Knowledge Sharing on mongoDB®

U Naing Win Tun

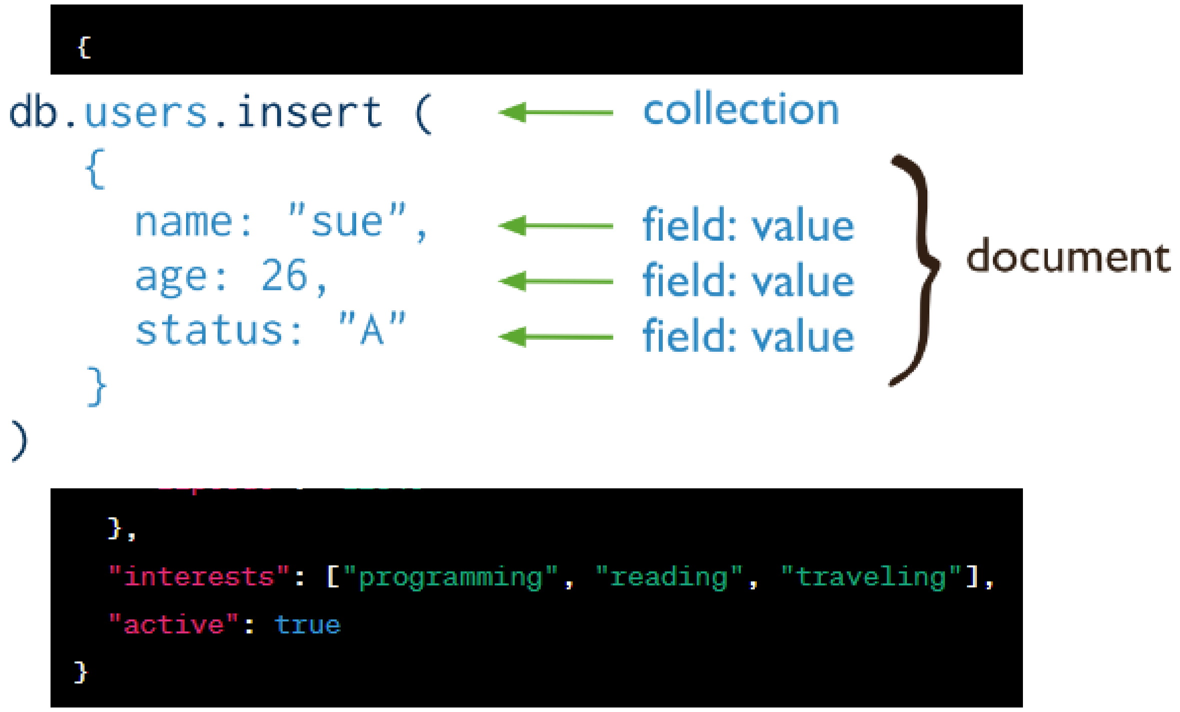
Image: https://commons.wikimedia.org/wiki/File:MongoDB_Logo.svg

Agenda

- Overview
- Environment
- Implementation on mongodb

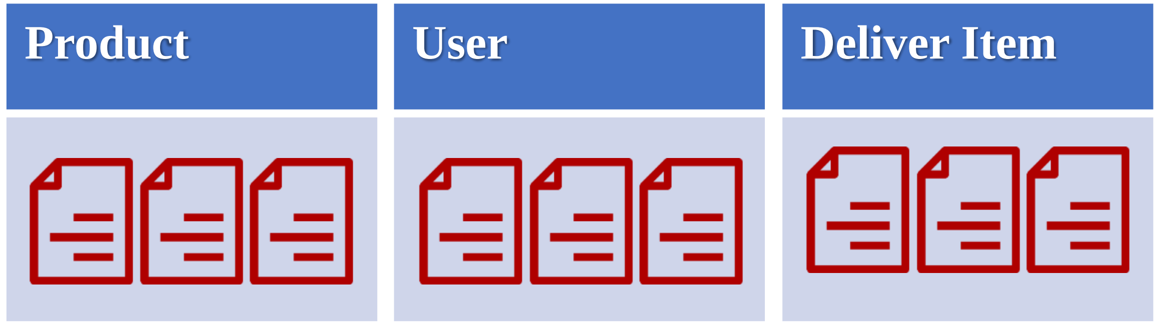
Overview

- Opensource document and NoSQL database
- Key-Value set pairs

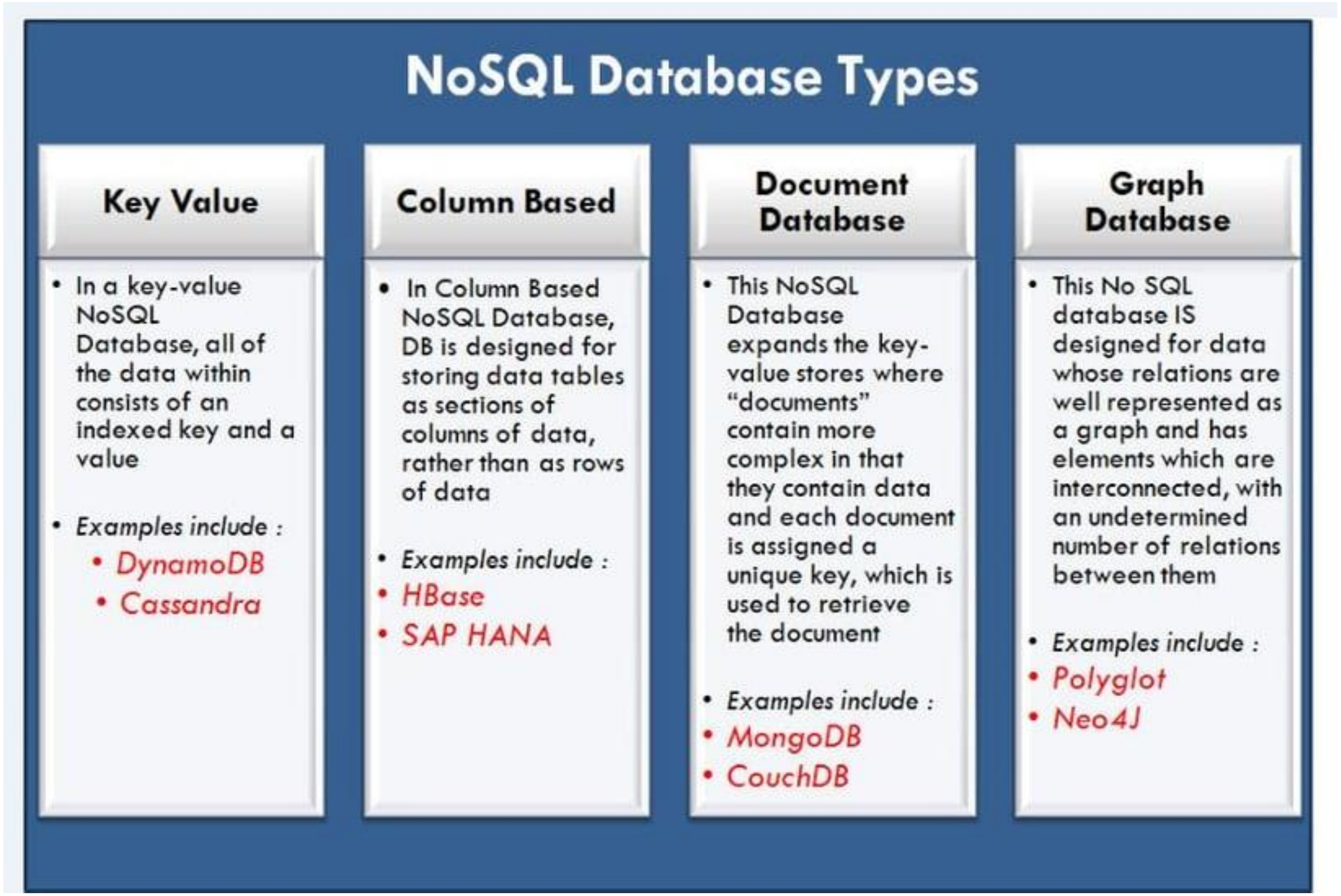


```
{
db.users.insert (
   {
      name: "sue",
      age: 26,
      status: "A"
   }
)
  },
  "interests": ["programming", "reading", "traveling"],
  "active": true
}
```

Comparison	
RDBMS	**MongoDB**
Table	Collection
Row	Document
Column	Field
Primary Key	Default
MySQL	Mongod
MySQL	mongosh



NOSQL



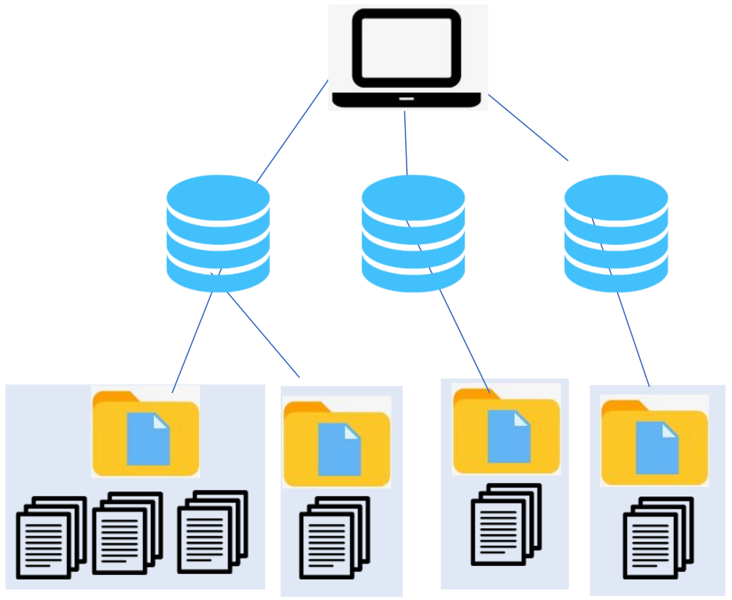
NoSQL Database Types

Key Value	Column Based	Document Database	Graph Database
• In a key-value NoSQL Database, all of the data within consists of an indexed key and a value • *Examples include :* • *DynamoDB* • *Cassandra*	• In Column Based NoSQL Database, DB is designed for storing data tables as sections of columns of data, rather than as rows of data • *Examples include :* • *HBase* • *SAP HANA*	• This NoSQL Database expands the key-value stores where “documents” contain more complex in that they contain data and each document is assigned a unique key, which is used to retrieve the document • *Examples include :* • *MongoDB* • *CouchDB*	• This No SQL database IS designed for data whose relations are well represented as a graph and has elements which are interconnected, with an undetermined number of relations between them • *Examples include :* • *Polyglot* • *Neo4J*

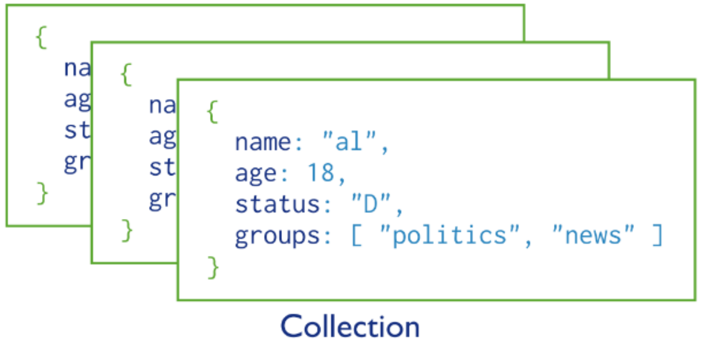
reference: https://www.clariontech.com/blog/applications-that-work-best-with-nosql-database

Model

- A Mongo system holds a set of databases
- A database holds a set of collections
- A collection holds a set of documents
- A document is a set of fields
- A field is a key-value pair
- A key is a name(string)
- A values is a
  - basic type like string , integer , float , binary , etc,…
  - a document ,or
  - an array of values

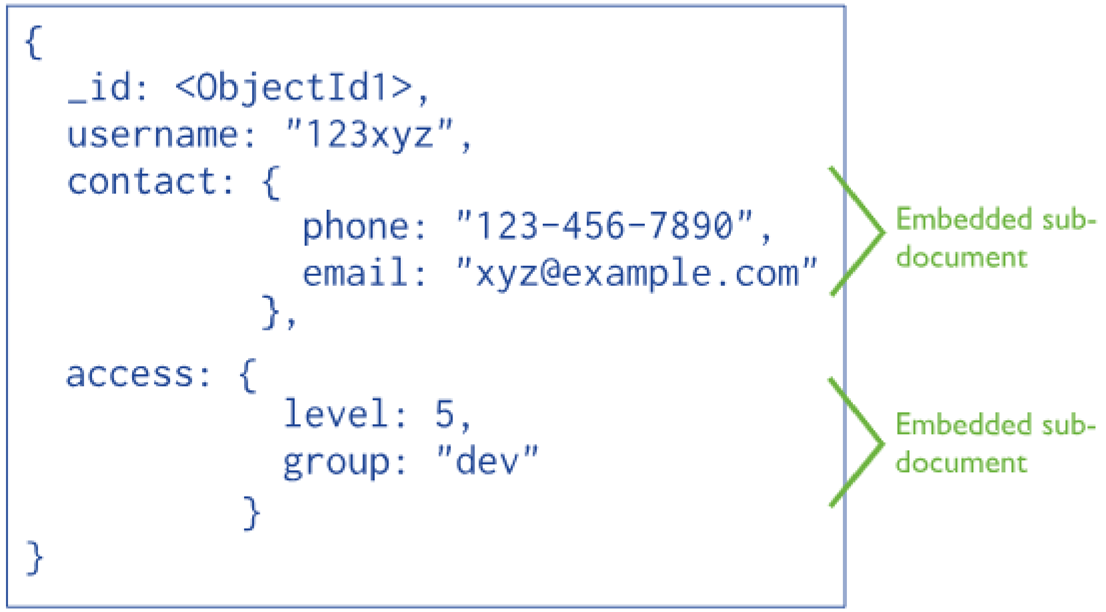


- Embedded data model
- Normalized data model

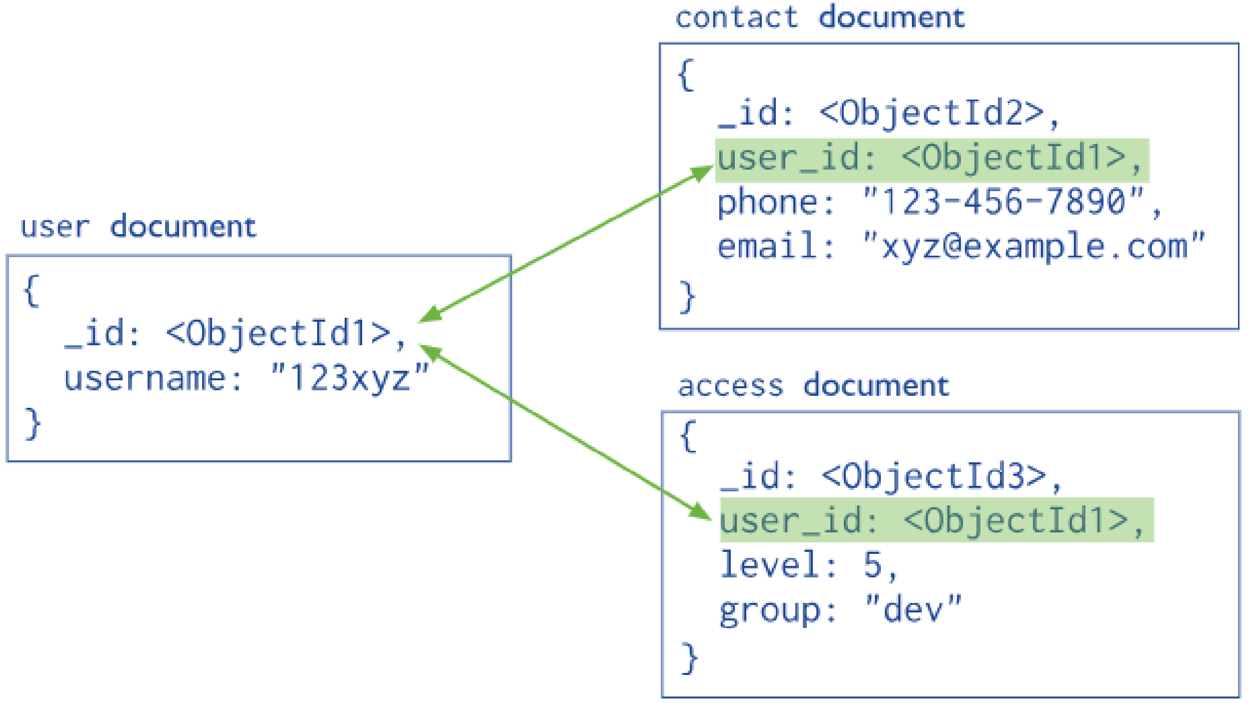


```
{
  name: "al",
  age: 18,
  status: "D",
  groups: [ "politics", "news" ]
}
```

Embedded and Normalized



```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
              phone: "123-456-7890",
              email: "xyz@example.com"
            },
  access: {
             level: 5,
             group: "dev"
          }
}
```



user document

```
{
  _id: <ObjectId1>,
  username: "123xyz"
}
```

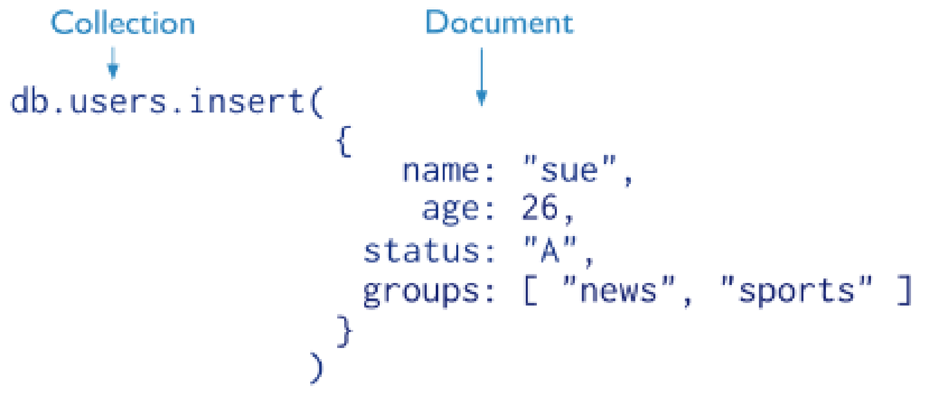
contact document

```
{
  _id: <ObjectId2>,
  user_id: <ObjectId1>,
  phone: "123-456-7890",
  email: "xyz@example.com"
}
```

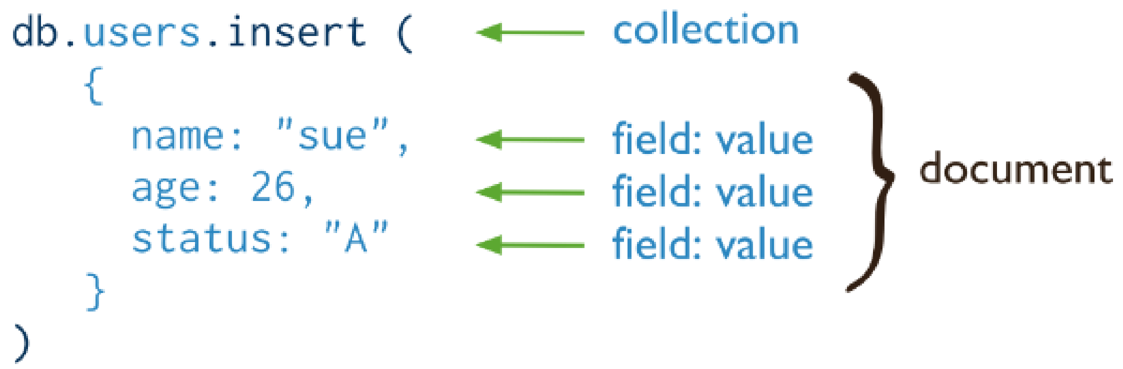
access document

```
{
  _id: <ObjectId3>,
  user_id: <ObjectId1>,
  level: 5,
  group: "dev"
}
```

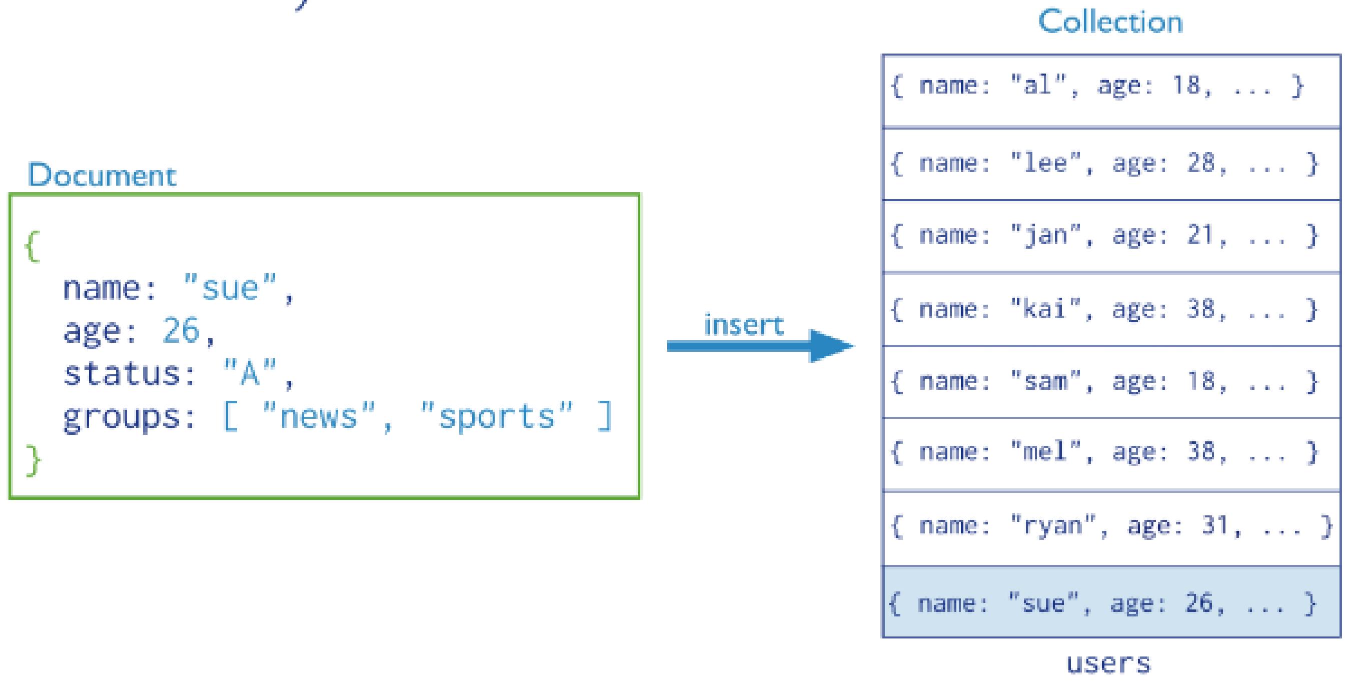
Method and Operators



```
db.users.insert(
    {
        name: "sue",
         age: 26,
      status: "A",
      groups: [ "news", "sports" ]
    }
)
```



```
db.users.insert (
   {
     name: "sue",
     age: 26,
     status: "A"
   }
)
```



```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

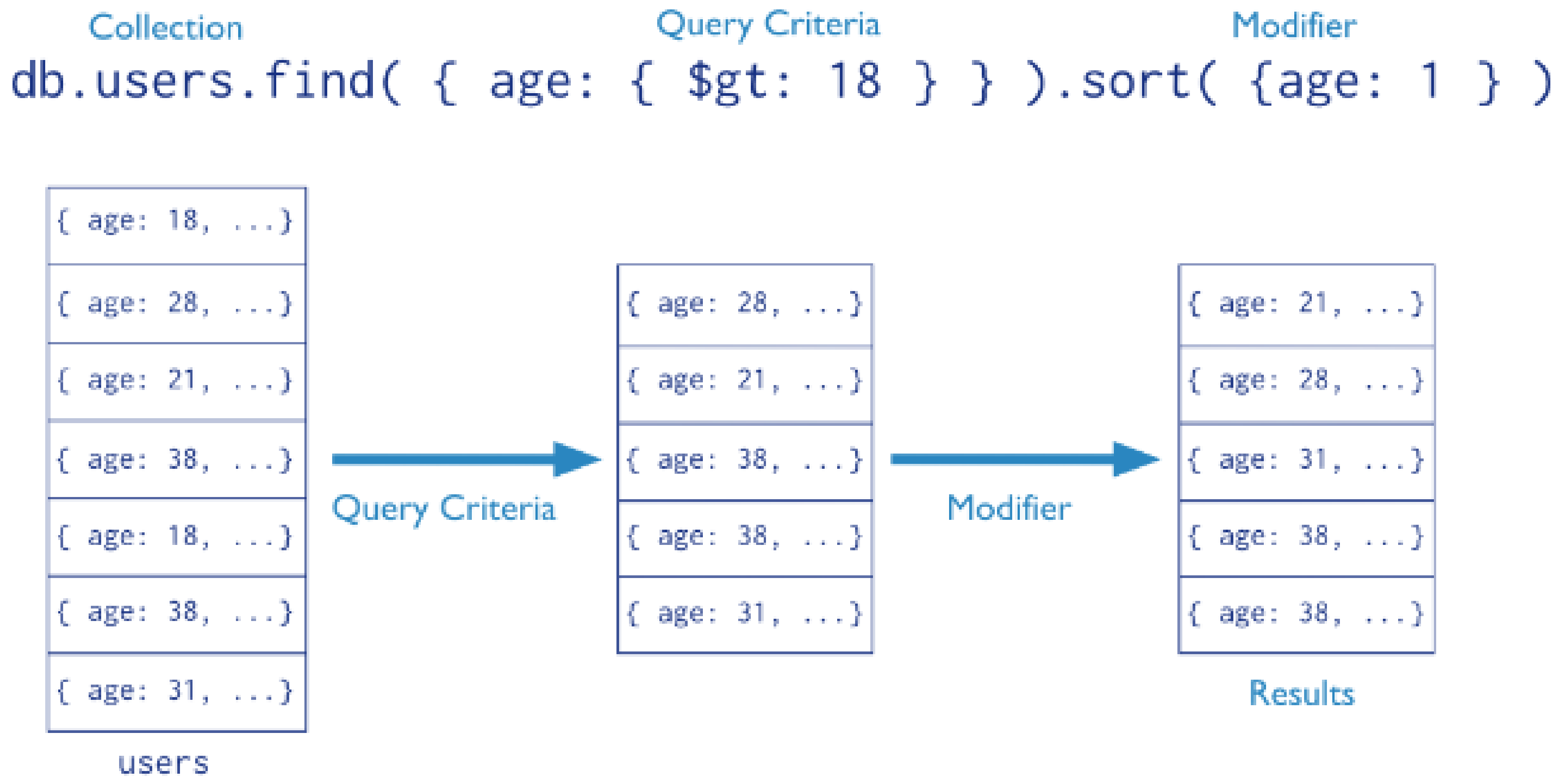
```
{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }
```

Mongodb Installation

Method and Operators

- Comparison operators
  - $eq, $ne, $gt, $gte, $lt, $lte, $in, $nin
- Logical operators
  - $and, $or, $nor, $not
- Update operators
  - $inc, $min, $max

Querying


Collection
Query Criteria
Modifier
db.users.find({ age: { $gt: 18 } }).sort({age: 1 })
{ age: 18, ...}
{ age: 28, ...}
{ age: 21, ...}
{ age: 38, ...}
{ age: 18, ...}
{ age: 38, ...}
{ age: 31, ...}
users
Query Criteria
{ age: 28, ...}
{ age: 21, ...}
{ age: 38, ...}
{ age: 38, ...}
{ age: 31, ...}
Modifier
{ age: 21, ...}
{ age: 28, ...}
{ age: 31, ...}
{ age: 38, ...}
{ age: 38, ...}
Results

Environment – *Snap glance*

- **www.mongodb.com**

- Indexing
- Aggregation
- Replication
- Sharding

Indexing

- Benefit:
  - Efficiency performance
  - Avoid collection scan (Table scan)
  - Effective indexing search

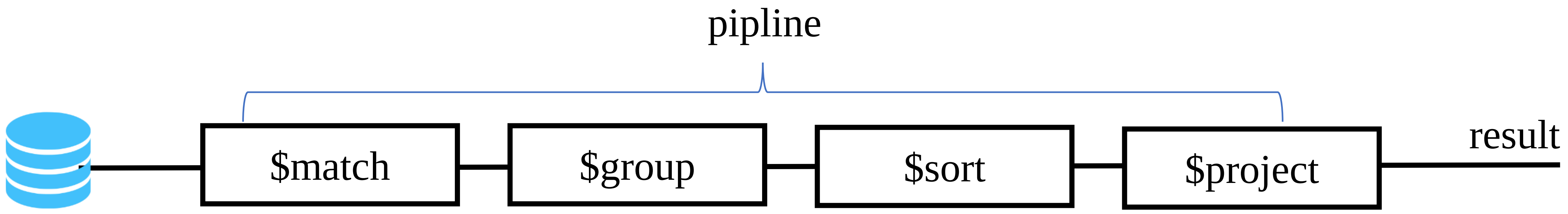
- Types
  a) Single field Index
  b) Compound Index
  c) Multi Key Index

a) db.coll_name.createIndex({field_name: 1 or -1})

b) db.coll_name.createIndex({field_name: type1, field_name:type2})

c) db.coll_name.createIndex({field_name: type})

Aggregation

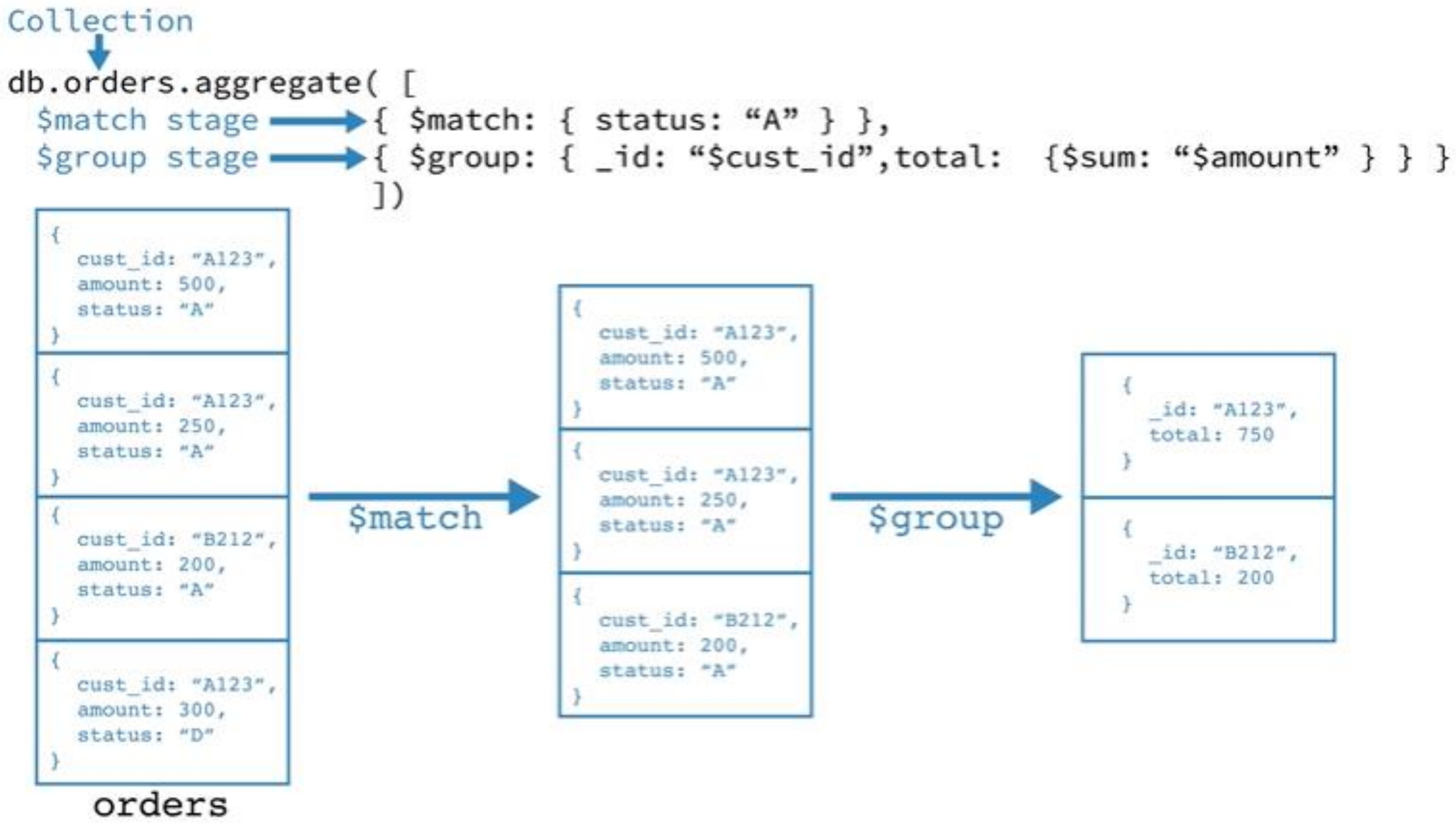


a) db.coll_name.createIndex({field_name: 1 or -1})

b) db.coll_name.createIndex({field_name: type1, field_name:type2})

c) db.coll_name.createIndex({field_name: type})

Aggregation



```
db.orders.aggregate( [
  { $match: { status: "A" } },
  { $group: { _id: "$cust_id",total:  {$sum: "$amount" } } }
])
```

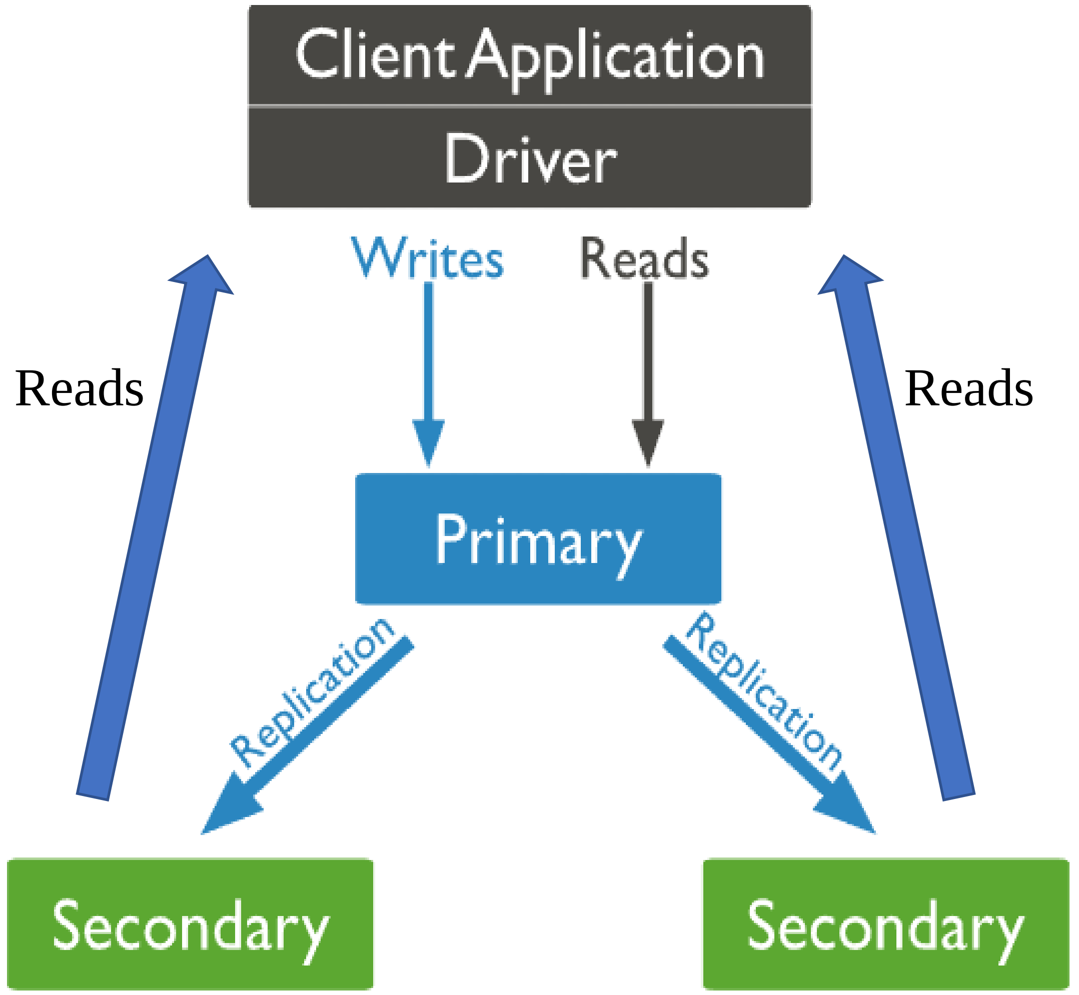
```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
{
  cust_id: "A123",
  amount: 300,
  status: "D"
}
```

```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
```

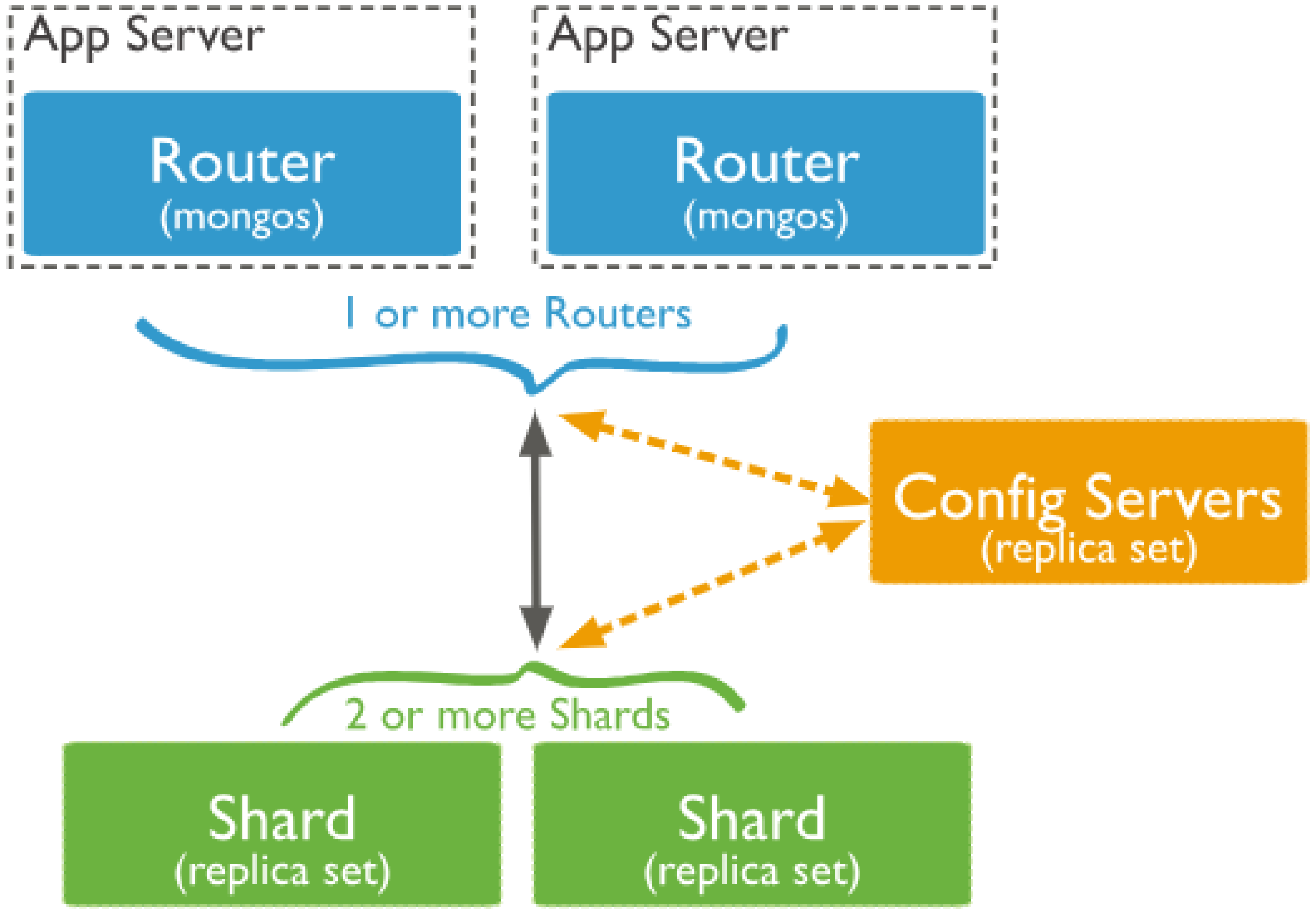
```
{
  _id: "A123",
  total: 750
}
{
  _id: "B212",
  total: 200
}
```

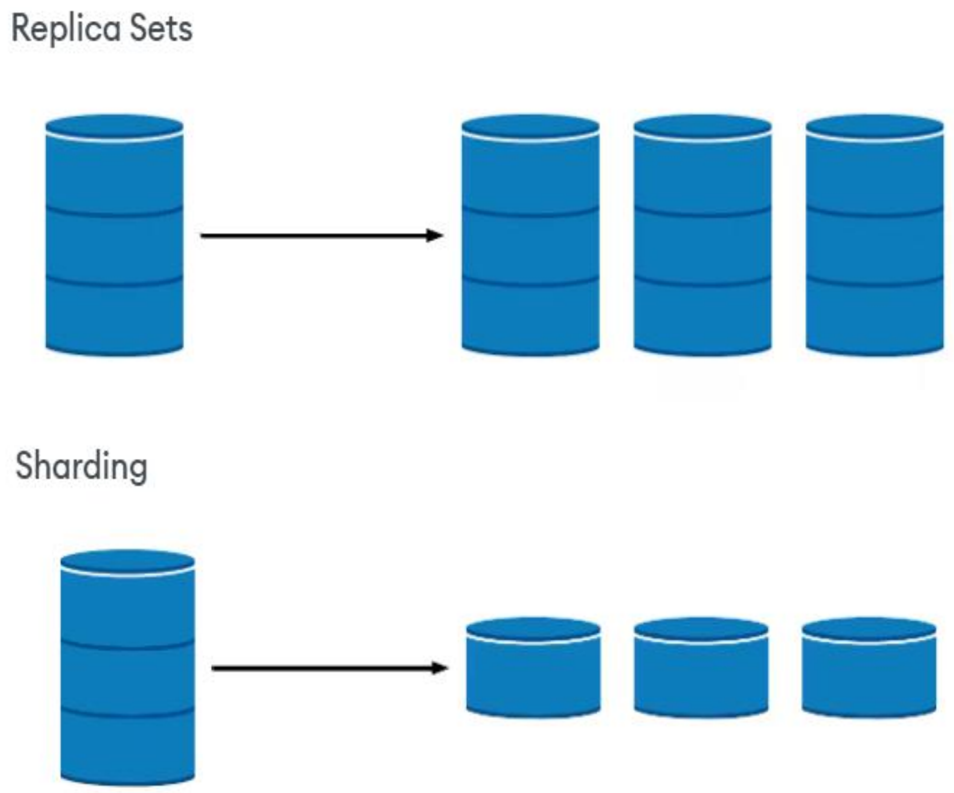
reference: https://devopedia.org/mongodb-aggregation

Replication

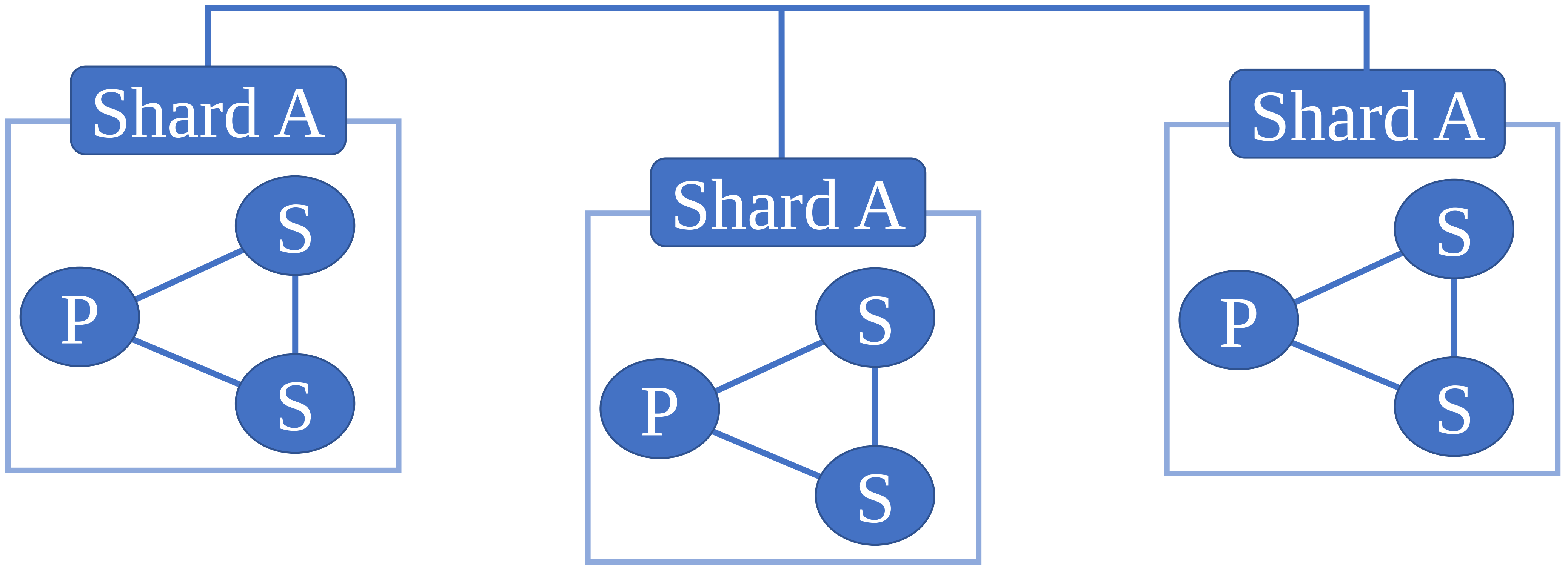


Sharding

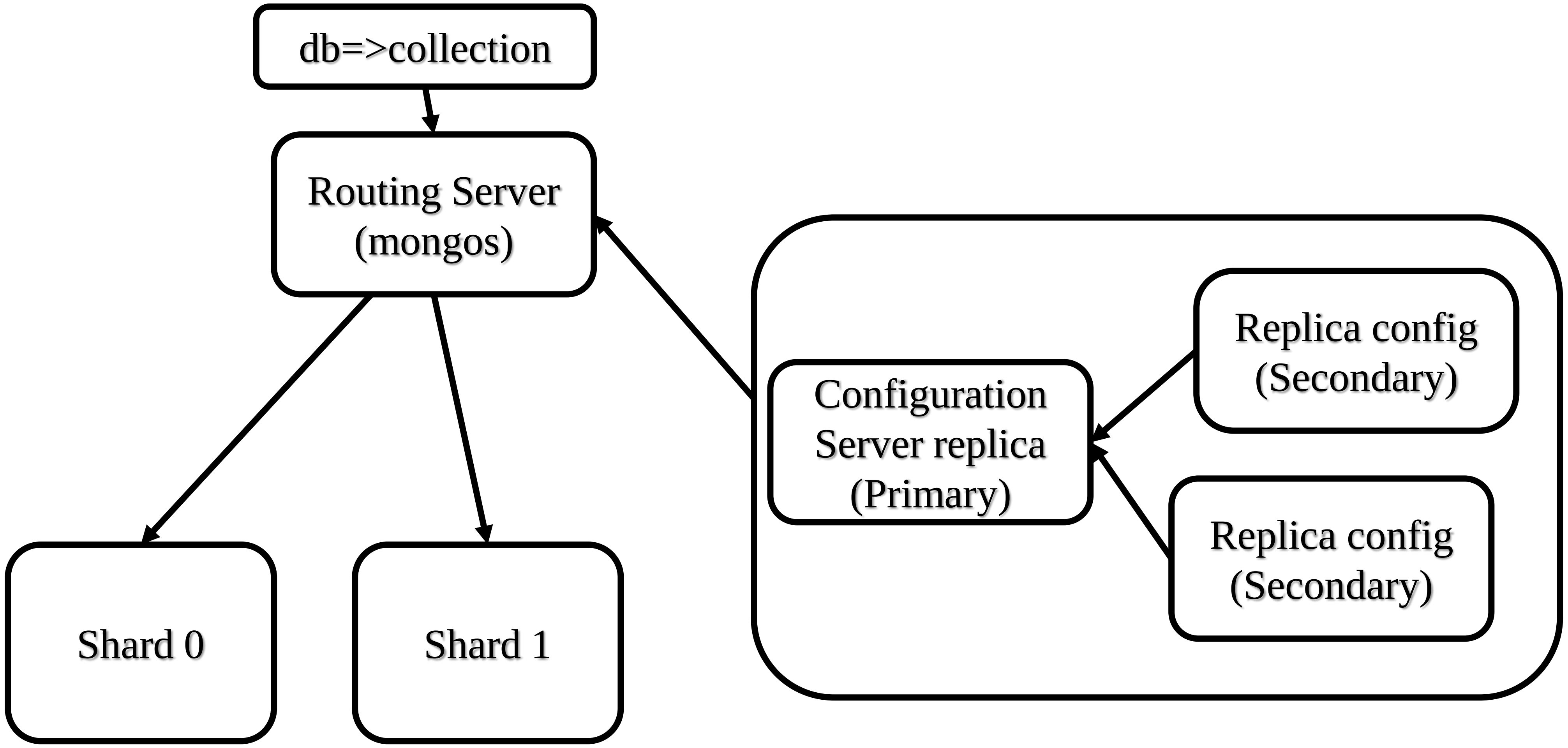

App Server
Router
(mongos)
App Server
Router
(mongos)
1 or more Routers
Config Servers
(replica set)
2 or more Shards
Shard
(replica set)
Shard
(replica set)


Replica Sets
Sharding

Sharding



Sharding


db=>collection
Routing Server (mongos)
Shard 0
Shard 1
Configuration Server replica (Primary)
Replica config (Secondary)
Replica config (Secondary)

CRUD with nodejs driver

node JS + mongoDB

Reference

- www.mongodb.com

mongoDB®

Thank you for your attention.

U Naing Win Tun

Image: https://commons.wikimedia.org/wiki/File:MongoDB_Logo.svg