

Introducción a MongoDB	2
Bases de datos	4
admin	5
local	5
config	5
Documentos	5
Colecciones	7
Polimorfismo	7
Clave Primaria (_id)	8
ObjectID	9
Diferencias entre MySQL y MongoDB	10
Material Complementario	11
Gestión del servicio de MongoDB 4.2	11
Instalación de MySQL	13

Introducción a MongoDB

[Mongo DB](#) es un gestor de datos NoSQL distribuido de tipo **documental** que almacena documentos en un formato similar a JSON¹ (para ser más exactos internamente usa BSON). JSON (JavaScript Object Notation) es la notación que se utiliza para guardar los documentos.

MongoDB está escrita en C++ y es multiplataforma y Open Source. Sin embargo también tiene soporte comercial.

El proyecto nació a finales del año 2007 como un proyecto interno de una empresa llamada 10Gen para usarlo en una aplicación de Internet que estaban desarrollando, pero en 2009 decidieron liberarlo como Open Source y dedicarse íntegramente a él, ofreciendo soporte comercial y servicios relacionados.

Su nombre proviene de la palabra en inglés Humongous, que significa literalmente "algo realmente grande", y se refiere a su capacidad de gestionar cantidades enormes de datos.

MongoDB 4.2 Virtualizado con Ubuntu 16.04

[Instalación completa - Paso a Paso](#)



Virtualización con Ubuntu 16.04

Si nunca trabajaste con una máquina virtual, revisar este link

[Instalación paso a paso](#)



Sus principales características son:

- Basado en el motor V8 de Google Chrome para JavaScript. Facilidad de aprendizaje por basarse en este lenguaje.
- Almacenamiento flexible basado en JSON sin necesidad de definir esquemas previamente.
- Soporte para creación de índices a partir de cualquier atributo.
- Alto rendimiento para consultas y actualizaciones.

¹ <https://tools.ietf.org/html/rfc7159>

- Consultas flexibles basadas en documentos.
- Alta capacidad de crecimiento, replicación y escalabilidad: puedes escalar horizontalmente simplemente añadiendo máquinas económicas (nodos) sin ver afectado el rendimiento ni complicar la gestión.
- Soporte para almacenamiento independiente de archivos de cualquier tamaño basado en GridFS.
- Soporte para tareas Map-Reduce si es necesario.

Este tipo de bases de datos no sustituyen a las bases de datos tradicionales, como SQL Server, Oracle o MySQL, sino que las complementan para ciertos tipos de aplicaciones especializadas. Las bases de datos documentales como MongoDB se utilizan para multitud de tareas, pero fundamentalmente cuando necesitamos flexibilidad en la definición de los datos, sencillez a la hora de acceder a éstos, gran rendimiento y posibilidad de crecer muy rápido.

Es adecuada para crear aplicaciones de Internet que registren muchos datos o que simplemente queramos crear de manera muy flexible, pero también para sistemas muy grandes como registradores de datos de sensores, que pueden llegar a recibir decenas o cientos de miles de lecturas de datos por segundo, pasando por gestores de datos de ventas, infraestructura de almacenamiento para redes sociales, juegos masivos online, gestores de contenidos, aplicaciones de análisis de datos y reporting.

Actualmente lo utilizan para empresas como eBay, Foursquare, SourceForge, The New York Times, The Guardian, SAP, o el propio CERN en su gran colisionador de hadrones, pero también muchas empresas pequeñas que quieren poder desarrollar de manera ágil, barata, sencilla y sin miedo a poder crecer más adelante. Al estar basada en JavaScript se lleva especialmente bien con todo tipo de aplicaciones web.

¿Qué nos encontramos en una solución que utiliza una base de datos MongoDB?



Veamos rápidamente algunas diferencias que nos ayudarán a entender mejor la esencia de esta base de datos:

MySQL	MongoDB
Base de Datos	Base de Datos
Tabla	Colección
Fila	Documento
Columna	Clave
Joins	Documentos Embebidos

Bases de datos

Una base de datos en MongoDB tiene un concepto similar al que usamos en MySQL. Las bases de datos agrupan [colecciones](#). Una única instancia en MongoDB puede manejar varias bases de datos. Cada base de datos, puede contener una, ninguna o varias colecciones.

Las bases de datos tienen las siguientes características:

- Cada base de datos puede tener sus propios permisos y archivos del sistema operativo por separado.
- Una buena práctica indica que es convenientes agrupar todas las colecciones de una aplicación específica en la misma base de datos.
- Las bases de datos separadas son útiles cuando se almacenan datos para diferentes aplicaciones que usan el mismo servidor de MongoDB.

Las bases de datos se identifican por un nombre. Tienen las siguientes restricciones:

- No es posible usar una cadena o string vacío.
- El nombre de la base de datos no puede contener ninguno de los siguientes caracteres: \ / . " * > < : | ¿ ? \$ espacio o \0 (valor nulo).
- Los nombres de las bases de datos se distinguen entre mayúsculas y minúsculas.
- Los nombres están limitados por un máximo de 64 bytes.

Las bases de datos en MongoDB se pueden consultar de la siguiente manera:

```
> show dbs
admin    0.000GB
config   0.000GB
local    0.000GB
```

admin

Es el nombre de la base de datos root. Si un usuario es añadido a esta base de datos, entonces el usuario obtiene permisos para todas las bases de datos. Existen determinados comandos que solo pueden ser ejecutados desde esta base de datos tales como listar todas las bases de datos o apagar el servidor.

local

Esta base de datos nunca será replicada y sirve para almacenar cualquier colección que debería ser local al servidor.

config

Para configuraciones de sharding, MongoDB usa esta base de datos, para almacenar información acerca de los shards que se crean.

Mediante la concatenación del nombre de una base de datos con una colección de dicha base de datos se consigue una cualificación entera del nombre de la colección denominado espacio de nombres. Por ejemplo si se usa la colección blog.posts en la base de datos unaj, el espacio de nombres de esa colección será unaj.blog.posts. Los espacios de nombres están limitados a 121 bytes de longitud, aunque en la práctica es mejor que sean menores de 100 bytes.

Documentos

Los documentos son la unidad básica de almacenamiento y organización de la información en MongoDB.

Su equivalente en el paradigma relacional es “la fila” o “la tupla”.

Recursos Adicionales

[Editor de documentos JSON](#) - [Otro editor JSON](#)

[Generador de documentos JSON aleatorio](#)



Dicho de otra manera, un documento es un conjunto ordenado de claves. Estas claves tienen asociados valores. Ejemplo de documentos:

```
1 {  
2   "nombre" : "pablo",  
3   "apellido" : "sabatino",  
4   "gustos" : ["chocolate", "futbol", "helado"]  
5 }
```

```
1 {  
2   "club" : "independiente",  
3   "estadio" : "libertadores de america",  
4   "capacidad" : 95000,  
5   "copas" : 19  
6 }
```

Los documentos en JSON siempre empiezan con una llave { y se cierran con otra } y están formados por pares “clave” : “valor”.

Características de los documentos:

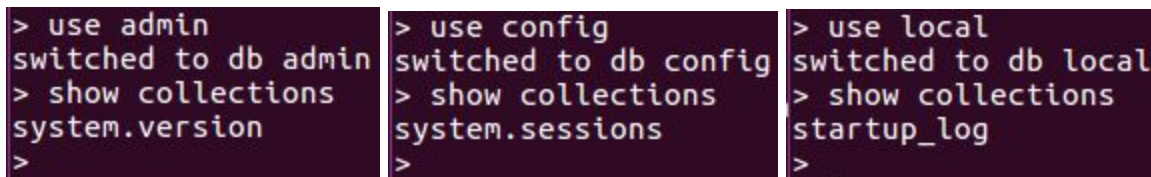
- Las claves son strings por lo que es posible cualquier tipo de carácter. Tener cuidado porque esta flexibilidad puede traer inconvenientes si no se usa convenientemente. Algunas restricciones son: Las claves no puede contar el carácter nulo \0. El punto . y el \$ deben evitarse dado que tienen características especiales en MongoDB.
- MongoDB es *key sensitive* (sensitivo tanto a mayúsculas como a minúsculas, como así también a los tipos de datos).
- Los documentos no pueden tener claves duplicadas. Los siguientes documentos son incorrectos:
- Los pares clave-valor están ordenados en los documentos. Por ejemplo el

documento:

Hay que tenerlo presente durante el desarrollo de las aplicaciones, ya que, MongoDB podría re-ordenarlos automáticamente en determinadas situaciones.

Colecciones

Una colección es un grupo de [documentos](#) de una base de datos en MongoDB. Constituye lo que serían las tablas en las bases de datos relacionales. Consultemos las colecciones de las bases de datos que tenemos:



```
> use admin
switched to db admin
> show collections
system.version
>

> use config
switched to db config
> show collections
system.sessions
>

> use local
switched to db local
> show collections
startup_log
>
```

Polimorfismo

Las colecciones son polimórficas, tienen esquema dinámico o sin estructura (schemaless), es decir, en una colección podemos tener muchos documentos y cada uno con una estructura diferente. Por este motivo, podemos decir que MongoDB es flexible.

Todos estos documentos, tienen diferentes claves y diferentes valores en algunos casos para dichas claves. Esto permite que sea muy flexible, dado que cualquier colección puede contener todos documentos con diferentes estructuras. La cuestión es entonces del porqué la necesidad de separar los documentos en colecciones, dada la flexibilidad que presenta MongoDB:

Mantener documentos con diferentes estructuras (posible en MongoDB) en una única colección, produciría inconvenientes para asegurar que cada consulta solo recupera documentos de un cierto tipo o que en el código de la aplicación implementa consultas para cada tipo de documento.

Una colección se identifica por su nombre. Se debe tener en cuenta las siguientes consideraciones:

- Una cadena vacía no es un nombre válido para una colección.
- Los nombres de las colecciones no pueden contener el carácter nulo \0. Debido a que el \0 se utiliza para indicar el final del nombre de una colección.
- No se debe crear una colección que empiece con SYSTEM puesto q es una palabra reservada. La usa MongoDB para colecciones internas. La colección system.users contiene los usuarios de la base de datos, la colección system.namespaces contiene información acerca de todas las colecciones de la base de datos.
- Para el nombre de las colecciones tampoco debe usarse el \$.

Una convención para organizar las colecciones consiste en definir subcolecciones usando espacios de nombres separados por punto (.). Por ejemplo una aplicación que tenga un blog podría usar la colección <blog.posts> y otra colección <blog.autores>.

Tener presente que nuestros documentos no pueden tener un tamaño superior a 16 MB. Es algo con lo que tendremos que tener cuidado cuándo todos nuestros datos estén dentro de mismo documento.

Aprendiendo un poco más...
[de colecciones y documentos](#)



Tipos de Datos
[Soportados por MongoDB 4.2](#)



Clave Primaria (_id)

Cada uno de los documentos tiene una clave "_id". Algunas características:

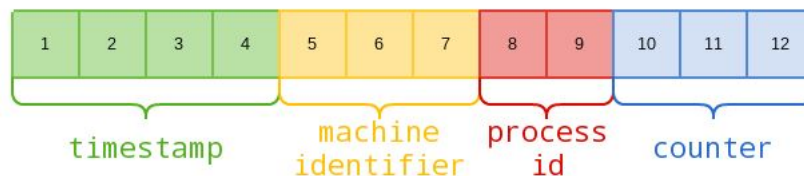
- El valor de esta clave puede ser de cualquier tipo si la gestiona el administrador de aplicaciones. Por defecto es del tipo ObjectID.
- Dentro de una colección, cada uno de los documentos debe tener un valor único y no repetido para la clave "_id" lo que garantiza que cada documento puede ser identificado dentro de la colección de manera única. Dos colecciones podrían tener

un documento con el mismo “_id” pero dentro de una colección no podría haber dos documentos con el mismo valor de “_id”.

- El tipo de datos para el “_id” si bien podría ser cualquiera, conviene que sea del tipo ObjectId. El tipo ObjectId es el tipo por defecto para los valores asociados a las claves “_id”. Es un tipo de datos especialmente diseñado para ser usado en ambientes distribuidos de manera que permita disponer de valores que sean únicos globalmente (en un cluster).

ObjectId

Cada valor ocupa 12 bytes, lo que permite representar una cadena de 24 dígitos hexadecimales (2 dígitos por cada byte).



Timestamp

Los primeros 4 bytes – del 1 al 4 – (timestamp o marca de tiempo)

Los primeros 4 bytes son una marca de tiempo en segundos. Identifican cuando el documento fue creado. Es el tiempo que pasó desde la [epoch](#) (tiempo unix). Debido a que aparecen en primer lugar, los ObjectIDs se ordenan obligatoriamente en orden de inserción, lo que produce que la indexación sobre los ObjectIDs sea muy eficiente.

Recurso adicional: [Convertidor epoch](#)

Lectura adicional: [Repasemos que es el timestamp o marca de tiempo](#)



Identificador de Máquina

Los siguientes 3 bytes – del 5 al 7 – (machine identifier o identificador de máquina) Estos 3 bytes alojan básicamente la información donde residen los datos. Se guarda el host del nombre del equipo en hash (MD5). Al tener estos 3 bytes, asegura que dos máquinas diferentes no podrán generar el mismo ObjectId. Por lo tanto, no podrán

generar colisiones.

ID de Proceso

Los siguientes 2 bytes – del 8 al 9 – (process id o ID de proceso)

Estos 2 bytes alojan la información respecto del ID del proceso. Estos 2 bytes aseguran que dos procesos diferentes en el mismo equipo no generarán el mismo ObjectID.

Estos primeros 9 bytes aseguran la unicidad a través del equipo y proceso para un segundo dado. Y los últimos 3 bytes son un contador que se incrementa y que es el responsable de la unicidad dentro de un segundo en un proceso.

Contador o Incremental

Los siguientes 3 bytes – del 10 al 12 – (counter o incremental)

Estos 3 bytes básicamente guardan un valor incremental. Teniendo estos 3 bytes, se asegurará que dentro del mismo segundo, en la misma máquina y del mismo proceso, se generará un ObjectID diferente.

Este sistema permite generar 2563 únicos ObjectIDs por proceso por segundo.

Cuando se inserta un documento, si el usuario no especifica un valor para la clave “_id”, mongoDB genera y asigna uno automáticamente.

Todos los documentos tienen “_id” que constituye como si fuera la primary key.

Los “_id” proveen detalle respecto del timestamp cuando el documento fue creado. Por lo tanto, no será más necesario agregarle fecha de creación a un documento dado que es posible obtenerlo del “_id”.

Diferencias entre MySQL y MongoDB

1. Representación de datos en las dos bases de datos. En MongoDB, los datos son una colección de documentos JSON mientras que en MySQL los datos están en tablas y filas. Formato de las bases de datos MongoDB en el JSON (Java Script Object Notation). JSON permite la transferencia de datos entre aplicaciones web y servidores utilizando un formato legible por humanos. Antes de JSON, XML se utilizó para esto. JSON se define en BSON de MongoDB (Binary JSON). El

formato binario de BSON proporciona una fiabilidad y una mayor eficiencia cuando se trata de velocidad (transferencia) y espacio de almacenamiento.

2. Uno de los mayores beneficios de las bases de datos relacionales como MySQL es la operación join (de hecho es una de las características de las base de datos relacionales). La operación permite consultas complejas en las bases de datos. MongoDB no soporta joins.
3. Con MongoDB, se puede tener un documento dentro de otro (se dice que están embebido).
4. MySQL soporta transacciones atómicas. Puede tener varias operaciones en una transacción y se puede utilizar el rollback y el commit. No hay soporte para las transacciones en MongoDB y la operación solo es atómica a nivel de documento.
5. Los documentos en MongoDB son polimórficos. Podemos desarrollar aplicaciones más rápido que utilizando el paradigma relacional.
6. En MongoDB des-normalizamos.
7. Una de las ventajas de MySQL sobre MongoDB es la consolidación en el mercado. Esto es principalmente porque NoSQL (MongoDB) es relativamente nuevo, mientras que MySQL existe desde hace años.
8. Las transacciones de MySQL como RDBMS cumplen las propiedades ACID. Esto no está presente en la base de datos MongoDB. Nota: Existe un cambio a partir de MongoDB 4.0 (aún no es la versión estable al momento de escribir este documento 1Q2018).
9. Escala horizontalmente (para aplicaciones distribuidas).
10. Utiliza servidores de baja prestaciones.

Material Complementario

Gestión del servicio de MongoDB 4.2

Monitorear el servicio:

```
pablo@pablo-VirtualBox:~$ sudo systemctl status mongod
[sudo] password for pablo:
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset:
   Active: inactive (dead)
     Docs: https://docs.mongodb.org/manual
lines 1-4/4 (END)
pablo@pablo-VirtualBox:~$
```

Levantar el servicio:

```
pablo@pablo-VirtualBox:~$ sudo systemctl start mongod
pablo@pablo-VirtualBox:~$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset:
   Active: active (running) since dom 2020-04-05 20:47:43 -03; 18s ago
     Docs: https://docs.mongodb.org/manual
    Main PID: 2010 (mongod)
    CGroup: /system.slice/mongod.service
            └─2010 /usr/bin/mongod --config /etc/mongod.conf

abr 05 20:47:43 pablo-VirtualBox systemd[1]: Started MongoDB Database Server.
abr 05 20:47:58 pablo-VirtualBox systemd[1]: Started MongoDB Database Server.
lines 1-10/10 (END)
```

Bajar y chequear el servicio:

```
pablo@pablo-VirtualBox:~$ sudo systemctl stop mongod
[sudo] password for pablo:
pablo@pablo-VirtualBox:~$
```

```
pablo@pablo-VirtualBox:~$ sudo systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset:
   Active: inactive (dead)
     Docs: https://docs.mongodb.org/manual

abr 05 20:47:43 pablo-VirtualBox systemd[1]: Started MongoDB Database Server.
abr 05 20:47:58 pablo-VirtualBox systemd[1]: Started MongoDB Database Server.
abr 05 20:49:24 pablo-VirtualBox systemd[1]: Stopping MongoDB Database Server...
abr 05 20:49:24 pablo-VirtualBox systemd[1]: Stopped MongoDB Database Server.
lines 1-9/9 (END)
```

Conectarnos a la consola (shell de mongoDB):

```
pablo@pablo-VirtualBox:~$ mongo
MongoDB shell version v4.2.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c928bc0b-e181-411d-9428-d49ae3f89f2f")
}
```

Shell de MongoDB:

```
monitoring()
---
>
```

Instalación de MySQL

El proceso de instalación de MySQL 5.7 Virtualizado está aplicado en los siguientes videos:

MySQL 5.7 Virtualizado con Ubuntu 16.04

[*Instalación completa - Paso a Paso*](#)



O podes acceder directamente al video que te interesa conocer:

[Video 0 - Proceso general de Instalación](#)

[Video 1 - Instalación del producto Virtual Box 6.1.14](#)

[Video 2 - Creación de una Máquina Virtual](#)

[Video 3 - Instalación de Ubuntu 16.04](#)

[Video 4 - Instalación de MySQL 5.7 y gestión básica del servicio](#)

Fuentes:

1. Artículo: http://techidiocy.com/_id-objectid-in-mongodb/ [Disponible: Abril 2017]
2. Artículo: <https://www.mongodb.com/es> [Disponible: Abril 2017]
3. Artículo: <http://www.jsoneditoronline.org/> [Disponible: Abril 2017]
4. Artículo: <https://tools.ietf.org/html/rfc7159> [Disponible: Abril 2017]
5. Artículo: <http://codebeautify.org/online-json-editor> [Disponible: Abril 2017]
6. Artículo: <https://www.mongodb.com/> [Disponible: Abril 2020]
7. Artículo:
http://bibing.us.es/proyectos/abreproy/12037/fichero/PFC_Sergio_Bellido_Sanchez%252FTema5_mongodb.pdf [Disponible Abril 2020]

