In this lecture, we will discuss…

- ✧ Rationale behind NoSQL
- ✧ Scaling Issues in RDBMS
- ✧ NoSQL: What is it?

Why RDBMS

- Relational Databases – popular and commonly used
- Initially designed for non distributed
- Low Cost RDBMS alternatives (PostgreSQL, MySQL, SQLLite)
- Very Transactional - across tables and commands, and can even be transactional across distributed resources (XA) -- at a cost
- Supports Joins -- across multiple tables allowing for normalized forms of data to be stored once

Why NoSQL

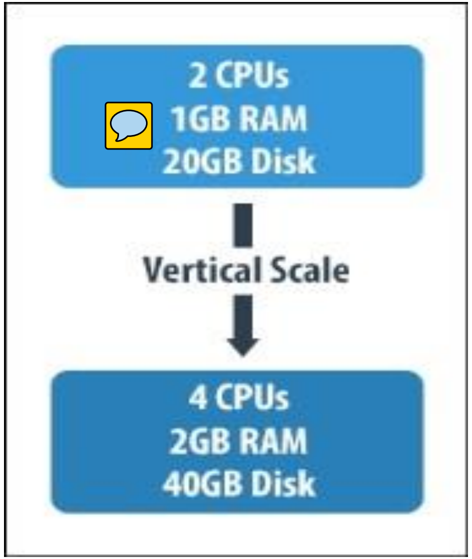
- Explosion in data
- Object/Relational Impedance mismatch
  - Objects are constantly being moved in/out of tables/rows
- RDBMS normalization and joins are powerful, but add up in cost
  - Complex objects stored across many tables and rows can be expensive to handle

Why NoSQL

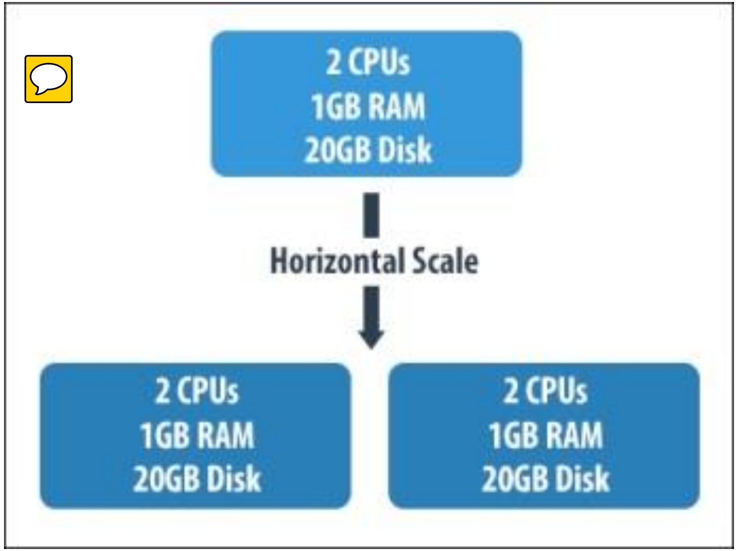
- ✧ "Big" data handling with better performance
- ✧ Supports unstructured data
  - Unique data type extensions can be easily integrated into existing collections
- ✧ Operational issues (scale, performance and availability)

Scaling Out

Vertical Scaling



Horizontal Scaling



What is NoSQL

- Stands for “Not Only SQL”
- No Fixed Schema
- Non-relational data storage systems

apache CouchDB relax

cassandra

redis

mongoDB

Couchbase

amazon DynamoDB

APACHE HBASE

MEMCACHED

Summary

- ✧ NoSQL – very popular and major companies especially social networking sites such as Twitter, Facebook, LinkedIn, and Digg use NoSQL DB
- ✧ Excellent performance and stability, fast and scalable and fairly simple model
- ✧ Supports unstructured format, which makes it very agile
- ✧ NoSQL is mostly gained when access patterns to complex objects are understood and modeled correctly up front

What’s Next?

Categories of NoSQL

In this lecture, we will discuss…

- Categories of NoSQL
- NoSQL vs. RDBMS

Categories of NoSQL – Key/Value

- Value can be String or JSON
- Key-value hash
- Solutions
  - Dynamo
  - Redis
  - Memcached

ID	Attributes
1234	John Doe

1235	{
	"Name": "Godfather",
	"Genre": "Drama",
	"Actor": "Robert DeNiro",
	"Director": "Francis Ford Coppola"
	}

Categories of NoSQL – Document

- Stores documents based up of tagged elements
- Persistent and query-able
- Solutions
  - MongoDB
  - CouchDB

```
{
   "id": 1234,
   "name": "Departed",
   "actors": [
      {
         "actor": "Leonardo DeCaprio"
      },
      {
         "actor": "Jack Nicholson"
      }
   ],
   "director": "Martin Scorsese",
   "genre": "drama"
}
```

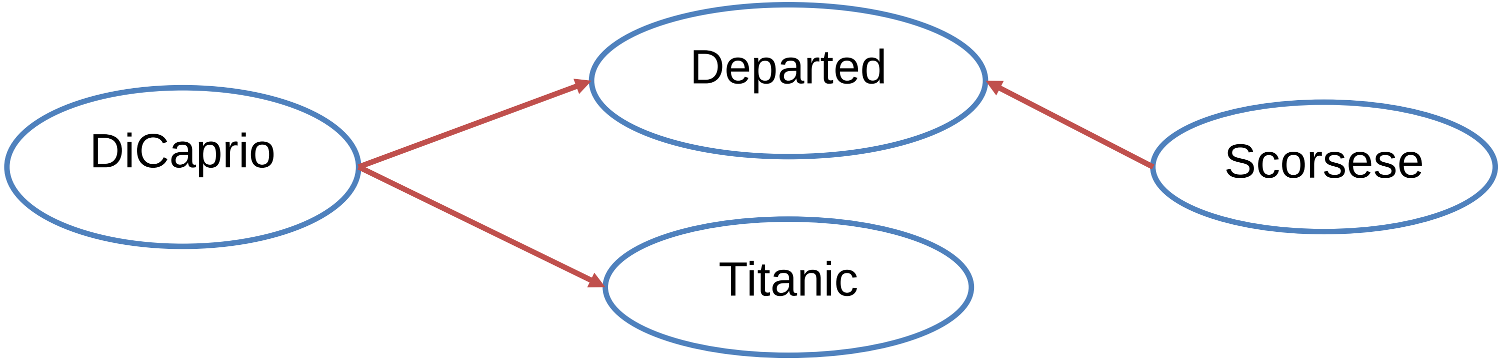
Categories of NoSQL – Column

- Uses flat structure, but with keys stored in columns rather than rows:
- Solutions
  - Cassandra
  - Hbase

ID	101	102	103
Name	The Godfather	The Departed	Titanic
Actor	Leonardo DiCaprio	Al Pacino	Leonardo DiCaprio
Director	Francis Ford Coppola	Martin Scorsese	James Cameron

Categories of NoSQL – Graph

- A network database that uses edges and nodes to represent and store data
- Solutions
  - Neo4J



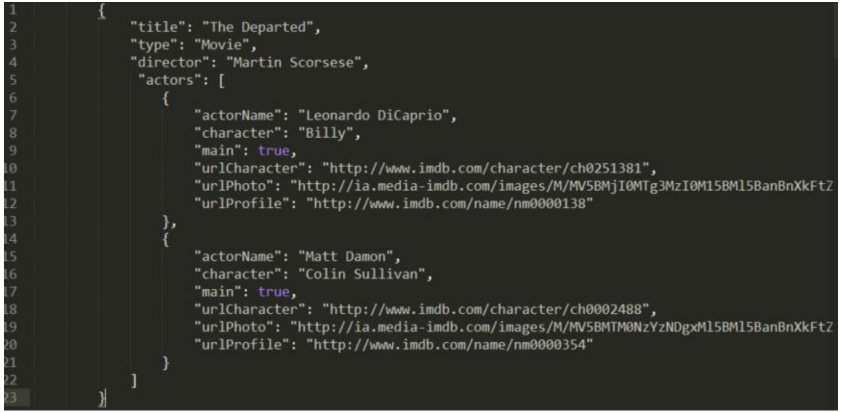
NoSQL – Not supported

✧ Joins are not supported

- Embedded documents or in middle tier code

✧ ACID Transactions

- Supported at a document level only



```
{
    "title": "The Departed",
    "type": "Movie",
    "director": "Martin Scorsese",
     "actors": [
        {
            "actorName": "Leonardo DiCaprio",
            "character": "Billy",
            "main": true,
            "urlCharacter": "http://www.imdb.com/character/ch0251381",
            "urlPhoto": "http://ia.media-imdb.com/images/M/MV5BMjI0MTg3MzI0M15BMl5BanBnXkFtZ
            "urlProfile": "http://www.imdb.com/name/nm0000138"
        },
        {
            "actorName": "Matt Damon",
            "character": "Colin Sullivan",
            "main": true,
            "urlCharacter": "http://www.imdb.com/character/ch0002488",
            "urlPhoto": "http://ia.media-imdb.com/images/M/MV5BMTM0NzYzNDgxMl5BMl5BanBnXkFtZ
            "urlProfile": "http://www.imdb.com/name/nm0000354"
        }
    ]
}
```

NoSQL vs RDBMS – How to pick?

- ✧ Nature of data
  - Row/column (structured) – RDBMS
  - Unstructured, complex (geo-spatial or engineering data) which needs nesting - NoSQL
- ✧ Schema
  - Static – RDBMS, Dynamic – NoSQL

NoSQL vs RDBMS – How to pick?

- ✧ Self contained – NoSQL, Joins – RDBMS
- ✧ Flexibility of query
  - RDBMS Joins allow for flexibility
  - NoSQL - Duplication of data, implement joins in middle-ware

Summary

- 4 different categories offering different choices
- Pick what is best for your application (Relational or NoSQL)

What’s Next?

MongoDB

In this lecture, we will discuss…

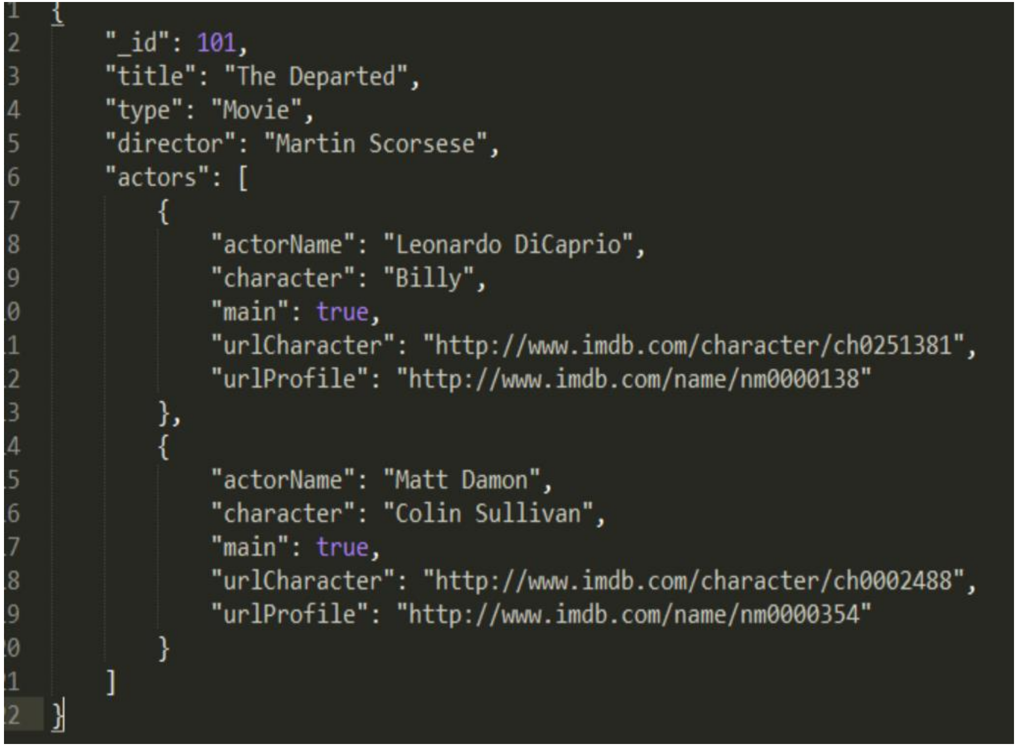
- ✧ What MongoDB is
- ✧ Reasons to use MongoDB

What is MongoDB

- Created by 10gen (term coined from hu**mongo**us)
- Definition:
  - MongoDB is an open source, document-oriented database designed with both scalability and developer agility in mind
- Storage: JSON-like documents and “schemaless”
- Well suited for Object Oriented programming

What is MongoDB?

- Stores data in BSON format (Binary JSON)
- Binary form for representing simple data structures and associative arrays



```
{
    "_id": 101,
    "title": "The Departed",
    "type": "Movie",
    "director": "Martin Scorsese",
    "actors": [
        {
            "actorName": "Leonardo DiCaprio",
            "character": "Billy",
            "main": true,
            "urlCharacter": "http://www.imdb.com/character/ch0251381",
            "urlProfile": "http://www.imdb.com/name/nm0000138"
        },
        {
            "actorName": "Matt Damon",
            "character": "Colin Sullivan",
            "main": true,
            "urlCharacter": "http://www.imdb.com/character/ch0002488",
            "urlProfile": "http://www.imdb.com/name/nm0000354"
        }
    ]
}
```

Document Store (Mapping)

RDBMS	MongoDB
Database	Database
Table, View	Collection
Row	JSON Document
Column	Field
Index	Index
Join	Embedded Document / Linking across Document
Foreign Key	Reference
Partition Key	Shard

Sample Query – SQL vs. Mongo

SQL	Mongo
CREATE TABLE movies(movieId int NOT NULL AUTO_INCREMENT, name VARCHAR(30), rating VARCHAR(6), PRIMARY KEY (movieId))	db.movies.insert({ "id": 10, "name": "Titanic", "rating": "R" })
SELECT * FROM movies	db.movies.find()
UPDATE movies SET rating = "NR" WHERE movieId = 101	db.movies.update({"id": 101 }, { $set: { rating: "NR" } })
DELETE FROM movies WHERE rating = "R"	db.movies.remove({ "rating": "R" })

Why MongoDB?

- ✧ “Queryable” documents
- ✧ No impedance mismatch between object and DB form
  - Ideal for web applications (fast retrieval)
- ✧ Quick and easy integration of new data variations
- ✧ Rich API support (multiple languages)

Ruby On Rails & Mongo

- ✧ Ruby Driver
  - http://docs.mongodb.org/ecosystem/tutorial/ruby-driver-tutorial/
- ✧ Mongoid
  - http://docs.mongodb.org/ecosystem/tutorial/ruby-mongoid-tutorial/

MongoDB Users

MetLife

SAP®

Expedia

Disney

Firebase

sourceforge

Forbes

foursquare™

Doodle®

theguardian

bit.ly

ADP®

craigslist

CollegeHumor

MongoDB Core Topics With Ruby/Rails

- ✧ MongoDB Ruby Driver
- ✧ Aggregation Framework
- ✧ GridFS – breaking large files in to smaller chunks
- ✧ Geo Spatial - index and query geospatial data
- ✧ Mongoid

Summary

- ✧ Open Source DB
- ✧ Automatic Scaling
- ✧ High Performance
- ✧ “Schema-less” and Document Oriented

What’s Next?

- ✧ MongoDB Installation

In this lecture, we will discuss…

- ✧ Install MongoDB
- ✧ Configure MongoDB
- ✧ Start MongoDB - `mongod`
- ✧ Launch MongoDB shell - `mongo`

MongoDB Installation Steps

https://www.mongodb.org/downloads

Download latest version

Supports all platforms

- ✧ Step 1: Download MongoDB (msi)
- ✧ Step 2: Mongo needs a default data folder (very important step)
  - Ex: /data/db or C:\data\db
  - Note: You can pick any folder but will need to provide the path while starting MongoDB

Helpful Configuration

- ✧ Journaling in MongoDB – allocates 3GB upfront
  - Write-ahead logging to guarantee write operations
- ✧ For casual development, turn off (maybe?)
- ✧ Setting "nojournal=true“ in mongod.conf will keep mongo from claiming this space for write-ahead journaling : `mongod --config /etc/mongod.conf`

Note: Do not turn off journaling in production system

Starting MongoDB

- ✧ Open a CMD window and go to the `$mongo_install/bin` folder

- ✧ Step 1 : Start mongoDB
  - `mongod`

- ✧ Step 2 : Start mongo shell
  - `mongo` – reference

- ✧ *Note: If your db path is not the default, make sure to launch with this command -* `mongod –dbpath /<path>`
  - Directory needs to have write permission

Summary

- MongoDB supports all OS
- MongoDB needs data folder
- Starting MongoDB - `mongod`
- Launching MongoDB shell – `mongo`

What’s Next?

- MongoDB Basics

In this lecture, we will discuss…

- ✧ Importing sample data
- ✧ Basics of MongoDB shell
- ✧ MongoDB collections
- ✧ IRB shell and MongoDB
- ✧ Basic MongoDB commands in IRB

MongoDB Basics

- Import dataset
  - Download sample zips.json file from MongoDB
  - Save the above file
  - Run the import command as in
-
  ```
  > mongoimport --db test --collection zips --drop --file zips.json
  ```

Database, Documents and Collections

- ✧ Mongo can create database on the fly
  - No need to create database beforehand
- ✧ Documents
  - Unit of storing data in a MongoDB database
  - JSON document
- ✧ Collection (similar to tables in DB)
  - Unit of storing data in a MongoDB database
  - Collection of documents

Collection Types

✧ Capped Collection

- Fixed-size collections that support high-throughput operations
- Insert and retrieve documents based on insertion order
- Once a collection fills its allocated space, it makes room for new documents by overwriting the oldest documents in the collection
- `db.createCollection("log", { capped : true, size : 5242880, max : 5000 } )`

Mongo Basics

- Start mongo shell

    -
      ```
      $ mongo
      ```
- Switch to test database

  ```
  > use test
  ```

- Test the data with a simple find command (note: we will cover this in more depth later)

  ```
  > db.zips.findOne()
  ```

    - The above command will return a single document from the zips collection.

MongoDB Ruby Driver Setup

- ✧ mongo-ruby driver
  - `gem update --system`
  - `gem install mongo`
  - `gem install bson_ext`
- ✧ Using gem

```
> require mongo
```

MongoDB Basics (irb shell)

- ✧ Start irb shell
- ✧ Type the following commands:

```
> require 'mongo'
> Mongo::Logger.logger.level =
  ::Logger::INFO
> db =
Mongo::Client.new('mongodb://localhost:27017')
> db=db.use('test')
> db.database.name
> db.database.collection_names
> db[:zips].find.first
```

Summary

- ✧ Basics of MongoDB
- ✧ Database, Document and Collection
- ✧ MongoDB Ruby Driver
  - `<irb>` shell

What's Next?

- ✧ CRUD Operations

Next Topic…..

Lesson 2 – CRUD operations