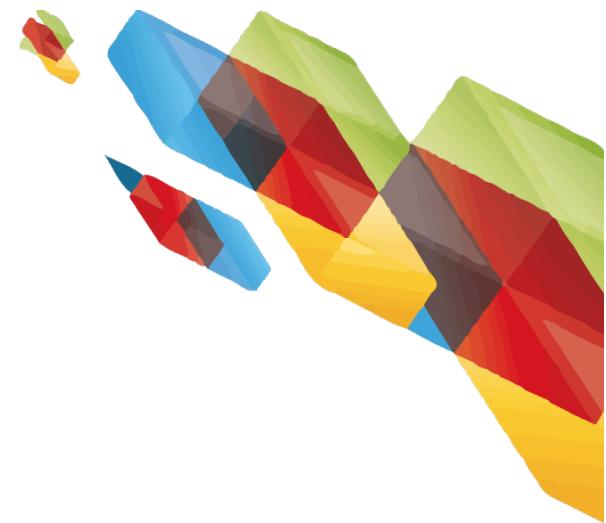


Ubaldo Taladriz
utaladriz@exe.cl
@utaladriz

Historia y características

- mongoDB = “Humongous DB”
- Open source
- Basada en documentos
- Rendimiento y Disponibilidad
- Escalamiento automático
- C-P en CAP
- Replicación y Sharding
- Map Reduce
- Updates rápidos
- Soporte completo para indexación

-blog.mongodb.org/post/475279604/on-distributed-consistency-part-1
-mongodb.org/manual



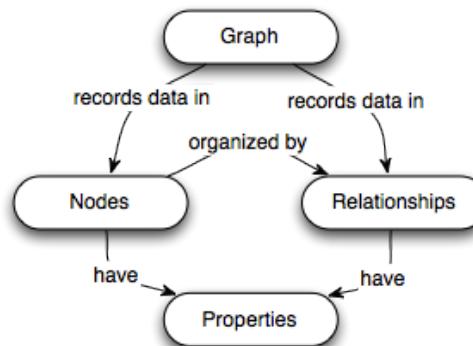
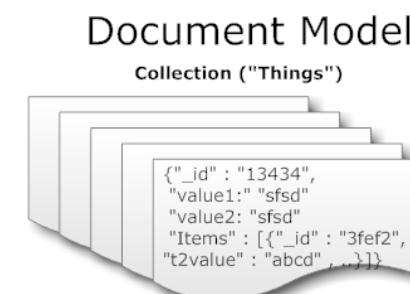
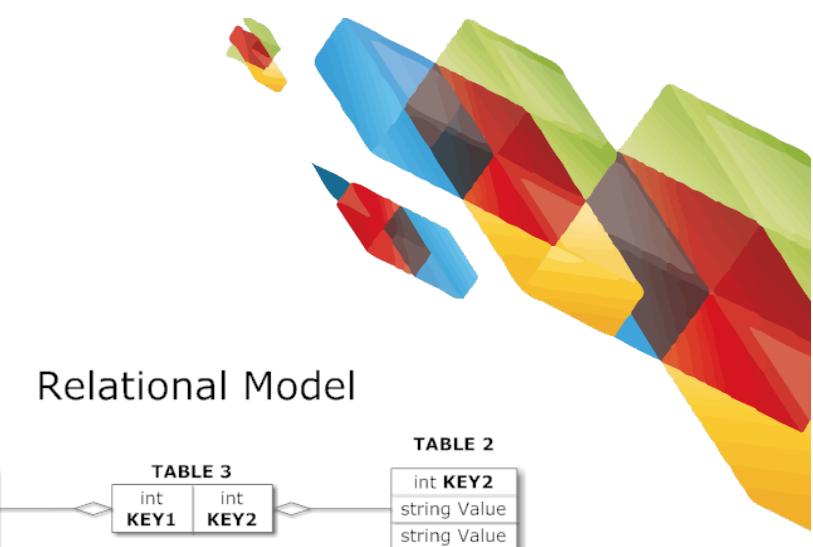
Tipos de Base de datos

Key/value (Dynamo)

Columnar/tabular (HBase)

Documentales (mongoDB)

De Grafos (neo4j, Titan)



<http://www.aaronstannard.com/post/2011/06/30/MongoDB-vs-SQL-Server.aspx>

Motivaciones

- Problemas con SQL
 - Esquema (¿?)
 - No escalan fácilmente (Diseñadas con tecnologías de los 90's)
 - Requiere de joins, que a veces no son intuitivos
- Algunos extras mongoDB
 - Soporte para distintos lenguajes (Java, Javascript, PHP, etc.)
 - "Run anywhere" (SO, VM's, cloud, etc.)
 - Mantiene los aspectos esenciales de una RDBMS pero con el aprendizaje de los sistemas noSQL de tipo key-value

http://www.slideshare.net/spf13/mongodb-9794741?v=qf1&b=&from_search=13

¿Alguien usa MongoDB ?



In Good Company



-Steve Francia, http://www.slideshare.net/spf13/mongodb-9794741?y=qf1&b=&from_search=13



Modelo de datos



- Basado en documentos (máximo 16 MB)
- Para almacenar documentos más grandes MongoDB provee la GridFS API
- Los Documentos son en formato BSON, que son pares campo-valor
- Cada documento es almacenado en una colección
- Colecciones
 - Tienen índices en común
 - Son como una tabla ...
 - Pero no hay nada que garantice que los documentos tienen la misma estructura
- Los documentos pueden ser anidados (Máx 100)

[-docs.mongodb.org/manual/](http://docs.mongodb.org/manual/)

JSON

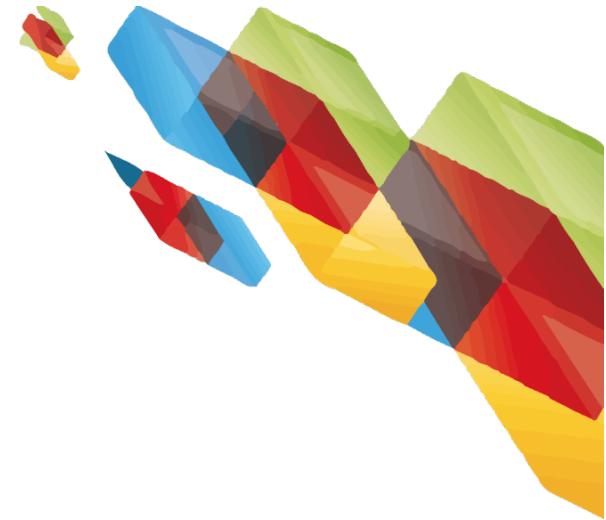
- ▶ “JavaScript Object Notation”
- ▶ Fácil de escribir y leer por humanos y fácil para computacionalmente procesarlos/generarlos
- ▶ Pueden contener objetos anidados
- ▶ Basado en
 - ▶ Pares nombre/valor
 - ▶ Listas ordenadas de valores

<http://json.org/>

BSON

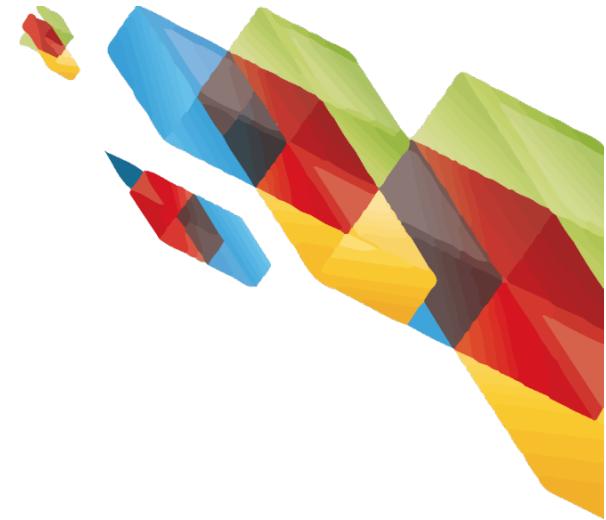
- “Binary JSON”
- Codificación binaria de documentos JSON
- También permiten “referencias”
- Su estructura permite tener objetos como valores reduciendo la necesidad de joins
- Metas
 - Liviano
 - Navegable
 - Eficiente(Decodificación y codificación)

<http://bsonspec.org/>



Ejemplo BSON

```
{  
  "_id": "37010"  
  "city": "ADAMS",  
  "pop": 2660,  
  "state": "TN",  
  "councilman": {  
    "name": "John Smith"  
    "address": "13 Scenic Way"  
  }  
}
```

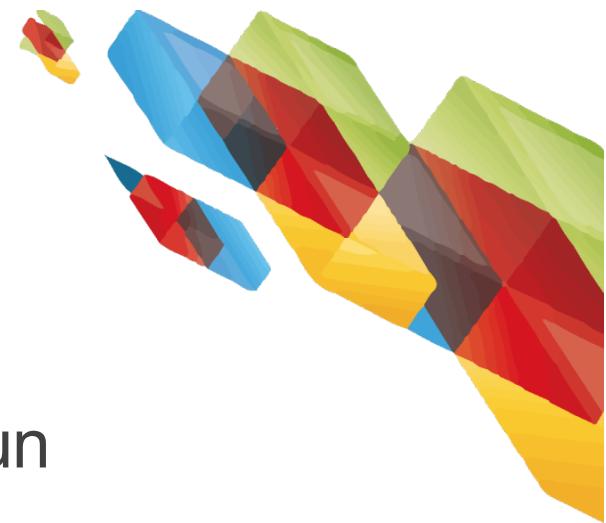


Tipos BSON

Tipo	Número
Double	1
String	2
Object	3
Array	4
Binary data	5
Object id	7
Boolean	8
Date	9
Null	10
Regular Expression	11
JavaScript	13
Symbol	14
JavaScript (with scope)	15
32-bit integer	16
Timestamp	17
64-bit integer	18
Min key	255
Max key	127

El numero puede ser usado con el operador \$type para consultar el tipo

El campo _id



- Por omisión cada documento contiene un campo `_id`. :
 - Actúa como llave primaria dentro de la colección
 - Es un valor único, inmutable y puede ser de cualquier tipo exceptuando arreglos.
 - El tipo de datos por omisión es **ObjectId**, el cual es “pequeño, único, rápido para generar y ordenar”
 - Ordenar por el valor de ObjectId es casi equivalente a ordenar en el tiempo de creación, eso implica casi costo cero a futuro.

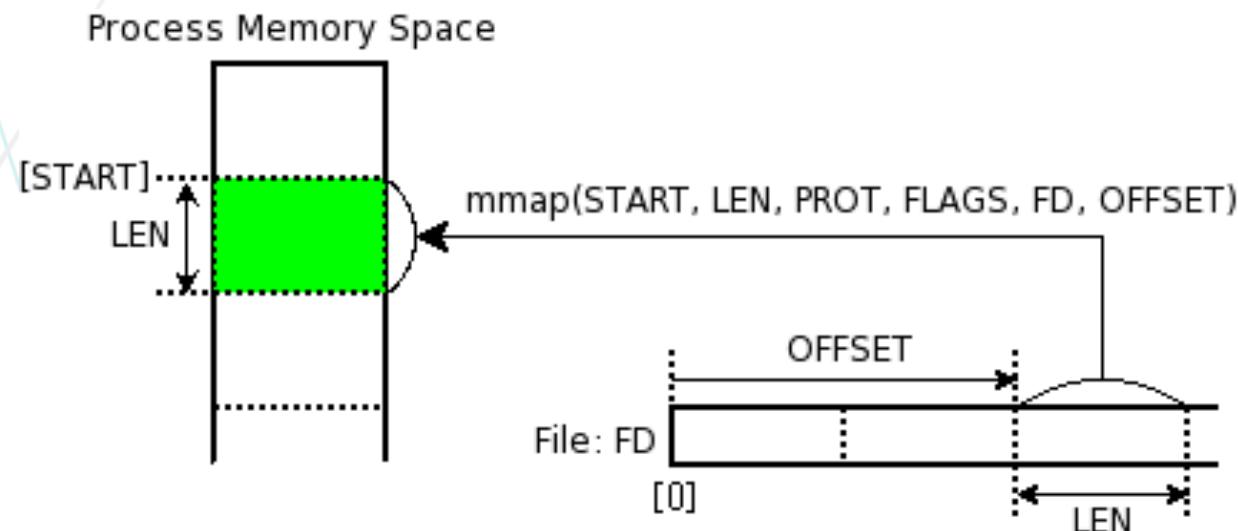
<http://docs.mongodb.org/manual/reference/bson-types/>

mongoDB vs. SQL

mongoDB	SQL
Documento / Document	Tupla
Colección / Collection	Tabla/Vista
PK: campo _id	PK: Cualquiera de los campos
No es uniforme	Uniforme basado en esquemas
Índices / Index	Índices /
Estructuras empotradas / Embedded Structure	Joins
Sharding (Es como una partición)	Particiones

Archivos mapeados en memoria

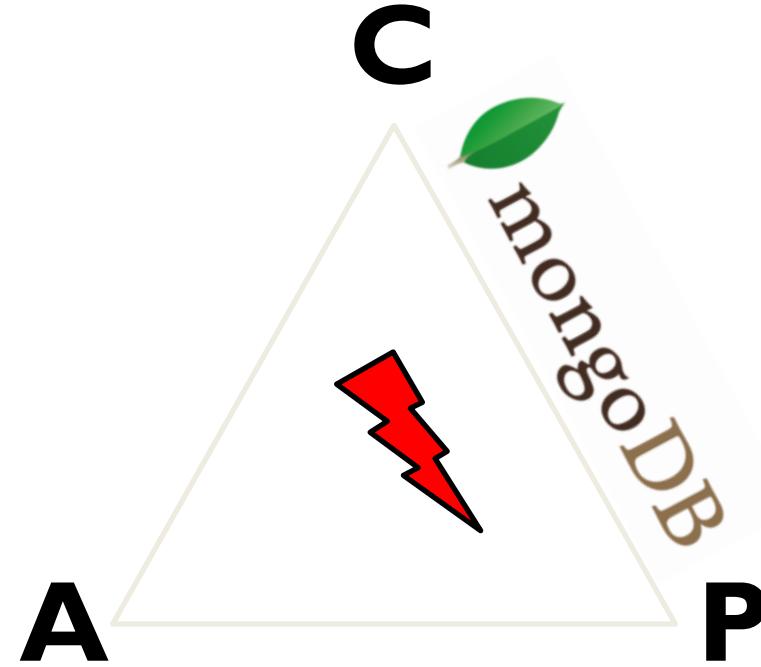
- Un archivo mapeado en memoria, es un segmento virtual, en el cual cada byte tiene una correlación uno a uno con los bytes de una porción del archivo”
- mmap()



13

Teoría de noSQL: CAP

- Muchos nodos (Sistemas distribuídos)
- *Los nodos contienen replicas de los datos particionados*
- Consistency / Consistencia
 - Todas las replicas tienen la misma versión de los datos
- Availability / Disponibilidad
 - El sistema permanece en operación aún cuando fallen nodos
- Partition tolerance / Particiones
 - Múltiples puntos de entradas
 - El sistema opera dividido



Teorema de CAP :
Satisfacer los tres al mismo tiempo en un sistema distribuido es imposible

14

ACID - BASE



- Atomicity / Atómica
- Consistency / Consistencia
- Isolation / Aislación
- Durability / Durables



- Disponibilidad básica (CP)
- Soft-state
- Eventualmente consistente (AP)

Pritchett, D.: BASE: An Acid Alternative (queue.acm.org/detail.cfm?id=1394128)

Instalación

Para instalar mongoDB, en el siguiente link se debe seleccionar la arquitectura y el sistema operativo apropiado: <http://www.mongodb.org/downloads>

Primero extraer los archivos (preferentemente en el drive C).

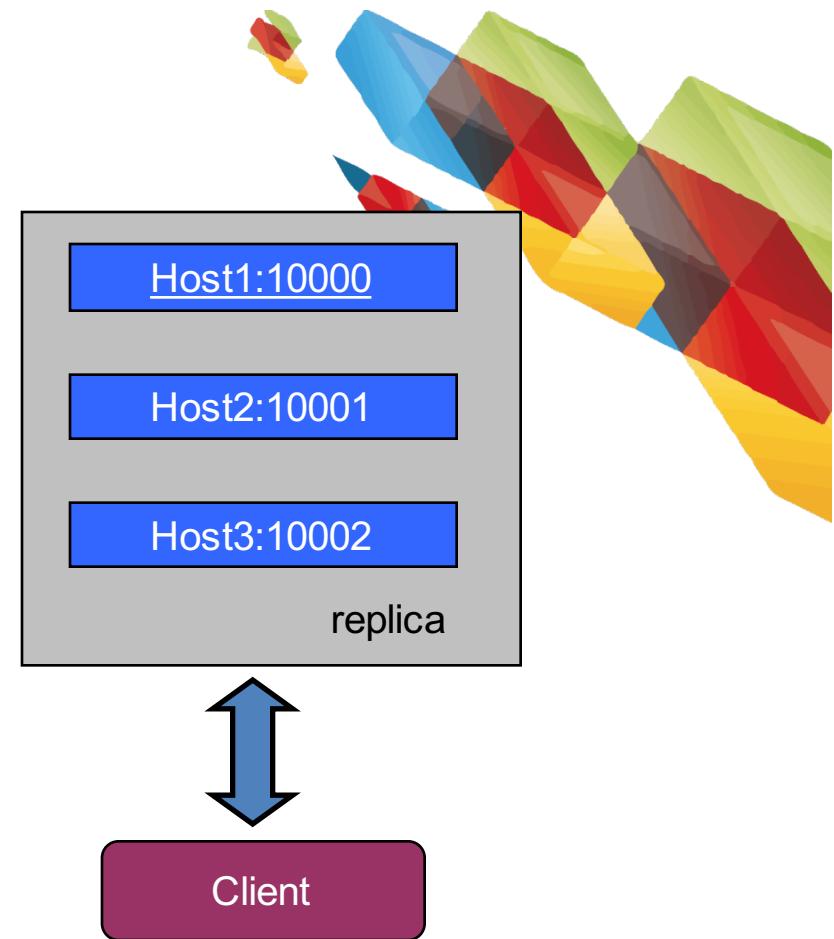
Finalmente se debe crear un directorio para los datos en algún lugar del drive para el uso de mongoDB

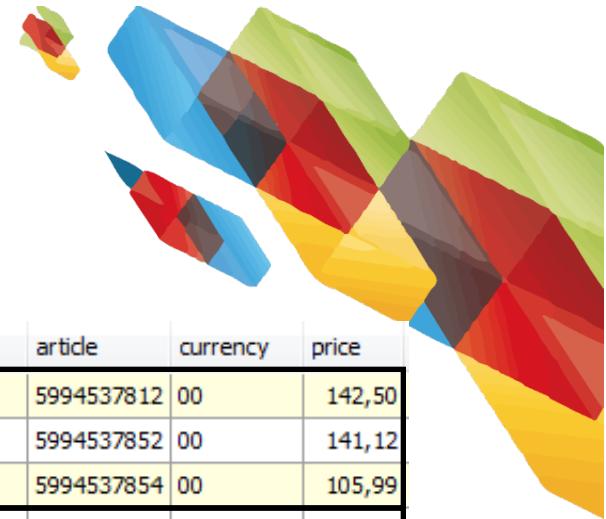
Ejemplo: “md data” seguido por “md data\db”

<http://docs.mongodb.org/manual/tutorial/install-mongodb-on-windows/>

Replicación

- Redundancia y tolerancia a fallas
- Zero downtime, para upgrades y mantenciones
- Replicación Master-slave en dos modalidades:
 - Strong Consistency
 - Delayed Consistency
- Replicación Geoespacial

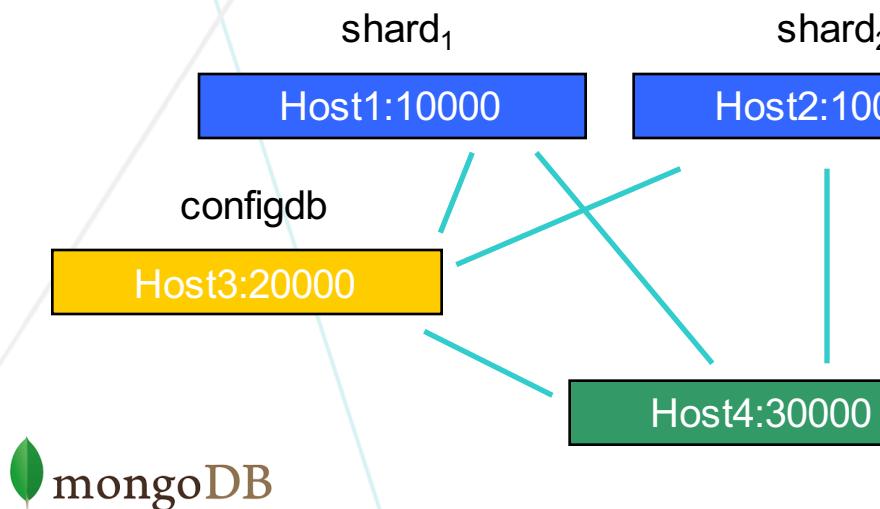




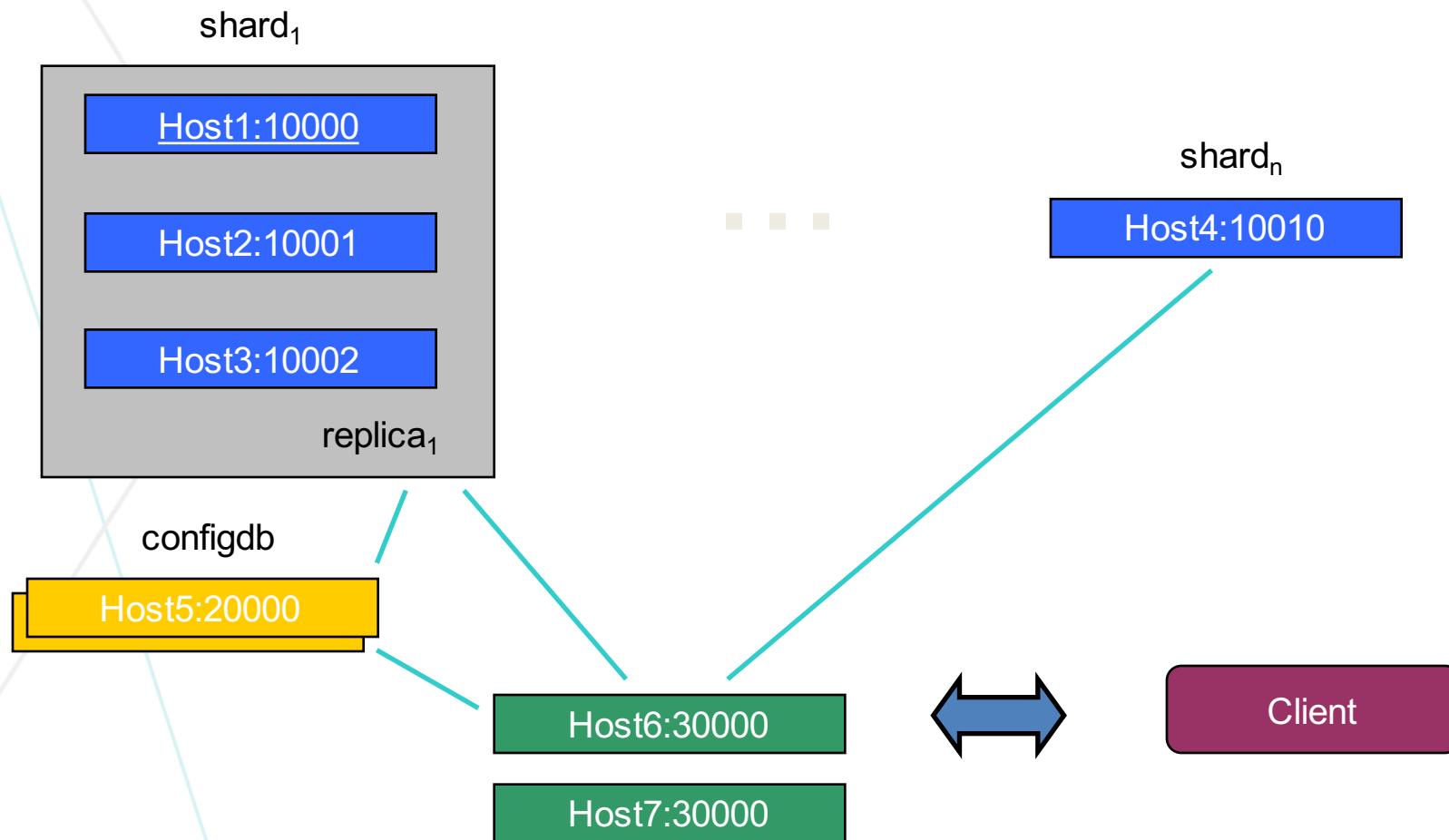
Sharding

- Particionamiento de los datos
- Escalamiento en el throughput de escritura
- Incremento de la capacidad
- Auto balanceo

id	company	customer	article	currency	price
4250250	020	073000	5994537812	00	142,50
4250251	020	073000	5994537852	00	141,12
4250252	020	073000	5994537854	00	105,99
4250253	020	073000	5994537856	00	109,52
4250254	020	073000	5994537862	00	131,49
4250255	020	073000	5994567308	00	29,86
4250256	020	073000	5994567422	00	57,13
4250257	020	073000	5994567428	00	68,59
4250258	020	073000	5994605089	00	51,09
4250259	020	073000	5994607975	00	93,93
4250260	020	073000	5994701005	00	74,22



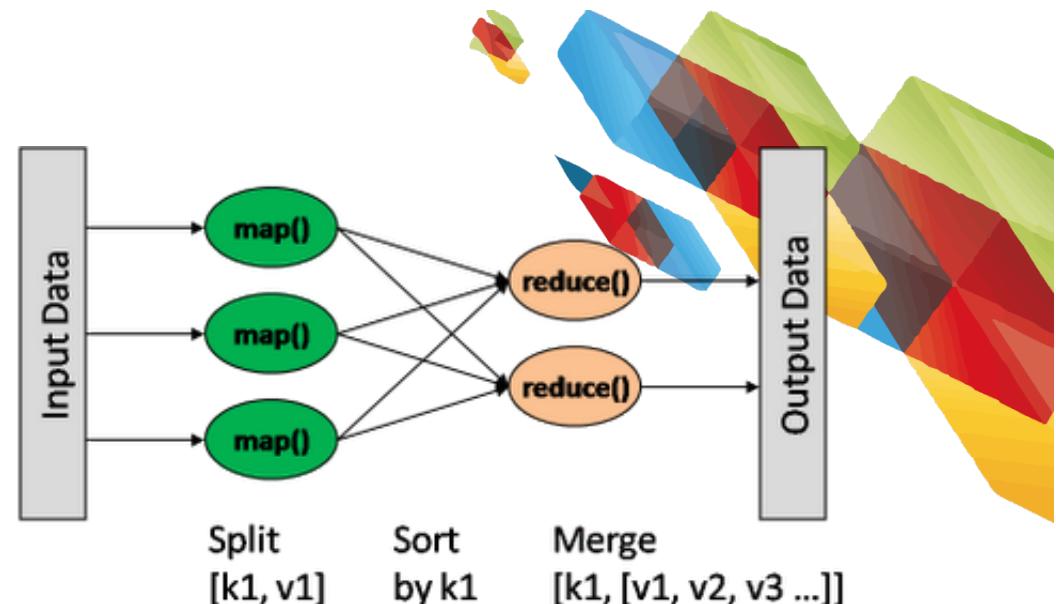
Configuración Mixta



Map/Reduce

```
db.collection.mapReduce(  
    <mapfunction>,  
    <reducefunction>,  
    {  
        out: <collection>,  
        query: <>,  
        sort: <>,  
        limit: <number>,  
        finalize: <function>,  
        scope: <>,  
        jsMode: <boolean>,  
        verbose: <boolean>  
    }  
)
```

```
var mapFunction1 = function() { emit(this.cust_id, this.price); };  
  
var reduceFunction1 = function(keyCustId, valuesPrices)  
{ return sum(valuesPrices); };
```



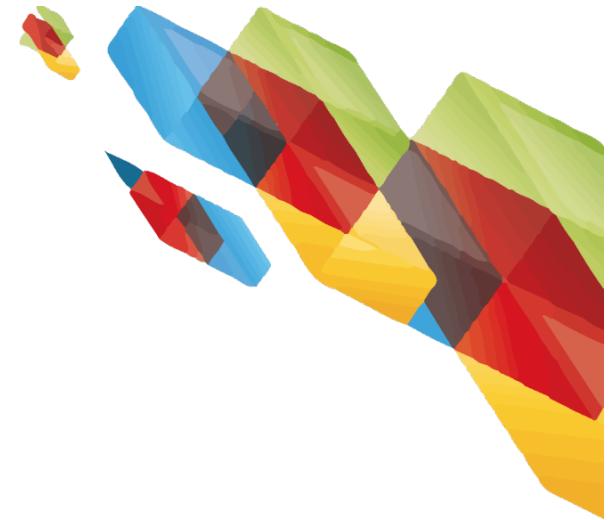
Instalación



En el directorio mongodb/bin ejecutar el programa mongod.exe para iniciar el motor de base de datos.

Para establecer la conexión con el servidor, abrir otra ventana en el mismo directorio y ejecutar el programa mongo.exe. Esto permite acceder al shell de MongoDB.

<http://docs.mongodb.org/manual/tutorial/getting-started/>



CRUD Básico

Create, Read, Update, Delete

CRUD: Utilizando el Shell

Para chequear cual base de datos se está usando se utiliza el comando: **db**

Mostrar todas las bases de datos: **show dbs**

Cambiar de base de datos o crear una: **use <nombre>**

Ver las colecciones existentes: **show collections**

Nota: Las bases de datos no son creadas hasta que se insertan datos

CRUD: Utilizando el Shell

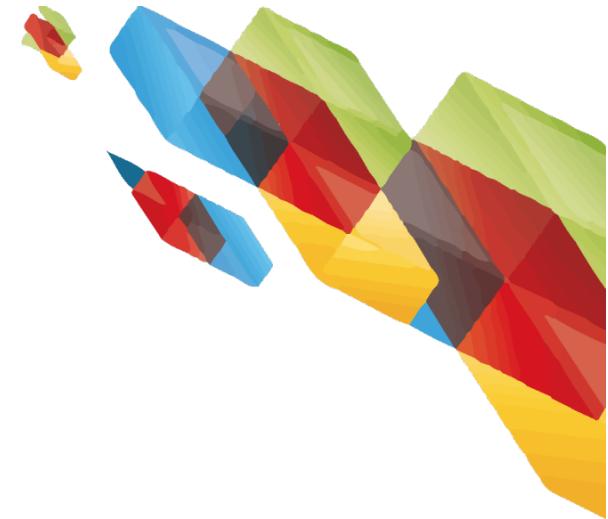
Para insertar documentos en una colección existente o **crear** una colección:

```
db.<colección>.insert(<documento>)
```

<=>

```
INSERT INTO <tabla>
VALUES(<valoresatributos>);
```

CRUD: Insertando datos



Insertando un documento

```
db.<colección>.insert({<campo>:<valor>})
```

Insertar un documento con un campo que no existe en la colección es algo completamente soportado y aceptado por el modelo de objetos BSON.

Para insertar múltiples documentos se puede usar un arreglo.

CRUD: Consultas

- ▶ Se hacen sobre las colecciones
- ▶ Obtener todos: `db.<colección>.find()`
 - ▶ Retorna un cursor, el cual muestra en el shell los primeros 20 resultados.
 - ▶ Se puede agregar `.limit(<número>)` para limitar los resultados
 - ▶ `SELECT * FROM <tabla>;`
- ▶ Obtener un documento: `db.<colección>.findOne()`

CRUD: Consultas

Para obtener un valor específico:

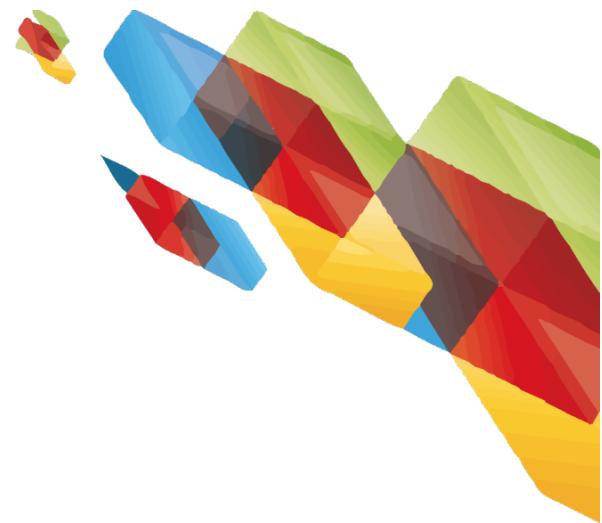
```
db.<colección>.find({<campo>:<valor>})
```

“AND”

```
db.<collection>.find({<campo1>:<valor1>,
                      <campo2>:<valor2>
                    })
```

```
SELECT *
FROM <tabla>
WHERE <campo1> = <valor1>
AND <campo2> = <valor2>;
```

CRUD: Consultas



OR

```
db.<colección>.find({ $or: [  
    <campo>:<valor1>  
    <campo>:<valor1>      ]  
})
```

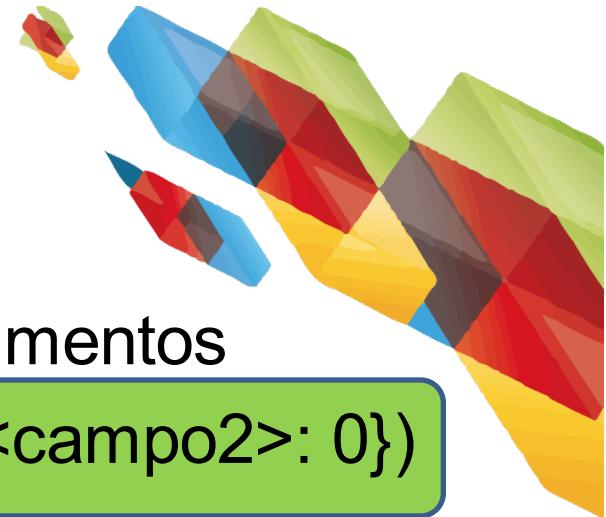
SELECT *

FROM <tabla>

WHERE <campo> = <valor1> OR <campo> = <valor2>;

Múltiples valores en el mismo campo

```
db.<colección>.find({<campo>: {$in [<valor1>, <valor2>]}})
```



CRUD: Consultas

Incluyendo/excluyendo campos de los documentos

```
db.<colección>.find({<campo1>:<valor>}, {<campo2>: 0})
```

```
SELECT campo1  
FROM <tabla> where <campo1> = <valor>;
```

```
db.<colección>.find({<campo>:<valor>}, {<campo2>: 1})
```

Encontrar documentos con un campo

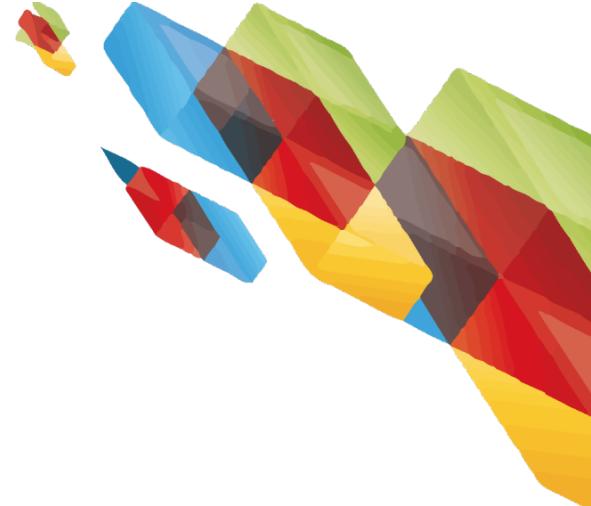
```
db.<colección>.find({<campo>: { $exists: true}})
```

CRUD: Actualización

```
db.<colección>.update(  
  {<campo1>:<valor1>}, //todos los docs con campo1 = valor1  
  {$set: {<campo2>:<valor2>}}, //cambiar el valor del campo2  
  {multi:true} )           //actualizar múltiples documentos
```

upsert: Crea un nuevo documento cuando no se obtienen resultados a partir del criterio de búsqueda.

```
UPDATE <tabla>  
SET <campo2> = <valor2>  
WHERE <campo1> = <valor1>;
```



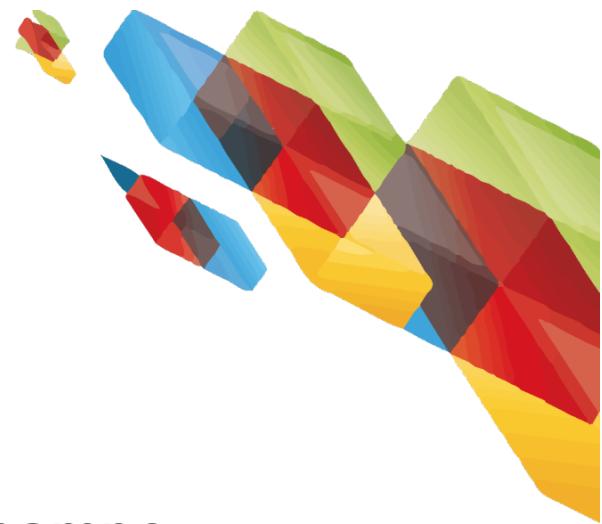
CRUD: Actualización

Para eliminar un campo

```
db.<colección>.update({<campo>:<valor>},  
                      { $unset: { <campo>: 1}})
```

Reemplazar todos los valores de un objeto

```
db.<collection>.update({<campo>:<valor>},  
                      { <campo>:<valor>,  
                        <campo>:<valornuevo>})
```



CRUD: Eliminar

Eliminar todos los registros para un valor de un campo

```
db.<colección>.remove({<campo>:<valor>})
```

```
DELETE FROM <tabla>
WHERE <campo> = <valor>;
```

Borrar solo el primer documento

```
db.<colección>.remove({<campo>:<valor>}, true)
```

CRUD: Aislación

- Por omisión, todas las escrituras son atómicas, a nivel de documento.
- Esto implica que todas las escrituras pueden ser intercaladas con otras operaciones.
- Se puede aislar las escrituras en colecciones que no estén en sharding (**unsharded**) agregando la opción \$isolated:1 el área de consulta :

```
db.<collection>.remove({<field>:<value>, $isolated: 1})
```