



mongoDB®



Tecnologías Avanzadas de Desarrollo

UD4: Bases de Datos NoSQL
Bases de Datos NoSQL Documentales: MongoDB

Olga M. Moreno Martín



Licencia



Toda la documentación de esta asignatura queda recogida bajo la licencia de Creative Commons

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

En el caso de incumplimiento o infracción de una licencia Creative Commons, el autor, como con cualquier otra obra y licencia, habrá de recurrir a los tribunales. Cuando se trate de una infracción directa (por un usuario de la licencia Creative Commons), el autor le podrá demandar tanto por infracción de la propiedad intelectual como por incumplimiento contractual (ya que la licencia crea un vínculo directo entre autor y usuario/licenciatario). El derecho moral de integridad recogido por la legislación española queda protegido aunque no aparezca en las licencias Creative Commons. Estas licencias no sustituyen ni reducen los derechos que la ley confiere al autor; por tanto, el autor podría demandar a un usuario que, con cualquier licencia Creative Commons, hubiera modificado o mutilado su obra causando un perjuicio a su reputación o sus intereses. Por descontado, la decisión de cuándo ha habido mutilación y de cuándo la mutilación perjudica la reputación o los intereses del autor quedaría en manos de cCuttacia Juez o Tribunal.

Olga M. Moreno Martín

Índice



Bases de Datos NoSQL

1. Introducción
2. ¿Qué son las bases de datos NoSQL?
3. Tipos
4. Bases de Datos Orientadas a Documentos
5. MongoDB
 1. Introducción
 2. Entorno de trabajo

Introducción



Las bases de datos NoSQL (Not Only Structured Query Language), se apoyan en dos conceptos clave, los cuales marcan la diferencia con los modelos “clásicos” anteriores. Estos conceptos que tienen tanto impacto, son el escalado horizontal, que extiende el almacenamiento y mejora el rendimiento, gracias a una arquitectura distribuida, y la eliminación del uso de la operación “join”, que se hace posible gracias a la organización de los datos al almacenarlos.

Al usar un modelo no relacional, las bases de datos NoSQL tienen la capacidad de gestionar cantidades masivas de datos desestructurados, estructurados y semiestructurados. Las bases de datos NoSQL han evolucionado para incluir una variedad de modelos diferentes.

Cassandra, HBase, Redis o MongoDB son solo varios ejemplos de las más de 225 motores de datos de estilo NoSQL.

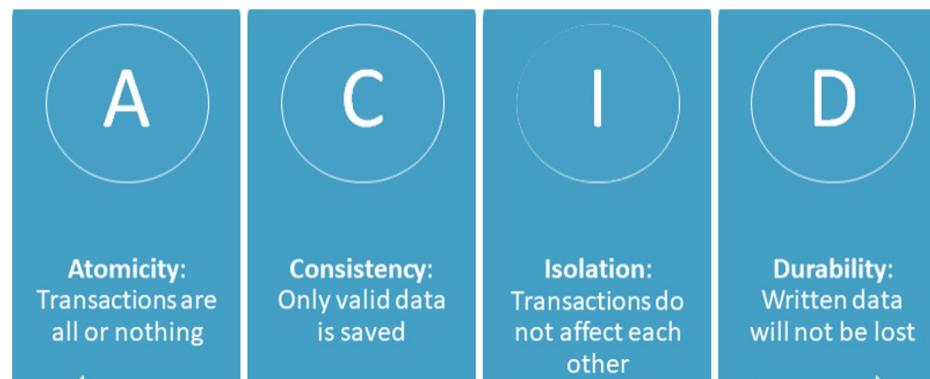
<http://nosql-database.org/>



Olga M. Moreno Martín

¿Qué son las bases de datos NoSQL?

Antes... Debemos tener presente que las aplicaciones que utilizan [bases de datos relacionales](#) se basan en transacciones ACID:



Estas características son incompatibles con la disponibilidad y el rendimiento necesarias en ciertas aplicaciones Web. Por ejemplo, si compramos un ítem en un portal online, se bloqueará una parte de la base de datos, en concreto el inventario, y cualquier otra persona en el mundo tendría que esperar hasta que se completase mi transacción.

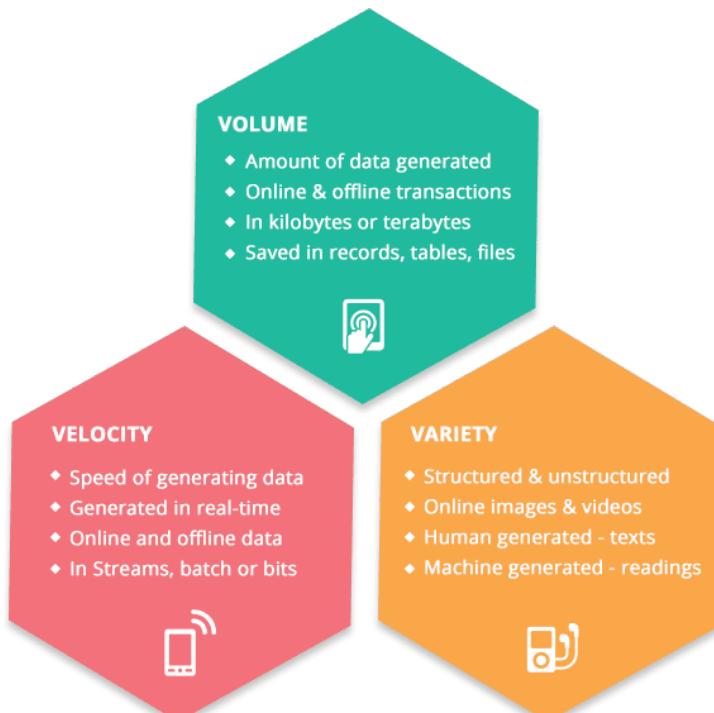
El portal web puede utilizar los datos almacenados en caché o incluso utilizar registros desbloqueados, resultando en inconsistencia. En un caso extremo, dos personas podrían comprar la última unidad de un ítem en la tienda.

Olga M. Moreno Martín

¿Qué son las bases de datos NoSQL?

Las bases de datos no relacionales son un grupo de tecnologías que surgen de la necesidad de tener sistemas de bases de datos **altamente disponibles, distribuidos y escalables horizontalmente.**

THE 3Vs OF BIG DATA



Tipos. Vista General.

- **Clave-Valor**

Cada elemento se almacena con un nombre de atributo (o clave) junto a su valor correspondiente.

- **Orientadas a Grafos**

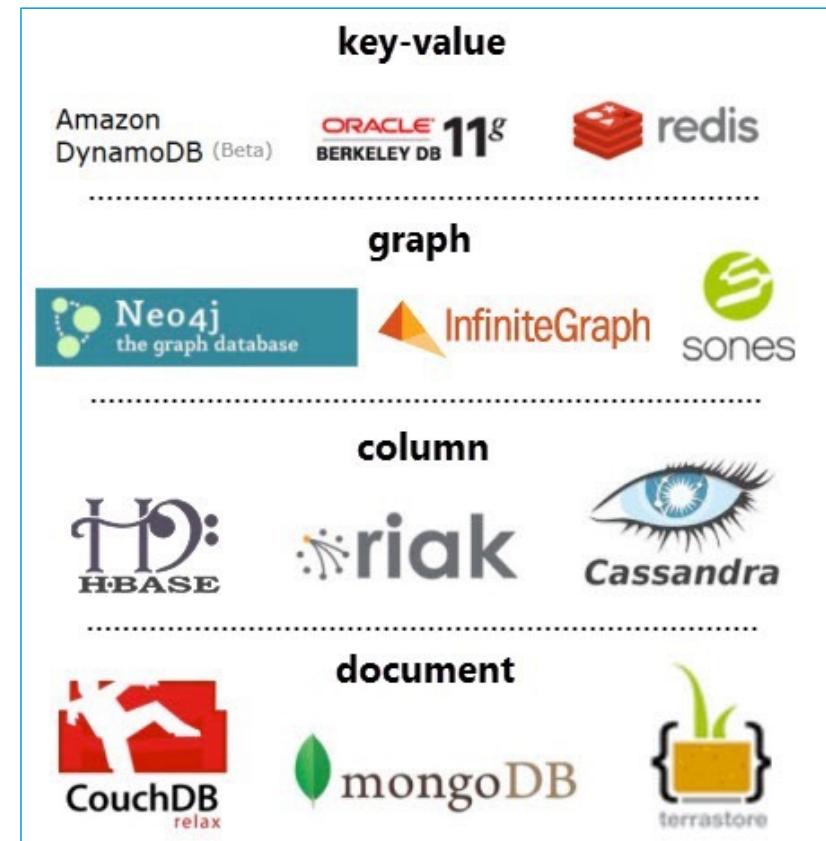
Almacenan información sobre redes, como pueden ser conexiones sociales, tuberías, tendido eléctrico,...

- **Columnares**

Los datos se almacenan como columnas, no como filas. Cada fila puede contener un número diferente de columnas.

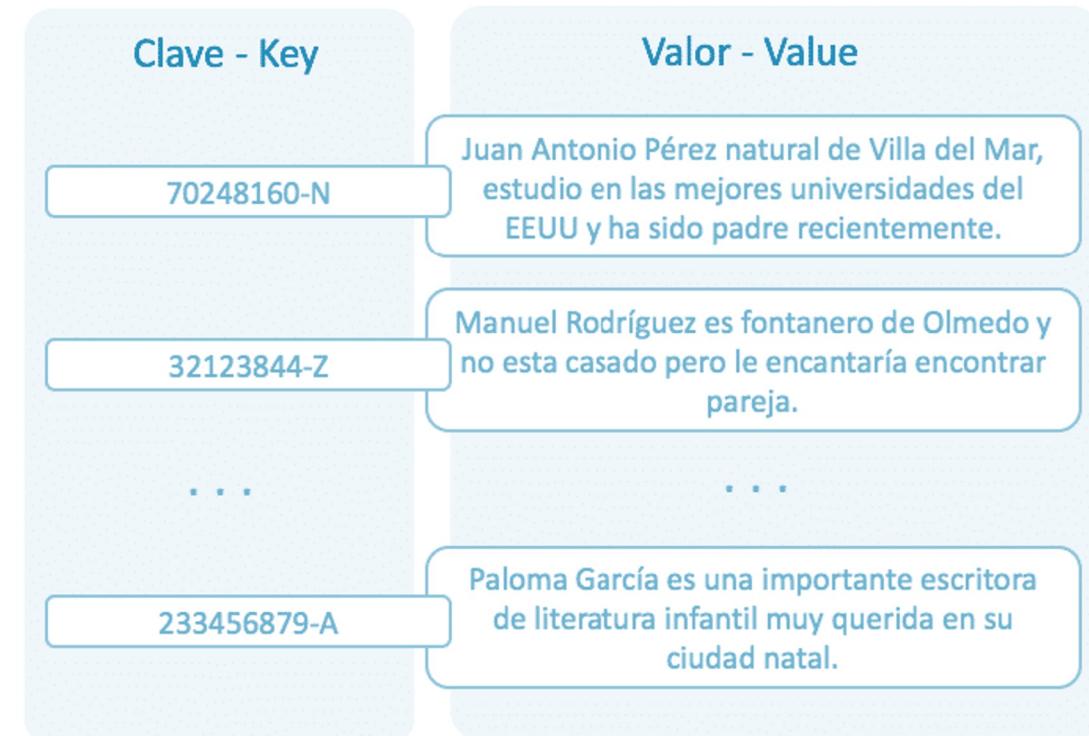
- **Documentales**

Cada clave almacena un documento con una estructura similar a JSON.



Tipos. Clave- Valor

Las bases de datos NoSQL clave-valor son un tipo de base de datos NoSQL que almacenan datos como pares clave-valor. Este modelo de datos es muy simple y ofrece una alta escalabilidad y rendimiento, lo que las hace ideales para aplicaciones que necesitan manejar grandes volúmenes de datos con tiempos de respuesta rápidos.



Tipos. Clave- Valor



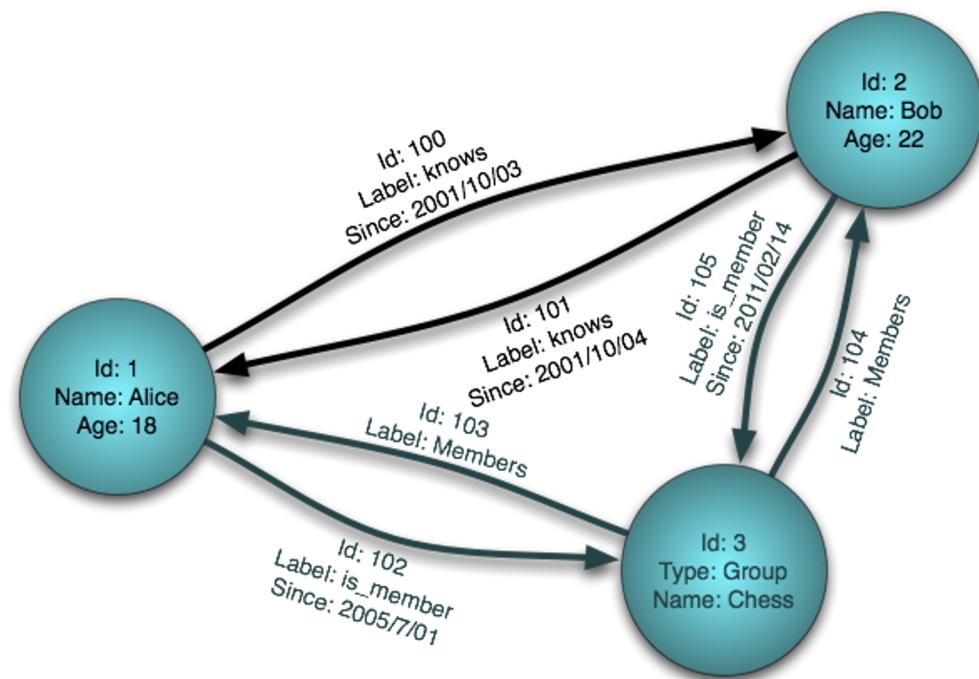
Ejemplos:

- [Redis](#): es una base de datos en memoria NoSQL clave-valor que admite una amplia variedad de estructuras de datos como listas, conjuntos y mapas. Redis se utiliza a menudo para caché de datos y aplicaciones en tiempo real debido a su alta velocidad de lectura y escritura.
- [Amazon DynamoDB](#): es una base de datos NoSQL clave-valor administrada en la nube que ofrece una alta escalabilidad y rendimiento. DynamoDB es compatible con documentos y admite una amplia variedad de estructuras de datos, incluidos números, cadenas, conjuntos y listas.
- [Riak](#): es una base de datos NoSQL clave-valor distribuida que se centra en la disponibilidad y la escalabilidad horizontal. Riak admite una amplia variedad de estructuras de datos, incluidos números, cadenas, mapas y conjuntos.
- [Oracle NoSQL Database](#): es una base de datos NoSQL clave-valor distribuida que admite una amplia variedad de estructuras de datos, incluidos documentos y listas. Oracle NoSQL Database está diseñado para ofrecer un rendimiento y una escalabilidad excepcionales en aplicaciones que necesitan procesar grandes volúmenes de datos en tiempo real.

Tipos. Orientadas a Grafos.

Las bases de datos NoSQL orientadas a grafos son un tipo de base de datos NoSQL que se basan en la teoría de grafos para almacenar y procesar datos. En este modelo de datos, los datos se representan como nodos y las relaciones entre los nodos se representan como aristas.

Las bases de datos NoSQL orientadas a grafos son útiles para aplicaciones que necesitan analizar relaciones complejas entre los datos, como redes sociales, análisis de datos de sensores, recomendaciones de productos y análisis de fraudes financieros.



Tipos. Orientadas a Grafos.

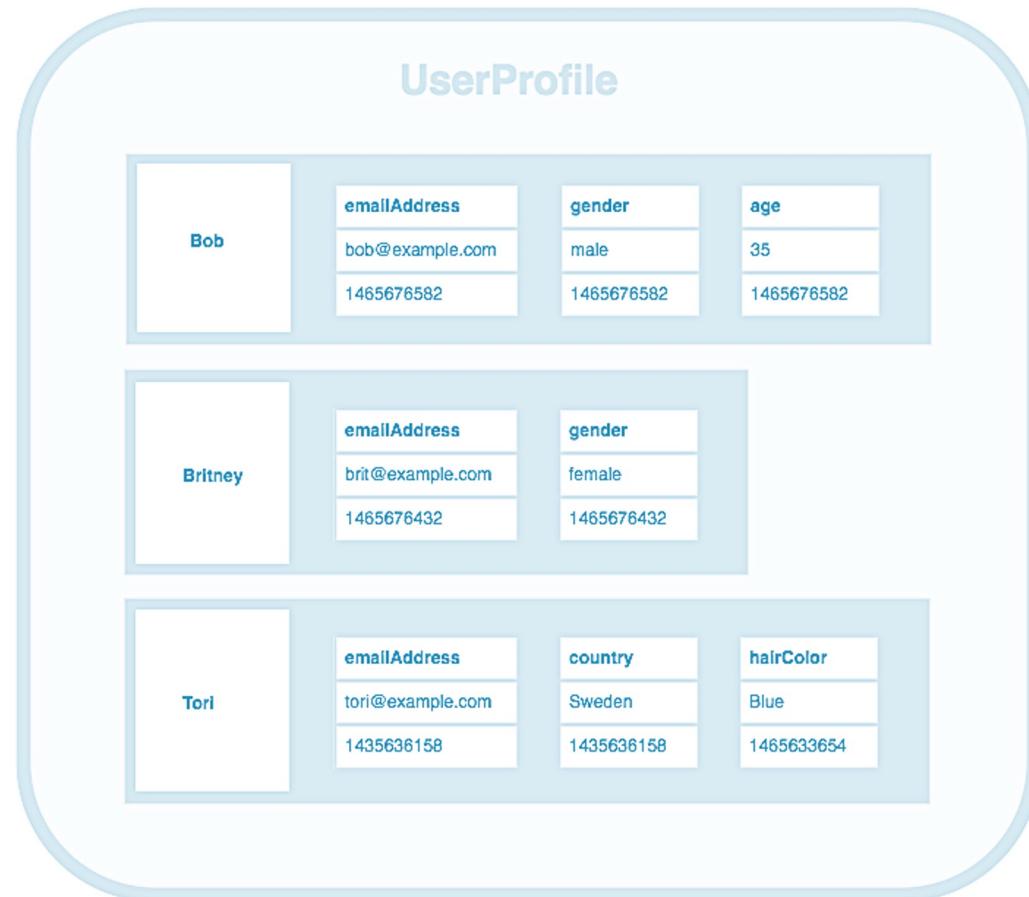


Ejemplos:

- [Neo4j](#): es una base de datos NoSQL orientada a grafos de código abierto que permite almacenar y procesar datos relacionales en forma de grafos. Neo4j es utilizado en una amplia variedad de aplicaciones, desde redes sociales y análisis de datos hasta análisis de fraudes y recomendaciones de productos.
- [Amazon Neptune](#): es una base de datos NoSQL orientada a grafos en la nube que está diseñada para almacenar y procesar grandes conjuntos de datos de grafos. Neptune es compatible con el lenguaje de consulta de grafos Gremlin y el lenguaje de consulta SPARQL.
- [OrientDB](#): es una base de datos NoSQL orientada a grafos de código abierto que admite múltiples modelos de datos, incluidos grafos, documentos y clave-valor. OrientDB es utilizado en aplicaciones que requieren análisis de datos complejos, como recomendaciones de productos y análisis de redes sociales.
- [ArangoDB](#): es una base de datos NoSQL de múltiples modelos que admite documentos, grafos y clave-valor. ArangoDB es utilizado en una amplia variedad de aplicaciones, desde análisis de redes sociales hasta análisis de datos de sensores.

Tipos. Columnares.

Las bases de datos NoSQL columnares son un tipo de base de datos NoSQL que almacenan datos en columnas, en lugar de en filas como lo hacen las bases de datos relacionales tradicionales. Estas bases de datos están diseñadas para manejar grandes volúmenes de datos y proporcionan una alta escalabilidad y disponibilidad.



Olga M. Moreno Martín

Tipos. Columnares.

Ejemplos:

- [Apache Cassandra](#): es una base de datos distribuida de alta disponibilidad que puede manejar grandes volúmenes de datos estructurados y no estructurados. Cassandra tiene una arquitectura descentralizada que le permite escalar horizontalmente y proporcionar alta disponibilidad y tolerancia a fallos.
- [HBase](#): es una base de datos NoSQL columnar basada en Hadoop que se ejecuta en el sistema de archivos Hadoop HDFS. HBase está diseñado para manejar grandes conjuntos de datos estructurados y no estructurados y proporciona alta disponibilidad y escalabilidad.
- [Amazon SimpleDB](#): es un servicio de base de datos NoSQL columnar en la nube que puede manejar grandes conjuntos de datos y proporciona una alta disponibilidad y escalabilidad. SimpleDB se integra bien con otros servicios de AWS y es una buena opción para aplicaciones web y móviles.
- [Google Bigtable](#): es una base de datos distribuida NoSQL columnar utilizada por Google para almacenar grandes cantidades de datos estructurados y no estructurados. Bigtable está diseñado para escalabilidad y alta disponibilidad, y es utilizado por muchas aplicaciones de Google, incluyendo Google Maps, Google Analytics y Google Finance.

Estas bases de datos NoSQL columnares son una buena opción para aplicaciones que necesitan manejar grandes volúmenes de datos y que necesitan escalar horizontalmente para manejar cargas de trabajo cada vez mayores.

Características de las BBDD NoSQL



Beneficios

- Soporte para grandes volúmenes de datos estructurados, semi-estructurados y sin estructurar.
- Permiten comenzar con un modelo sencillo y con el paso del tiempo, añadir nuevos campos, ya sean sencillos o anidados a datos ya existentes.
- Sprints ágiles, iteraciones rápidas y frecuentes commit/push de código.
- El modelo crece de la mano del código de aplicación
- Arquitectura eficiente y escalable diseñada para trabajar con clusters de ordenadores más económica transparente para el desarrollador.
- En NoSQL, se permite la inserción de datos [sin un esquema predefinido](#).
 - Facilita la modificación de la aplicación en tiempo real
 - No hay interrupciones de servicio
 - Desarrollo más rápido
 - Integración de código más robusto
 - Menos tiempo empleado en la administración de la base de datos.

Características de las BBDD NoSQL



Escalabilidad:

SQL : Auto-Sharding

- Dividen los datos entre un número arbitrario de servidores, sin que la aplicación sea consciente de la composición del pool de servidores.
- Los datos y las consultas se balancean entre los servidores.

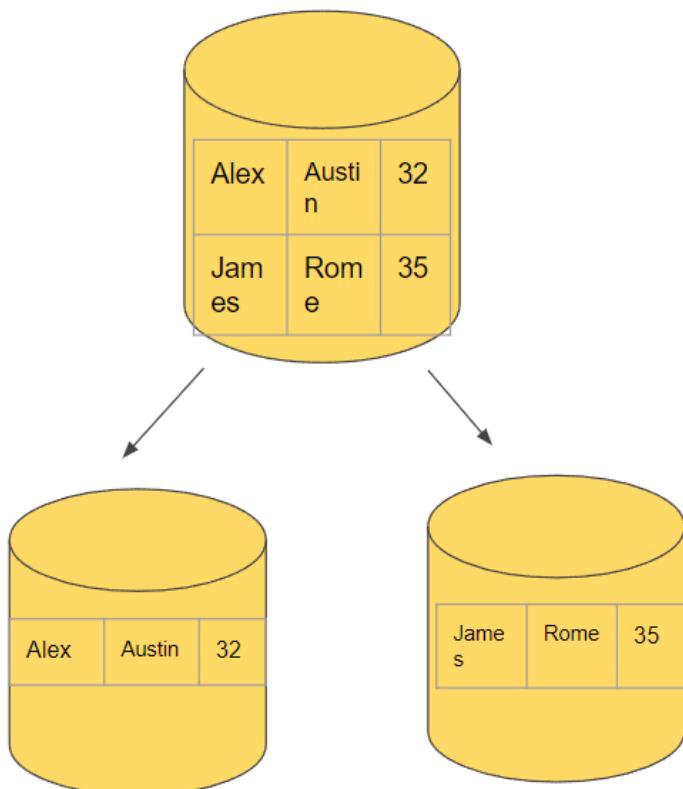
NoSQL: Replicación

Mantiene copias idénticas de los datos en múltiples servidores

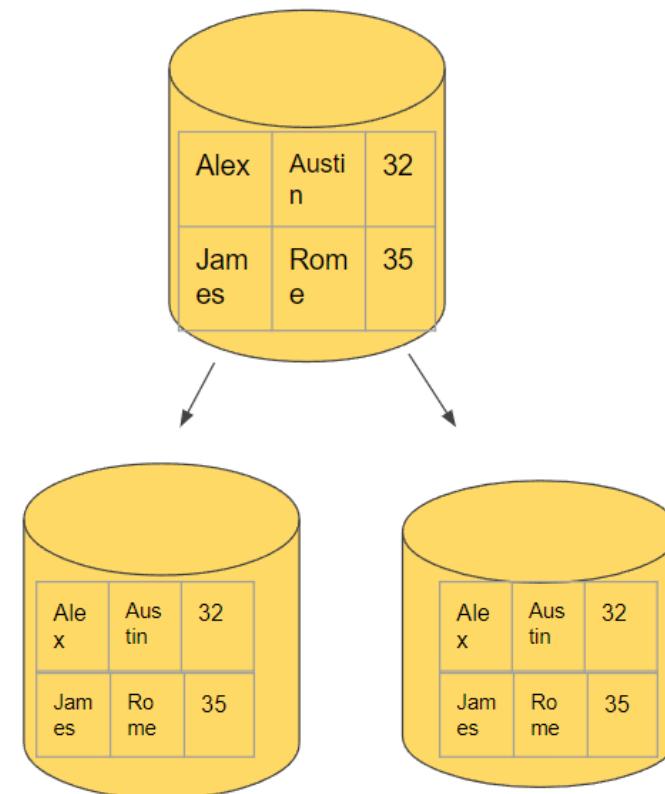
- Facilita aplicaciones robustas, incluso si alguno de los servidores sufre algún problema.
- La mayoría de sistemas NoSQL soportan la replicación automática
- Desde el punto de vista del desarrollador, el entorno de almacenamiento es virtual y ajeno al código de aplicación.

Características de las BBDD NoSQL

Sharding



Replication



Olga M. Moreno Martín

Características de NoSQL



Beneficios de la Replicación

- **Escalabilidad y rendimiento:** distribuye las consultas en diferentes nodos
- **Disponibilidad,** ofreciendo tolerancia a fallos de hardware o corrupción de la base de datos.
 - Al replicar los datos vamos a poder tener una copia de la base de datos, dar soporte a un servidor de datos agregados/informes, o tener nodos a modo de copias de seguridad que pueden tomar el control en caso de caída del nodo principal.
- **Aislamiento** (la i en ACID - isolation), entendido como la propiedad que define cuándo y cómo al realizar cambios en un nodo se propagan al resto de nodos.
 - Copias sincronizadas para separar procesos de la base de datos de producción
 - Informes o copias de seguridad en nodos secundarios → elimina el impacto negativo en el nodo principal
 - Sistema sencillo para separar el entorno de producción del de preproducción.

Las Bases de Datos Orientadas a Documentos



Las Bases de Datos Orientadas a Documentos o Documentales son el tipo de BD NoSQL más populares.

Productos de BD orientadas a documentos: [MongoDB](#), Couchbase y CouchDB.

Utilizan una aproximación Clave-Valor con diferencias importantes:

- Almacena los valores como documentos (entidades semiestructuradas de datos que se almacenan como strings o representaciones binarias de strings).
- Documentos típicamente en un formato estándar como JSON (JavaScript Object Notation) o XML (Extensible Markup Language).
- Documentos almacenan tanto estructura como contenido.
- En un único documento se almacenan todos los atributos de una entidad (en vez de almacenar cada atributo de una entidad con una clave separada).

Ejemplo: documento [JSON](#):

```
{ nombre: "Luis", apellido: "García", cargo: "Gerente", despacho: "2-120",  
teléfono: "555-222-3456", }
```

Bases de Datos Orientadas a Documentos



Estos tienen la particularidad de que no poseen un esquema definido sino que cada documento puede tener distintos datos, sin importar cantidad o tipo de datos.

- **Positivo:** si necesitas un campo más en un registro en específico no es necesario remodelar toda la tabla solo por un par.
- **Negativo:** la salida de datos será variable, por lo que si necesitas una salida más estructurada no es conveniente el uso de bases de datos NoSQL.

Al ser no relacionales no poseen relaciones, por lo que nos podemos olvidar de esos joins que hacen más lentas nuestras consultas. Si hay maneras de agregar relaciones en el caso de que sean necesarias pero la idea en este tipo de base de datos es tener las menos posibles.

- **Positivo:** Mejor rendimiento, consultas más rápidas. Suelen ser pocas colecciones y cada una tiene un propósito.
- **Negativo:** cuando editamos datos hay que hacerlo en todas las colecciones en las que se encuentran.

Las bases de datos NoSQL permiten el escalado horizontal de forma sencilla y poseen un mejor rendimiento a nivel de miles de consultas de lectura y escritura.

Bases de Datos Orientadas a Documentos



Las BD orientadas a Documentos no necesitan definir un esquema predefinido de añadir datos:

- Cuando añadimos un documento se crea la estructura de datos subyacente necesaria.
- La “falta de esquema” da a los desarrolladores más flexibilidad que con las BD relacionales.

Ejemplo:

{ nombre: "Luis", apellido: "García", cargo: "Gerente", despacho: "2- 120", teléfono: "555-222-3456", } }	{ nombre: "Roberto", apellido: "Rodríguez", cargo: "Asesor", despacho: "2-130", teléfono: "555-222-3478", f.alta: "1-Feb-2010", f.baja: "12-Ago-2014" } }
--	--

- No hay problema en que f.alta y f.baja no estén en el documento de “Luis”.

Los desarrolladores pueden añadir atributos si lo necesitan, pero: sus programas son responsables de gestionarlo bien.

Olga M. Moreno Martín

Bases de Datos Orientadas a Documentos



Un documento puede incluir a otro documento (documentos embebidos), listas de documentos/valores...

→ Esto elimina la necesidad de unir/combinar (join) documentos como hacemos con las tablas.

Ejemplo utilizando JSON:

```
{  
  "id_cliente":187693,  
  "nombre": "Ana Pérez",  
  "dirección":{  
    "calle": "C/ Mayor, 12",  
    "ciudad": "Móstoles",  
    "provincia": "Madrid",  
    "CP": "28933" },  
  "primer_pedido": "15/01/2013",  
  "ultimo_pedido": "27/06/2014"  
}
```

} Documento
Embebido

Bases de Datos Orientadas a Documentos



En resumen:

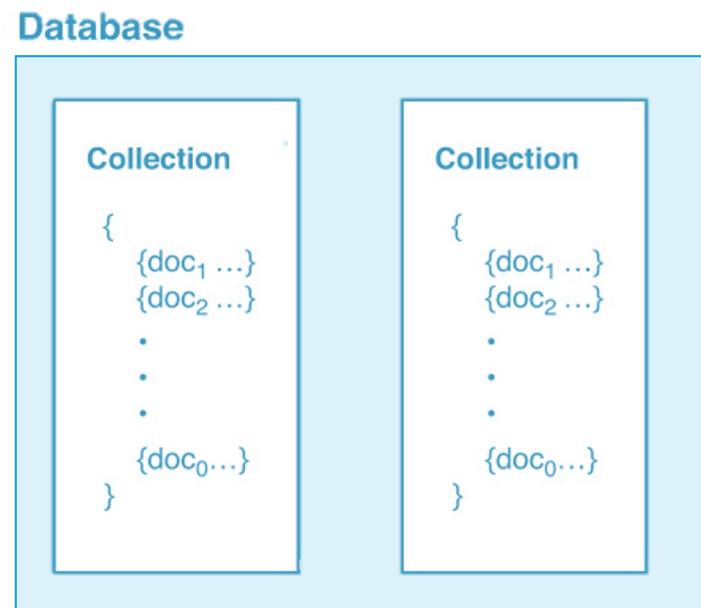
- Combinan la **flexibilidad** de las Bases de Datos NoSQL con la posibilidad de gestionar datos complejos.
- Los documentos contienen tanto información de la **estructura** como contenido (datos).

Un documento es un conjunto de pares clave-valor.

- Las claves en un par nombre-valor indican un atributo y se representan como cadenas de caracteres.
- Los valores en un par nombre-valor es el dato asignado al atributo y pueden ser tipos de datos básicos (números, cadenas o booleanos) o estructuras (arrays u objetos).
- JSON y XML son dos formatos utilizados habitualmente para definir documentos.
- Existe gran **libertad en la estructura** de los documentos.
 - No precisan definir la estructura de los datos a priori.
 - Son flexibles en cuanto a los atributos utilizados.
- Permiten **consultar y filtrar colecciones** de documentos:
 - Permiten consultas con múltiples atributos.

Bases de Datos Orientadas a Documentos

En una BD orientada a Documentos los documentos “similares” se agrupan en colecciones. Una BD puede tener un [conjunto de colecciones](#), que están compuestas a su vez por un [conjunto de documentos](#).



Bases de Datos Orientadas a Documentos.

Diseño.

En el diseño de una Base de Datos orientada a Documentos hay que tener en cuenta los siguientes aspectos:

- qué colecciones de documentos
- qué esquema, cuánta desnormalización
- cómo modelar las relaciones
- qué y cuántos índices utilizar

¿Cuándo hay que usar documentos embebidos y cuándo referencias a otros documentos?

- Los principios de diseño en Bases de Datos NoSQL hay que aplicarlos con flexibilidad.
- Siempre se deben considerar los beneficios y desventajas de un principio de diseño en una situación particular.

Bases de Datos Orientadas a Documentos. Colecciones.

Gestión de múltiples documentos en colecciones:

- Los documentos se agrupan en colecciones de documentos “similares”.
- Los documentos en una colección no tienen por qué tener estructuras idénticas, pero deberían compartir alguna estructura común.
- **Un aspecto clave es decidir cómo organizar los documentos en colecciones** (dado que no es obligatorio que estén relacionados).

¿Cómo diseñar la BD?



Entidades separadas para cada producto



Olga M. Moreno Martín



Una entidad para todos los productos indicando el tipo de producto



Bases de Datos Orientadas a Documentos



¿Cómo decidir [cómo organizar los datos](#) en una o más colecciones de documentos?

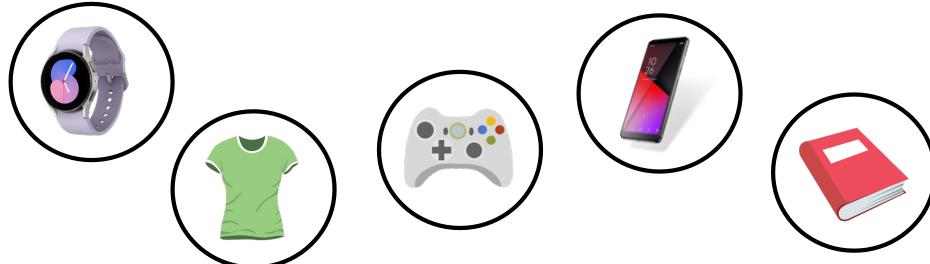
Podemos empezar con: [¿Cómo se utilizan los datos?](#) [¿Cuáles son las consultas más frecuentes?](#)

- ¿Cuál es el número medio de productos comprados por cliente?
- ¿Cuál es el rango (mínimo, máximo) de productos adquiridos por cliente?
- ¿Cuáles son los 20 productos más populares por cliente?
- ¿Cuál es el valor medio de las ventas por producto (precio estándar por cliente-coste del producto)?
- ¿Cuántos productos de cada tipo se vendieron en los últimos 30 días?

Todas las consultas utilizan datos de todos los tipos de productos, y únicamente la última necesita subtotales por tipo de producto. Este es un buen indicador de que los productos deberían estar en una colección de documentos única.

Bases de Datos Orientadas a Documentos

¿Cómo diseñar la BD?



Entidades separadas para cada producto

Una entidad para todos los productos indicando el tipo de producto



En las BD NoSQL en vez de empezar por los datos, tratando de organizarlos, podemos empezar por las consultas (aplicaciones) para entender cómo se usarán los datos.

Otro motivo en favor de una colección de documentos única es que **probablemente haya nuevos tipos de productos**. Si crece hasta decenas o centenares de tipos, el **número de colecciones** será poco manejable.

Bases de Datos Orientadas a Documentos.

Esquema.

Schemaless:

Cuando trabajamos con BD relacionales definimos un esquema (una especificación que describe la estructura de un objeto, en este caso una tabla).

```
CREATE TABLE customer ( customer_ID integer, name varchar(100), street  
varchar(100), city varchar(100), state varchar(2), zip varchar(5),  
first_purchase_date date, last_purchase_date date);
```

En este esquema todos los clientes tienen los mismos datos.

En una BD Relacional se diseñan las tablas antes de que los desarrolladores del software puedan interactuar con las tablas de la BD.

En las BD orientadas a Documentos **no se requiere este paso**, ya que los desarrolladores pueden crear colecciones y documentos insertándolas directamente en la BD (sin especificación previa del esquema).

Bases de Datos Orientadas a Documentos. Esquema.

Schemaless:

- No existe la necesidad de elaborar un modelo que contenga todos los posibles campos.
- Sin embargo, existe una **organización implícita** en los documentos que se insertan. Esta organización está explícita en el código que manipula los documentos.
- Los documentos (a diferencia de las tuplas) son **polimórficos**: no es obligatorio que tengan la misma estructura de datos. Podemos añadir un dato diferente a un documento, sin que el resto de documentos tengan que tener dicho dato.
- Las Bases de Datos orientadas a Documentos tienen un **esquema polimórfico**, es decir, que pueden tener al mismo tiempo “varios esquemas”.

Bases de Datos Orientadas a Documentos. Documentos Embebidos.

Documentos embebidos (embedding):

Esta decisión implica la **desnormalización** de los datos, almacenando dos documentos relacionados en un único documento (en MongoDB mismo `_id`).

- ¿Cuándo? En general:
 - Relaciones del tipo agregación (“o contiene”) entre entidades.
 - Relaciones Uno a Uno.
 - Relaciones 1:N (Uno a Pocos) donde la parte de muchos siempre aparecen o se consultan desde el contexto del documento padre o de nivel superior.
- Las **operaciones** a este documento son mucho **menos costosas** para el servidor que las operaciones que involucran múltiples documentos.
- **BENEFICIO:** Recuperación más rápida de información.
- **PERO:** No podemos acceder al documento embebido si no es a través del documento padre.

Bases de Datos Orientadas a Documentos. Modelado de Relaciones.

Resumen:

Modelado de relaciones habituales en BD orientadas a Documentos:

- Uno a muchos
- Muchos a muchos
- Jerarquías

Embeber documentos vs referencias

- Tamaño de N
- Acceso a los datos

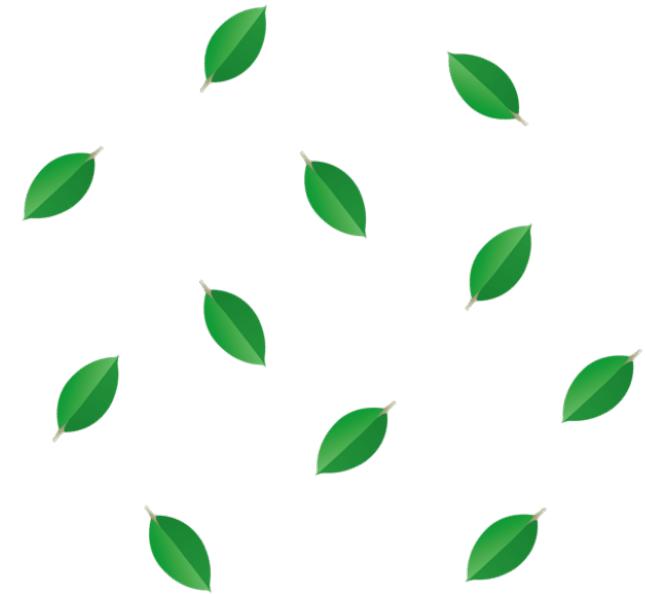
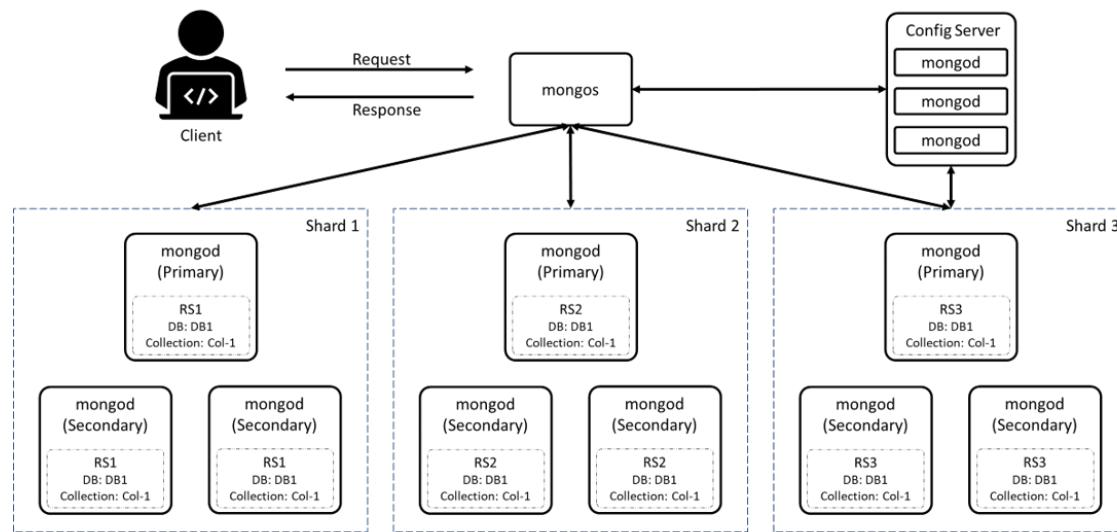
Bases de Datos Orientadas a Documentos. Modelado de Relaciones.

Al diseñar una BD hay que intentar identificar el **número ideal de índices**:

- Excesivos: reducirá el rendimiento de las escrituras
- Pocos: reducirán el rendimiento de lecturas

Aspectos a tener en cuenta:

- Ratio lectura/escritura (BD para análisis...)
- Campos que se utilizarán en las búsquedas
- BD con **fuerte carga de lecturas** suelen tener muchos índices, especialmente si no hay patrones claros de consultas.
- BD con **fuerte carga de escrituras** (p.e. sensor): minimizar número de índices (identificadores), equilibrio.
 - Si aún así es importante la latencia de consultas, puede plantearse la creación de una segunda BD con datos agregados para lecturas (DW)




mongoDB

Introducción a MongoDB

¿Qué es MongoDB?

Mongo proviene de la palabra humongous (enorme o gigantesco en inglés)

Base de Datos NoSQL (Almacenamiento de Datos No-Relacionales) orientada a documentos.

```
{ nombre : "Ana", edad : "30", ciudad:"Móstoles", provincia :"Madrid", telf:["916678989", "678998877"] }
```

Características

- BD NoSQL más popular del mercado (<http://db-engines.com/en/ranking>)
- Almacenamiento de documentos JSON (almacenados en BSON)
- Estructuras más cercanas a los programas
- Implementación fácil
- Rápida (“eliminación de necesidad de joins”)
- Robusta y escalable
- Esquemas dinámicos / flexibles (schemaless): dos documentos no tienen porqué tener el mismo esquema. No es necesario definirlo antes de añadir datos.
 - Al añadir un documento se crea la estructura de datos subyacente necesaria

Introducción a MongoDB

Ejemplo de Documento en MongoDB

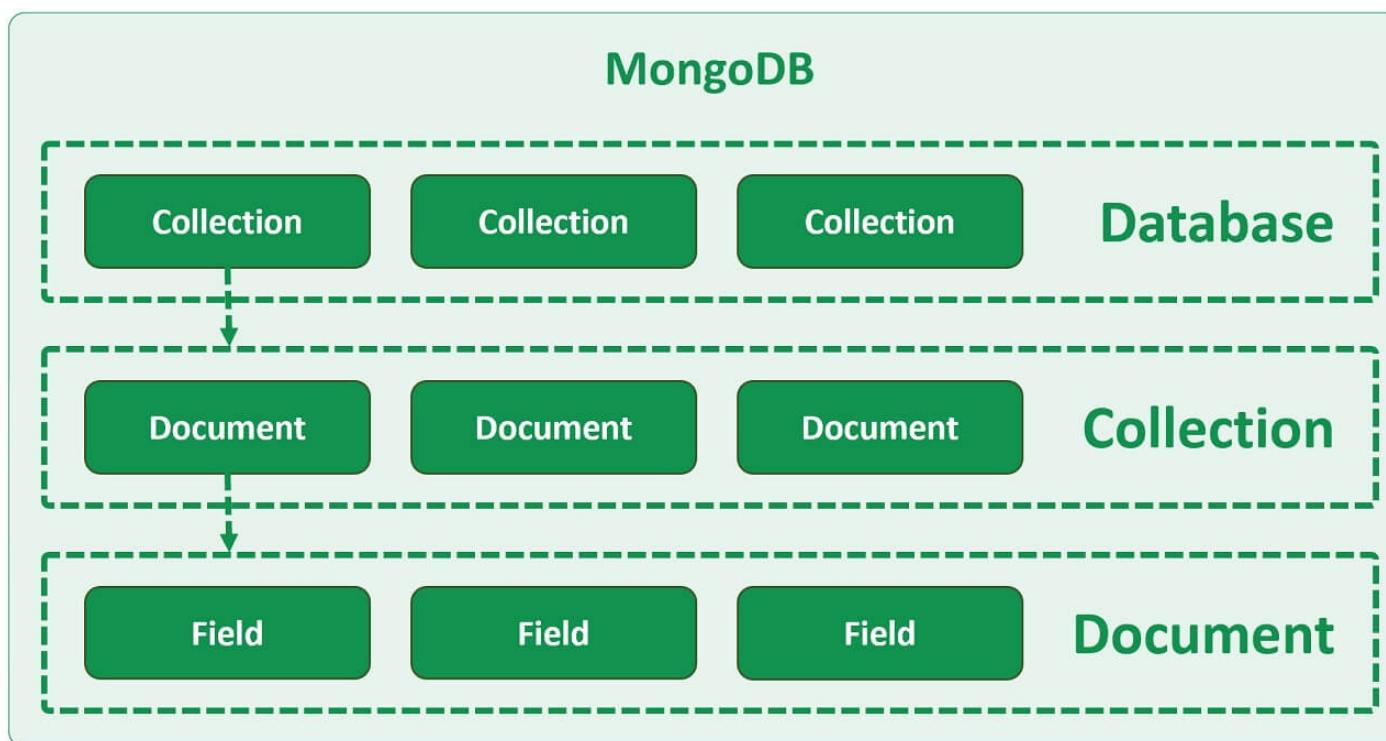


```
1  {
2    "_id": "5cf0029caff5056591b0ce7d",
3    "firstname": 'Jane',
4    "lastname": 'Wu',
5    "address": {
6      "street": '1 Circle Rd',
7      "city": 'Los Angeles',
8      "state": 'CA',
9      "zip": '90404'
10   }
11 }
```

The diagram illustrates the structure of a MongoDB document. It features a central rectangular box containing the JSON code for a single document. To the right of the box are three circular icons: one with a gear and checkmark, one with a globe, and one with a lightbulb and brackets {}, representing schema validation, global reach, and querying respectively. Below the box is another circular icon with a magnifying glass and brackets {}, representing search functionality.

Olga M. Moreno Martín

Introducción a MongoDB



Comparativa MongoDB - MySQL



Caracteristica	MongoDB	MySQL
Cloud, SaaS, Web	si	si
Desarrolladores	MongoDB Inc.	Oracle Corporation
SO	Multiplataforma	Multiplataforma
Lenguaje query	Javascript	SQL
Mapa reducido	si	no
Convercion de DB	si	no
Analisis de performance	si	no
Virtualización	si	no
Modelo de integridad	BASE	ACID
Atomicidad	condicional	si
Aislamiento	no	si
Transacciones	no	si
Integridad referencial	no	si
CAP	CP	CA
Escalabilidad horizontal	si	condicional
Modo de replicación	Maestro-Esclavo	Maestro - Maestro/Esclavo

MongoDB y Las Bases de Datos NoSQL



En NoSQL no existen las transacciones. Aunque nuestra aplicación puede utilizar alguna técnica para simular las transacciones, aunque en el caso de MongoDB no tiene esta capacidad. Solo garantiza operaciones atómicas a nivel de documento. Si las transacciones indispensables en nuestro desarrollo, deberemos pensar en otro sistema.

Tampoco existen los JOINS. Para consultar datos relacionados en dos o más colecciones, tenemos que hacer más de una consulta. En general, si nuestros datos pueden ser estructurados en tablas, y necesitamos las relaciones, es mejor que optemos por un RDBMS clásico.

Consultas de agregación. MongoDB tiene un framework para realizar consultas de este tipo llamado Aggregation Framework. También puede usar MapReduce. Aún así, estos métodos no llegan a la potencia de un sistema relacional. Si vamos a necesitar explotar informes complejos, deberemos pensar en utilizar otro sistema.

Configuración del entorno de Trabajo



Aunque la instalación de MongoDB Community Edition 6.0.5 (current) es open source, no es el mejor de los recursos para trabajar con Mongo.

La propia compañía nos ofrece varias alternativas, una muy interesante es MongoDB Atlas, que nos proporciona de forma gratuita una instancia EC2 de AWS para trabajar con el motor.

<https://www.mongodb.com/es/cloud/atlas/efficiency>

Pero nosotros conocemos Docker y vamos a trabajar con un contenedor local.

- Instalación de herramienta gráfica MongoDB Compass Community Edition

<https://www.mongodb.com/products/compass>

- Instalación de Editor de MongoDB: Studio 3T (antes Robo 3T) es una Free – Trial donde luego nos da las funciones principales gratis. <https://studio3t.com/>
- Instalación de Mongo Express: Interfaz gráfica desde el navegador. Podemos utilizarla con el contenedor Docker es gratuita y muy cómoda.

Configuración del entorno de Trabajo

Para crear un contenedor con mongo solo necesitamos crear un archivo ejecutable con Docker Compose como el que aparece a continuación:

docker-compose.yml

```
version: '3.1'

services:

mongo:
  image: mongo
  restart: always
  environment:
    MONGO_INITDB_ROOT_USERNAME: root
    MONGO_INITDB_ROOT_PASSWORD: example
  ports:
    - 27017:27017
```

Si queremos instalar express también podemos agregar el siguiente código al fichero:

```
mongo-express:
  image: mongo-express
  restart: always
  ports:
    - 8081:8081
  environment:
    ME_CONFIG_MONGODB_ADMINUSERNAME: root
    ME_CONFIG_MONGODB_ADMINPASSWORD: example
    ME_CONFIG_MONGODB_URL: mongodb://root@example:mongo:27017/
```

Adjunto a la Unidad Didáctica



Configuración del entorno de Trabajo

Si todo transcurre con normalidad veremos los contenedores corriendo:

		NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
Extensions	BETA	mongo	-	Running (2/2)	-		
		2 containers					
		mongo-express-1	mongo-express:latest	Running	8081	23 days ago	
		508b3383eaa6					
		mongo-1	mongo:latest	Running	27017	23 days ago	
		6a760c477d37					

Y en el puerto 8081,
podremos operar con
Mongo Express:



The screenshot shows the Mongo Express interface. At the top, there's a header with a green info icon and the text "Mongo Express Database". Below this is a table titled "Databases" with three entries:

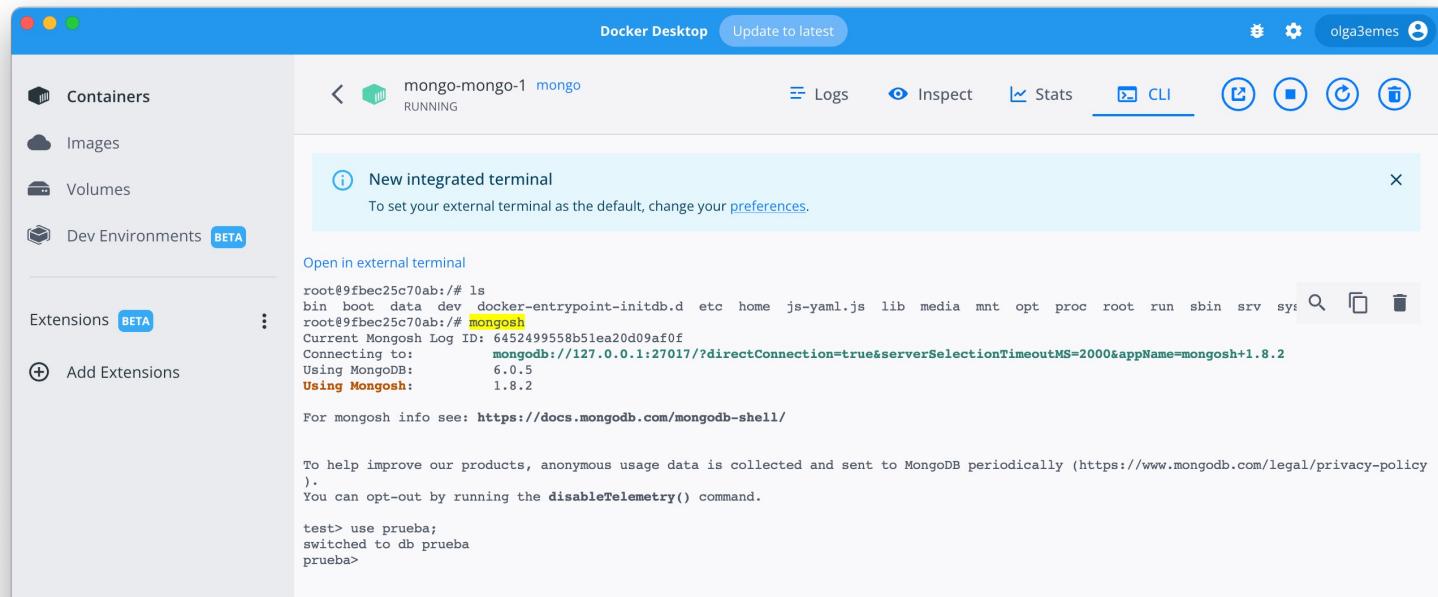
	Databases	
	admin	
	config	
	local	

Server Status

Turn on admin in config.js to view server stats!

Consola de MongoDB

Para utilizar MongoDB desde consola podemos acceder al CLI de Docker:



Al escribir: **mongosh** podremos acceder a todas las funciones del motor.

- A partir de la versión 6.0 MongoDB usa este comando, pero en versiones anteriores habría que escribir *mongo*.

Consola de MongoDB. Comandos iniciales.

Por defecto, nos conectamos a la BD prueba (aún no existirá)

Para cambiar de BD o crear una nueva:

- `use BDNueva`

Identificador tiene longitud máxima de 64 caracteres

Para saber a qué BD se está conectado:

- `db`

Para listar las BD existentes

- `show databases`

Para borrar una BD (a la que estemos conectados)

- `db.dropDatabase()`

Para ver la versión de MongoDB que tenemos instalada

- `db.version()`

Bibliografía



- MogoDB: Notes for Professional . GoalKiker – Free Programming Books
- Dan Sullivan (2015). NoSQL for Mere Mortals. Addison-Wesley Professional
- Dan McCreary & Ann Kelly (2014). Making Sense of NoSQL. A guide for managers and the rest of us. Manning Publications
- Gaurav Vaish (2013). Getting Started with NoSQL. Your guide to the world and technology of NoSQL. PACKT Publishing
- Joe Celko (2014). Joe Celko's Complete guide to NoSQL. What every SQL professional needs to know about nonrelational databases. Morgan Kauffmann

¿Preguntas?

Olga M. Moreno Martín

