**Trabajo de Investigación**

**Materia : Base de Datos**

**Alumno : Luis Esteban Broggi**

Tabla de contenido

[¿Qué es un motor de Base de Datos ?](#_¿Qué_es_un)1

[Características del motor seleccionado **MongoDB**](#_2._Características)2

[Ventajas y Desventajas con relación al Motor Seleccionado](#_3.Ventajas_y_Desventajas)3

[Comparativa de motores de BD](#_4._Comparativa_de)4

[Conclusiones](#_5._Conclusiones)5

[Bibliografía](#_6._Bibliografía)6

# **¿Qué es un motor de Base de Datos?**

El Motor de base de datos es el servicio principal para almacenar, procesar y proteger los datos. El Motor de base de datos proporciona acceso controlado y procesamiento de transacciones rápido para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa.

Un motor de BD permite crear base de datos relacionales para el procesamiento de transacciones en línea o datos de procesamiento analíticos en línea. Se pueden crear tablas para almacenar datos y objetos de base de datos como índices, vistas y procedimientos almacenados para ver, administrar y proteger los datos.

Nos encontramos en un momento muy interesante para el sector de las Bases de Datos y hay muchos tipos de bases de datos en el mercado. Por un lado, tenemos las bases de datos relacionales, actualmente más establecidas. Por otro lado, aparecen las bases de datos NoSQL. Decimos aparecen, pero ya están en el mercado desde hace más de 10 años. Junto a estos dos tipos de bases de datos aparecen las bases de datos híbridas (SQL/NoSQL), bases de datos in-memory y bases de datos as a service.

Se puede utilizar opciones SQL como Server Management StudioSQL Server Management Studio para administrar los objetos de bases de datos y SQL Server Profiler para capturar eventos de servidor. También opciones como MySQL, MSSQL, SQLite, MS Access y PostgreSQL entre otros.

U otras opciones comerciales como :

Oracle:

Base de datos que puede correr en casi cualquier sistema operativo. De Oracle destacamos la abundancia de perfiles con experiencia en esta tecnología y la gran cantidad de herramientas que hay para su administración y monitorización.

IBM DB2:

Suele ser la segunda base de datos más utilizada en entornos Unix/Linux después de Oracle. Es claramente un indiscutible ganador en Mainframe. Hay perfiles profesionales para DB2, pero no tanto como para Oracle. Por otro lado, el perfil de DB2 Mainframe no tiene por qué saber moverse en DB2 linux/unix.

Microsoft SQL Server:

Base de datos con compatibilidad únicamente para sistemas Windows. Hay muchos perfiles expertos en SQL Server y no es difícil su adquisición. Su integración con Microsoft Azure ha mejorado mucho su flexibilidad y rendimiento.

Teradata:

Destaca sobre los demás tipos de bases de datos por su capacidad de almacenamiento y de análisis de datos. Suele ser utilizado en grandes instalaciones de Big Data.

SAP Sybase:

Aunque ya no vive los momentos de éxito de hace una década, sigue destacando por su escalabilidad y rendimiento.

Informix

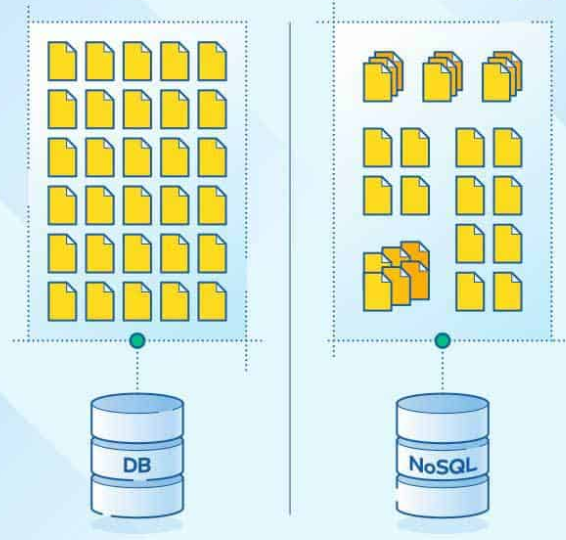
También existen opciones NoSQL ,como se comentaba previamente.

NoSQL (a veces llamado "no sólo SQL") es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas.

Las bases de datos NoSQL utilizan una variedad de modelos de datos para acceder y administrar datos, como documentos, gráficos, clave-valor, en-memoria y búsqueda. Estos tipos de bases de datos están optimizados específicamente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles, lo que se logra mediante la flexibilización de algunas de las restricciones de coherencia de datos en otras bases de datos.

Una diferencia clave entre las bases de datos de NoSQL y las bases de datos relacionales tradicionales, es el hecho de que NoSQL es una forma de almacenamiento no estructurado.

Ejemplo



Esto significa que NoSQL no tiene una estructura de tabla fija como las que se encuentran en las bases de datos relacionales.

¿Cuáles son las más recomendadas?

Orientadas a documentos :

son aquellas que gestionan datos semi estructurados. Es decir, documentos. Estos datos son almacenados en algún formato estándar como puede ser XML, JSON o BSON.

* [MongoDB](http://www.mongodb.org/): probablemente la base de datos NoSQL más famosa del momento. En octubre del año pasado, MongoDB conseguía 150 millones de dólares en financiación, convirtiéndose en una da los startups más prometedores. Algunas compañías que actualmente utilizan MongoDB son Foursquare o eBay.
* [CouchDB](http://couchdb.apache.org/): es la base de datos orientada a documentos de Apache. Una de sus interesantes características es que los datos son accesibles a través de una API Rest. Este sistema es utilizado por compañías como Credit Suisse y la BBC.

Orientadas a columnas:

Este tipo de bases de datos están pensadas para realizar consultas y agregaciones sobre grandes cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros.

En esta categoría encontramos:

* [Cassandra](http://cassandra.apache.org/): incluida en esta sección, aunque en realidad sigue un modelo híbrido entre orientada a columnas y clave-valor. Es utilizada por Facebook y Twitter (aunque dejaron de usarla para almacenar tweets).
* [HBase](http://hbase.apache.org/). Escrita en Java y mantenida por el Projecto Hadoop de Apache, se utiliza para procesar grandes cantidades de datos. La utilizan Facebook, Twitter o Yahoo.

De clave valor:

Estas son las más sencillas de entender. Simplemente contienen una clave y su valor. Cuando se quiere recuperar un dato, simplemente se busca por su clave y se recupera el valor.

En esta categoría encontramos

[DynamoDB](http://aws.amazon.com/es/dynamodb/): desarrollada por Amazon, es una opción de almacenaje que podemos usar desde los Amazon Web Services. La utilizan el Washington Post y Scopely.

* [Redis](http://redis.io/): desarrollada en C y de código abierto, es utilizada por Craiglist y Stack Overflow ([a modo de caché](http://meta.stackoverflow.com/questions/69164/does-stack-overflow-use-caching-and-if-so-how/69172#69172)).

En grafo:

Basadas en la teoría de grafos utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como redes y conexiones sociales.

En esta categoría encontramos:

* [Infinite Graph](http://www.objectivity.com/infinitegraph): escrita en Java y C++ por la compañía Objectivity. Tiene dos modelos de licenciamiento: uno gratuito y otro de pago.
* [Neo4j](http://www.neo4j.org/): base de datos de código abierto, escrita en Java por la compañía Neo Technology. Utilizada por compañías como HP, Infojobs o Cisco.

Como se ve, los tipos son muy diferentes. Si uno elige un modelo SQL o NoSQL, es necesario asegurar bien que es lo que se necesita.

Preguntas comunes a la hora de elegir una base de datos son:

* A cuantos clientes quiero dar servicio de forma concurrente.
* Que tamaño de datos voy a necesitar gestionar.
* Voy a necesitar implementar trabajos en “batch” que accederán a la base de datos
* Qué exigencia de tiempo de respuesta necesito dar a mis clientes
* Cómo voy a escalar mi base de datos según vaya aumentando el número de clientes y transacciones
* Como voy a monitorizar mi base de datos para conseguir el menor posible tiempo de indisponibilidad.
* Necesito una base de datos relacional o una no SQL
* Comportamiento de la base de datos ante caída. Como se comporta con problemas.

Tabla Comparativa SQL vs NoSQL



# **2. Características**

Para este trabajo de investigación seleccionamos el motor de base de datos NoSql **MONGODB**.



Para los que no conocen qué es MongoDB, este es un motor de base de datos NoSQL basado en documentos. Esto quiere decir que Mongo guarda los datos en estructuras parecidas a un JSON.

Antes de comenzar, es buenos familiarizarse con los términos empleados en MongoDB:

Colecciones:

Semejante a una tabla en el mundo de bases de datos relacionales y sirven para agrupar documentos.

Documentos:

La unidad fundamental en que se guarda información dentro de MongoDB, son estructuras parecidas a un JSON. Semejante a una fila en una tabla de una base de datos relacional.

Drivers

Son las librerías empleadas por nuestras aplicaciones para comunicarse con MongoDB.

7 características por las cuales se recomienda MongoDB:

1. Modelamiento de datos

Si estás pensando en emplear NodeJS como tu backend, es posible que hayas escuchado hablar de arquitecturas MERN (Mongo Express React Node.js) o MEAN (Mongo Express Angular Node.js) es común usar Mongo debido a que los datos se almacenan en estructuras parecidas a un JSON, esto hace que el flujo de datos dentro de la aplicación no tenga mayores cambios en la estructura de datos.

Si no estás usando un stack basado en JavaScript, no hay problema ya todos los drivers de MongoDB vienen con serializadores que convierten las estructuras de datos de tu lenguaje de programación en documentos que MongoDB puede entender.

1. Escalabilidad

Si estás esperando un gran flujo de usuarios es ideal que la base de datos que elijas pueda escalar con la demanda, MongoDB al ser una base de datos distribuida puede escalar no solamente de forma vertical (CPU y RAM) sino que también de forma horizontal (creando más nodos).

1. Flexibilidad

El usar una base de datos basado en documentos y que sea Schema Less como lo es MongoDB hace que tu base de datos crezca con tu aplicación sin tener que ejecutar scripts que crean campos con valores por defecto cada vez que quieras agregar un campo nuevo en tus registros. Dentro de MongoDB es normal que los documentos dentro de una colección no tengan exactamente los mismos campos.

1. Alta disponibilidad

Si tienes usuarios en muchas partes del mundo y quieres que las aplicaciones se encuentren distribuidas regionalmente, MongoDB te permite tener clúster distribuidos. Esto mejora la velocidad de consulta al disminuir la latencia que existe entre el clúster de base de datos y el servicio que ejecuta la query. Además de la ventaja adicional de que, si una región no se encuentra disponible, las otras pueden mantener la aplicación disponible.

1. Transacciones

Si vienes del mundo SQL y habías estado prevenido de usar MongoDB debido a que no ofrecía transacciones, te cuento que en su versión 4.0 MongoDB nos trae transacciones ACID (Atomicity, Consistency, Isolation, Durability) entre múltiples documentos.

1. Poderosa sintaxis para hacer consultas

En MongoDB tenemos múltiples operadores que nos permiten crear consultas poderosas con poco código, además que contamos con las Agregaciones que son un mecanismo que nos permite realizar operaciones entre múltiples colecciones.

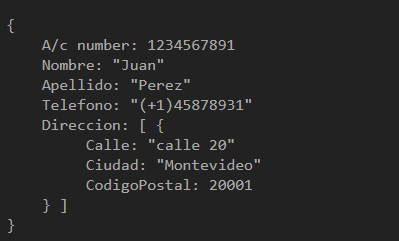
1. Es de código abierto

Esto es una ventaja mayúscula ya que no tienes que pagar licencias para usar MongoDB dentro de tu proyecto de forma comercial, además de que existen muchos proveedores que ofrecen MongoDB como SaaS como es el caso de MongoDB Atlas, MLab entre otros.

Otras características de MongoDB

Los datos guardados con MongoDB son de fácil lectura para los usuarios, como ya mencionamos se guarda en formato JSON por lo cual a modo de ejemplo quedan de la siguiente manera:

Ejemplo



Se puede ver que en los documentos en lugar de crear relaciones con otra tabla se puede especificar de forma anidada más datos, como es el caso de la dirección en el ejemplo. Los documentos nos permiten almacenar toda la información que queramos, lo cual a veces puede ocasionar problemas en la consistencia de los datos.

MongoDB utiliza un lenguaje de consultas no estructurado por lo que realizamos las consultas especificando el nombre del documento con las propiedades que queremos filtrar.

Este tipo de consultas permite una amplia variedad de operadores, los puedes consultar desde su [sitio oficial](https://docs.mongodb.com/manual/reference/operator/query/).

Como vemos en la tabla comparativa MongoDB en el teorema CAP se inclina a CP (Consistencia y Tolerancia a particiones) esto significa que todos los clientes acceden a una vista consistente de la base de datos. Lo cual implica que los usuarios de un nodo deben esperar a que los otros nodos se sincronicen para poder ser visibles y editables, en este caso la disponibilidad queda en segundo plano frente a la consistencia.

En el ámbito de la seguridad MongoDB utiliza un control de acceso basado en roles con privilegios flexibles, sus características de seguridad incluyen autenticación, auditoría y autorización. También permite el uso de TLS/SSL con el propósito de encriptar los datos y que solo sean accesibles para el cliente.

**\*\*\*Datos de interés :**

Modelo de desarrollo : Código abierto

Desarrollador(es) : MongoDB Inc.

Lanzamiento inicial : 2009

Última versión estable: 4.0.8

Última versión en pruebas: 4.1.9

Género Base de datos : NoSQL

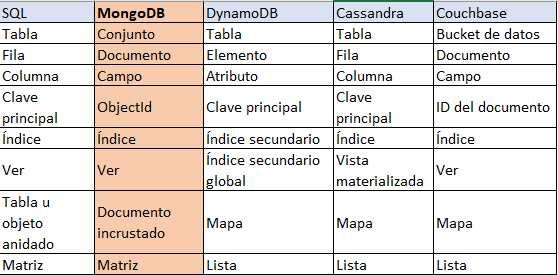
Programado en C++

Sistema operativo : Multiplataforma

Licencia : GNU AGPL v3.0 (drivers: licencia Apache)

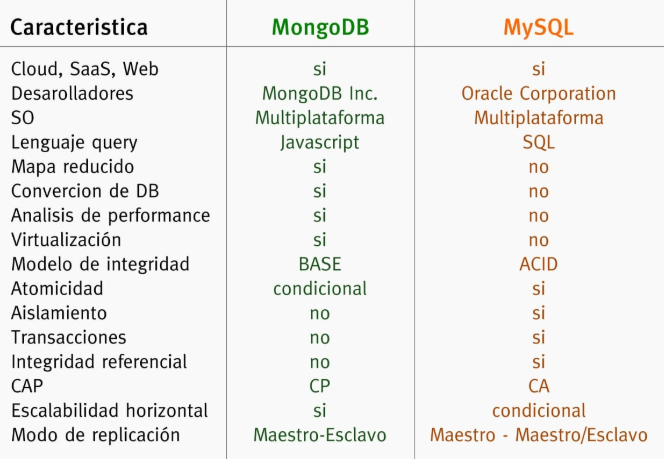
# **3.Ventajas y Desventajas con relación al Motor Seleccionado**

La siguiente tabla compara la terminología utilizada por las bases de datos NoSQL seleccionadas con la terminología utilizada por las bases de datos SQL. En Color el motor de BS seleccionado MongoDB



Ahora revisamos y comparamos contra otro motor BD como MySQL

Tabla comparativa con respecto a MySQL



En cuanto a performance y velocidad, comparación contra MySQL:

Los datos de la gráfica fueron tomados en base a 1.000.000 registros con MySQL 5.7.9 y MongoDB 3.2.0 utilizando las configuraciones por defecto en un servidor Ubuntu con 8 CPUs virtuales y 32 GB de ram en entornos separados.



MongoDB es más rápido que MySQL gracias a su capacidad para manejar grandes cantidades de datos no estructurados, permitiendo realizar consultas de manera sensible al workload (carga de trabajo).   Mientras que MySQL suele ser más lento al momento de manejar grandes bases de datos.

Cuando conviene utilizar MongoDB o MySQL, esta otra tabla comparativa puede ayudar :

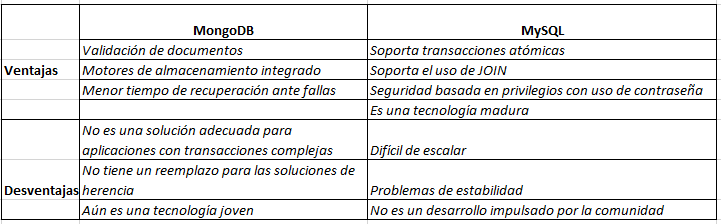


Cuando elegir una u otra siempre va a depender de las necesidades y características de tu proyecto, ninguna es mejor que la otra simplemente es diferente y funcionan mejor para diferentes casos.

MySQL ofrece alto rendimiento, flexibilidad, protección confiable de datos, alta disponibilidad y es fácil de administrar. Si se indexa correctamente puede mejorar notoriamente el rendimiento, facilitar las consultas y mejorar la robustez. Cabe destacar que es una mejor opción si vamos a realizar transacciones complejas.

Por otro lado, MongoDB como ya hemos mencionado es una mejor opción si no tienes un esquema definido o manejas datos no estructurados. Si manejas grandes volúmenes de datos y bases de datos muy grandes MongoDB te ofrecerá un mejor rendimiento, por otra parte, si necesitas almacenar muchos datos como documentos/archivos MongoDB es una mejor opción que te ayudara mucho más en ese aspecto.

Tabla Comparativa de ventas y desventajas:



En cuanto a la Instalación :

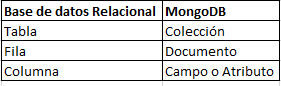
La instalación de una instancia del servidor es muy sencilla.

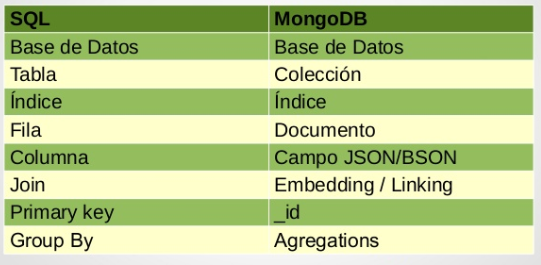
Simplemente tenemos que bajar los binarios para nuestro sistema operativo. Hay versiones par Windows, Linux y MacOs. Una vez bajados podremos arrancar el servicio de MongoDB con un solo comando.

*mongod --dbpath data*

Con este comando arrancamos el servicio mongod, que empezará a escuchar peticiones por el puerto 27017. Es importante indicar el parámetro --dbpath, con la ruta dónde se almacenarán los ficheros de nuestra base de datos.

Comparación con otras bases de dato relacional en base a los siguientes términos :



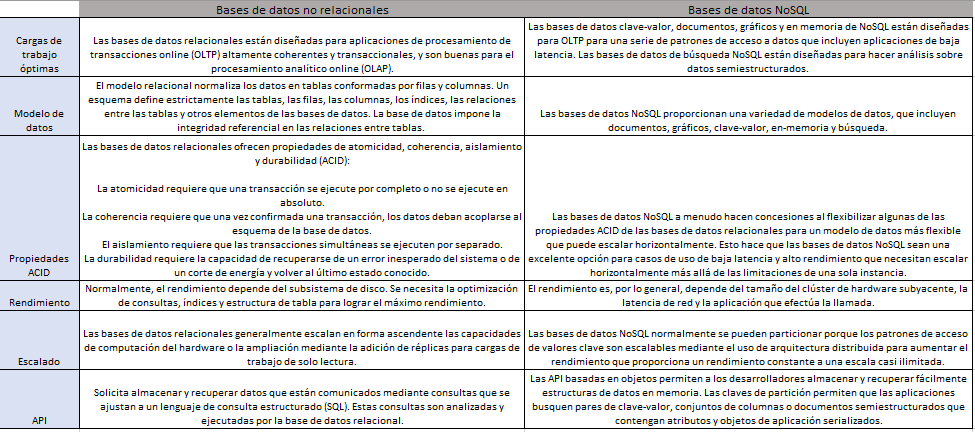


# **4. Comparativa de motores de BD**

SQL (relacional) en comparación con NoSQL (no relacional)

Durante décadas, el modelo de datos predominante utilizado para el desarrollo de aplicaciones era el modelo de datos relacional empleado por bases de datos relacionales como Oracle, DB2, SQL Server, MySQL y PostgreSQL. No fue sino hasta mediados y finales de la década del 2000 que otros modelos de datos comenzaron a adoptarse y aumentó su uso significativamente. Para diferenciar y categorizar estas nuevas clases de bases de datos y modelos de datos, se acuñó el término "NoSQL". Con frecuencia, los términos "NoSQL" y "no relacional" se usan indistintamente.

Aunque hay muchos tipos de bases de datos NoSQL con distintas características, en la tabla siguiente se muestran algunas de las diferencias entre las bases de datos SQL y NoSQL.



NoSQL vs SQL; cuándo utilizar qué tipo de base de datos:

•Cuando los datos deben ser consistentes sin dar posibilidad al error utilizar una base de datos relacional, SQL.

•Cuando nuestro presupuesto no se puede permitir grandes máquinas y debe destinarse a máquinas de menor rendimiento, NoSQL.

•Cuando las estructuras de datos que manejamos son variables, NoSQL.

•Análisis de grandes cantidades de datos en modo lectura, NoSQL.

•Captura y procesado de eventos, NoSQL.

•Tiendas online con motores de inteligencia complejos, NoSQL



# 

# **5. Conclusiones**

**MongoDB** es, actualmente, una de las bases de datos **NoSQL** más populares. MongoDB es una base de datos NoSQL orientada a documentos. A diferencia de las bases de datos relacionales, donde se almacenan los registros como filas de una tabla, con MongoDB estos registros son almacenados en documentos BSON (un derivado de los documentos JSON).

*MongoDB, de la palabra en inglés ''Hu****mongo****us'' que significa : enorme.*

Ya explicamos anteriormente que es un sistema de base de datos NoSQL orientado a documentos.

En base a la información de los capítulos anteriores tenemos claro Lo que MongoDB no puede hacer:

* no hay tablas de BBDD
* no hay JOINS
* no hay transacciones
* no usa un esquema de datos

Usar bases de datos MongoDB es arriesgado, salvo que uno sepa lo que esta utilizando, porque esta tecnología no solo es bastante novedosa y a desarrolladores más experimentados les aporta muchos beneficios (sobre todo, en lenguajes web), pero también tiene sus consecuencias, y hay que tenerlas en cuenta.

Básicamente, MongoDB está muy vinculado a JavaScript (un lenguaje muy relacionado con la Web), y, de hecho, la propia consola de MongoDB es un intérprete JavaScript.

Para trabajar con bases de datos MongoDB (y en general, con todas las bases de datos NoSQL), es necesario y muy importante tener en cuenta una serie de aspectos:

* No existe integridad referencial
* No existe una base de datos separada, sino que es un documento gigante sin ningún tipo de normalización, esto es, aquí no hay reglas de diseño, sino que dependen muy mucho de la información (los registros se llaman documentos)
* El lenguaje SQL no es el que se utiliza para interactuar con la base de datos, sino que es uno nuevo
* Las matrices son base fundamental para trabajar con los documentos

En resumen, se puede guardar lo que se quiera y cómo se quiera, sin ningún tipo de reglas, significa flexibilidad. Siendo esto bueno o malo depende como se lo mire.

En cuanto al modelo relacional ya tiene más de 40 años de uso, así que la evolución de los productos y las herramientas de bases de datos relacionales poseen una gran maduración. Las bases NoSQL aún no están del todo estandarizadas, así que cada una de ellas posee características propias en cuanto a las consultas y no necesariamente mantienen compatibilidad con las instrucciones SQL. Sin embargo, el hecho de que hoy en día se tenga que lidiar con cantidades de datos tan grandes, abren un extenso panorama para estos repositorios de datos. NoSQL es una buena opción para aquellas empresas que detectan problemas de desempeño y escalabilidad o costes debido a grandes volúmenes de datos.

El planteamiento de los ingenieros de MongoDB a la hora de crearla fue simple: *necesitamos una base de datos suficientemente sencilla para proyectos pequeños y que sea capaz de escalar mucho más allá que una base de datos relacional*. Así que crearon una base de datos NoSQL orientada a documentos, en este caso, JSON. Al carecer de integridad referencial, MongoDB permite “repartir” la información entre distintos servidores sin problemas y montar clúster de computación realmente gigantescos. De ahí su nombre.

Si bien MongoDB está surgiendo con mucha fuerza es posible que no sea una buena elección para un próximo proyecto. Puede ser más fácil elegir una base de datos relacional.

La integridad referencial es algo crítico. No es fácil tener un fallo en este sentido.

Si los datos son altamente referenciales y dependen mucho de sus relaciones, es mejor opción usar una base de datos relacional. No obstante, en muchas ocasiones es posible transformar esos datos en un objeto JSON con pocas o ninguna relación.

Si uno se encuentra en un proyecto importante en proceso y sin experiencia en MongoDB ni bases de datos NoSQL. En ese caso es mejor opción ir por la opción relacional.

No obstante, si uno está comenzando un nuevo proyecto y cuenta con el tiempo, puede empezar con MongoDB, hay drivers para una infinidad de lenguajes de programación y es fácil ver el potencial que tiene.

Por lo tanto, si uno cuenta con el tiempo y los recursos puede ser una buena opción. MongoDB es una base de datos distribuida, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube

# 

# **6. Bibliografía**

1. Platzi.com
2. Prezi.com
3. aws.amazon.com/es/nosql/
4. blog. pandorafms.com/blog/es/bases-de-datos-nosql/
5. blog. genbeta.com/desarrollo
6. MongoDB official website : docs.mongodb.com/manual/reference/operator/query/
7. IBM official website: ibm.com/software/data/db2/
8. Oracle official website: oracle.com/us/solutions
9. Blog. blog.bi-geek.com
10. Orientador del Aprendizaje de la UNIDAD 1: ¿Qué es una Base?
11. leanpub.com/mongodbcastellano
12. tutorialesenpdf.com/mongodb/
13. Udemy.com
14. researchgate.net/publication/323184317\_MongoDB
15. Official Web site: mongodb.com/es
16. MongoDB desde Cero (libro PDF)