Introduction to Database Systems: CS312
More On MongoDB

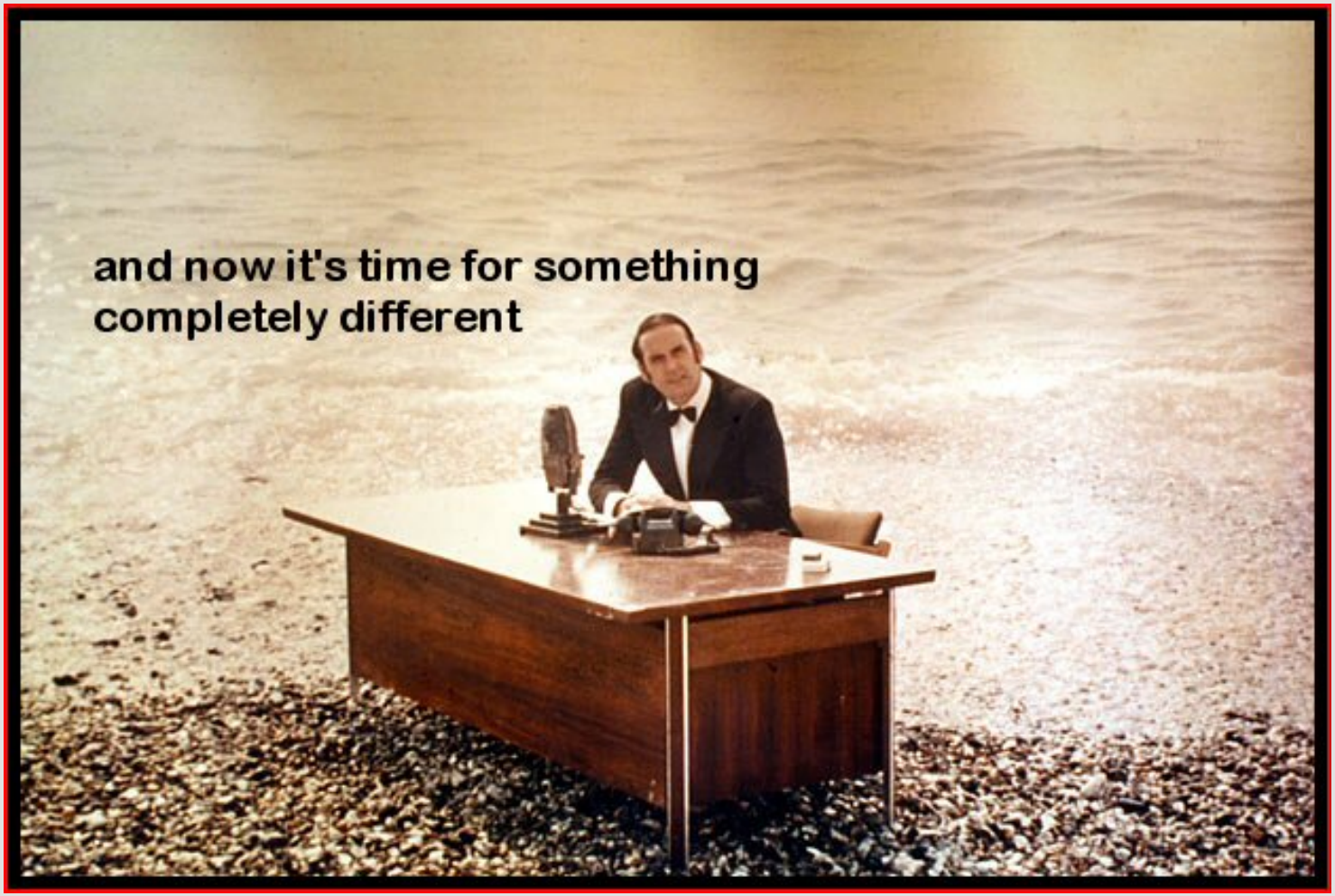
Oliver Bonham-Carter

27 March 2019

Quiz 2

Let's Review a Bit

Review
Big Data Calling
Comparison
Sharding
Start MongoDB



Time to Review

Quiz 2

Friday 29^{th} March, During Lab at 2:30pm

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

Terms

- Define the following terms.
  - Migrations
  - Normalization (of attributes and columns)
  - Data redundancy
  - Primary keys, used by Django
  - Foreign keys, again, used by Django

Simple answers

- Themes from lab work
  - How was Python used in a lab to perform queries (generally speaking)?
  - How were Python queries generalized to make query writing more simple?
  - What was the library that we used in Python to connect to a database?
  - Why is it necessary to have software able to manage a database and not a person?

Quiz 2

Friday 29^{th} March, During Lab at 2:30pm

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

Django commands

- Describe the basic command line operations and what they do in Django.

Django simple answers

- How did we design an App's *schema* in Django?
- Explain what the following lines of code do.

```
url(r'^(?P<album_id>[0-9]+)$', views.detail, name = 'detail'),
```

```
<li><a href="/music/{{ album.id }}/">{{ album.album_title }}</a></li>
```

NoSQL

- Theory of NoSQL
- MongoDB
  - System commands and simple queries

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

- `https://www.mongodb.com/`

NoSQL Versus RDBMS

Relational DB Management Systems such as SQLite3

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

- NoSQL has less stringent schema definition; schema is *defined* during population
- NoSQL designed to handle distributed, large databases
- NoSQL as established software? SQL systems were developed about 30 years ago and NoSQL systems, more recently. Have all the bugs been worked out in NoSQL code and theory?
- NoSQL support: SQL companies and firms (such as Microsoft and others) have been in business long enough to be able to have resources to support to clients all over the world. NoSQLs are often created by start-ups that may not have the necessary resources to supply large amounts world-wide support.

NoSQL Versus RDBMS

Relational DB Management Systems such as SQLite3

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

- Business Intel and Analytics:
  - NoSQL technology was built to handle Web 2.0 data handling,
  - playing to web app requirements that function beyond the *insert-read-update-delete* functionalities.
- Database Maintenance and Admin:
  - NoSQL databases are *smarter* in how they are able to deal with data but they are often more complicated to configure and taylor for specific types of applications. Think: MongoDB has own server to manage.

Direct Comparisons I

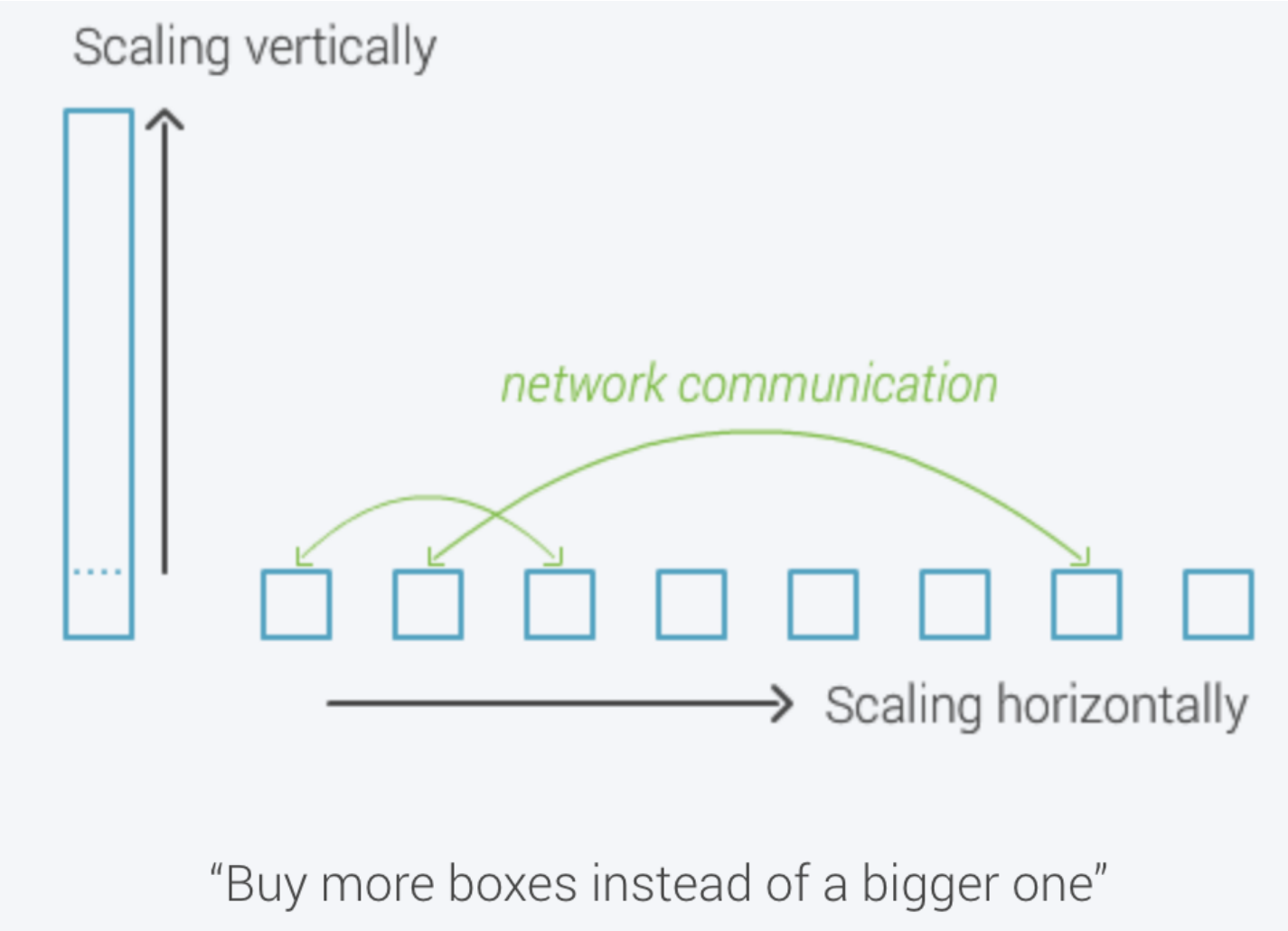
- Elastic scalability
  - RDBMSs are difficult to put into clusters,
  - NoSQL databases software is designed to take advantage of nodes and assimilate new nodes for transparent expansion.
  - Designed for use on low-cost commodity hardware.
- Big data applications
  - Each day we create even more data than the previous day.
  - RDBMS are able to keep up with the data requirements at a cost, however NoSQL systems are prepared to handle arbitrary amounts of data.

Direct Comparisons II

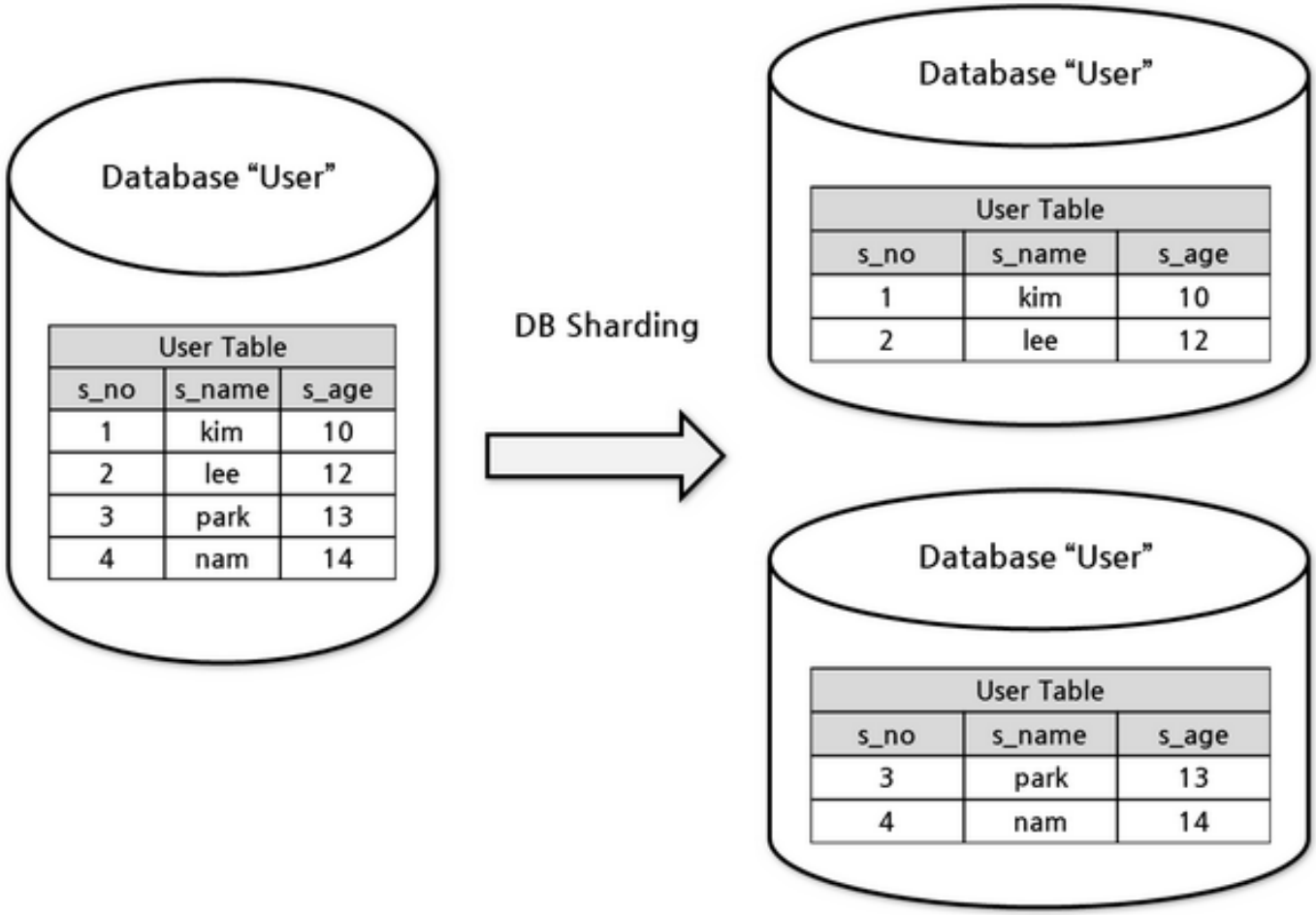
- Database administration
  - RDBMSs require the services of expensive administrators to design, install and maintain the systems
  - NoSQL still require lots of maintenance however, the *performance to maintenance* ratio is higher in terms of the amount of data services completed.
- Economy
  - RDBMSs require installation of expensive storage systems and proprietary servers
  - NoSQL databases can be easily installed in inexpensive commodity hardware clusters as transaction and data volumes increase. (Sharding is often built-in)

Scalable

Review
Big Data Calling
Comparison
Sharding
Start MongoDB



Sharding


Database "User"
User Table
s_no s_name s_age
1 kim 10
2 lee 12
3 park 13
4 nam 14
DB Sharding
Database "User"
User Table
s_no s_name s_age
1 kim 10
2 lee 12
Database "User"
User Table
s_no s_name s_age
3 park 13
4 nam 14

Sharding

Auto-sharding: Auto scalability

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

- Sharding: The partitioning of a large database into smaller parts (i.e., *shards*) which can be easily managed.
- A database shard is a horizontal partition of data in a database or search engine.
- An individual partition, called a *shard* is held on a separate database server instance to distribute the load across more hardware.
- Simple method: database containing data {A through Z} is placed into two smaller databases: database {A though H} and database {I though Z}

Special Features

- MongoDB handles sharding automatically
  - large-scale applications
- Scale horizontally over commodity hardware (sharding)
  - Lots of relatively inexpensive servers (Managing data may be expensive)
- Keep the functionality that works well in RDBMSs
  - Ad hoc queries
  - Fully featured indexes:
    - An index is a copy of selected columns of data from a table that can be searched very efficiently

Getting started with Mongo

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

Setup a data directory (if you have not already done so)

```
mkdir ~/mongodbData
```

Start the Mongo server with data directory as a parameter
Note: Control-C to exit.

```
mongod --dbpath ~/mongodbData/
```

With new another terminal, start the Mongo client

```
mongo
```

Find databases or collections, from Mongo's client

```
show dbs
show collections
```

Begin a new database, from Mongo's client

```
use myDB
```

Getting started with Mongo

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

Enter data into your first collection (i.e., a *table*)

```
db.people.insert({name:'James Bond', shoes:'brown'})
db.people.insert({name:'S. Holmes', shoes:'black'})
db.people.insert({name:'Wonder Woman', shoes:'boots'})
db.people.insert({name:'Batman', shoes:'black'})
db.people.insert({name:'Flash', shoes:'slippers'})
```

A general query of the collection (people)

```
db.people.find()
db.people.find().pretty()
```

- Where is the schema!?

Getting started with Mongo

Review
Big Data Calling
Comparison
Sharding
Start MongoDB

A specific query the collection (people)

```
db.people.find({shoes:'brown'})
db.people.find({shoes:'black'})
db.people.find({},{"name":1,"shoes":'brown',"_id":0})
db.people.find({},{"name":1,"shoes":'black',"_id":0})
db.people.find({},{"name":1,"shoes":'boots',"_id":0})

db.people.find({shoes:'black'}).pretty()
db.people.find({shoes:'boots'}).pretty()

db.people.find({name:'Wonder Woman'}).pretty()
db.people.find({name:'Batman'}).pretty()
db.people.find({"name":"S. Holmes"},{"shoes":'boots',"_id":0}).pretty()
```

Drop the collection (people): Destroy the data, remove collection

```
db.people.drop()
```