





# mongoDB

## 101

dailos rafael díaz lara  
julio 2015

 @ddialar  [linkedin.com/in/ddialar](https://www.linkedin.com/in/ddialar)

**Bases de Datos Relacionales**

**Bases de Datos No Relacionales**

**Introducción a MongoDB**

**Comparativa entre Bases de Datos Relacionales y MongoDB**

**Principales elementos de MongoDB**

**Iniciando MongoDB**

**Operaciones CRUD en MongoDB**

**Replicación de datos**

**Sharding de datos**



# **Bases de Datos Relacionales**

Bases de Datos No Relacionales

Comparativa entre Modelos de Datos

Introducción a MongoDB

Principales elementos de MongoDB

Iniciando MongoDB

Operaciones CRUD en MongoDB

Replicación de datos

Sharding de datos





## Tabla de Datos




## ***Estructura básica de Relación***

	Campo 1	Campo 2	Campo 3	...	Campo n
	Texto	Texto	Entero	...	Booleano
Registro 1					
Registro 2					
Registro 3					
...					
Registro n					



## ***Ejemplo de Relación***

	Nombre	Apellido	Edad	...	Hijos
	Texto	Texto	Entero	...	Booleano
Registro 1	Antonio	Fajardo	24	...	true
Registro 2	Sandra	Alonso	19	...	false
Registro 3	Carmelo	Bautista	43	...	false
...	...	...	...	...	...
Registro n	Alicia	Marqués	28	...	true

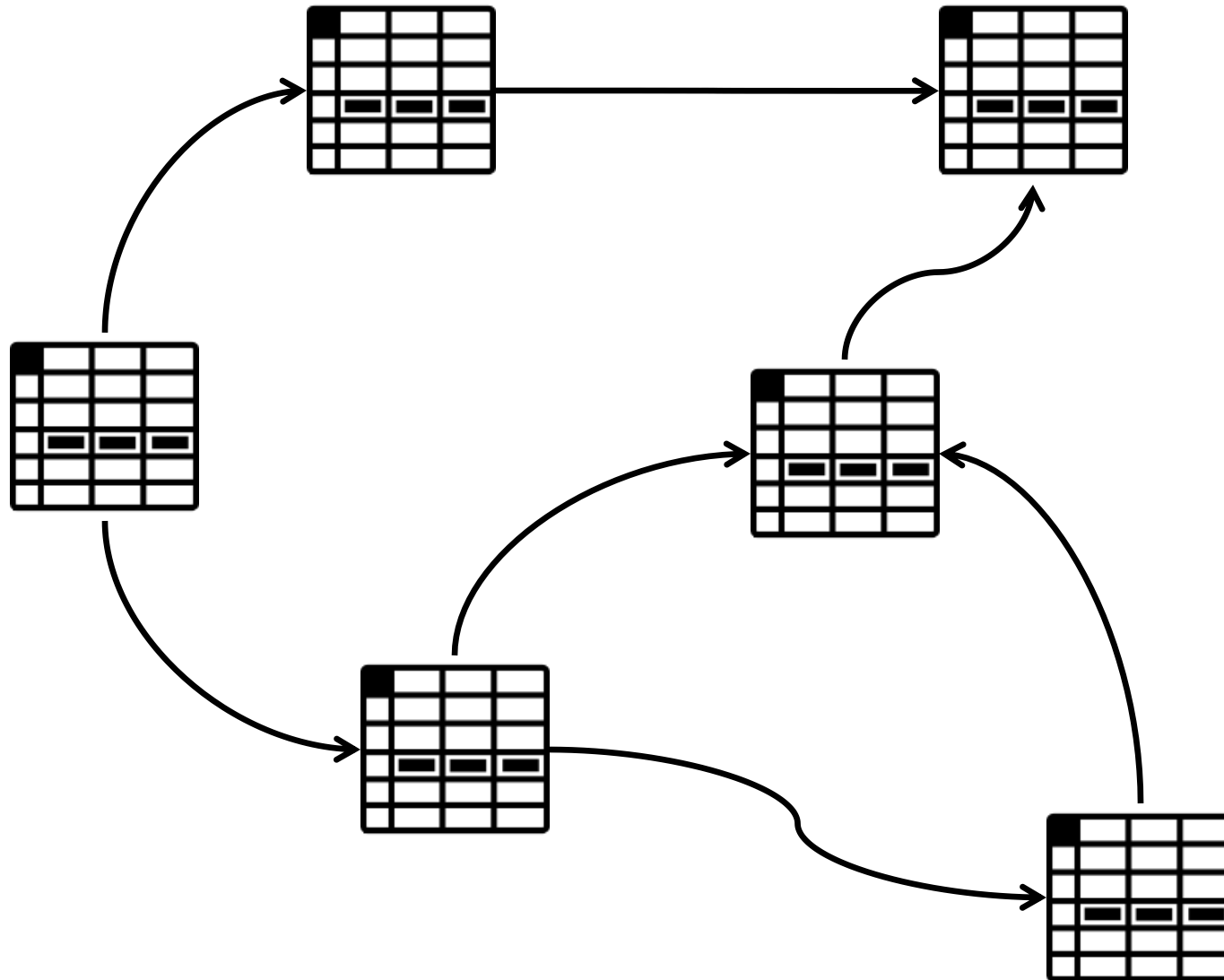


# Tabla de Datos

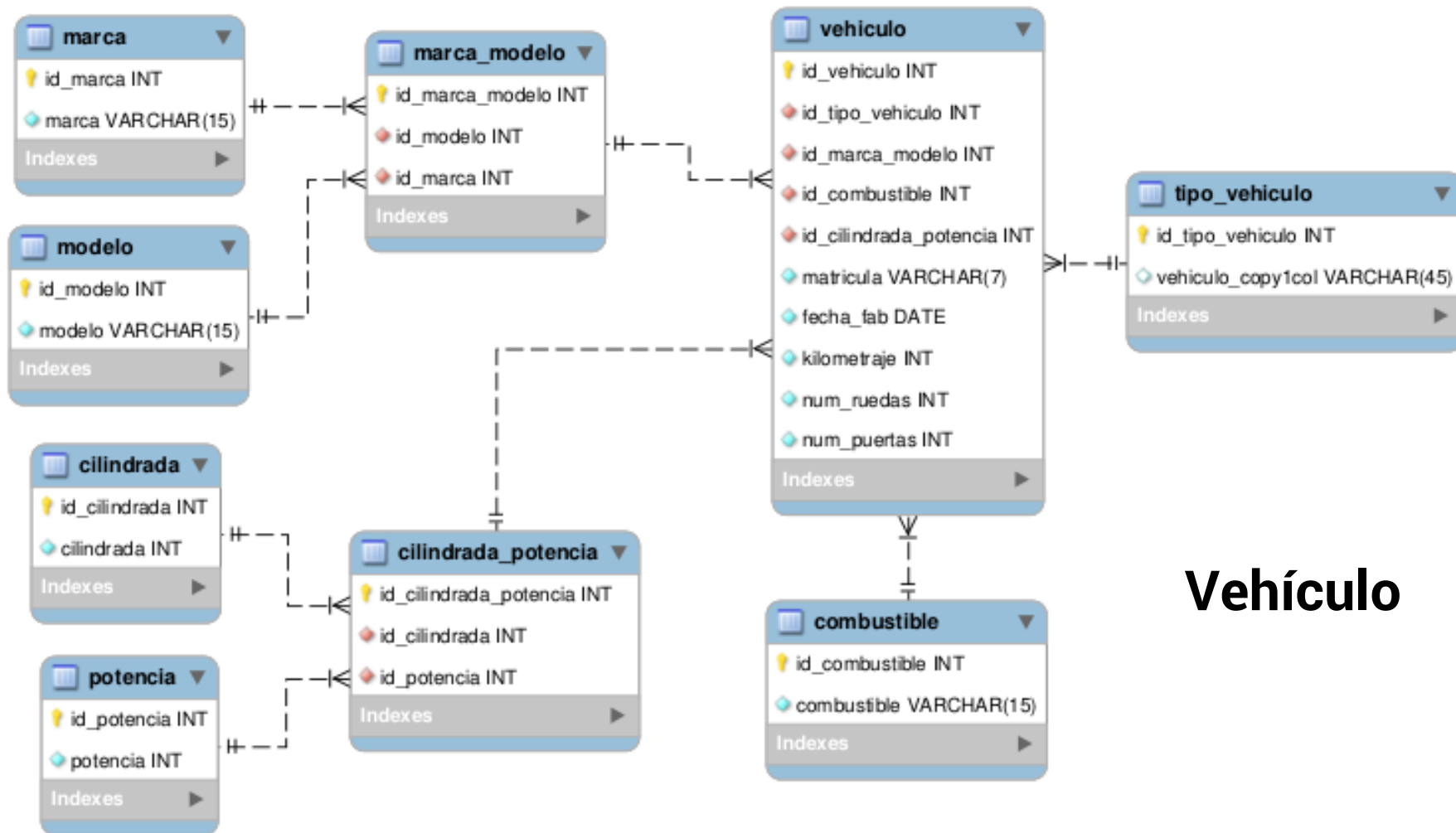

...





Bases de datos relacionales



**Vehículo**

Bases de Datos Relacionales

**Bases de Datos No Relacionales**

Comparativa entre Modelos de Datos

Introducción a MongoDB

Principales elementos de MongoDB

Iniciando MongoDB

Operaciones CRUD en MongoDB

Replicación de datos

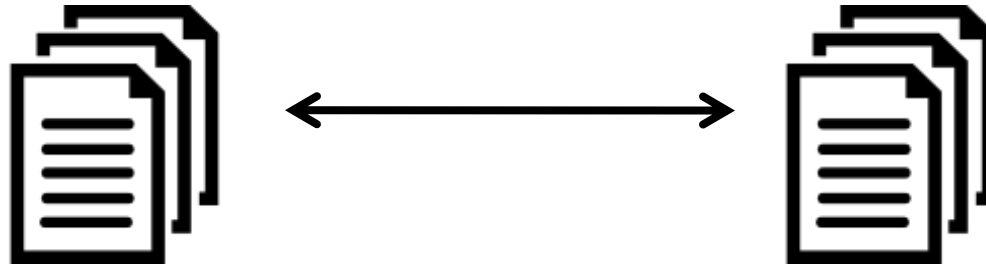
Sharding de datos



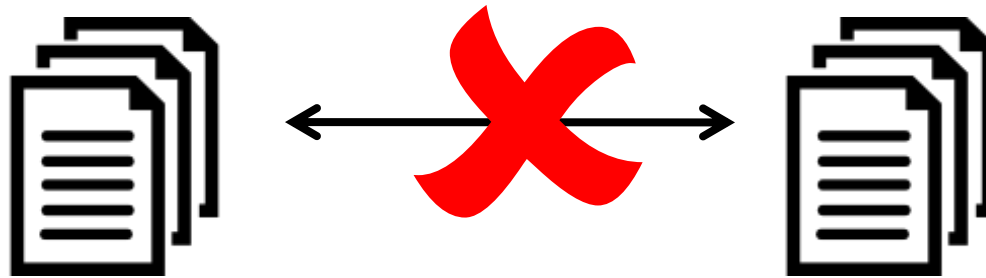




# ¿Relaciones?

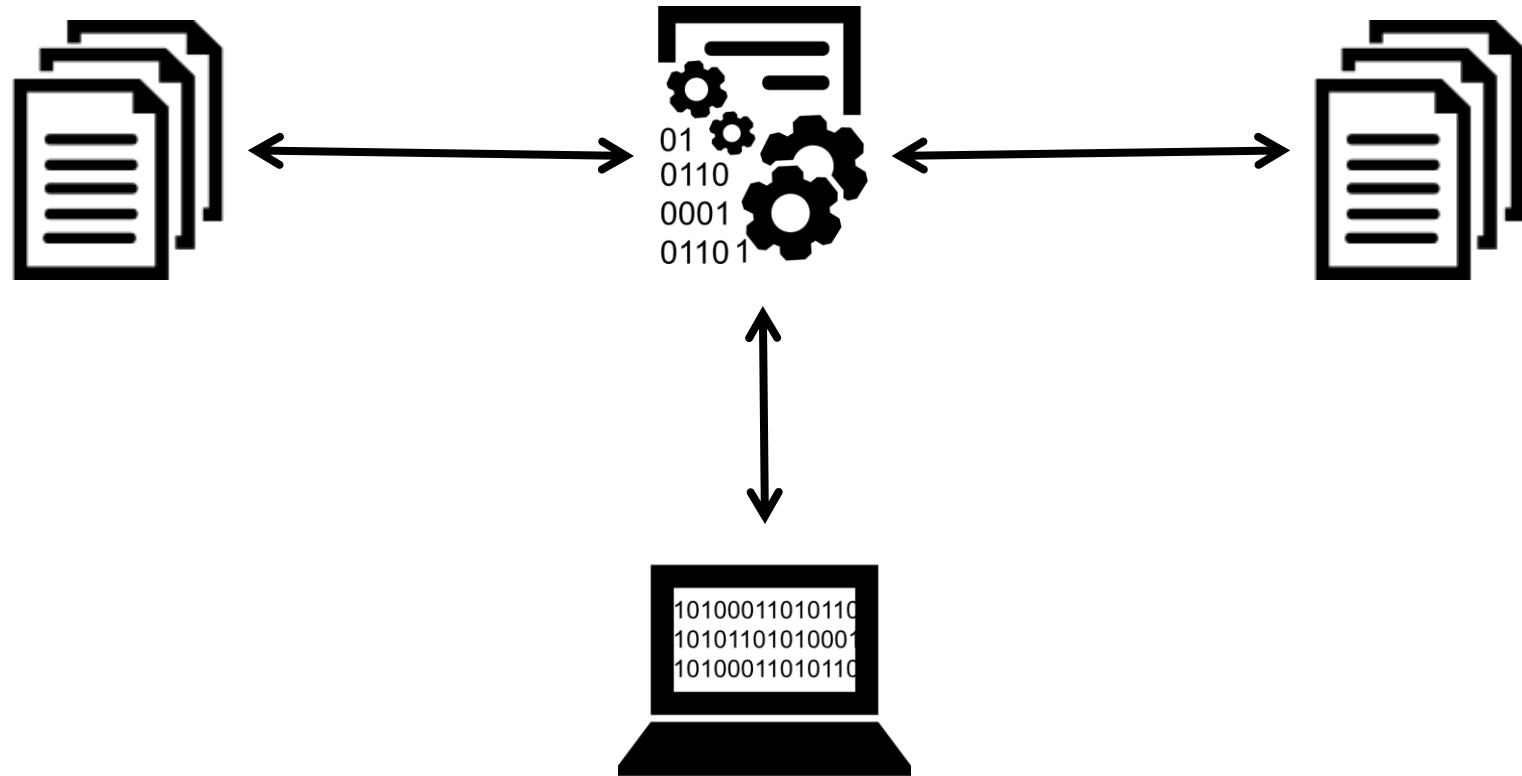


# ¿Relaciones?



Bases de datos no relacionales

# ¿Relaciones?



Bases de datos no relacionales



Bases de Datos Relacionales

Bases de Datos No Relacionales

**Comparativa entre Modelos de Datos**

Introducción a MongoDB

Principales elementos de MongoDB

Iniciando MongoDB

Operaciones CRUD en MongoDB

Replicación de datos

Sharding de datos



¿Cuándo usar un modelo u otro?

# Teorema CAP



Comparativa entre modelos de datos

# Teorema CAP

## **C**onsistency (Consistencia de los datos)

Capacidad para que todos los clientes de la base de datos tengan la misma perspectiva de los datos, en todo momento.

## **A**vailability (Disponibilidad de los datos)

Capacidad para que todos los clientes de la base de datos puedan leer y escribir datos.

## **P**artition Tolerance (Distribución de los datos)

Capacidad para que cualquier sistema pueda continuar funcionando aunque los datos estén repartidos entre varios servidores.



Comparativa entre modelos de datos

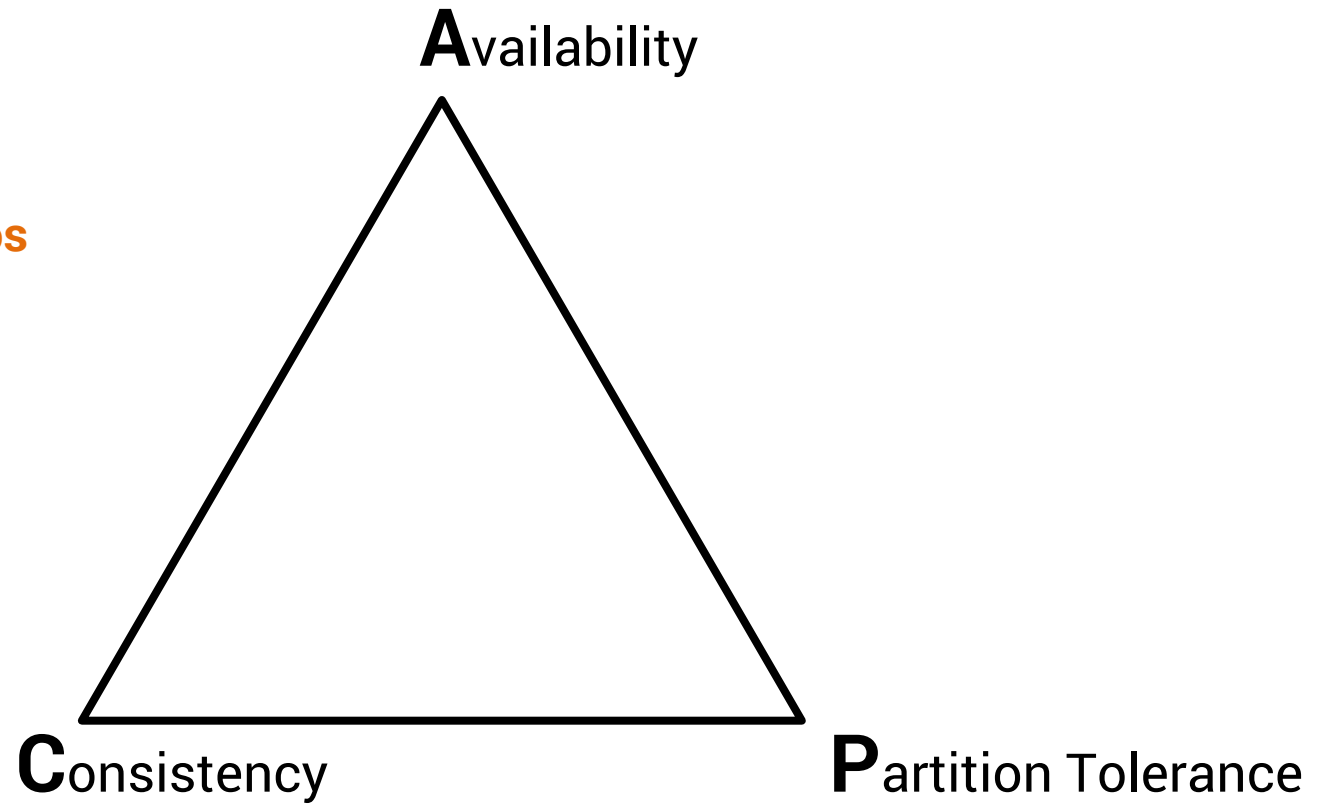
# Modelos de Datos

Relacional

Clave-Valor

Orientado a Columnas

Orientado a Documentos



Comparativa entre modelos de datos

# Modelos de Datos

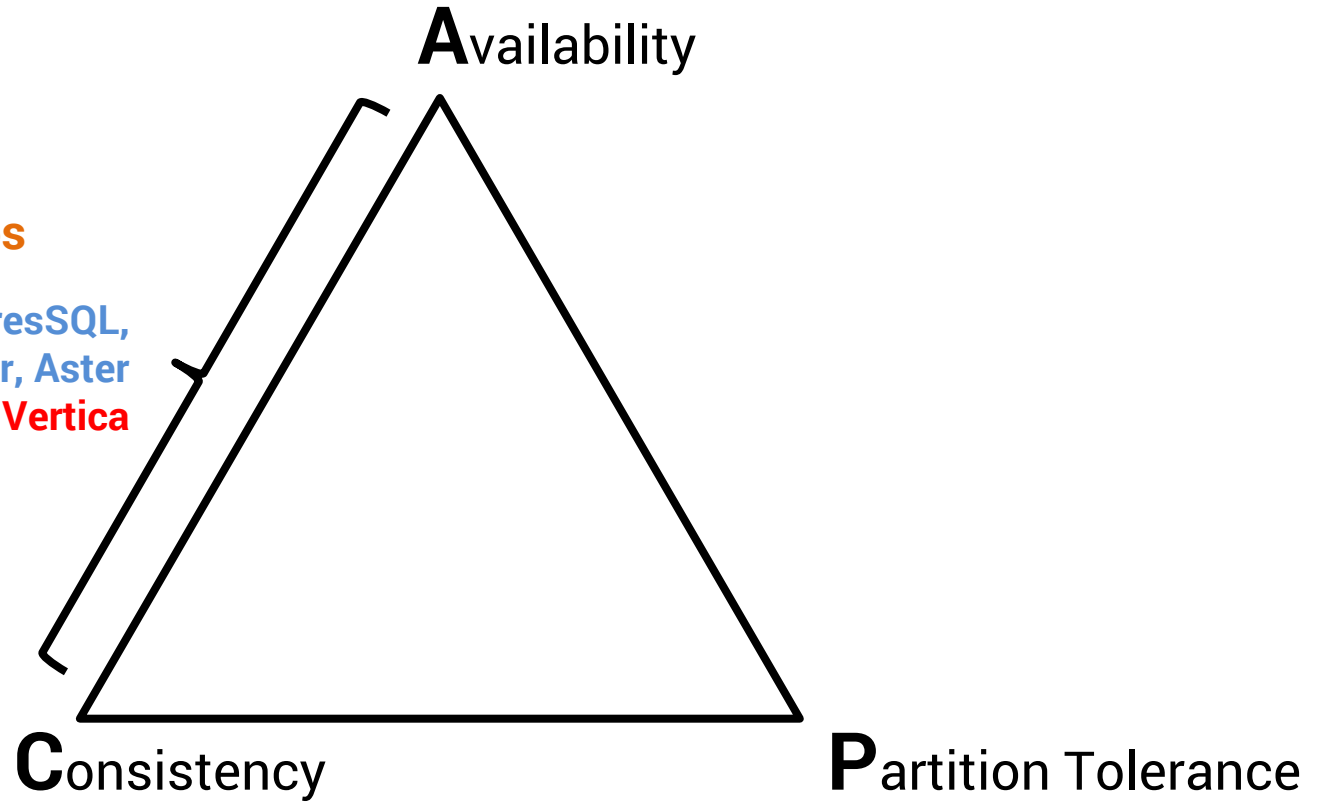
Relacional

Clave-Valor

Orientado a Columnas

Orientado a Documentos

MySQL, PostgreSQL,  
MS SQL Server, Aster  
Data, Greenplum, **Vertica**



Comparativa entre modelos de datos

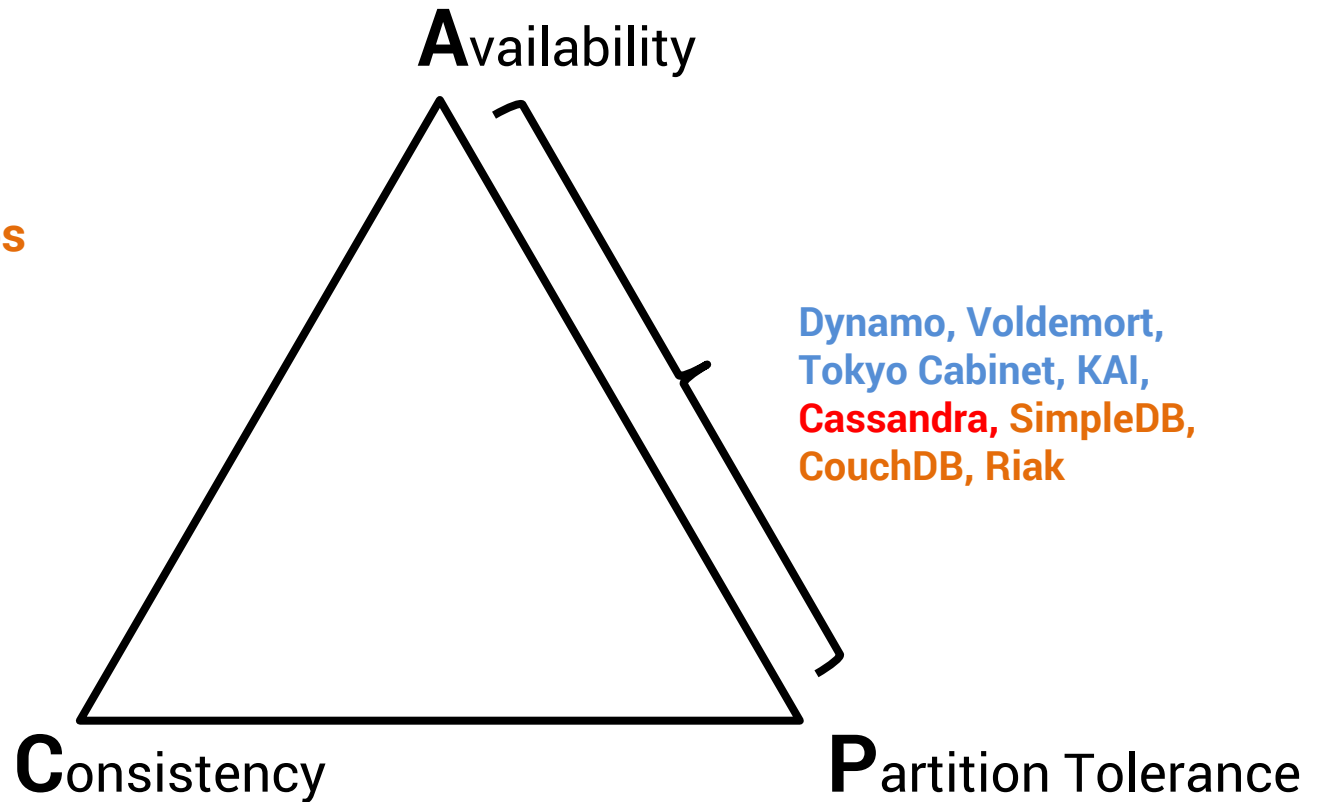
# Modelos de Datos

Relacional

Clave-Valor

Orientado a Columnas

Orientado a Documentos



Comparativa entre modelos de datos

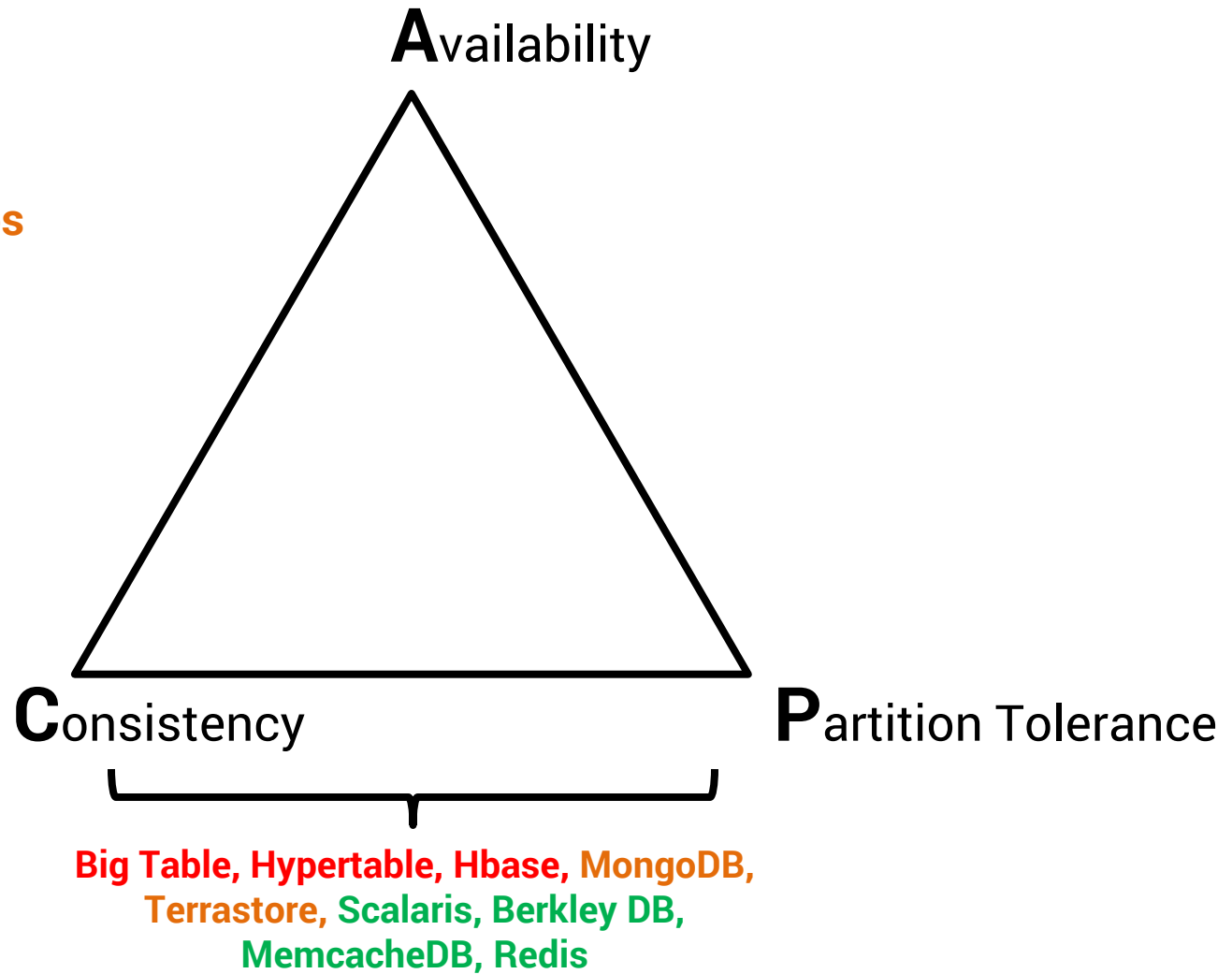
# Modelos de Datos

Relacional

Clave-Valor

Orientado a Columnas

Orientado a Documentos



Comparativa entre modelos de datos

# Modelos de Datos

Relacional

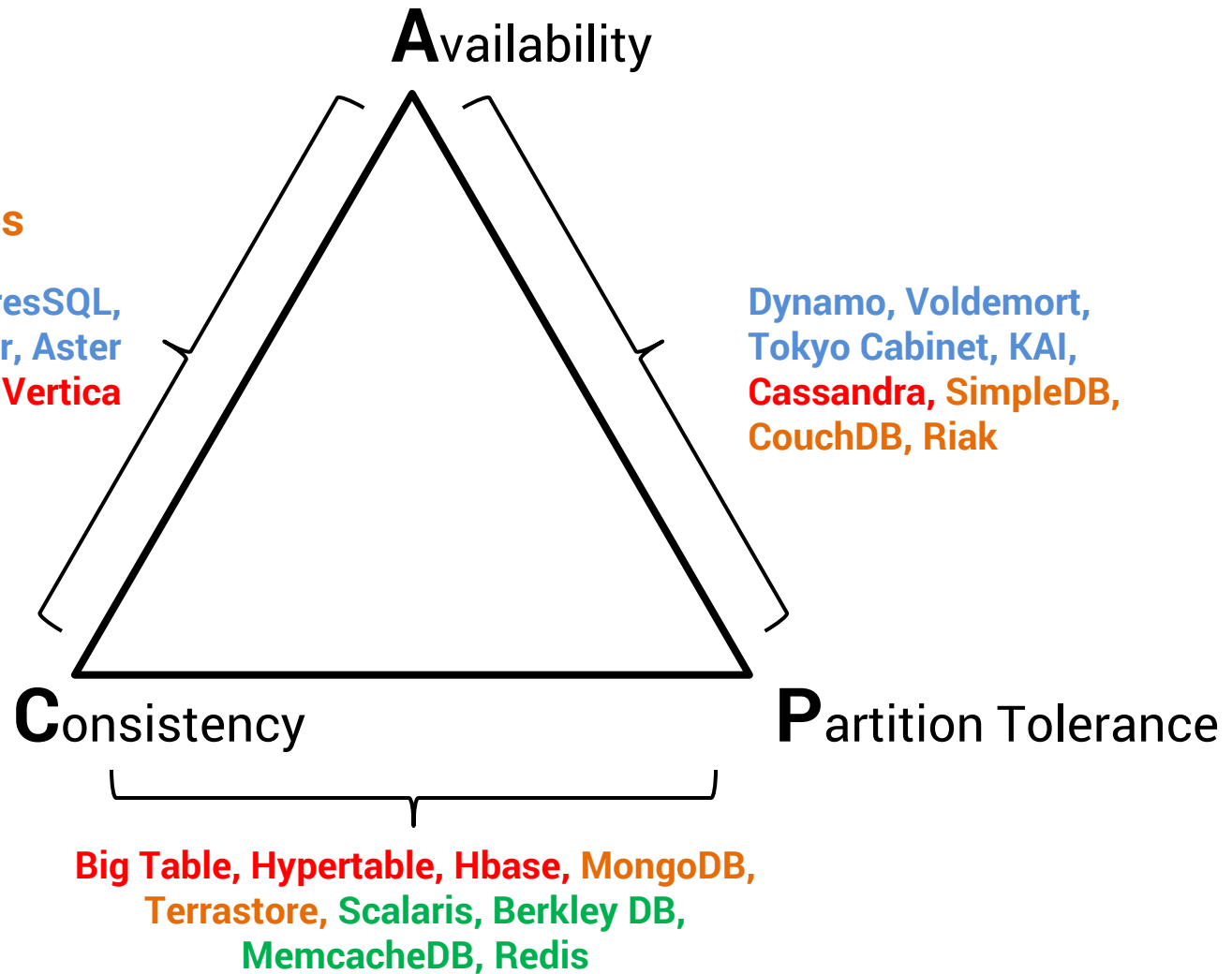
Clave-Valor

Orientado a Columnas

Orientado a Documentos

MySQL, PostgreSQL,  
MS SQL Server, Aster  
Data, Greenplum, **Vertica**

Dynamo, Voldemort,  
Tokyo Cabinet, KAI,  
**Cassandra**, **SimpleDB**,  
**CouchDB**, Riak



Comparativa entre modelos de datos



# Semejanzas entre Relacional y No Relacional

## Relacional



## No Relacional




Base de Datos ↔ Base de Datos

Tabla ↔ Colección

Registro ↔ Documento

Índice ↔ Índice

Procedimiento Almacenado ↔ Definido en aplicación



Comparativa entre modelos de datos

Bases de Datos Relacionales

Bases de Datos No Relacionales

Comparativa entre Modelos de Datos

**Introducción a MongoDB**

Principales elementos de MongoDB

Iniciando MongoDB

Operaciones CRUD en MongoDB

Replicación de datos

Sharding de datos



# ¿Qué es mongoDB ?

Gestor de bases de datos Open-Source.

Escrito en C++.

Basado en el almacenamiento de documentos.

Desarrollado por la empresa **10gen** en Octubre de 2007.

Organizado como un sistema de ficheros.

Acceso a elementos a través de “Dot Notation”.

Sistema multi-plataforma operativo para



# Principales propiedades de mongoDB

## Consultas Ad Hoc

Operaciones específicas para situaciones específicas, realizando consultas por campo, rangos de valores y Expresiones Regulares (RegExp) las cuales, devuelven campos específicos y grupos de documentos.

## Indexación sobre colecciones

Para acelerar operaciones CRUD se pueden llegar a crear hasta **40** índices por colección.

## Balanceado de Carga

La replicación de datos entre diferentes servidores permite asegurar el acceso a los datos y la creación automática de backups.

Mediante el Sharding de datos, podemos distribuir la carga de peticiones entre los distintos servidores.



Bases de Datos Relacionales

Bases de Datos No Relacionales

Comparativa entre Modelos de Datos

Introducción a MongoDB

**Principales elementos de MongoDB**

Iniciando MongoDB

Operaciones CRUD en MongoDB

Replicación de datos

Sharding de datos





**Base de Datos**



**Documentos**



**Colección**



**Índices**



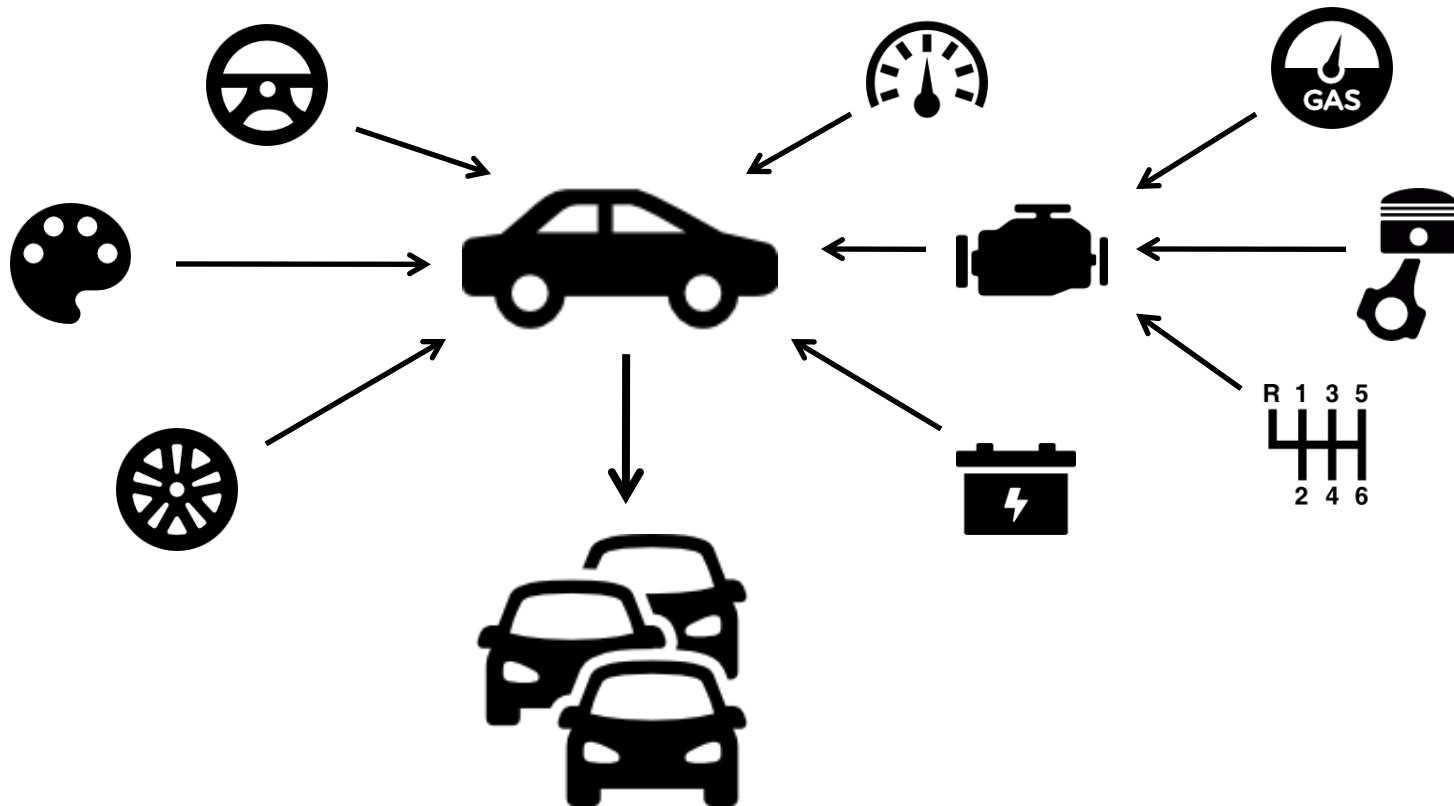
**Principales elementos de MongoDB**

# Documentos



## ¿Para qué se usan?

Almacenar toda la información relativa a un único elemento el cual, pertenece a un conjunto de datos relacionados con un mismo área.



Principales elementos de MongoDB

# Documentos



## JSON

JavaScript Object Notation

Normativa RFC 7159

Basado en un conjunto Calve-Valor

Tamaño máximo **16 MB**.

Tamaño Máximo

**16 MB**

```
{
  "posts": [
    {
      "_id": "ID del artículo",
      "author": "Autor del artículo",
      "date": "Fecha de publicación",
      "title": "Título del artículo",
      "body": "Contenido del artículo",
      "permalink": "Permalink al artículo",
      "tags": ["Tag 1", "Tag 2", "Tag 3", "Tag n"],
      "comments": [
        {
          "author": "Autor del comentario 1",
          "email": "Correo electrónico del autor",
          "body": "Contenido del comentario 1"
        }, {
          "author": "Autor del comentario 2",
          "email": "Correo electrónico del autor",
          "body": "Contenido del comentario 2"
        }
      ]
    }
  ]
}
```



## Principales elementos de MongoDB



# Índices



```
{
  "users": [
    {
      → "_id": "ID del usuario 1",
        "username": "Nombre del usuario 1",
        "password": "Contraseña del usuario 1"
    }, {
      → "_id": "ID del usuario 2",
        "username": "Nombre del usuario 2",
        "password": "Contraseña del usuario 2"
    }
  ]
}
```



# Índices



## De Campo Simple

Destinados a organizar la información en base a un único parámetro de información.

## De Campo Múltiple ( Multi-Key )

Agrupación de campos simples para facilitar consultas complejas.

## Geoespaciales

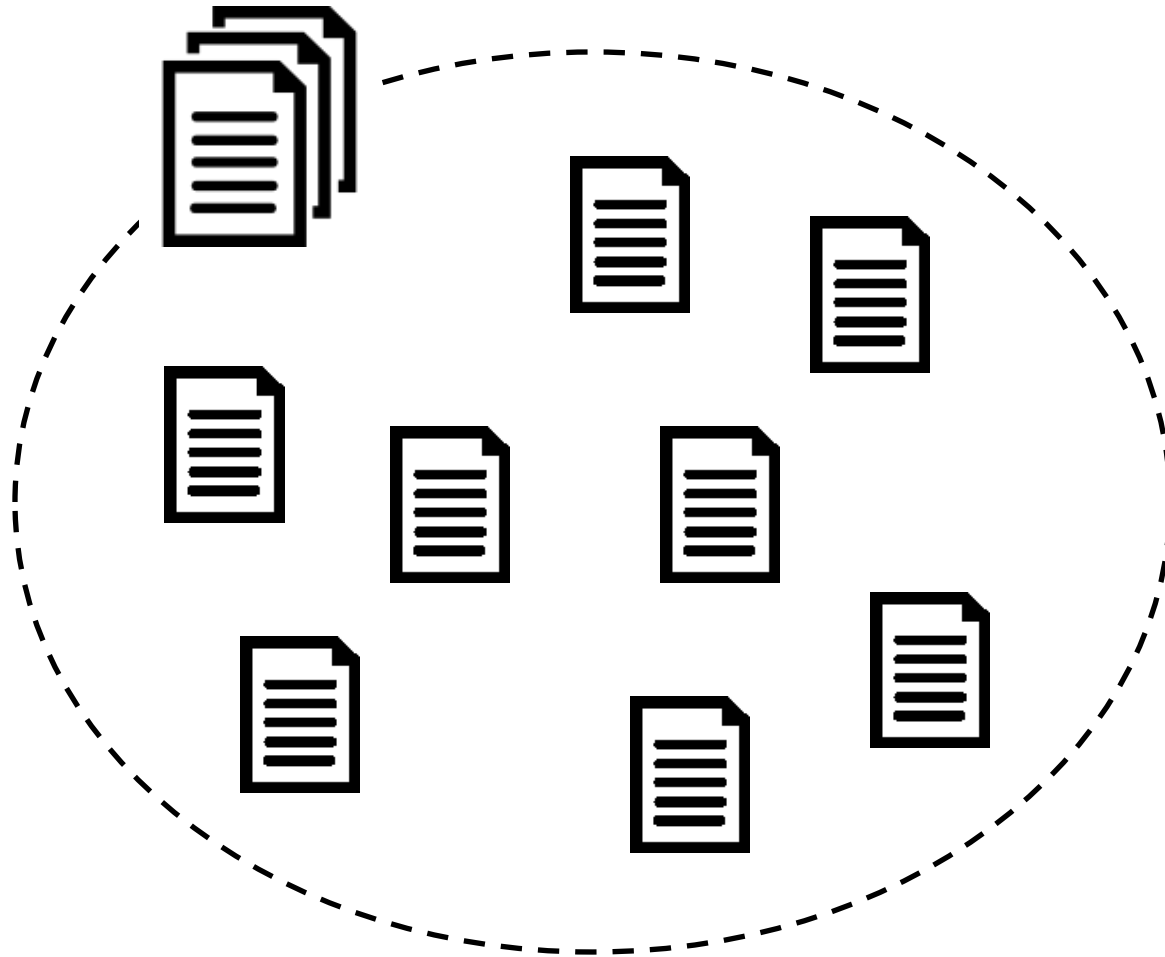
Permite realizar búsquedas en base a valores espaciales y de posicionamiento mediante datos **GeoJSON**.

## De Texto

Permite realizar búsquedas dentro de campos que contenga un gran volumen de texto.

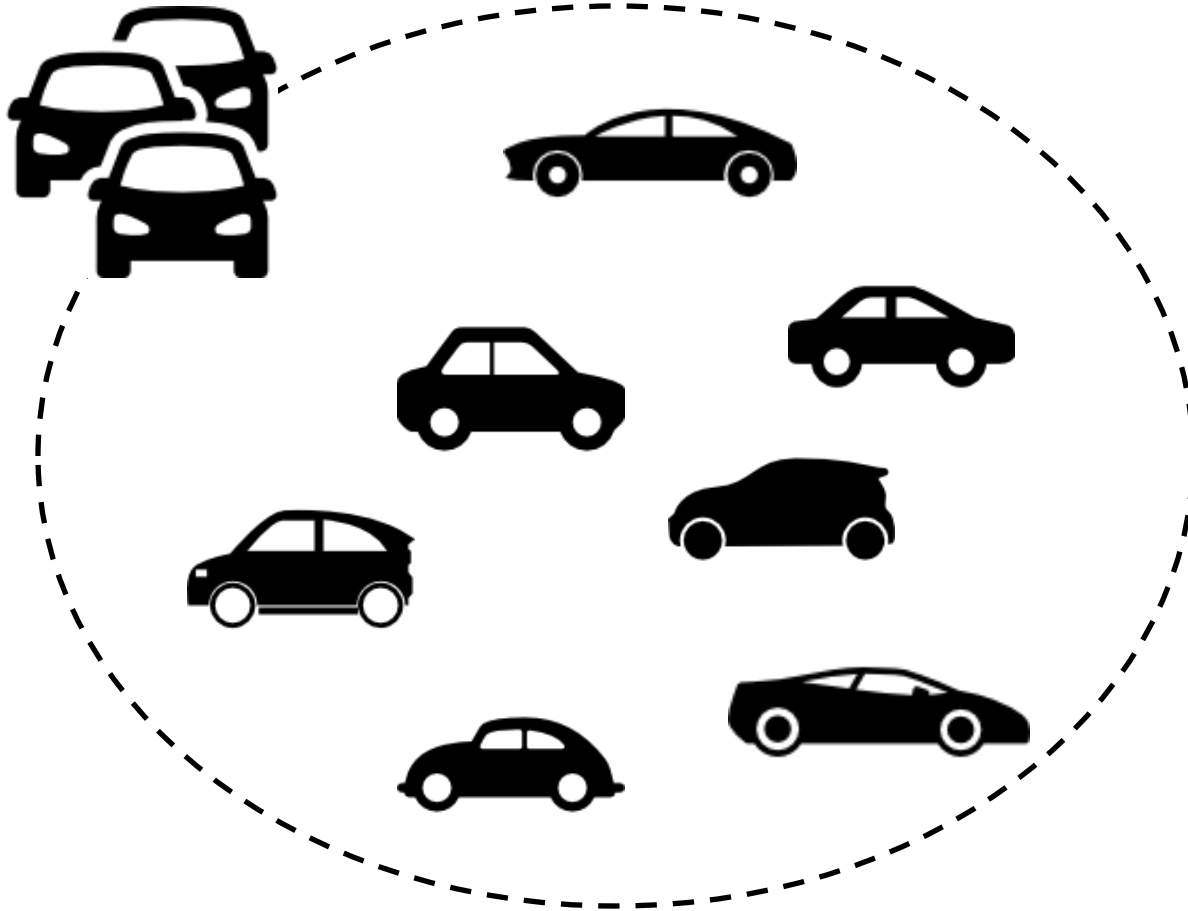


# Colecciones

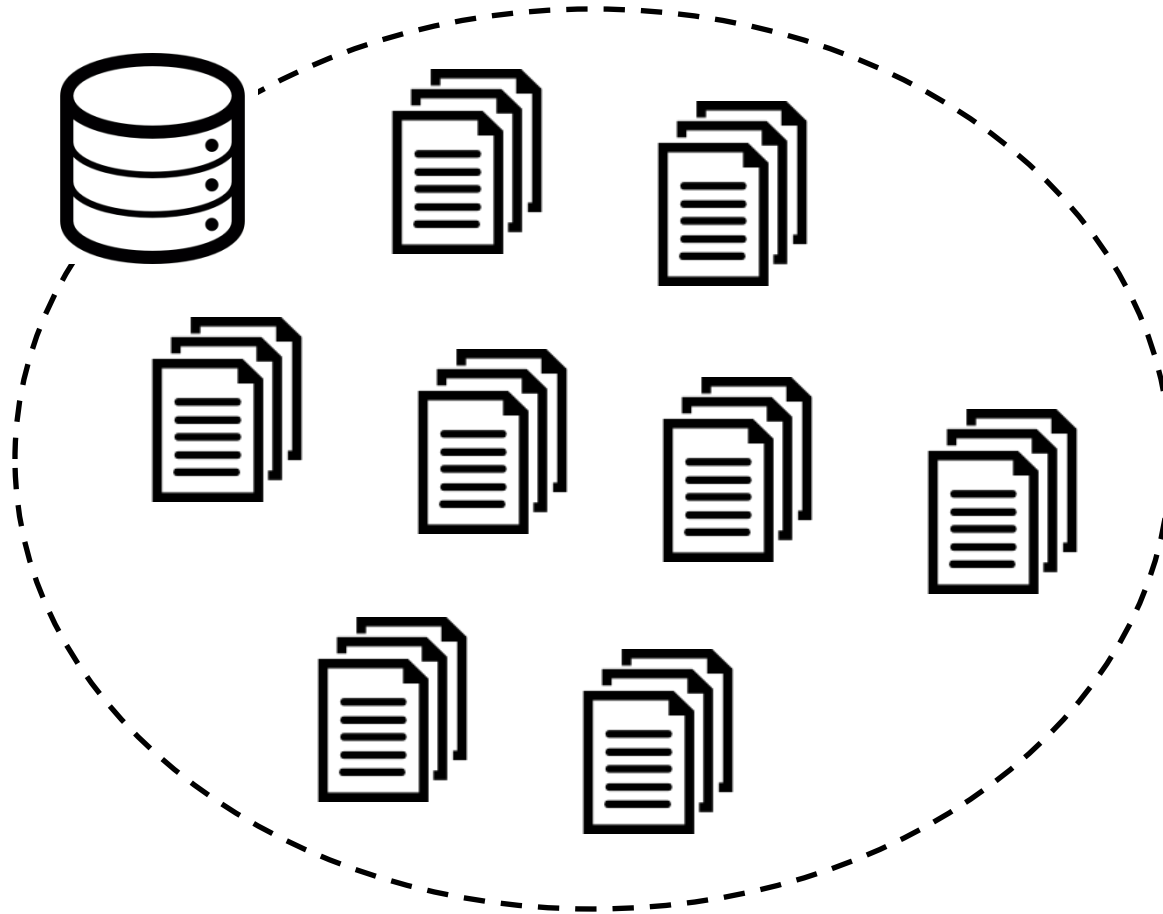


Principales elementos de MongoDB

# Colecciones



Principales elementos de MongoDB



# Bases de Datos



 mongoDB  
32 bits

**$\leq 2$  GB**



 mongoDB  
64 bits

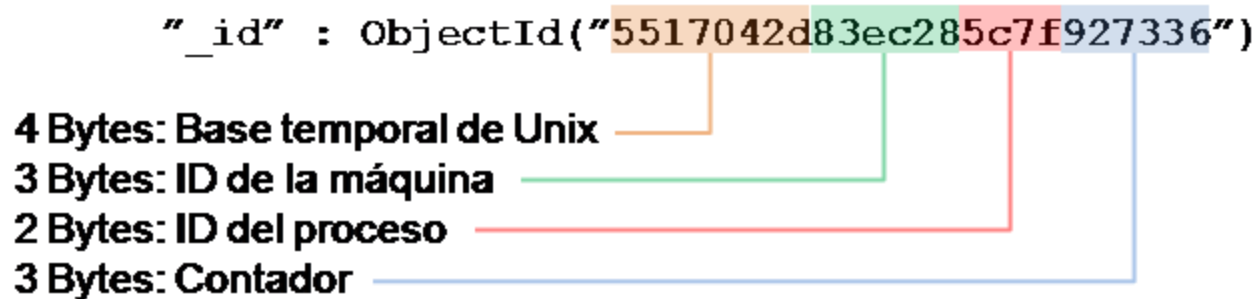
**Sin Límites**



 Principales elementos de MongoDB

# Tipos de Datos

**ObjectId** Dato por defecto asignado a los campos `_id`. Valor numérico de **12 Bytes**.



**Números** Enteros de 32 y 64 bits así como Decimales (Double).

**Cadenas de caracteres**

**Booleanos**

**Fecha/Hora** Tipos Date y Timestamp.

**Null**



Bases de Datos Relacionales

Bases de Datos No Relacionales

Comparativa entre Modelos de Datos

Introducción a MongoDB

Principales elementos de MongoDB

**Iniciando MongoDB**

Operaciones CRUD en MongoDB

Replicación de datos

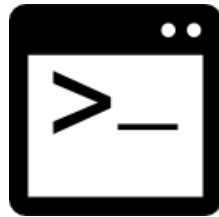
Sharding de datos







**mongod**



**mongo**



**mongos**



# mongod



Instancia del servidor de bases de datos.

**\$ mongod --dbpath <path> --port <puerto>**

## Parámetros

Path por defecto **/data/db**

Puerto de conexión **27017**

```
$ mongod --dbpath ./
2015-05-10T14:02:48.907+0000 I CONTROL 32-bit servers don't have journaling enabled by default. Please use --journal if you want durability.
2015-05-10T14:02:48.909+0000 I CONTROL Hotfix KB2731284 or later update is not installed, will zero-out data files
2015-05-10T14:02:48.960+0000 I CONTROL [initandlisten] MongoDB starting : pid=9596 port=27017 dbpath=./ 32-bit host=LaraStation
2015-05-10T14:02:48.961+0000 I CONTROL [initandlisten]
2015-05-10T14:02:48.961+0000 I CONTROL [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
2015-05-10T14:02:48.961+0000 I CONTROL [initandlisten] ** 32 bit builds are limited to less than 2 GB of data (or less with --journal).
2015-05-10T14:02:48.962+0000 I CONTROL [initandlisten] ** Note that journaling defaults to off for 32 bit and is currently off.
2015-05-10T14:02:48.962+0000 I CONTROL [initandlisten] ** See http://dochub.mongodb.org/core/32bit
2015-05-10T14:02:48.962+0000 I CONTROL [initandlisten]
2015-05-10T14:02:48.962+0000 I CONTROL [initandlisten] targetMinOS: Windows XP SP3
2015-05-10T14:02:48.962+0000 I CONTROL [initandlisten] db version v3.0.1
2015-05-10T14:02:48.962+0000 I CONTROL [initandlisten] git version: 534b5a3f9d10f00cd27737fbcd951032248b5952
2015-05-10T14:02:48.963+0000 I CONTROL [initandlisten] build info: windows sys.getwindowsversion(major=6, minor=1, build=7601, platform=2, service_pack='Service Pack 1') BOOST_LIB_VERSION=1_49
2015-05-10T14:02:48.963+0000 I CONTROL [initandlisten] allocator: system
2015-05-10T14:02:48.963+0000 I CONTROL [initandlisten] options: { storage: { dbPath: "./" } }
2015-05-10T14:02:49.047+0000 I NETWORK [initandlisten] waiting for connections on port 27017
```

```
2015-05-10T14:02:49.047+0000 I NETWORK [initandlisten] waiting for connections on port 27017
```



## Iniciando MongoDB

# mongod



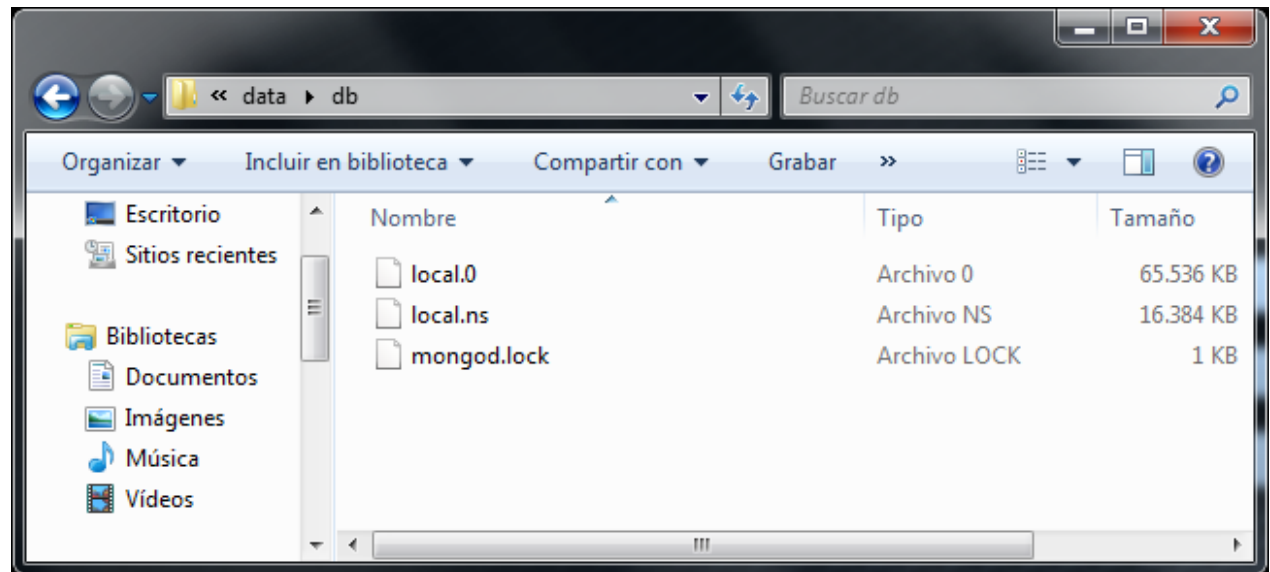
Instancia del servidor de bases de datos.

```
$ mongod --dbpath <path> --port <puerto>
```

## Parámetros

Path por defecto  
**/data/db**

Puerto de conexión  
**27017**



## Iniciando MongoDB

# mongo



Consola de comandos de MongoDB

```
$ mongo [url:port/db]
```

```
$ mongo
2015-05-10T14:01:18.971+0000 I CONTROL   Hotfix KB2731284 or later update is not installed, will zero-out
data files
MongoDB shell version: 3.0.1
connecting to: test
Server has startup warnings:
2015-05-10T08:31:17.902+0000 I CONTROL   [initandlisten]
2015-05-10T08:31:17.902+0000 I CONTROL   [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
2015-05-10T08:31:17.903+0000 I CONTROL   [initandlisten] **       32 bit builds are limited to less than 2
GB of data (or less with --journal).
2015-05-10T08:31:17.903+0000 I CONTROL   [initandlisten] **       Note that journaling defaults to off for
32 bit and is currently off.
2015-05-10T08:31:17.903+0000 I CONTROL   [initandlisten] **       See http://dochub.mongodb.org/core/32bit
2015-05-10T08:31:17.904+0000 I CONTROL   [initandlisten]
>
```



## Iniciando MongoDB

# mongo



Consola de comandos de MongoDB

```
$ mongo [url:port/db]
```

```
2015-05-10T14:02:48.963+0000 I CONTROL [initandlisten] build info: windows sys.getwindowsversion(major=6
, minor=1, build=7601, platform=2, service_pack='Service Pack 1') BOOST_LIB_VERSION=1_49
2015-05-10T14:02:48.963+0000 I CONTROL [initandlisten] allocator: system
2015-05-10T14:02:48.963+0000 I CONTROL [initandlisten] options: { storage: { dbPath: "./" } }
2015-05-10T14:02:49.047+0000 I NETWORK [initandlisten] waiting for connections on port 27017
2015-05-10T14:05:45.857+0000 I NETWORK [initandlisten] connection accepted from 127.0.0.1:31860 #1 (1 co
nnection now open)
```

( Consola del proceso **mongod** )



## Iniciando MongoDB

# mongo



Consola de comandos de MongoDB

```
$ mongo [url:port/db]
```



JavaScript



Iniciando MongoDB

# show databases



Lista todas las bases de datos presentes en el servidor.

**> show databases**

```
> show databases
contactos      0.078GB
institutos     0.078GB
local          0.078GB
test           0.078GB
>
> db
test
>
```



# use <database>



Conectamos con una determinada base de datos o la creamos si no existe.

```
> use <database>
```

```
> db
test
>
> use contactos
switched to db contactos
>
```





# show collections



Muestra las colecciones que contiene una determinada base de datos.

```
> show collections
```

```
> use contactos  
switched to db contactos  
>  
> show collections  
system.indexes  
trabajo  
>
```



# mongoimport



**NOTA:** Debe emplearse fuera de la consola de MongoDB.

Permite importar datos JSON a una determinada base de datos.

```
$ mongoimport -d <database> -c <collection> < archivo.json
```

```
$ mongoimport -d contactos -c trabajo < contactos.json
2015-05-12T05:23:54.284+0000    connected to: localhost
2015-05-12T05:23:55.334+0000    imported 10 documents
```

Código disponible en

<https://github.com/ddialar/MongoDB-101-Introduccion>



## Iniciando MongoDB

# show collections



```
> show collections
```

```
> use contactos  
switched to db contactos  
>  
> show collections  
system.indexes  
→ trabajo  
>
```



# help



Proporciona información sobre las funcionalidades disponibles.

`> db.help()`

```
> db.help()
DB methods:
  db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [ just calls db.runCom
mand(...) ]
  db.auth(username, password)
  db.cloneDatabase(fromhost)
  db.commandHelp(name) returns the help for the command
  db.copyDatabase(fromdb, todb, fromhost)
  db.createCollection(name, { size : ..., capped : ..., max : ... } )
  db.createUser(userDocument)
  db.currentOp() displays currently executing operations in the db
  db.dropDatabase()
  db.eval(func, args) run code server-side
  db.fsyncLock() flush data to disk and lock server for backups
  db.fsyncUnlock() unlocks server following a db.fsyncLock()
  db.getCollection(cname) same as db['cname'] or db.cname
  db.getCollectionInfos()
  db.getCollectionNames()
  db.getLastError() - just returns the err msg string
```



Bases de Datos Relacionales

Bases de Datos No Relacionales

Comparativa entre Modelos de Datos

Introducción a MongoDB

Principales elementos de MongoDB

Iniciando MongoDB

**Operaciones CRUD en MongoDB**

Replicación de datos

Sharding de datos





## Create

Insertar nuevos documentos dentro de una colección.



## Read

Consultas basadas en un conjunto de parámetros específicos.



## Update

Modificación y/o actualización de los datos ya almacenados.



## Delete

Eliminación de datos almacenados.



# Create

Método insert()



**> db.<collection>.insert(<JSON>)**

```
> db
contactos
>
> show collections
system.indexes
trabajo
>
> db.trabajo.insert({"name":"John","surname":"Smith","age":"26"})
WriteResult({ "nInserted" : 1 })
>
```



Operaciones CRUD en MongoDB

# Read

## Métodos findOne() y find()



```
> db.<collection>.findOne([criterio],[proyección])
```

### **criterio**

Documento JSON que nos permitirá establecer el filtro de búsqueda a utilizar en la consulta.

### **proyección**

Documento JSON que nos permitirá definir de qué campos de los resultantes de la búsqueda, serán entregados por el método.





# Read

## Métodos findOne() y find()



> db.<collection>.findOne()

> use contactos

> db.trabajo.findOne()

```
> db.trabajo.findOne()
{
  "_id" : 0,
  "name" : "Hipolito",
  "surname" : "Demuth",
  "email" : "h.dem@gmail.com",
  "pass" : "NtT4PT2C",
  "phone" : "202-555-0114",
  "hobbies" : [
    "tenis",
    "escalada",
    "jardinería"
  ]
}
```



# Read

## Métodos findOne() y find()



> db.<collection>.findOne(<criterio>)

> db.terrestres.findOne({"name":"Anton"})

```
> db.trabajo.findOne({"name":"Anton"})
{
  "_id" : 9,
  "name" : "Anton",
  "surname" : "Pearman",
  "email" : "a.pea@gmail.com",
  "pass" : "m7cApeX8",
  "phone" : "202-555-0180",
  "hobbies" : [
    "tenis",
    "escalada",
    "jardinería"
  ]
}
```



# Read

## Métodos findOne() y find()



```
> db.<collection>.findOne(<criteria>,<proyección>)
```

```
> use gasolinas
```

```
> db.terrestres.findOne({"name":"Anton"}, {"_id":0,"hobbies":1})
```

```
> db.trabajo.findOne({"name":"Anton"}, {"_id":0,"hobbies":1})  
{ "hobbies" : [ "tenis", "escalada", "jardinería" ] }
```



# Read

## Métodos findOne() y find()



```
> db.<collection>.find([criterio],[proyección])
```

### **criterio**

Documento JSON que nos permitirá establecer el filtro de búsqueda a utilizar en la consulta.

### **proyección**

Documento JSON que nos permitirá definir de qué campos de los resultantes de la búsqueda, serán entregados por el método.



# Read

## Métodos findOne() y find()



```
> db.<collection>.find()
```

```
> use contactos
```

```
> db.trabajo.find()
```

```
> db.trabajo.find().pretty()
```

```
> db.trabajo.find().toArray()
```



# Read

Métodos findOne() y find()



```
> db.<collection>.find(<criterio>)
```

```
> db.trabajo.find({"hobbies": "informática"})
```

Resultado: 4 contactos.



# Read

Métodos findOne() y find()



## Expresiones Regulares en el Criterio de búsqueda

```
> db.<collection>.find(<criterio>)
```

```
> db.trabajo.find({"name" : /^R/})
```

Validador de RegExp  
<http://rubular.com/>



# Read

## Métodos findOne() y find()



```
> db.<collection>.find(<critério>,<proyección>)
```

```
> db.trabajo.find(  
  {"name":/^R/},  
  {"_id":0,"name":1,"hobbies":1})
```

Resultado: Aficiones de Rosaria y Roselyn.







### ¿Cuántos resultados obtengo?

```
> db.<collection>.find(...).count()
```

```
> db.trabajo.find().count() → 11
```

```
> db.trabajo.find({"name": /^R/}).count() → 2
```



# Read

Métodos findOne() y find()



## Limitando el número de resultados obtenidos

```
> db.<collection>.find(...) .limit(<n>)
```

```
> db.trabajo.find({"hobbies":"tenis"}) → 4
```

```
> db.trabajo.find({"hobbies":"tenis"}) .limit(2) → 2
```





## Ordenando los resultados obtenidos

```
> db.<collection>.find(...) .sort(<campo:1|-1>)
```

```
> db.trabajo.find({}, {"_id":0,"name":1}) .sort({"name":1})
```



# Update

## Método update()



```
> db.<collection>.update (<critério>,  
                           <modificación>  
                           [,opciones])
```

### critério

Documento JSON que nos permitirá establecer el filtro de búsqueda a utilizar en la selección.

### modificación

Documento JSON donde indicamos qué vamos a modificar en los documentos obtenidos a través del criterio de búsqueda.

### opciones

Documento JSON que puede contener los siguientes comportamientos:

**upsert:true|FALSE** Estando a true, si el documento a modificar no existe, crea uno nuevo con esos parámetro.

**multi:true|FALSE** Estando a true, la modificación afecta a todos los documentos que coincidan con el criterio de búsqueda.



# Update

Método update()



## Introduciendo nuevos campos

```
> db.trabajo.update(  
  {"name": "John", "surname": "Smith"},  
  {"$set": {"email": "j.smi@gmail.com", "pass": "123"}},  
  {"upsert": false, "multi": false})
```



# Update

Método update()



## Introduciendo nuevos campos

**Antes**

```
{
  "_id" : ObjectId("554fb46845387691741fd4de"),
  "name" : "John",
  "surname" : "Smith",
  "age" : 26
}
```

**Después**

```
{
  "_id" : ObjectId("554fb46845387691741fd4de"),
  "name" : "John",
  "surname" : "Smith",
  "age" : 26,
  "email" : "j.smi@gmail.com",
  "pass" : "123"
}
```



# Update

Método update()



## Modificando el valor de un campo

```
> db.trabajo.update(  
  {"name": "John", "surname": "Smith"},  
  {"$set": {"pass": "123QWERTY"}},  
  {"upsert": false, "multi": false})
```



# Update

Método update()



## Modificando el valor de un campo

Antes

```
{
  "_id" : ObjectId("554fb46845387691741fd4de"),
  "name" : "John",
  "surname" : "Smith",
  "age" : 26,
  "email" : "j.smi@gmail.com",
  → "pass" : "123"
}
```

Después

```
{
  "_id" : ObjectId("554fb46845387691741fd4de"),
  "name" : "John",
  "surname" : "Smith",
  "age" : 26,
  "email" : "j.smi@gmail.com",
  → "pass" : "123QWERTY"
}
```





# Update

Método update()



## Incrementando el valor de un campo

```
> db.trabajo.update(  
  {"name":"John","surname":"Smith"},  
  {"$inc":{"age":10}},  
  {"upsert":false,"multi":false})
```



# Update

Método update()



## Incrementando el valor de un campo

**Antes**

```
{
  "_id" : ObjectId("554fb46845387691741fd4de"),
  "name" : "John",
  "surname" : "Smith",
  "age" : 26,
  "email" : "j.smi@gmail.com",
  "pass" : "123QWERTY"
}
```

**Después**

```
{
  "_id" : ObjectId("554fb46845387691741fd4de"),
  "name" : "John",
  "surname" : "Smith",
  "age" : 36,
  "email" : "j.smi@gmail.com",
  "pass" : "123QWERTY"
}
```



# Update

Método update()



## Principales operaciones que se pueden realizar

**Selectores  
de Campo**

- \$inc
- \$mul
- \$rename
- \$set
- \$unset
- \$min
- \$max
- \$currentDate

**Selectores  
de Array**

- \$addToSet
- \$pop
- \$pull
- \$push
- \$each
- \$slice
- \$sort
- \$position



# Delete

Método remove()



```
> db.<collection>.remove(<criterio>[,opciones])
```

## **criterio**

Documento JSON que nos permitirá establecer el filtro de búsqueda a utilizar en la selección.

## **opciones**

Documento JSON que puede contener el siguiente comportamiento:

**justOne:true|FALSE** Estando a true, en caso de que el criterio de selección coincida con múltiples documentos almacenados, sólo se eliminará el primero de ellos.



# Delete

Método remove()



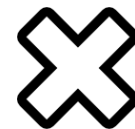
**Eliminar a todos los contactos cuyo nombre empieza por R**

```
> db.trabajo.remove(  
  {"name" : /^R/} ,  
  {"justOne" : false})
```



# Delete

Método remove()



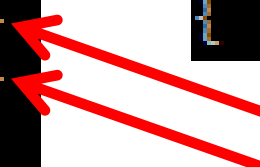
**Eliminar a todos los contactos cuyo nombre empieza por R**

**Antes**

```
{ "name" : "Alonzo" }  
{ "name" : "Anton" }  
{ "name" : "Boyce" }  
{ "name" : "Chanda" }  
{ "name" : "Florine" }  
{ "name" : "Hipolito" }  
{ "name" : "John" }  
{ "name" : "Phyllis" }  
{ "name" : "Rosaria" }  
{ "name" : "Roselyn" }  
{ "name" : "Shelby" }
```

**Después**

```
{ "name" : "Alonzo" }  
{ "name" : "Anton" }  
{ "name" : "Boyce" }  
{ "name" : "Chanda" }  
{ "name" : "Florine" }  
{ "name" : "Hipolito" }  
{ "name" : "John" }  
{ "name" : "Phyllis" }  
{ "name" : "Shelby" }
```



**Operaciones CRUD en MongoDB**

Bases de Datos Relacionales

Bases de Datos No Relacionales

Comparativa entre Modelos de Datos

Introducción a MongoDB

Principales elementos de MongoDB

Iniciando MongoDB

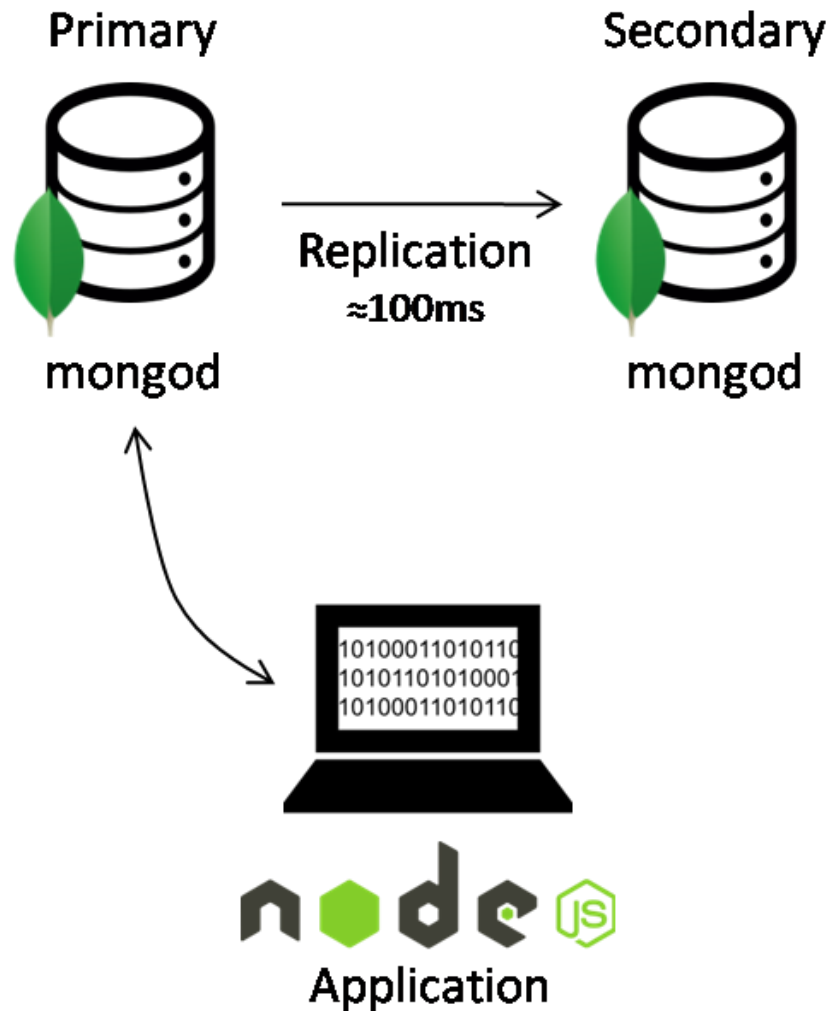
Operaciones CRUD en MongoDB

**Replicación de Datos**

Sharding de datos

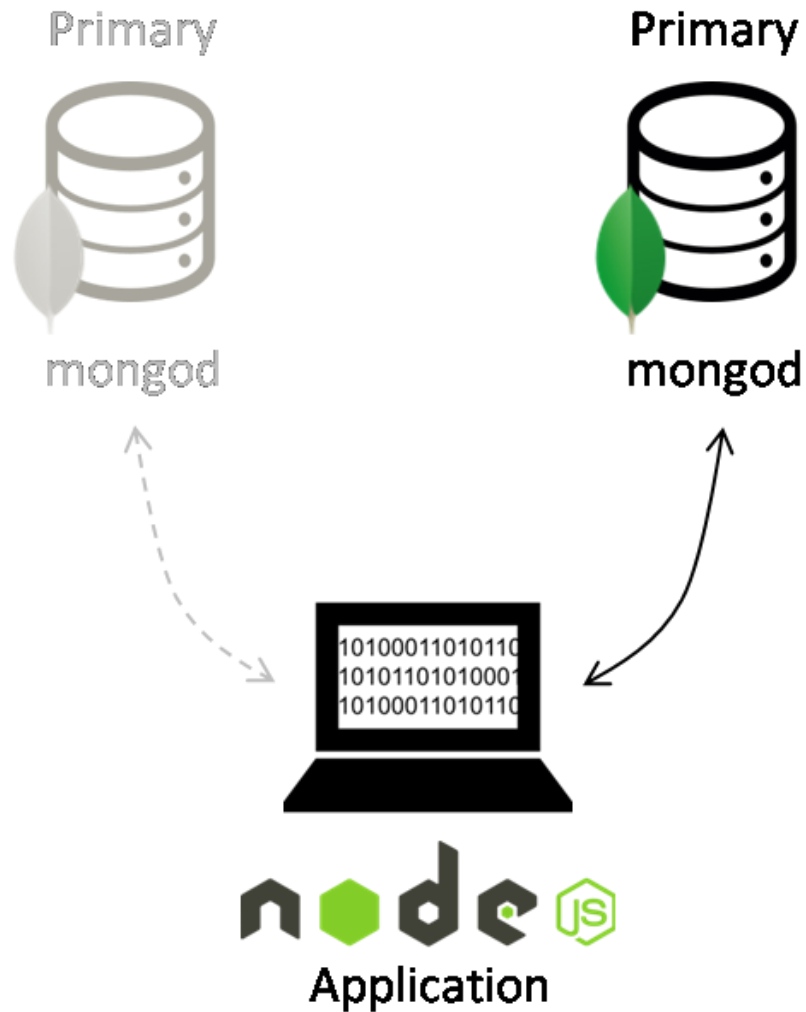


# Concepto General



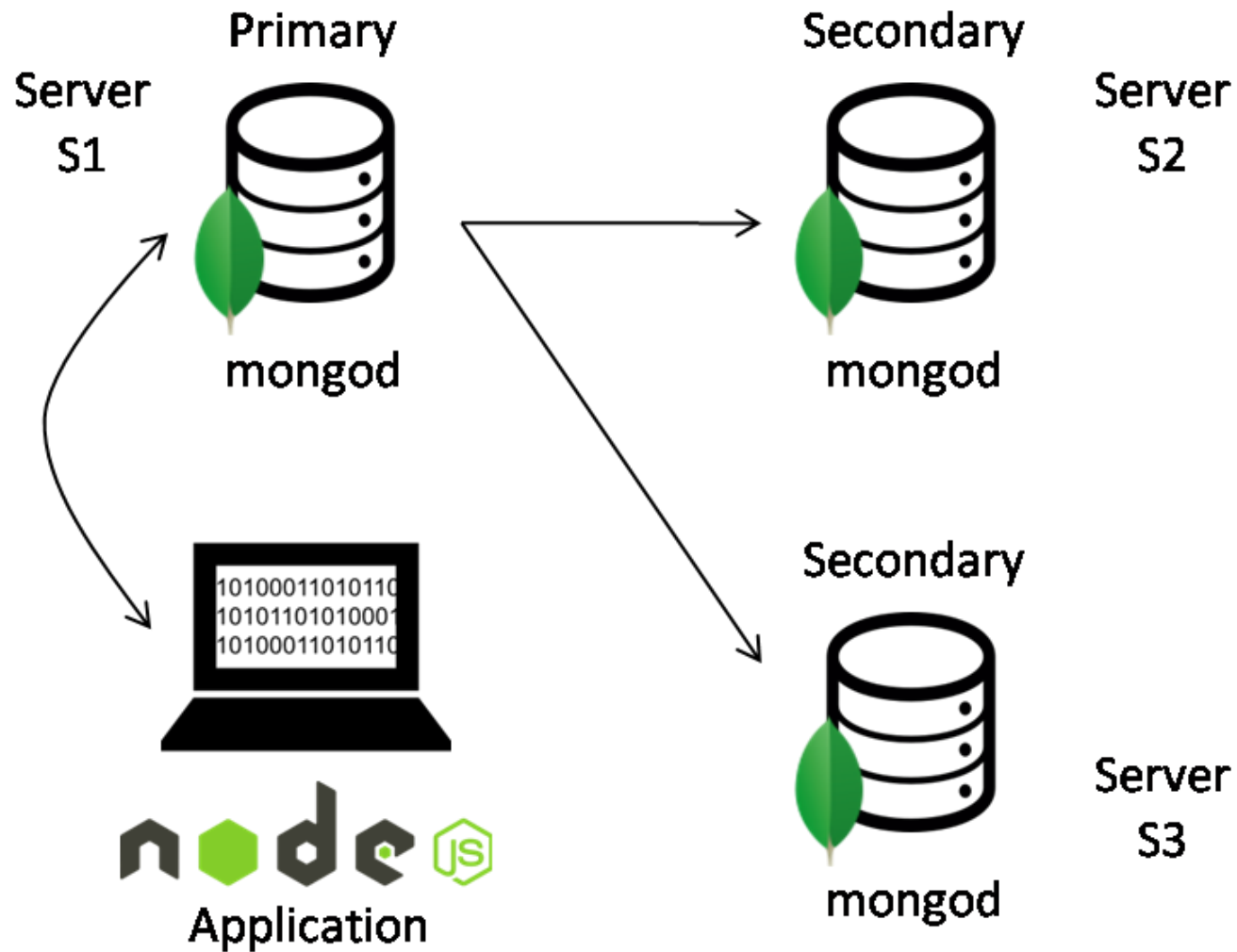


# Concepto General



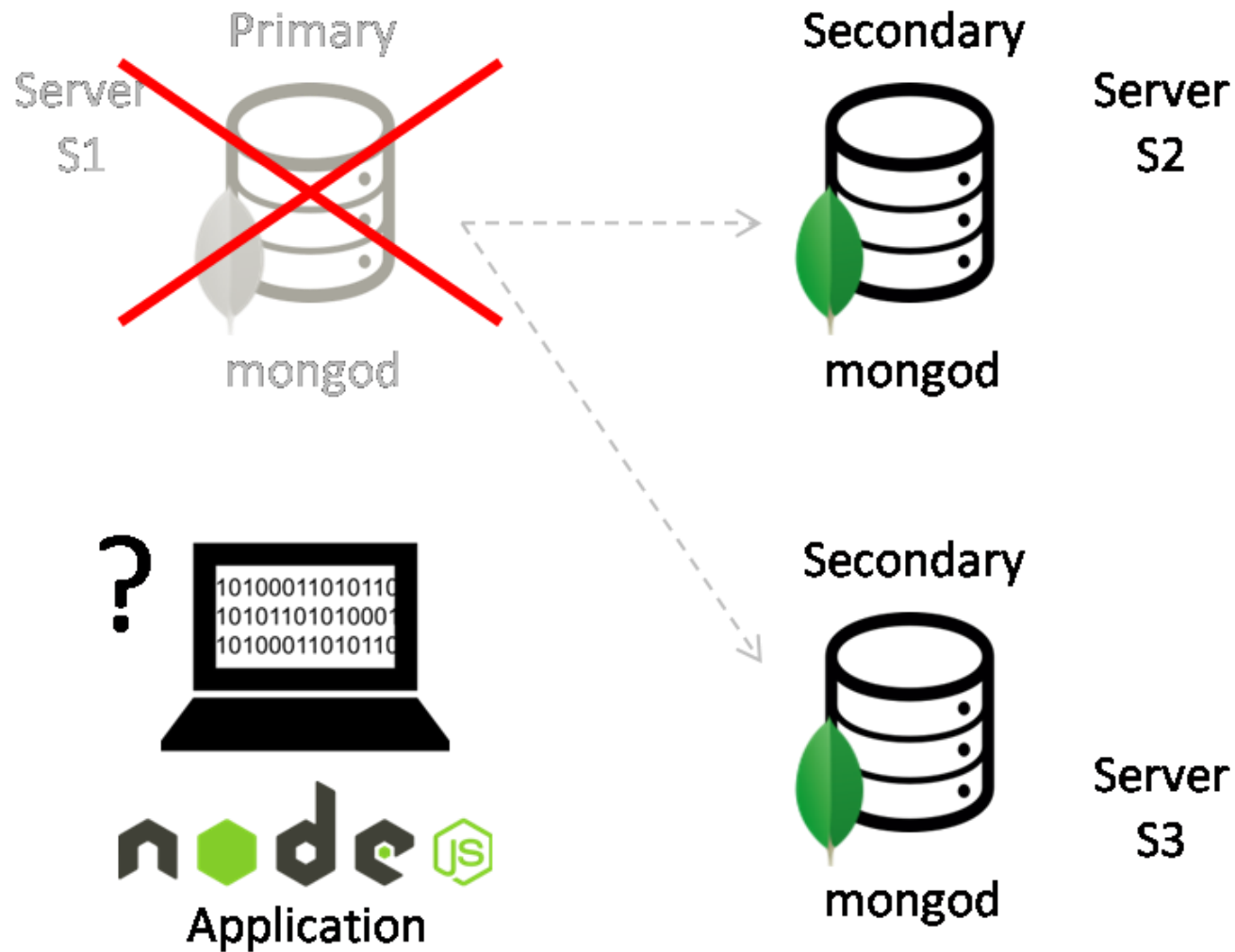
Replicación de Datos

# Funcionamiento



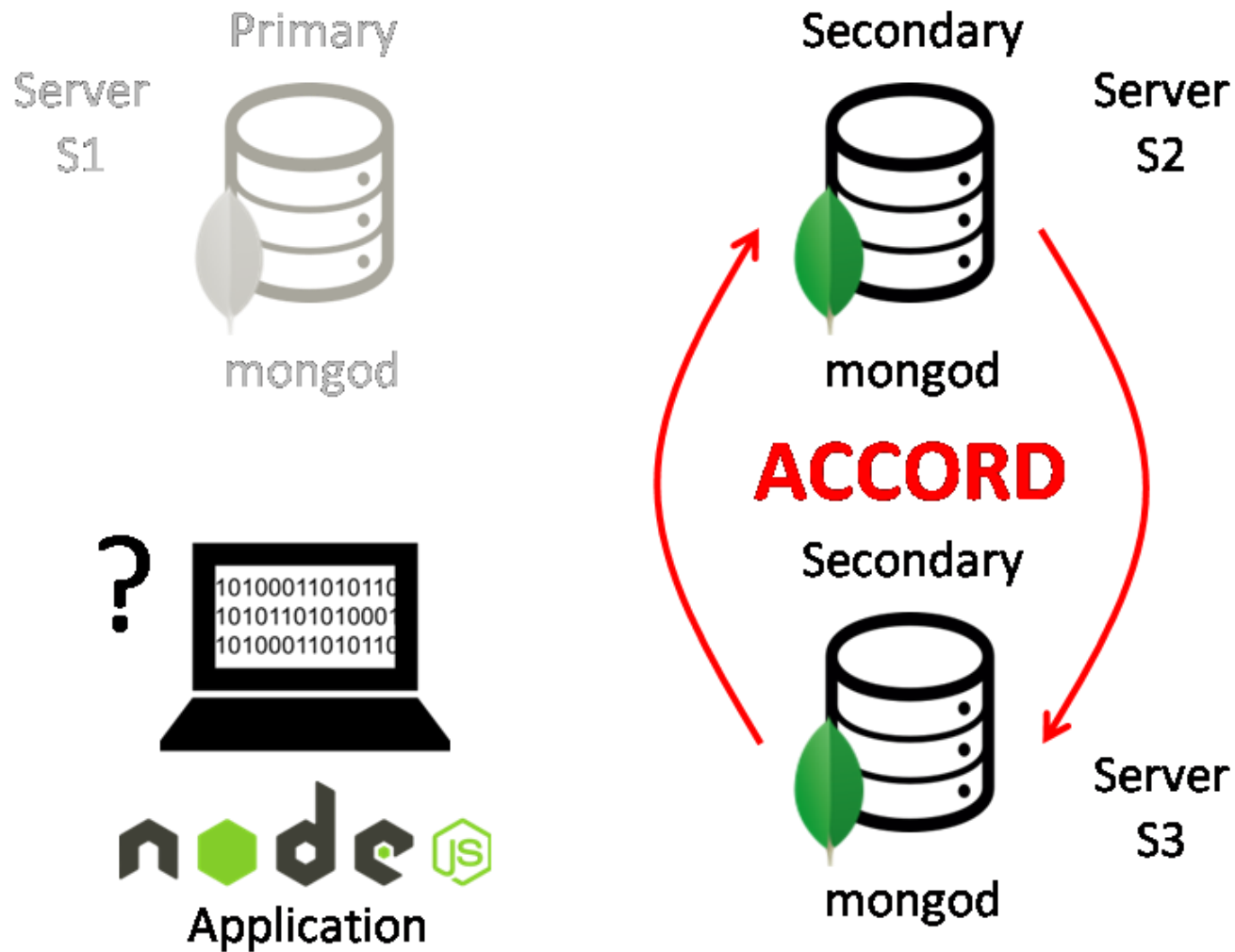
Replicación de Datos

# Funcionamiento



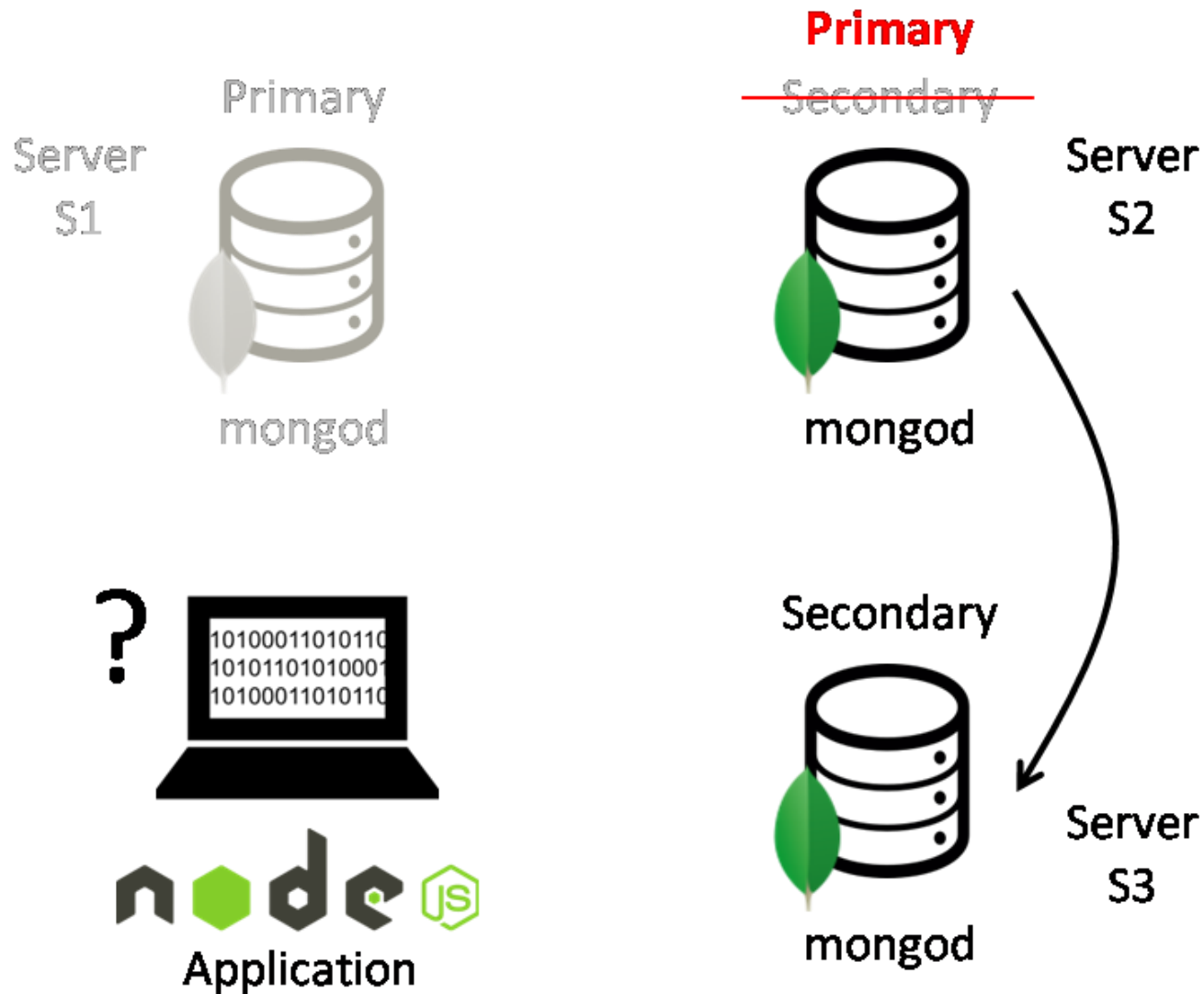
Replicación de Datos

# Funcionamiento



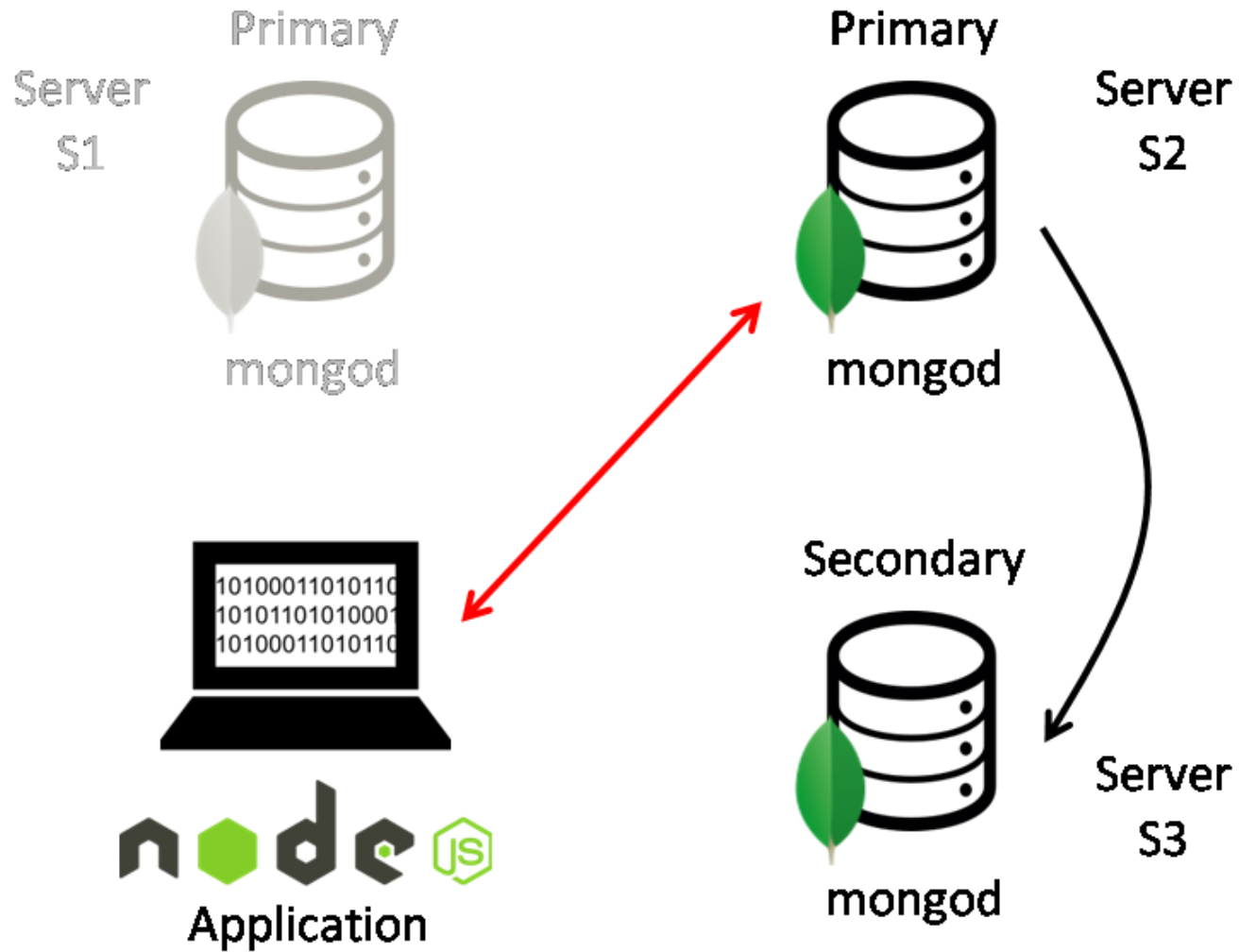
Replicación de Datos

# Funcionamiento



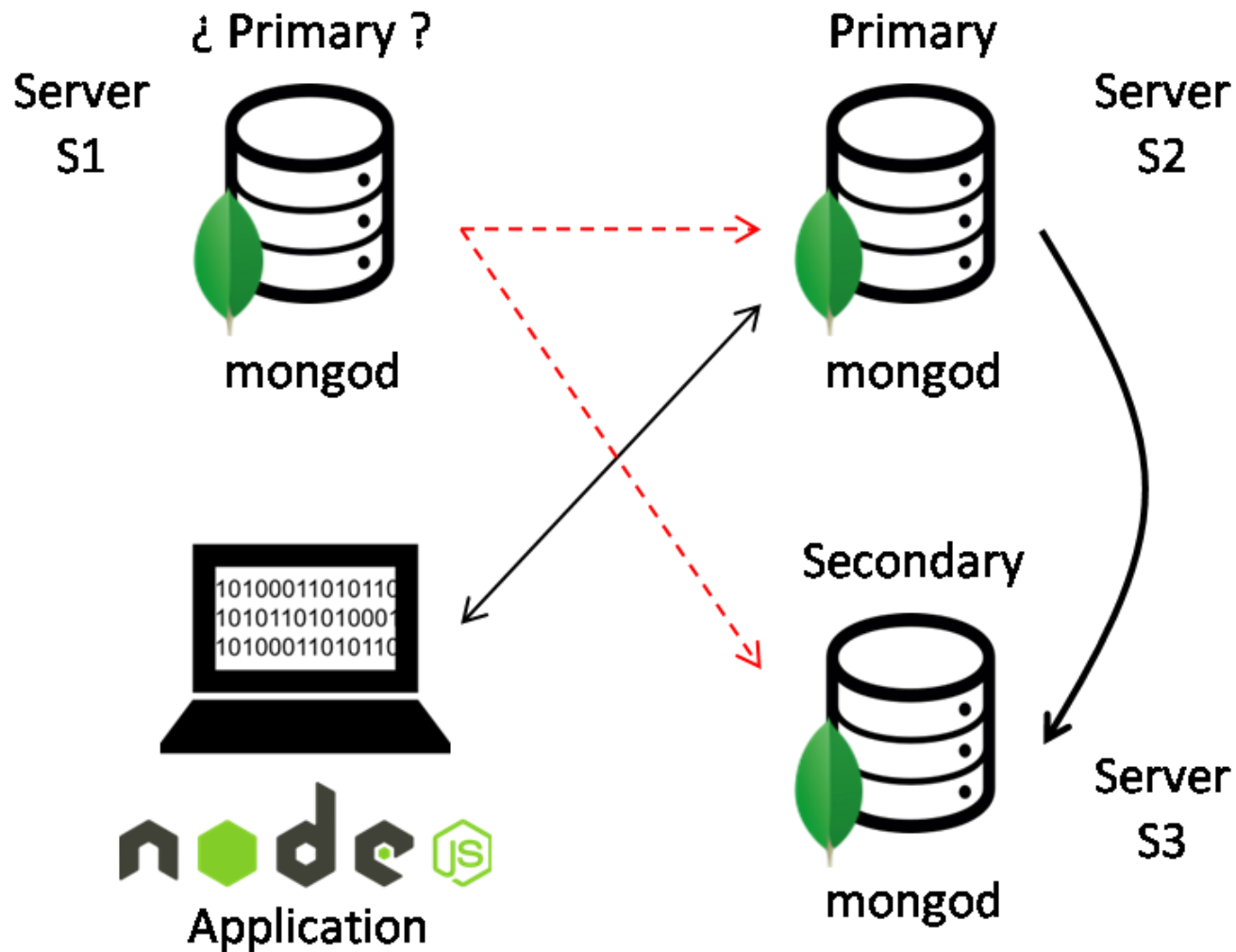
Replicación de Datos

# Funcionamiento



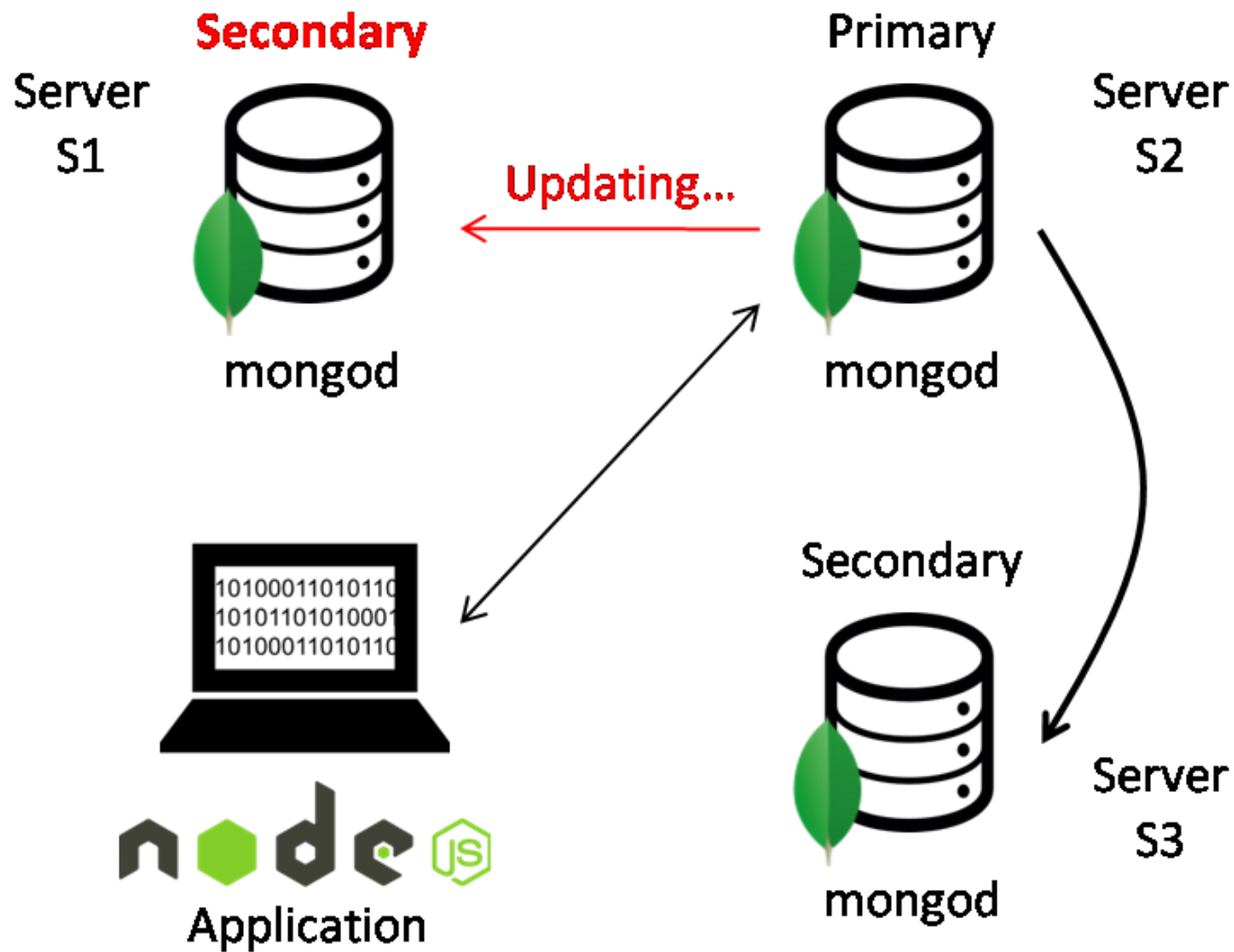
Replicación de Datos

# Funcionamiento



Replicación de Datos

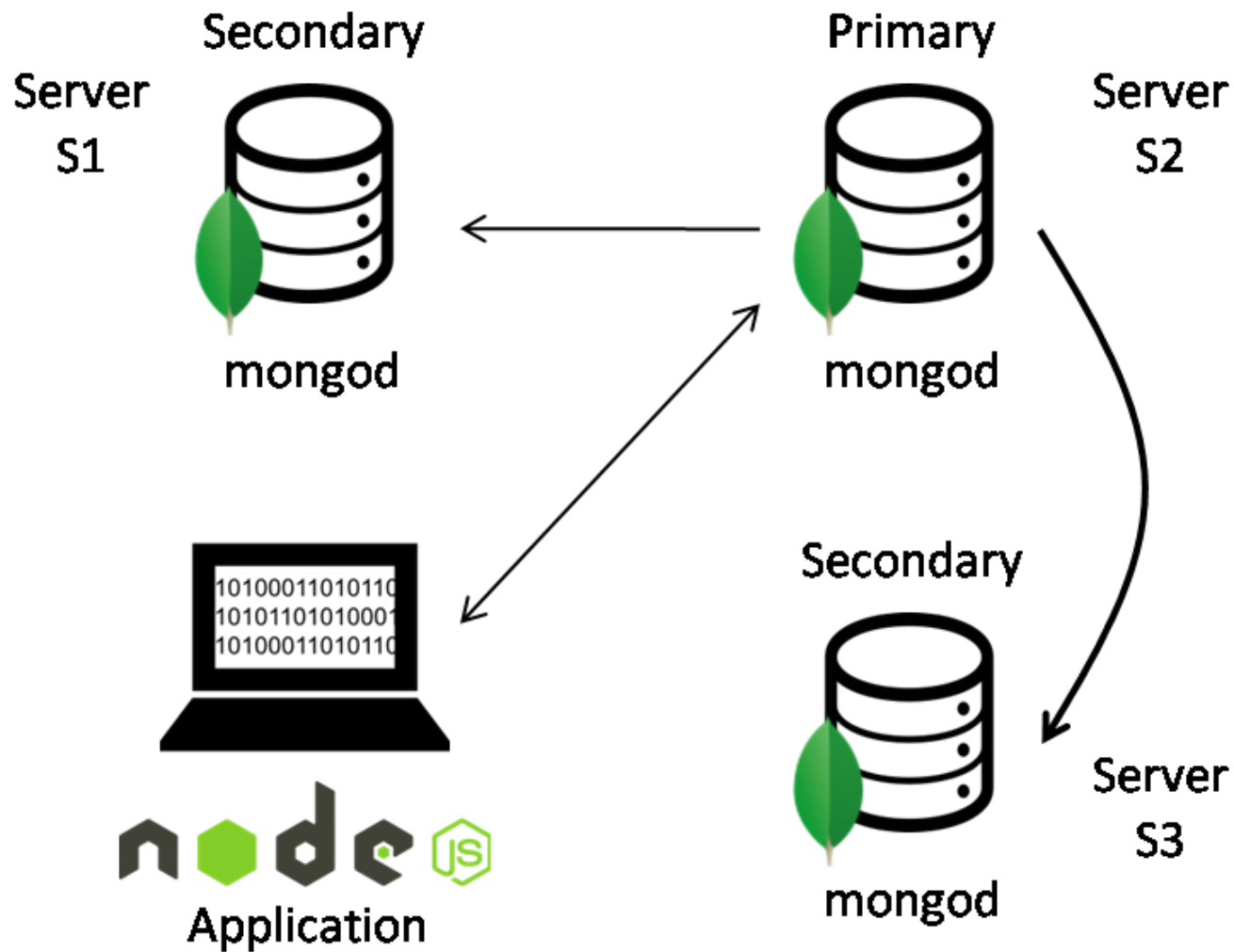
# Funcionamiento



Replicación de Datos

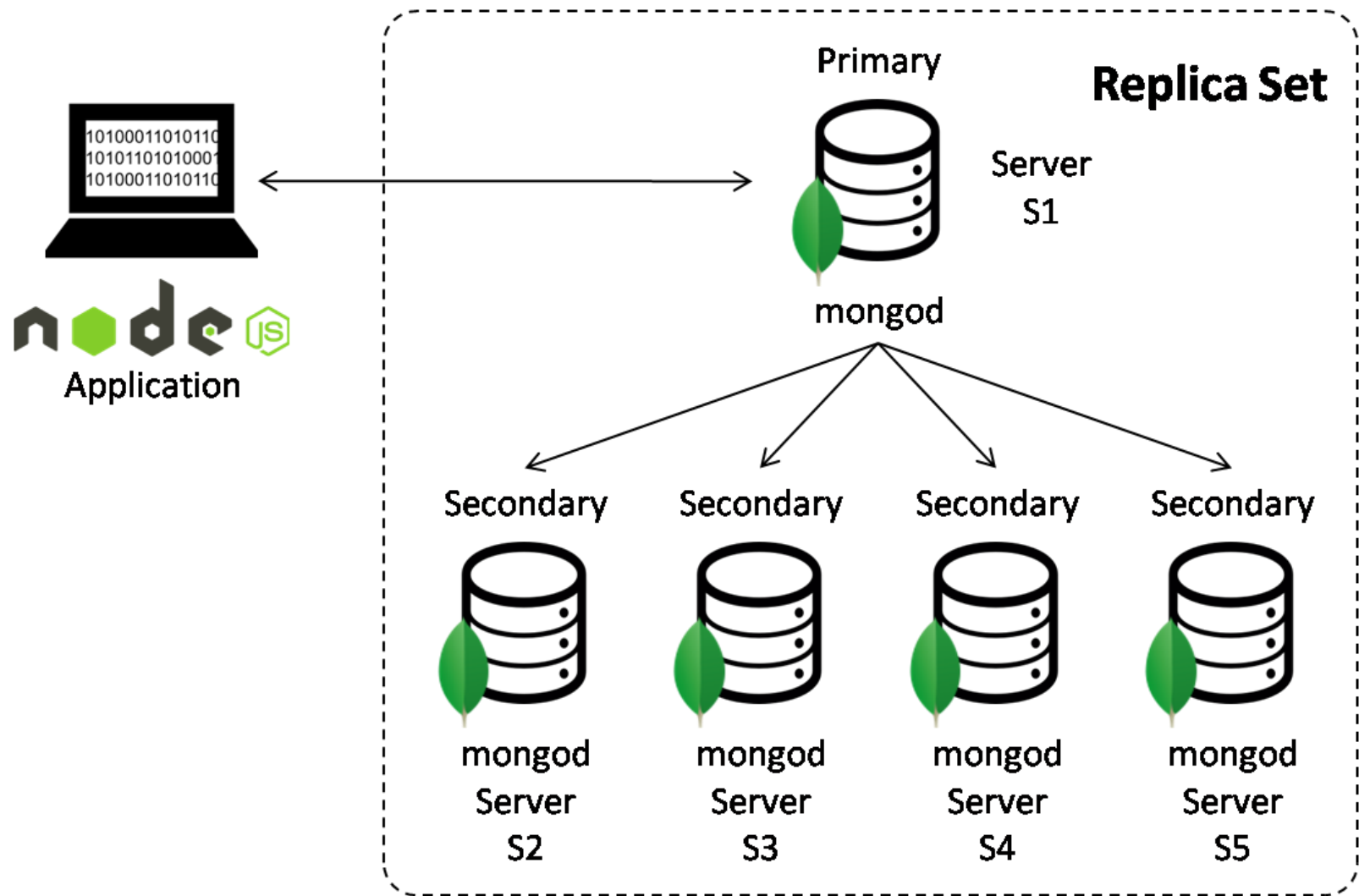


# Funcionamiento



Replicación de Datos

# Funcionamiento



Replicación de Datos

# Tipos de Nodos



**mongod**

## Primary

Será el único que reciba las operaciones de escritura aunque también puede recibir operaciones de lectura. Sólo puede haber uno por cada Replica Set.

## Secondary

Actualizan su contenido desde el nodo Primary y reciben únicamente operaciones de lectura.

## Arbitrer

No almacena datos ni recibe operaciones CRUD, pero es fundamental cuando se debe seleccionar un nuevo nodo Primary.

## Delayed

Es un tipo de nodo Secondary cuya actualización se realiza con una determinada demora en el tiempo.

## Hidden

Es un tipo de nodo Secondary pero que no recibe ningún tipo de operación de lectura.

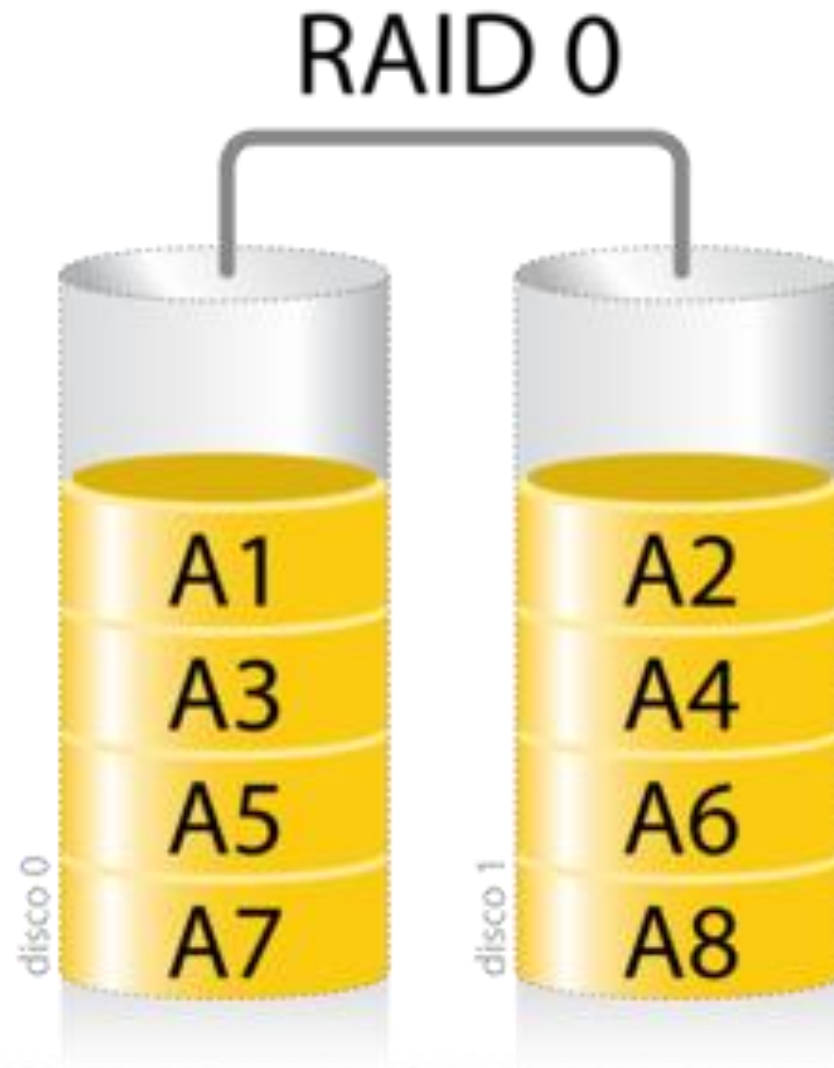


# Replicación de Datos

Bases de Datos Relacionales  
Bases de Datos No Relacionales  
Comparativa entre Modelos de Datos  
Introducción a MongoDB  
Principales elementos de MongoDB  
Iniciando MongoDB  
Operaciones CRUD en MongoDB  
Replicación de Datos  
**Sharding de datos**

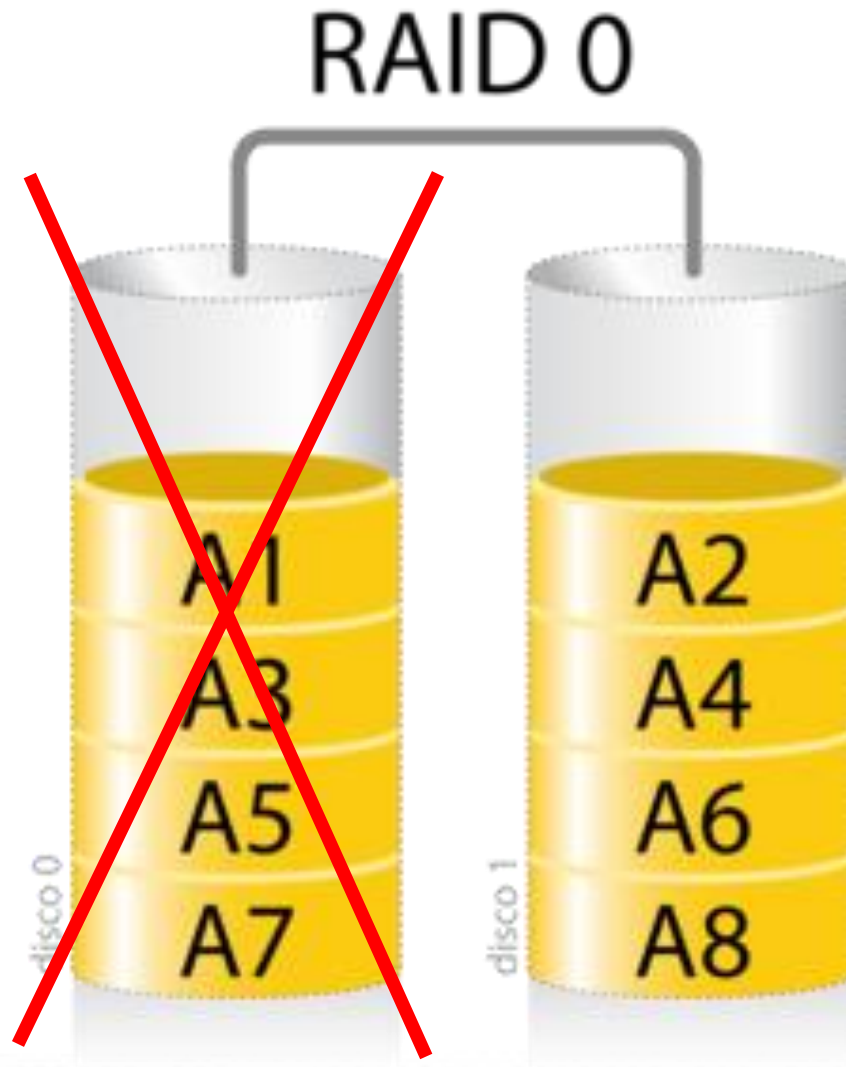


# Concepto General



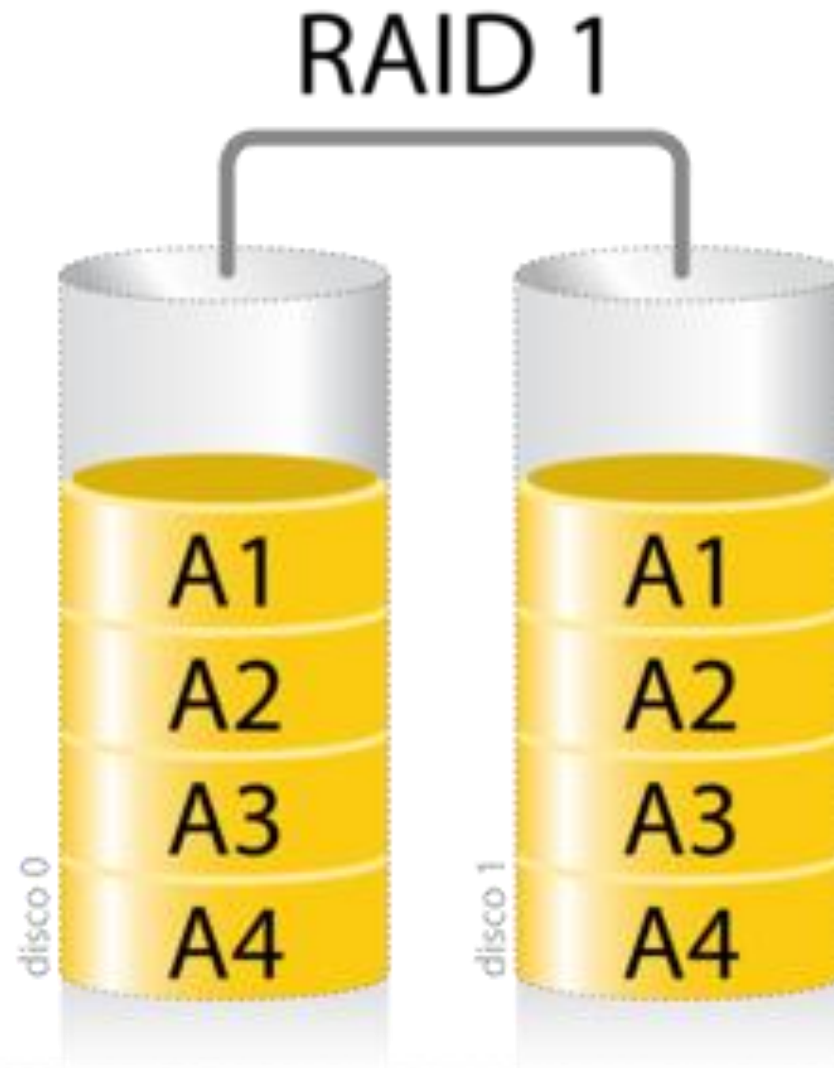
Sharding de Datos

# Concepto General



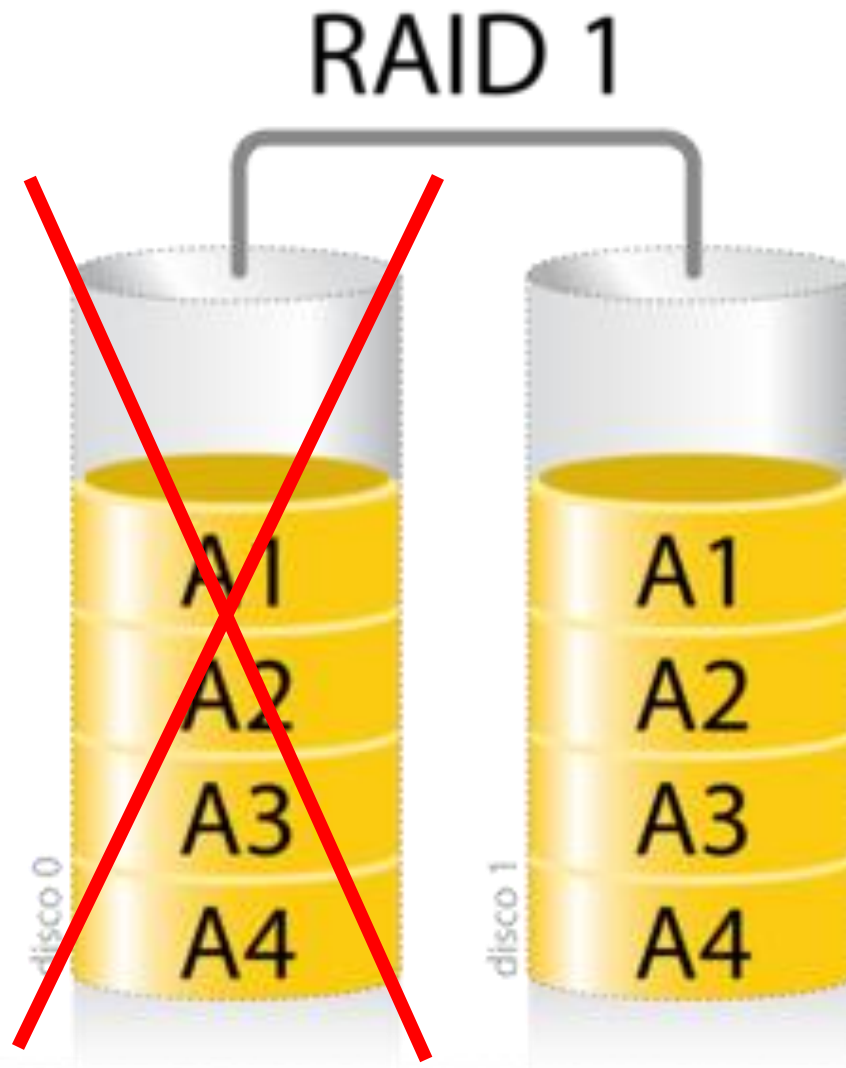
Sharding de Datos

# Concepto General



Sharding de Datos

# Concepto General

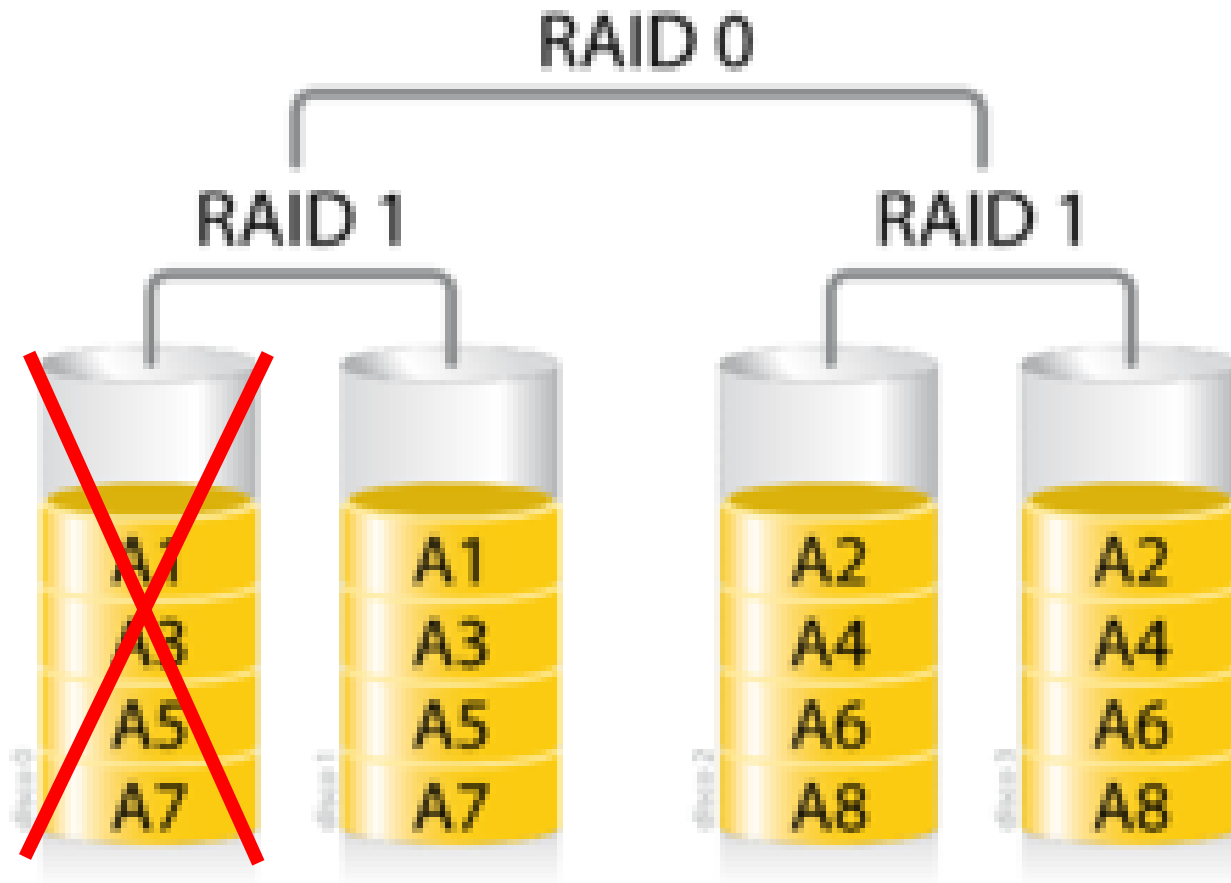


Sharding de Datos



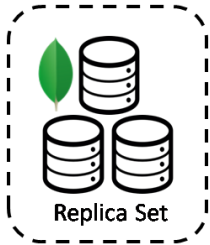
## Concepto General

# RAID 10



Sharding de Datos

# Componentes de un Sharded Cluster



Shard

## Shard

Replica Set en cuyo interior se almacena parte de la información gestionada por el sistema.



mongos

## Shard Manager

Proceso **mongos** que se encarga de gestionar las conexiones que llegan al cluster y redirigirlas al shard correspondiente.



mongod


## Config Server

Proceso **mongod** encargado de almacenar el estado del cluster así como de controlar dónde se encuentra almacenado cada pedazo de información.



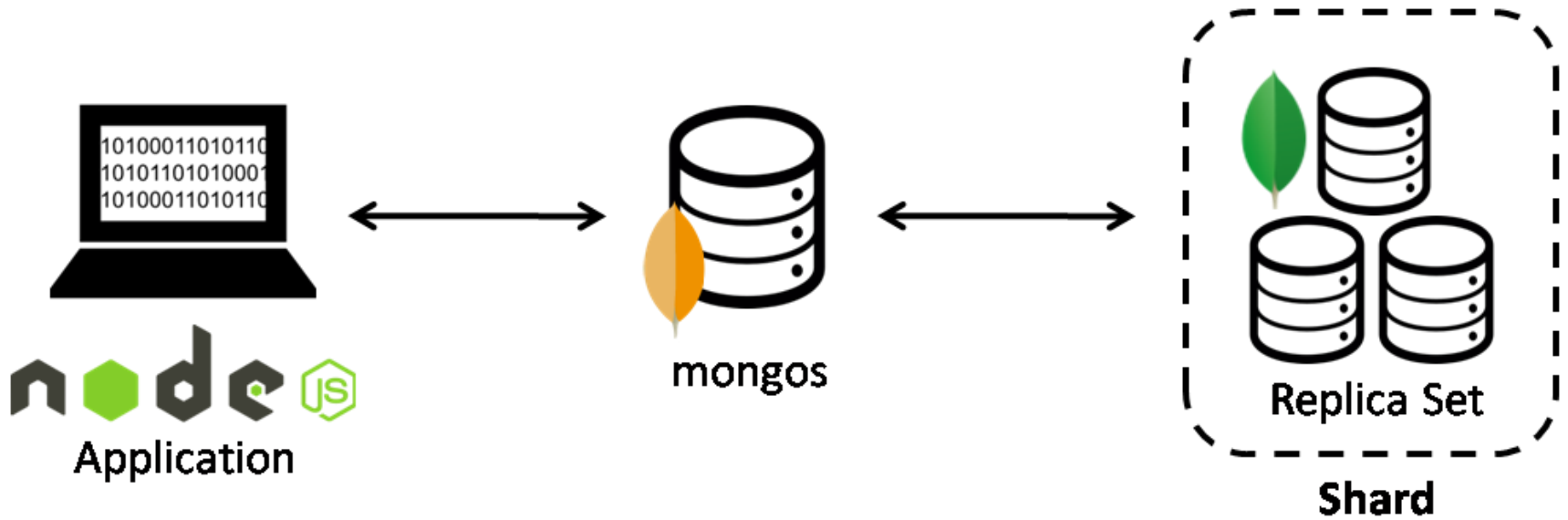
## Shard Key

Índice empleado en el cluster para subdividir los datos.

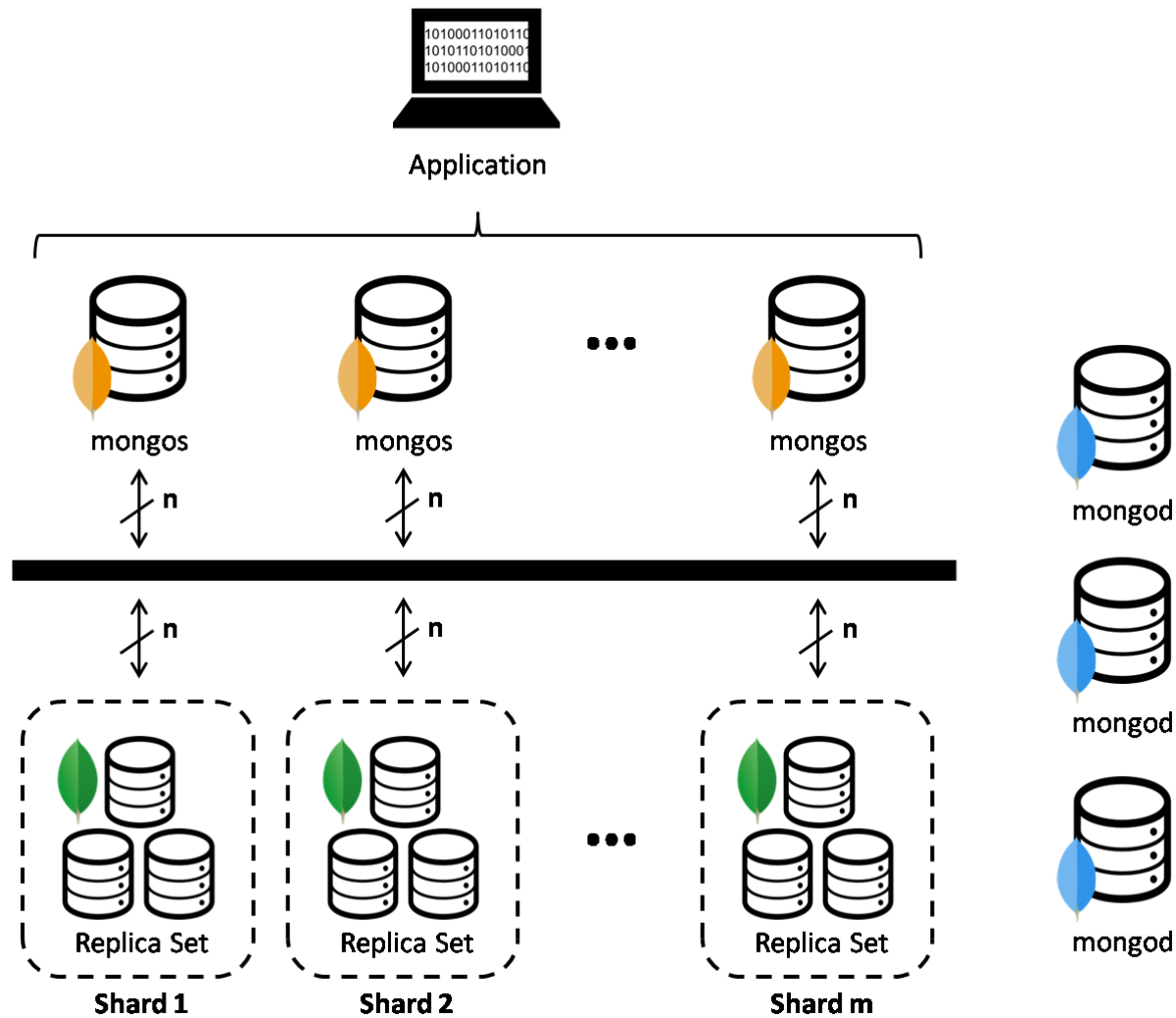


## Sharding de Datos

# ¿Cómo funciona?

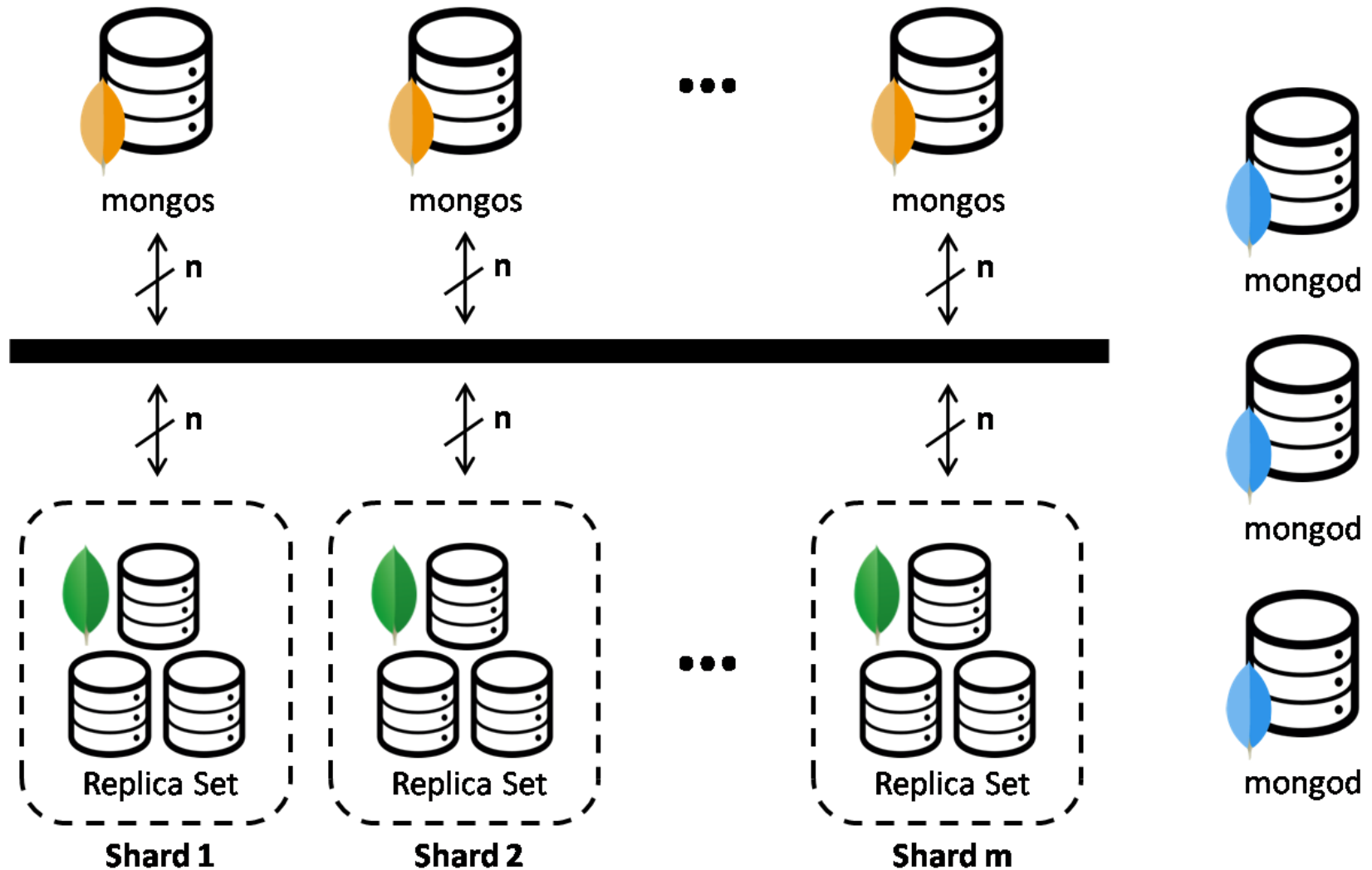


# ¿Cómo funciona?



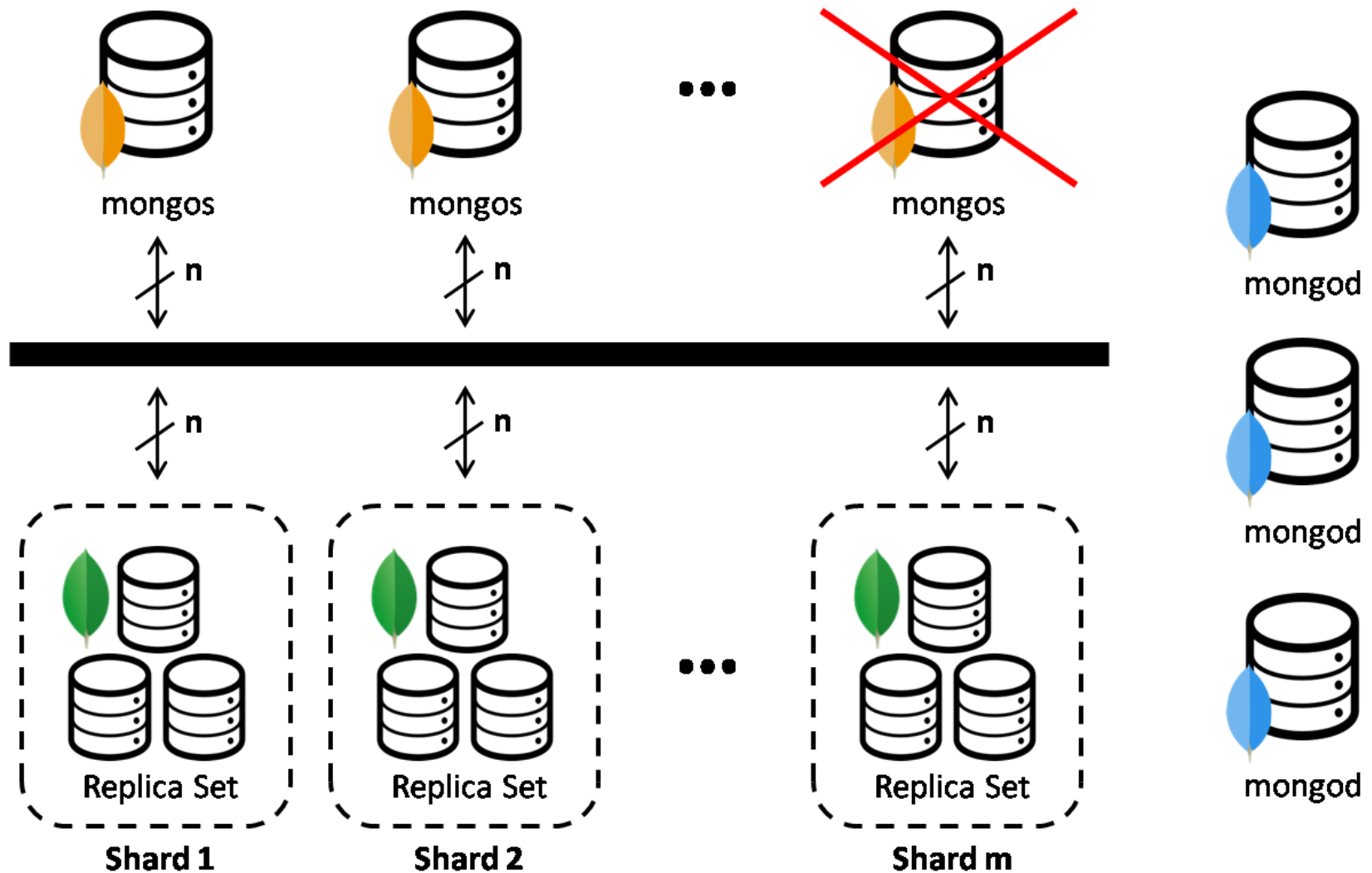
Sharding de Datos

# Fallo en un Shard Manager



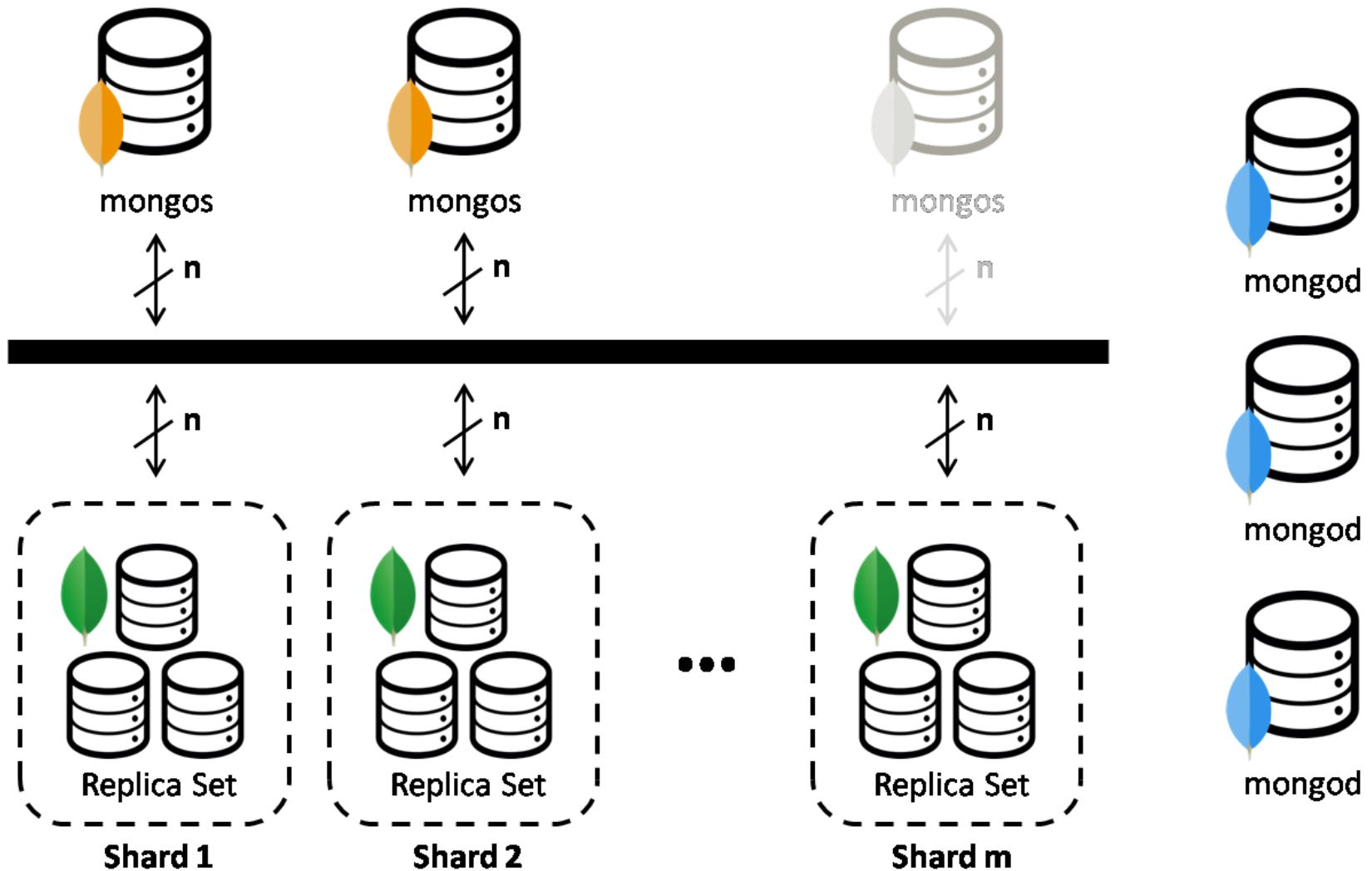
Sharding de Datos

# Fallo en un Shard Manager



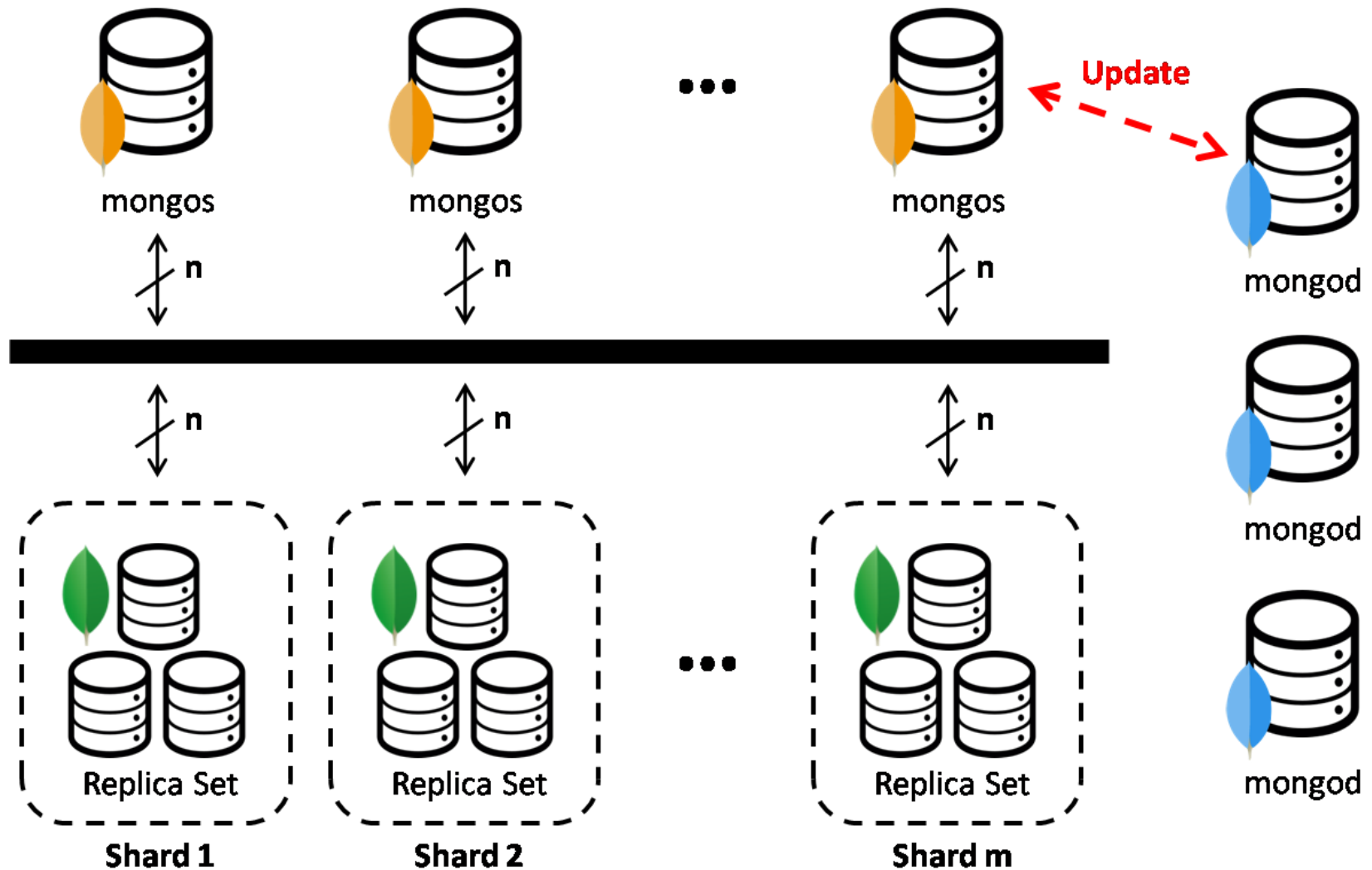
Sharding de Datos

# Fallo en un Shard Manager



Sharding de Datos

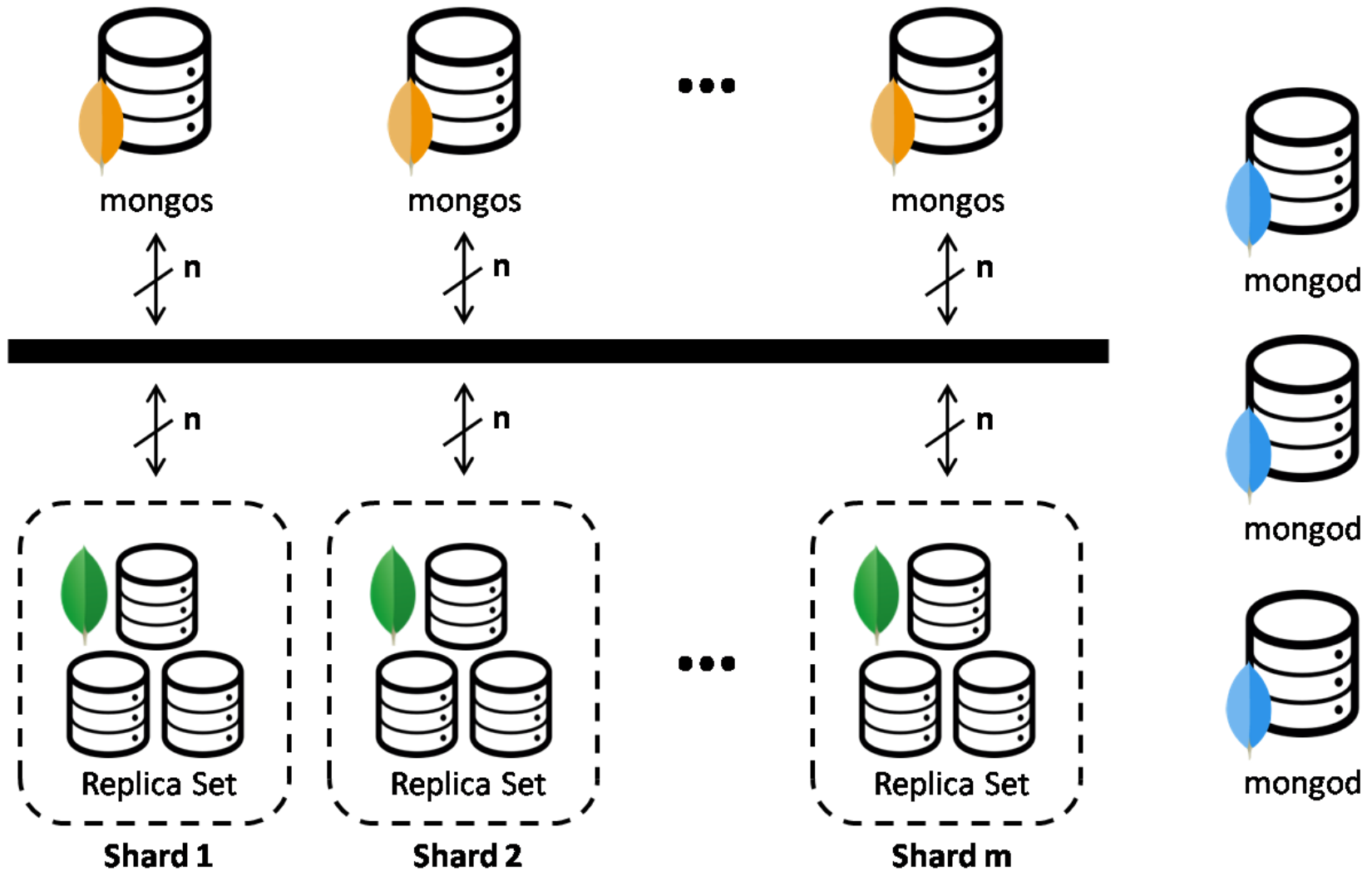
# Fallo en un Shard Manager



Sharding de Datos

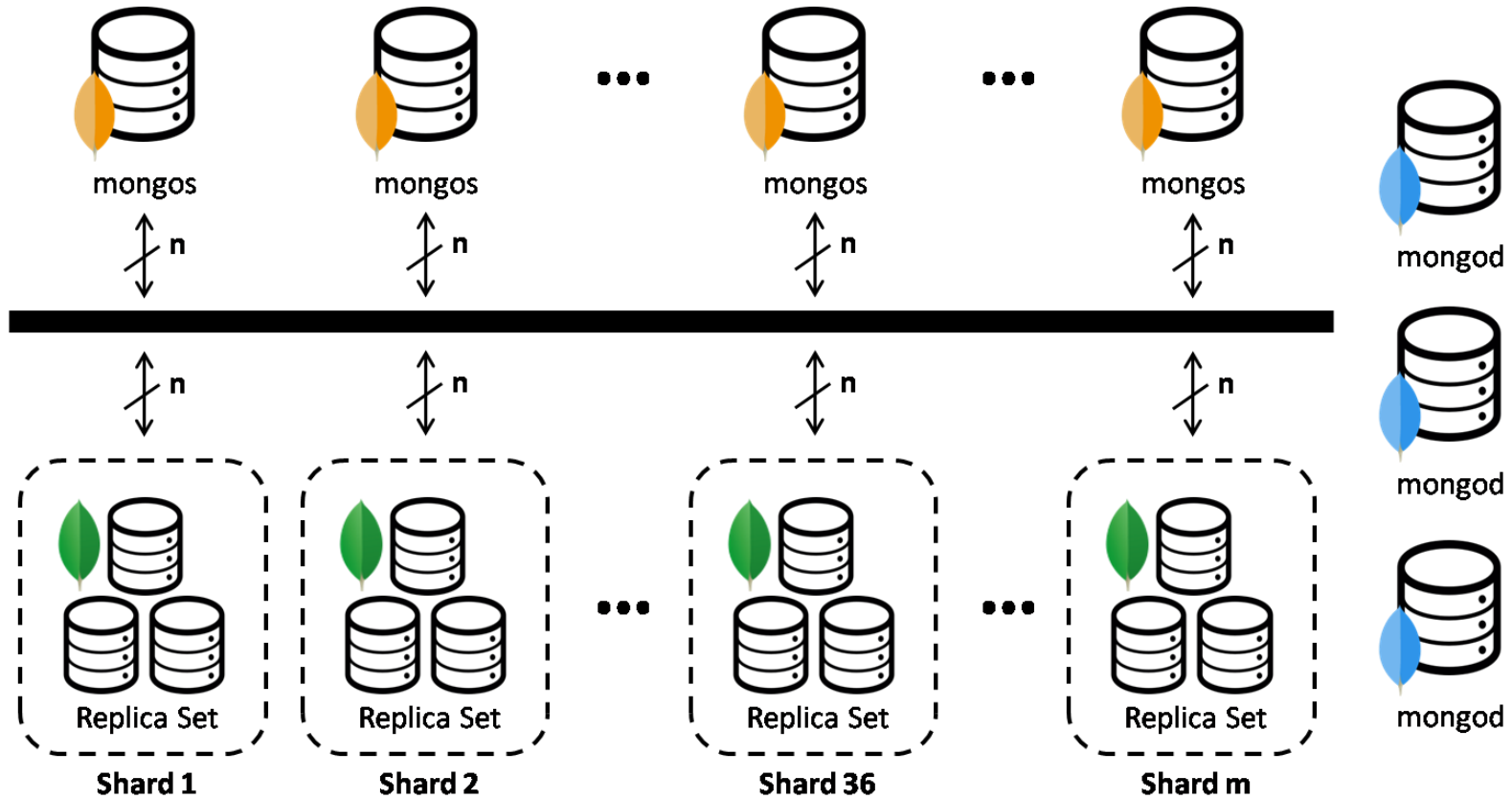


# Modificación en el Sharded Cluster



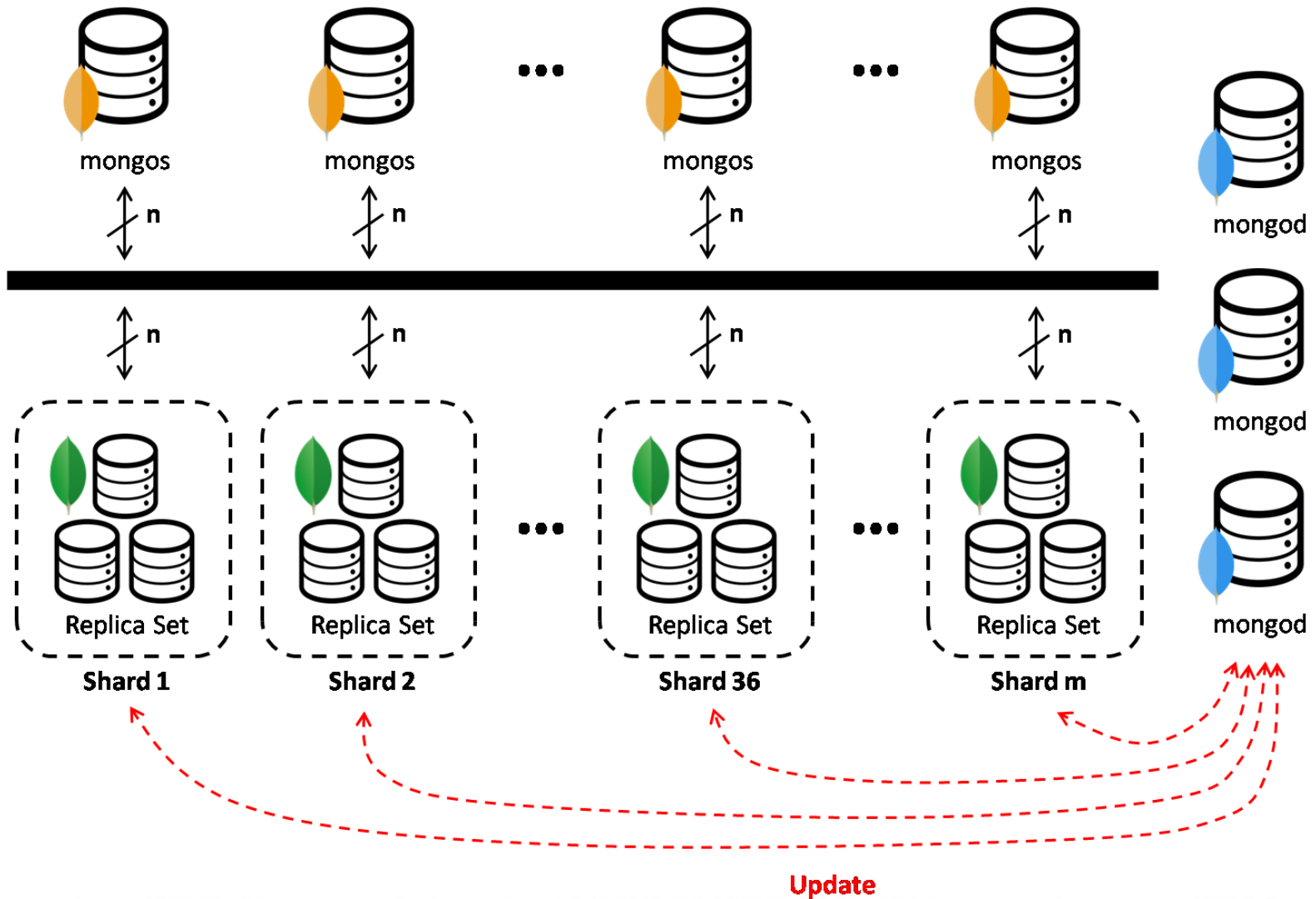
Sharding de Datos

# Modificación en el Sharded Cluster



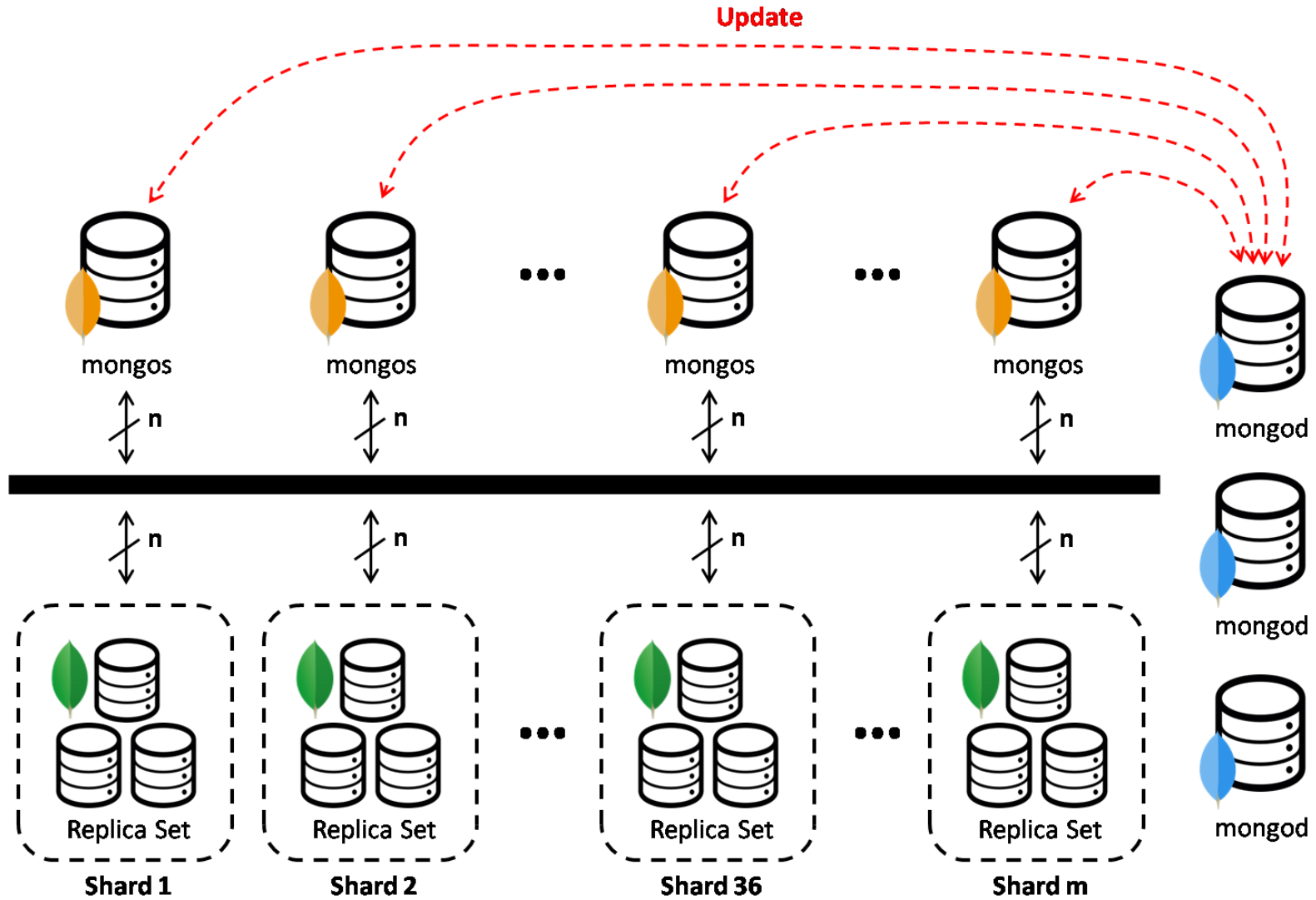
Sharding de Datos

# Modificación en el Sharded Cluster



Sharding de Datos

# Modificación en el Sharded Cluster





Sharding de Datos



# mongoDB

## 101

dailos rafael díaz lara  
julio 2015

 @ddialar  [linkedin.com/in/ddialar](https://www.linkedin.com/in/ddialar)



## Página web oficial

<http://docs.mongodb.org/manual/>

## MongoDB University

<https://university.mongodb.com/>

M102 – MongoDB for DBAs

M202 – MongoDB Advanced deployments and operations

M101JS – MongoDB for Node.js Developers

M101J – MongoDB for Java Developers

M101P – MongoDB for Developers

M101N – MongoDB for .Net Developers

## Certificaciones oficiales

<https://university.mongodb.com/exams>