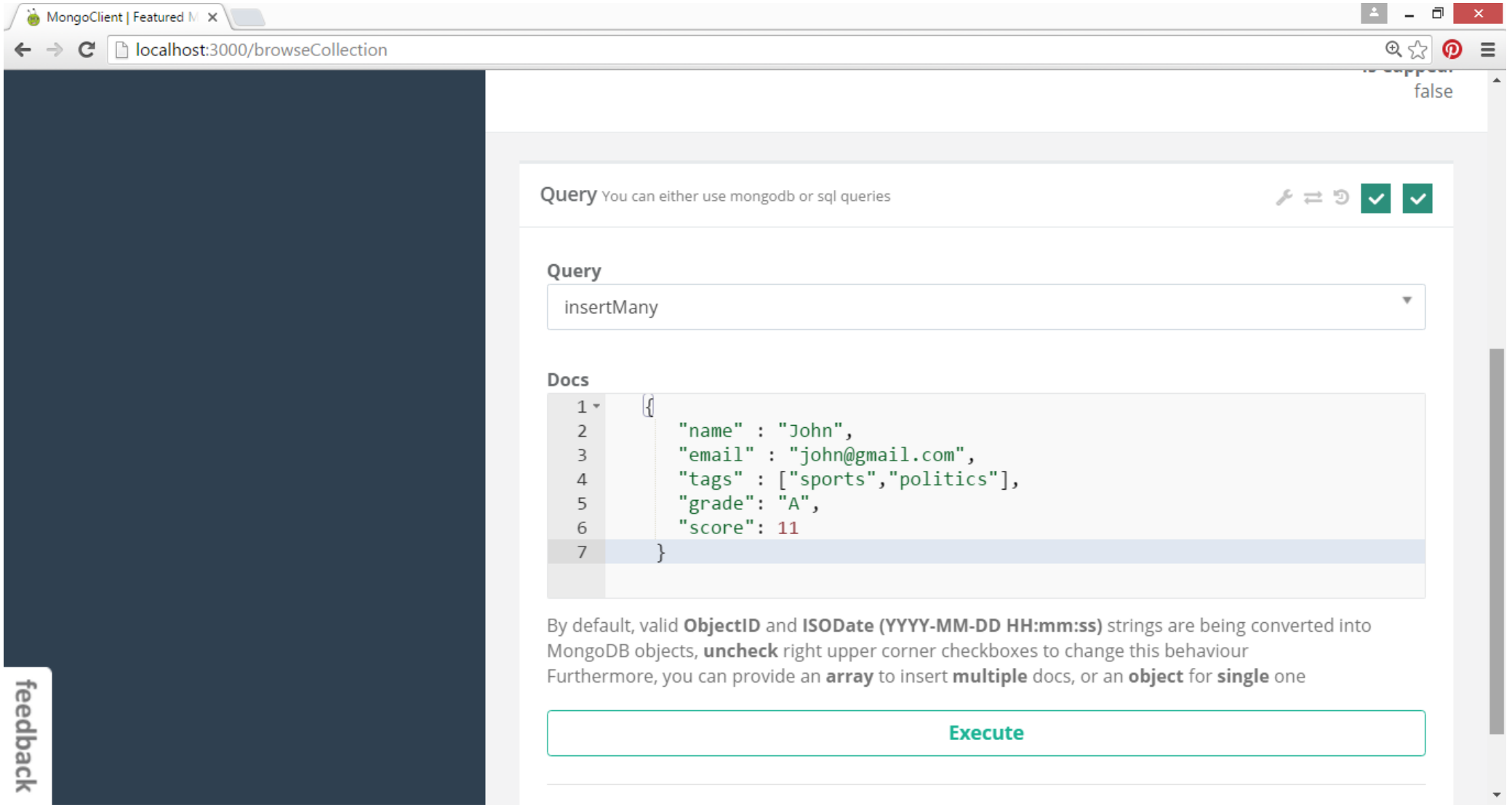

MongoClient | Featured
localhost:3000/browseCollection
Refresh Collections
Settings
Disconnect
restaurants
Home / restaurants
Count:
3
Index Count:
1
Size:
0.00 MBs
Total Index Size:
0.04 MBs
feedback

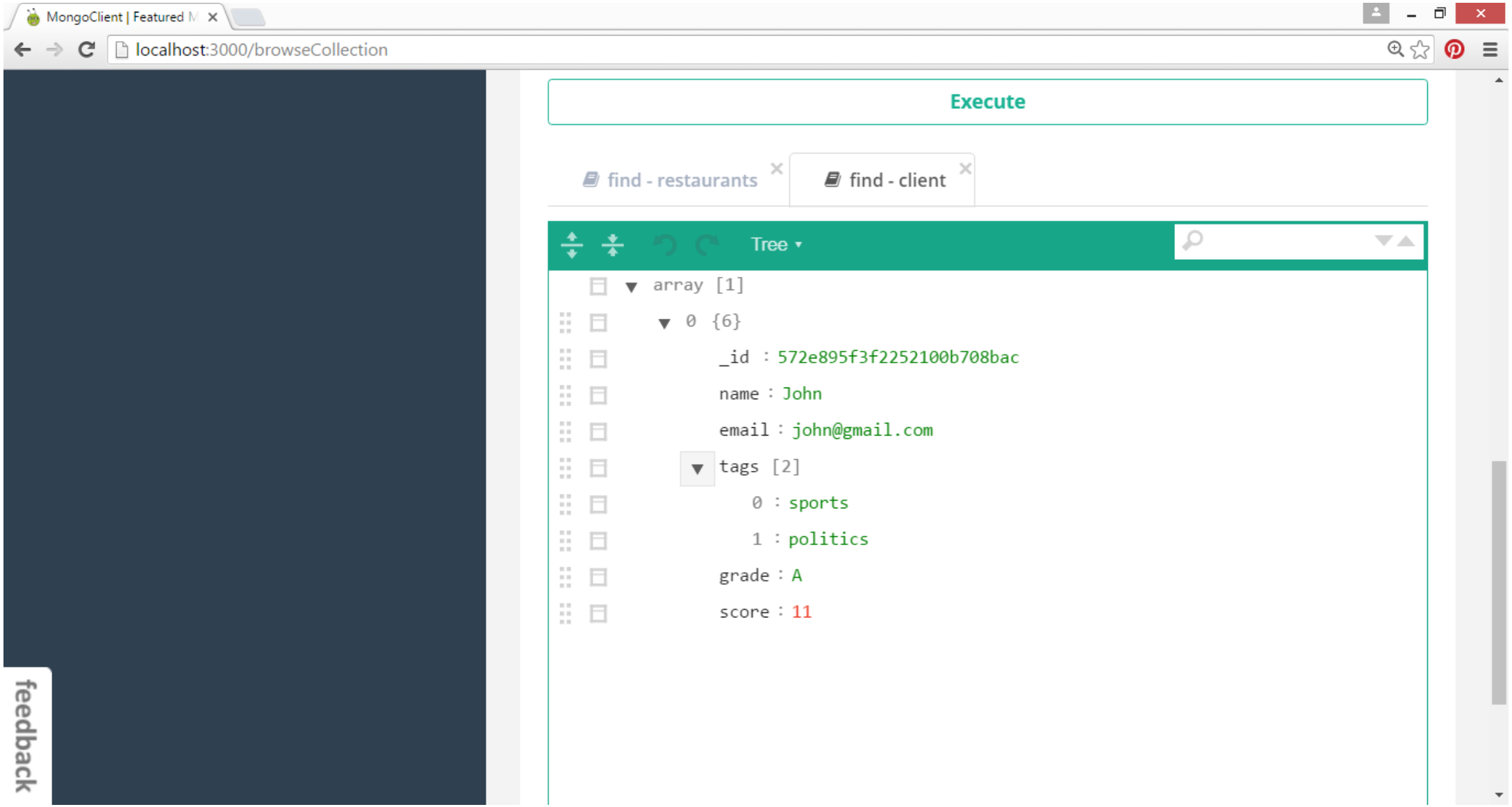

MongoClient | Featured M
localhost:3000/browseCollection
Query
find
Selector
1
By default, valid ObjectID and ISODate (YYYY-MM-DD HH:mm:ss) strings are being converte
into MongoDB objects, uncheck right upper corner checkboxes to change this behaviour
Options
Choose one or more options..
Execute
feedback
POR
21:16


MongoClient | Featured M
localhost:3000/browseCollection
Execute
find - restaurants
Tree
array [3]
0 {3}
_id : 572e4c431d569630bfd9890e
name : Botequim Joao
tipo : comida caseira
1 {3}
_id : 572e50401d569630bfd9890f
name : Dominos
tipo : pizzaria
2 {3}
feedback


Create Collection
INF1802
Name
client
cliente
Is Capped
Size (If it's capped)
In Bytes
Max Documents (If it's capped)
Auto Index ID
Close
Create
Disconnect
3 Objects
0.04 MBs
Storage Size
0.04 MBs
Index Size
0 Extents


MongoClient | Featured M
localhost:3000/databaseStats
INF1802
localhost : 27017
test
Database Stats
Management
Collections
Add
client
Drop
restaurants
Drop
feedback
Search for something...
Refresh Collections
Settings
Disconnect
Browse
Home / Browse
Idellv
The server is up for 1540 seconds
The performances which are shown here are coming from serverStatus and stats queries. Therefore, before having a look at below graphs consider mongoclient is already executing serverStatus at regular intervals. See here for more info
Mongodb version is 3.2.6
Process id is 2036
Process name is mongod.exe
1 Collections
3 Objects
0.00 MBs Data Size
0.04 MBs Storage Size
1 Indexes
0.04 MBs Index Size
0.00 MBs File Size
0 Extents
Memory Usage
500 MB


MongoClient | Featured
localhost:3000/browseCollection
false
Query You can either use mongodb or sql queries
Query
insertMany
Docs
{
"name" : "John",
"email" : "john@gmail.com",
"tags" : ["sports","politics"],
"grade": "A",
"score": 11
}
By default, valid ObjectID and ISODate (YYYY-MM-DD HH:mm:ss) strings are being converted into MongoDB objects, uncheck right upper corner checkboxes to change this behaviour
Furthermore, you can provide an array to insert multiple docs, or an object for single one
Execute
feedback


MongoClient | Featured M
localhost:3000/browseCollection
Execute
find - restaurants
find - client
Tree
array [1]
0 {6}
_id : 572e895f3f2252100b708bac
name : John
email : john@gmail.com
tags [2]
0 : sports
1 : politics
grade : A
score : 11
feedback