UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

TIA
Tecnología, Investigación y Academia

NoSQL: ¿es necesario ahora?

NoSQL: Is it Necessary Now?

Miguel Ángel Colorado Pérez[1]

Para citar este artículo: Colorado, M. (2017). NoSQL: ¿es necesario ahora?. *TIA*, 5(2), pp. 174-179.

Resumen

Las bases de datos relacionales han sido el auge para el almacenamiento de información por casi 40 años, solucionando problemas de redundancia, tratamiento y obtención de información; pero con la evolución de los equipos de cómputo, que son cada vez mejores en almacenamiento y procesamiento, nuevas necesidades en el manejo de información han surgido como respuesta a la manera de almacenamiento y manejo de información.

Palabras clave: CAP, clave valor, escalabilidad, SQL.

ARTÍCULO
DE INVESTIGACIÓN

Fecha de recepción:
01-06-2015
Fecha de aceptación:
15-06-2017

ISSN: 2344-8288
Vol. 5 No. 2
Julio - diciembre 2017
Bogotá-Colombia

Abstract

Relational databases have been booming for information storage for nearly 40 years, solving problems of redundancy, processing and retrieval of information. But with evolution of computer equipment, which are increasingly better in storage and processing, new information management needs have emerged as a response to the way information is stored and managed.

Keywords: CAP; key value; scalability; SQL.

[1] Ingeniero de Sistemas. Ingeniero de Software, Universidad Distrital Francisco José de Caldas. Desarrollador Senior, Meridian Group SAS. Correo electrónico: mac72652@gmail.com

INTRODUCCIÓN

Con la evolución de internet y los nuevos servicios sobre la red, el crecimiento en la generación de información ha sido considerable y no antes visto hasta el desarrollo alcanzado con la web 2.0; aplicaciones como las redes sociales, visualización de fotos, videos y estar conectados en todo momento son las razones por las cuales se ha generado este crecimiento exponencial.

SQL es el lenguaje utilizado para la gestión de bases de datos relacionales surgido a finales de los años 70 [1]; su éxito de debe a la facilidad para entender y operar los datos sin importar la aplicación que esté utilizando o los diferentes motores de bases de datos.

Como se describe más adelante en el teorema de CAP, las bases de datos SQL cumplen con el concepto de ACID (atómicas, consistentes, aisladas y durables); esto significa que el modelo relacional debe respetar las ciertas reglas mencionadas a continuación.

La atomicidad significa que una transacción se ejecuta completamente o no se ejecuta, la consistencia indica que el dato está en un estado válido o inválido, el aislamiento indica que cada transacción es independiente entre sí y la durabilidad comprende que al terminar una transacción la información perdura en el tiempo.

Sobre las bases de datos relacionales se generaron estrategias para la solución de la nueva gestión de la cantidad de información, una de estas era aumentar las características en rendimiento de procesamiento y capacidad de almacenamiento de información; esta estrategia resulta muy costosa y, a raíz de esto, surgieron nuevas formas especializadas para el manejo de información, mejorando con la experiencia. Para cada negocio se desarrolló de maneras diferentes surgiendo así aplicaciones más maduras y generándose lo que ahora se conoce como NoSQL, resolviendo nuevos escenarios como Twitter, Facebook, Linkedin.

El escalamiento en bases de datos que cumplen con el concepto de ACID son difíciles de configurar para estar en ambiente distribuido, lograr un escalamiento horizontal (utilizar varios equipos en red para el almacenamiento y tratamiento de la información) y que los tiempos de respuesta sean aceptables.

NoSQL se presenta en varios sitios web como ir en contra de lo que SQL ofrece, pero en muchos otros más se afirma que significa *Not only* SQL, resaltando que no son la solución completa a la gestión de información, pero sí una propuesta con fundamentos para los nuevos retos en tecnologías de información de la actualidad.

Antes del modelo relacional, el concepto de redes o jerárquicas fueron definidos; ahora, para nuevas necesidades, se han retomado estos conceptos para las bases de datos NoSQL.

TEOREMA DE CAP O TEOREMA DE BREWER

Antes de analizar las soluciones que brindan las bases de datos no relacionales NoSQL, se debe entender cómo se clasifican las bases de datos según las características de gestión de información en esquemas distribuidos para obtener escalabilidad [2]: consistencia, disponibilidad y tolerancia a particiones.

Según este teorema se entiende que un sistema con datos compartidos puede aprobar dos de las tres características anteriormente mencionadas; con base en este teorema podemos clasificar y visualizar las categorías de bases de datos en la Figura 1.

La consistencia indica que cada uno de los equipos/nodos del sistema maneja y adaptan la misma información al mismo tiempo, la disponibilidad muestra que para todas las peticiones da una respuesta, sea correcta o de fallo; por último, la tolerancia a particiones señala que, si uno de los nodos falla, el sistema sigue funcionando correctamente.

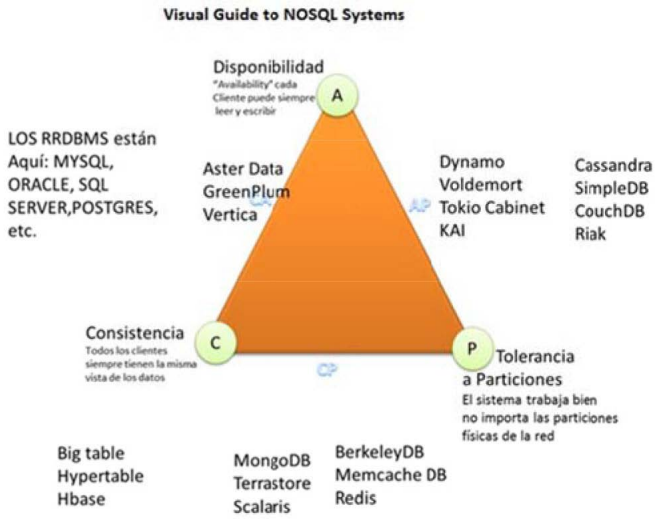

Figura 1. Guía de bases de datos

Fuente: [14].

Las bases de datos relacionales se comportan con las características de consistencia y disponibilidad, pero no permiten una tolerancia a particiones, lo cual confirma su dificultad en la escalabilidad horizontal y funcionamiento distribuido.

Como se señaló anteriormente, el concepto de ACID para las bases de datos relacionales de las NoSQL cumple las características del concepto de sistemas BASEs (*Basically Avaliable Soft state Eventually consistent*), que significa básicamente disponible, siendo aceptables respuestas aproximadas a la información donde los datos son consistentes en el tiempo, por ejemplo, en Facebook no es de importancia que en el mismo instante en que se realice una publicación todos los amigos visualicen el estado.

Las bases de datos no relacionales dejan de darle mayor importancia a la consistencia o a la disponibilidad, para solucionar las necesidades en los nuevos sistemas o formas de negocios que están surgiendo en la actualidad; las bases de datos no relacionales se han dividido en cuatro grandes grupos que se describen a continuación [3].

Dependiendo de las taxonomías de las bases de datos no relacionales se pueden dividir en llave-valor, documentos, familias de columnas o por grafos. Esto indica a como se almacena la información en las bases de datos NoSQL.

CLAVE-VALOR

Es la más popular, quizá por su sencillez en el aprendizaje y funcionalidad; intensifica su rendimiento en la obtención de información por la forma de almacenarla, de tal forma que cada elemento esta referenciado por una clave única. En el elemento se pueden agregar cualquier tipo de valor.

Lo mínimo que se tiene en una NoSQL con características clave-valor son las siguientes:

- Agregar valor y llave: put (key, value).
- Obtener valor: get(key).
- Eliminar llave: remove(key).

Con estas tres sencillas operaciones, se puede considerar una base de datos NoSQL de tipo llave valor.

Algunas de las más famosas son descritas a continuación.

- Redis [4].

Este motor de base de datos mantiene los datos en memoria como una base de datos persistente en disco, para facilitar el acceso a los datos, algunos indicadores de ellos tienen funciones para realizar contadores (incrementar o decrementar), también se tienen operaciones entre conjuntos (intersecciones, uniones, etc.). Redis también ofrece listas, colas de forma clave-valor. Una característica interesante es una simulación similar a como funciona el cache de las aplicaciones, en donde a unos registros con su clave puede asignársele un tiempo de expiración.

- SimpleDB [5].

Es el servicio de base de datos no relacional que ofrece Amazon en su plataforma de la nube promete alta disponibilidad y flexibilidad con un fácil administrador web, como todos los servicios en la nube de Amazon. Se relaciona mucho con el servicio de Amazon S3 en el que ofrece almacenamiento de documentos (archivos) y en SimpleDB se almacena la referencia o propietario de dichos documentos.

- BerkeleyDB [6].

Desarrollado por el apoyo de la universidad de Berkeley y la empresa *Sleepycat Software*, que posteriormente sería parte de Oracle. La característica que llama la atención es que permite realizar, a través de copias instantáneas (*snapshot*), modificaciones a los mismos registros en el mismo instante.

Entonces, ¿para qué casos puede utilizarse una NoSQL de tipo llave-valor?

Puede decirse que para escenarios en los que se tienen demasiadas transacciones en un corto tiempo, esto porque el procesamiento fuerte es en memoria RAM. Es recomendable para aplicaciones de análisis de datos y captura de información en tiempo real.

ORIENTADO A DOCUMENTOS

Son la extensión a las de clave-valor, solo que tienen un formato definido, como en una de las más famosas, MongoDB, la cual utiliza como formato a JSON, que es lo que almacena como documento y le asocia una clave única. Son altamente escalables, tienen conceptos de bases de datos relacionales muy similares como el de bases de datos, tablas que se llaman colecciones y el de registro o fila es el documento en una base de datos de este tipo. Todo documento tiene una clave única, muy similar a como se piensa una base de datos relacional y el desarrollo de una aplicación, un desarrollador normalmente piensa en esto al generar su código, pero el modelo relacional no lo plasma de esta manera.

El esquema del diseño de los documentos es libre y puede acomodarse a como se requiera para la implementación *software*, no todos los documentos deben ser iguales y con la misma estructura, pueden ser diferentes en una misma colección; es la característica más sobresaliente, por lo menos desde la perspectiva de un desarrollador, porque en comparación con el modelo relacional, si se desea actualizar una tabla y agregar una columna, se deben realizar varios procedimientos de actualización, mientras que en el orientado a documentos no es necesario.

Unos ejemplos de motores de este tipo son los mencionados a continuación.

- MongoDB [7].

Es la que puede parecerse más a un lenguaje como SQL tiene formas de realizar consultas, inserciones y eliminaciones; utiliza Javascript para realizar consultas u operaciones con los datos, se logra ejecutar Javascript sobre el servidor. Es muy útil en la creación de información sobre el que las inserciones se realizan sobre memoria y en algún tiempo posterior escribe esto sobre disco.

- CouchDB [8].

La característica principal es su forma de tratamiento de la información a través del protocolo HTTP con REST (a través de GET o POST), en donde en lugar de escribir un SQL (SELECT) se escribe una url y se accede a los datos. Tiene implementado por defecto el algoritmo Map/Reduce de Google para la distribución de tareas en diferentes nodos [9]. Mantiene su esquema del modelo libre, para ser implementado según las necesidades del desarrollo de *software*. Al igual que MongoDB utiliza Javascript para realizar funciones como vistas sobre los datos.

Al igual que en las de clave-valor la pregunta es, ¿en qué casos se debe utilizar este tipo de bases de datos?

Funciona muy bien para información que no va a cambiar mucho, que se utilice la base de datos como un repositorio de información. La forma de acceder a la información es casi siempre la misma. De igual manera funciona para información que se va a persistir en algún momento y debe ser insertada en poco tiempo.

ORIENTADO A GRAFOS

Buscan representar la información como grafos, con sus componentes nodos determinando las relaciones a través de las aristas. Se utilizan para información que es usada a menudo, ofreciendo hash distribuido y estructuras de datos sencillos como arrays asociativos o almacenes como los anteriormente mencionados de clave-valor.

Algunos de los más conocidos son los mencionados a continuación.

- Neo4j [10].

Se describe como un motor de persistencia transaccional en Java; tiene un servidor como una extensión con una API REST en Java. Promete que, aunque la cantidad de información aumente, el rendimiento no se verá comprometido.

- Flockdb (Twitter) [11].

Es la base de datos diseñada por Twitter, diferenciándose por su misma funcionalidad principal en Twitter, donde solo interesa la relación de un nodo (en este caso un nodo es una cuenta de Twitter). Según el artículo "Exploration of NoSQL: FlockDB" [12] habla de la cantidad de información que se almacena y de los nodos que son requeridos para el funcionamiento de Twitter, en donde se ha llegado a un tráfico pico de 20 000 escrituras por segundo y 100 000 lecturas por segundo.

Este tipo de base de datos, como se vio anteriormente, son ampliamente utilizadas por redes sociales para conocer tendencias de los usuarios o de los movimientos en las comunidades; funcionan muy bien para identificar las relaciones de las personas que pertenecen a las redes de internet. Son altamente distribuibles para su tolerancia a particiones respuestas en todo momento

ORIENTADO EN COLUMNAS

Su característica principal es, como lo dice su nombre, guardar la información en columnas en lugar de en filas como el modelo relacional; a cada una de las columnas se le asigna una clave única y evita almacenar valores nulos, ganando velocidad en lectura, pero no es eficiente en realizar escrituras o actualizaciones.

Las más sobresalientes de este tipo son las descritas a continuación.

- Cassandra [12].

Originalmente fue desarrollada para Facebook y luego se donó a la fundación Apache. Es totalmente descentralizada y eventualmente consistente con los datos; tiene altísima disponibilidad al copiar la información en cada uno de sus nodos.

- Hbase [13].

Está desarrollada en Java por Apache Software Fundation's Hadoop, es altamente distribuible y con baja tolerancia a fallos. Al trabajar con Hadoop funciona muy bien con el concepto del algoritmo de Map Reduce para el tratamiento eficiente de operaciones con gran cantidad de información.

Se utilizan generalmente para soluciones analíticas de *business intelligence* en donde tiene sentido leer por columnas y mostrar resultados resumidos y no tener que leer registro por registro y realizar agregaciones de toda una tabla en esquema relacional. De igual manera, para aplicaciones sobre internet como redes sociales, es sencilla la distribución de información en sus nodos, para bajar los cuellos de botella en las peticiones de sus afiliados.

CONCLUSIONES

Las bases de datos no relacionales no son la única solución a los nuevos retos que se enfrentan a las tecnologías de información actualmente, siendo estas alternativas adaptadas a diferentes tipos de negocios. El principal problema que intentan solucionar el NoSQL es la escalabilidad y la tolerancia a la partición, por la cantidad de información a gestionar.

La información en NoSQL se debe guardar y modelar dependiendo de cómo se va a consultar, sobre cada uno de los tipos es diferente su modelo y diseño de base de datos. A diferencia de SQL, el NoSQL todavía es muy joven como tecnología, así que es muy probable que esta tecnología no sea estable o tenga muchos cambios en poco tiempo.

NoSQL no es la solución a todos los problemas de almacenamiento de información y cada uno de los tipos soluciona problemas específicos.

TRABAJO A FUTURO

El trabajo a futuro lleva a formalizar una metodología para la elección entre bases de datos relacionales y no relacionales, dependiendo de los requerimientos del *software* y del negocio sobre el cual se diseña el sistema informático.

Así se diseñan diferentes escenarios de negocios para comprender mejor cómo elegir entre una u otra solución o quizá realizar híbridos entre las soluciones de *software*.

REFERENCIAS

[1] Campus MVP. (2015). *Qué es el lenguaje SQL*. Recuperado de: http://www.campusmvp.es/recursos/post/Que-es-el-lenguaje-SQL.aspx

[2] Berkeley. (2000). Towards Robust Tow*ards Robust. Distributed Systems*. Recuperado de: http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

[3] Pech, F. (2011). *Base de datos NoSQL*. Recuperado de: http://www.tamps.cinvestav.mx/~fpech/ddb/files/slides/nosql.pdf

[4] Redis. (2015). *Redislab*. Recuperado de: http://redis.io/

[5] Amazon. (2015). *Amazon SimpleDB*. Recuperado de: http://aws.amazon.com/es/simpledb/

[6] Oracle. (2015). *Oracle Berkeley DB 12c Release 1*. Recuperado de: http://docs.oracle.com/cd/E17076_04/html/index.html

[7] Mongodb. (2015). *MongoDB Atlas. Database as a Service*. Recuperado de: http://www.mongodb.com/

[8] CouchDB, relax (2015). *Data Where You Need It*. Recuperado de: http://couchdb.apache.org/

[9] Google. (s.f.). *www.research.google.com*. Recuperado de: http://research.google.com/archive/mapreduce.html

[10] Neo4j. (2015). *Introducing Neo4j 3.2. The Graph Foundation for Internet-scale Applications*. Recuperado de: http://neo4j.com/

[11] Quora. (2015). *FlockDB*. Recuperado de: https://www.quora.com/FlockDB

[12] Cassandra.apache, (2015). *Manage Massive Amounts of Data, Fast, Without Losing Sleep*. Recuperado de: http://cassandra.apache.org/

[13] Hbase.apache. (2015). *Welcome to Apache HBase*. Recuperado de: http://hbase.apache.org/

[14] Sepúlveda, W. (s.f.). *Bases de datos NOSQL*. Recuperado de: http://basesdedatosnosql.blogspot.com/