UNIVERSITAT DE BARCELONA

Master in Fundamental Principles of Data Science

Dr Rohit Kumar

Today’s Objective

- Introduction to NoSQL
- Introduction to MongoDB

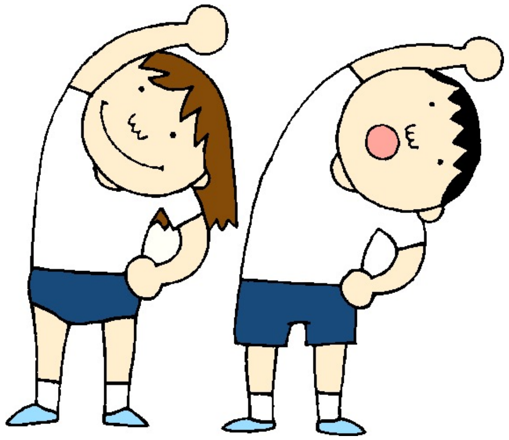
UNIVERSITAT DE BARCELONA

NoSQL

What is NoSQL

Scalable

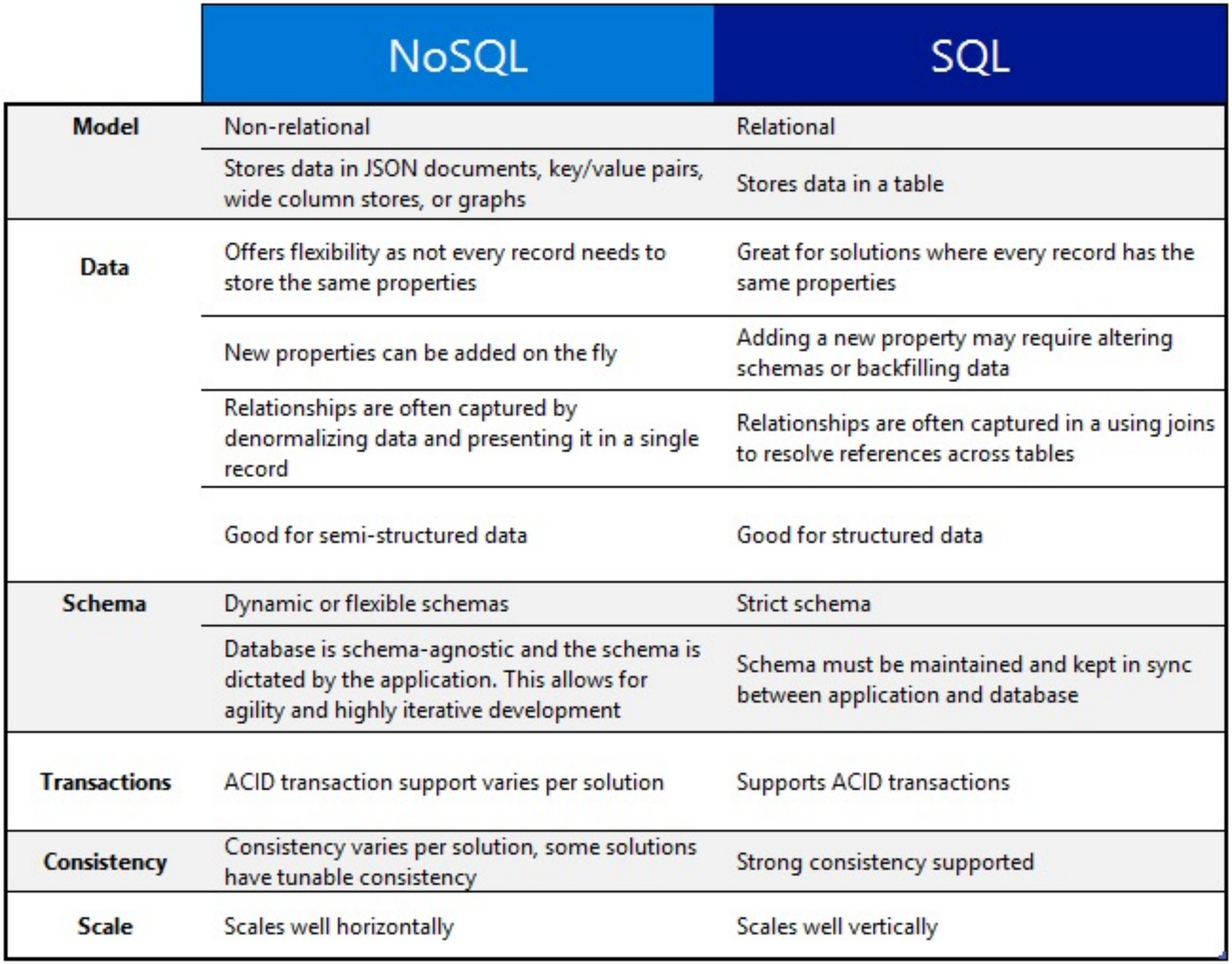
Fast



Flexible and Easy

SQL to NoSQL

NoSQL => “Not only SQL.”



	NoSQL	SQL
Model	Non-relational	Relational
	Stores data in JSON documents, key/value pairs, wide column stores, or graphs	Stores data in a table
Data	Offers flexibility as not every record needs to store the same properties	Great for solutions where every record has the same properties
	New properties can be added on the fly	Adding a new property may require altering schemas or backfilling data
	Relationships are often captured by denormalizing data and presenting it in a single record	Relationships are often captured in a using joins to resolve references across tables
	Good for semi-structured data	Good for structured data
Schema	Dynamic or flexible schemas	Strict schema
	Database is schema-agnostic and the schema is dictated by the application. This allows for agility and highly iterative development	Schema must be maintained and kept in sync between application and database
Transactions	ACID transaction support varies per solution	Supports ACID transactions
Consistency	Consistency varies per solution, some solutions have tunable consistency	Strong consistency supported
Scale	Scales well horizontally	Scales well vertically

Source: https://olive4oyl.wordpress.com/2016/08/31/sql-vs-nosql/

Advantages of NoSQL

- Can be used as Primary or Analytic Data Source
- Big Data Capability
- No Single Point of Failure
- Easy Replication
- It provides fast performance and horizontal scalability.
- Can handle structured, semi-structured, and unstructured data with equal effect

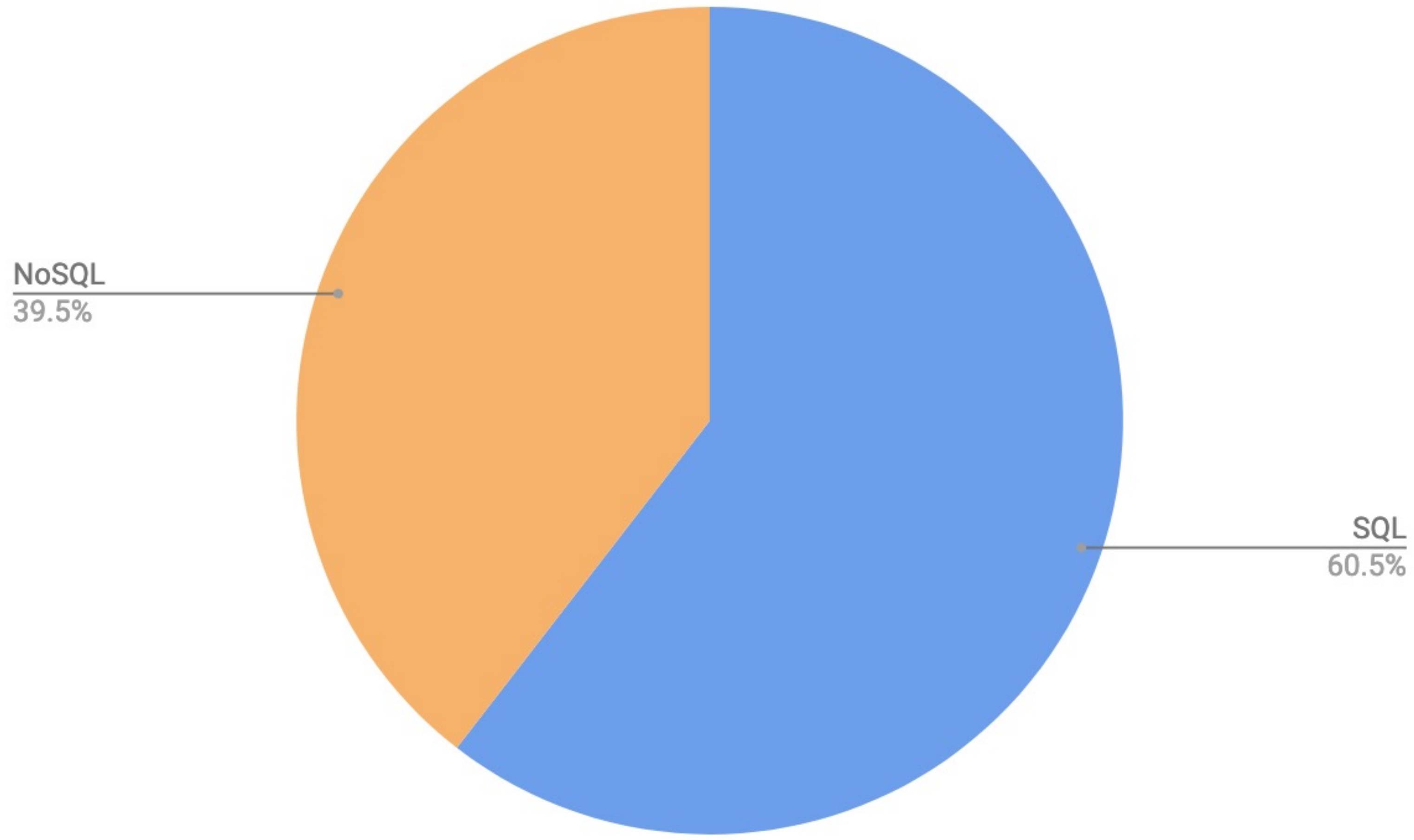
Advantages of NoSQL

- NoSQL databases don't need a dedicated high-performance server
- Handles big data which manages data velocity, variety, volume, and complexity
- Excels at distributed database and multi-data center operations
- Offers a flexible schema design which can easily be altered without downtime or service disruption

Disadvantages of NoSQL

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- Doesn't work as well with relational data
- The learning curve is stiff for new developers

SQL vs NoSQL



Src: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/

UNIVERSITAT DE BARCELONA

NoSQL

Key Value

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

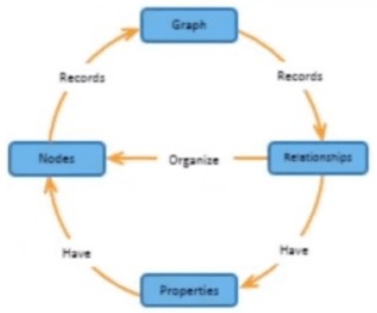
Document-Based

Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based

Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based



Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Key Value

This key/value type database allow clients to read and write values using a key as follows:

- Get(key), returns the value associated with the provided key.
- Put(key, value), associates a value with the key.
- Multi-get(key1, key2, .., keyN), returns the list of values associated with the list of keys.
- Delete(key), removes the entry for the key from the data store.

Key	Value
“India”	{“B-25, Sector-58, Noida, India – 201301”}
“Romania”	{“IMPS Moara Business Center, Buftea No. 1, Cluj-Napoca, 400606″,City Business Center, Coriolan Brediceanu No. 10, Building B, Timisoara, 300011”}
“US”	{“3975 Fair Ridge Drive. Suite 200 South, Fairfax, VA 22033”}

Use cases

During the holiday shopping season, an e-commerce website may receive billions of orders in seconds. Key-value databases can handle the scaling of large amounts of data and extremely high volumes of state changes while servicing millions of simultaneous users through distributed processing and storage.

Key-value databases also have built-in redundancy, which can handle the loss of storage nodes.

Document-Based

A document database is a type of nonrelational database that is designed to store and query data as JSON-like documents.

Document databases enable flexible indexing, powerful ad hoc queries, and analytics over collections of documents.

{country:"India", {Street: "B-25, City:"Noida", State:"UP", Pincode:"201301"} }

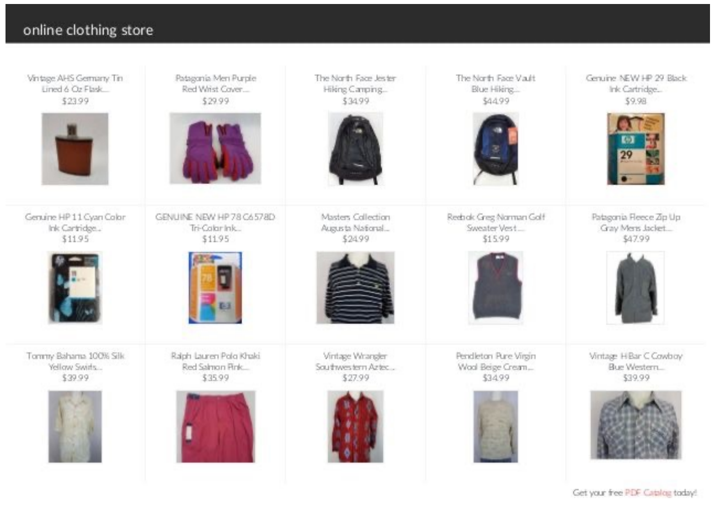
{country :"Romania", {Boulevard:"Coriolan Brediceanu No. 10", Block:"B, Ist Floor", City: "Timisoara", Pincode: 300011"} }

{country :"US", {Latitude:"40.748328", Longitude:"-73.985560"} }

Use cases

A document database is a great choice for content management applications such as blogs and video platforms.

Use cases


online clothing store
Vintage AHS Germany Tin Lined 6 Oz Flask... $23.99
Patagonia Men Purple Red Wrist Cover... $29.99
The North Face Jester Hiking Camping... $34.99
The North Face Vault Blue Hiking... $44.99
Genuine NEW HP 29 Black Ink Cartridge... $9.98
Genuine HP 11 Cyan Color Ink Cartridge... $11.95
GENUINE NEW HP 78 C6578D Tri-Color Ink... $11.95
Masters Collection Augusta National... $24.99
Reebok Greg Norman Golf Sweater Vest... $15.99
Patagonia Fleece Zip Up Gray Mens Jacket... $47.99
Tommy Bahama 100% Silk Yellow Swirls... $39.99
Ralph Lauren Polo Khaki Red Salmon Pink... $35.99
Vintage Wrangler Southwestern Aztec... $27.99
Pendleton Pure Virgin Wool Beige Cream... $34.99
Vintage H Bar C Cowboy Blue Western... $39.99
Get your free PDF Catalog today!

Document databases are efficient and effective for storing catalog information.

Search Based Engines (kind of document store)

- A search-engine database is a type of nonrelational database that is dedicated to the search of data content.
- Search-engine databases use indexes to categorize the similar characteristics among data and facilitate search capability.
- Example: Elastic Search or Splunk

Use Cases

Text search

Search-engine databases can handle full-text search faster than relational databases.

Logging and analysis

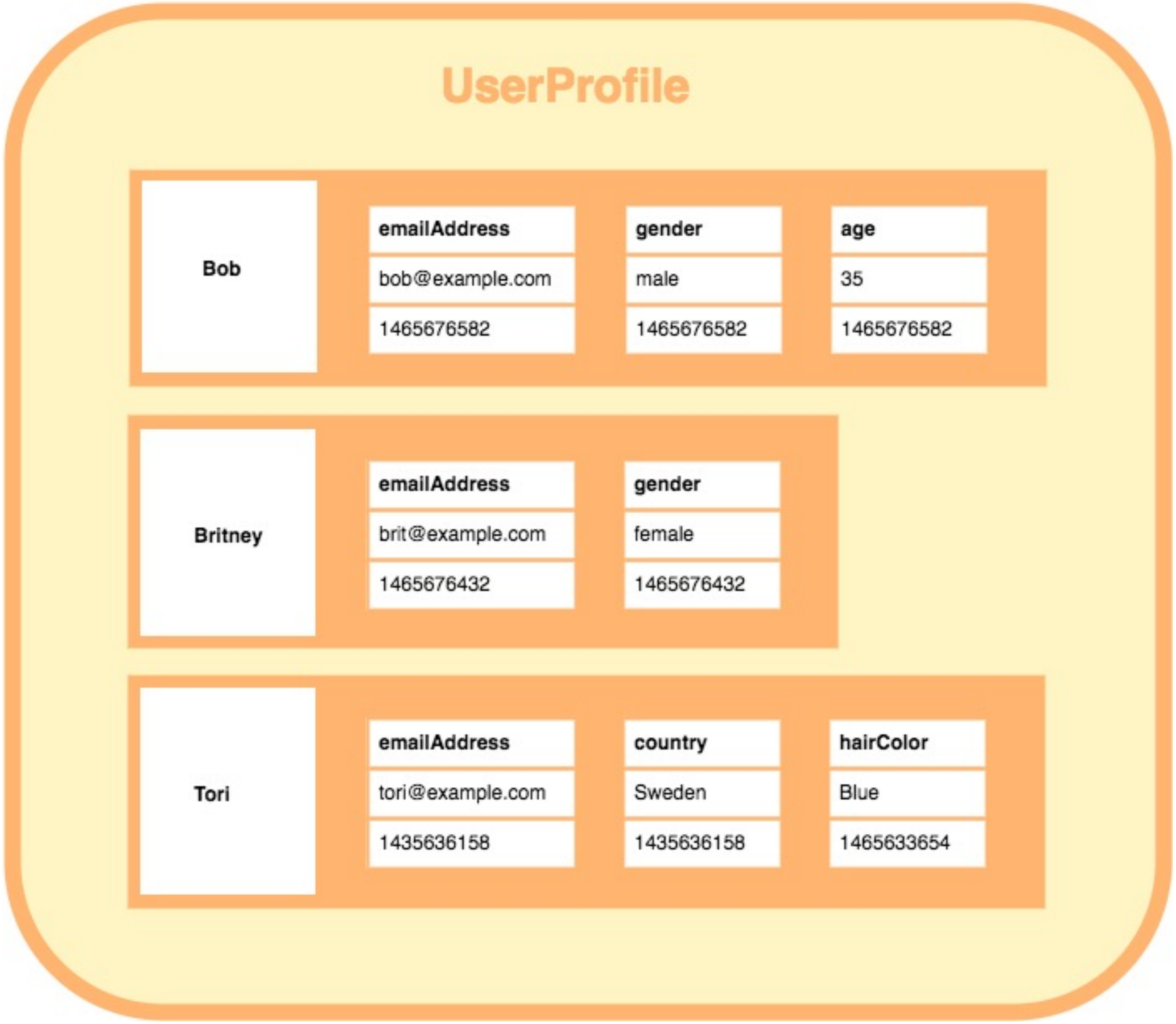
Maintaining larger applications that are either distributed across several nodes or consist of several smaller applications searching for events in log files can become tedious.

Column-Based

Column-Based
Example:
BigTable, Cassandra,
Hbase,
Hypertable

- A column store database is a type of database that stores data using a column oriented model.
- Columns store databases use a concept called a keyspace. A keyspace is kind of like a schema in the relational model. The keyspace contains all the column families (kind of like tables in the relational model), which contain rows, which contain columns.

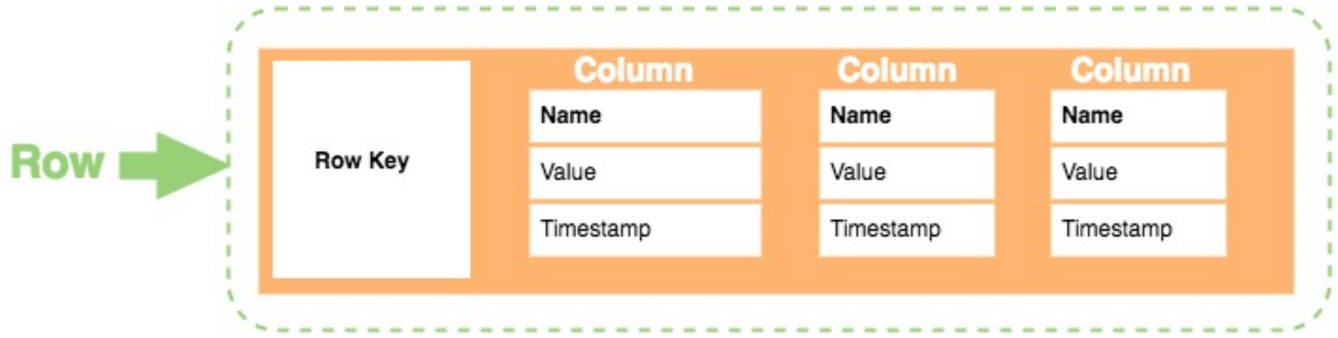
Column-Based



Column-Based

- A **column family** consists of multiple rows.
- Each **row** can contain a different number of columns to the other rows. And the columns don’t have to match the columns in the other rows (i.e. they can have different column names, data types, etc).
- Each **column** is contained to its row. It doesn’t span all rows like in a relational database. Each column contains a name/value pair, along with a timestamp. Note that this example uses Unix/Epoch time for the timestamp.

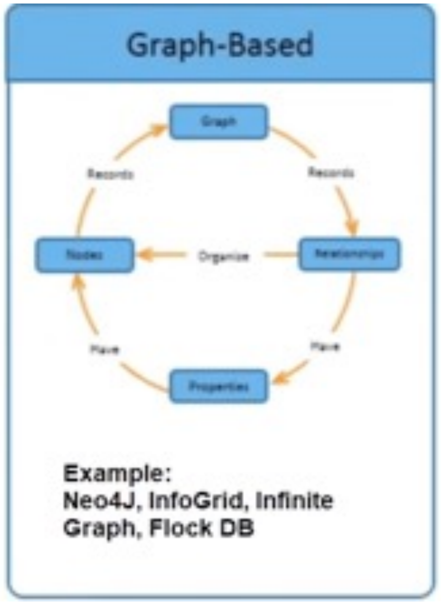
Column-Based



- **Row Key**. Each row has a unique key, which is a unique identifier for that row.
- **Column**. Each column contains a name, a value, and timestamp.
- **Name**. This is the name of the name/value pair.
- **Value**. This is the value of the name/value pair.
- **Timestamp**. This provides the date and time that the data was inserted. This can be used to determine the most recent version of data.

UNIVERSITAT DE BARCELONA

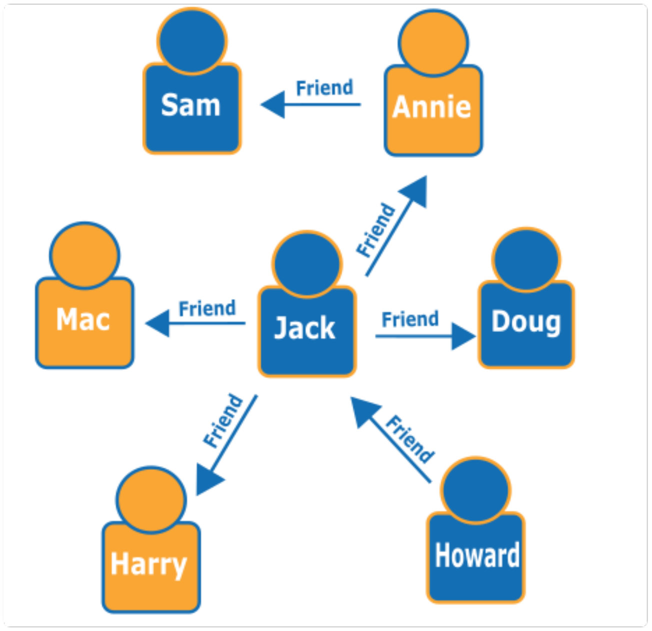
Graph Based

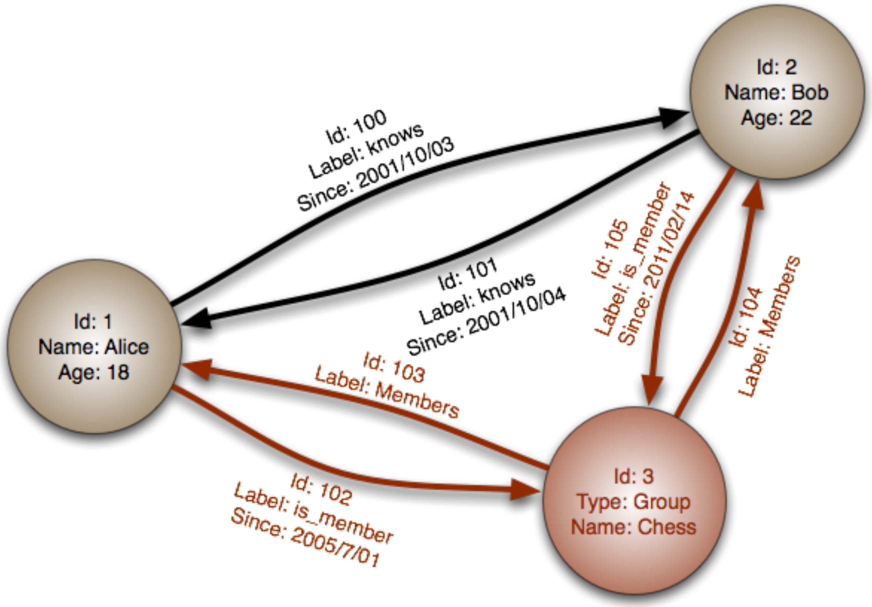


- Graph databases are purpose-built to store and navigate relationships.
- These databases uses edges and nodes to represent and store data.
- Nodes are organised by some relationships with one another, which is represented by edges between the nodes.
- Both the nodes and the relationships have some defined properties.

UNIVERSITAT DE BARCELONA

Graph Based





Use cases

Graph databases are capable of sophisticated **fraud prevention**.

UNIVERSITAT DE BARCELONA

Use cases

Graph databases are a good choice for recommendation applications.

UNIVERSITAT DE BARCELONA

Types of NoSQL

Types	Performance	Scalability	Flexibility	Complexity
Key-Value	high	high	high	None
Column Store	high	high	Moderate	Low
Document Store	high	Variable (high)	high	Low
Graph Database	Variable	Variable	high	high

MongoDB

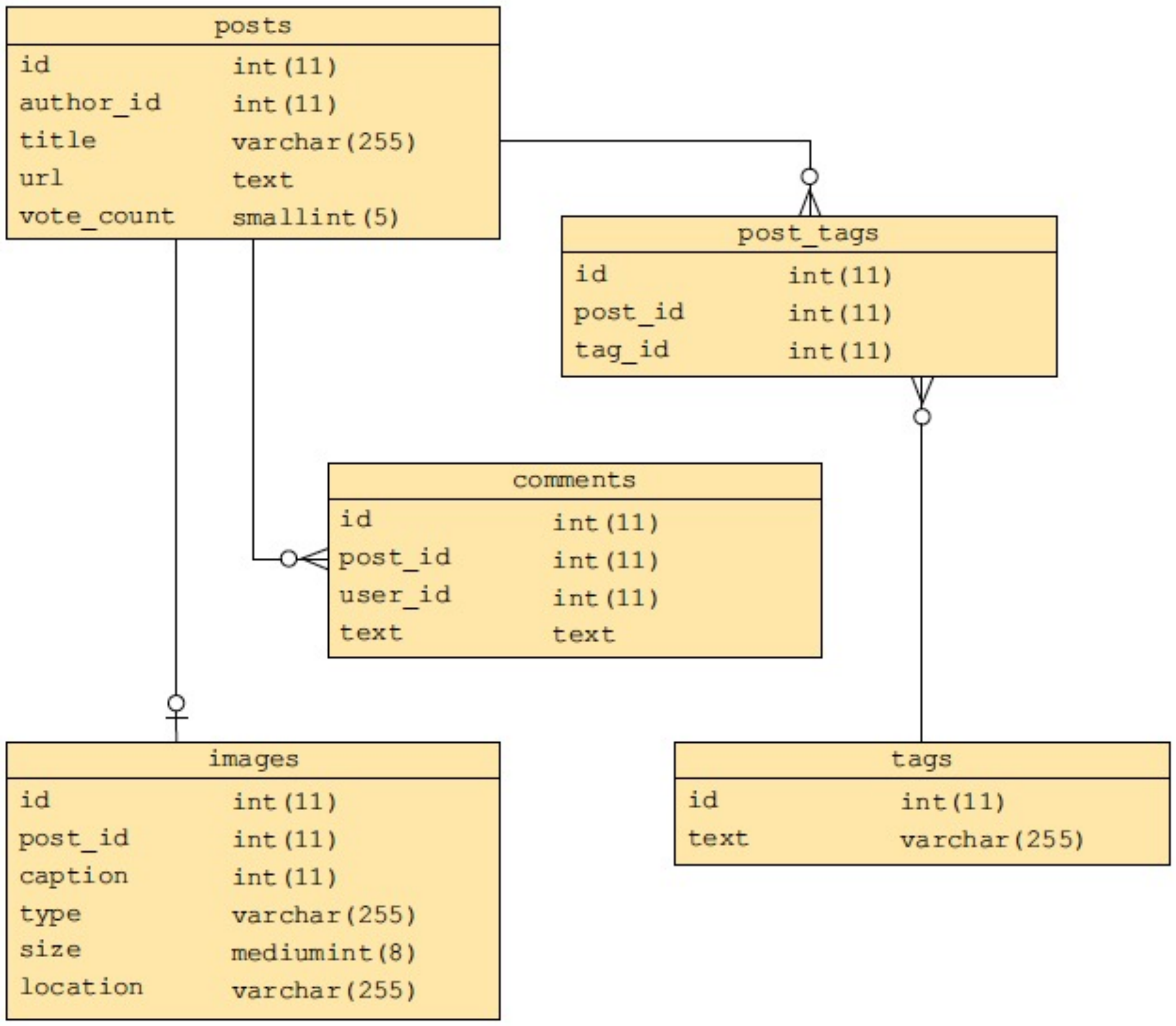
Document Model

Lets take an example of a post/blog on a social media platform like redit or twitter.

Every post contains following elements:

1) Title of the post
2) Tags assosicated with the post
3) Comments made
4) Votes
5) Author
6) URL of the post
7) Images or video in the post

In Relational Model



posts	
id	int(11)
author_id	int(11)
title	varchar(255)
url	text
vote_count	smallint(5)

post_tags	
id	int(11)
post_id	int(11)
tag_id	int(11)

comments	
id	int(11)
post_id	int(11)
user_id	int(11)
text	text

images	
id	int(11)
post_id	int(11)
caption	int(11)
type	varchar(255)
size	mediumint(8)
location	varchar(255)

tags	
id	int(11)
text	varchar(255)

In Document Model

```
{
  _id:
ObjectID('4bd9e8e17cefd644108961bb'),
  title: String,
  url: String,
  author: String,
  vote_count: Integer,
  tags: Array[
    String
  ],
  image: ImageObject,
  comments: Array[
    CommentObject
  ]
}
```

ImageObject:

```
{
url: String,
caption: String,
type: String,
size: Integer,
data: String
}
```

CommentObject:

```
{
user:String,
text:String
}
```

In Document Model

```
{
  _id: ObjectID('4bd9e8e17cefd644108961bb'),
  title: 'Adventures in Databases',
  url: 'http://example.com/databases.txt',
  author: 'msmith',
  vote_count: 20,
  tags: [
    'databases',
    'mongodb',
    'indexing'
  ],
  image: {
    url: 'http://example.com/db.jpg',
    caption: 'A database.',
    type: 'jpg',
    size: 75381,
    data: 'Binary'
  },
  comments: [
    {
      user: 'bjones',
      text: 'Interesting article.'
    },
    {
      user: 'sverch',
      text: 'Color me skeptical!'
    }
  ]
}
```

Advantage

Suppose you want to find all posts tagged with the term politics having more than 10 votes.

```
SELECT * FROM posts
INNER JOIN posts_tags ON posts.id = posts_tags.post_id
INNER JOIN tags ON posts_tags.tag_id == tags.id
WHERE tags.text = 'politics' AND posts.vote_count > 10;
```

In MongoDB

```
db.posts.find({'tags': 'politics', 'vote_count': {'$gt': 10}});
```

UNIVERSITAT DE BARCELONA

SQL to Mongo

SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
table	collection
row	document
column	field
index	index
table joins	$lookup, embedded documents
primary key Specify any unique column or column combination as primary key.	primary key In MongoDB, the primary key is automatically set to the __id field.

SQL to Mongo

SQL Terms, Functions, and Concepts	MongoDB Aggregation Operators
WHERE	$match
GROUP BY	$group
HAVING	$match
SELECT	$project
ORDER BY	$sort
LIMIT	$limit
SUM()	$sum
COUNT()	$sum $sortByCount
join	$lookup
SELECT INTO NEW_TABLE	$out

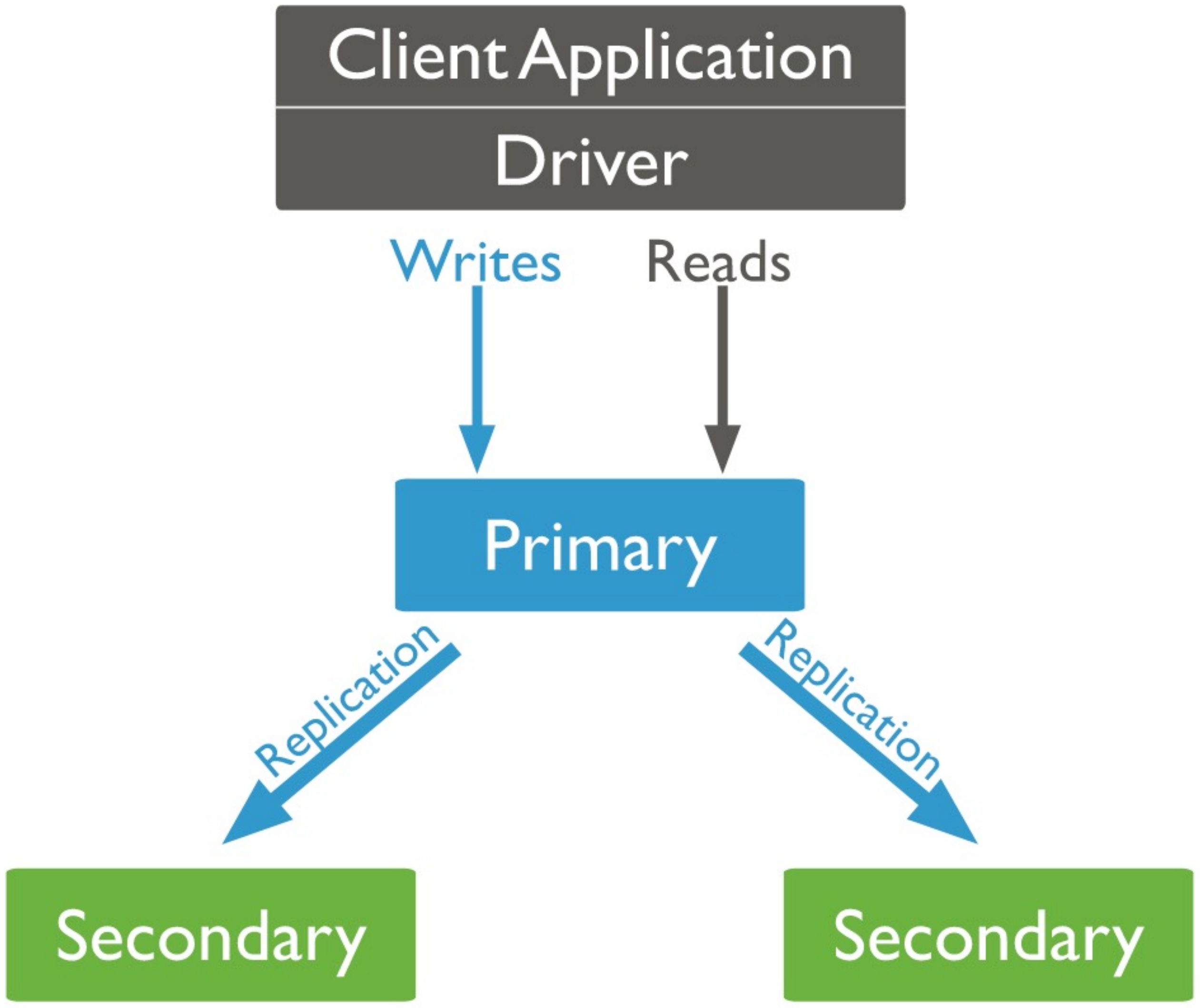
High Availability in Mongo

MongoDB’s replication facility, called replica set, provides:

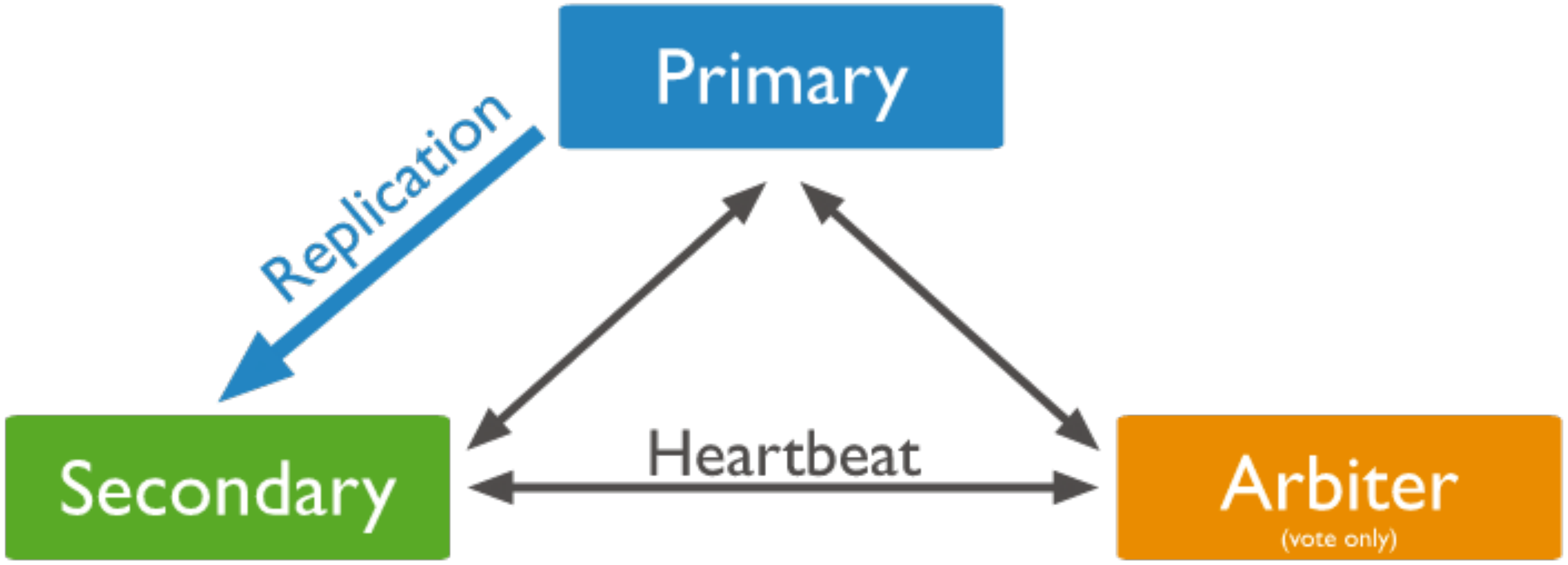
- automatic failover
- data redundancy.

A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and increasing data availability.

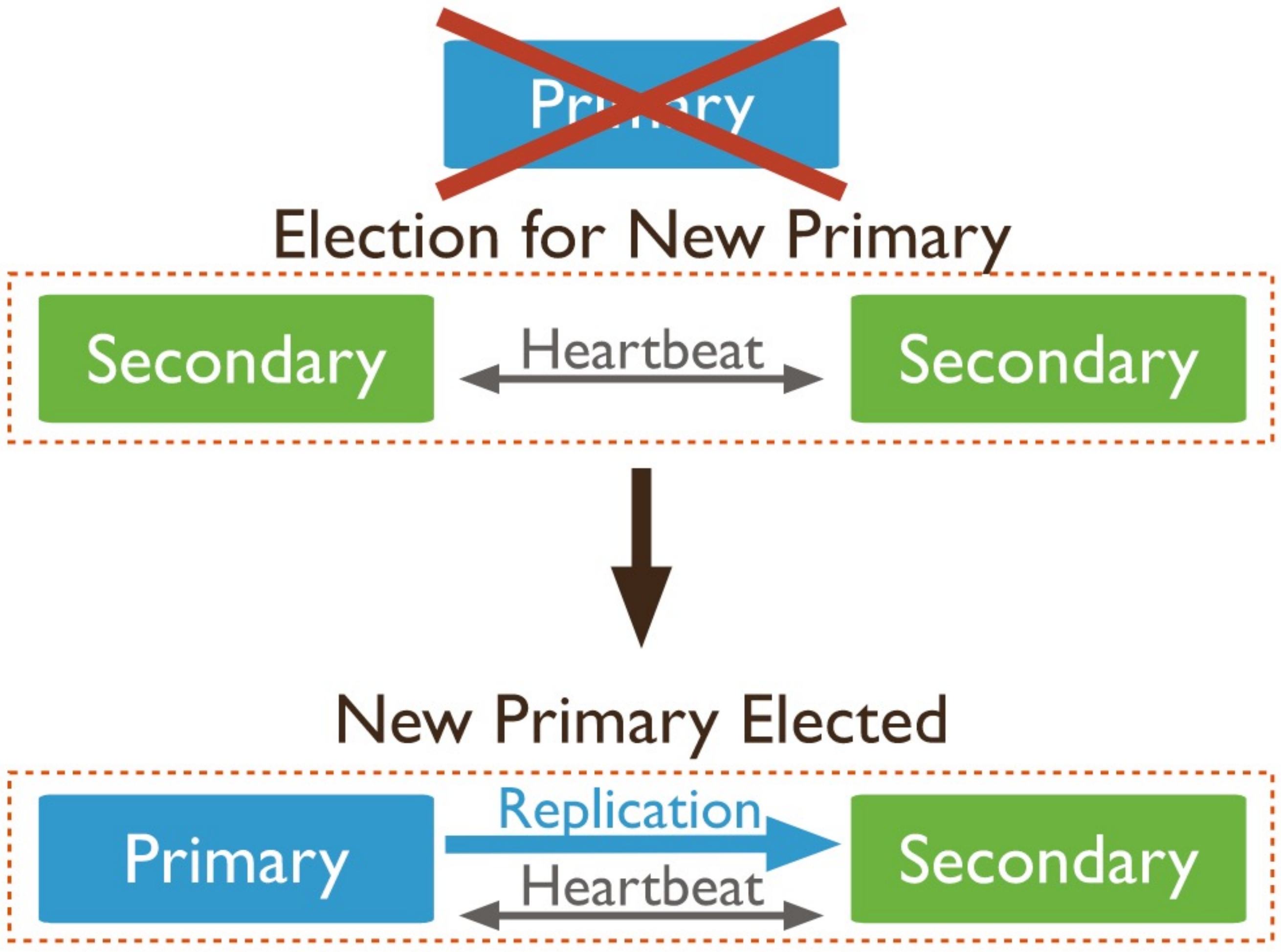
Replication



Replication



Automatic Failover



Why Replication?

- To keep your data safe
- High (24*7) availability of data
- Disaster recovery
- No downtime for maintenance (like backups, index rebuilds, compaction)
- Read scaling (extra copies to read from)
- Replica set is transparent to the application

UNIVERSITAT DE BARCELONA

What about Scalability?

Sharding

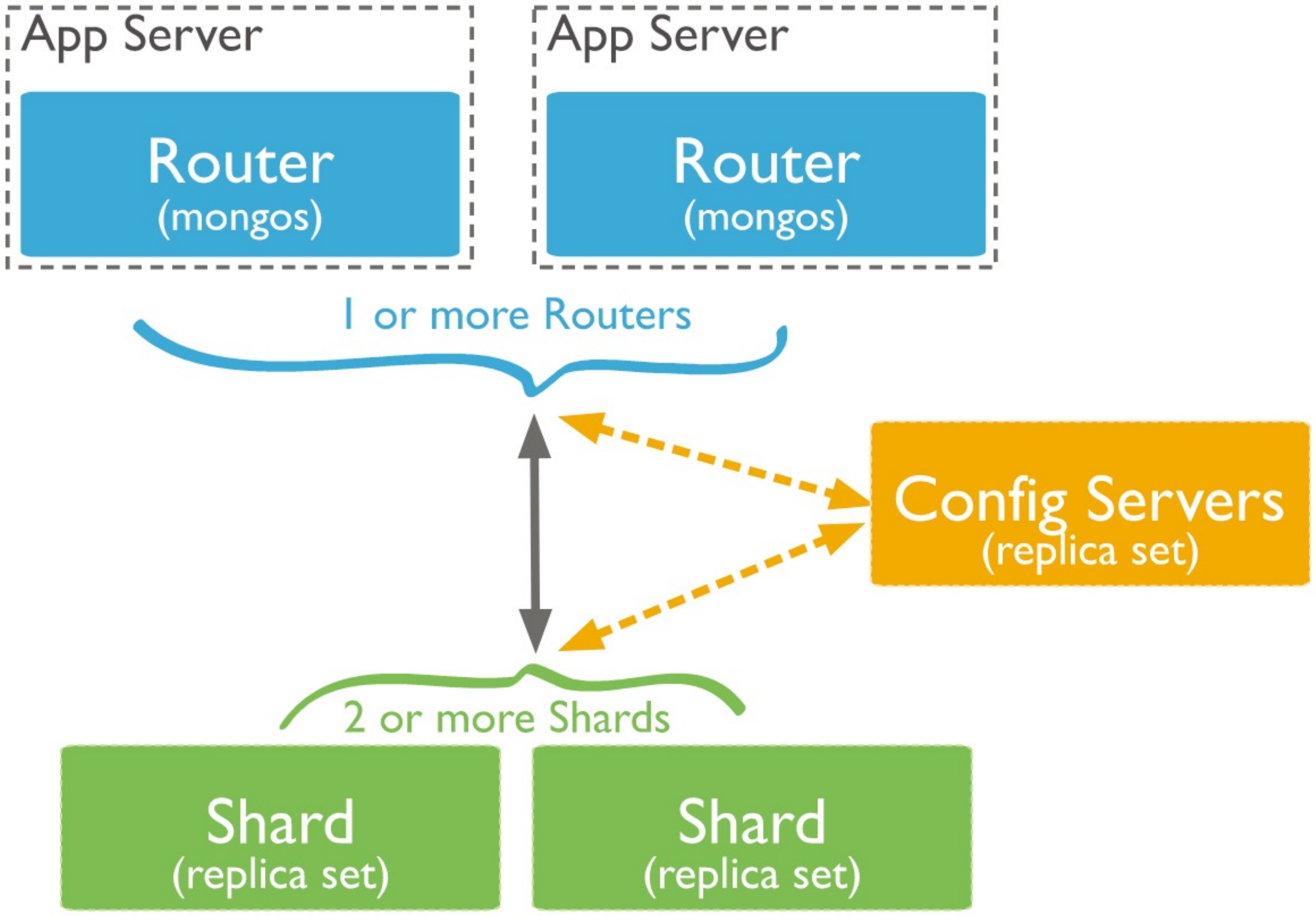
Sharding is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

Sharding

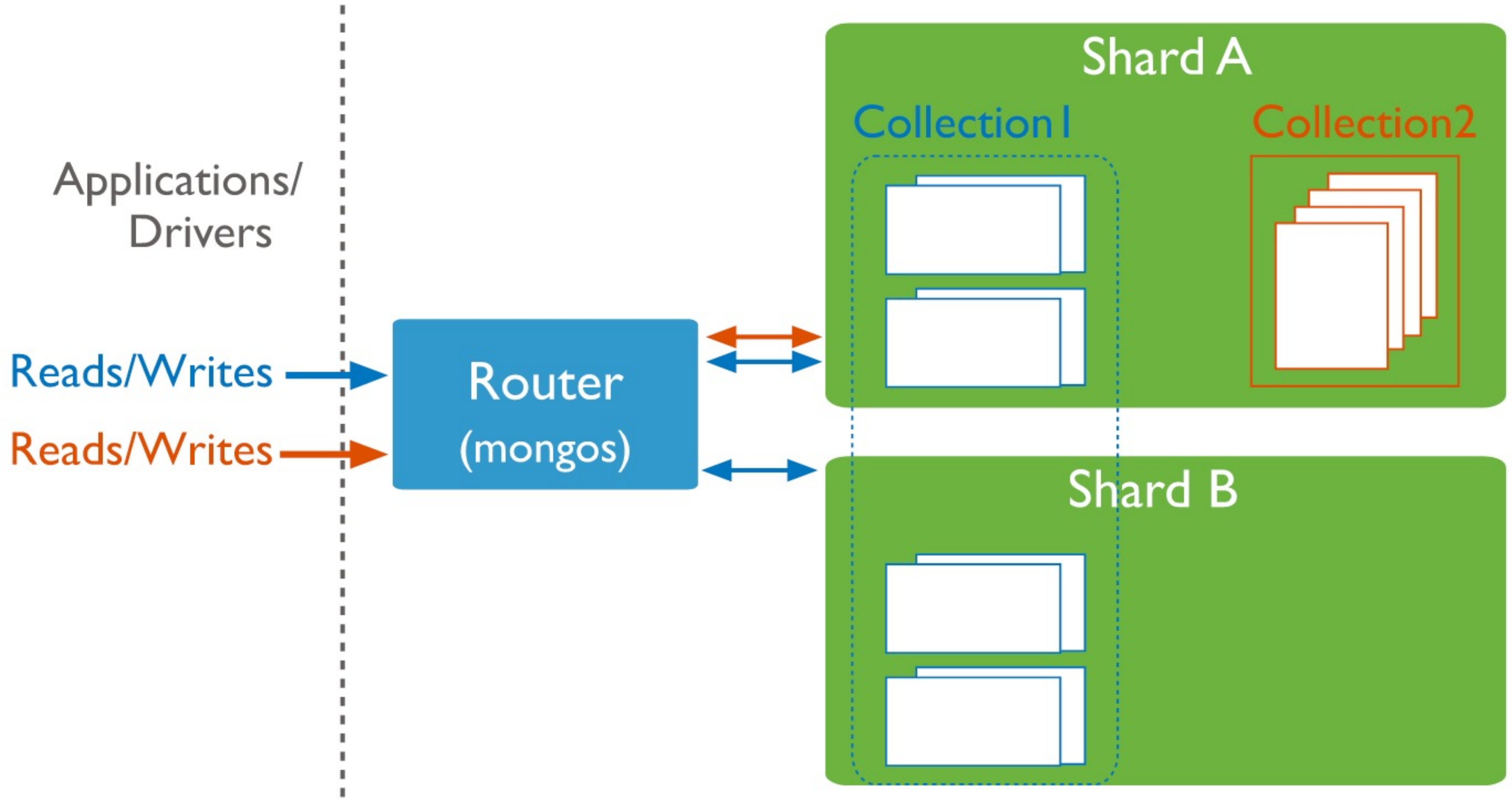
A MongoDB sharded cluster consists of the following components:

- **shard**: Each shard contains a subset of the sharded data. Each shard can be deployed as a replica set.
- **mongos**: The mongos acts as a query router, providing an interface between client applications and the sharded cluster.
- **config servers**: Config servers store metadata and configuration settings for the cluster. As of MongoDB 3.4, config servers must be deployed as a replica set (CSRS).

Sharding

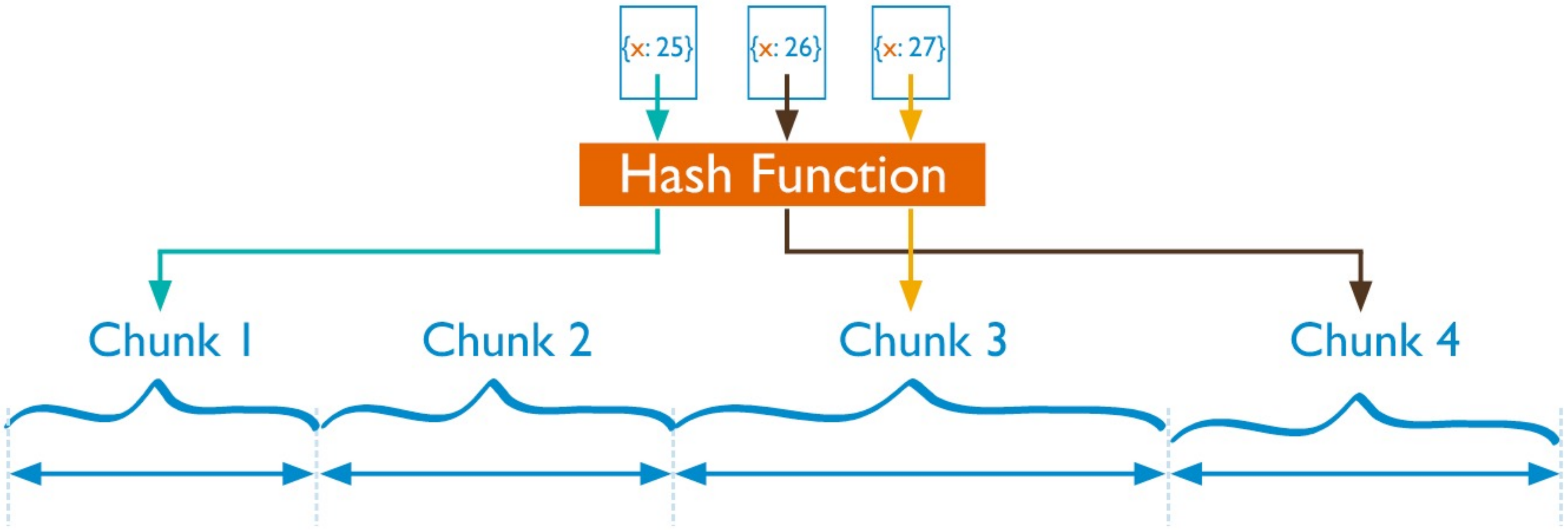

App Server
Router
(mongos)
App Server
Router
(mongos)
1 or more Routers
Config Servers
(replica set)
2 or more Shards
Shard
(replica set)
Shard
(replica set)

Sharding


Applications/
Drivers
Reads/Writes
Reads/Writes
Router
(mongos)
Shard A
Collection1
Collection2
Shard B

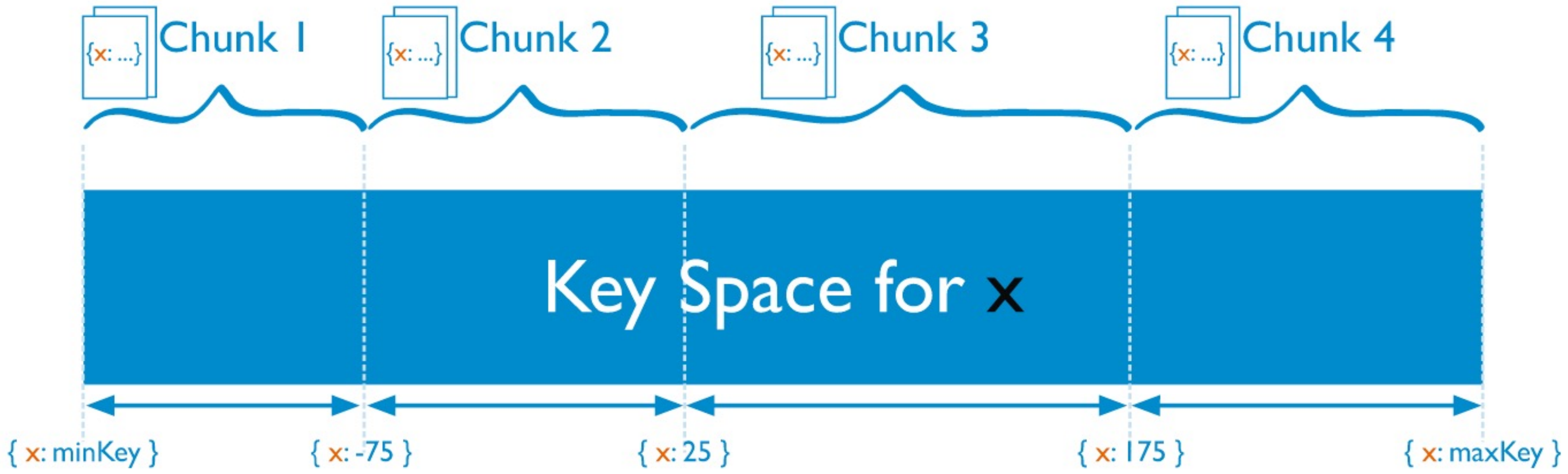
Sharding Strategy

Hashed Sharding involves computing a hash of the shard key field's value. Each chunk is then assigned a range based on the hashed shard key values.

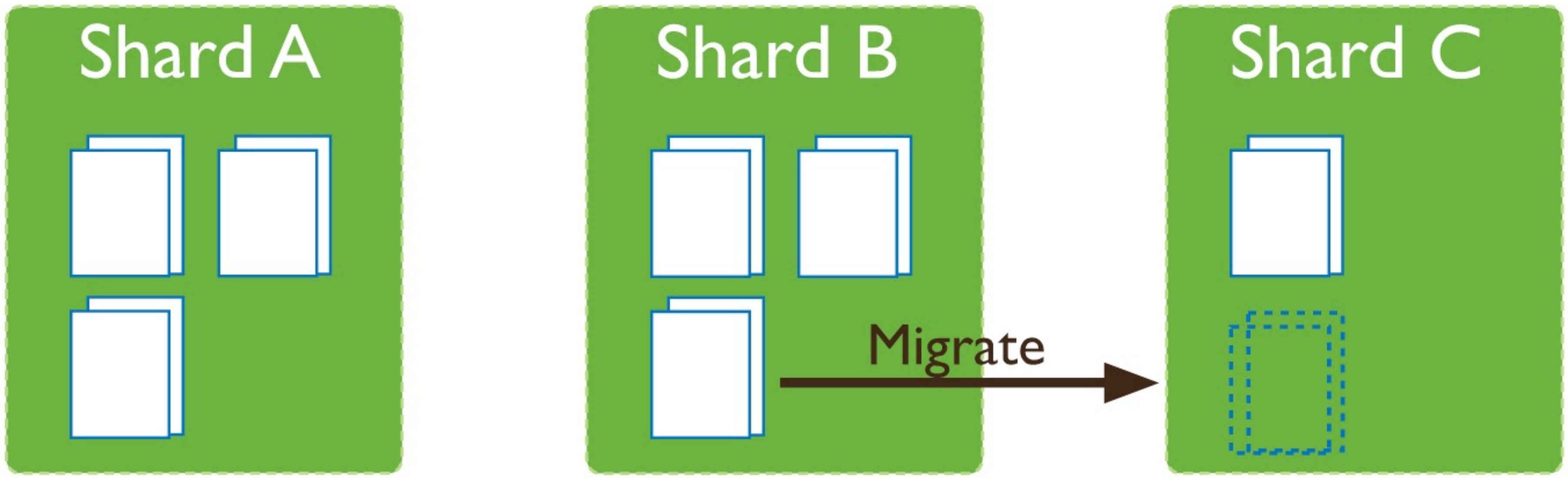


Sharding

Ranged sharding involves dividing data into ranges based on the shard key values. Each chunk is then assigned a range based on the shard key values.



Sharded Cluster Balancer



MongoDB Indexes

MongoDB indexes use a B-tree data structure.

```
db.collection.createIndex( { name: -1 } )
```

MongoDB Indexes

- **Single Field**

- **Compound Index**
  - { a: 1, b: 1 } can support a sort on { a: 1, b: 1 } but *not* on { b: 1, a: 1 }.

  - Sort order is important: an index key pattern { a: 1, b: -1 } can support a sort on { a: 1, b: -1 } and { a: -1, b: 1 } but **not** on { a: -1, b: -1 } or {a: 1, b: 1}.
  - Index prefix: { a:1, b: 1, c: 1, d: 1 }
    - { a: 1 }
    - { a: 1, b: 1 }
    - { a: 1, b: 1, c: 1 }

Specific Index types

- Multikey Index
- Text Index
- Wildcard Index
- 2dsphere Indexes
- 2d Indexes
- geoHaystack Indexes
- Hashed Indexes

https://docs.mongodb.com/manual/indexes/

Before Next week

1) Run the following command and start mongoDB using docker

docker run -p 27017:27017 -v <an empty folder path to store data>:/data/db mongo

Make sure to replace ***<an empty folder path to store data>*** with an actual folder path in your disk.

2) Install MongoDB Compass
https://docs.mongodb.com/compass/master/install/