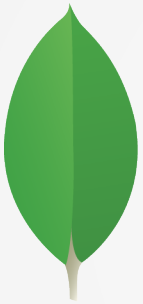


mongoDB®

Bhakti Atul Pradhan

a drop of KNOWLEDGE

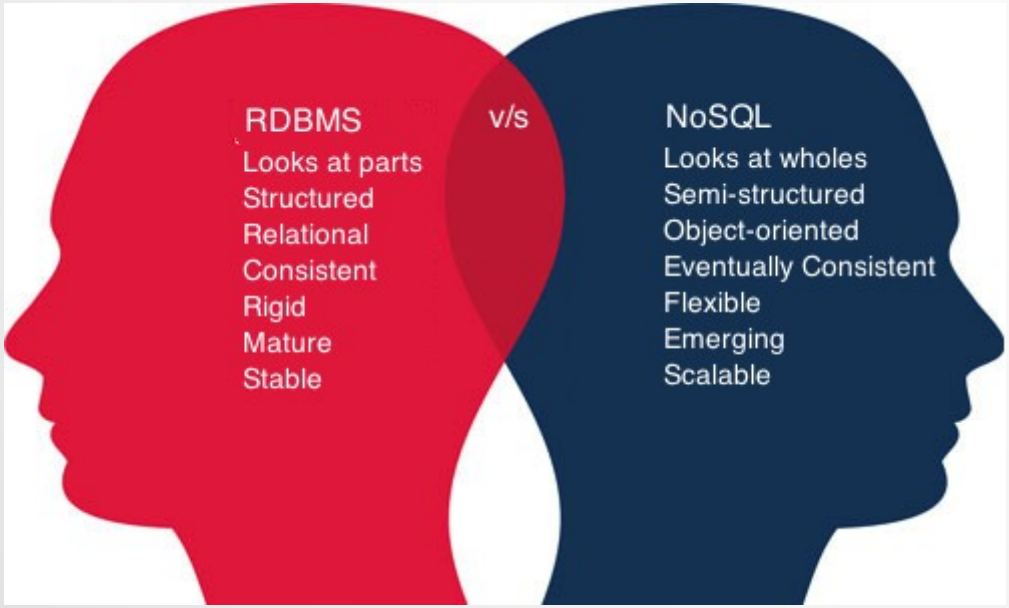
Agenda

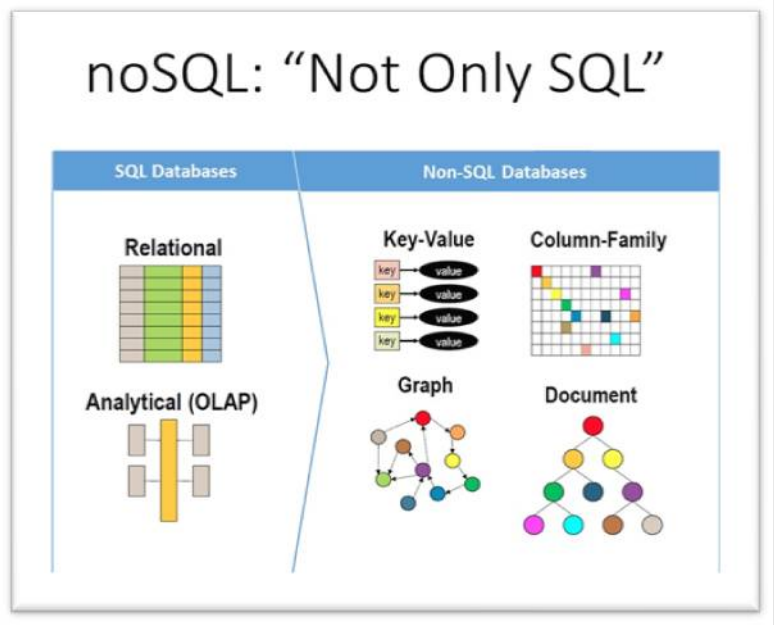
What / Who MongoDB

- MongoDB – hu**mongo**us DB
- MongoDB is a free and open-source cross-platform document-oriented database that provides high performance, high availability, and automatic scaling.
- MongoDB document database with dynamic schema.
- Developed by 10 gen in 2007, later renamed to MongoDB Inc in 2013.
- It is published under a combination of the GNU (General Public License) and the Apache License.

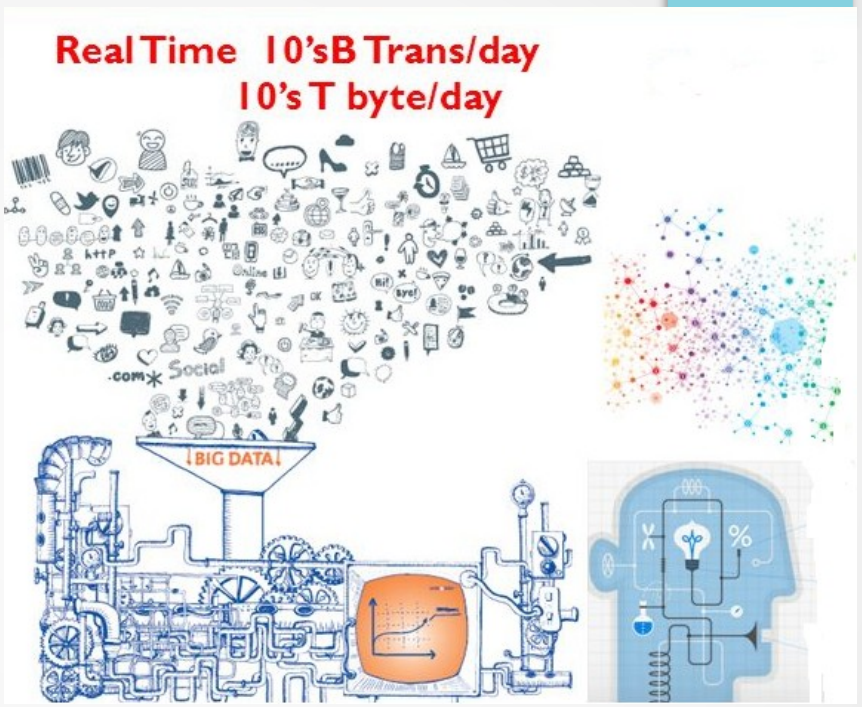
NO SQL (Not Only SQL)

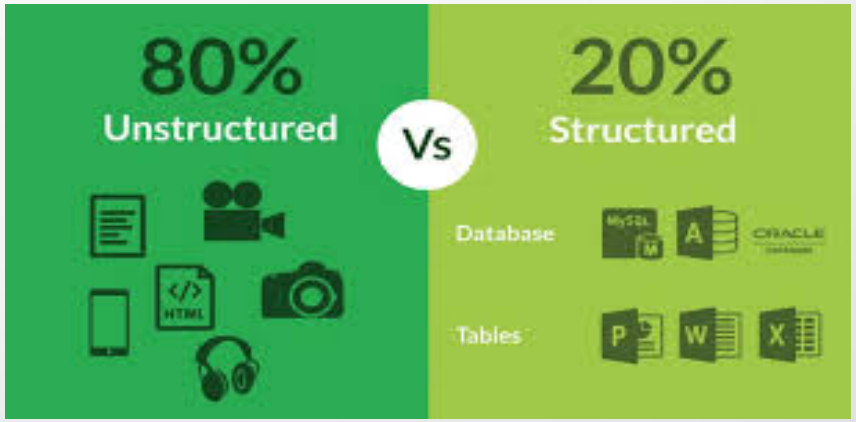
A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.





Why NoSQL / MongoDB





Why NoSQL / MongoDB

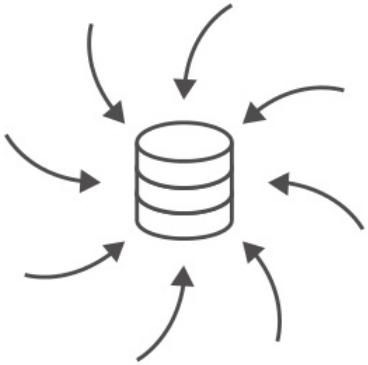
Factors Driving Modern Applications

Mobile

- 2 Billion smartphones
- Mobile now >50% internet use
- 26 Billion devices on IoT by 2020

Data

- 90% data created in last 2 years
- 80% enterprise data is unstructured
- Unstructured data growing 2X rate of structured data



Social

- 72% of internet use is social media
- 2 Billion active users monthly
- 93% of businesses use social media

Cloud

- Compute costs declining 33% YOY
- Storage costs declining 38% YOY
- Network costs declining 27% YOY

MongoDB Features

- Document Based
- Distributed
- High Performance
- Rich Query Language
- High Availability — Replication
- High Scalability – Sharding
- Support for Multiple Storage Engines
- Dynamic — No rigid schema.
- Flexible – field addition/deletion have less or no impact on the application
- Heterogeneous Data
- Data Representation in JSON or BSON
- Geospatial support
- Document-based query language that's nearly as powerful as SQL
- Easy Integration with common languages java, node etc and BigData Hadoop too.
- Cloud distributions such as AWS, Microsoft, RedHat,dotCloud and SoftLayer etc

When MongoDB?

mongoDB

- High performance (1000's – millions queries/sec)
- Need flexible schema
- Rich querying with any number of secondary indexes
- Need for replication across multiple data centers globally
- Need to deploy rapidly and scale on demand (start small and grow easily).
- 99.99999% availability
- Real Time Analysis
- Geospatical Querying
- Processing in real time not in batch
- Deploy over cloud computing and storage architectures
- Agile Project
- Need Strong Data consistency
- Advance Security
- Building Next Gen Solution

Mongo UseCases

MongoDB is good for	MongoDB is less good
✔ Single View ✔ IOT ✔ Big Data ✔ Real Time Analytics ✔ Catalog ✔ Content Management ✔ …….	✗ Search Engine ✗ Slicing and Dicing of data. ✗ Nano sec latency writing data real time ✗ Update beyond 99.999% ✗ Batch processing

Use Mongo Where

- You Expect a High Write Load
- You need High Availability in an Unreliable Environment (Cloud and Real Life)
- You need to Grow Big (and Shard Your Data)
- Your Data is Location Based
- Your Data Set is Going to be Big (starting from 1GB) and Schema is Not Stable

How MongoDB

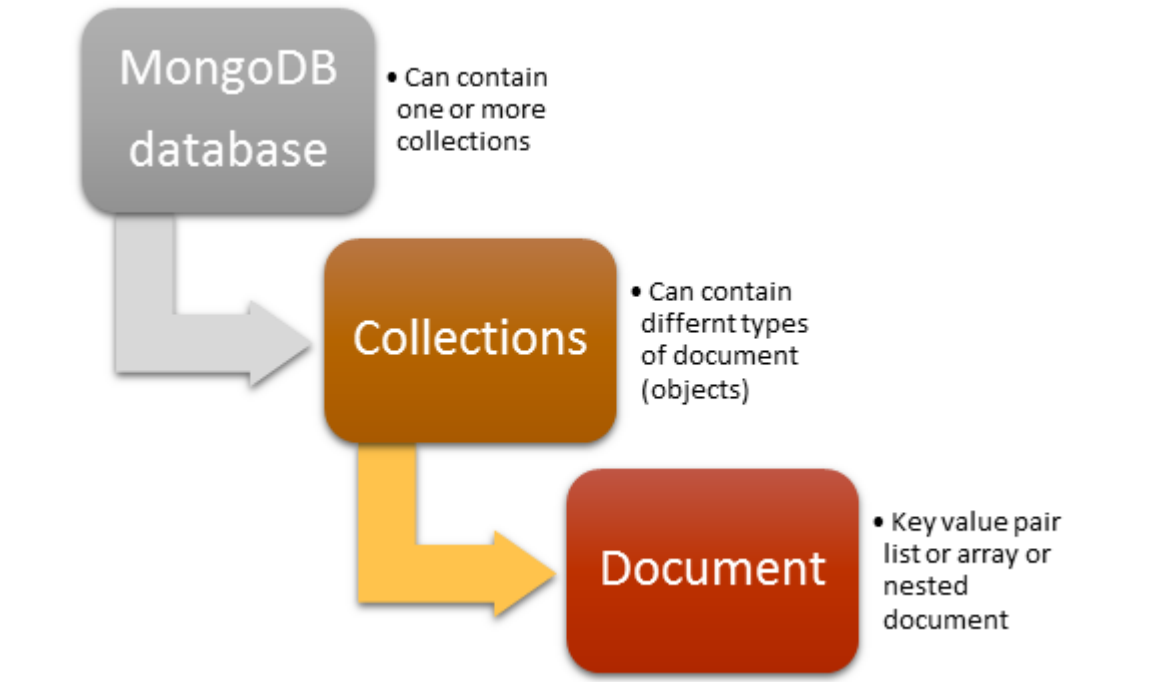
- Getting Started
- Basics with Mongo Shell
- Mongo Compass
- MongoDB CRUD
- Data Modeling
- Indexes
- Replication
- Sharding

Getting Started

- Server
  - MongoDB Atlas is a cloud-hosted service for provisioning, running, monitoring, and maintaining MongoDB.
  - Local – Community/Enterprise

- Client
  - Mongo Shell - an interactive JavaScript interface to MongoDB
  - Mongo Compass - GUI for MongoDB

Basics



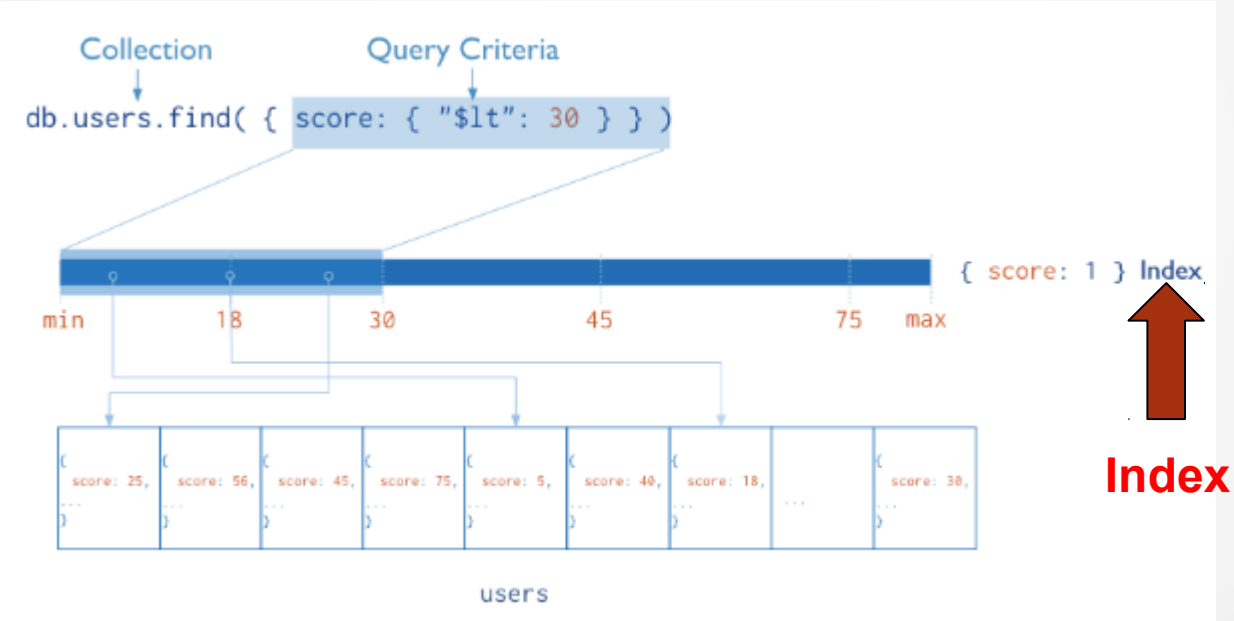
SQL	MongoDB
Table/View	Collection
Row/Tuple	Document
Column	Field
Primary key	_id field, ObjectId
Index	Index
View	View
Joins	$lookup, Embedded document
Uniformity Schema	Uniformity not Required
Foreign Key	Reference

CRUD

- Create
  - db.collection.insert(<document>)
  - db.collection.save(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- Read
  - db.collection.find(<query>, <projection>)
  - db.collection.findOne(<query>, <projection>)
- Update
  - db.collection.update(<query>, <update>, <options>)
- Delete
  - db.collection.remove(<query>, <justOne>)

Indexes

Indexes are special data structures that store a small portion of the collection's data set in an easy to traverse form.



Indexes

- Creation index
  - db.users.createIndex({ score: 1 })
- Show existing indexes
  - db.users.getIndexes()
- Drop index
  - db.users.dropIndex({score: 1})
- Explain—Explain
  - db.users.find().explain()
  - Returns a document that describes the process and indexes

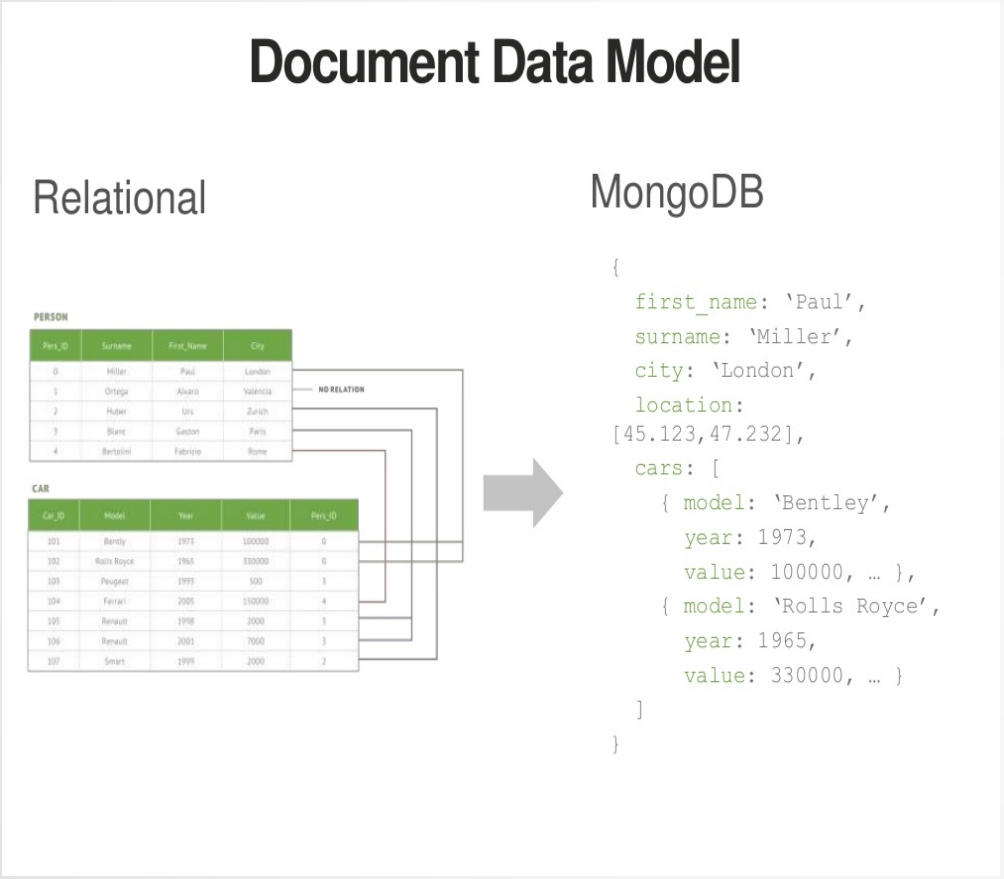
Indexes - Types

- Default Index
- Single Index
- Compound Index
- Unique Index
- MultiKey Index
- Sparse Index
- Geospatial Index
- Text Index

Schema Design

- 3NF vs Application Driven Schema
- Embedded or Not Embed (Linking)
  - Access same time by the application
  - Its existence is dependent on the parent existence
  - MongoDB cannot be more than 16MB, so if document gets greater than 16MB move to different collection

Schema Design



- One to One
  - Person, current address → Embed
- One to Many
  - City, Person → Link/Reference
- One to Few
  - Posts, Comments → Embed
- Many to Many
  - Books : Authors → Link
  - Students : Teachers → Link

Replication

- A process of synchronizing data across multiple servers.
- provides redundancy and increases data availability with multiple copies of data on different database servers.
- protects a database from the loss of a single server.
- allows you to recover from hardware failure and service interruptions.

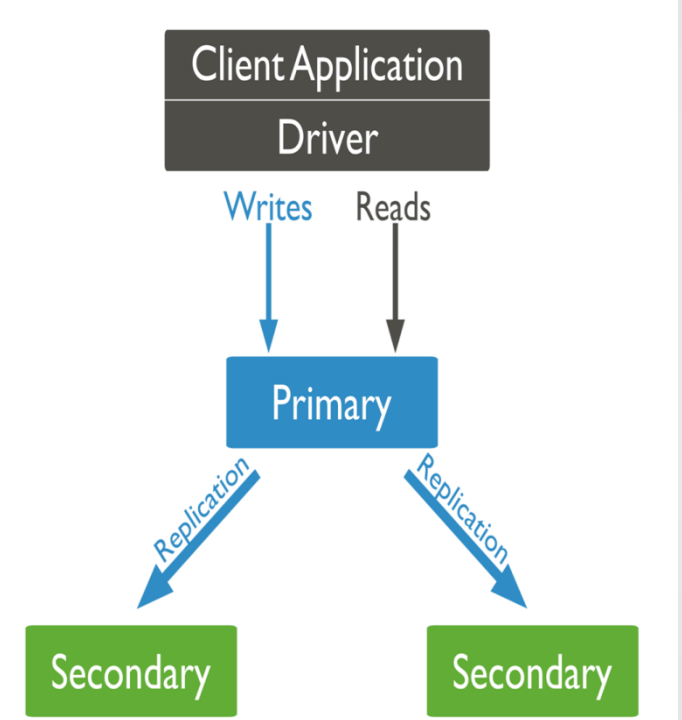


Figure 1: Diagram of default routing of reads and writes to the primary.

Replication – Replica Set

•Replica set is a group of two or more mongod instances having same data set i.e. nodes > 3 - usually an odd no.
• Only one node is primary node and remaining nodes are secondary
• primary node receives all write operations.
• All data replicates from primary to secondary node
• An arbiter node is added for election when only two or even nodes are present in the replica set

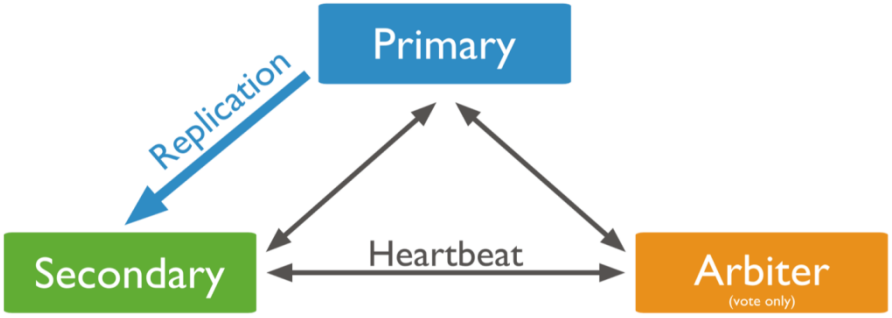


Figure 3: Diagram of a replica set that consists of a primary, a secondary, and an arbiter.

Replication

- At the time of automatic failover or maintenance, election establishes for primary and a new primary node is elected.
- After the recovery of failed node, it again joins the replica set and works as a secondary node.

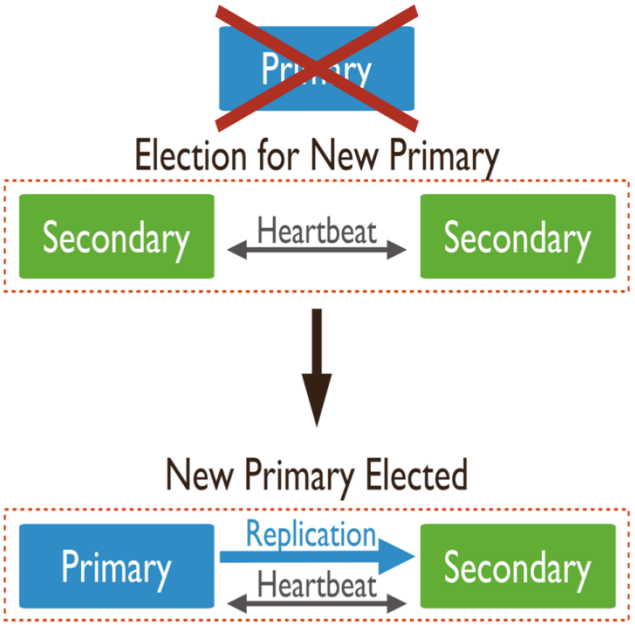
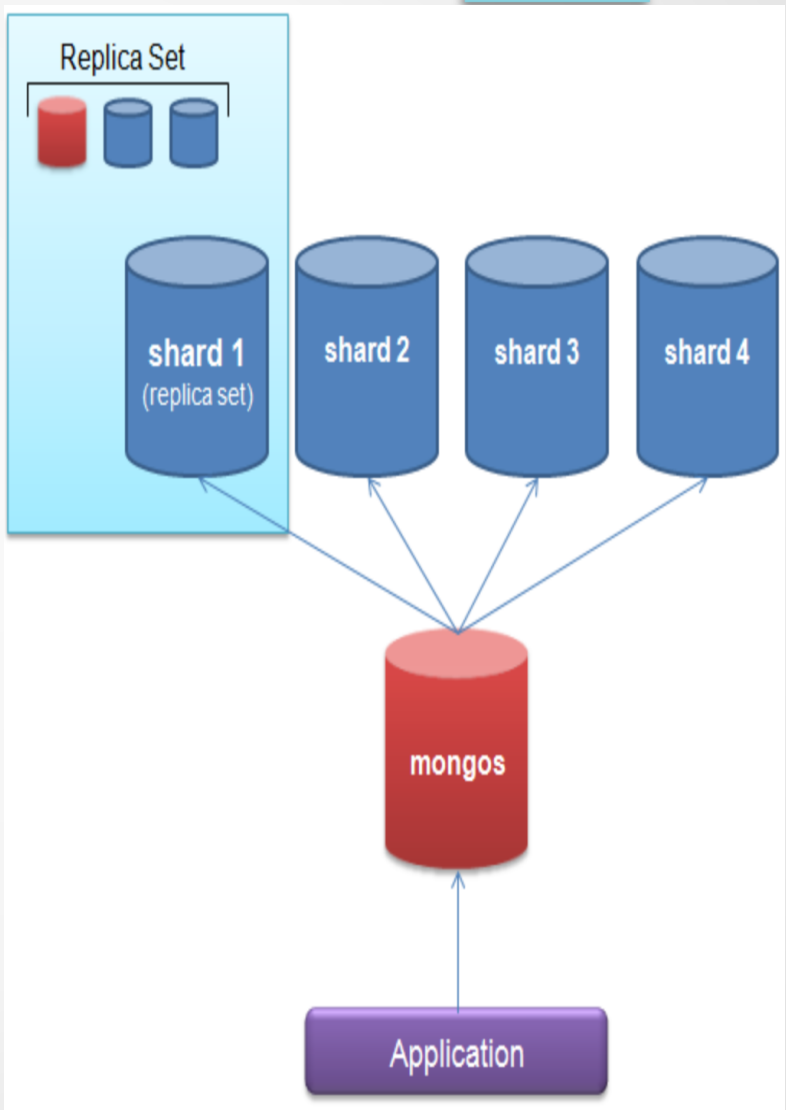


Figure 21: Diagram of an election of a new primary. In a three member replica set with two secondaries, the primary becomes unreachable. The loss of a primary triggers an election where one of the secondaries becomes the new primary

Sharding

- A method for distributing data across multiple machines.
- It is horizontal scaling.
- It involves dividing the system dataset and load over multiple servers
- Application interact with router



Questions

Thank You

NO SQL (Not Only SQL) VS SQL

	SQL	NOSQL
History	Developed in 1970s to deal with first wave of data storage applications	Developed in late 2000s to deal with limitations of SQL databases,
Examples	MySQL, Oracle Database	MongoDB, Cassandra, HBase,
Data Model	table based databases	document based, key-value pairs, graph databases or wide-column stores.
Schemas	Predefined	Dynamic
Scaling	Vertically	Horizontally
Supports multi-record ACID transactions	Yes	Will support in 4.0 version
hierarchical data storage	No	Yes