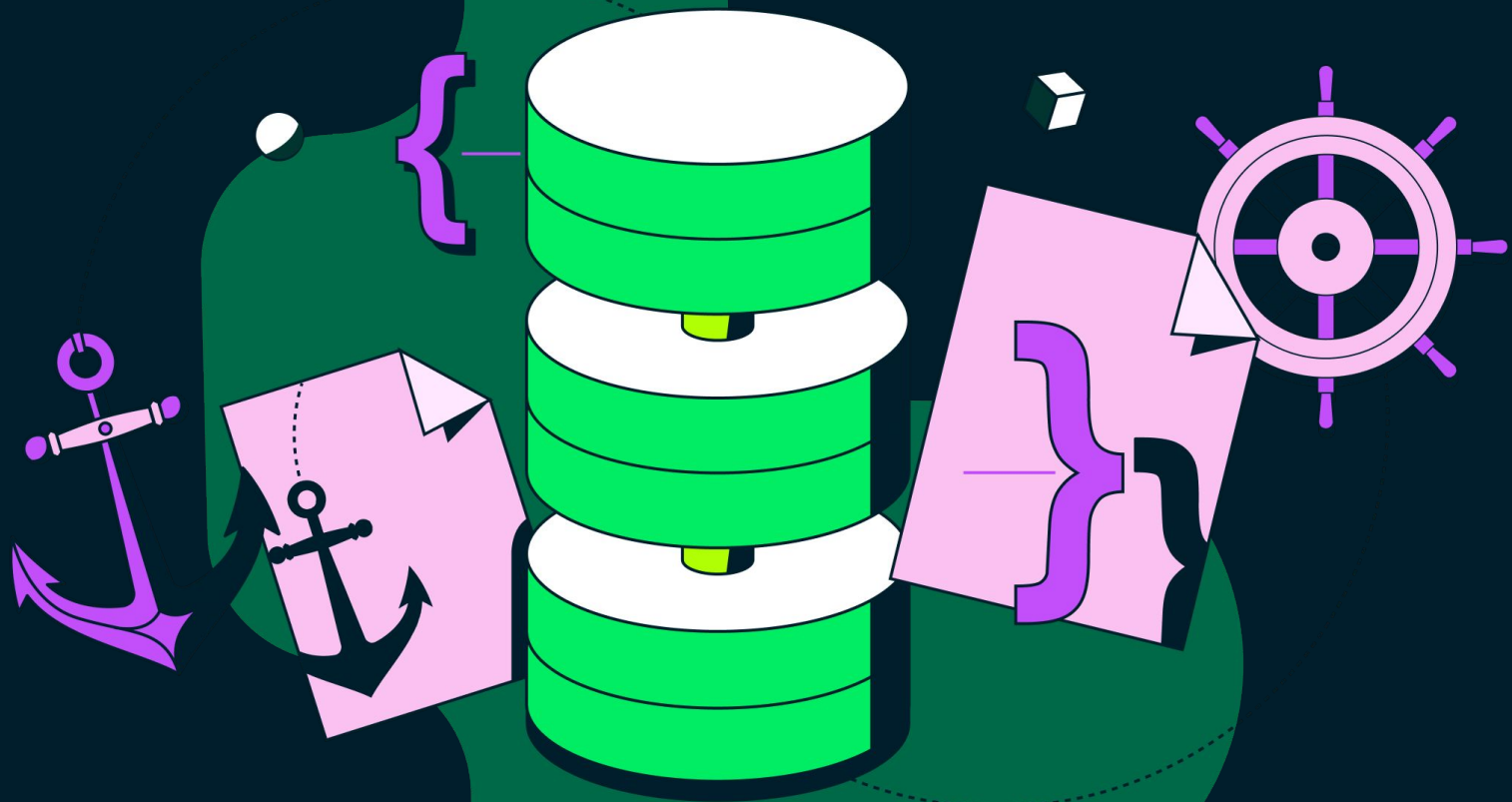




MongoDB®

# ¿Qué es **MongoDB**?







**Nicolas Molina**  
@nicobytes

# Curso de **Modelado de Datos** con MongoDB





 MongoDB®



Experts



**Microsoft®**  
Most Valuable  
Professional





Platzi

# Bases de datos **NoSQL**



# Documentales

---

En estas bases de datos se empareja cada clave con una estructura de datos compleja que se denomina 'documento'.





Cloud Firestore



**Couchbase**

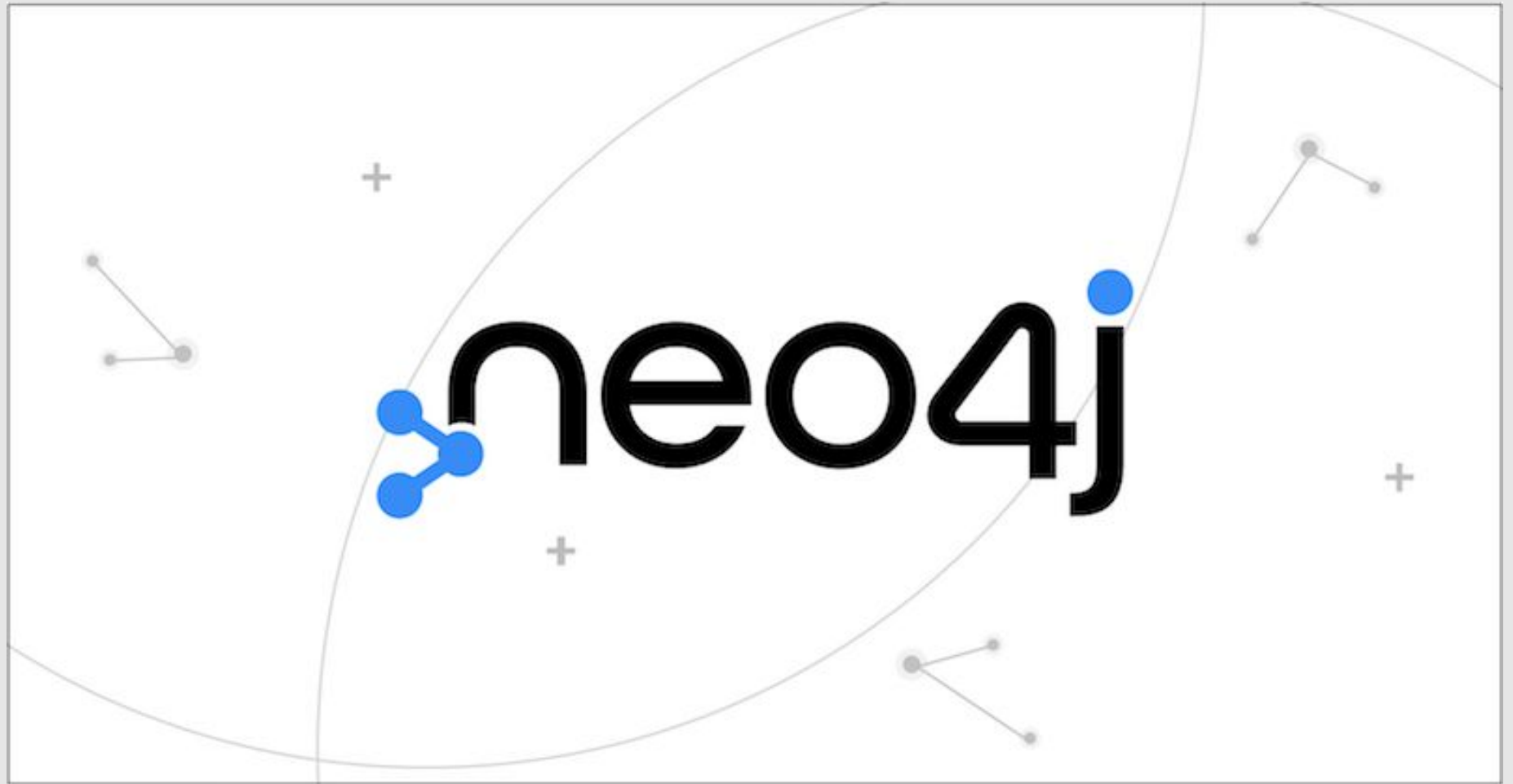


# Grafos

---

Se utilizan para almacenar información sobre redes de datos, como las conexiones sociales.



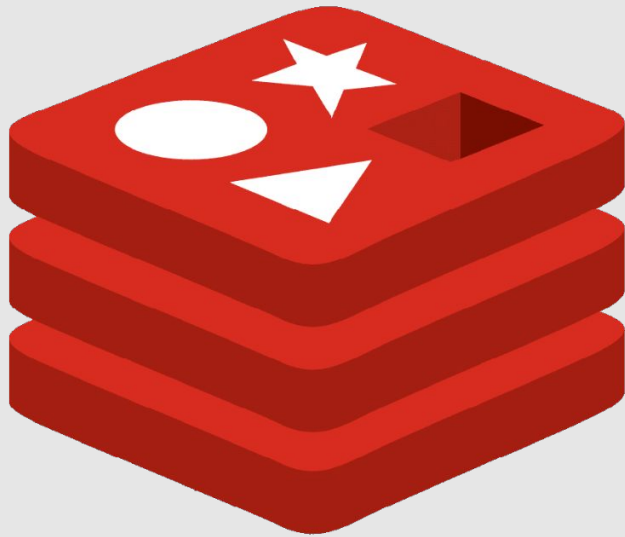


# Clave - Valor

---

Son las bases de datos **NoSQL** más simples. Cada elemento de la base de datos se almacena como un nombre de atributo (o «clave»), junto con su valor.





# redis

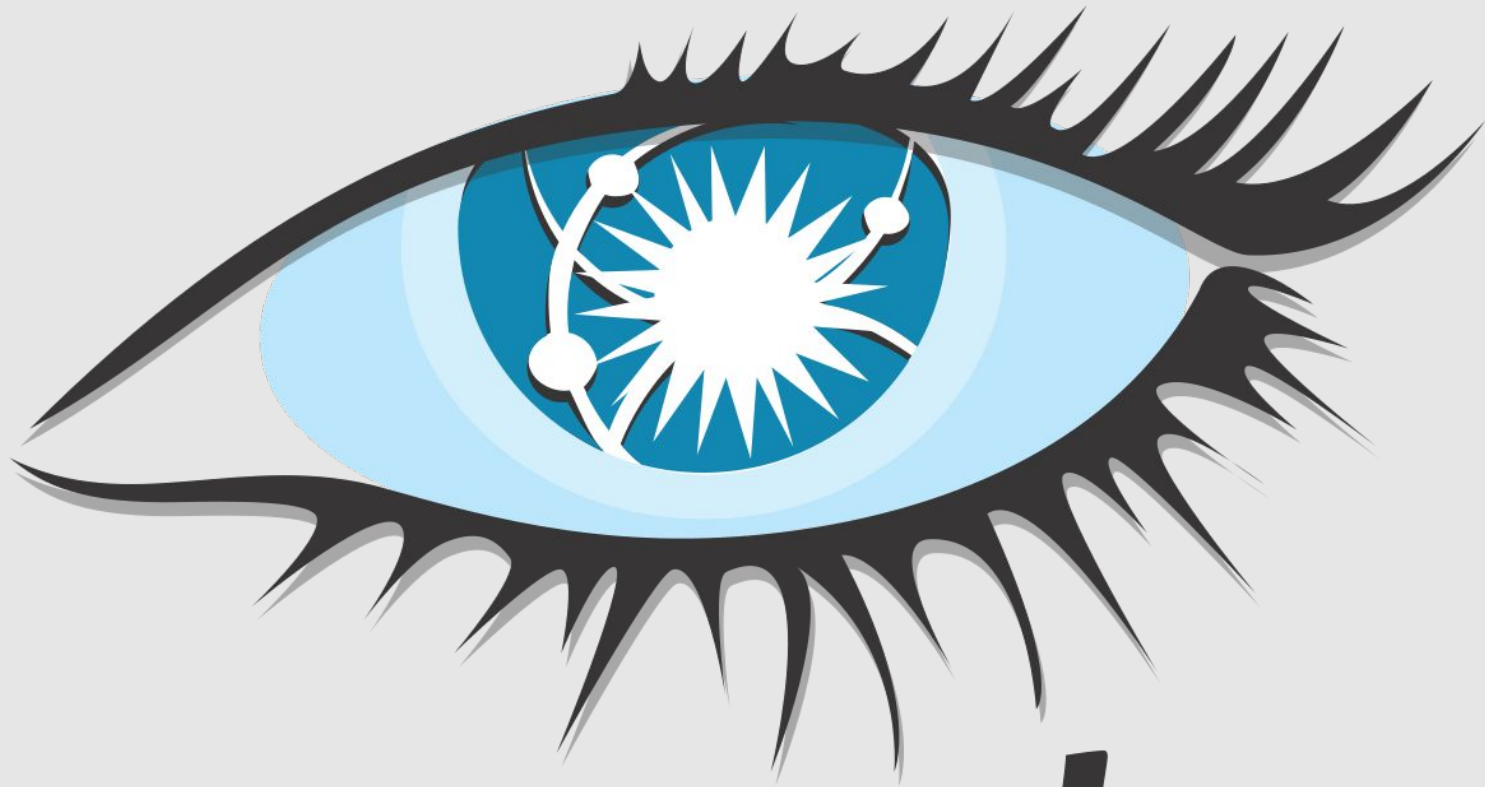


# Orientadas a Columnas

---

Estas bases de datos, como Cassandra o HBase, permiten realizar consultas en grandes conjuntos de datos y almacenan los datos en columnas, en lugar de filas.



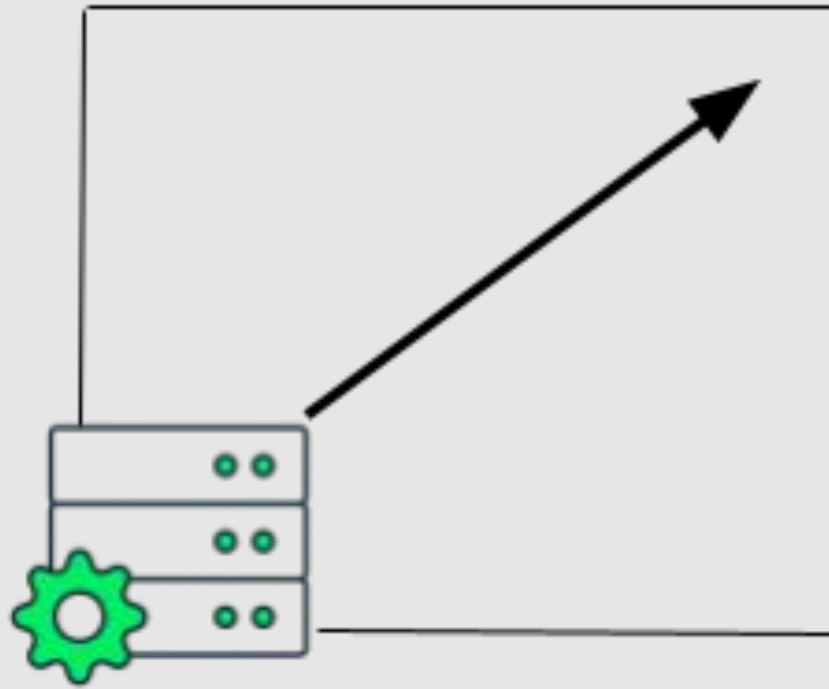


*cassandra*



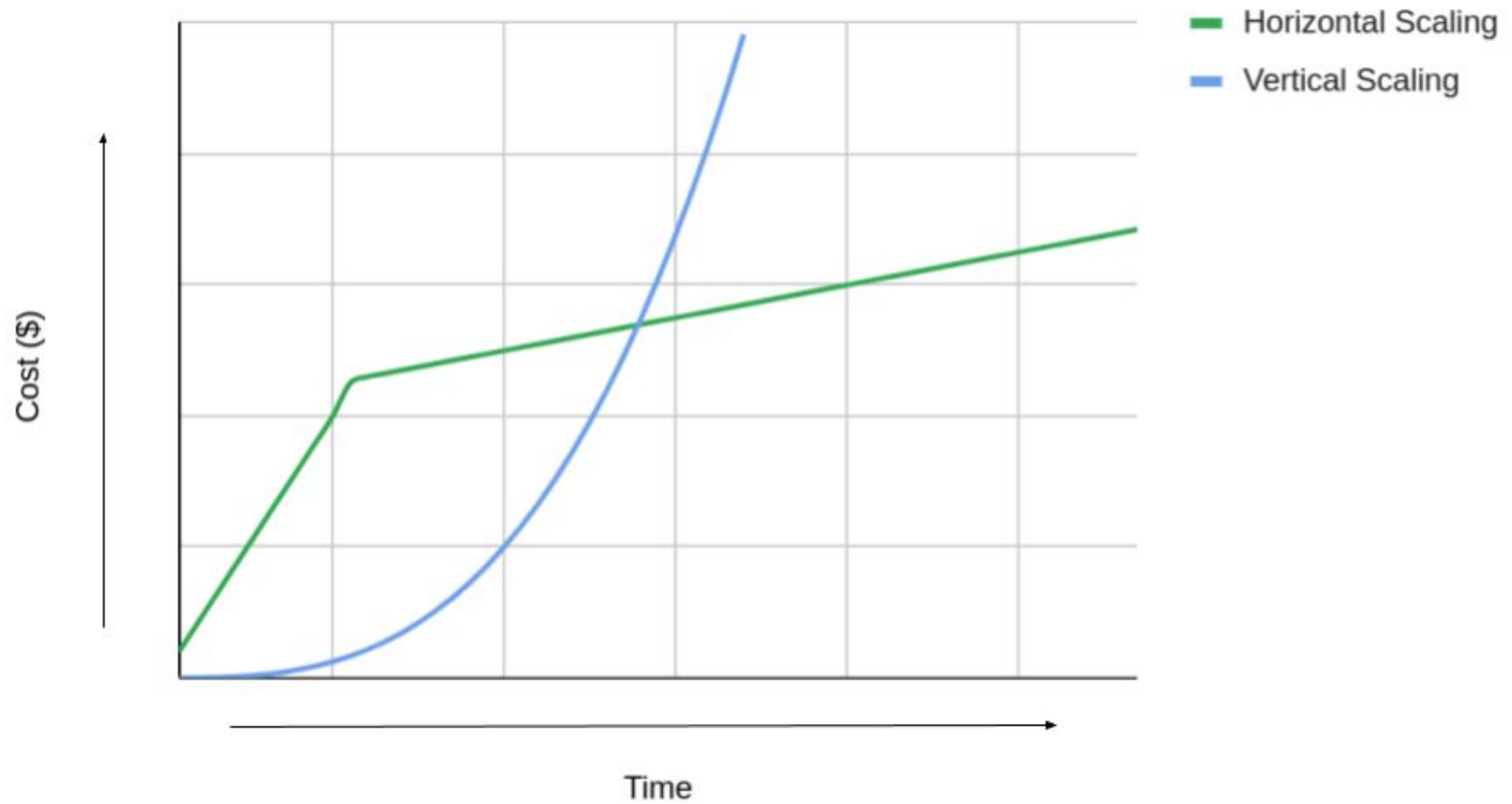
# Escalamiento horizontal





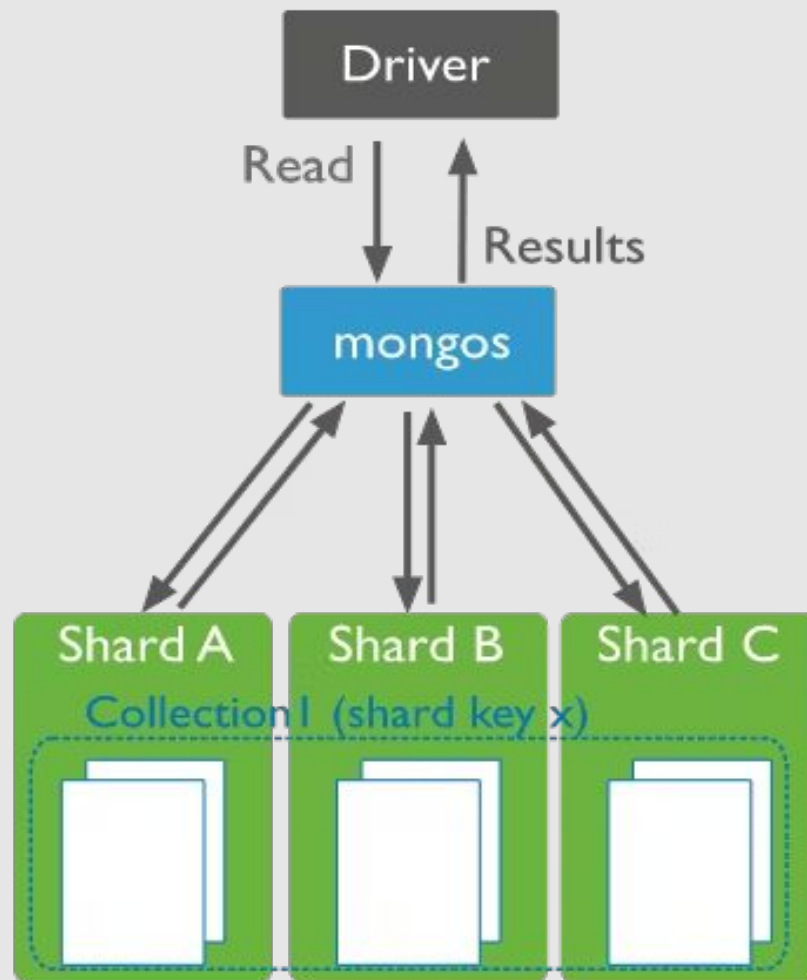


## Horizontal Scaling and Vertical Scaling



# ¿Qué es la Replicación?







# ¿Qué son los documentos?





# Document

---

Una forma de organizar y almacenar información con un conjunto de pares **clave-valor**.



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

← field: value  
← field: value  
← field: value  
← field: value

```
{  
  "_id": "5c8eccc1caa187d17ca6ed16",  
  "city": "ALPINE",  
  "zip": "35014",  
  "loc": {  
    "y": 33.331165,  
    "x": 86.208934  
  },  
  "pop": 3062,  
  "state": "AL"  
}
```

```
{  
  "_id": "5c8eccc1caa187d17ca6ed16",  
  "city": "ALPINE",  
  "zip": "35014",  
  "loc": {  
    "y": 33.331165,  
    "x": 86.208934  
  },  
  "pop": 3062,  
  "state": "AL"  
}
```

```
{  
  "_id": "5c8eccc1caa187d17ca6ed16",  
  "city": "ALPINE",  
  "zip": "35014",  
  "loc": {  
    "y": 33.331165,  
    "x": 86.208934  
  },  
  "pop": 3062,  
  "state": "AL"  
}
```







# ¿Qué son las colecciones?



# Collection

---

**MongoDB** almacena documentos en una colección, usualmente con campos comunes entre sí.



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

```
{
```

```
  name:  
  age:  
  status:  
  groups:
```

```
}
```

```
{
```

```
  name:  
  age:  
  status:  
  groups:
```

```
}
```

```
{
```

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

```
}
```

## Collection

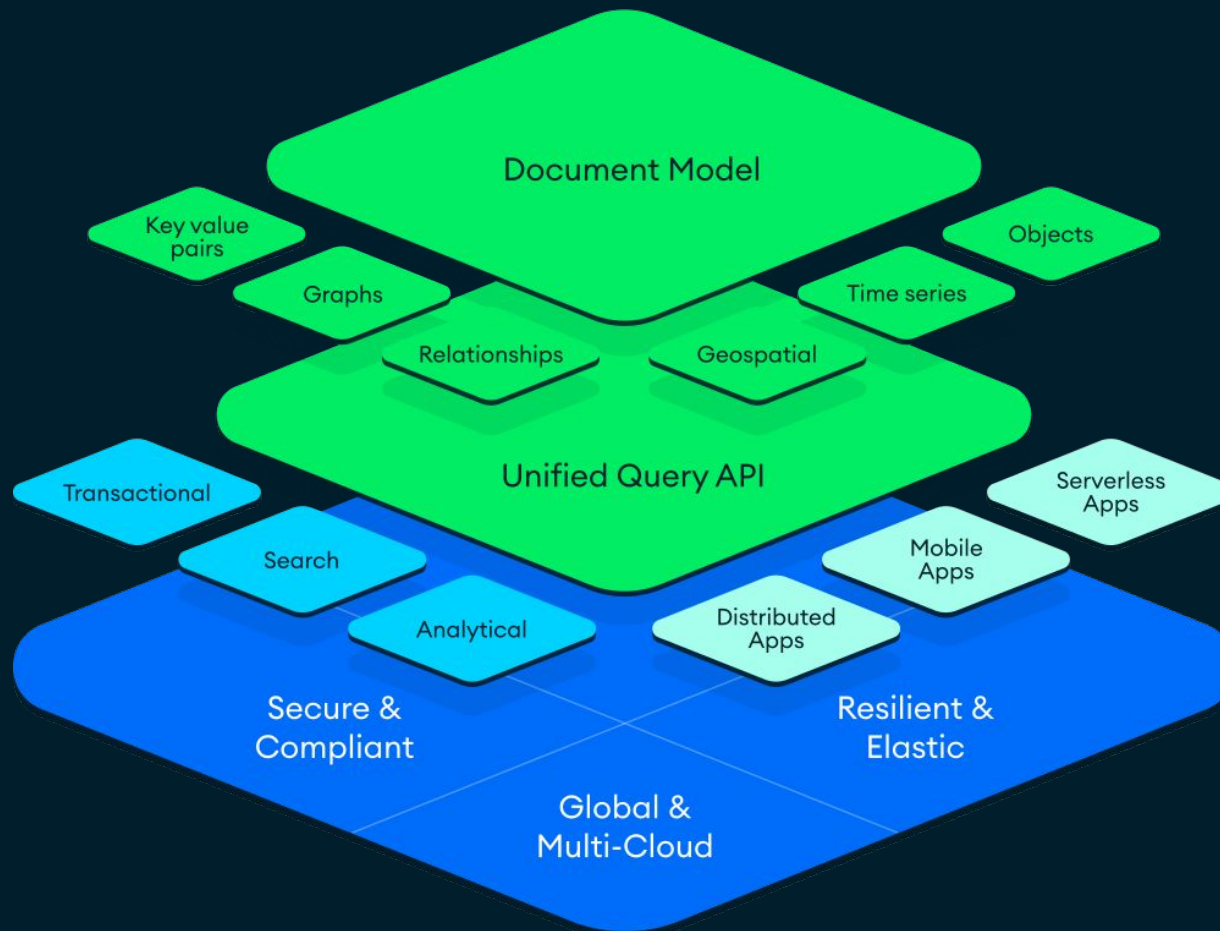


# Crea tu primer DB NoSQL



# ¿Qué es Mongo Atlas?





# Free Cluster

---

- 512MB to 5GB of storage.
- Shared RAM.
- Upgrade to dedicated clusters for full functionality.
- No credit card required to start.





# Usando **Mongo** **Compass**





# MongoDB en VSCode



// MongoDB Playground - Test

```
1 // Select the database to use.
2 use ["test"];
3
4 // Insert a few documents into the sales collection
5 db.sales.insertMany([
6   { "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 4,
7     "date" : new Date ("2021-03-01") },
8   { "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1,
9     "date" : new Date ("2021-04-01") },
10  { "_id" : 3, "item" : "abc", "price" : 10, "quantity" : 8,
11    "date" : new Date ("2021-05-01") },
12  ]);
13
14 // Build an aggregation to view total sales
15 const aggregation = [
16   { $match: { date: { $gte: new Date("2022-01-01"),
17     $lt: new Date("2022-01-01") } } },
18   { $group: { _id: "$item", totalSaleAmount: { $sum:
19     [ "$price", "$quantity" ] } } },
20 ];
21 db.sales.aggregate(aggregation)
```

{ Playground Result

```
1 {
2   "_id": "abc",
3   "totalSaleAmount": 120
4 },
5 {
6   "_id": "jkl",
7   "totalSaleAmount": 20
8 }
```



# MongoDB corriendo en Docker



# Conectándonos usando **mongosh**



# JSON vs. BSON



# Ventajas de **JSON**

---

- Amigable.
- Se puede leer.
- Es un formato muy usado.



# Desventajas de **JSON**

---

- Basado en texto.
- Consume mucho espacio.
- Es limitado: string, boolean, number, arrays.

```
{  
  "_id": "5c8eccc1caa187d17ca6ed16",  
  "city": "ALPINE",  
  "zip": "35014",  
  "loc": {  
    "y": 33.331165,  
    "x": 86.208934  
  },  
  "pop": 3062,  
  "state": "AL"  
}
```

# Ventajas de **BSON**

---

- Representación binaria de **JSON**.
- No consume espacio.
- Alto rendimiento.
- Tipos de datos: +, date, raw binary, integer, long, float.

# Desventajas de **BSON**

---

- No es estándar.
- Es un formato para la máquina.

```
{  
  "hello" : "world"  
}
```

\x16\x00\x00\x00	// total document size
\x02	// 0x02 = type String
hello\x00	// field name
\x06\x00\x00\x00world\x00	// field value
\x00	// 0x00 = end of object



# Insertando un documento



# Insertando varios documentos





# Actualizando un documento



# Operators

---

- **\$set**  
Sets the value of a field in a document.
- **\$inc**  
Increments the value of the field by the specified amount.
- **ObjectId**  
Function to find a doc with objID

# Actualizando varios documentos



# Operators

---

- **\$set**  
Sets or create the value of a field in a document.
- **\$rename**  
Renames a field.
- **\$unset**  
Removes the specified field from a document.

# Array Update Operators



# Operators

---

- **\$push**  
Adds an item to an array.
- **\$pull**  
Removes all array elements that match a specified query.
- **\$in**  
Matches any of the values specified in an array.

# Insert or Update?



# Operators

---

- **\$pop**  
Removes the first or last item of an array.



# Eliminando documentos



# Comparison Query Operators



# Usando \$eq & \$ne



# Operators

---

- **\$eq**  
Matches values that are equal to a specified value.
- **\$ne**  
Matches all values that are not equal to a specified value.

# Usando `$gt`, `$gte`, `$lt` y `$lte`



# Operators

---

- **\$gte**  
Matches values that are greater than or equal to a specified value.
- **\$gt**  
Matches values that are greater than a specified value.

# Operators

---

- **\$lte**

Matches values that are less than or equal to a specified value.

- **\$lt**

Matches values that are less than a specified value.

# Otros ejemplos con \$gt, \$gte, \$lt y \$lte



# Usando \$regex



# Operators

---

- **\$regex**

Allows use of aggregation expressions within the query language.

# MongoDB Atlas Search





# Projection



# Operadores para **Arrays**



# Operators

---

- **\$in**  
Matches any of the values specified in an array.
- **\$nin**  
Matches none of the values specified in an array.

# Operators

---

- **\$all**

Matches arrays that contain all elements specified in the query.

- **\$elemMatch**

Selects documents if element in the array field matches all the specified \$elemMatch conditions.



# Operators

---

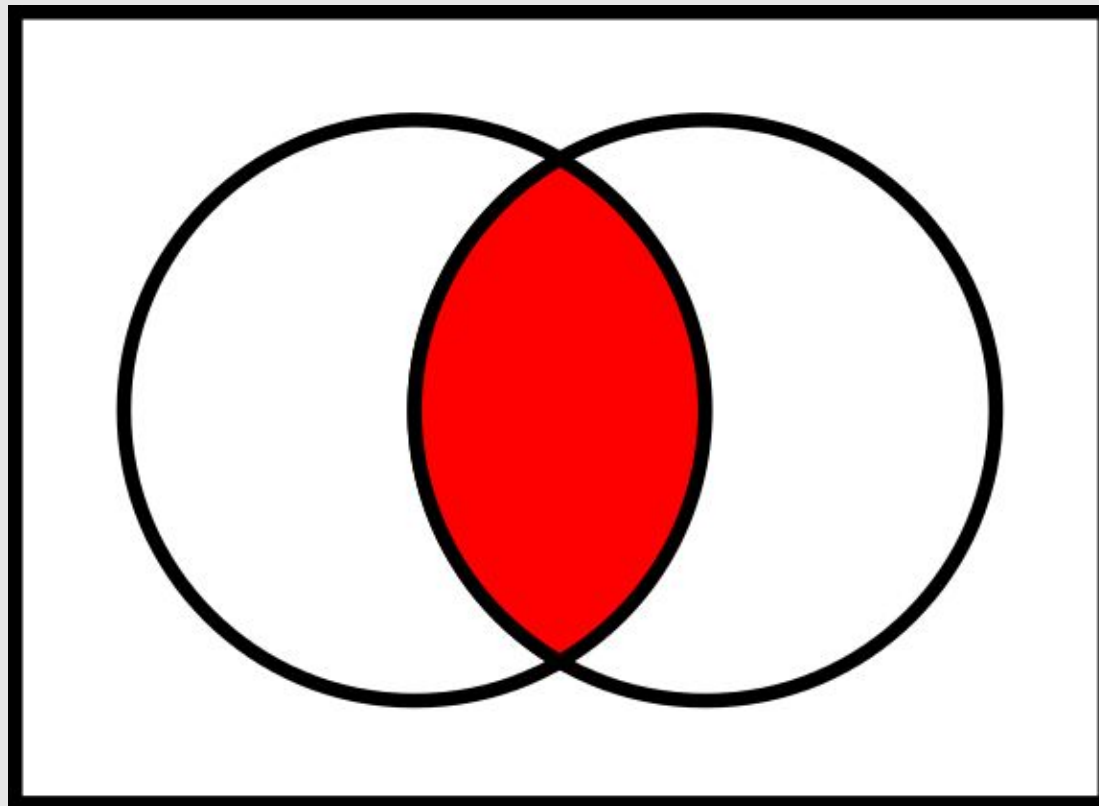
- **\$size**

Selects documents if the array field is a specified size.

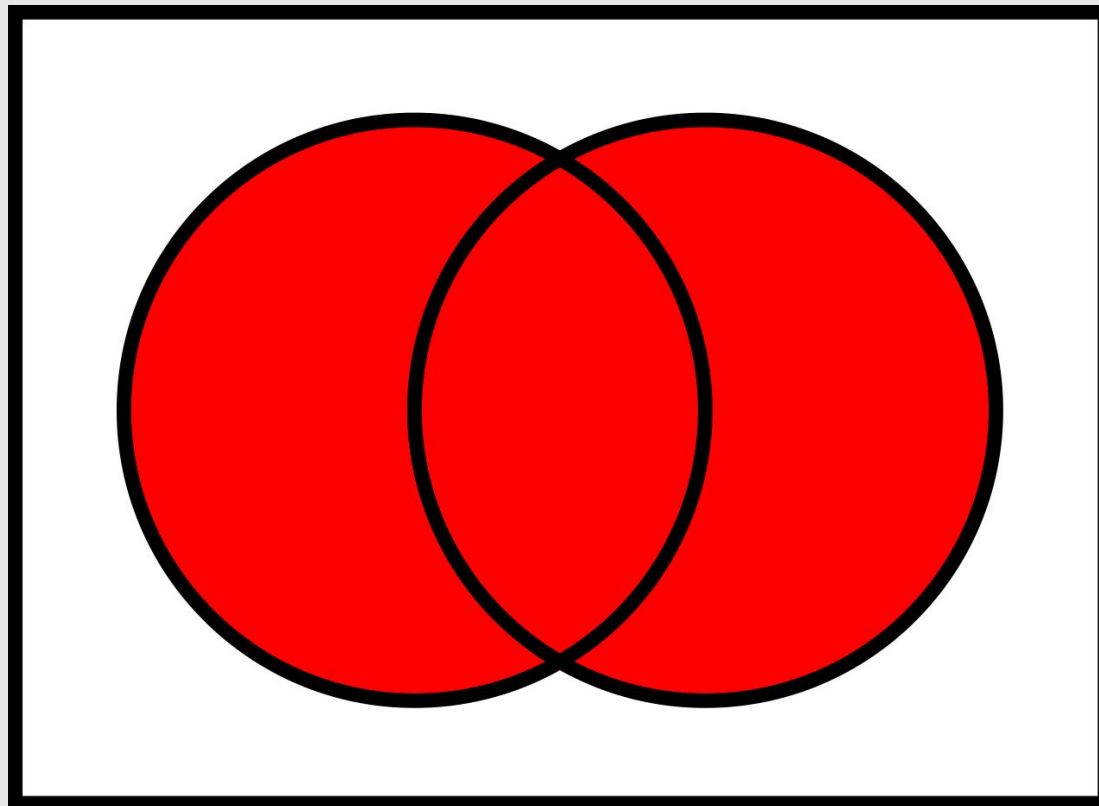
# Operadores lógicos



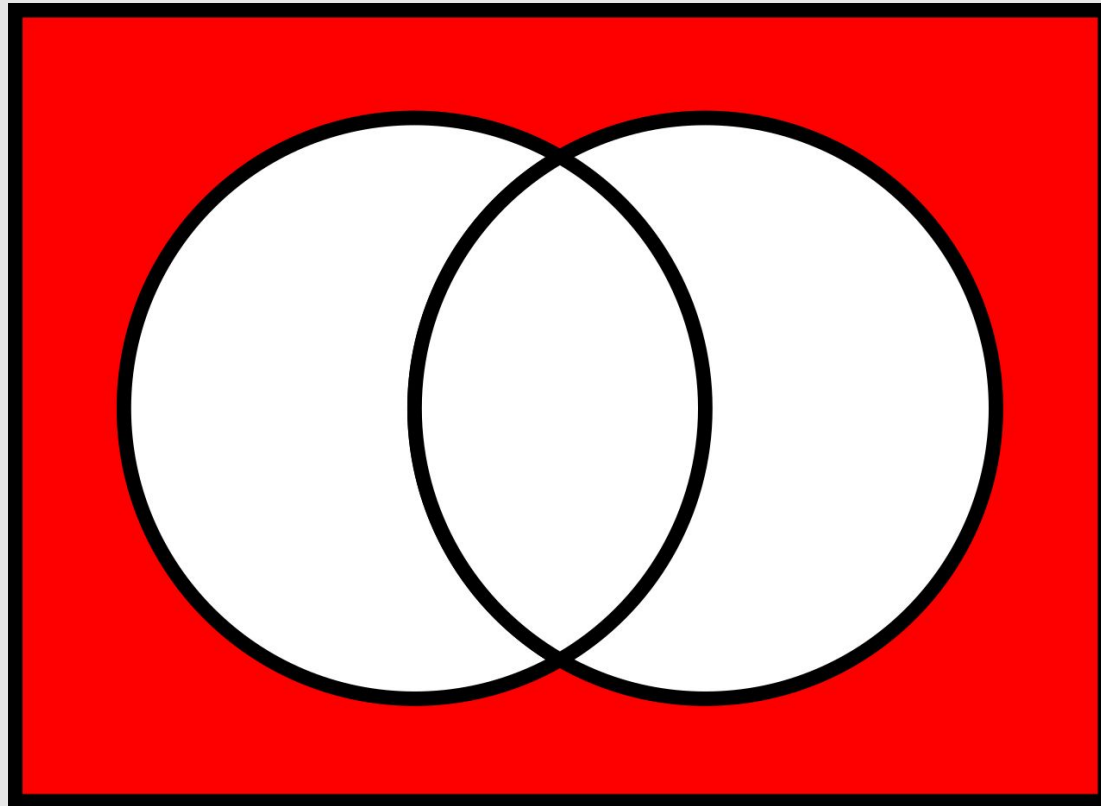
# AND



# OR



# NOR



# Operators

---

- **\$and**

Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.

- **\$not**

Inverts the effect of a query expression and returns documents that do not match the query expression.

# Operators

---

- **\$nor**

Joins query clauses with a logical NOR returns all documents that fail to match both clauses.

- **\$or**

Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

# Expressive operator





# Operators

---

- **\$expr**

Allows use of aggregation expressions within the query language.

# Consultas a Arrays y SubDoc

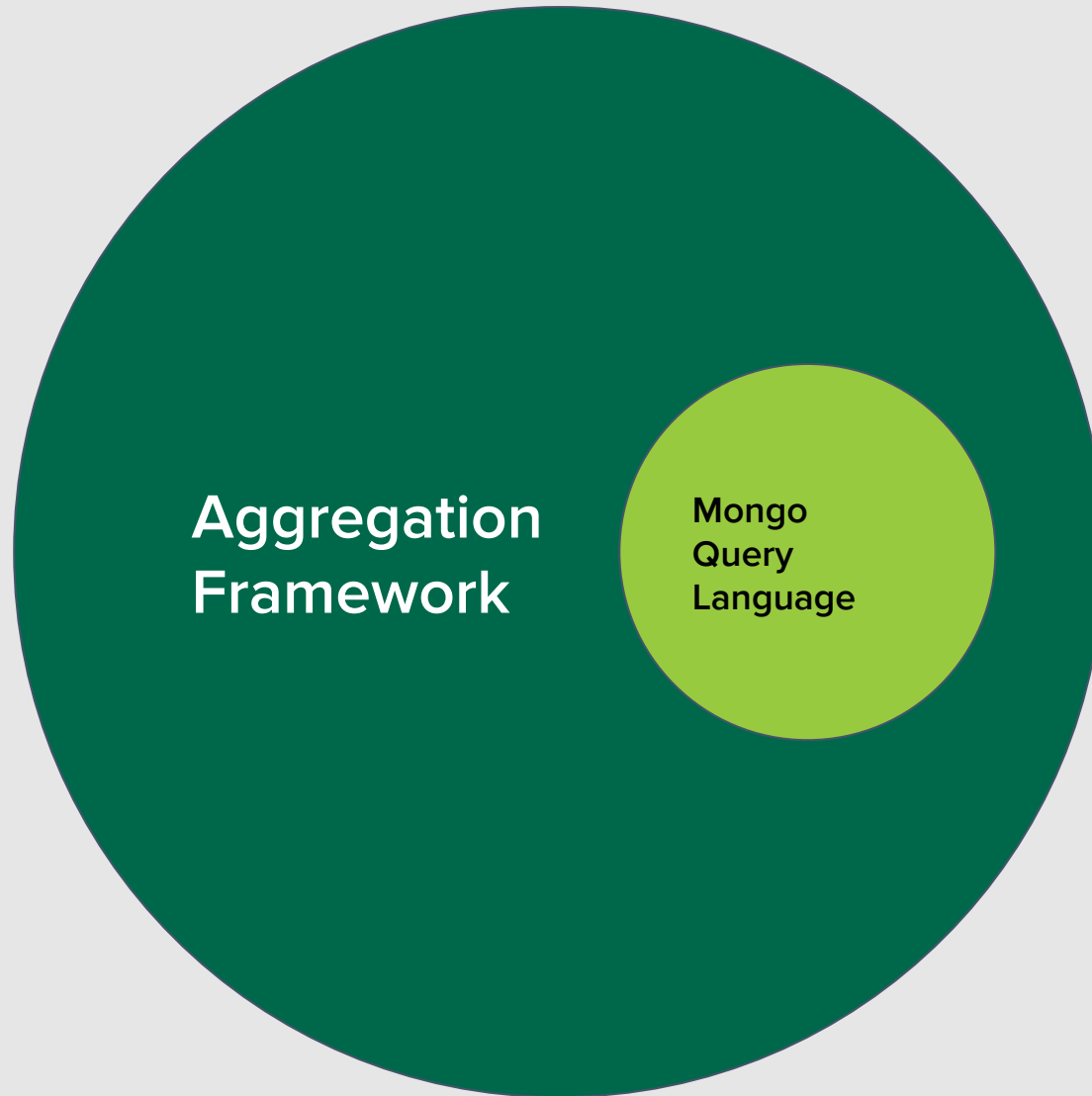


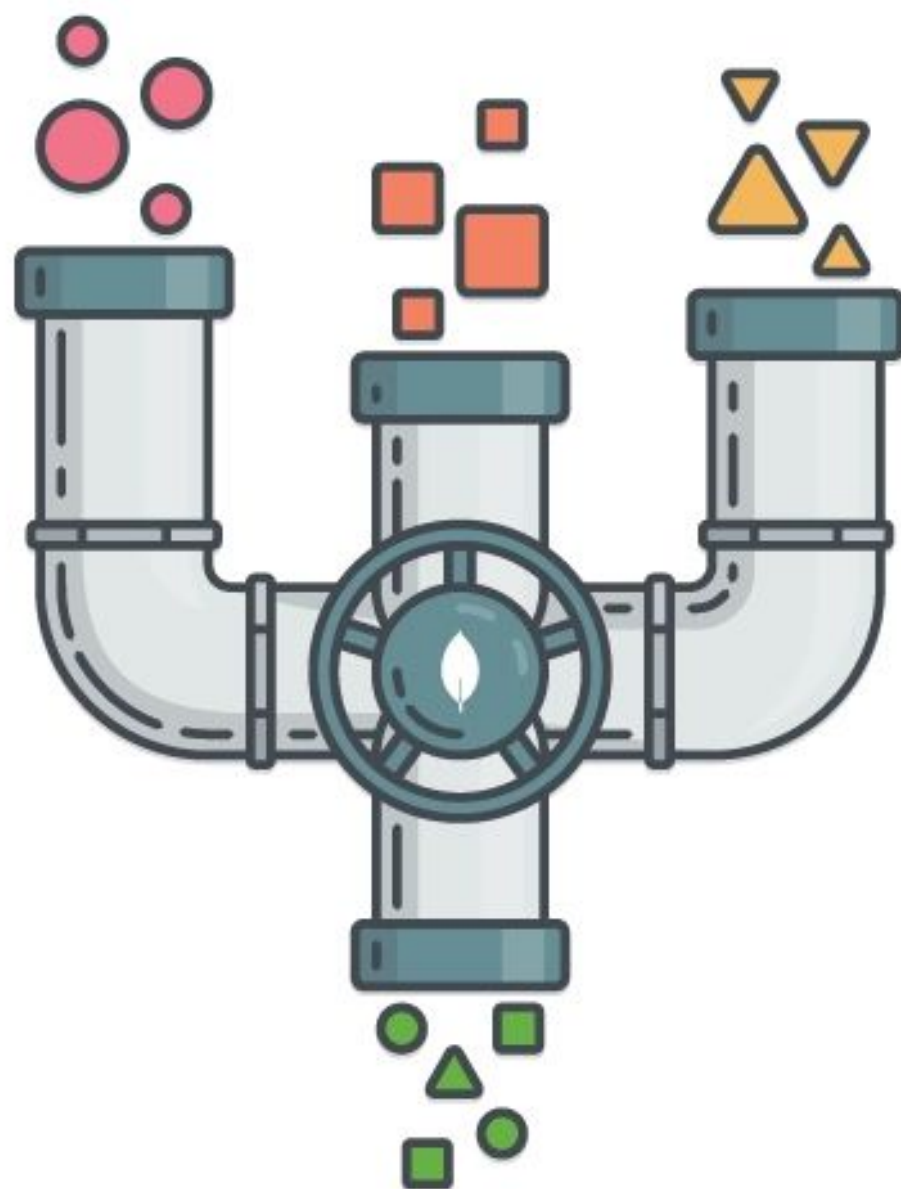
# Array **Update** Operators



# Aggregation Framework











GAZ  
NATURAL  
911

energie

GAZ  
NATURAL

GAZ  
NATURAL



# Sort & Limit



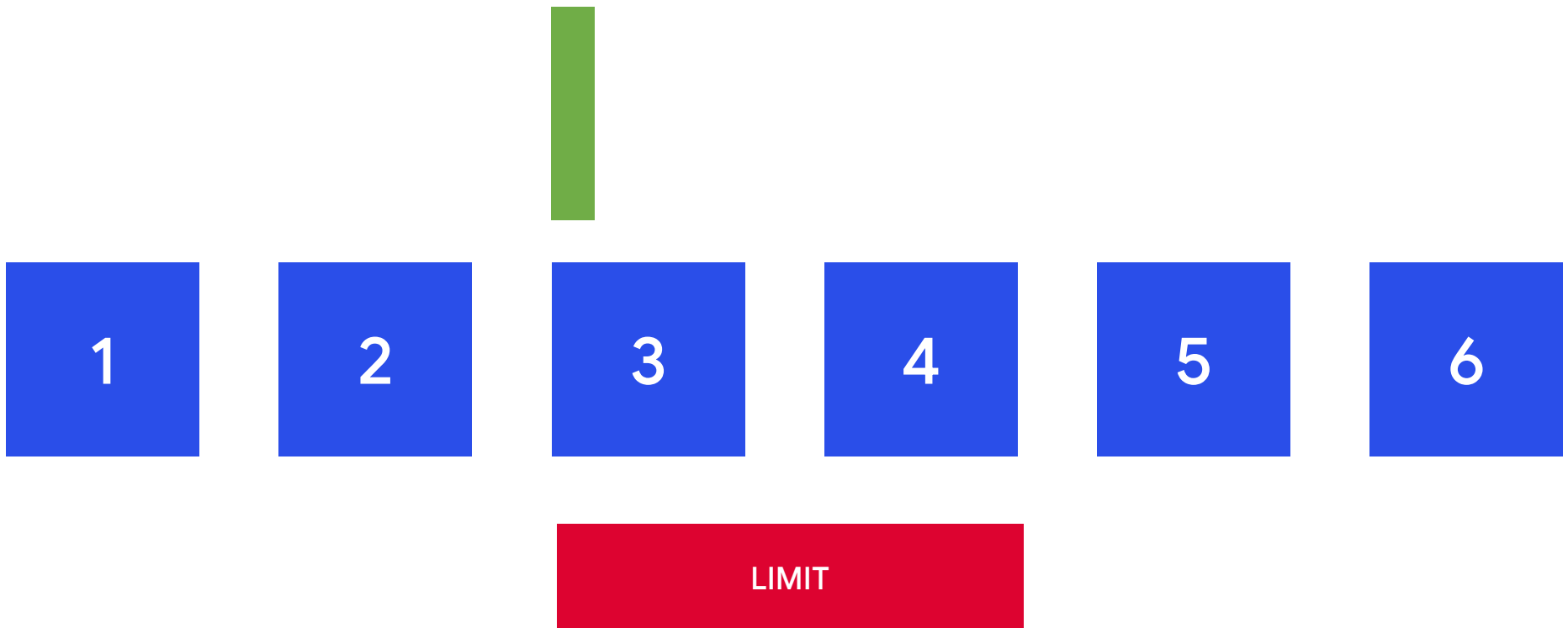


# Pagination

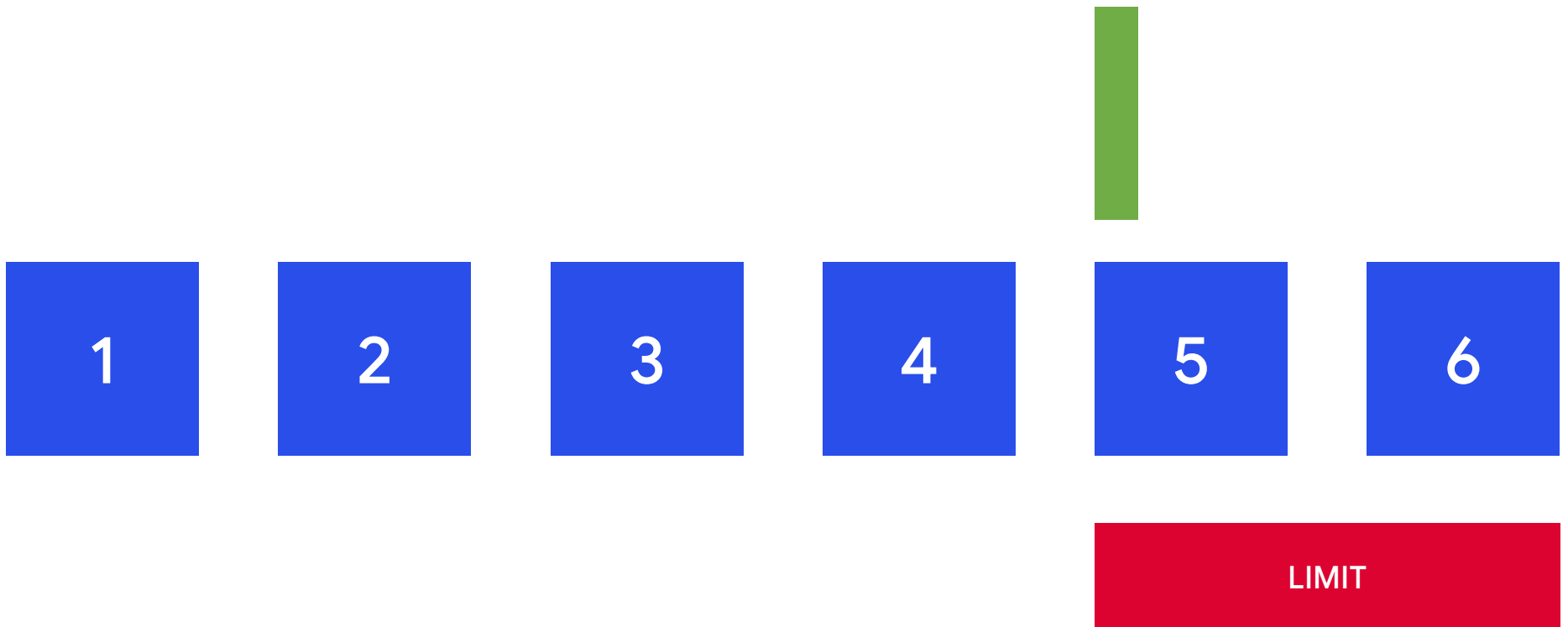


limit = 2

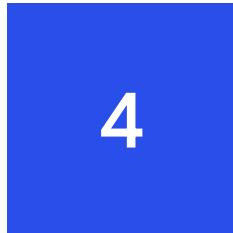
skip / offset = 2



limit = 2  
skip / offset = 4



limit = 2  
skip / offset = 0

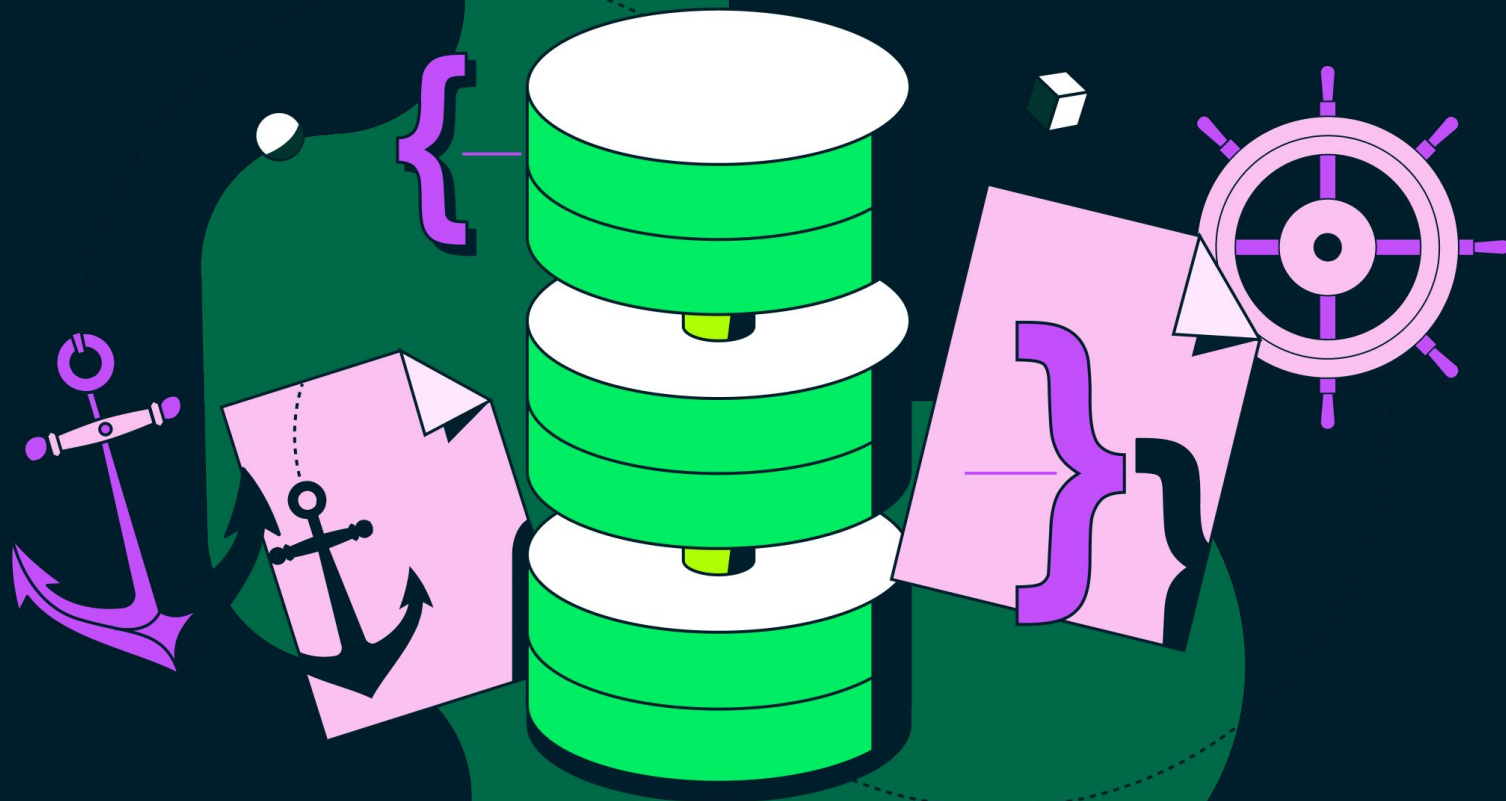


# Atlas Features





MongoDB®



```
from pymongo import MongoClient  
  
client = MongoClient(port=27017)  
  
db=client.business  
  
fivestarcoun = db.reviews.find({'rating': 5}).count()  
  
result = db.reviews.update_one({...})
```





```
const { MongoClient } = require("mongodb");  
  
const client = new MongoClient("uri");  
  
const database = client.db('sample_mflix');  
  
const movies = database.collection('movies');  
  
const query = { title: 'Back to the Future' };  
const movie = await movies.findOne(query);
```



# Curso de **Modelado de Datos** con MongoDB





**Nicolas Molina**  
@nicobytes