mongoDB

Enrique Barra

MEAN

Recursos online

- Un tutorial muy muy completo:
  - http://www.tutorialspoint.com/mongodb/index.htm
- Página oficial:
  - https://www.mongodb.org/
- Documentación:
  - https://docs.mongodb.com/manual/

Historia

- 2007 – primer desarrollo por 10gen
- 2009 – open source
- 2010 – production ready (v1.4)
- 2013 – MongoDB cierra $150M en financiación
- 2017 – v3.6
- 2018 – v4.0
- Today –
  - más de $231M invertidos desde 2007
  - MongoDB inc. se valora en $14200M

En Producción

10gen | the MongoDB company

https://www.mongodb.com/who-uses-mongodb

QUÉ ES MONGODB

- MongoDB es una tecnología de base de datos NoSQL
- El nombre viene de la palabra en ingles “hu**mongo**us” que significa enorme
- MongoDB es orientada a **documentos** (recordar hay: clave-valor, grafo, documento y columnas)

- Una **base de datos MongoDB** contiene **colecciones**

- Una **colección** esta formada por **documentos**

- Cada **documento** esta compuesto de campos

Características MongoDB

- **Consultas Ad hoc**
  - MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares.
  - Las consultas pueden devolver un campo específico del documento pero también puede ser una función JavaScript definida por el usuario.
- **Indexación**
  - Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios.
  - El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales. Para mejorar búsquedas y ordenamientos.
- **Replicación**
  - MongoDB soporta el tipo de 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 del que se caiga el servicio con el maestro actual.

Características MongoDB

- **Balanceo de carga**
  - MongoDB se puede escalar de forma horizontal usando el concepto de "shard"
  - El desarrollador elije una clave shard, la cual determina cómo serán distribuidos los datos en una colección
  - Un shard es un maestro con uno o más esclavos
  - 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
- **Almacenamiento de archivos**
  - MongoDB puede ser utilizado con un sistema de archivos, tomando la ventaja de la capacidad que tiene MongoDB para el balanceo de carga y la replicación de datos utilizando múltiples servidores para el almacenamiento de archivos
  - Esta función (que es llamada GridFS) está incluida en los drivers de MongoDB y disponible para los lenguajes de programación que soporta MongoDB
- **Agregación**
  - La función MapReduce y el operador aggregate() puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación
  - Estos mecanismos permiten que los usuarios puedan obtener el tipo de resultado que se obtiene cuando se utiliza el comando SQL "group-by"
- **Ejecución de JavaScript del lado del servidor**
  - MongoDB tiene la capacidad de realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas: db.system.js.save
  - http://docs.mongodb.org/manual/tutorial/store-javascript-function-on-server/

¿Qué es un documento?

- No es más que un JSON (aunque se almacenará como BSON):

```
{
    _id: “123”,
    title: "MongoDB: The Definitive Guide",
    authors: [
       { _id: "kchodorow", name: "Kristina Chodorow“ },
       { _id: "mdirold", name: “Mike Dirolf“ }
    ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    thumbnail: BinData(0,"AREhMQ=="),
    publisher: {
        name: "O’Reilly Media",
        founded: 1980,
        locations: ["CA”, ”NY” ]
    }
}
```

ID DEL DOCUMENTO

- _id: “123”
- _id: ObjectId(7df78ad8902c)

- Sirve para asegurar la unicidad del documento
- Se puede asignar al insertar o si no se asigna será Mongo el que lo autoasigne

- Si lo asignamos nosotros podemos poner cualquier número (que sea único). Por ejemplo un buen id sería el DNI
- Si lo asigna Mongo será del tipo ObjectId(7df78ad8902c) :
  - Número hexadecimal de 12 bytes
  - Los primeros 4 bytes son del timestamp, los 3 siguientes el id de la máquina, los 2 siguientes el id de proceso de mongodb y los últimos 3 un valor incremental

Terminología y concepto

RDBMS	MongoDB
Base de datos	Base de datos
Tabla, vista	Colección
Fila	Documento (JSON, BSON)
Columna	Campo
Indice	Indice
Join	Documento incrustado y enlaces (embedded document and links)
Primary key	Primary key (por defecto el _id)
Foreign Key	Referencia
Partición	Shard

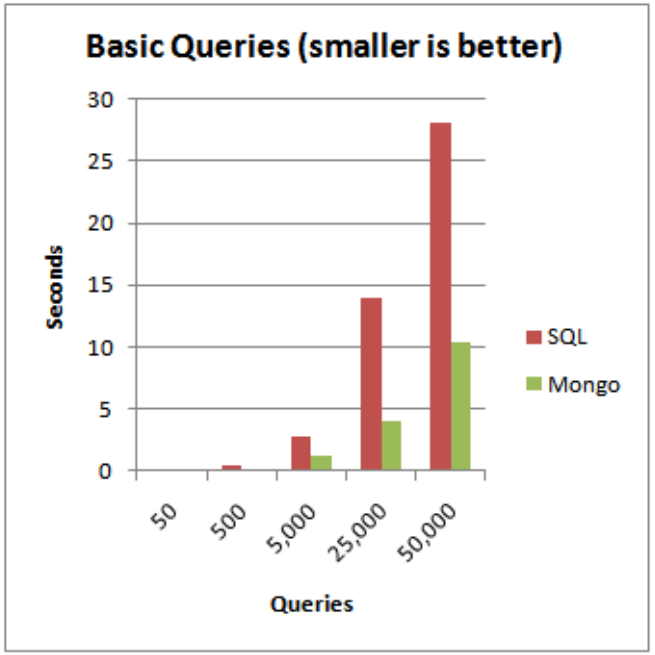
PARECIDOS CON RELACIONAL

- Almacena datos de forma persistente
- Colecciones son como las tablas
- Documentos son como las filas
- Campos del documento con como las columnas
- Tiene índices
- YA hay transactions (desde la versión 4.0)
  - Las operaciones sobre un documento siempre han sido atómicas -> podremos reestructurar nuestros esquema para que sean como transacciones
  - **Desde MongoDB 4.0 hay transacciones multidocumento**
  - Atención a la tolerancia a errores y tardanzas en actualizar

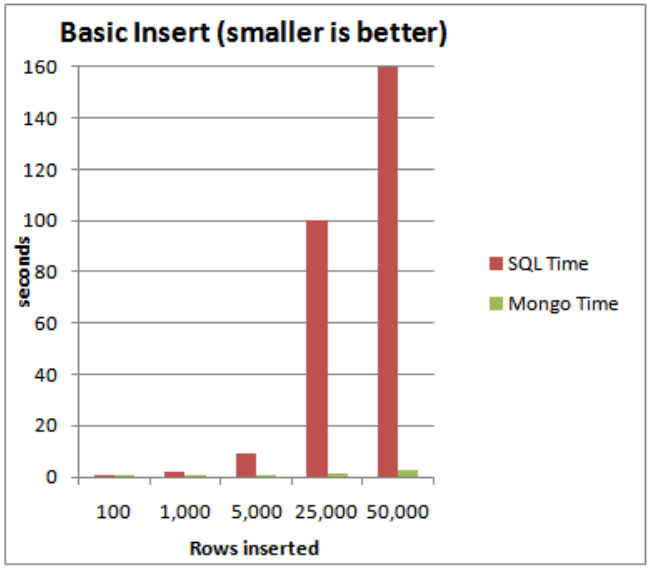
- La diferencia principal viene del hecho de que las bases de datos relacionales definen columnas a nivel de la tabla mientras que bases de datos orientadas a documentos definen sus campos a nivel de documento

DIFERENCIAS CON RELACIONAL

- No tienen esquema declarado
  - Aunque todos los documentos veremos que tienen un esquema similar y que conviene pensarlo bien antes
- No hay join
- No hay constraints
- Aunque tenga transactions (desde v4.0) no hay que abusar, implican bloquear varias colecciones y esperas => disparan los tiempos.



- Si se usan de manera adecuada y para lo que son, resultan mucho más eficientes y escalables



NOVEDADES MONGODB 4.0

- **Multi-Document ACID Transactions**

> IMPORTANT:
>
> In most cases, multi-document transaction incurs a greater performance cost over single document writes, and the availability of multi-document transaction should not be a replacement for effective schema design. For many scenarios, the denormalized data model (embedded documents and arrays) will continue to be optimal for your data and use cases. That is, for many scenarios, modeling your data appropriately will minimize the need for multi-document transactions.

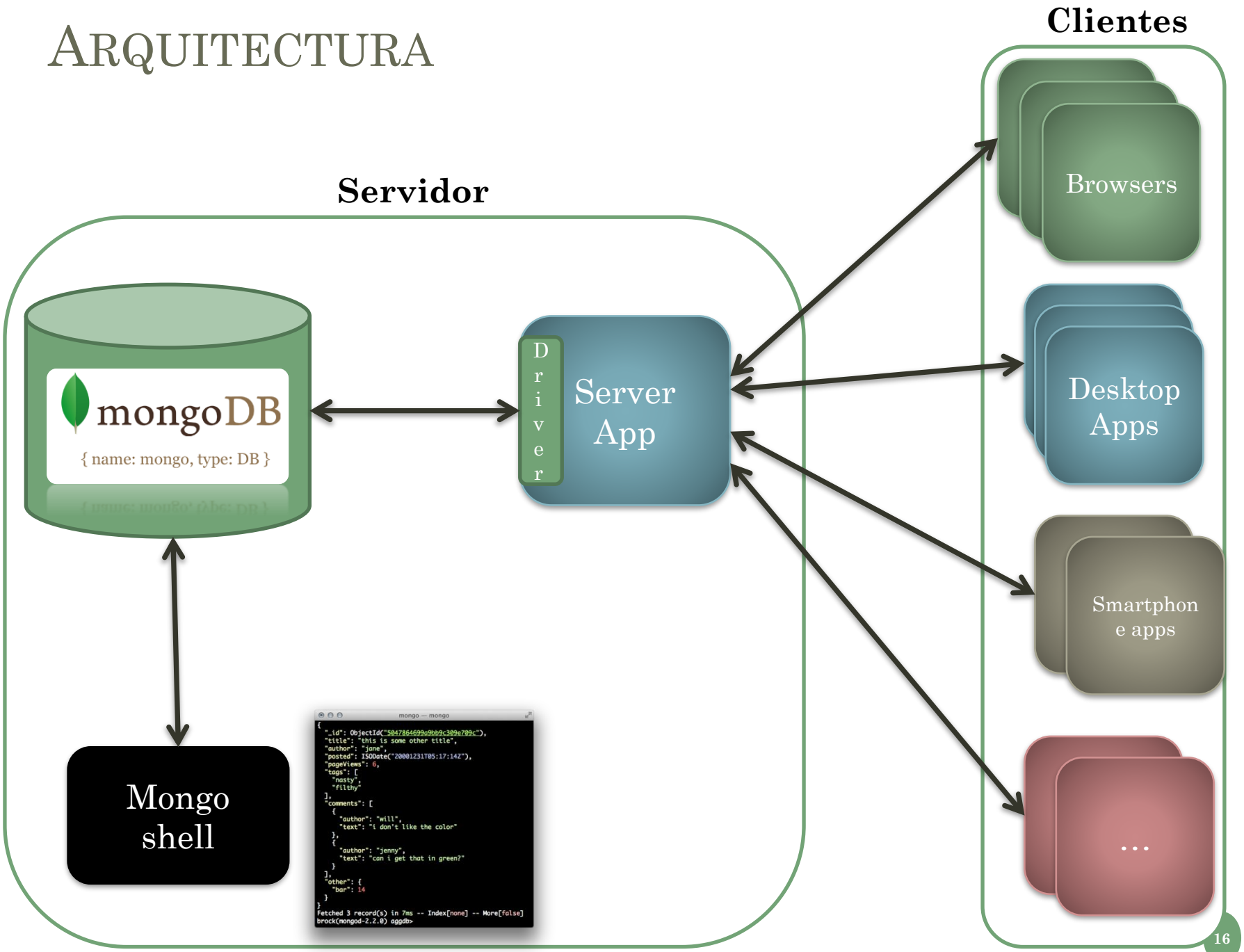
- **Snapshot Read Concern**
  - **readConcern** option helps in achieving consistency, and isolation properties of the data
  - ensures that a consistent view of the data is returned to the client, irrespective of whether that data is being simultaneously modified by concurrent operations
- **Non-Blocking Secondary Reads**
  - MongoDB previously blocked secondary reads while **oplog entries** were applied
- **Extensions to Change Streams**
  - Change streams introduced in version 3.6 helps applications to access real-time data changes without the complexity. In 4.0 Change Streams can be configured to track changes across an entire database or whole cluster

Novedades MongoDB 4.0

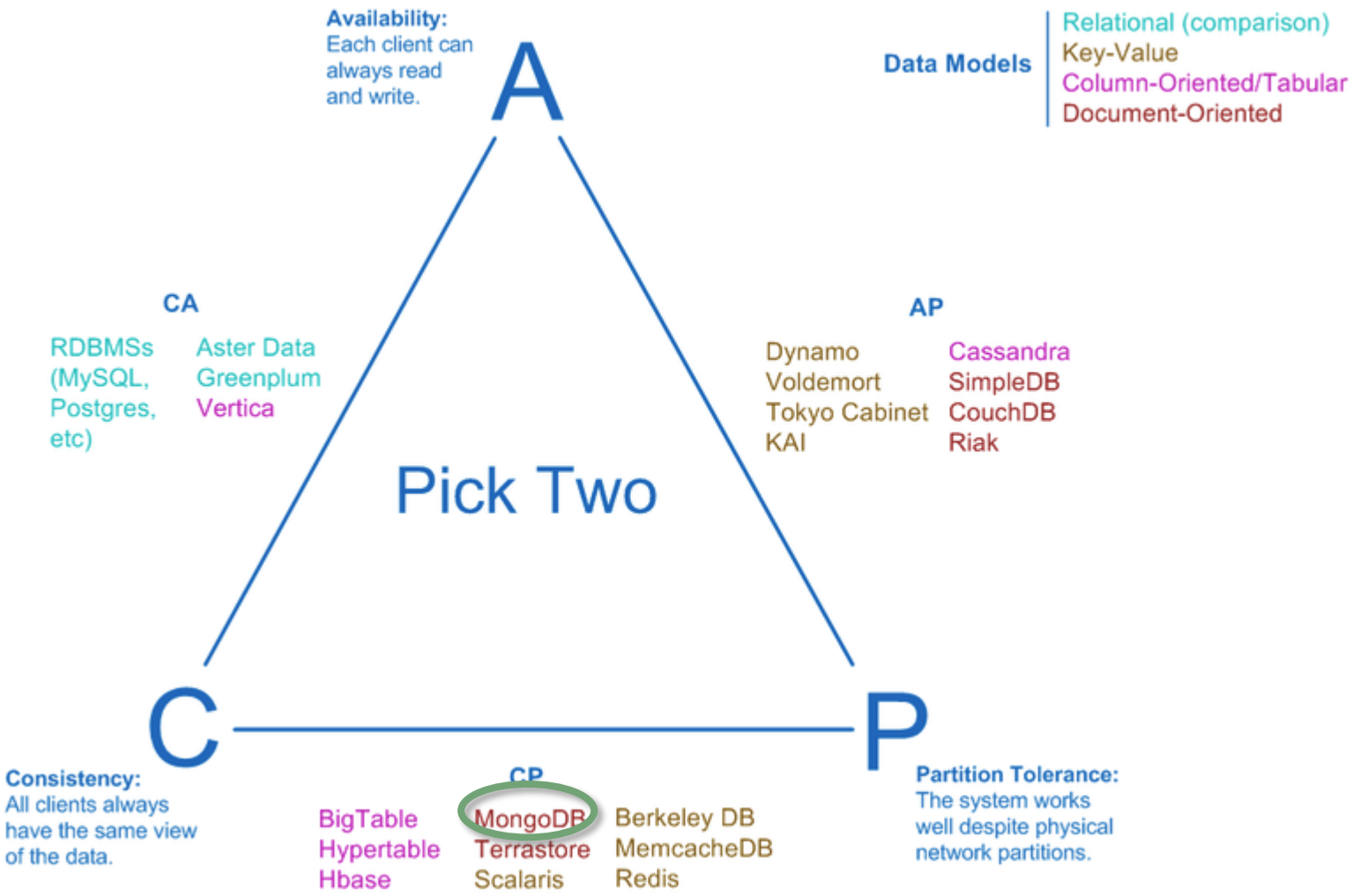
- **Data Type Conversions**
  - A new expression **$convert** has been added to the aggregation framework
- **Improved Migrations Throughput**
- **Improved Sharding Operations**
- **Slow Query Logging on mongos**
- **…**
- **Ver todas:**
- **https://docs.mongodb.com/manual/release-notes/4.0/**
- **https://docs.mongodb.com/manual/release-notes/4.2/**
- **https://docs.mongodb.com/manual/release-notes/4.4/**

- **MongoDB 4.2 features:**
  - **Sharded** Multi-Document ACID Transactions
  - …

Arquitectura



CAP


Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

Recordemos ACID

- ***Atomicity - Consistency - Isolation – Durability***
- **A**tomicidad, **C**onsistencia, aislamiento y **D**urabilidad
- Una secuencia de operaciones (transacción):
  - Se ejecutará del todo o nada **(A)**
  - Una vez completada, la BD quedará en un estado en el que no se viola ninguna restricción de integridad **(C)**
  - Las transacciones concurrentes son independientes y no se afectan unas a otras **(I)**
  - Las modificaciones efectuadas por una transacción podrán recuperarse ante fallas del sistema **(D)**

- En el mundo relacional estamos familiarizados con las transacciones ACID, que garantizar la consistencia y estabilidad de las operaciones pero requieren lockings sofisticados

MONGO Y LAS PROPIEDADES ACID

- Atomicidad:
  - **Transacciones a nivel de documento (y desde 4.0 también multidocumento)**
  - Y como no hay joins, si queremos actualizar varios documentos tendremos que hacerlo por programación o con varias queries separadas
- Consistencia
  - Consistencia de un solo servidor es sencilla de garantizar
  - Pero nodos secundarios pueden estar desactualizados con respecto al primario. Tiene **consistencia eventual**, sólo se garantiza la consistencia tras un periodo de tiempo
- Aislamiento
  - MongoDB tiene un operador **$isolation** (aunque no funciona con particiones ni provee transacciones)
  - https://docs.mongodb.org/manual/reference/operator/update/isolated/
- Durabilidad
  - Por defecto los ficheros se escriben a disco cada 60 seg, y el jounal de operaciones cada **100 ms**
  - Pero es altamente configurable (parámetros s*yncdelay* and *journalCommitInterval)*