NoSQL and MongoDB

NoSQL vs SQL, MongoDB

SoftUni Team

Technical Trainers

Software University

https://about.softuni.bg/

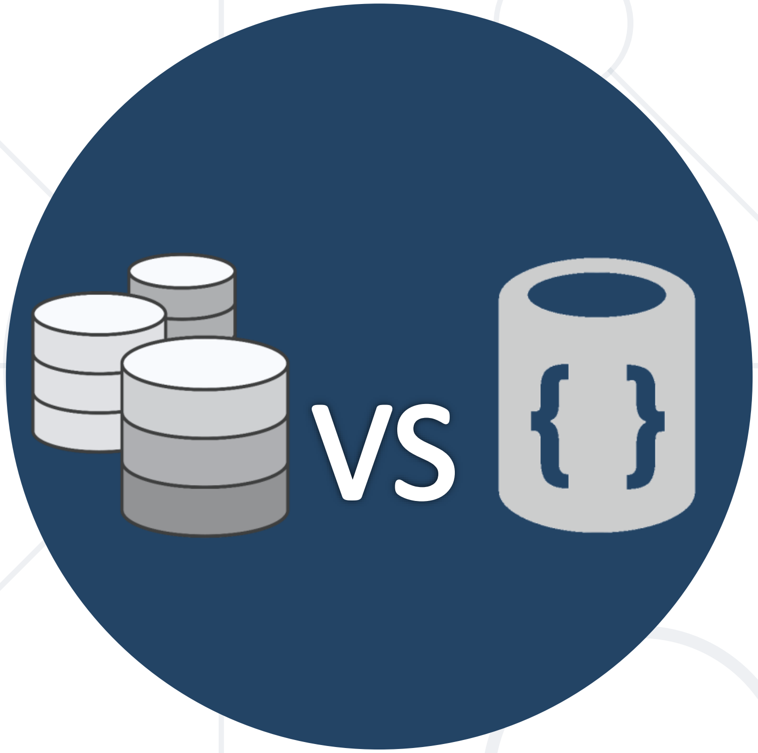
Table of Contents

Have a Question?

Software University

[sli.do](sli.do)

#csharp-db



Relational and Non-Relational Databases

Differences and Examples

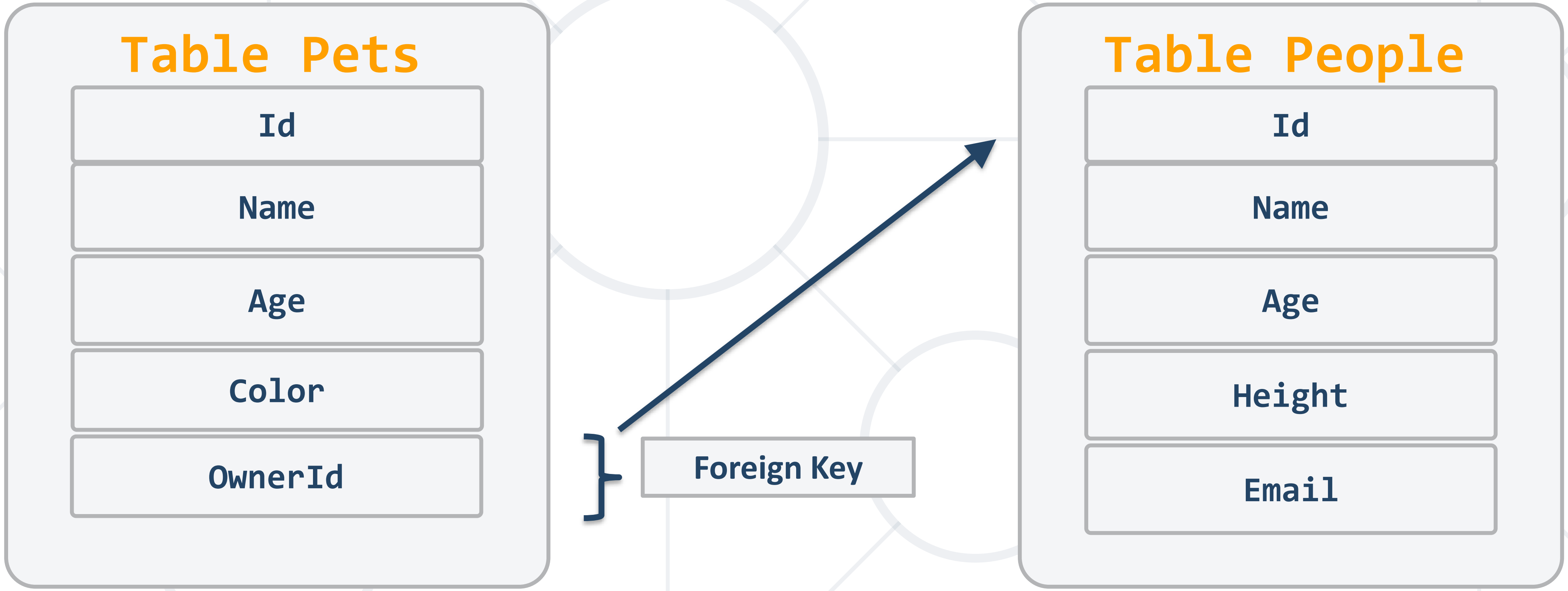
Relational Database

- Organizes data into one or more **tables** of **columns** and **rows**
- Unique **key** identifying each **row** of data
- Almost all relational databases use **SQL** to **extract** data

```
SELECT * FROM Students
```

- **Relations** between tables are done using **Foreign Keys (FK)**
- Such databases are **Oracle**, **MySQL**, **SQL Server**, etc..

Relational Database - Example



Non-relational Database (NoSQL)

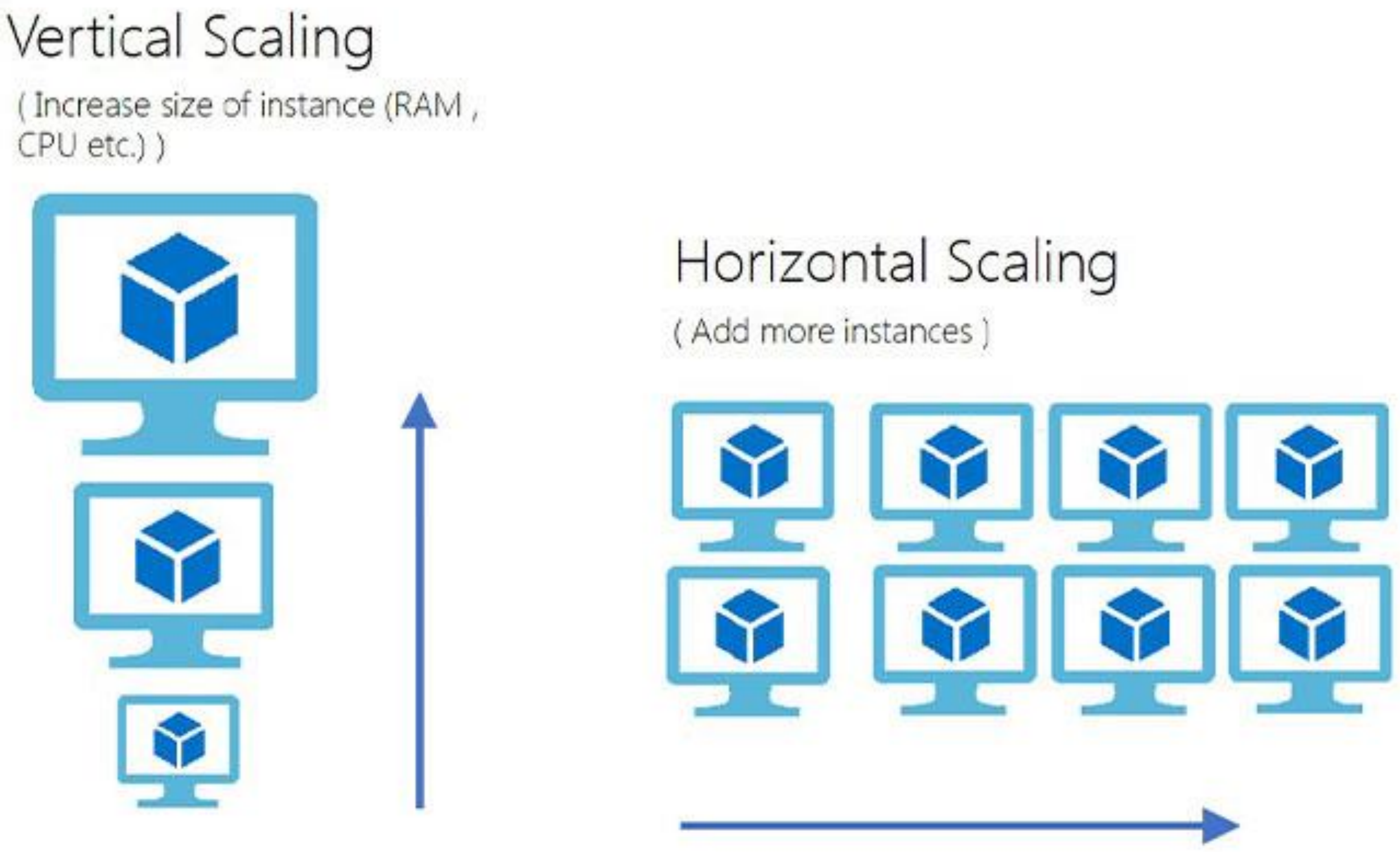
- NoSQL Databases are non tabular, and store data differently than relational tables
- Key-value **stores**

```
{
   "_id": ObjectId("59d3fe7ed81452db0933a871"),
   "email": "peter@gmail.com",
   "age": 22
}
```

- **SQL** query is **not** used in NoSQL systems
- More **scalable** and **provide** superior **performance**

Database Scalability

Database Scalability (1)



Database Scalability (2)

- The ability of a system's database to **scale up or down**, depending on the requirements
  - Enables the database to grow to a larger size to support more transactions
  - There are two types of database scalability
    - Vertical Scaling or Scale-up
    - Horizontal Scaling or Scale-out

Vertical Scaling

- Refers to the process of adding more physical resources to the existing database server for improving the performance such as
  - Storage
  - Memory
  - CPU
- Helps in upgrading the capacity of the existing database server

Vertical Scaling Pros and Cons

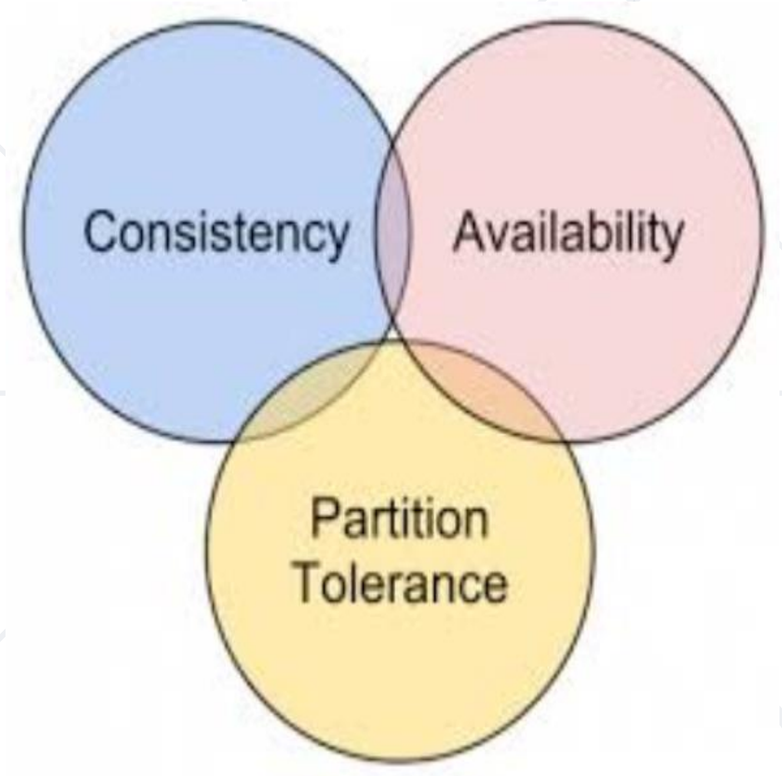
- **Pros**
  - It consumes less power
  - You need to handle and manage just one system
  - Cooling costs are less than horizontal scaling
  - Implementation isn't difficult
- **Cons**
  - Risk of hardware failure which can cause bigger outages
  - Limited scope of upgradeability in the future

Horizontal Scaling

- Adds more servers with less RAM and processors
  - The ability to **increase the capacity** by connecting multiple software or hardware entities in a such manner that they function as a single logical unit
  - If a cluster requires more resources to improve its performance and provide high availability, then **the administrator can scale-out by adding more servers** to the cluster

Horizontal Scaling Pros and Cons

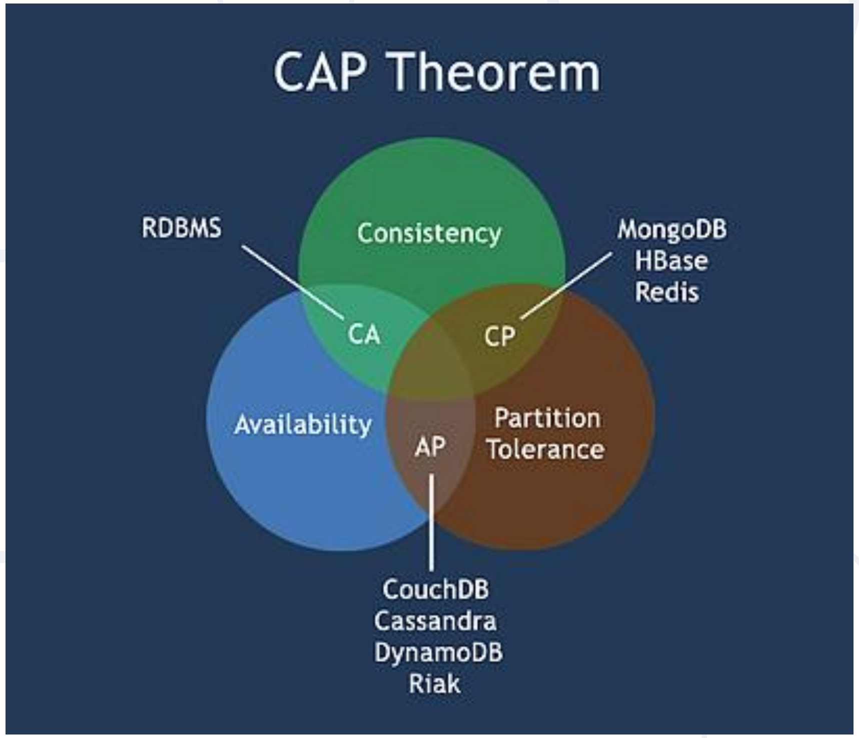
- **Pros**
  - Easy to upgrade
  - Resilience is improved due to the presence of discrete, multiple systems
  - Supports linear increases in capacity
- **Cons**
  - It has a bigger footprint in the Data Center
  - Adds complexity to the system
  - Introduces data syncing problems
  - Dependent on the CAP theorem



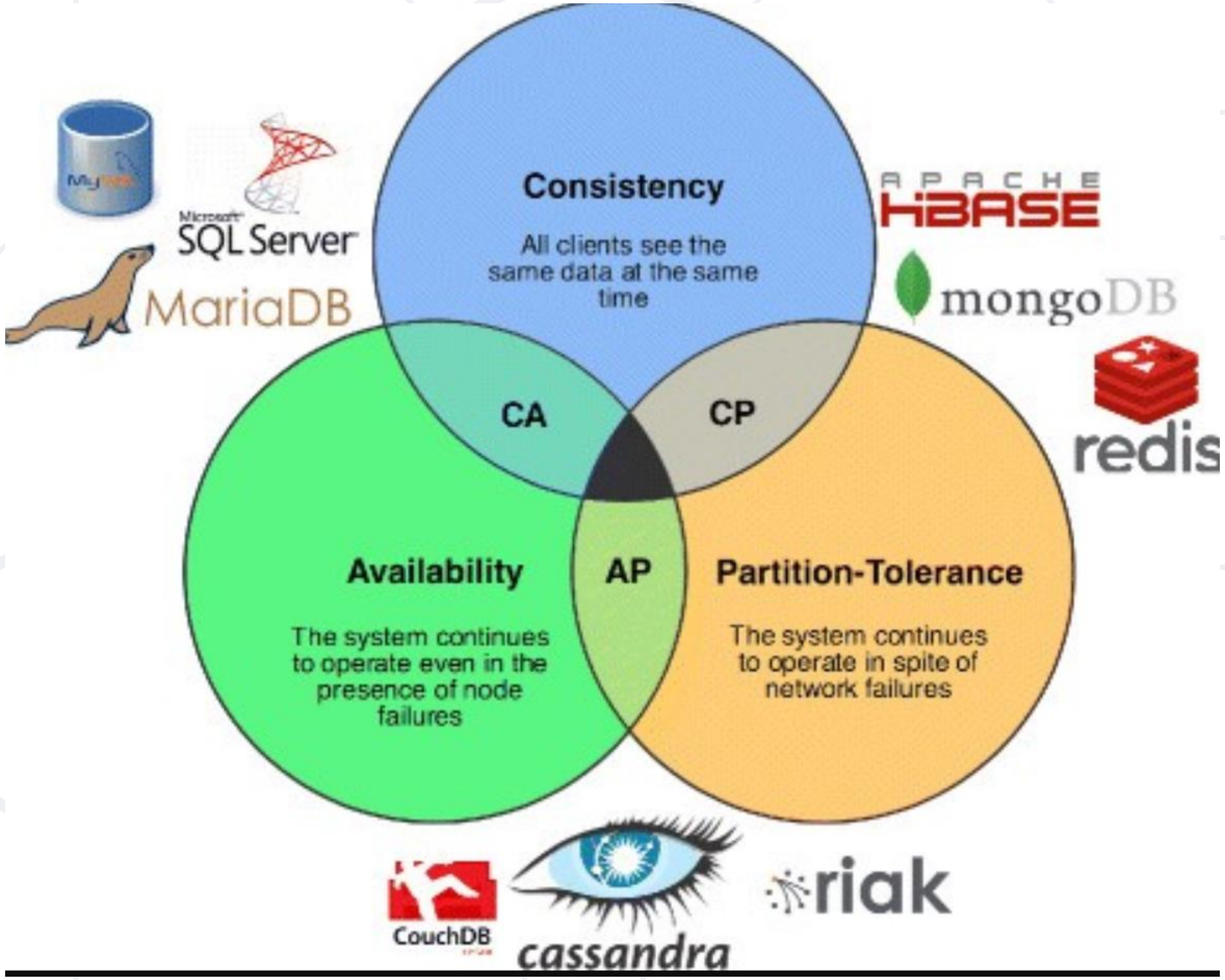
CAP Theorem

What is the CAP Theorem?

- The CAP theorem states that a distributed system can deliver **only two of three** desired characteristics
  - Consistency
  - Availability
  - Partition tolerance



CAP


Consistency
All clients see the same data at the same time
Availability
The system continues to operate even in the presence of node failures
Partition-Tolerance
The system continues to operate in spite of network failures
CA
CP
AP
MySQL
Microsoft SQL Server
MariaDB
APACHE HBASE
mongoDB
redis
CouchDB
cassandra
riak

The 'CAP' in the CAP Theorem, Explained (1)

- Consistency
  - All clients see the same data at the same time, no matter which node they connect to
    - Whenever data is written to one node, it must be instantly forwarded or replicated to all the other nodes in the system before the write is deemed 'successful'

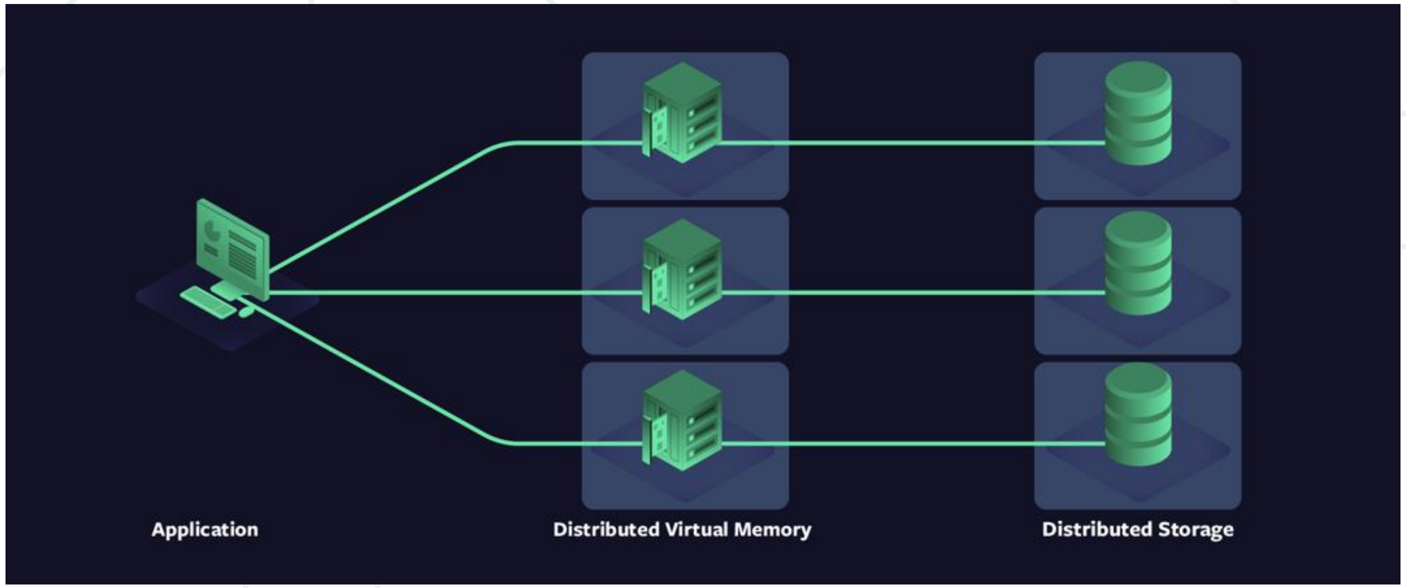
The 'CAP' in the CAP Theorem, Explained (2)

- Availability
  - Any client making a request for data gets a response, even if one or more nodes are down
- Partition tolerance
  - The cluster must continue to work despite any number of communication breakdowns between nodes in the system

Distributed Systems

Distributed Systems

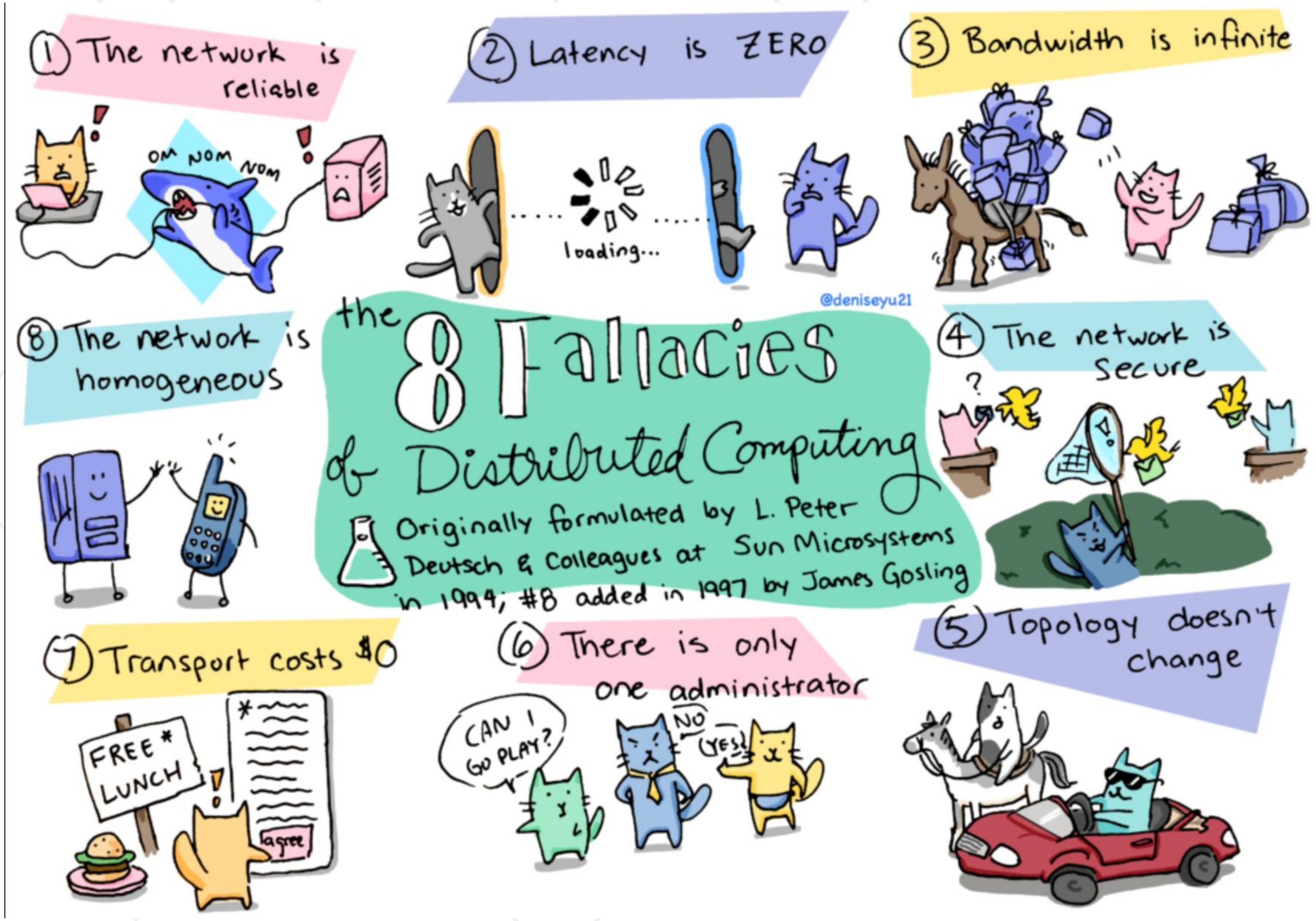
- A network that stores data on more than one physical or virtual machines at the same time
- NoSQL databases are often distributed, and the data is stored on multiple computers



8 Fallacies of Distributed Systems

- The 8 fallacies are
  - The network is reliable
  - Latency is zero
  - Bandwidth is infinite
  - The network is secure
  - Topology doesn't change
  - There is one administrator
  - Transport cost is zero
  - The network is homogeneous

Cats and Fallacies

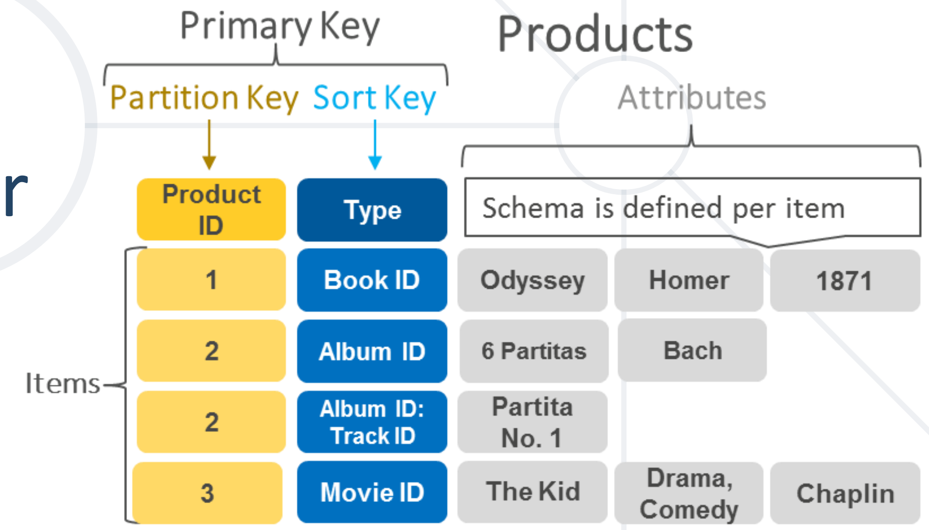


① The network is reliable
OM NOM NOM
② Latency is ZERO
loading...
③ Bandwidth is infinite
@deniseyu21
the 8 Fallacies of Distributed Computing
Originally formulated by L. Peter Deutsch & Colleagues at Sun Microsystems in 1994; #8 added in 1997 by James Gosling
⑧ The network is homogeneous
④ The network is secure
⑦ Transport costs $0
FREE* LUNCH
agree
⑥ There is only one administrator
CAN I GO PLAY?
NO
YES!
⑤ Topology doesn't change

Key-Value Databases

Key-Value Databases

- Key-value databases work by storing and managing associative arrays
  - Keys serve as a unique identifier to retrieve an associated value
  - Values can be anything from simple objects, like integers or strings, to more complex objects



Product ID	Type	Schema is defined per item		
1	Book ID	Odyssey	Homer	1871
2	Album ID	6 Partitas	Bach	
2	Album ID: Track ID	Partita No. 1		
3	Movie ID	The Kid	Drama, Comedy	Chaplin

Document-Oriented Databases

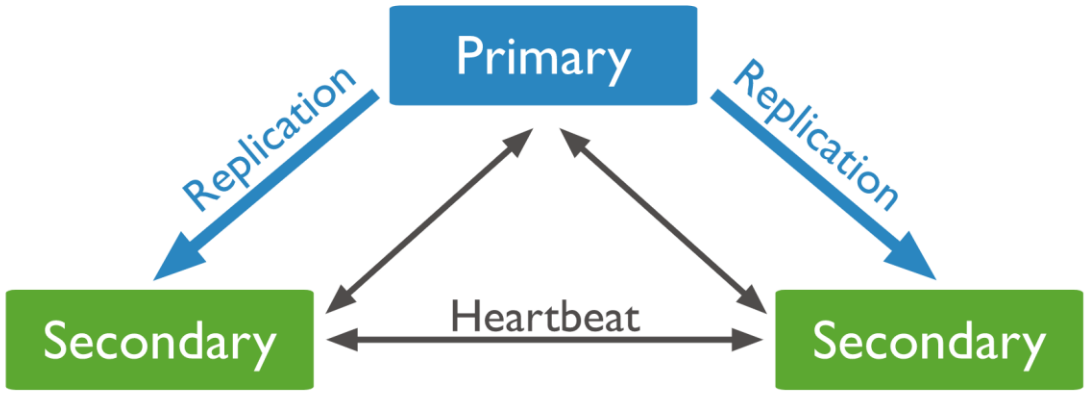
Document-Oriented Databases

- Document-oriented databases, or document stores, are NoSQL databases that store data in the form of documents
  - Document stores are a type of key-value store
    - Each document has a unique identifier
    - Document itself serves as the value
  - Usually stored as JSON, XML, Proto-Buff, etc.

```
{
    "FirstName": "Bob",
    "Address": "5 Oak St.",
    "Hobby": "sailing"
}
```

Replication

- A **replica** set is a group of mongod instances that maintain the same data set
- If the primary is unavailable, an eligible secondary will hold an election to elect itself the new primary
- All reads and writes happen from the primary (configurable)

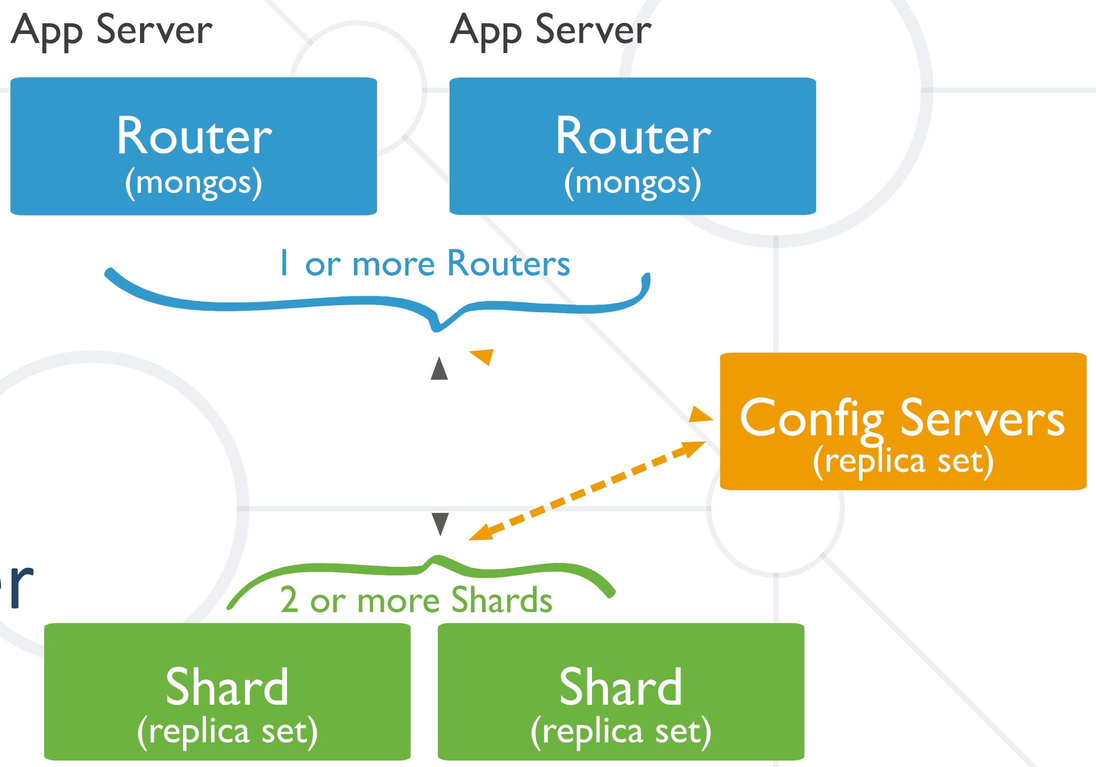


Sharding

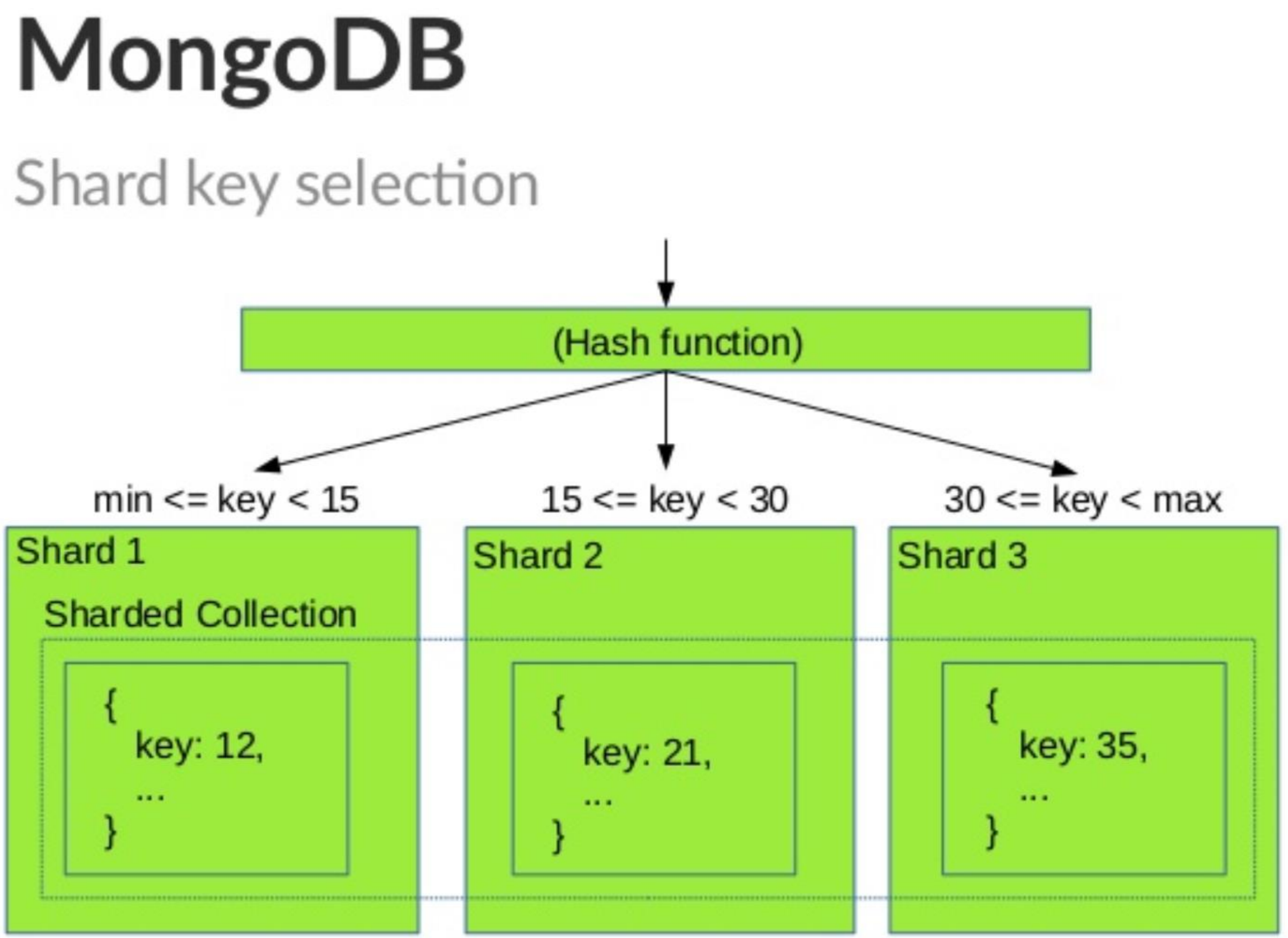
- **Sharding** is a method for distributing data across multiple machines
- MongoDB uses sharding to support deployments with very large data sets and high throughput operations
- Database systems with large data sets or high throughput applications can challenge the capacity of a single server

Sharded Cluster

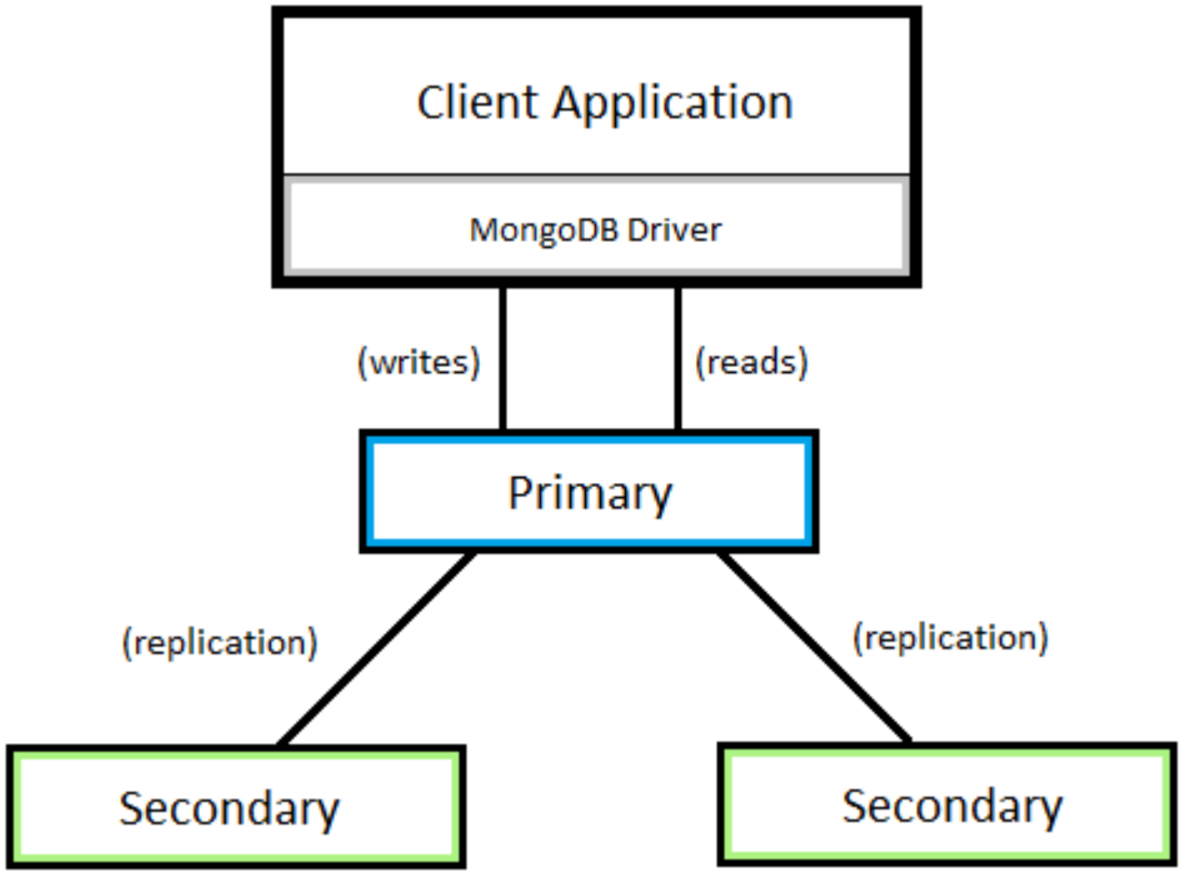
- A MongoDB **sharded cluster** consists of the following components
  - **Shard** – each contains a subset of the sharded data
  - **Mongos** - a query router, providing an interface between client applications and the sharded cluster
  - **Config Servers** - store metadata and configuration settings for the cluster



Sharding


MongoDB
Shard key selection
(Hash function)
min <= key < 15
15 <= key < 30
30 <= key < max
Shard 1
Sharded Collection
{ key: 12, ... }
Shard 2
{ key: 21, ... }
Shard 3
{ key: 35, ... }

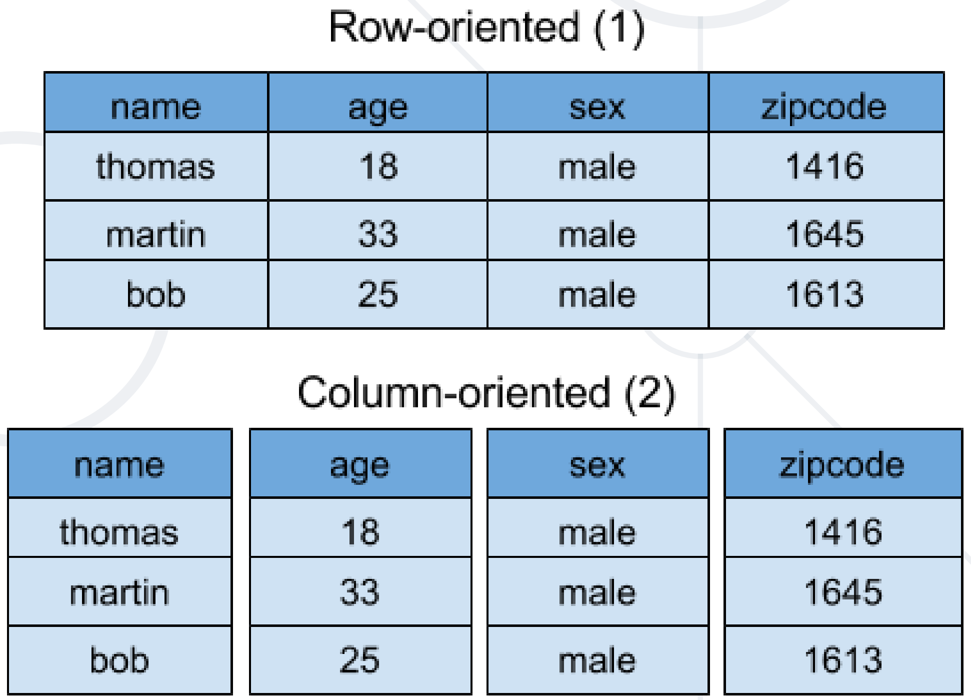
Replication



Columnar Databases

Columnar Databases (1)

- Columnar databases, are database systems that store data in columns
  - Each column is stored in a separate file or region in the system's storage
  - Examples of columnar databases are Cassandra, Hbase, Redshift, etc.



Row-oriented (1)

name	age	sex	zipcode
thomas	18	male	1416
martin	33	male	1645
bob	25	male	1613

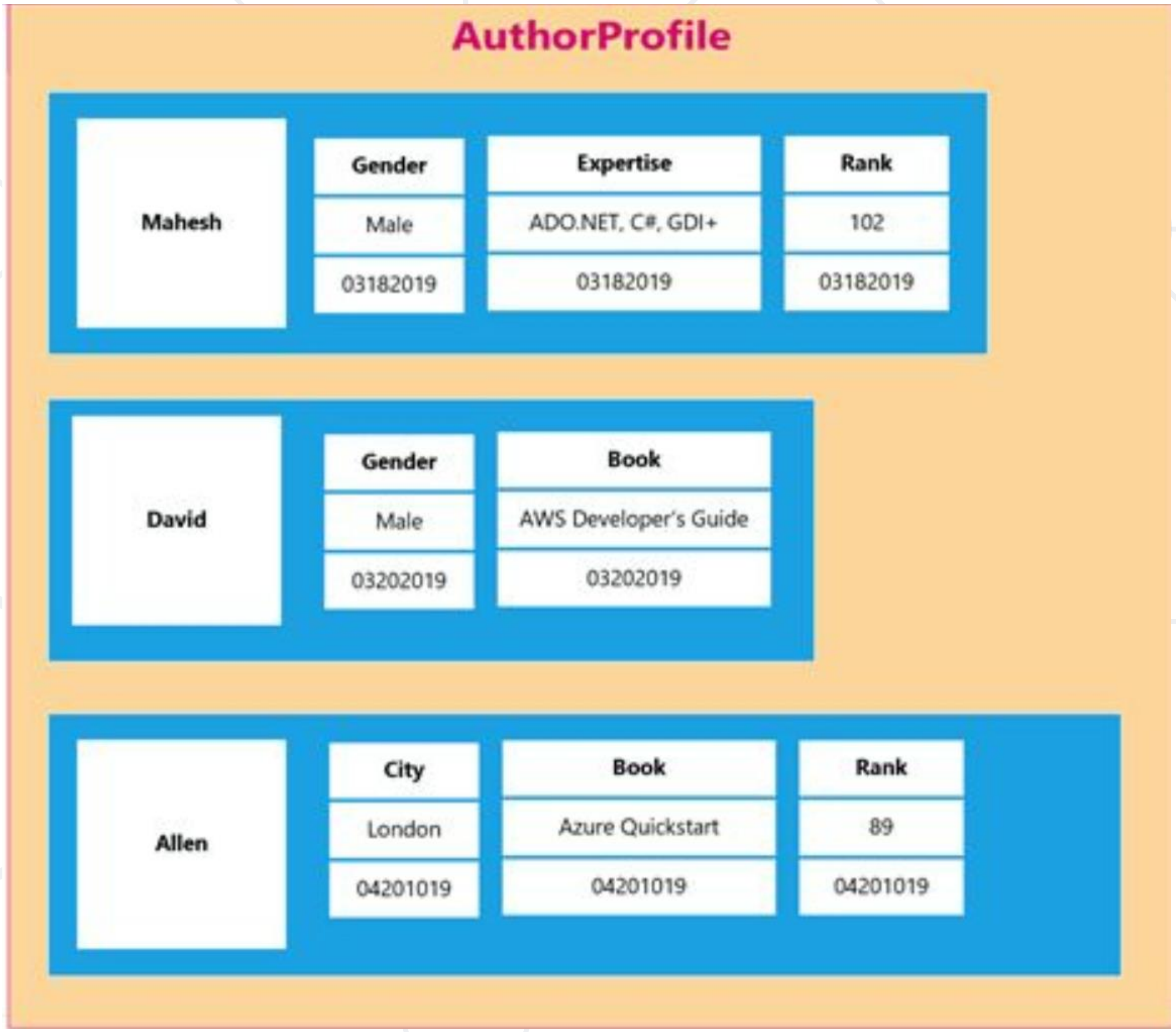
Column-oriented (2)

name	age	sex	zipcode
thomas	18	male	1416
martin	33	male	1645
bob	25	male	1613

Columnar Databases (2)

- Key benefits of column store databases include faster performance in load, search, and aggregate functions
- Column-oriented organizations are more efficient when an aggregate needs to be computed over many rows but only for a notably smaller subset of all columns of data
- Not efficient when many columns of a row are required at the same time

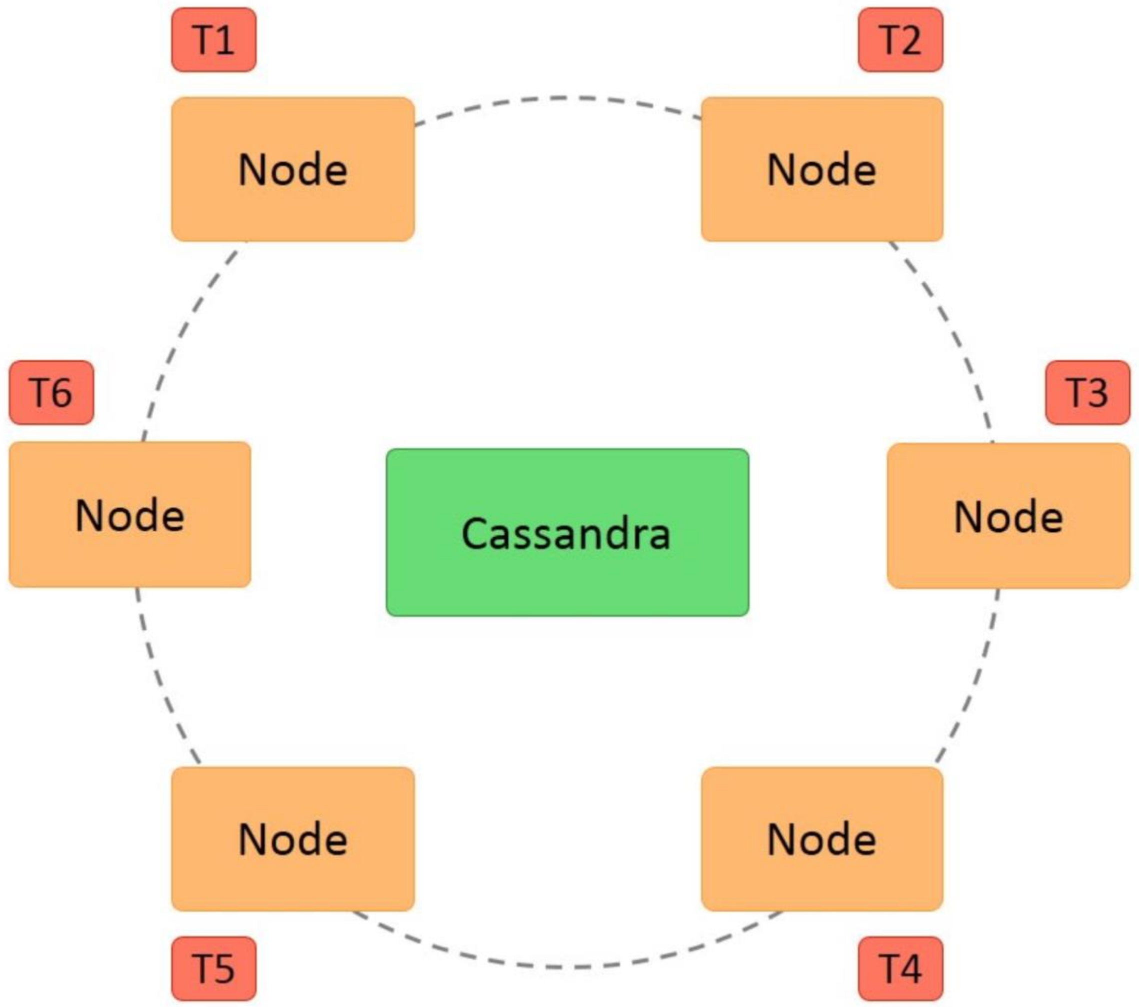
Columnar Databases (3)



Cassandra

- Generally considered AP (in CAP)
- Every node in the cluster has the same role. There is no single point of failure.
- Data is distributed across the cluster (so each node contains different data)
- Failed nodes can be replaced with no downtime.
- Eventually consistent (configurable)
- Uses CQL for queries

Cassandra (2)



Cassandra Usage

- **Discord** switched to Cassandra to store billions of messages from MongoDB in November, 2015
- **Netflix** uses Cassandra as their back-end database for their streaming services
- **Apple** uses 100,000 Cassandra nodes
- **Uber** uses Cassandra to store around 10,000 features
- Many more applications

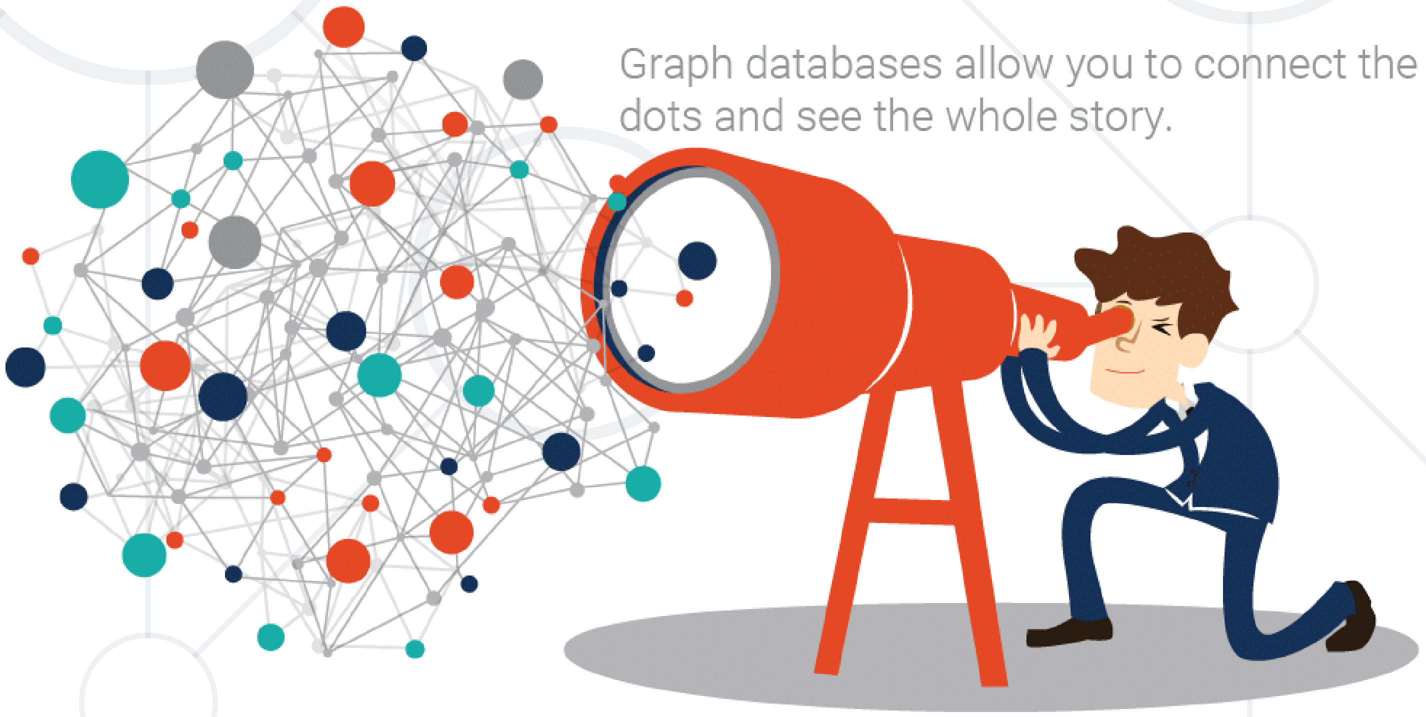
Graph Database

What is a Graph Database? (1)

- Allow simple and fast retrieval of complex hierarchical structures that are difficult to model in relational systems
- No universal query language is present for graph databases (like SQL). Each database has own implementation of queries

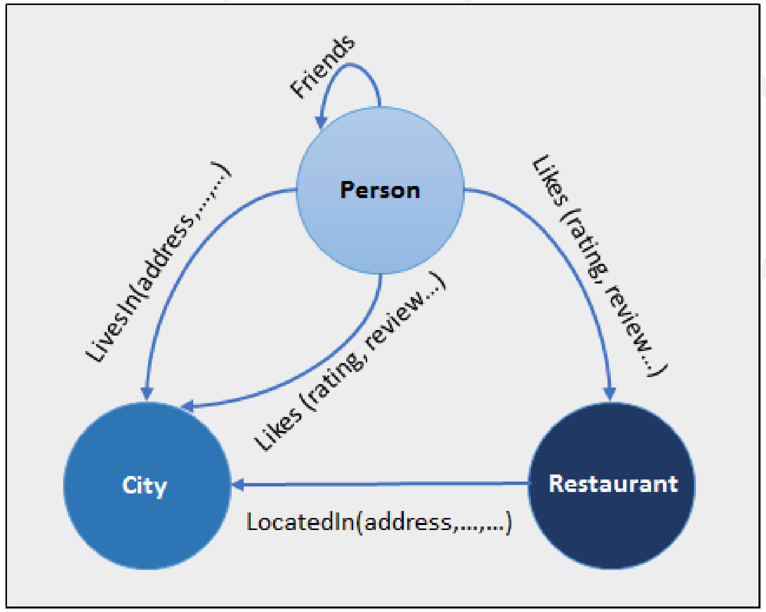
What is a Graph Database? (2)

- A graph database contains a collection of nodes and edges
  - A node represents an object
  - An edge represents the connection or relationship between two objects


Graph databases allow you to connect the dots and see the whole story.

What is a Graph Database? (3)

- Each node is identified by a unique identifier that expresses key-value pairs
- Each edge is defined by a unique identifier that details a starting or ending node, along with a set of properties

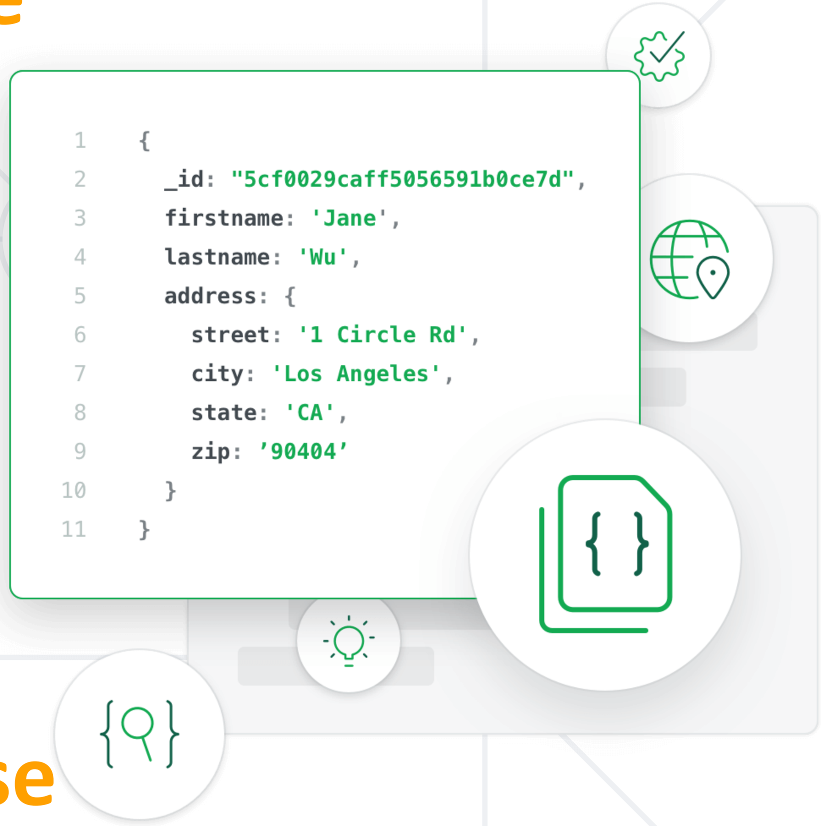

Friends
Person
LivesIn(address,...,...)
Likes (rating, review...)
Likes (rating, review...)
City
Restaurant
LocatedIn(address,...,...)

MongoDB Overview

Installation, Configuration, Startup

What is MongoDB?

- MongoDB is a **document database**
- It stores data in flexible, **BSON** documents
- The document model maps to the objects in the application code, making data easy to work with
- MongoDB is a **distributed database** at its core



```
{
  _id: "5cf0029caff5056591b0ce7d",
  firstname: 'Jane',
  lastname: 'Wu',
  address: {
    street: '1 Circle Rd',
    city: 'Los Angeles',
    state: 'CA',
    zip: '90404'
  }
}
```

Install MongoDB

- Download from: https://www.mongodb.com/download-center
- When **installed**, MongoDB needs a **driver**
  - One to use with Node.js, .NET, Java, etc..
  - MongoDB C#/.NET driver: https://docs.mongodb.com/drivers/csharp

Working with MongoDB GUI

- Choose one of the many
- For example
  - Robo 3T → https://robomongo.org/download
  - NoSQLBooster → https://nosqlbooster.com
  - Compass → https://www.mongodb.com/products/compass

Working with MongoDB Shell Client

- Start the shell from a **CLI**
  - Type the command **mongo**

```
show dbs
```

Shows **all** databases in the data **folder**

```
use mytestdb
```

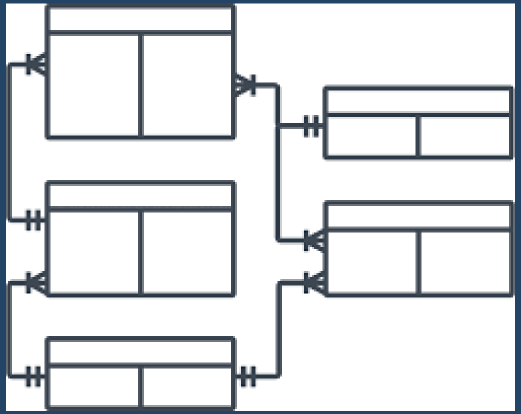
```
db.mycollection.insert({"name":"George"})
```

```
db.mycollection.find({"name":" George"})
```

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

Gets **all** entries in the database

  - Additional information at https://docs.mongodb.com/manual/reference/mongo-shell/



CRUD Operations

Connect to MongoDB

- To **connect** to a MongoDB cluster, use the connection string for your cluster

```
using MongoDB.Bson;
using MongoDB.Driver;
…
var client = new MongoClient(
"mongodb+srv://<username>:<password>@<cluster-address>/test?w=majority"
);
var database = client.GetDatabase("Example");
var collection = database.GetCollection<Interactions>("Interactions");
```

Select

- To **select** a document use Linq

```
var result = IMongoCollectionExtensions
             .AsQueryable(collection)
             .FirstOrDefault(s => s.SiteName == "Example");
```

Update

- **FindOneAndUpdate()**

```
var update = MongoDB.Driver.Builders.Update.Set(s => s.SiteName,
       "New Example");

collection.FindOneAndUpdate(s => s.SiteName == "Example",
update);
```

Delete and Insert

- `DeleteOne()`
  - Deletes the first document that meets the filter

```
collection.DeleteOne(e => e.Name =="Example");
```

- `InsertOne()`
  - Inserts a new document

```
collection.InsertOne(newItem);
```

Summary

- **NoSQL**
- Ability of a system's DB to **scale up** or **down**
- **CAP** Theorem
- **Distributed** Systems
- **Key-value**, **Document-oriented**, **Columnar** and **Graph** DBs
- **MongoDB** Overview
- **CRUD** Operations

Questions?

Software University

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

SUPER HOSTING .BG

Coca-Cola HBC Bulgaria

Flutter™ International

INDEAVR Serving the high achievers

AMBITIONED

DRAFT KINGS

SOFTWARE GROUP

BOSCH

Postbank Решения за твоето утре

PHAR VISION

SmartIT

DXC TECHNOLOGY

createX

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License