MongoDB

An Introduction

What’s MongoDB?

A schema-less database
Non-relational database
Stores data as JSON
Stores hierarchical data

SQL vs NoSQL (Why MongoDB?)

SQL	NoSQL
Rigid Structure	Schema less
Stores entries as rows	Each entry is a document(JSON)
Tables	Collections
Columns	Keys
Data fetched using SQL	Object Oriented access

Relational data model

Highly-structured table organization with rigidly-defined data formats and record structure.

Document data model

Collection of complex documents with arbitrary, nested data formats and varying "record" format.

What’s JSON?

- A way of representing data.
- Consists of key-value pairs and arrays.
- Supports hierarchical data.

- {‘name’: ‘anirudh’}
- {‘name’: ‘anirudh’, age: 19}
- {‘name’: ‘anirudh’, age: 19, ‘pet’: {‘name’: tazo, ‘breed’: ‘Japanese Spitz’}
- {‘name’: ‘anirudh’, ‘subjects’: [‘English’, ‘Maths’, ‘History’]}

Installing MongoDB

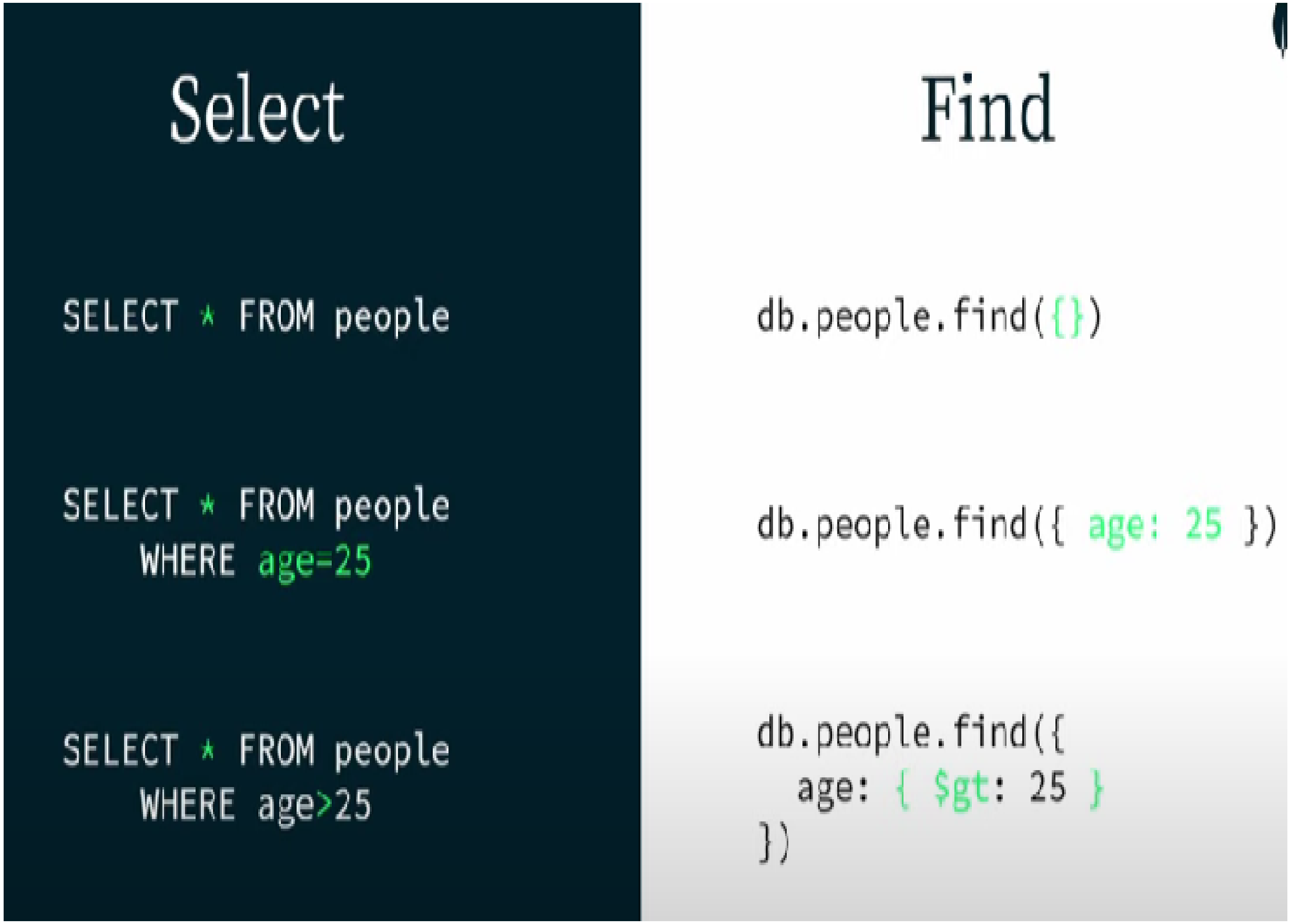
1. Download mongodb
   https://fastdl.mongodb.org/linux/mongodb-linux-i686-3.0.5.tgz
2. tar -zxvf ~/Downloads/mongodb-linux-i686-3.0.5.tgz
3. mkdir ~/mongodb
4. cp -a ~/mongodb-linux-i686-3.0.5/. ~/mongodb
5. sudo mkdir -p /data/db
6. sudo chown -R $(whoami) /data
7. cd ~/mongodb/bin
8. ./mongod
9. Open another terminal window
10. ~/mongodb/bin/mongo

MongoDB Shell - Basic Operations

db	Shows the name of the currently active database
show dbs	Show a list of all available databases
use dbname	Makes the database mentioned after use keyword active
show collections	Shows all the collections present in the currently active database
db.col_name.insert(document)	Insert the JSON document inside the ‘col_name’ collection
db.col_name.find()	List all documents in the ‘col_name’ collection

MongoDB Shell - Basic Operations

db.col_name.find(SEL_CRITERIA)	List all documents that match the selection criteria
db.col_name.update(SEL_CRITERIA, $set: {NEW_DATA})	Modify data or add new data to an existing document that matches the SEL_CRITERIA
db.col_name.update(SEL_CRITERIA, $set: {NEW_DATA}, {multi: true})	Modify data or add new data to all documents that matches the SEL_CRITERIA
db.col_name.remove()	Remove all documents in a collection
db.col_name.remove(SEL_CRITERIA)	Remove all documents in a collection that match the SEL_CRITERIA



Select

```
SELECT * FROM people
```

```
SELECT * FROM people
    WHERE age=25
```

```
SELECT * FROM people
    WHERE age>25
```

Find

```
db.people.find({})
```

```
db.people.find({ age: 25 })
```

```
db.people.find({
  age: { $gt: 25 }
})
```

Example

```
> use library

switched to db library

> db

library

> db.books.insert({"title": "Harry Potter", "author": "JK Rowling", "part": 3, "publisher": "Bloomsbury"})

WriteResult({ "nInserted" : 1 })

> db.books.find( )

{ "_id" : ObjectId("55cb530b4ec394ab29d49353"), "title" : "Harry Potter", "author" : "JK Rowling", "part" : 3,
"publisher" : "Bloomsbury" }

> db.books.insert({title: "The Hunger Games", author: "Suzanne Collins"})

WriteResult({ "nInserted" : 1 }
```

Example

PRETTY

```
> db.books.find().pretty()
{
	"_id" : ObjectId("55cb530b4ec394ab29d49353"),
	"title" : "Harry Potter",
	"author" : "JK Rowling",
	"part" : 3,
	"publisher" : "Bloomsbury"
}
{
	"_id" : ObjectId("55cb55cecb58e0459b8def7b"),
	"title" : "The Hunger Games",
	"author" : "Suzanne Collins"
}
```

$exists

```
> db.books.find({part: {$exists: true}})
{ "_id" : ObjectId("55cb530b4ec394ab29d49353"), "title" : "Harry Potter", "author" : "JK Rowling", "part" : 3,
"publisher" : "Bloomsbury" }
```

Example

```
> db.books.find({title: "Harry Potter"})
{ "_id" : ObjectId("55cb530b4ec394ab29d49353"), "title" : "Harry Potter", "author" : "JK Rowling", "part" : 3,
"publisher" : "Bloomsbury" }
```

$set

```
> db.books.update({title: "Harry Potter"},{$set: {copies_sold: 123456}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.books.find({title: "Harry Potter"}).pretty()
{
        "_id" : ObjectId("55cb530b4ec394ab29d49353"),
        "title" : "Harry Potter",
        "author" : "JK Rowling",
        "part" : 3,
        "publisher" : "Bloomsbury",
        "copies_sold" : 123456
}
> db.books.update({title: "The Hunger Games"}, {$set: {copies_sold: 654321}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

$gt

```
> db.books.find({copies_sold: {$gt: 234567}})
{ "_id" : ObjectId("55cb55cecb58e0459b8def7b"), "title" : "The Hunger Games", "author" : "Suzanne Collins",
"copies_sold" : 654321 }
```

Example

AND

```
> db.books.find({title: "Harry Potter", part: 3})
{ "_id" : ObjectId("55cb530b4ec394ab29d49353"), "title" : "Harry Potter", "author" : "JK Rowling", "part" : 3,
"publisher" : "Bloomsbury", "copies_sold" : 123456 }
```

$or

```
> db.books.find({$or:[{title: "Harry Potter"}, {author: "Suzanne Collins"}]})
{ "_id" : ObjectId("55cb55cecb58e0459b8def7b"), "title" : "The Hunger Games", "author" : "Suzanne
Collins", "copies_sold" : 654321 }
{ "_id" : ObjectId("55cb530b4ec394ab29d49353"), "title" : "Harry Potter", "author" : "JK Rowling", "part" : 3,
"publisher" : "Bloomsbury", "copies_sold" : 123456 }

> db.books.remove({title: "Harry Potter"})
WriteResult({ "nRemoved" : 1 })
> db.books.find()
{ "_id" : ObjectId("55cb55cecb58e0459b8def7b"), "title" : "The Hunger Games", "author" : "Suzanne
Collins", "copies_sold" : 654321 }
```

Query Operators

Operation	Syntax	Example
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()
Less Than	{<key>:{$lt:<value>}}	db.mycol.find({"likes":{$lt:50}}).pretty()
Less Than Equals	{<key>:{$lte:<value>}}	db.mycol.find({"likes":{$lte:50}}).pretty()
Greater Than	{<key>:{$gt:<value>}}	db.mycol.find({"likes":{$gt:50}}).pretty()
Greater Than Equals	{<key>:{$gte:<value>}}	db.mycol.find({"likes":{$gte:50}}).pretty()
Not Equals	{<key>:{$ne:<value>}}	db.mycol.find({"likes":{$ne:50}}).pretty()