Curso Nodejs

Clase 3

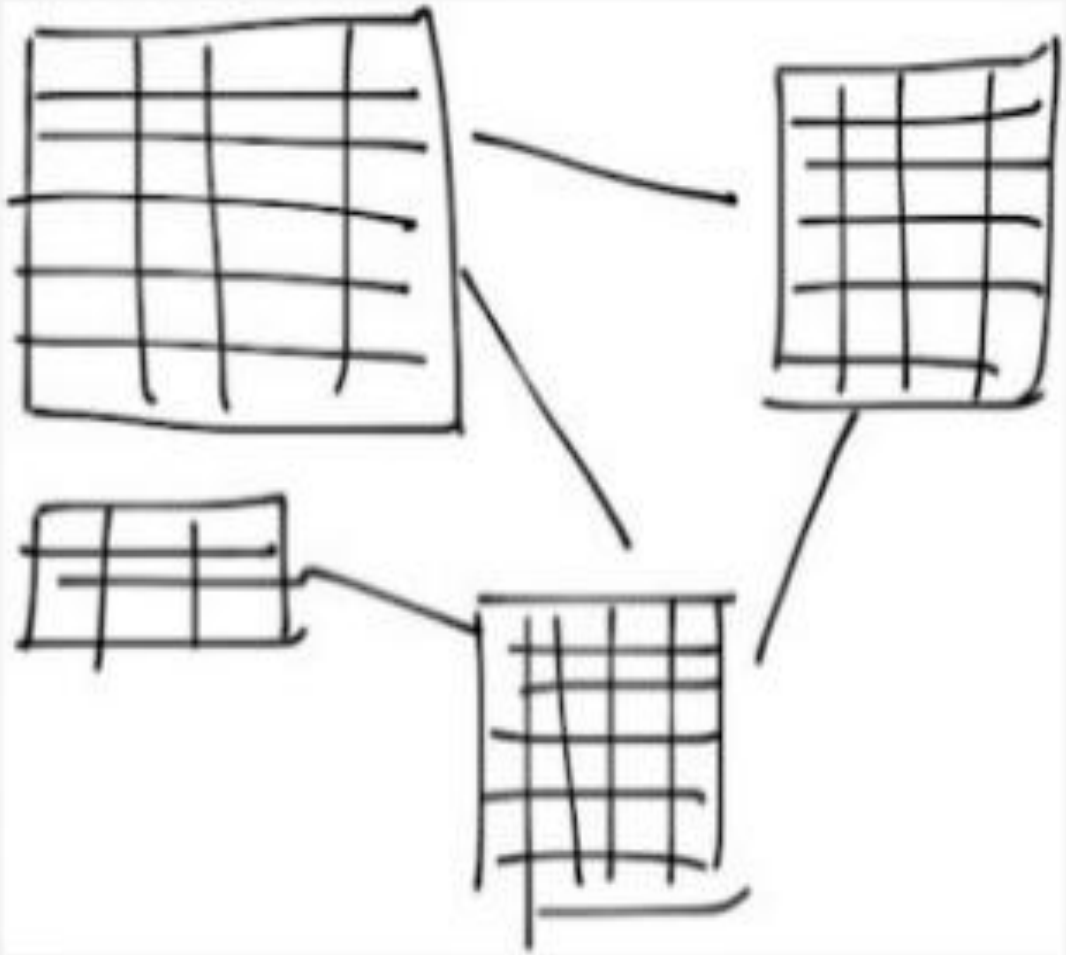
Repasemos lo visto…

Que se acuerdan?

MongoDB

- Es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre.
- Está escrito en C++.
- Funciona en sistemas operativos Windows, Linux, OS X y Solaris.

Antigua concepción

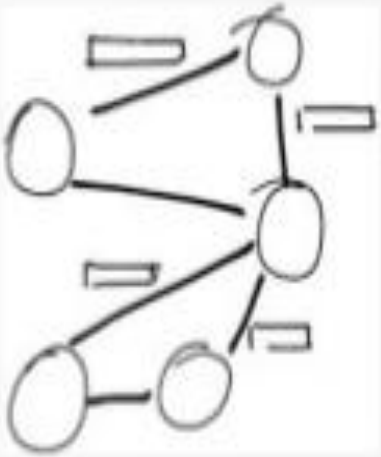


Db's No-Relacionales

Key-Value

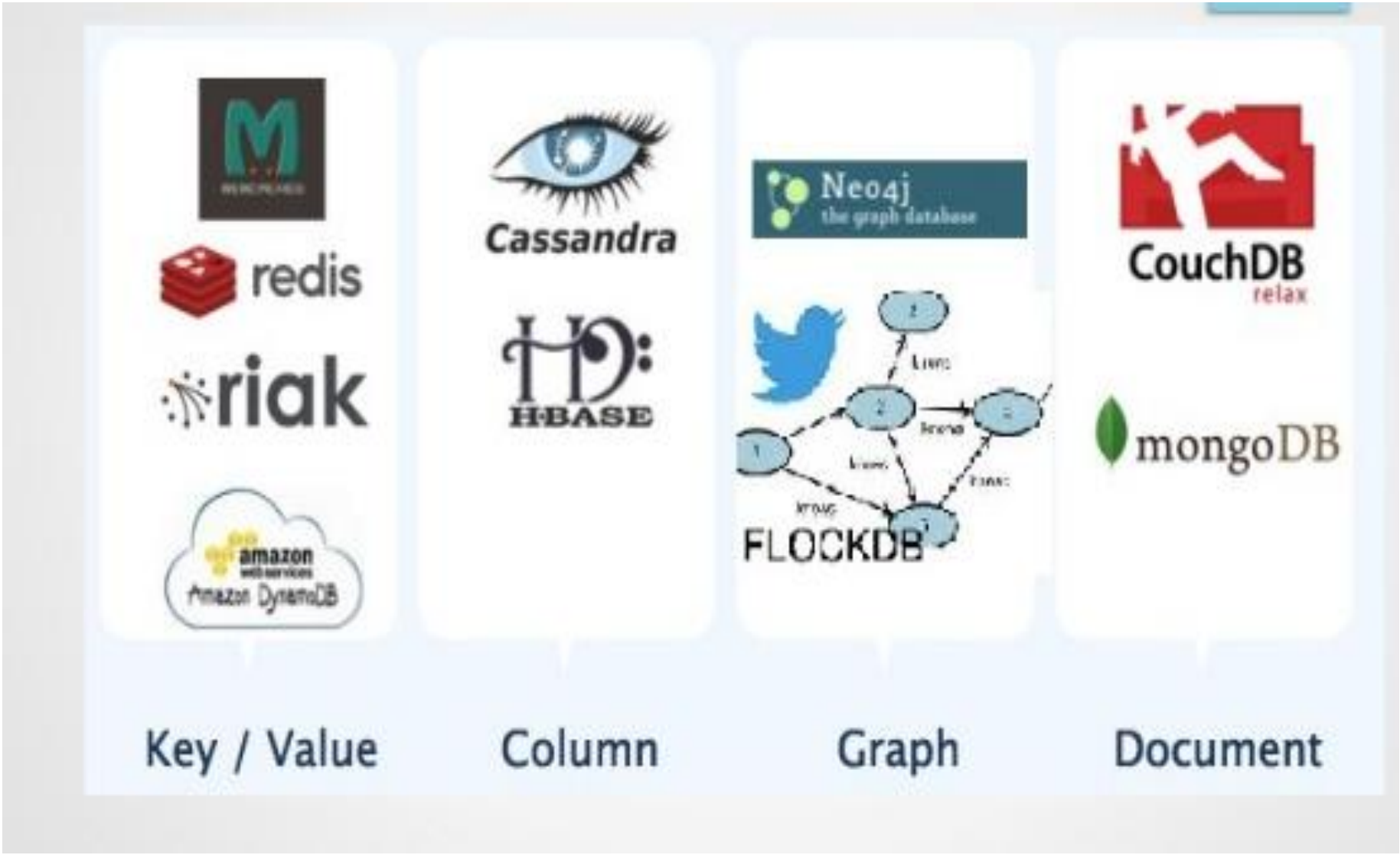
Column

Graph



Document

Ejemplos de Db's no relacionales



Caracteristicas

- **Velocidad**
- **Sistema de consulta**

Terminologia

- Cada registro o conjunto de datos se denomina **documento**.
- Los documentos se pueden agrupar en **colecciones**.
- Se pueden crear **índices** para algunos atributos de los documentos.

Formato

- Los distintos documentos se almacenan en formato **BSON**, o Binary JSON.
- Pero una de las ideas claves en los sistemas NoSQL es que el almacenamiento es barato.
- Sin embargo trabajaremos siempre sobre un documento en **JSON** tanto al almacenar como al consultar información.

Ejemplo:

```
{
  "_id"          : "4da2c0e2e999fb56bf000002"
  "title"        : "Una introducción a MongoDB",
  "body"         : "Lorem ipsum dolor sit amet…",
  "published_at" : "2011-05-09T18:17:07-07:00",
  "author_info"  : {
                     "_id"  : "4dc8919331c0c00001000002"
                     "name" : "Carlos Paramio"
                   },
  "tags"         : ["MongoDB", "NoSQL", "Bases de datos"]
  "comments"     : [
                     {
                       "author_info" : { "name" : "Jorge Rubira", "email" :
"email1@example.com" },
                       "body"        : "Test",
                       "created_at"  : "2011-05-10T10:14:01-07:00"
                     },
                     {
                       "author_info" : { "name" : "Txema Rodríguez", "email" :
"email2@example.com" },
                       "body"        : "Otro test",
                       "created_at"  : "2011-05-10T10:14:09-07:00"
                     }
                   ]
  "liked_by"     : ["4d7cf768e999fb67c0000001", "4da34c62ba875a19d4000001"]
}
```

Instalacion

- http://www.mongodb.org/downloads

- Press Win + R, then type cmd, then press Ctrl + Shift + Enter. (7)
- Press Win + X, then press A. (8)

MongoDB como Servicio

- mkdir c:\data\db
- mkdir c:\data\log

- echo logpath=c:\data\log\mongod.log> "C:\Program Files\MongoDB 2.6 Standard\mongod.cfg" echo dbpath=c:\data\db>> "C:\Program Files\MongoDB 2.6 Standard\mongod.cfg"

MongoDB como Servicio

- sc.exe create MongoDB binPath= "\"C:\Program Files\MongoDB 2.6 Standard\bin\mongod.exe\" --service --config=\"C:\Program Files\MongoDB 2.6 Standard\mongod.cfg\"" DisplayName= "MongoDB 2.6 Standard" start= "auto“

- [SC] CreateService SUCCESS
- net start MongoDB

Comandos Basicos

- Navegamos hasta la carpeta bin dentro de MongoDB

- Ejecutamos: mongo.exe

Comandos Basicos

- Use
- Show
- Db
- Insert
- Find
- findOne

- Ex: db.prueba01.insert({key: “value”})

Ejemplos de uso

Quienes usan MongoDb

Mongo Db indexing

Recién cuando haciamos .find({propiedad: filtro})

Estabamos indexando los resultados por el valor de “propiedad”

Mongo Db indexing

- Soporte a la eficiencia en la ejecución de búsquedas

-Son otras estructuras de datos que guardan pequeñas porciones de datos haciendo fácil navegar-los

Mongo Db indexing

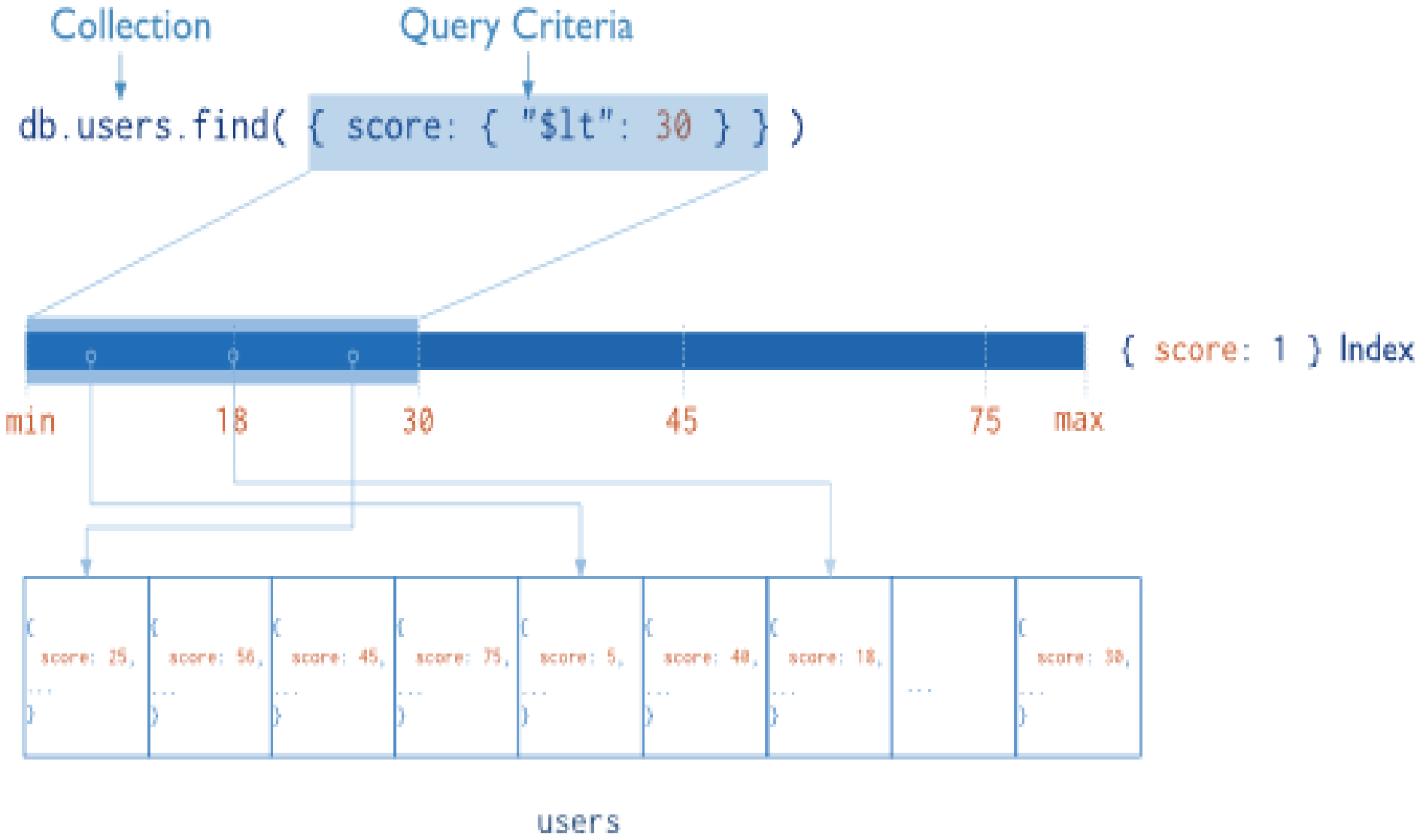
Los índices almacenan el valor de un campo especifico o grupo de campos ordenados por valor.

Concepto similar a otros motores de DB soporta:

- Cualquier campo de los documentos

- Cualquier sub-campo de los documentos

Mongo Db indexing



Replicación

Replicación maestro-esclavo.

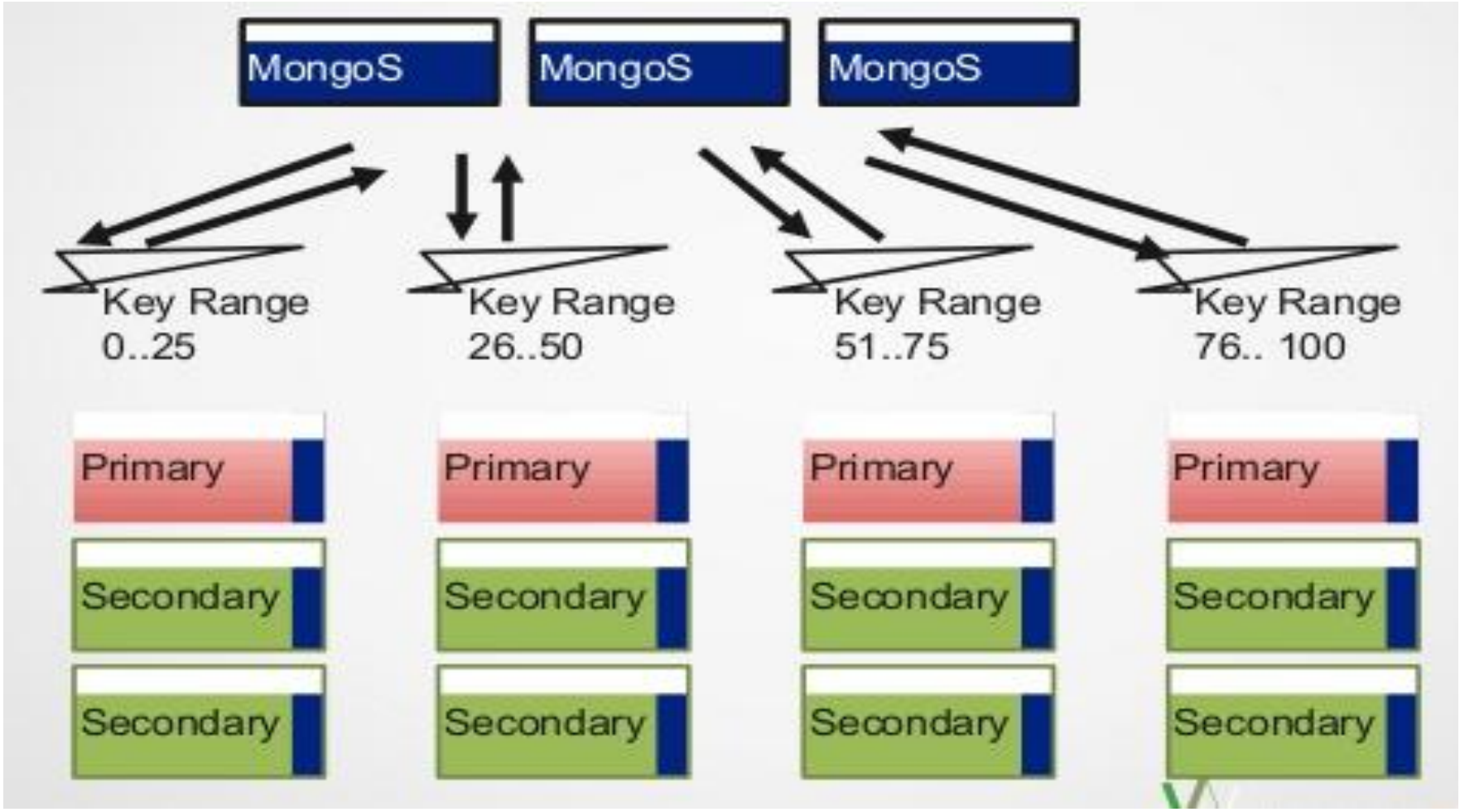
- El maestro puede ejecutar comandos de lectura y escritura.

-El esclavo puede copiar los datos del maestro y sólo se puede usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras.

El esclavo tiene la habilidad de poder elegir un nuevo maestro en caso de que se caiga el servicio con el maestro actual.

Replication

Sharding



Balanceo de carga

MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o duplicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware.

La configuración automática es fácil de implementar bajo MongoDB y nuevas máquinas pueden ser agregadas a MongoDB con el sistema de base de datos corriendo.

Agregación

Las operaciones de agregación permiten procesar los documentos y devolver datos ya computados.

Pueden agrupar datos de muchos documentos y realizar múltiples operaciones en el set de datos para devolver un resultado de mucho mayor valor.

Agregación - Tipos

MongoDb nos provee de tres tipos
de agregación:

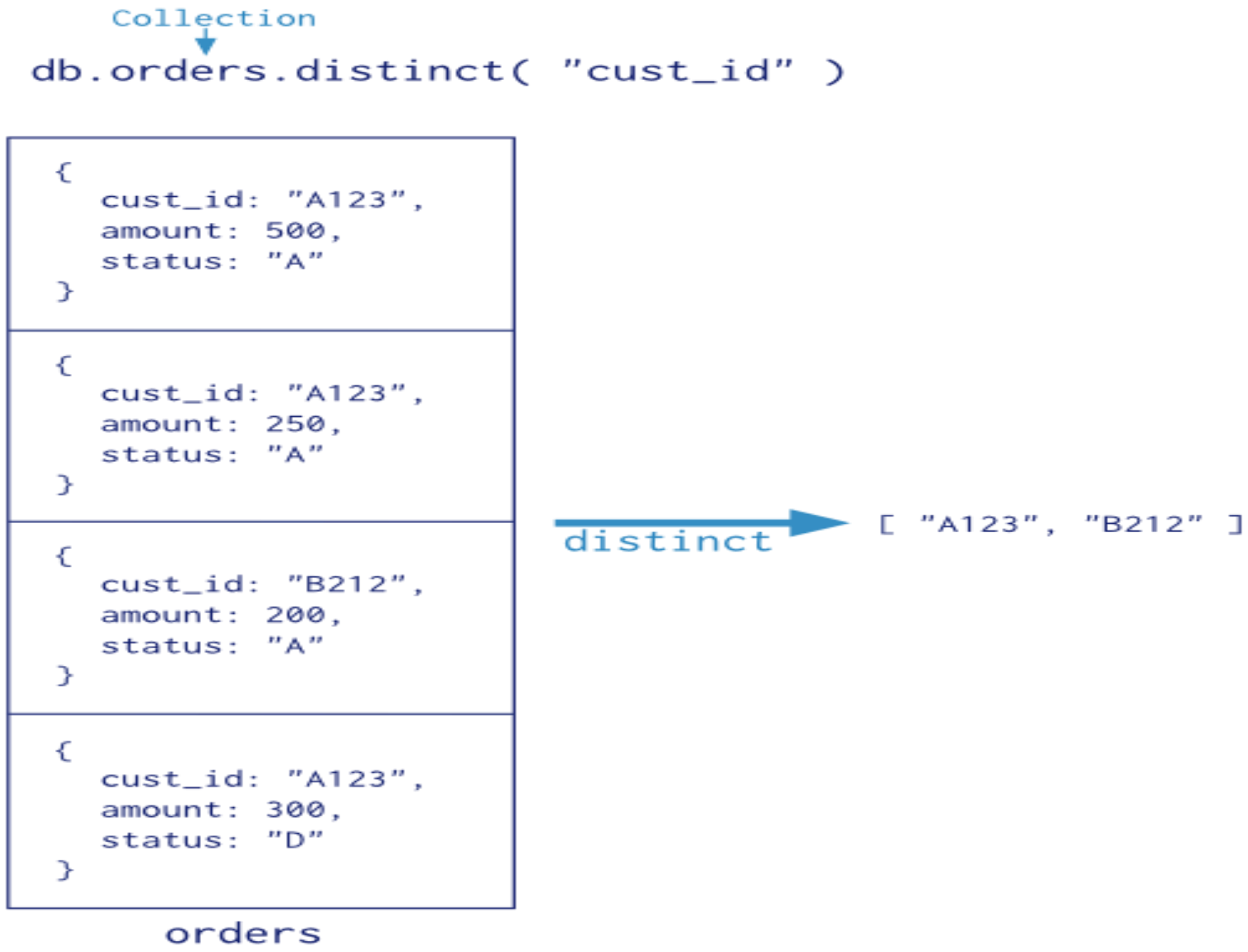
-pipeline o conducto
-map-reduce o mapear y reducir
-single purpose o de simple propósito

Agregación – Single purpose

Se refiere a una gran clase de operaciones de manipulación de datos que se computan pensando en el resultado sobre una entrada y un procedimiento especifico

MongoDb provee un numero de agregaciones que realizan operaciones de agregación especifica sobre un set de datos.

Es mucho mas limitada respecto a pipeline o reduce-map, pero proveen una semántica realmente sencilla para operaciones de procesamiento comunes.


Collection
db.orders.distinct("cust_id")
{
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"
}
orders
distinct
["A123", "B212"]

Agregación – Single Purpose

// devuelve la cantidad de documentos de la colección

db.records.count()

// devuelve la cantidad de documentos que machean con el criterio de búsqueda

db.records.count({ a: 1 })

```
// Records
{ a: 1, count: 4 }
{ a: 1, count: 2 }
{ a: 1, count: 4 }
{ a: 2, count: 3 }
{ a: 2, count: 1 }
{ a: 1, count: 5 }
{ a: 4, count: 4 }
```

EJEMPLO:

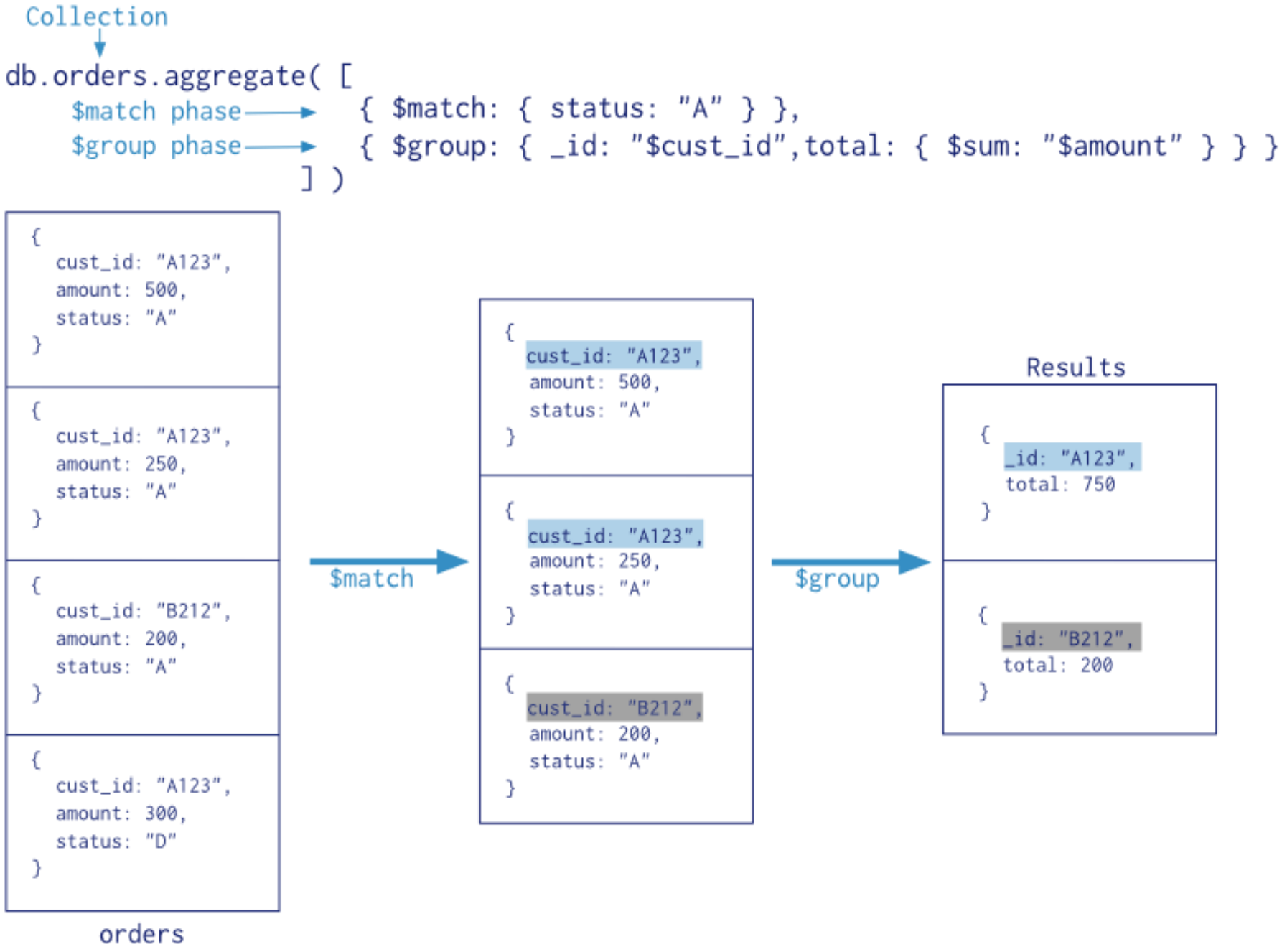
```
db.records.group( {
   key: { a: 1 },
   cond: { a: { $lt: 3 } },
   reduce: function(cur, result) { result.count += cur.count },
   initial: { count: 0 }
} )
```

Resultado:

```
[
  { a: 1, count: 15 },
  { a: 2, count: 4 }
]
```

Agregación - Pipeline

Los documentos son enviados y procesados a través de diferentes fases para lograr un resultado unificado.

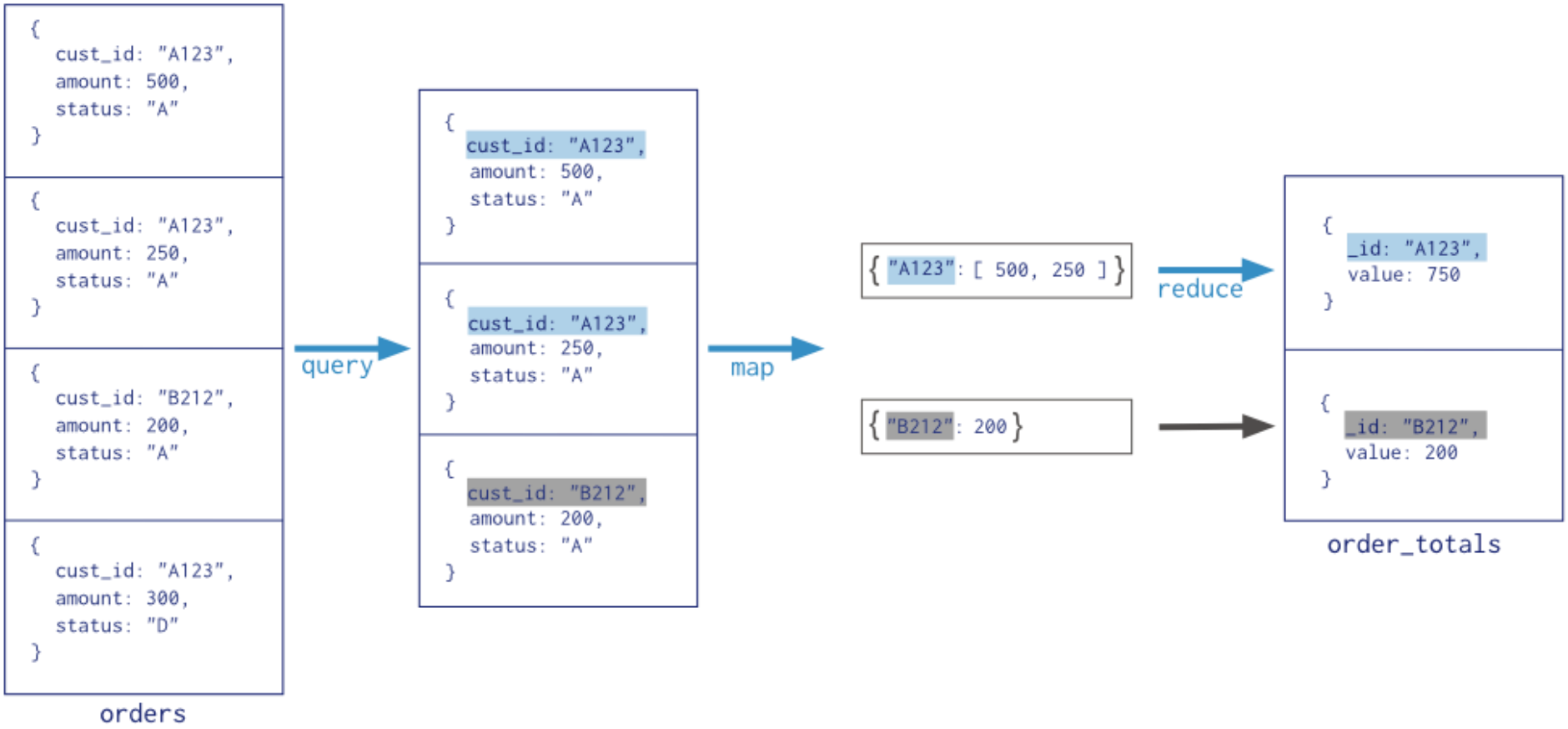

Collection
db.orders.aggregate([
$match phase
{ $match: { status: "A" } },
$group phase
{ $group: { _id: "$cust_id",total: { $sum: "$amount" } } }
])
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }
orders
$match
{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
$group
Results
{ _id: "A123", total: 750 }
{ _id: "B212", total: 200 }

Agregación - Map-Reduce

Paradigma utilizado para mapear, condensar y reducir grandes volúmenes de datos. Aplicando sobre ellos si es requerido alguna operación.

Collection

```
db.orders.mapReduce(
        map      →   function() { emit( this.cust_id, this.amount ); },
        reduce   →   function(key, values) { return Array.sum( values ) },
                     {
        query    →     query: { status: "A" },
        output   →     out: "order_totals"
                     }
                   )
```



Collection

```
db.orders.mapReduce(
        map      →   function() { emit( this.cust_id, this.amount ); },
        reduce   →   function(key, values) { return Array.sum( values ) },
                     {
        query    →     query: { status: "A" },
        output   →     out: "order_totals"
                     }
                   )
```