



Ben Chams - Fotolia



Ben Cramer - Editor

Módulo BIG DATA Mongo

Por

Ing. Elizabeth León Guzmán, Ph.D.

Agenda

- MongoDB
- Modelo de Datos
- Características
- Arquitectura
- Operaciones Principales
- Aplicaciones en la Industria



MongoDB

- Base de datos open-source noSQL orientada a documentos.
- Un registro en MongoDB es un documento, el cuál es una estructura de datos de pares campo/valor.
- Los documentos tienen una estructura similar a JSON.
- MongoDB almacena una colección de documentos.




MongoDB

Documento

- Conjunto de claves con valor asociado.
- La representación de un documento se puede realizar en una estructura, dependiendo del lenguaje de programación: map, hash o diccionario. En javascript:

```
{ "saludo": "hola mundo" }
```



clave valor

Más complejo, con múltiples claves/valor:

```
{ "saludo": "hola mundo", "despedida": "hasta luego" }
```



Tipos de datos diferentes

MongoDB

Documento

- Valores en los documentos, pueden ser documentos embebidos.
- Usados para organizar datos. Ej:

```
{  
  "nombre" : "Juan Perez",  
  "direccion" : {  
    "calle" : "Cra 26 # 44-20",  
    "ciudad" : "Bogotá",  
    "pais" : "Colombia"  
  }  
}
```

¡Representación natural!

MongoDB

Documento

```
{  
  "nombre" : "Jose Diaz",  
  "edad": 27  
  "direccion" : {  
    "calle" : "Cra 6 # 14-9",  
    "ciudad" : "Bogotá",  
    "pais" : "Colombia"  
  }  
  "telefono":{  
    "casa": "2354676"  
    "celular":"311562763"  
  }  
}
```

MongoDB

Documentos

- Los documentos son almacenados en formato BSON (JSON binario – **J**ava **S**cript **O**bject **N**otation)

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

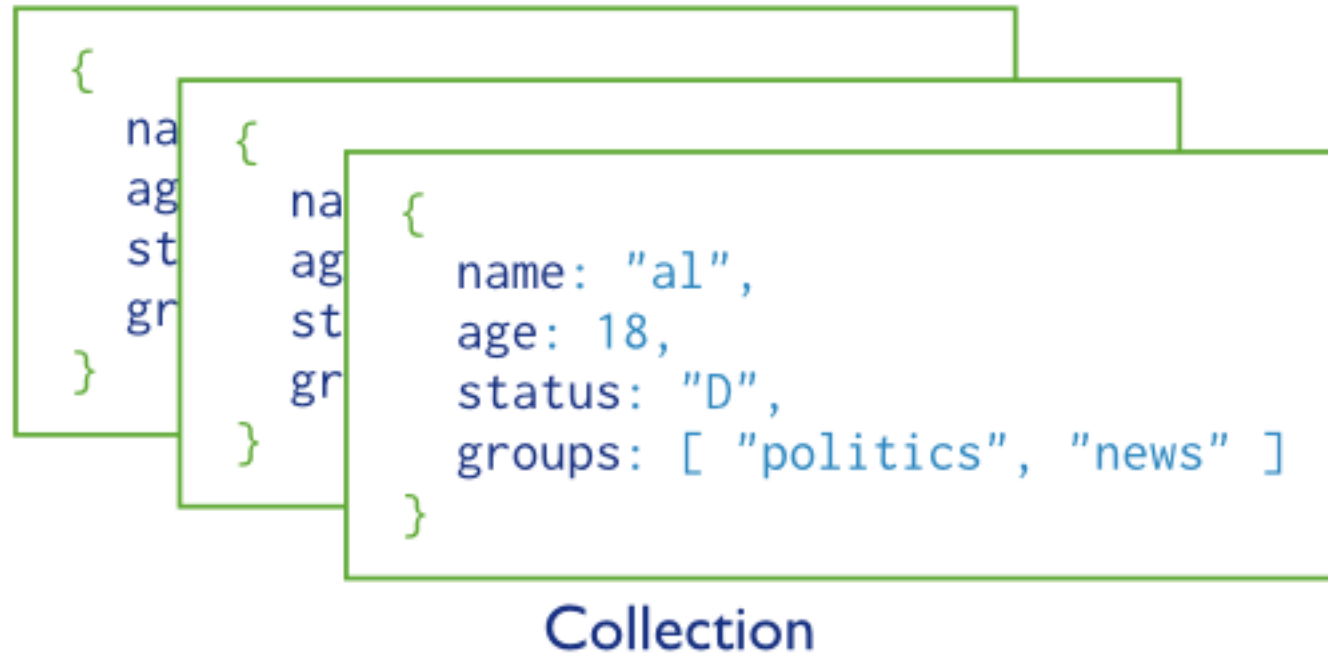


← field: value
← field: value
← field: value
← field: value

- Ventajas del uso de documentos:
 - Los documentos corresponden a tipos de datos nativos en muchos lenguajes de programación.
 - Documentos embebidos y arreglos reducen la necesidad de Joins costosos.

MongoDB

Documento



Colecciones:

Son grupos de documentos relacionados.

Son el equivalente a las tablas en una base de datos relacional.

Los documentos son de diferente forma (diferentes campos).

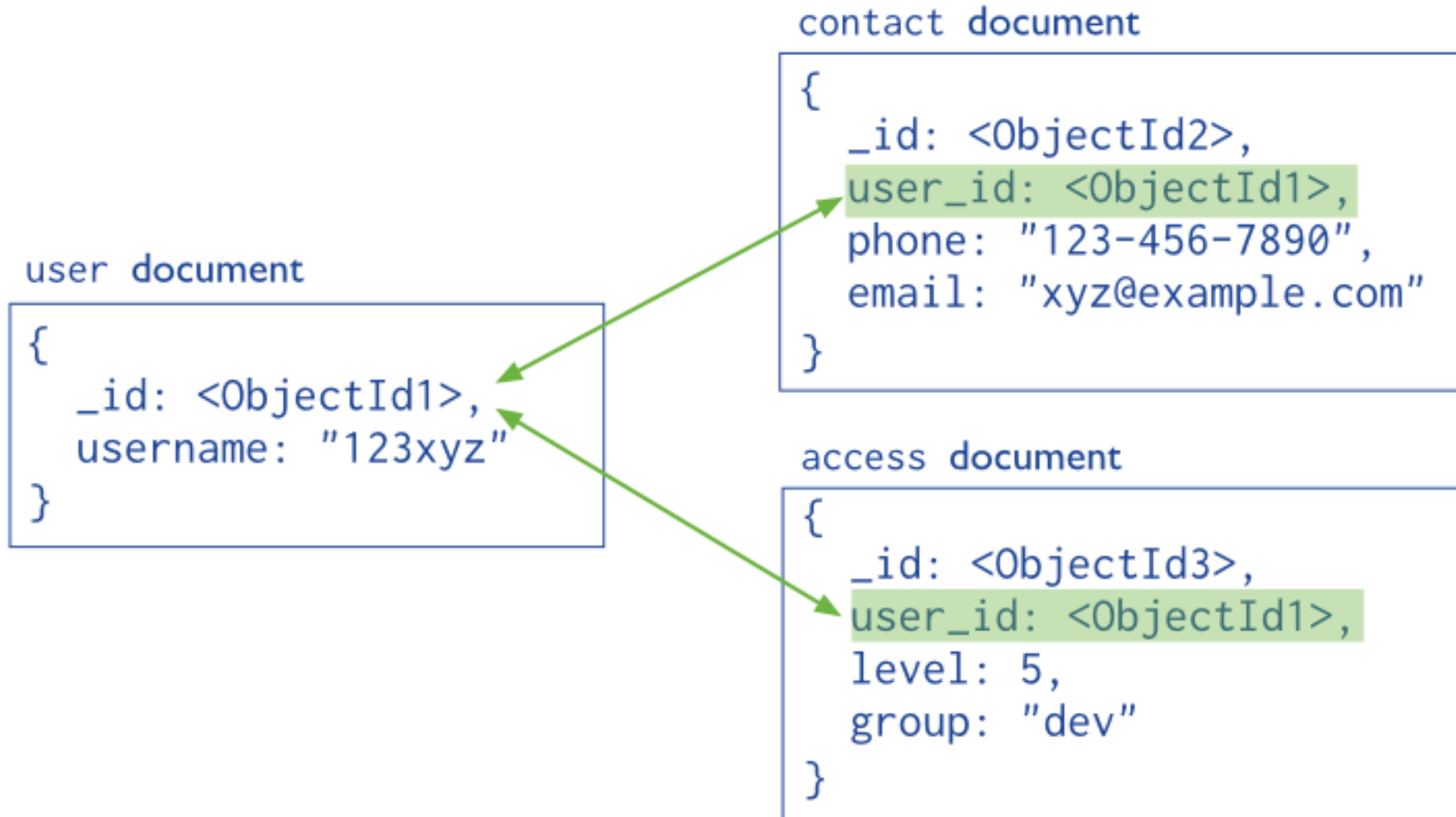
MongoDB – Características

- Bases de datos:
 - Son colecciones de documentos



Modelo de Datos

- Referencias (Normalizado):



Modelo de Datos

- Embebidos:

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

Embedded sub-document

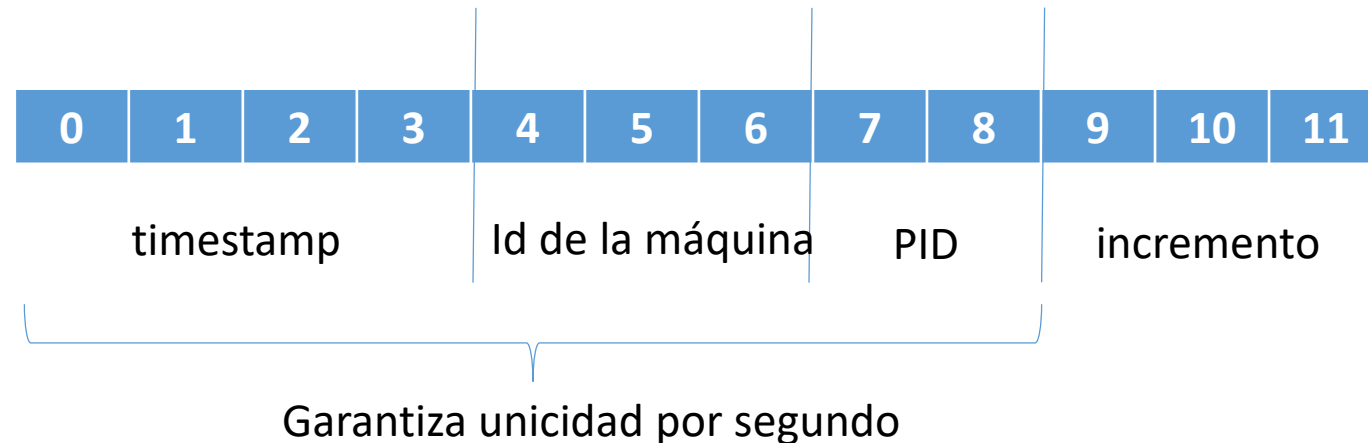
Embedded sub-document

MongoDB

Identificador Único

- ***ObjectId***

- Tipo de dato por defecto de las claves
- Fácil de generar globalmente de manera única usando varias máquinas (ambiente fragmentado)
- 12 bytes para almacenar (hexadecimal string)
- Es creado usando el timestamp, el identificador único de la máquina, PID (identificador de proceso), incremento secuencial

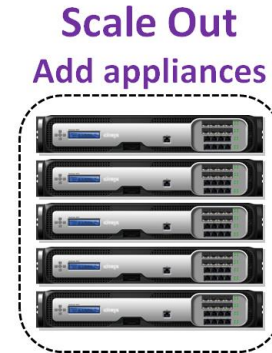


Características

- Fácil de usar
- Sin esquemas predefinidos:
 - Tanto claves como valores, no tienen tipo de dato ni tamaño fijo
 - Fácil de adicionar y remover columnas (campos)
 - Desarrollo fácil

Características

- Terabytes son comunes
- Fácil de escalar
- “Scaling out” (escalable horizontalmente): adicionar espacio o incrementar el rendimiento (comprar servidores(nodos) y adicionarlos al cluster). Difícil de administrar miles de máquinas...
- Dividir datos a través de varios servidores
- Mongo automáticamente:
 - balancea y carga los datos.
 - redistribuye los documentos, y
 - dirige las solicitudes a las máquinas correctas.
- Desarrolladores se concentran en programar las aplicaciones y no del escalamiento



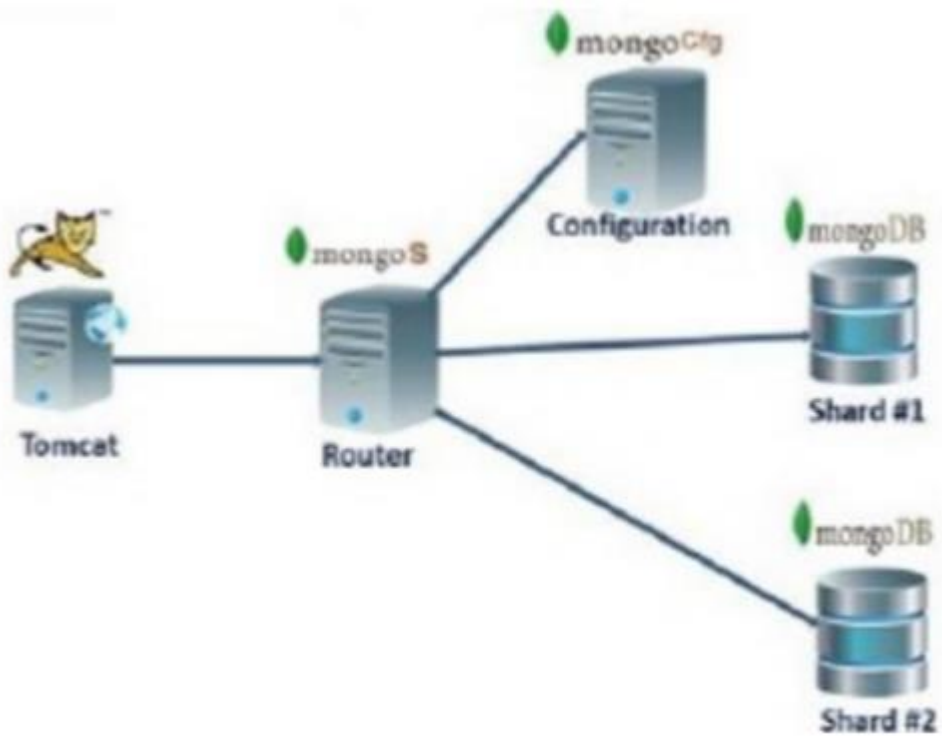
Características

- Indexación
 - Indices secundarios -> Consultas rápidas
 - Indices únicos, compuestos, geoespaciales, y “full-text”
- Tipos de colecciones especiales:
 - Colecciones de datos que pueden desaparecer en cierto momento. EJ: sesiones
 - Colecciones de tamaño fijo (Logs)
- Almacenamiento en archivo
 - Soporta protocolo fácil de usar para almacenar grandes archivos de datos y metadatos.

Características

- Rendimiento es lo más importante, sin sacrificar velocidad (Alta eficiencia):
 - Usa mucha RAM (cache)
 - Automáticamente escoger los índices apropiados para las consultas. Incluyen claves en documentos embebidos y arreglos
 - El soporte para datos embebidos reduce las operaciones I/O en el sistema de BD.
- Lenguaje de consultas rico:
 - Soporte para operaciones CRUD.
 - Agregación de datos.
 - Búsqueda de texto.
 - Consultas geo-espaciales.
- Alta disponibilidad:
 - Replica sets.
 - Recuperación automática de fallos.
 - Redundancia de datos.

Arquitectura



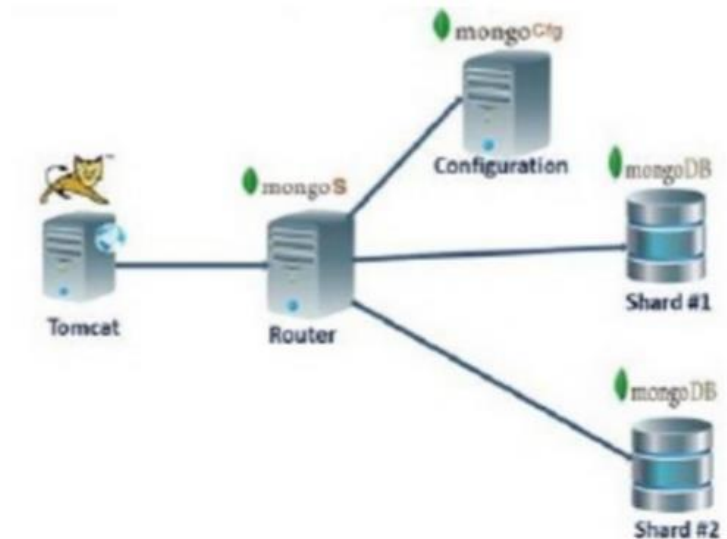
Conexión usando fragmentación “sharding”



Conexión “Stand-alone” (No fragmentada)

Herramientas

- **Mongod**: Motor central de la base de datos. Tiene tres funcionalidades:
 - Standalone server, Config server, Shard partition
- **MongoS**: “Database router”.
 - Recepción y entrega de datos
 - Balanceo
 - Mantenimiento de mongoCfg
- **GridFS**: sistema de almacenamiento

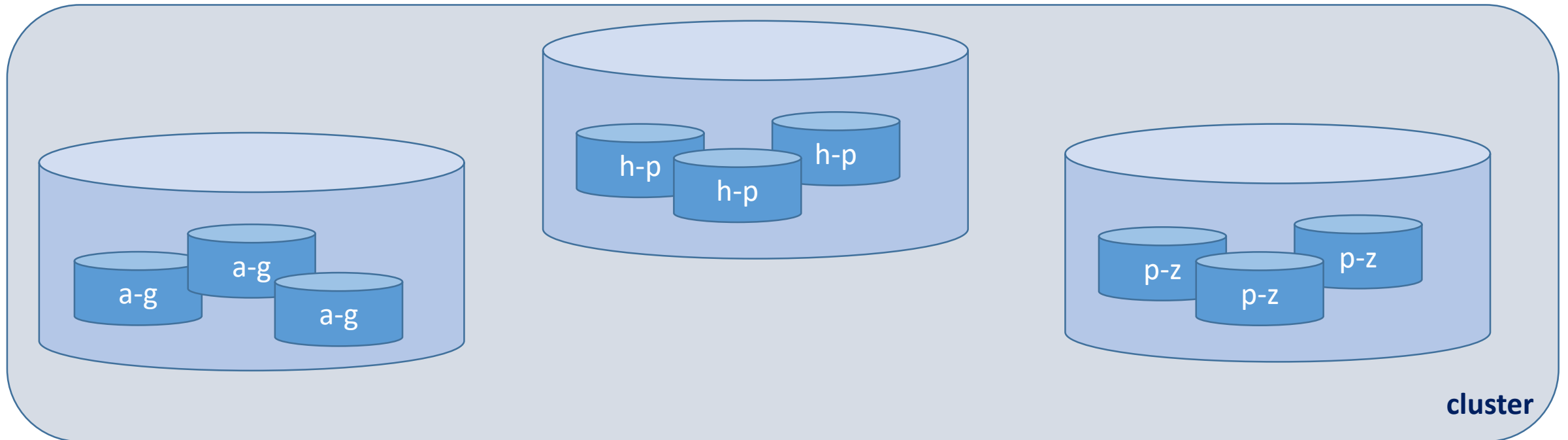


Fragmentación (Sharding)

- Método para dividir los datos en los múltiples servidores. Mongo lo realiza de manera automática:
 - “cluster transparente”, usa enrutamiento MongoS. Son la cara de cualquier aplicación. MongoS reenvía la consulta al servidor/es correctos y devuelve al cliente la respuesta.
 - Disponibilidad de lectura y escritura. El clúster debe permitir fallar tantos nodos como límite. Mongo permite redundancia, de forma que si cae un nodo, otro continua con los procesos.
 - Crecimiento ágil. El cluster añade o elimina capacidad cuando lo necesite

Fragmentación (Sharding)

- Un fragmento (shard) es uno o varios servidores de un cluster, responsables de un subconjunto de datos. Si el *shard* está compuesto por más de un servidor, cada uno tendrá una copia idéntica de los datos.

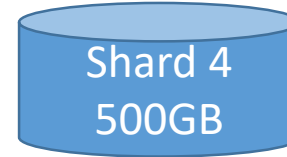


Distribución de “shards”/fragmentos

Fragmentación (Sharding)

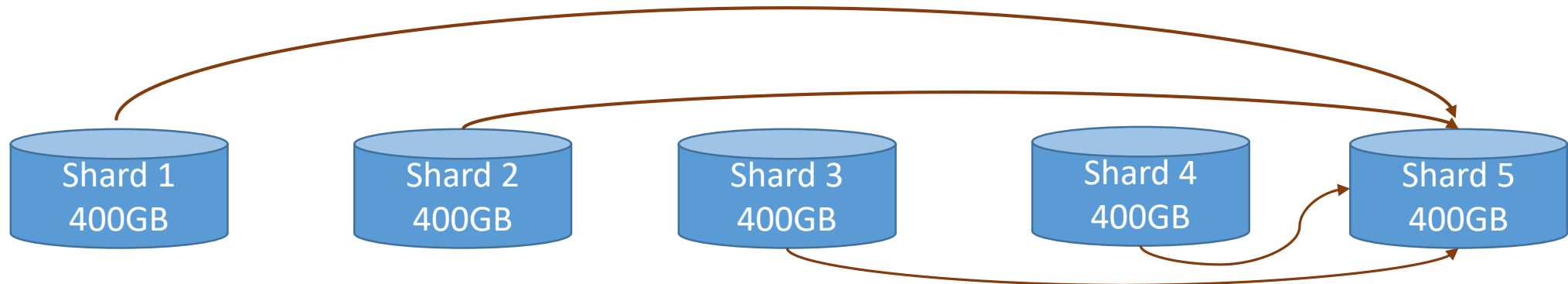
- Al definir rangos, puede que un *shard* quede con más datos (desbalanceo), para eso mongo:
 - Usa múltiples rangos
 - Mover pocos datos entre los servidores para el balanceo
 - Divide conjuntos de datos muy grandes en dos de forma automática

Primer conjunto de datos



Fragmentación (Sharding)

- Al definir rangos, puede que un *shard* quede con más datos (desbalanceo), para eso mongo:
 - Usa múltiples rangos
 - Mover pocos datos entre los servidores para el balanceo
 - Divide conjuntos de datos muy grandes en dos de forma automática



Si los datos crecen, Mongo divide los datos, creando nuevos subconjuntos, De forma balanceada

Fragmentación (Sharding)

Antes de que la colección sea fragmentada

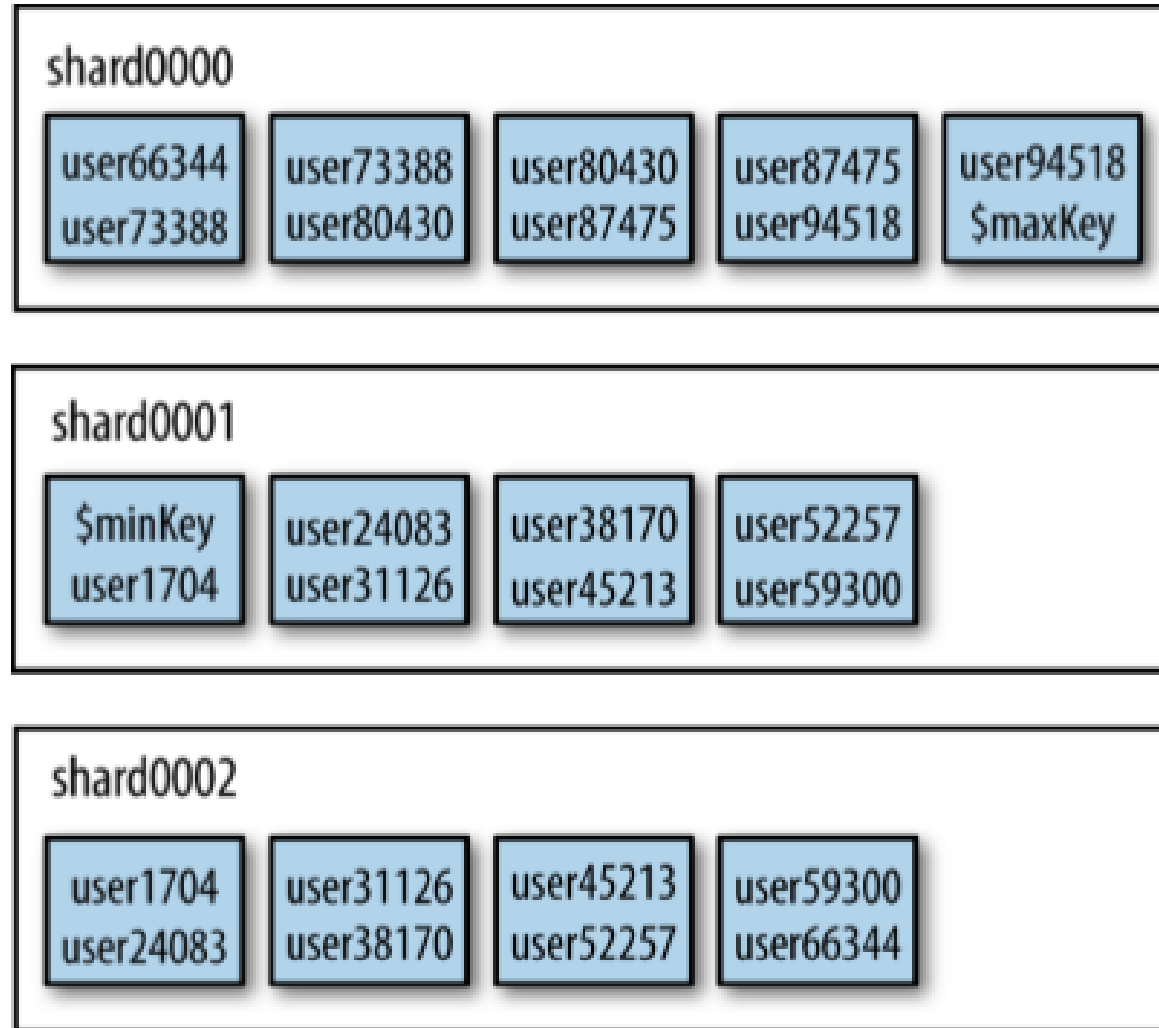


“Sharding” divide en pequeños fragmentos basado en la clave del shard



Sharding

Los fragmentos pueden ser distribuidos en el cluster



Herramientas

mongoimport: Archivos CSV/JSON/TSV

bsondump: Convierte de BSON a JSON

mongoexport: Archivos CSV/JSON

mongodump: Hot Backup (binary)

mongorestore: Utiliza un archivo generado por mongodump y rescata la información.

Operaciones principales

- MongoDB proporciona el mongo-shell para acceder y manipular las bases de datos.
- Crear o seleccionar base de datos:
`use myDB`
- Crear una colección:
 - Se crean automáticamente, si no existen, al ingresar un documento o mediante el método:
`db.createCollection()`

Operaciones Principales

- Insertar documento en colección:

`db.collection.insert()`

```
db.users.insert (  ← collection
{
  name: "sue",      ← field: value
  age: 26,          ← field: value
  status: "A"       ← field: value
}                  } document
)
```

Operaciones Principales

- Leer documentos de una colección:

`db.collection.find()`

```
db.users.find(  
  { age: { $gt: 18 } },  
  { name: 1, address: 1 }  
) .limit(5)
```

← collection
← query criteria
← projection
← cursor modifier

- Actualizar documentos de una colección:

`db.collection.update()`

```
db.users.update(  
  { age: { $gt: 18 } },  
  { $set: { status: "A" } },  
  { multi: true }  
)
```

← collection
← update criteria
← update action
← update option

Operaciones Principales

- Eliminar documentos de una colección:

`db.collection.remove()`

```
db.users.remove(  
    { status: "D" }  
)
```

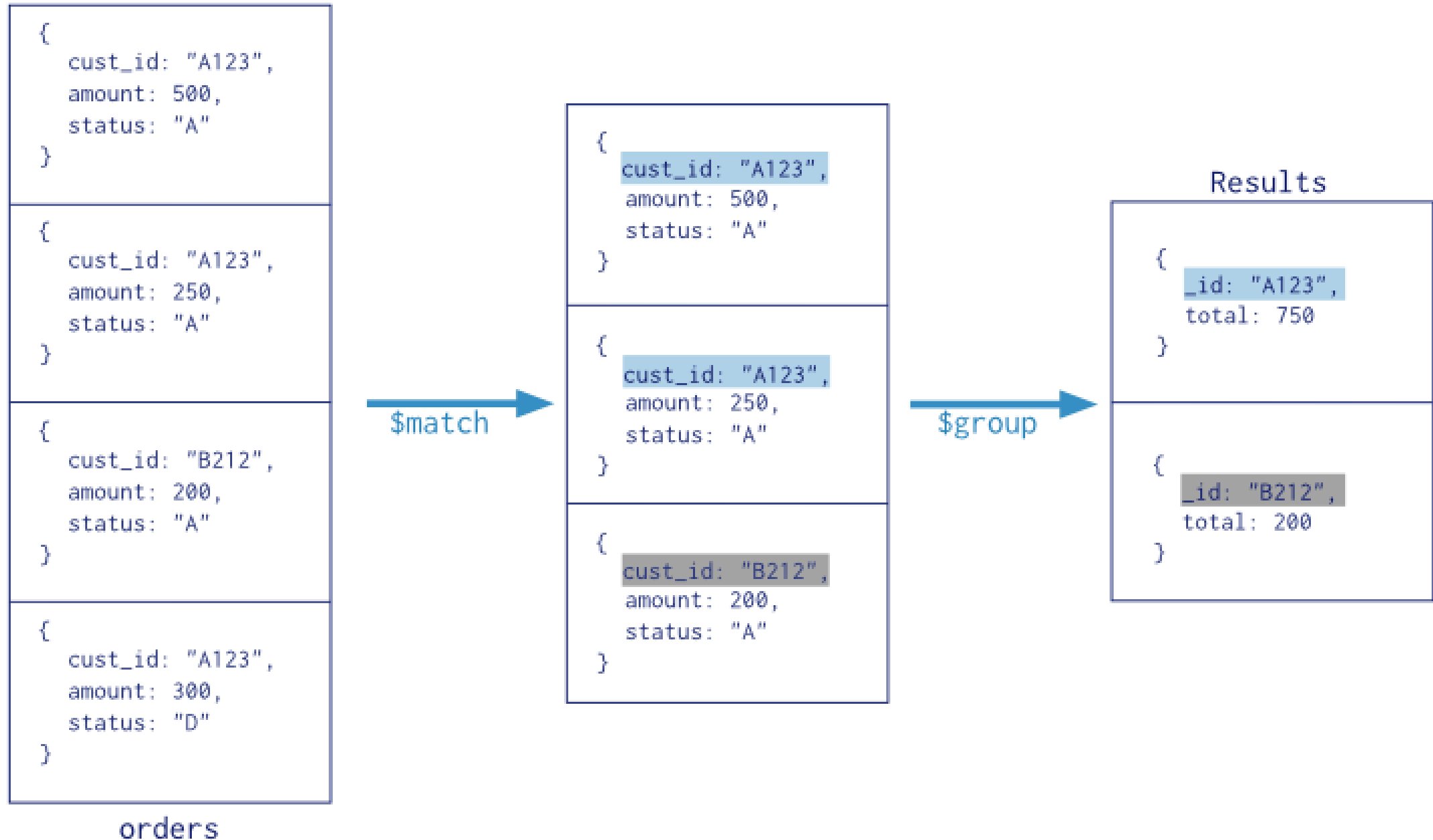
← collection
← remove criteria

Operaciones Principales

- Agregación de documentos:

Collection
↓
db.orders.aggregate([
 \$match stage → { \$match: { status: "A" } },
 \$group stage → { \$group: { _id: "\$cust_id", total: { \$sum: "\$amount" } } }
])

- Agregación de documentos:

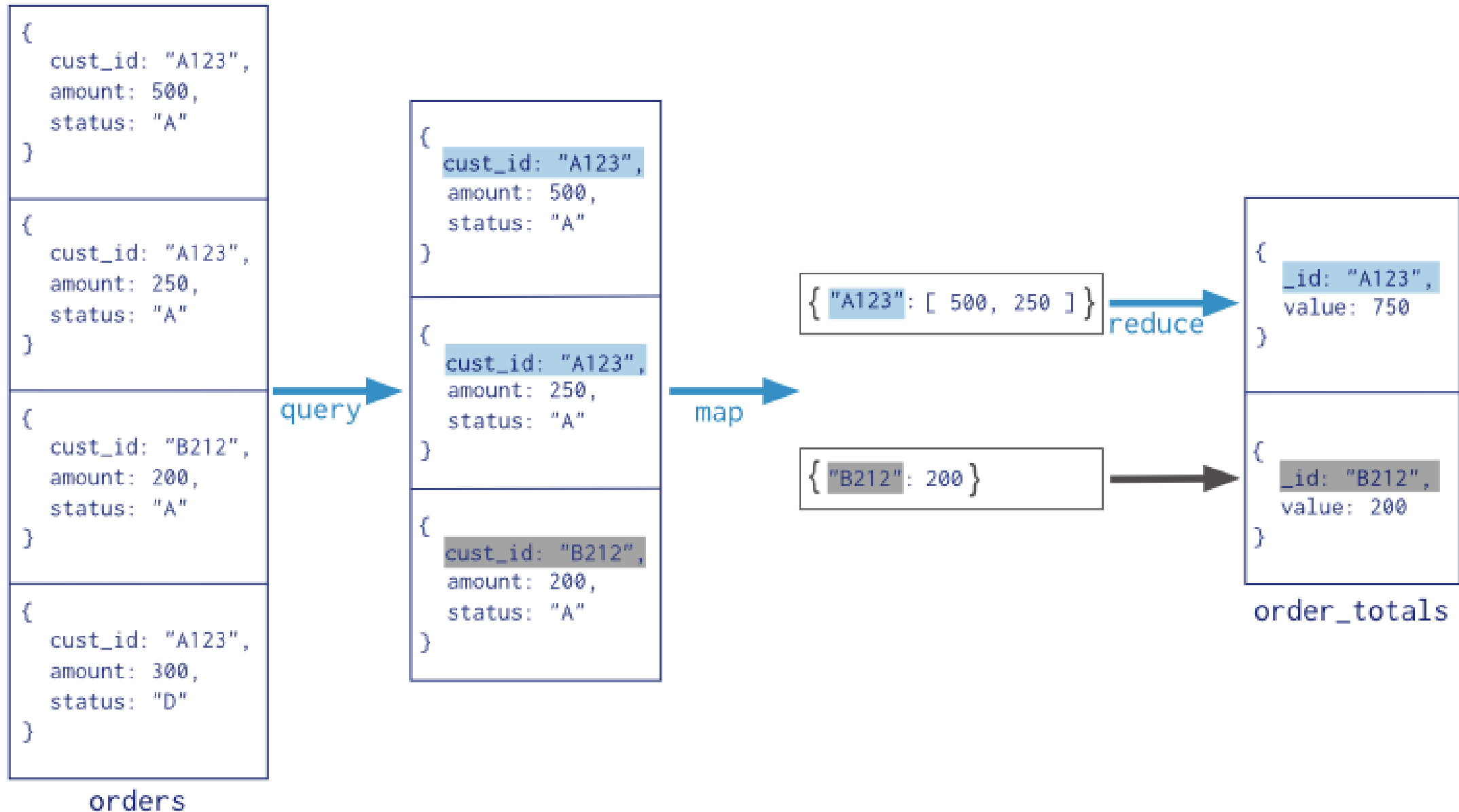


Operaciones Principales

- Map-Reduce:

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


- Map-Reduce:



Índice

Doc1 (“el niño juega en el parque, en el parque hay muchos árboles”)

Doc2 (“el fenómeno del niño acabará pronto ”)

Doc3 (“en el parque niños elevan cometa ”)

Doc4 (“el niño eleva cometa”)

	Doc 1	Doc 2	Doc 3	Doc 4
acabará	0	1	0	0
arboles	1	0	0	0
cometa	0	0	1	1
del	0	1	0	0
eleva	0	0	1	1
el	3	1	1	1
en	2	0	1	0
fenómeno	0	1	0	0
hay	1	0	0	0
juega	1	0	0	0
muchos	1	0	0	0
niño	1	1	1	1
parque	2	0	1	0
pronto	0	1	0	0

Índice

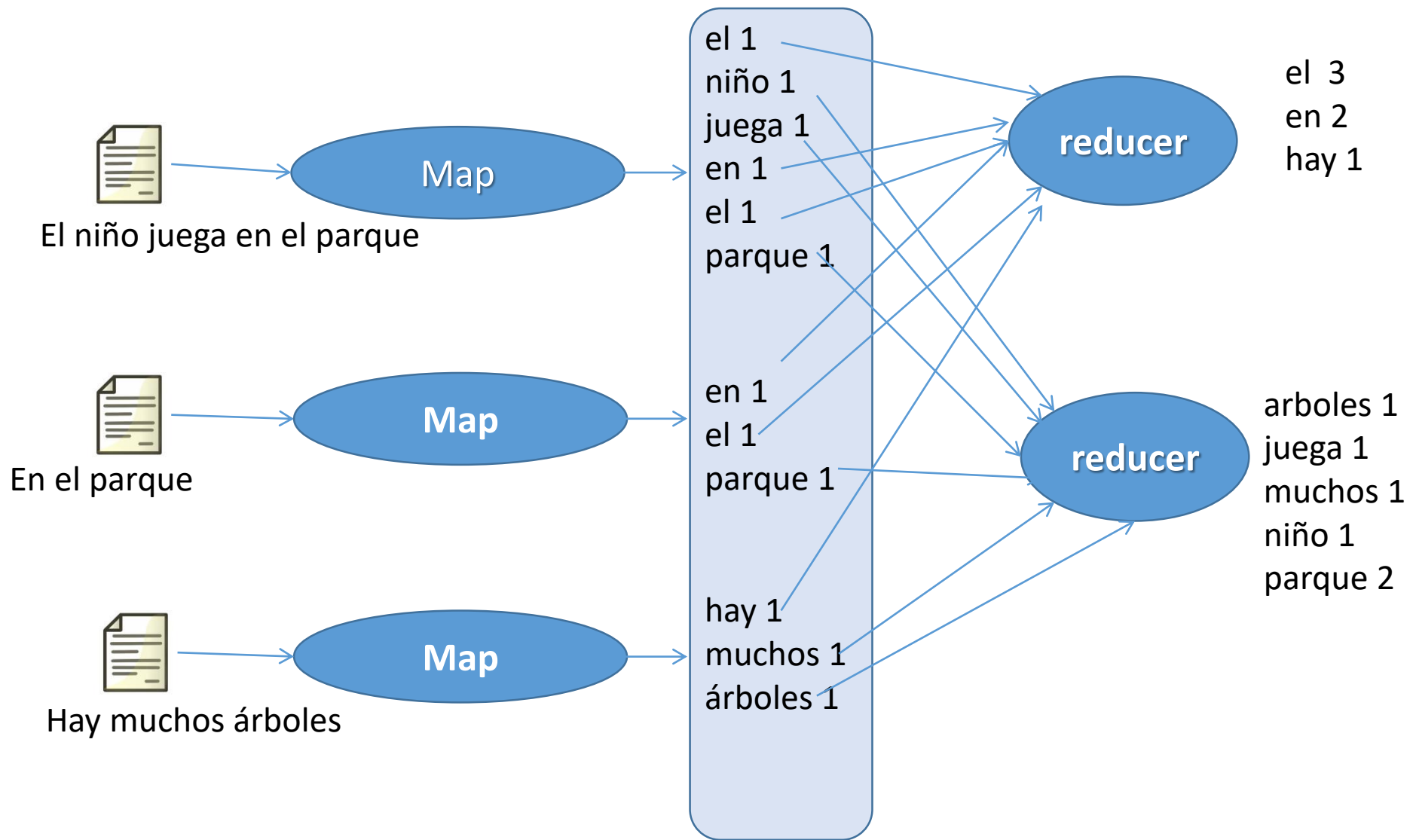
Doc1 (“el niño juega en el parque, en el parque hay muchos árboles”)

Doc2 (“el fenómeno del niño acabará pronto ”)

Doc3 (“en el parque niños elevan cometa ”)

Doc4 (“el niño eleva cometa”)

	Doc 1	Doc 2	Doc 3	Doc 4
acabará	0	1	0	0
arboles	1	0	0	0
cometa	0	0	1	1
del	0	1	0	0
eleva	0	0	1	1
el	3	1	1	1
en	2	0	1	0
fenómeno	0	1	0	0
hay	1	0	0	0
juega	1	0	0	0
muchos	1	0	0	0
niño	1	1	1	1
parque	2	0	1	0
pronto	0	1	0	0



Operaciones Principales

- Considere el siguiente ejemplo:

```
db.stores.insert(  
  [  
    { _id: 1, name: "Java Hut", description: "Coffee and cakes" },  
    { _id: 2, name: "Burger Buns", description: "Gourmet hamburgers" },  
    { _id: 3, name: "Coffee Shop", description: "Just coffee" },  
    { _id: 4, name: "Clothes Clothes Clothes", description: "Discount clothing" },  
    { _id: 5, name: "Java Shopping", description: "Indonesian goods" }  
  ]  
)
```

- Se crea la colección *store* con cinco documentos.

Operaciones Principales

- Búsqueda de Texto:
 - Para realizar búsqueda de texto, MongoDB utiliza índices de texto [text index](#) y el operador [\\$text](#).
- Text Index:
 - Cualquier campo de tipo *string* o arreglo de *strings*.
 - Para realizar búsqueda de texto es necesario crear un *text index* en la colección.

```
db.stores.createIndex( { name: "text", description: "text" } )
```

- Operador \$text:
 - Tokeniza el string de búsqueda por medio de espacios y signos de puntuación.
 - Ejecuta un *OR* lógico con todos los tokens en el string de búsqueda.

Operaciones Principales

- Por ejemplo, encontrar todos los *store* conteniendo cualquier termino de la lista “coffee”, “shop” y “java”.

```
db.stores.find( { $text: { $search: "java coffee shop" } } )
```

- Buscar frase exacta:

```
db.stores.find( { $text: { $search: "java \"coffee shop\"" } } )
```

- Exclusión de término:

```
db.stores.find( { $text: { $search: "java shop -coffee" } } )
```

Operaciones Principales

- Resultados ordenados:

```
db.stores.find(  
  { $text: { $search: "java coffee shop" } },  
  { score: { $meta: "textScore" } }  
) .sort( { score: { $meta: "textScore" } } )
```

- Los resultados son ordenados por orden de relevancia.
- La relevancia se calcula mediante un puntaje calculado para cada documento.
- El puntaje indica que tan bien el documento empata la consulta.

Aplicaciones en la Industria

- Internet de las Cosas.
- Comercio electrónico.
- Procesamiento de información y análisis en tiempo real.
- Administración de contenido.
- Algunas compañías que utilizan MongoDB en producción:
 - Facebook
 - Forbes
 - SAP
 - Codecademy
 - Disney
 - Github

Referencias

- Arquitectura MongoDB, <https://www.mongodb.com/mongodb-architecture>
- Documentación MongoDB, <https://docs.mongodb.com/manual/>
- MongoDB. The definitive Guide. Kristina Chodorov. O'Reilly. 2013