Bases de Datos NoSQL

¿Qué es NoSQL?

NoSQL (o Not Only SQL) es un término que engloba una amplia variedad de tecnologías relacionadas con bases de datos que han sido desarrolladas en respuesta a la necesidad de almacenamiento del gran volúmen de datos generados por usuarios, objetos y productos, la frecuencia con la que dichos datos son accedidos y los requerimientos de rendimiento y procesamiento. Se presentan en contraposición a las bases de datos tradicionales que no fueron diseñadas para enfrentarse a los retos de escalabilidad y agilidad que presentan las aplicaciones modernas, ni tampoco para aprovechar las ventajas ofrecidas por sistemas de almacenamiento económico y potencia de procesamiento disponibles actualmente.

Las mayorías de las empresas utilizan sistemas desarrollados sobre bases de datos relacionales. Las bases de datos relacionales se han ganado esta posición predominante por buenas razones, no han dejado de ser útiles. No obstante, por motivos técnicos, como puede ser la necesidad de escalado rápido en las capacidades de sus sistemas de procesamiento y almacenamiento, por motivaciones económicas, consistentes en buscar alternativas económicas viables, no sólo en lo que a licencias se refiere, sino también en lo relacionado con almacenamiento y procesamiento, o bien debido a la necesidad de rápida adaptación a los requisitos del mercado en forma de desarrollos ágiles que requieren otro tipo de metodologías, se está tendiendo al empleo de técnicas NoSQL que permitan lidiar con los datos.

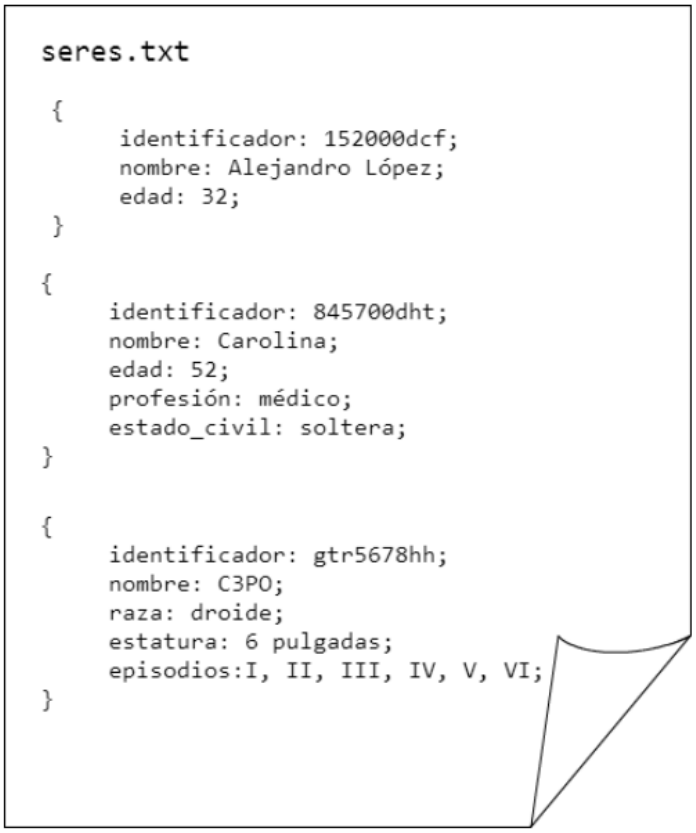
Por regla general, las bases de datos NoSQL se están convirtiendo en la primera alternativa a las bases de datos relacionales, ya que cumplen requisitos cada vez más necesarios como son escalabilidad, disponibilidad y tolerancia ante fallos. Las principales características de esta tecnología son:

Modelo de datos "schemaless"

Se trata de un modelo de almacenamiento de datos que permite almacenar cualquier dato que se desee, normalmente en pares clave-valor, sin necesidad de conocer las claves o los tipos. Puesto que la recuperación de la información ahí contenida dependerá de las aplicaciones de gestión desarrolladas explícitamente para ello, se suele considerar que el esquema presente en las bases de datos relacionales ha sido desplazado a las aplicaciones que corren sobre ellas. En cualquier caso, lo que cumplen este tipo de modelos es que no requieren esquema, no que necesariamente carezcan de él.

Supóngase el ejemplo de una base de datos que almacenase los datos en un archivo de texto con la sintaxis clave-valor. Además, dichos elementos son propiedades de un determinado objeto mayor, que será unívocamente determinado por un identificador. Así, el archivo seres.txt podría contener algo como lo siguiente:

En la ilustración a la derecha se observa un ejemplo de base de datos schemaless. Todos los objetos que forman parte del conjunto seres tienen en común un identificador, que será el que permita acceder a ellos unívocamente. Sin embargo, el resto de pares clave-valor, a pesar de que en algunos casos coincida la clave, no tiene porqué seguir un esquema determinado. Por ejemplo la clave nombre contiene nombre y apellido en el caso del ser 152000dcf, mientras que en el ser 845700dht únicamente contiene el nombre. Dicha clave, además, puede contener números, como es el caso del ser gtr5678hh. También difieren las claves definidas en cada ser, por ejemplo, la clave episodios sólo tiene sentido si el ser es un personaje de la Guerra de las Galaxias, pero no en el resto de casos. En este tipo de modelos no es necesario rellenar claves que carecen de sentido para determinados tipos de objeto.

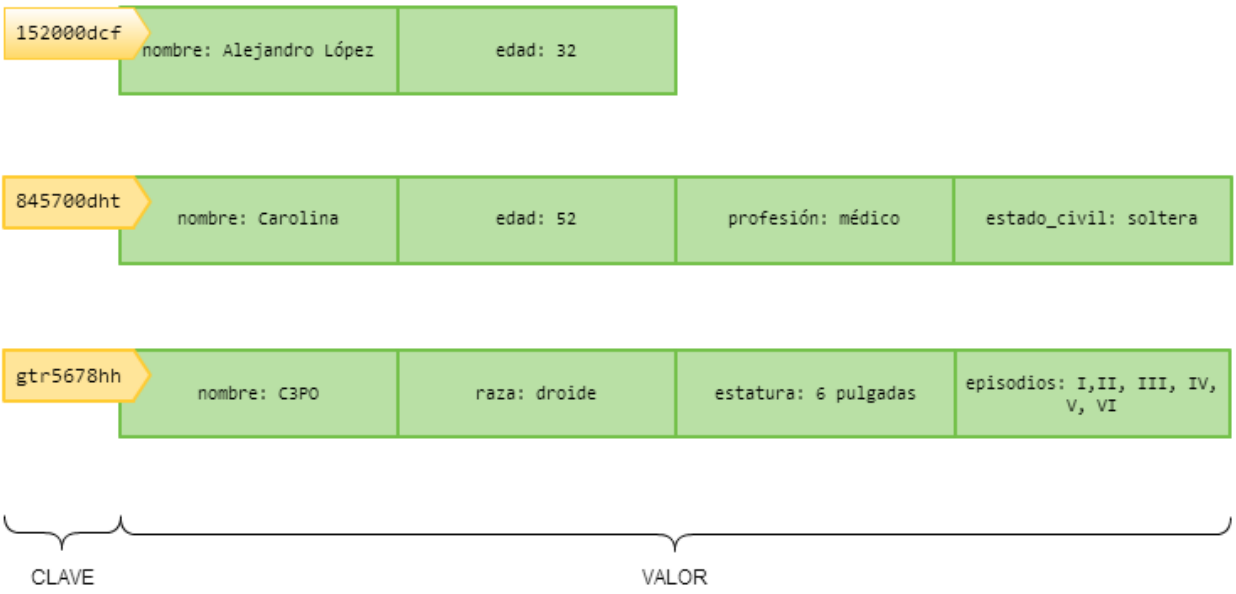


Tipos de BD NoSQL

En los últimos años se han desarrollado numerosas bases de datos que responden al esquema NoSQL. Cada una tiene una serie de características que las hacen más adecuadas para una determinada tarea. En esta sección se detallan los diferentes tipos existentes, qué características tienen y algunos ejemplos de cada una de ellas.

Clave-valor

Este tipo de base de datos provee modelo más simple de modelado consistente en una clave y un valor asociado a dicha clave. Las consultas de búsqueda sobre los datos así almacenados se deben hacer por medio de su clave, los valores aparecen opacos frente al motor de búsqueda. Si se compara con el modelo relacional, cada clave se corresponde con una columna, pero en este caso sólo existen registros independientes, no hay estructura. En la siguiente figura se muestra un ejemplo de modelado del conjunto de datos seres. En este caso, cada uno de los seres sólo podría ser accedido mediante su clave y el motor de la base de datos no entendería qué significa cada valor, así no sería posible, por ejemplo, obtener los nombres o las edades de los seres sin recuperar el registro completo.



La ventaja que presentan estas bases de datos es que son muy rápidas en lo que a lectura/escritura se refiere, únicamente supone un acceso a disco.

Por otra parte, existen tipos de bases de datos clave-valor diferenciadas por la forma en la que almacenan los datos y mantienen su consistencia, en particular:

- ***Almacenamiento en memoria***: Se trata de bases de datos que almacenan los registros en memoria principal, no en disco y son, por tanto, muy rápidas ya que se elimina el tiempo de latencia de búsqueda en disco. En esta categoría se incluyen memcached, Repcached, Oracle Coherence, Infinispan, Websphere eXtreme scale, JBoss cache y Terracotta Ehcache, entre otras.
- ***Almacenamiento clave-valor regular***: En este caso los registros sí son almacenados en disco. Algunos ejemplos son: Amazon SimpleDB, Flare, Schema-free, RAMCLoud, Twisted Storage, Redis, Tokyo Cabinet, Lightcloud, NMDB y Lux IO entre otros.
- ***Almacenamiento clave-valor eventualmente consistente***: Un sistema de base de datos eventualmente consistente es aquel que cumple el Teorema de CAP. Una definición informal sobre su significado es la proporcionada por Doug Terry et. al. Algunos ejemplos de bases de datos NoSQL eventualmente consistentes son Amazon Dynamo, Voldemort y Dynomite, entre otras.

Orientadas a documento

Las bases de datos orientadas a documento son una extensión del modelo anterior. Los datos se almacenan en un formato estructurado más complejo, denominado documento, cuyos elementos son entendidos por el software que compone la base de datos. Debido a ello no existe una limitación tan estricta sobre la forma de búsqueda, en estos casos es posible analizar los datos almacenados haciendo queries sobre más campos, no únicamente la clave.

Gracias a esta característica se trata de bases de datos adecuadas para aplicaciones orientadas a contenido (Facebook, blogs, etc.).

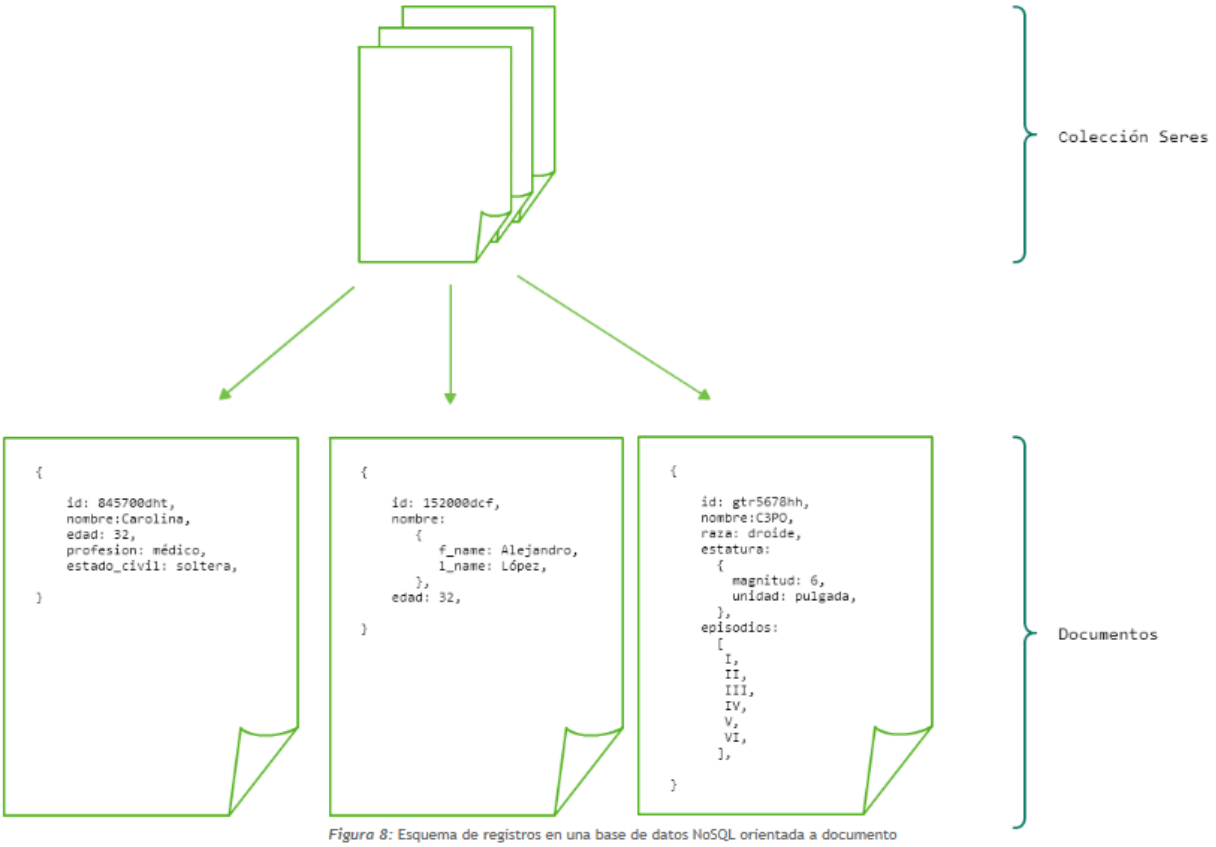


Figura 8: Esquema de registros en una base de datos NoSQL orientada a documento

El elemento central de este tipo de bases de datos es el documento. Se trata de una estructura, más o menos compleja, que encapsula información siguiendo un determinado formato estandarizado. No obstante, este estándar puede variar notablemente entre implementaciones. Cada documento se almacena en disco codificado siguiendo un formato que, de nuevo, depende de la implementación. Los más comunes son JSON/BSON, XML y YAML.

A su vez, los documentos se suelen agrupar en colecciones. Se trata de estructuras que aglutinan documentos relacionados en concepto y que, generalmente, quedan unívocamente determinados por un ID que puede ser generado automáticamente por la base de datos. En el ejemplo anterior la colección podría ser seres y la información de cada ser de la colección aparece codificada en documentos. Debido al hecho de que en este tipo de bases de datos la información es transparente para el motor de búsqueda, sí sería posible buscar seres por nombre, edad, estatura, etc., además de por ID.

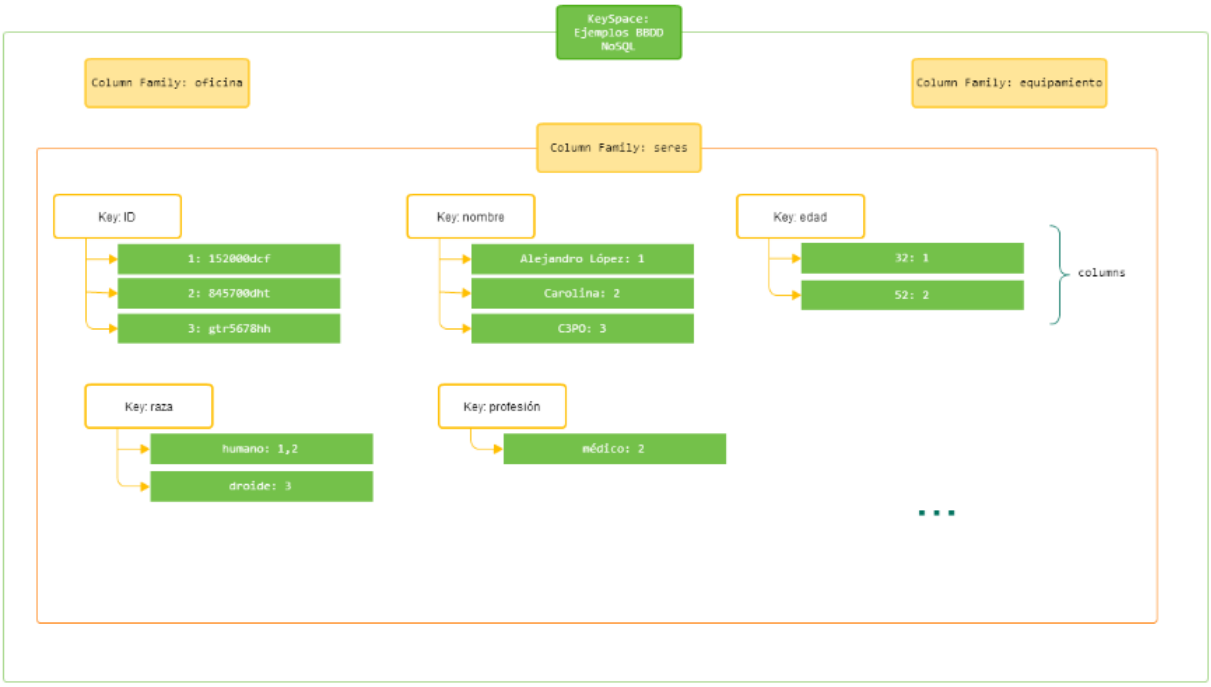
La generación automática de IDs de nuevo depende de la estrategia de distribución de los datos que siga la base de datos. Normalmente son los IDs los que determinan en qué nodo de la red distribuida se

almacenará un determinado documento, de manera que la correcta generación de dichos IDs es fundamental para asegurar, en la medida de lo posible, la distribución uniforme de la información.

Algunos ejemplos de bases de datos orientadas a documento son CouchDB, MongoDB, Apache JackRabbit, ThruDB, CloudKit, Lotus Domino y Terrastore, entre otras.

Orientadas a columna

Las bases de datos relacionales almacenan la información en forma de filas, es decir, cada nuevo registro insertado en la misma es una nueva fila y las búsquedas se realizan, en consecuencia, por filas. Sin embargo, en este tipo de bases de datos los registros se relacionan con las columnas (aunque no necesariamente en una relación 1-1), y las búsquedas se realizan por columnas. Esta diferencia reside en la forma de almacenar los registros en disco: Mientras que en el caso relacional dos registros contiguos se corresponden con filas, en el modelo NoSQL orientado a columna dos registros continuos se refieren a la misma columna.



A su vez, estas columnas, también denominadas tuplas, se agrupan dentro de estructuras superiores en la jerarquía que permiten identificarlas. En particular:

- ColumnFamily: Es la estructura inmediatamente superior a la columna y su cometido es agruparlas de forma que se identifique qué grupo de columnas contiene.
- Key: Es el identificador del registro, a su vez puede contener diferente número de columnas.
- Keyspace: Se trata del nivel superior en la jerarquía, identifica el conjunto total de datos, puede ser, por ejemplo, el nombre de la aplicación cuyos datos se estén almacenando.

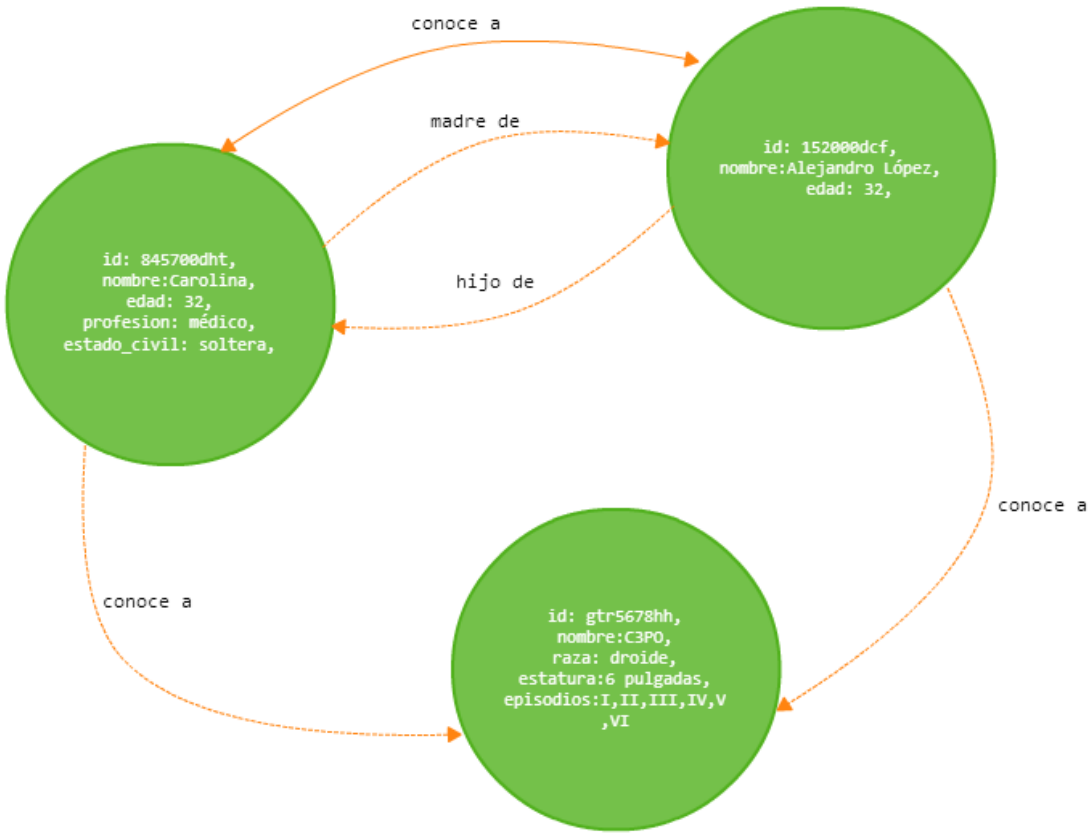
En la figura anterior se muestra el ejemplo de los datos de seres almacenados en una base de datos orientada a columna. Se ha añadido la raza a los registros 152000dcf y 845700dht para apreciar la potencia de este tipo de estructura. Mientras que en otro tipo de base de datos el campo humano se almacenaría repetidamente, en este caso sólo se guardan referencias a los IDs que se corresponden con esa raza. Además, de esta forma, se agilizan las tareas de análisis que implican el recorrido de todos los registros, puesto que se almacenan contiguos y se relacionan por índices de acceso rápido.

Algunos ejemplos de bases de datos orientadas a columna son Google BigTable, HBase, Cassandra e HyperTable, entre otras.

Orientadas a grafo

Las bases de datos orientadas a grafos son aquellas que emplean estructuras de tipo grafo para almacenar la información. Los elementos que la componen son nodos, propiedades y enlaces:

- Nodos: Son los elementos centrales del grafo, cada uno de ellos se corresponde con un objeto determinado, por ejemplo un usuario, una Web, una empresa, etc.
- Propiedades: Son atributos que caracterizan permanentemente a los nodos.
- Enlace: Los enlaces definen las relaciones entre nodos, dos nodos pueden o no estar conectados por un enlace, de forma bidireccional o no y con unos determinados atributos.



Generalmente se emplean para almacenar información sobre redes o relaciones sociales ya que en este tipo de bases de datos los elementos cambiantes son las conexiones entre los nodos del grafo, tanto su forma como sus atributos. Gracias a su estructura es posible utilizar teoría de grafos para realizar búsquedas sobre las mismas, esto las convierte en bases de datos muy adecuadas para almacenar datos en los que la información requerida dependa de los enlaces que conexionan los nodos.

El ejemplo de seres no sería muy adecuado para almacenarlo en bases de datos de este tipo, ya que constan más de características permanentes que de relaciones. En cualquier caso, por motivos de continuidad, en la Figura anterior se representa una idea de cómo podrían almacenarse los datos de seres en una base de datos orientada a grafo.

Algunos ejemplos de bases de datos orientadas a grafo son Neo4J e HyperGraphDB, entre otras.

En la Web NoSQL Databases, es posible encontrar un listado actualizado con las bases de datos NoSQL disponibles en la actualidad clasificadas por categorías.

Ejemplos de uso BD NoSQL

Como se ha visto, no existe un tipo de sistema de almacenamiento que encaje en las necesidades de todos los usuarios. Dependiendo de la aplicación, la priorización de los requerimientos puede ser distinta, una aplicación que se fundamente en la estabilidad de los datos quedará mejor gestionada con una base de datos SQL, mientras que en una aplicación que deba escalar rápidamente será más conveniente el uso de tecnologías NoSQL.

En esta sección se describen para qué aplicaciones será más conveniente emplear cada tipo de almacenamiento

1- Almacenamiento NoSQL tipo clave-valor

Este tipo de almacenamiento suele ser una buena solución cuando la aplicación únicamente requiere o genera un tipo concreto de objeto y las búsquedas sobre esos objetos se realizan mediante un único atributo. En ese caso el uso de bases de datos tipo clave-valor resultan muy convenientes por simplicidad y rapidez.

Supóngase, por ejemplo, una web que se adapta según el usuario conectado y, para ello, se deben realizar un gran número de consultas a una base de datos RDBMS. Supongamos, además, que la información de un mismo usuario cambia en pocas ocasiones, y que, cuando lo hace, el cambio se refiere directamente a ese mismo usuario, es decir, es un cambio controlado. En ese caso, el sistema se puede modelar tomando el identificador de usuario como clave y el resto de información contenida como valor, de forma que se pueda agilizar la etapa de recuperación de datos.

Un ejemplo de este caso podría ser una aplicación similar a una parte de Facebook, la parte en la que el usuario actualiza su propio estado y, posiblemente, incluso en relación con actualizaciones de otros usuarios.

2- Almacenamiento NoSQL tipo documento

Una aplicación que encajaría en un sistema de base de datos orientada a documento sería aquella en la que se necesitase almacenar diferentes tipos de objetos y, además, realizar búsquedas empleando como clave cualquiera de ellos. Por ejemplo, una aplicación de gestión de vehículos, en la que se integren vehículos y conductores. En este caso es un requisito de la aplicación almacenar información de naturaleza muy variada (vehículos, conductores, seguros, piezas, mecánicos, citas, etc.) y, además, realizar consultas sobre cualquier campo.

No obstante, es necesario plantearse las restricciones en la consistencia de los datos, es decir, la importancia del factor concurrencia. Si la aplicación puede tolerar consistencia eventual en los datos, con atomicidad y aislamiento limitados, entonces las bases de datos orientadas a documento son una buena opción. Sin embargo, si los datos deben cumplir completa atomicidad y consistencia, por ejemplo si se desean bloquear usuarios en función de los intentos de acceso fallido, entonces la alternativa del modelo relacional resulta más adecuada.

En aplicaciones de este tipo, por tanto, puede ser conveniente implementar un sistema mixto que incluya ambas opciones.

3- Almacenamiento NoSQL orientado a columna

Los casos de uso de este tipo de base de datos son similares a los orientados a documento, es decir, muchos tipos de objetos con requerimientos de búsquedas variadas y flexibles. Sin embargo, en estos casos, debido a la forma de almacenar la información en disco, son bases de datos más orientadas a proyectos con un alto flujo de datos que, además, pueden proporcionar garantías más fuertes en lo que a consistencia se refiere a cambio de una mayor complejidad.

Supóngase el caso de una aplicación tipo eBay, en la que se requiere un particionamiento de los datos tanto horizontal como vertical:

- Búsqueda de clientes por país, de forma que se requiera eficiencia en las búsquedas en un rango de un tamaño considerable y altamente cambiante.
- Independencia de la información que cambia en pocas ocasiones, como por ejemplo direcciones de usuario, o raramente, como el número de pujas en curso para un usuario.

Aunque este tipo de particionamiento se puede conseguir en un sistema orientado a documento, mediante el uso de múltiples colecciones de diferentes dimensiones, el particionamiento es más sencillo en el caso orientado a columna.

4- Almacenamiento NoSQL orientado a grafo.

Debido a las características de estas bases datos su uso es muy concreto y fácil de determinar. En este caso los elementos cambiantes son las relaciones entre nodos, no los nodos en sí mismos y por eso suelen ser muy adecuadas para caracterizar redes de comunicaciones.

5- Modelo RDBMS

Si se requieren altos niveles de consistencia y fiabilidad, entonces las ventajas del modelo relacional son bien conocidas. Si la aplicación requiere un gran número de tablas con diferentes tipos de datos, un sistema centralizado, un lenguaje de consultas altamente extendido y soporte, entonces, siempre que el volumen de datos generado lo permita, un sistema relacional es el más conveniente.

Un buen ejemplo puede ser la aplicación de gestión de vehículos anterior en la que, además, se deban realizar búsquedas sobre cumplimiento de leyes, matrículas, fechas concretas, números de licencias, etc. Dichas búsquedas son más susceptibles del cumplimiento de ACID, de forma que la elección de una base de datos RDBMS es más adecuada.

No obstante, estas ventajas son dependientes de las necesidades de escalabilidad. Aunque actualmente la escalabilidad de las base de datos no relacionales se está viendo mejorada en gran medida, se suele asumir que la aplicación no demanda actualizaciones o joins que incluyan muchos nodos ya que la coordinación de las transacciones y el movimiento de datos entre nodos distribuidos en un esquema relacional puede ser prohibitivo e incluso, en la mayor parte de los casos, inexistente.

Finalmente coloco una tabla con las diferencias principales entre las BD SQL de las BD NoSQL.

	SQL	*NoSQL*
Almacenamiento	Modelo relacional (filas y columnas)	Modelo no relacional, diferente según la BBDD.
Flexibilidad	Esquema fijo, cada columna tiene establecido qué puede almacenar y cada fila debe cumplir las restricciones.	Esquema dinámico, los datos se pueden añadir *al vuelo* y cada entrada no necesariamente cumple los mismos requisitos.
Escalabilidad	Escalan verticalmente	Escalan horizontalmente
ACID	Cumplen ACID	Sacrifican, en ocasiones, consistencia por rendimiento y escalabilidad.