Системный Дизайн

nosql

Сравнение баз данных

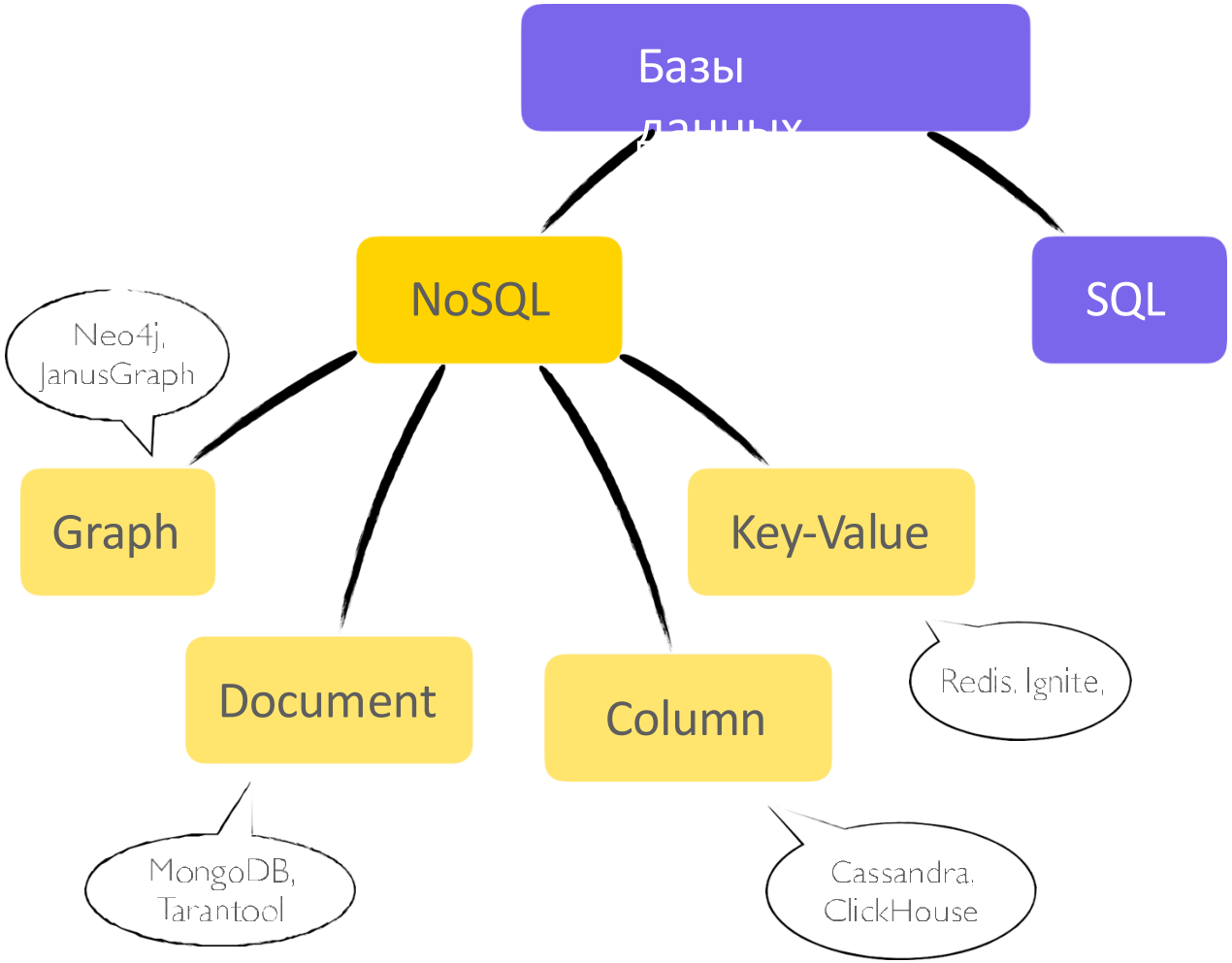
SQL

- Таблицы с фиксированными строками и столбцами
- Создавались для снижения дублирующих данных
- Жёсткая схема данных
- Вертикальное масштабирование*
- Управление связными данными со сложными соединениями

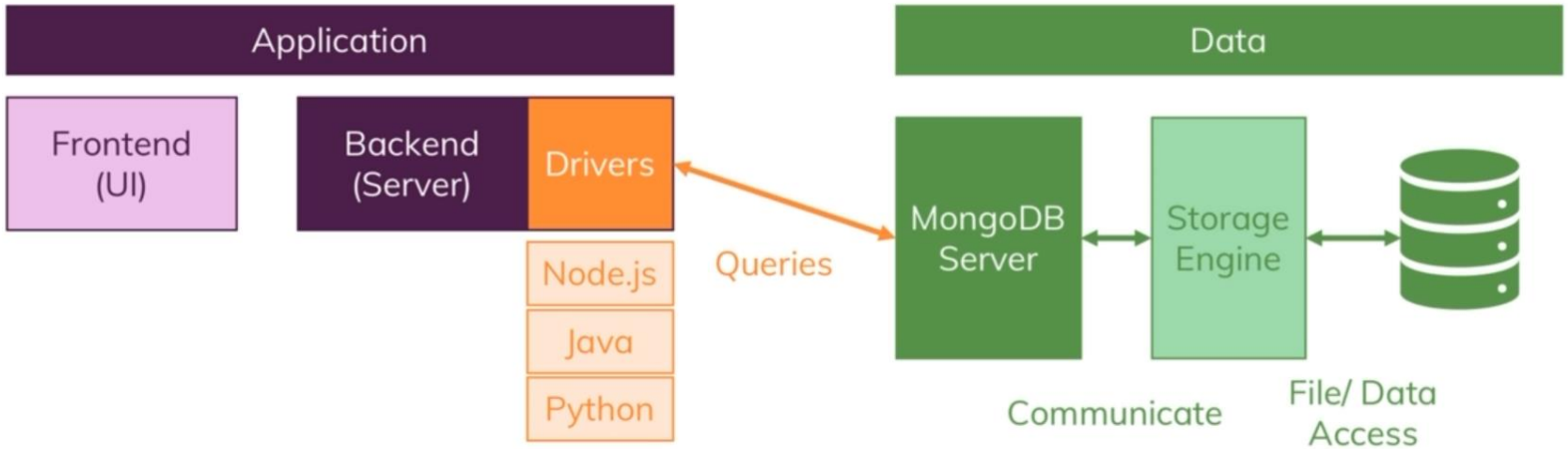
NoSQL

- Json’ы / пары ключ-значение / таблицы со строками и динамическими столбцами / узлы и звенья
- Создавались для просто масштабирования и возможности быстрых изменений
- Гибкая схема данных
- Горизонтальное масштабирование
- Управление данными в части фильтрация, группировки или упорядочивания, особенно в real-time

Нереляционные БД (NoSQL)



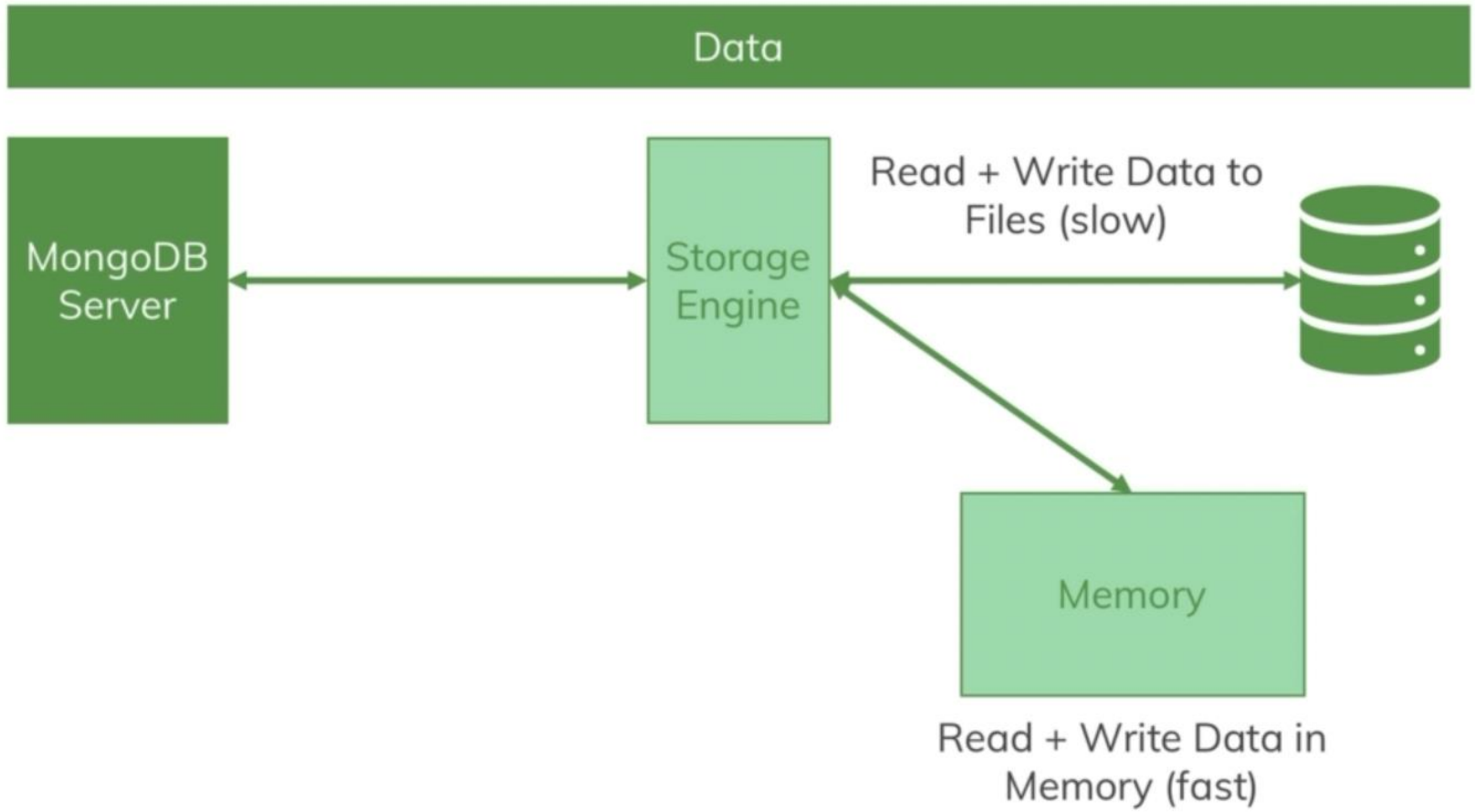
Работа с MongoDB



- Драйвера есть для большинства языков программирования
- Данные сохраняются не напрямую а с помощью Storage Driver (по умолчанию WiredTiger)

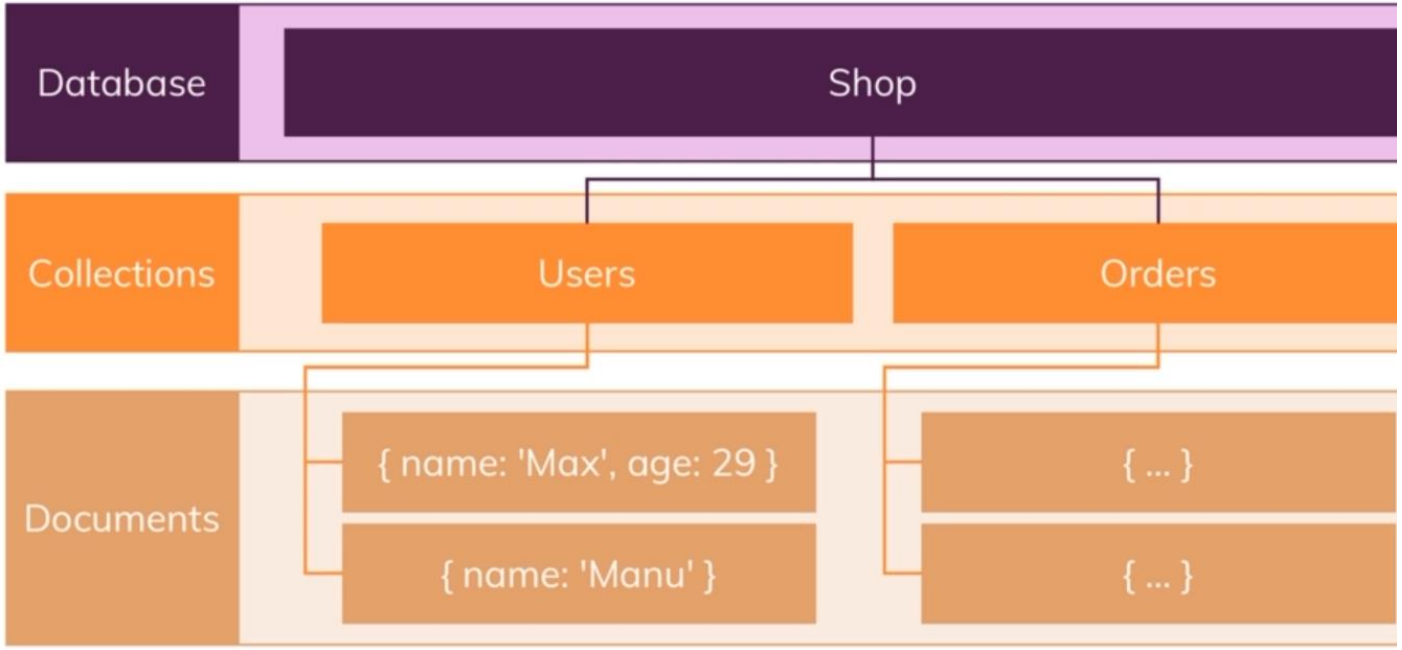
Data Layer

- Содержит встроенные механизмы кеширования для ускорения работы
- Содержит персистентное хранилище



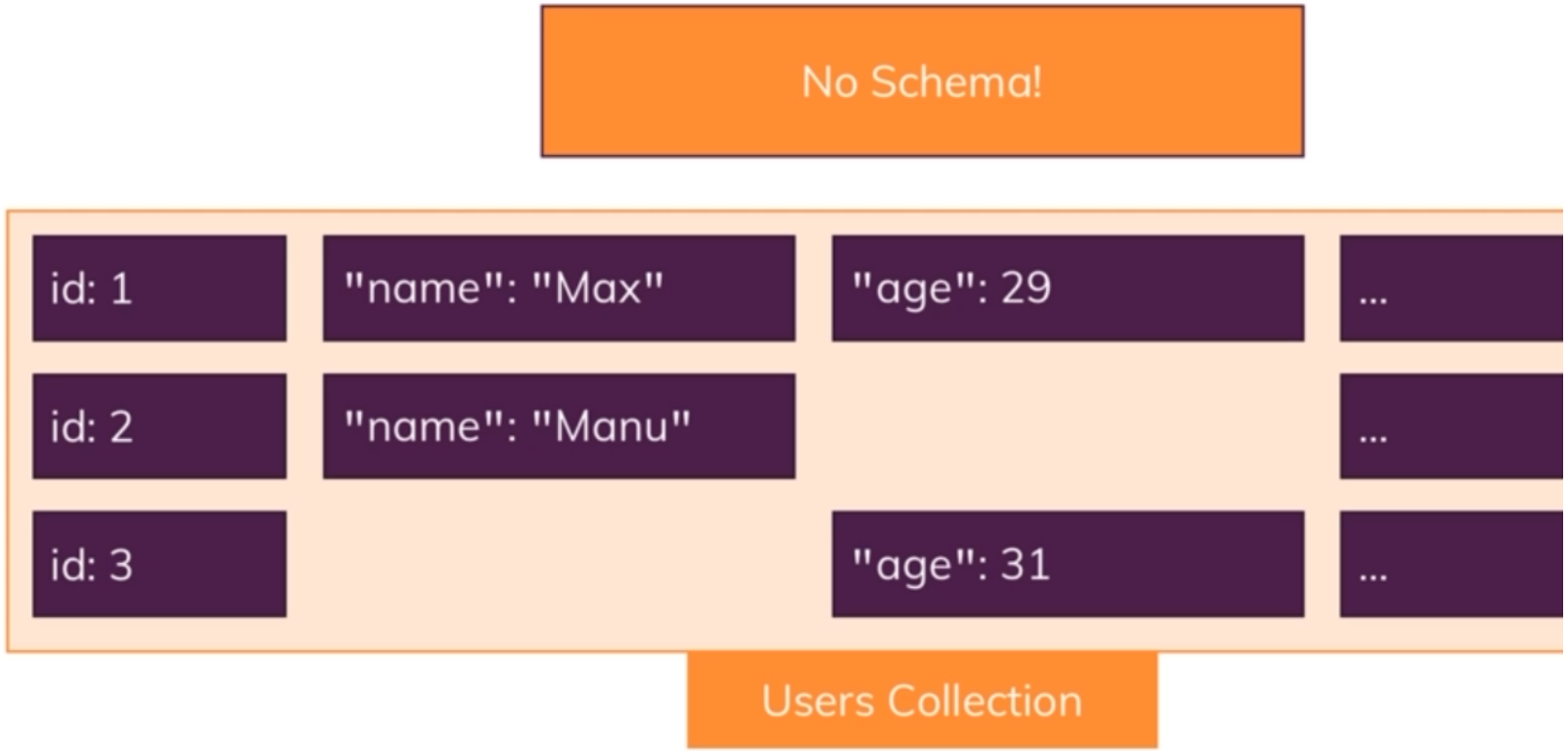
Как работает Mongo

- Отсутствие таблиц – вместо них Колекции
- Отсутствие схем (каждый документ может отличаться от другого)
- Документ - json



BSON Data Structure

- BSON — это двоичный формат сериализации
- Отсутствуют «связи» – вместо этого данные храниться вместе



Целостность данных в MongoDB

MongoDB использует концепцию, называемую «eventual consistency», означающую, что, хотя данные в конечном итоге могут стать согласованными на всех узлах, могут быть моменты, когда разные узлы имеют немного разные версии данных.

Этот подход хорошо подходит для таких приложений, как аналитика в реальном времени и управление контентом, где немедленная согласованность не является высшим приоритетом.

Установка и запуск

```
version: '3.2'
    services:
        mongo:
            image: mongo:5.0
            container_name: mongodb

            restart: on-failure

            volumes:
                - mongodbdata:/data/db

            ports:
                - '27017:27017'
            healthcheck:
                test: echo 'db.runCommand("ping").ok' | mongosh localhost:27017/test --quiet
    volumes:
        mongodbdata:
```

Простой пример – вставка данных

```
from pymongo import MongoClient
# Connect to MongoDB
client = MongoClient('mongodb://mongodb:27017/')
db = client['mydatabase'] # Specify the database name
collection = db['mycollection'] # Specify the collection name

document = {
'name': 'Jane Doe',
'age': 30,
'email': 'janedoe@example.com'
}
# Insert the document into the collection

result = collection.insert_one(document)
# Check if the insertion was successful
if result.acknowledged:
        print('Document inserted successfully.')
        print('Inserted document ID:', result.inserted_id)
else:
        print('Failed to insert document.')
```

Простой пример – чтение данных

```
from pymongo import MongoClient
import json

# Connect to MongoDB
client = MongoClient('mongodb://mongodb:27017/')
db = client['mydatabase'] # Specify the database name
collection = db['mycollection'] # Specify the collection name

# Retrieve documents based on specific conditions
query = {
'age': {'$gte': 29}, # Retrieve documents where age is greater than or equal to 29
}
documents = collection.find(query)
# Iterate over the retrieved documents for document in documents:
for document in documents:
                json_document = json.dumps(document, indent=2, default=str)
                print(json_document)
```

Базовые операции

CRUD

Консоль управления

```
# mongo – консоль управления (на сервере, где установлен mongodb)
show dbs # показывает базы данных
use shop # подключается (и создает если надо) базу данных

# создадим коллекцию и добавляет элементы
db.products.insertOne({name:"A Book", price: 29, details: {pages: 500}})
# выведем все данные коллекции и отформатирует
db.products.find().pretty()
```

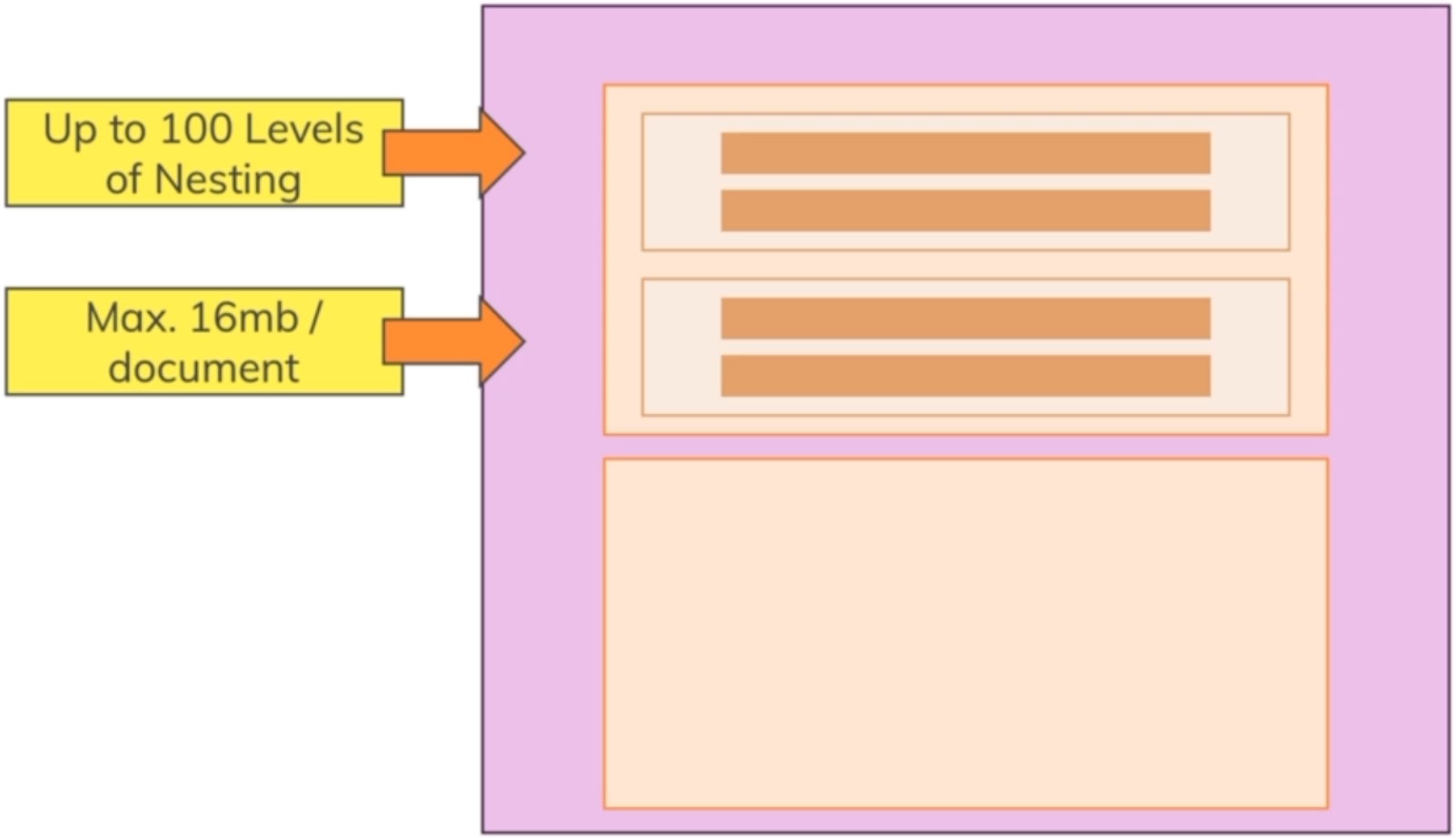
CRUD команды

```
# выведем первый элемент
db.products.findOne()

# выведем определенные данные все с name “A Book”

db.products.find({name:"A Book"}).pretty()

# выведем определенные данные все что дороже 20
db.products.find({price:{$gt:20}}).pretty()


# изменим объект updateOne / updateMany


db.products.updateOne({_id:ObjectId("65d8eae6042cdb8f763a1be1")},{$set:
{price: 1000}})

# удалим объект delete/ deleteOne
db.products.deleteOne({_id:ObjectId("65d8eae6042cdb8f763a1be1")})
```

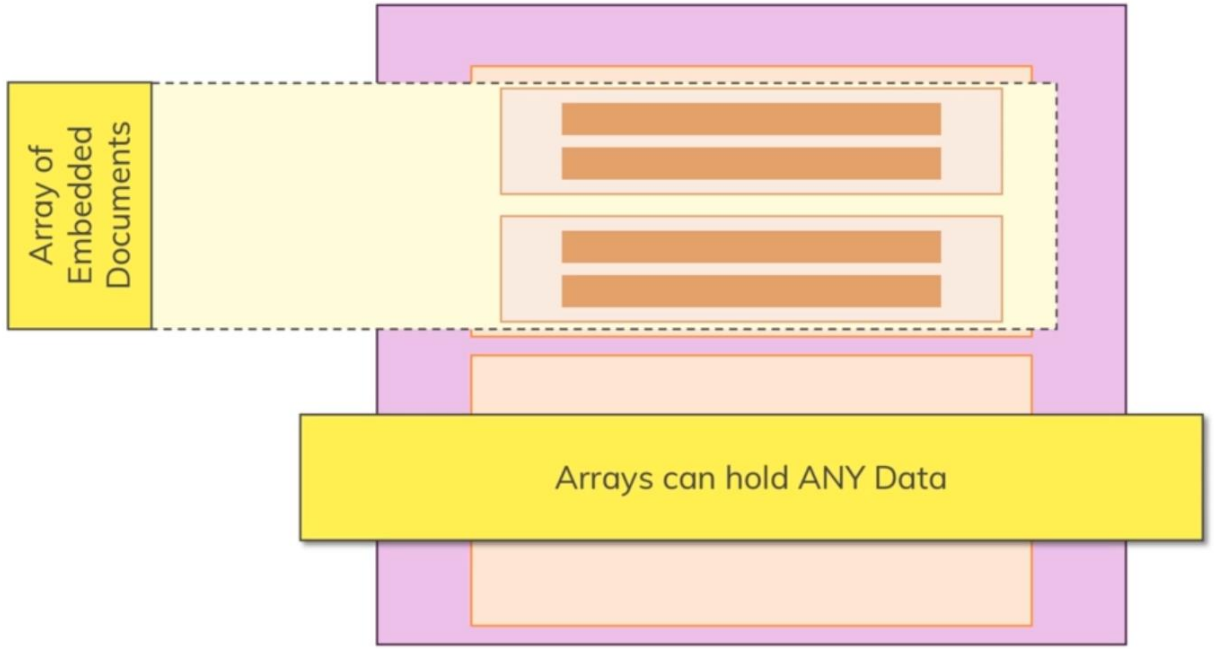
Чтение данных

```
# выведем первый элемент
db.products.findOne()
# получает крсор по которому можно перемещаться с помощью it
db.products.find({name:"A Book"}).pretty()

# возвращает сразу все данные
db.products.find({name:"A Book"}).toArray()


# применяет к каждому элементу команду
db.products.find({name:"A Book"}).forEach((prod)=>{printjson(prod)})


# возвращает нужные данные (name)
db.products.find({},{name:1}).pretty()
```

Ограничение на документы



Массивы



Ограничение:

Максимальный размер одного документа в MongoDB составляет 16 МБ (мегабайт).

MongoDB не накладывает жестких ограничений на количество элементов в массиве, но размер массива все равно влияет на общий размер документа.

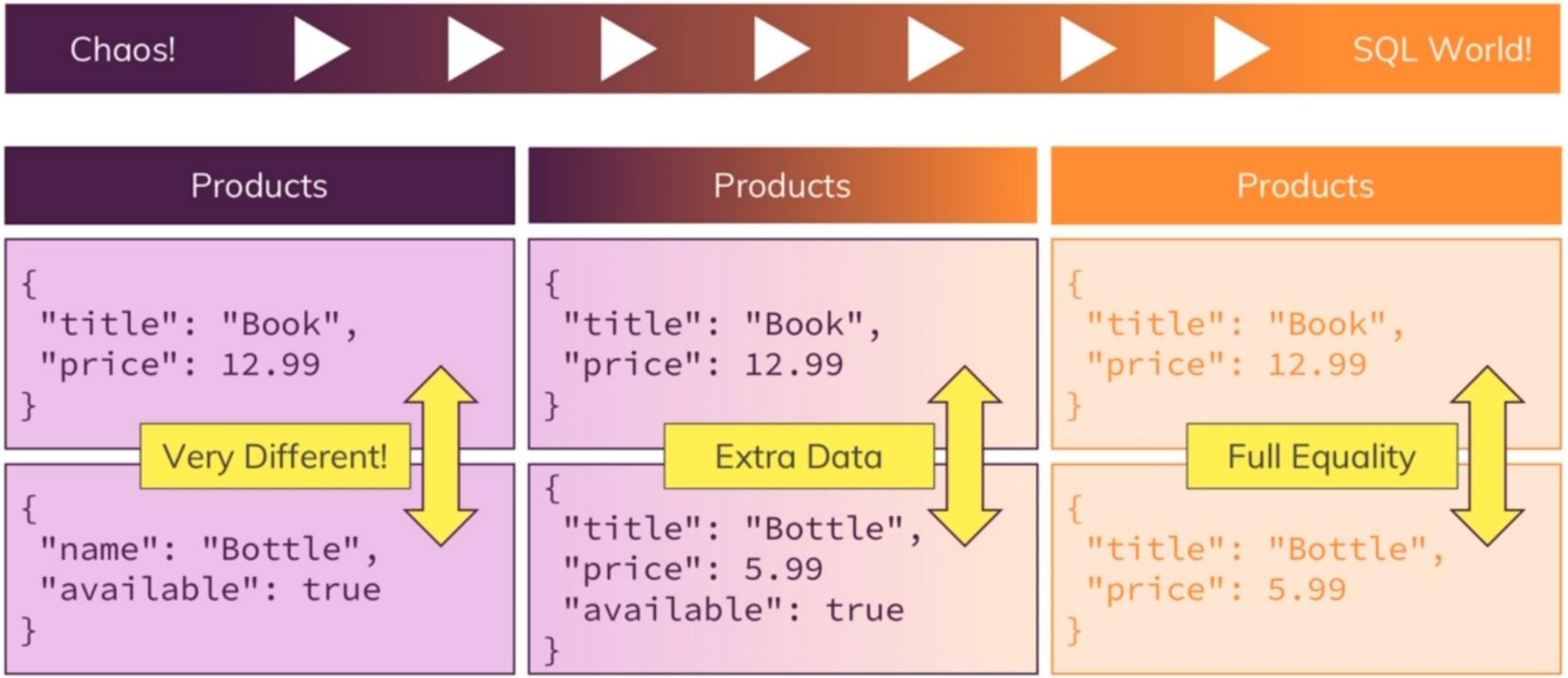
Массивы

```
db.products.insertOne({
"_id": ObjectId("60d5f4e01c9d440000a1b2c3"),
"name": "Smartphone",
"price": 599.99,
"reviews": []
});

db.products.updateOne(
  { _id: ObjectId("60d5f4e01c9d440000a1b2c3") },
  {
    $push: {
      reviews: {
        $each: [
          { "user": "Alice", "rating": 4, "comment": "Great phone!" },
          { "user": "Bob", "rating": 5, "comment": "Excellent performance!" }
        ]
      }
    }
  }
)

db.products.findOne(   { _id: ObjectId("60d5f4e01c9d440000a1b2c3") })
```

Схемы, типы и связи

MongoDB поощряет работу без схем документов

Со схемами и без



Типы данных

- **Text** - “A book”
- **Boolean** - true
- Numbers
  - **Integer** (int32)
  - **NumberLong** (int64)
  - **NumberDecimal** – 12.99
- **ObjectId** – ObjectId(“.....”)
- ISODate – **ISODate**(“2024-12-12”)
- Timestamp – **Timestamp**(12421532)
- Embeded Document – { a: {...}}
- Array – [...]

https://mongodb.github.io/node-mongodb-native/6.3/

Работаем с типами данных

```
# мегасложный объект (по умолчанию используется NodeJs driver)

db.companies.insertOne({ name: "Pche Lines Inc", isStartup: true,
employees: 33, funding: 1223124124214l, details: { ceo: "Max Steal"},
tags :[ { title: "super"} , {title : "prefect"}], foundingDate: new
Date('2014-01-01'), insertedAt:new Timestamp()})
```

Дизайн данных

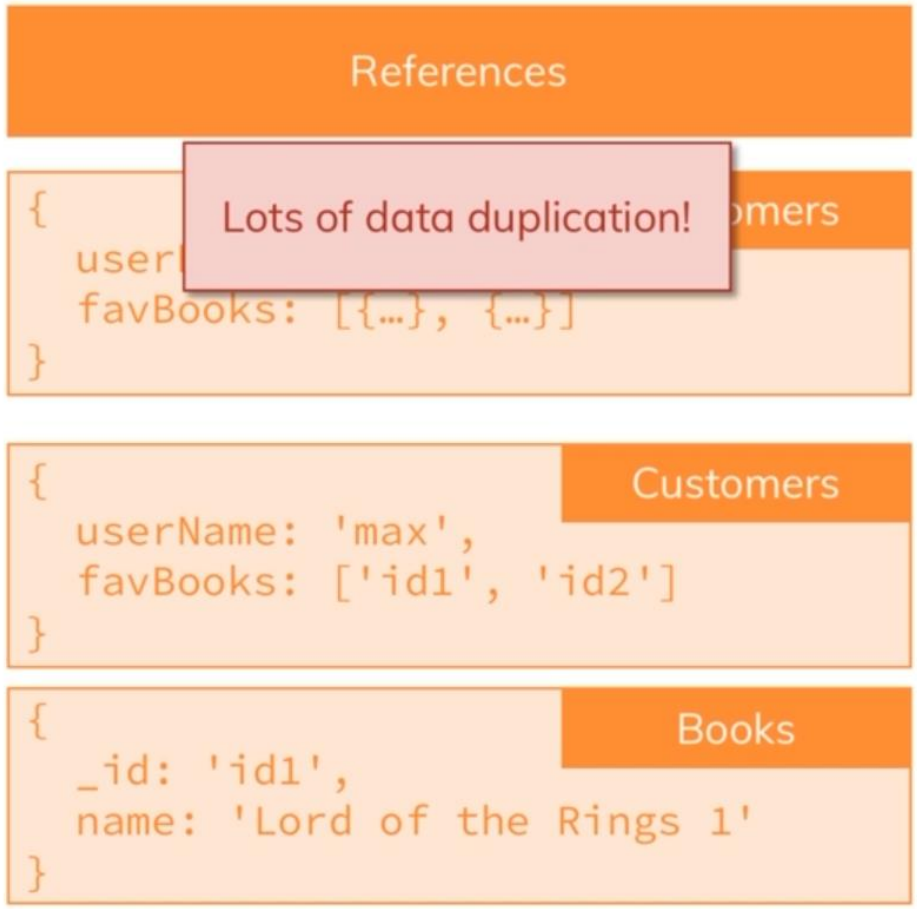
- Какие данные понадобятся моему приложению?
- В какие приложениях они понадобятся?

- Какие данные нужно будет отображать?

- Как часто я буду читать данные?

- Как я буду менять данные?

Связи
embeded documents vs references



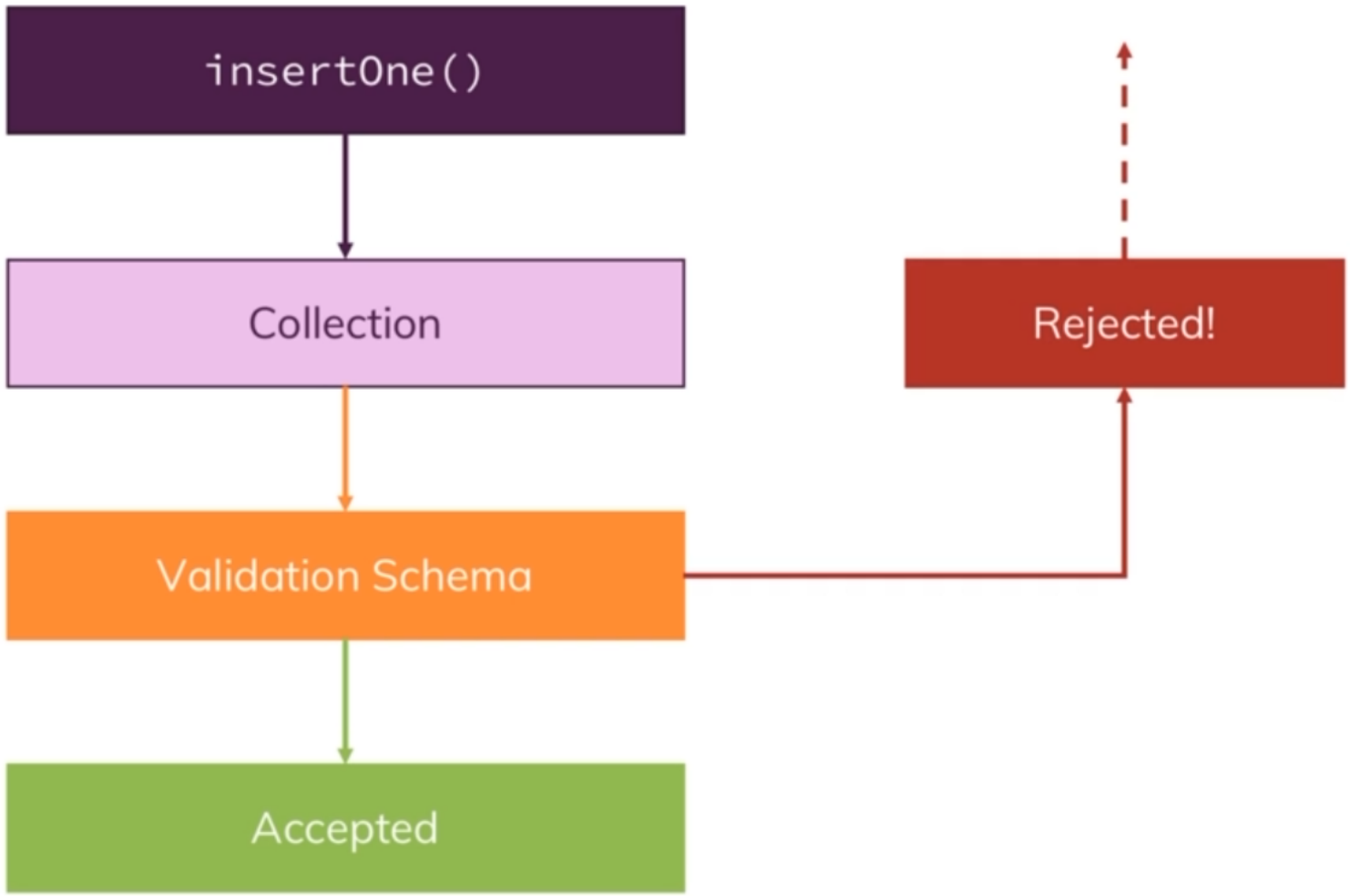
id можно задавать руками

```
> db.users.insertOne({ _id: "user-01", name: "Ivan Petrov"})
{ "acknowledged" : true, "insertedId" : "user-01" }
> db.users.findOne()
{ "_id" : "user-01", "name" : "Ivan Petrov" }
> db.orders.insertOne({ time: new Timestamp() , details : "Sone description" , user : "user-01" })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("65d9a5abf5793bcbf97d988e")
}
> db.orders.findOne()
{
        "_id" : ObjectId("65d9a5abf5793bcbf97d988e"),
        "time" : Timestamp(1708762539, 1),
        "details" : "Sone description",
        "user" : "user-01"
}
```

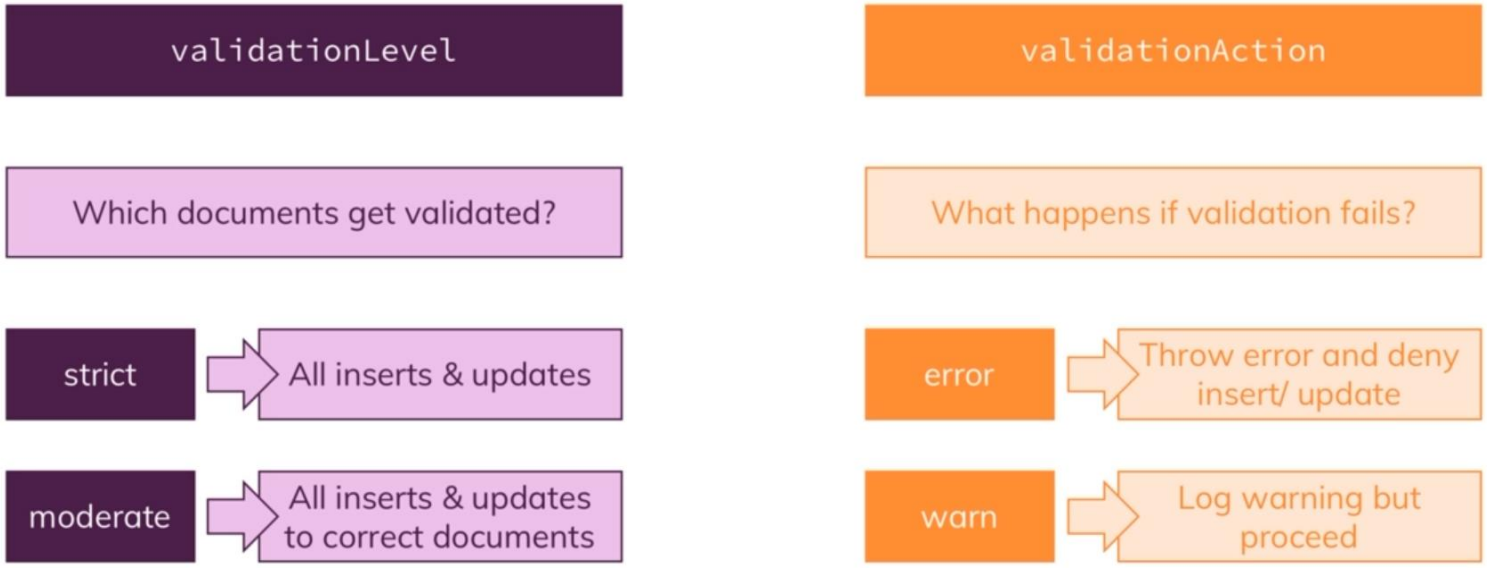
Связи - итого

Nested / Embedded Documents	References
Group data together logically	Split data across collections
Great for data that belongs together and is not really overlapping with other data	Great for related but shared data as well as for data which is used in relations and standalone
Avoid super-deep nesting (100+ levels) or extremely long arrays (16mb size limit per document)	Allows you to overcome nesting and size limits (by creating new documents)

Валидация схемы



Способы валидации



Использование схем

```
db.createCollection('posts', {
  validator: {
    $jsonSchema: {
      bsonType: 'object',
      required: ['title', 'text', 'creator', 'comments'],
      properties: {
        title: {
          bsonType: 'string',
          description: 'must be a string and is required'
        },
        text: {
          bsonType: 'string',
          description: 'must be a string and is required'
        },
        creator: {
          bsonType: 'objectId',
          description: 'must be an objectid and is required'
        },
        comments: {
          bsonType: 'array',
          description: 'must be an array and is required',
          items: {
            bsonType: 'object',
            required: ['text', 'author'],
            properties: {
              text: {
                bsonType: 'string',
                description: 'must be a string and is required'
              },
              author: {
                bsonType: 'objectId',
                description: 'must be an objectid and is required'
              }
            }
          }
        }
      }
    }
  }
});
```

Валидация

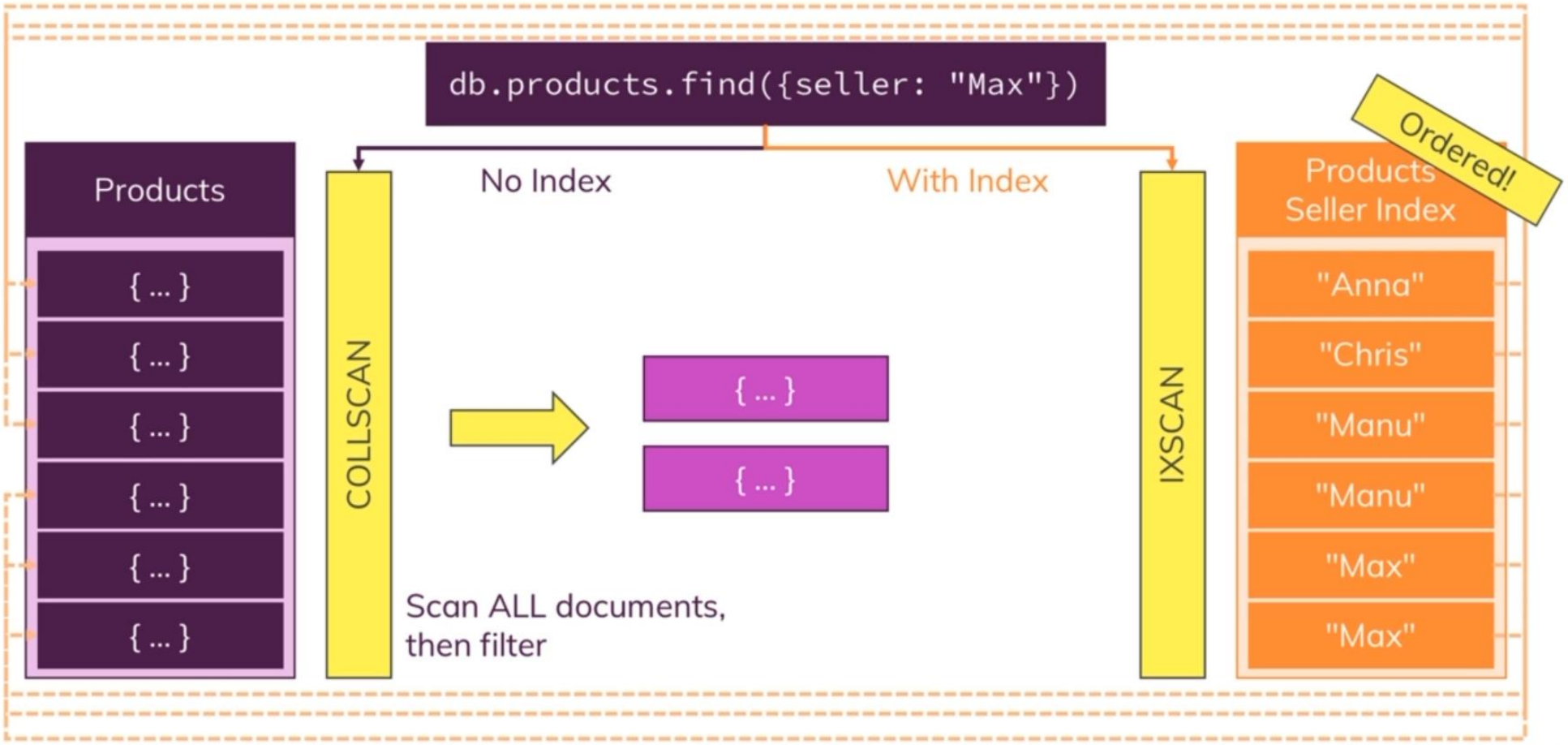
```
> db.posts.insertOne({title: "Some title", text: "Some text",
creator: ObjectId("65d9ac4ff5793bcbf97d988f"), comments: [{text:
"Coool!" , author: ObjectId("65d9ac4ff5793bcbf97d988f")}]})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("65d9acbcf5793bcbf97d9890")
}
> db.posts.insertOne({title: "Some title", text: "Some text",
comments: [{text: "Coool!" , author:
ObjectId("65d9ac4ff5793bcbf97d988f")}]})
WriteError({ ….
```

Можно поменять на «предупреждение»

```
db.runCommand({
  collMod: 'posts',
  validator: {
    $jsonSchema: { ....
    }
  },
  validationAction: 'warn'
});
```

Индексы в MongoDB

Поиск по индексам



Заполним данными mongodb

```
mongoimport /opt/05_mongo/persons.json -d
arch -c contacts --jsonArray
```

Запросы

```
> db.contacts.find({"dob.age" : { $gt: 60}}).pretty()
# все старше 60

> db.contacts.explain().find({"dob.age" : { $gt: 60}})
# анализ запроса winning plan – COLLSCAN

> db.contacts.explain("executionStats").find({"dob.age" : { $gt: 60}})
# executionTimeMillis" : 6 (mlsc)
# totalDocsExamined" : 5000
```

Параметры запросов

- **$eq** (равно)
- **$ne** (не равно)
- **$gt** (больше чем)
- **$lt** (меньше чем)
- **$gte** (больше или равно)
- **$lte** (меньше или равно)
- **$in** определяет массив значений, одно из которых должно иметь поле документа
- **$nin** определяет массив значений, которые не должно иметь поле документа

Создаем индекс

```
> db.contacts.createIndex({"dob.age": -1}) # параметр указывает на порядок сортировки
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
> db.contacts.explain("executionStats").find({"dob.age" : { $gt: 60}})
# "stage" : "IXSCAN" , "stage" : "FETCH"
# вначале ищет указатели на документы, потом их достает из коллекции
# executionTimeMillis" : 2 (mlsc)
# totalDocsExamined" : 1222
```

! плохой индекс – только мешает, поскольку данные будут запрашиваться не в один шаг, а в два

Составной индекс

```
> db.contacts.dropIndex({"dob.age": -1}) # удалим индекс
> db.contacts.createIndex({"dob.age": 1, gender: 1}) # "dob.age_1_gender_1"
> db.contacts.explain("executionStats").find({"dob.age" : 35, "gender" : "male" })
> db.contacts.explain("executionStats").find({"dob.age" : { $gt: 20}}) # использует
> db.contacts.explain("executionStats").find({"gender" : "male"}) # не использует
> db.contacts.getIndexes() # возвращает перечень индексов
```

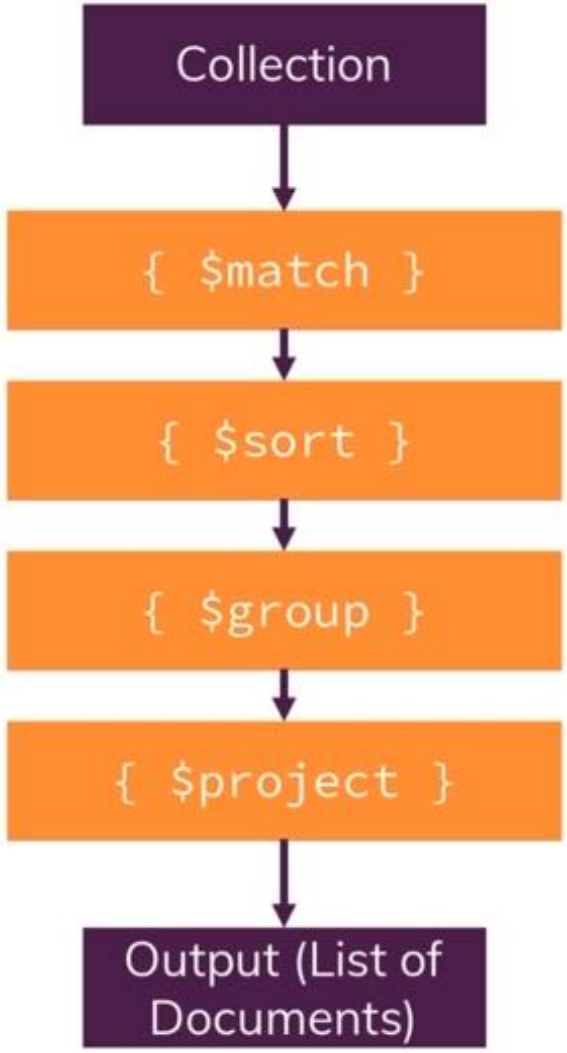
Text Indexes

- текстовый индекс убирает «стоп» слова (a, the ...)
- может быть один на коллекцию

Text Indexes

```
> db.contacts.createIndex({"location.street": "text"})

> db.contacts.explain("executionStats").find({$text : {$search : "road"}})

> db.contacts.find({$text : {$search : "road"}}).count()
# по одному слову

> db.contacts.find({$text : {$search : "road street"}}).count()
# по вхождению одного из слов

> db.contacts.find({$text : {$search : "\"road street\""}}).count()
# по вхождению фразы
```

Aggregation Framework



https://www.mongodb.com/docs/manual/core/aggregation-pipeline/

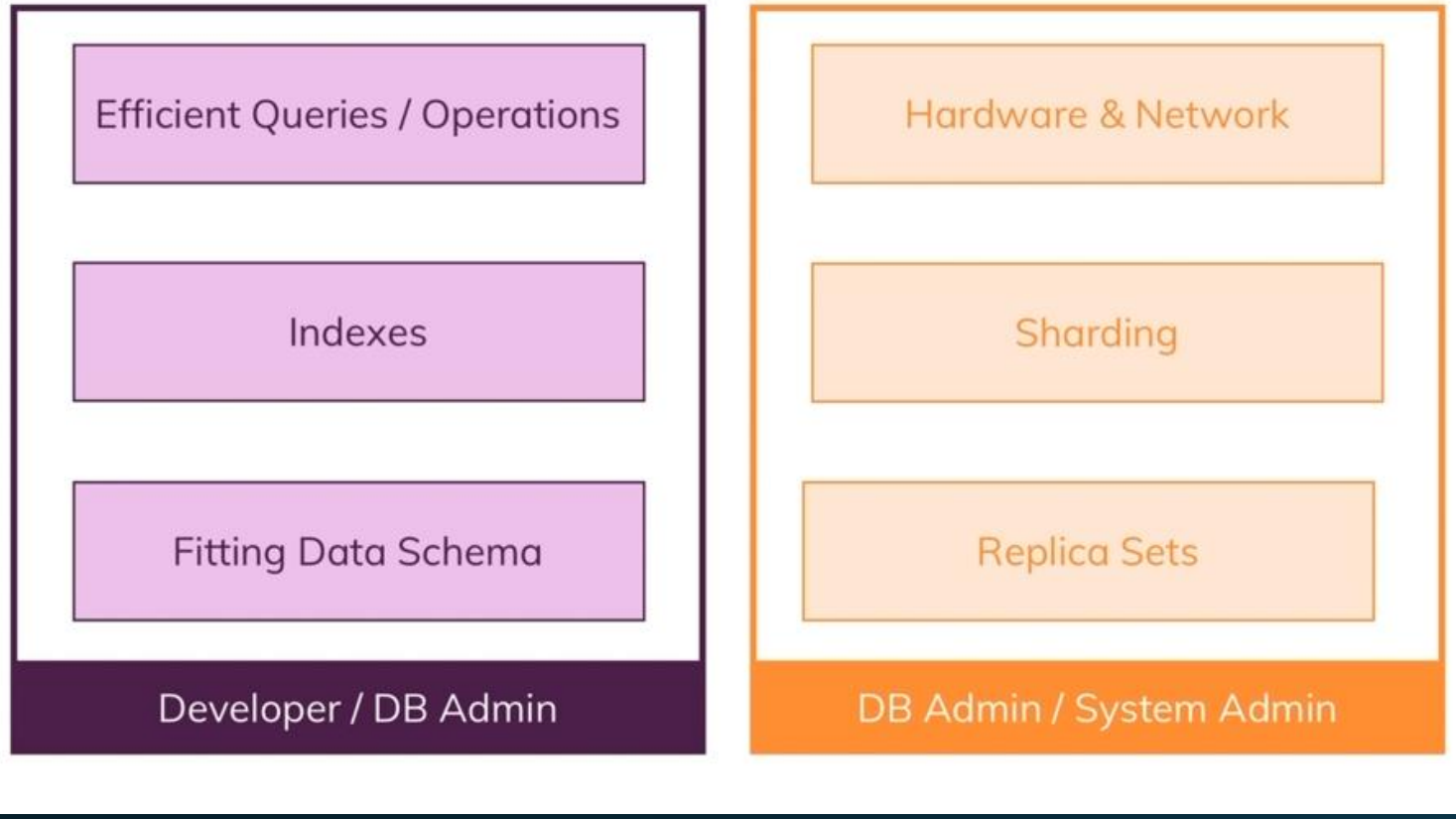
Пример – выборка и группировка

```
> db.contacts.aggregate([{$match : {gender : "male"} }, {$group : { _id:"$nat" , totalAge :
{$avg : "$dob.age"}}}, { $sort : { totalAge : -1} }])

{ "_id" : "DK", "totalAge" : 49.32592592592592 }
{ "_id" : "IE", "totalAge" : 48 }
{ "_id" : "NZ", "totalAge" : 47.962406015037594 }
{ "_id" : "BR", "totalAge" : 47.93617021276596 }
{ "_id" : "TR", "totalAge" : 47.776978417266186 }
{ "_id" : "AU", "totalAge" : 47.75974025974026 }
{ "_id" : "CA", "totalAge" : 47.66451612903226 }
{ "_id" : "GB", "totalAge" : 47.523809523809526 }
{ "_id" : "DE", "totalAge" : 47.294520547945204 }
{ "_id" : "ES", "totalAge" : 47.276595744680854 }
{ "_id" : "IR", "totalAge" : 46.93377483443709 }
{ "_id" : "FI", "totalAge" : 46.71974522292994 }
{ "_id" : "FR", "totalAge" : 46.66013071895425 }
{ "_id" : "CH", "totalAge" : 46.553333333333335 }
{ "_id" : "NO", "totalAge" : 46.51034482758621 }
{ "_id" : "NL", "totalAge" : 45.2 }
{ "_id" : "US", "totalAge" : 45.0234375 }
```

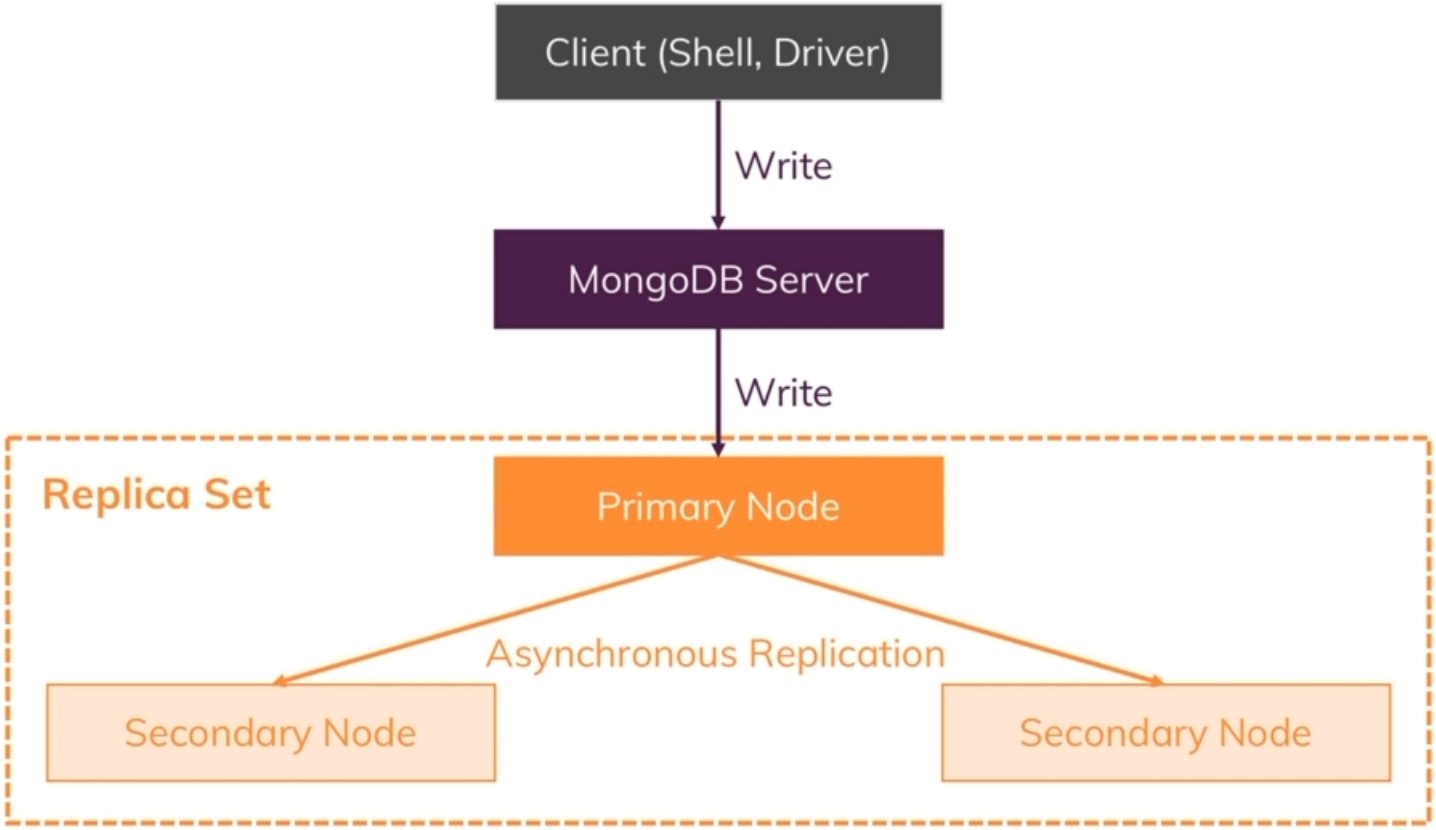
Запросы $lookup

```
> db.orders.aggregate([{ $lookup: { from: "users" , localField: "user" , foreignField : "_id" , as: "order_users"}}]).pretty()

{
        "_id" : ObjectId("65d9a5abf5793bcbf97d988e"),
        "time" : Timestamp(1708762539, 1),
        "details" : "Sone description",
        "user" : "user-01",
        "order_users" : [
                {
                        "_id" : "user-01",
                        "name" : "Ivan Petrov"
                }
        ]
}
```

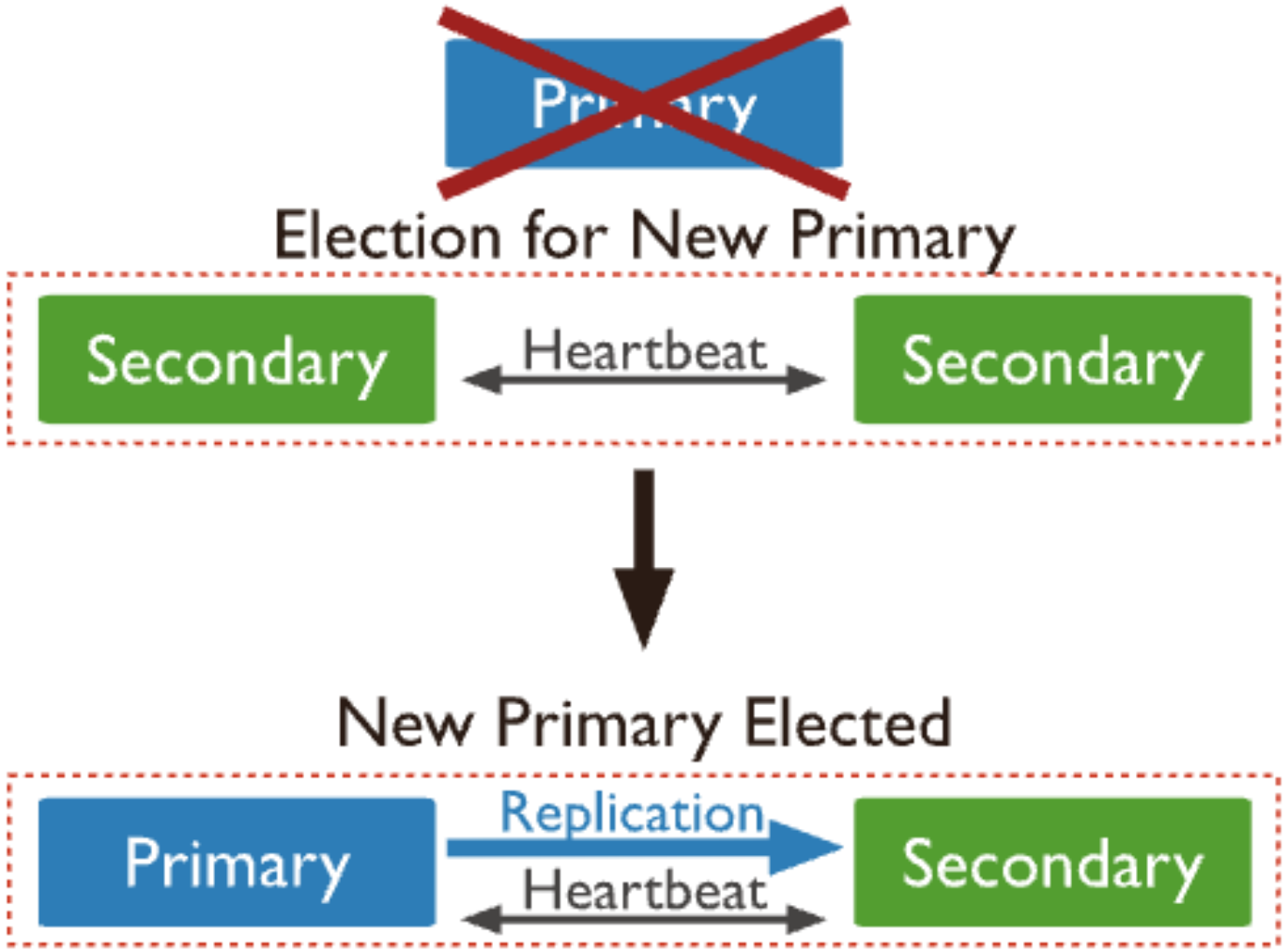
Performance & Fault Tolerance



Replica Set


Client (Shell, Driver)
Write
MongoDB Server
Write
Replica Set
Primary Node
Asynchronous Replication
Secondary Node
Secondary Node

Replica Set



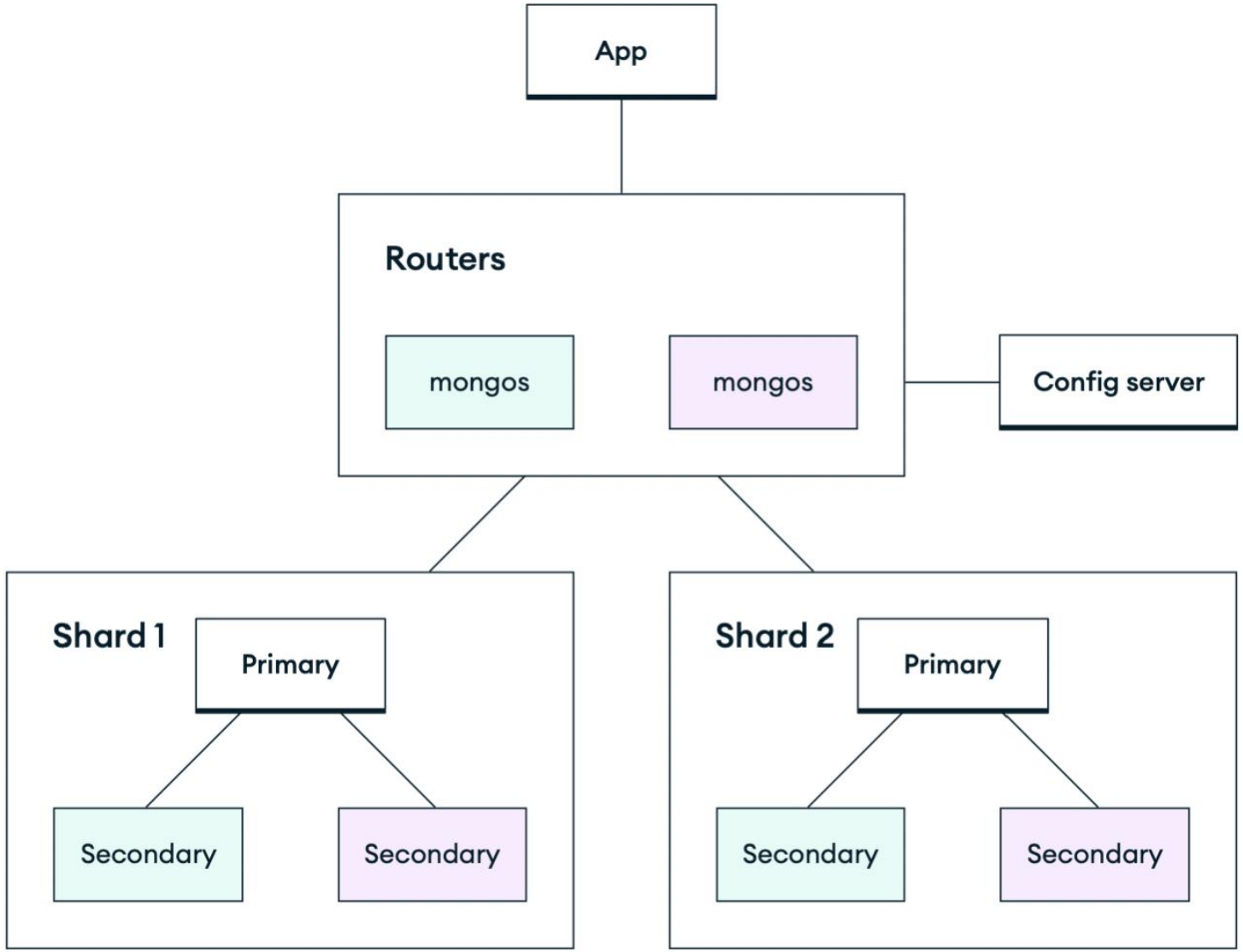
https://www.mongodb.com/docs/manual/core/replica-set-elections/

Пример (example_02)

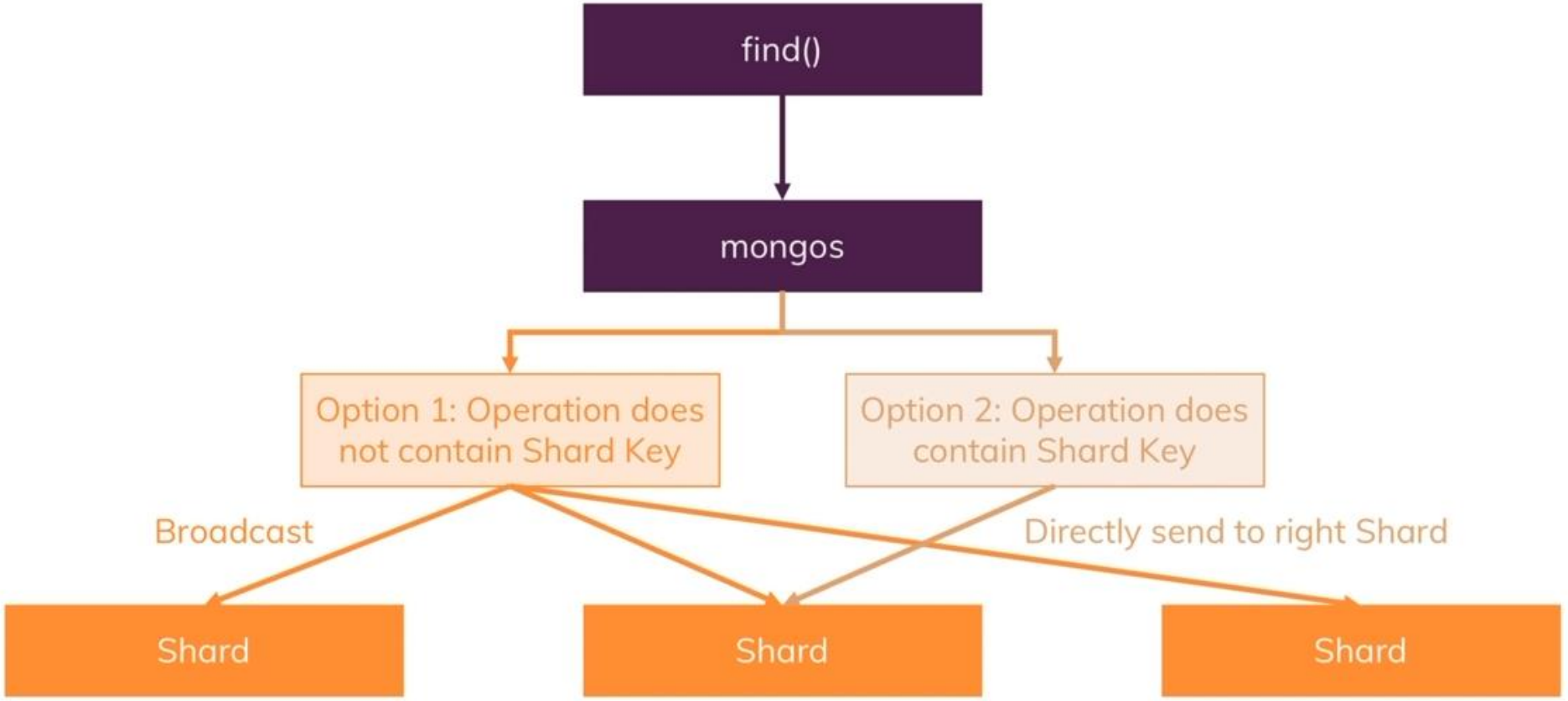
```
> rs.status()
# выведет состояние replica- set
```

```
mongosh --host mongo1:27017 <<EOF
var cfg = {
        "_id": "myReplicaSet",
        "version": 1,
        "members": [
                {
                        "_id": 0,
                        "host": "mongo1:27017",
                        "priority": 2
                },
                {
                        "_id": 1,
                        "host": "mongo2:27017",
                        "priority": 0
                },
                {
                        "_id": 2,
                        "host": "mongo3:27017",
                        "priority": 0
                }
        ]
};
rs.initiate(cfg);
EOF
```

Sharding


App
Routers
mongos
mongos
Config server
Shard 1
Primary
Secondary
Secondary
Shard 2
Primary
Secondary
Secondary

Два варианта запроса



Ссылки

- Сравнение SQL и NoSQL
https://www.mongodb.com/nosql-explained/nosql-vs-sql
-
Спецификация MQL
https://www.mongodb.com/docs/manual/tutorial/query-documents/