Revista Tlamati Sabiduría

Migración de Bases de Datos SQL a NoSQL

Angeles Cruz Manjarrez Antaño
Universidad Autónoma de Guerrero
Unidad Académica de Ingeniería
Chilpancingo, Guerrero.
(01 747) 491 22 27
angeles.cruzmanjarrez@gmail.com

José Mario Martínez Castro
Universidad Autónoma de Guerrero
Unidad Académica de Ingeniería
Chilpancingo, Guerrero.
jmariomtz@yahoo.com

René E. Cuevas Valencia
Universidad Autónoma de Guerrero
Unidad Académica de Ingeniería
Chilpancingo, Guerrero.
reneecuevas@gmail.com

RESUMEN

Desde el momento en que los humanos han requerido almacenar datos que lleven a la obtención de información oportuna y fiable también se ha buscado la manera más eficiente de manipular dichos datos. Con el tiempo la creciente cantidad de información que se intenta manejar supera por mucho la infraestructura existente, esto es, se ha llegado a la necesidad de buscar soluciones que permitan almacenar cada vez más información incluso de la que se podría imaginar. El tema propuesto viene de la necesidad explícita de migrar datos de una plataforma determinada a otra que permita mayor flexibilidad y manipulación de datos más extensos y variados.

Palabras Clave

Migración, Bases de Datos, SQL, NoSQL, CSV, JSON, MongoDB.

1. INTRODUCCIÓN

El almacenamiento de datos en algún momento se enfrenta a la necesidad inerte de realizar migración. La migración es un proceso un tanto complicado considerando los puntos que deben tomarse en cuenta al momento de llevarla a cabo, tales como la fuente de datos y el destino de los mismos, la estructura existente y la movilidad hacia una nueva, el análisis de la persistencia y depuración de los datos, las plataformas entre las que se llevará a cabo la migración, entre otros.

La capacidad de almacenamiento del servidor de datos, nuevas necesidades para el software así como los nuevos requerimientos de los usuarios finales, sin lugar a dudas, llevan a pensar seriamente en mover la información de una estructura a otra. Dadas las nuevas necesidades, la cantidad de información que ingresa día a día a la base de datos, aunado a la información ya existente proporciona una idea de lo que se necesita para el futuro tomando en cuenta el crecimiento de los mismos, la variación en la información y el tipo de datos a almacenar.

Por los puntos expuestos, este trabajo de investigación pretende analizar los puntos a considerar en el proceso de migración de datos entre modelado relacional y el NoSQL sin dejar de lado las necesidades que orillan a este cambio y algunas de las opciones que pueden considerarse.

2. PROCESO DE MIGRACIÓN DE DATOS

La migración de datos consiste en la transferencia de materiales digitales de un origen de datos a otro. Se trata de una consideración clave para cualquier implementación, actualización o consolidación de un sistema informático.

Existen diversos motivos para realizar una migración, tales como la preservación o difusión de los contenidos, mejoras en el funcionamiento, cumplir con nuevos requerimientos de usuario o de software, la interoperabilidad, la actualización de versiones, la estandarización de la tecnología, la reducción de costos al optar por un software libre, el aumento en el volumen de datos, nuevos procesos de negocio o mejoras en la seguridad o el control de la información, entre otros escenarios posibles. [3]

El proceso de migración requiere de pasos específicos para llevarse a cabo, tales como, observación de la fuente de datos y el destino de los mismos, de los formatos en los que se presenta la información para identificar las conversiones para adecuarse a las nuevas necesidades. Se necesita conocer la flexibilidad para adaptar la nueva estructura de modo que se preserven los datos realmente necesarios, lo que llevará más adelante a la depuración en caso de ser necesario. Realizar pruebas previas a la migración definitiva servirá de mucha ayuda a la persona encargada de esta

tarea para poder identificar las excepciones que no hayan sido previstas. Por último la ejecución de la migración en su totalidad.

2.1 Técnicas de migración de base de datos

Al momento de plantear la necesidad de migrar datos lo primero en lo que se puede pensar es, qué, cómo y el tiempo que llevará realizar esta tarea. La utilización de técnicas permite realizar un trabajo ordenado, conocer las estructuras de las bases de datos involucradas, familiarizarse con los datos y su distribución, considerar aquellos datos que requerirán atención especial, aquellos que en adelante no serán indispensables, entre otros. A continuación se listan algunos de los pasos a tomar en cuenta antes, durante y posterior a la migración.

- **Planificación**. Lo más importante al migrar una base de datos es llevar a cabo un proceso de planificación y análisis del trabajo. Debe considerarse el análisis de la estructura de la base de datos origen y la planificación.
- **Contador de registros**. Se debe hacer un conteo de los registros existentes, los registros migrados exitosamente y conservar los registros que no se logren migrar, esto dará a los usuarios la certeza de que su información es coherente y no ha sufrido pérdidas. En este punto puede tomarse en cuanta la opinión de los administradores de base de datos para tomar acciones correctivas acerca de los datos que no hayan podido migrarse.
- **Mapeador de tipos de datos**. Algunas plataformas no soportan algunos tipos de datos, así que es necesario planificar el mapeo de los campos en la nueva base de datos.
- **Restricciones y trigers**. Antes de iniciar la migración de la BD (Base de Datos), es recomendable deshabilitar los Trigers y/o restricciones que nos puedan generar error al momento que el SMBD (Sistema Manejador de Base de Datos) ejecute el proceso de escritura de los datos.
- **Codificación de caracteres**. Cuando el copiado se realiza de forma automática, es necesario identificar la codificación de caracteres que la base de datos destino espera, pues así evitaremos el reemplazo automático de caracteres o en su caso, pérdida de los mismos.
- **Pruebas**. Toda buena implementación requiere una fase de pruebas, podría pensarse en presentación de los reportes utilizados por los usuarios de la aplicación para tener la certeza de que no existirá ausencia de información al momento de poner en marcha la migración. Este paso da la oportunidad de observar y corregir las excepciones no controladas.
- **Implementación**. Como su nombre lo dice, es la fase que implica poner en marcha la migración física de los datos. Según los requerimientos debe identificarse el momento apropiado y el tiempo estimado que puede tardarse en llevarse a cabo. Existen sistemas que pueden ser pausados por minutos, horas e incluso días enteros, por el contrario existen otros tantos que no pueden detener su marcha pues representaría perdidas económicas para la empresa que lo requiera.
- **Monitoreo**. Una vez finalizada la fase de implementación es recomendable realizar observaciones muy de cerca para asegurarse de que la información que se ha migrado es consistente y fiable, en caso de no ser así puede requerirse corrección de errores.

3. BASES DE DATOS NoSQL

El término NoSQL (Not Only SQL), se refiere a una multitud de bases de datos que intentan solventar las limitaciones que el modelo relacional se encuentra en entornos de almacenamiento masivo de datos, y concretamente en las que tiene en el momento de escalar, donde es necesario disponer de servidores muy potentes y de balanceo de carga. [4]

La propuesta de bases de datos NoSQL no es excluyente de las SQL, por el contrario podrían incluso utilizarse como complemento una de la otra. Los años de uso e implementación de bases de datos relacionales ha dado como resultado tener una mente predispuesta a generar sentencias SQL que permitan ingresar y manipular datos así como generar reportes informativos casi inconscientemente, lo que lleva en algún momento a poner cierta resistencia al uso de este nuevo paradigma, sin embargo esto no debe ser una limitante puesto que existen bases de datos como MongoDB o Neo4j que son fáciles de aprender y son libres, lo que los hace aún más interesantes por el hecho del soporte, la basta documentación y las redes de información que propician sus usuarios.

Algunos ejemplos de base de datos NoSQL:

- Bases de datos orientadas a documentos. Cassandra, HBase utilizada por Facebook, MongoDB y CouchDB, Google's BigTable
- Bases de datos orientadas a grafos. Neo4j.
- Bases de datos orientadas a objetos. Db4Objects de Versant y Objectivity/DB.

3.1 Características

El teorema CAP o teorema Brewer, dice que en sistemas distribuidos es imposible garantizar a la vez: consistencia, disponibilidad y tolerancia a particiones (Consistency-Availability-Partition Tolerance). A continuación se describen las características:

- Consistencia: al realizar una consulta o inserción siempre se tiene que recibir la misma información, con independencia del nodo o servidor que procese la petición
- Disponibilidad: que todos los clientes puedan leer y escribir, aunque se haya caído uno de los nodos.
- Tolerancia a particiones: a veces traducido como tolerancia a fallos. Es una traducción que no me gusta, ya que se confunde con el punto anterior. Los sistemas distribuidos pueden estar divididos en particiones (generalmente de forma geográfica). Así que esta condición implica, que el sistema tiene que seguir funcionando aunque existan fallos o caídas parciales que dividan el sistema.

Para ser escalables y distribuidas, las bases de datos NoSQL, siguen distintos métodos, por lo que no todas cumplen los mismos puntos del teorema CAP.

- **AP,** garantizan disponibilidad y tolerancia a particiones, pero no la consistencia, al menos de forma total. Algunas de ellas consiguen una consistencia parcial a través de la replicación y la verificación.

- **CP**, garantizan consistencia y tolerancia a particiones. Para lograr la consistencia y replicar los datos a través de los nodos, sacrifican la disponibilidad.

- **CA**, garantizan consistencia y disponibilidad, pero tienen problemas con la tolerancia a particiones. Este problema lo suelen gestionar replicando los datos. [5]

En la Figura 1, se observa cómo se reparten algunas de las bases de datos según las condiciones que cumplen del teorema CAP.

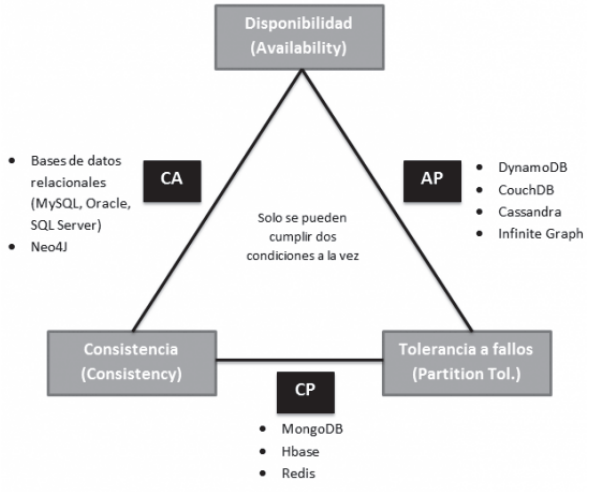

Figura 1. Teorema de CAP.

Es necesario tener en cuenta, que esta clasificación no es definitiva, ya que algunos de estos sistemas NoSQL pueden configurarse para cambiar su comportamiento.

Por ejemplo MongoDB es CP por defecto. Pero también se puede configurar el nivel de consistencia, eligiendo el número de nodos a los que se replicarán los datos[6].

3.2 Clasificación

Una de las clasificaciones más significativas de los sistemas de almacenamiento NoSQL es la basada en el tipo de estructura de datos o el esquema bajo el cual se almacenan los datos.

- **Basadas en documentos**, almacenan a información como si fueran documentos. Por lo general utilizan formato JSON o XML. El funcionamiento es muy similar a las clave-valor, sólo que en este caso el atributo clave es el nombre que se le pone al fichero.

- **Orientadas a grafos**, este tipo de bases de datos almacena la información en forma de grafo, de forma que las relaciones entre nodos pueden tener atributos. Son recomendables para sistemas que se puedan representar en forma de red de manera sencilla.

- **Orientadas a columnas**, almacenan toda la información en columnas de esta forma las lecturas son muy rápidas, pero se sacrifica mucho tiempo para las escrituras, por lo que no son recomendables a no ser que el número de lecturas sea muy superior al número de escrituras.

- **Clave-valor**, es la forma más usada, su uso es muy parecido al de una tabla de Hash, en la que se almacena una clave, y un valor. El valor se suele almacenar como un tipo "BLOB" de esta forma el sistema es independiente al tipo de datos que se quiere almacenar. Su principales características son:
    - Son muy rápidas para las operaciones de lectura/escritura.
    - Fácilmente escalables particionando la clave, de esta forma se pueden almacenar las distintas claves en distintos servidores dependiendo de su valor inicial.

3.3 Los que han migrado a NoSQL

Por lo anterior, no debería sorprender el hecho de que compañías y grandes marcas han migrado su información de base de datos relacionales a NoSQL, el motivo es: el crecimiento acelerado de la información que almacenan, la concurrencia de usuarios, las transacciones que se realizan cada segundo, la necesidad de obtener y mostrar información de manera rápida y confiable, todo esto encaminado a la seriedad que proyectan hacia sus usuarios. Entre las marcas más conocidas se puede observar a:

- MongoDB: CISCO, FourSquare, MetLife, MTv.
- Cassandra: Facebook, Twitter, Spotify, Microsoft, Instragram, GE, Disney, Adobe, ebay.
- BigTable: Google.
- Dynamo: Amazon.
- Project Voldemort: LinkedIn.
- Entre muchos otros.

4. MIGRACIÓN DE SQL A NoSQL

Pueden existir infinidad de motivos por los cuales se desea migrar datos, entre los más comunes se encuentran la necesidad de mover los datos de un origen a otro dada por el cambio de plataforma, las dificultades que representa el trabajar con grandes cantidades de información, la estructura actual sobre la que se soportan los datos no es lo suficientemente extensa, la infraestructura con que cuenta la organización para almacenar su información no es suficiente, inclusión de campos para almacenamiento de tipos de datos no soportados por la estructura de base de datos actual, nuevos requerimientos del software que alimenta de datos a la base de datos, nuevas necesidades de los usuarios finales, y así un sin fin

de argumentos totalmente válidos para llevar a cabo esta fase llámese de mantenimiento o actualización.

Como en toda migración, hay puntos importantes que necesitan analizarse, entre los cuales se pueden encontrar: el volumen de la información a migrar, los tipos de datos a migrar, se debe también conocer la estructura de la base de datos actual con el fin de empatar campos y hacer las conversiones correspondientes antes de realizar alguna acción que pueda provocar pérdida de datos o que la migración simplemente no concluir de manera correcta por la generación de errores incontrolados.

Los manejadores de base de datos así como el lenguaje que utilizan son pieza clave en este proceso.

Para no quedar fuera de contexto se debe conocer a lo que se enfrenta. Las sentencias SQL a las que se están acostumbrados quedaran un poco de lado, sin embargo citamos algunas comparativas para estar familiarizados con el tema, ver tabla 1.

Tabla 1. Tabla de términos y conceptos MongoDB

Términos y Conceptos	
SQL	**MongoDB**
database	database
table	Collection
row	Document
column	Field
Index	index
join	-
primary key	primary key (_id)

SQL STATEMENT	MONGO QUERY LANGUAGE STATEMENT
CREATE TABLE USERS (a Number, b Number)	db.createCollection("users")
INSERT INTO USERS VALUES(1,1)	db.users.insert({a:1,b:1})
SELECT a,b FROM users	db.users.find({}, {a:1,b:1})
SELECT * FROM users	db.users.find()
SELECT * FROM users WHERE age=33	db.users.find({age:33})
SELECT * FROM users WHERE age>33	db.users.find({'age':{$gt:33}})
SELECT * FROM users WHERE name LIKE "%Joe%"	db.users.find({name:/Joe/})
SELECT * FROM users WHERE age>33 AND age<=40	db.users.find({'age':{$gt:33,$lte:40}})
SELECT * FROM users ORDER BY name DESC	db.users.find().sort({name:-1})

SQL STATEMENT	MONGO QUERY LANGUAGE STATEMENT
SELECT * FROM users WHERE a=1 and b='q'	db.users.find({a:1,b:'q'})
SELECT * FROM users WHERE a=1 or b=2	db.users.find({ $or : [{ a : 1 } , { b : 2 }] })
SELECT DISTINCT last_name FROM users	db.users.distinct('last_name')
UPDATE users SET a=1 WHERE b='q'	db.users.update({b:'q'}, {$set:{a:1}}, false, true)
DELETE FROM users WHERE z="abc"	db.users.remove({z:'abc'});

Figura 2. Sentencias básicas lenguaje de consultas MongoDB.

Como se ha mencionado, entre las ventajas de los sistemas not only SQL se encuentran: la velocidad de inserción, actualización y consulta de datos, en la figura 3 se tiene una tabla comparativa del tiempo aproximado de pruebas comparativas realizadas entre MySQL y MongoDB.

Test	MySQL	MongoDB
Insert 10000 records	16,327	1,912
Update 1000 records based on a non-key update	31,151	6,484
Update 1000 records based on a key query	1,592	1,097

Unit in millisecond <

Figura 3. Comparativa de tiempos, MySQL y MongoDB.

5. ORIGEN Y DESTINO

Fuentes de Datos: CSV, XML, JSON, MySql, Oracle, etc

Colección de datos Objetivo: JSON

Una vez que se tenga claro dónde se está, hacia dónde se quiere ir y, sabiendo que los datos son la base de toda información requerida para la toma de decisiones, no se pasará de una estructura a otra por simple vanidad, si bien las bases de datos NoSQL ofrecen gran capacidad de escalabilidad, espacio para almacenamiento, flexibilidad, entre otras, no se debe dejar de lado las posibles necesidades de tener una estructura fija que permita realizar la manipulación segura consciente de la información.

La figura 4 es una representación gráfica del almacenamiento en bases de datos relacionales comparado con las bases de datos NoSQL de tipo basadas en documentos.

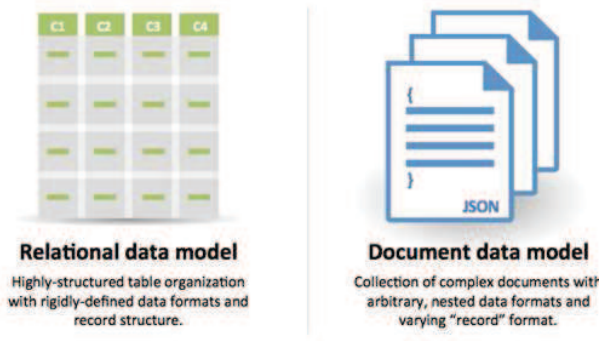

Figura 4. Modelos relacional vs orientado a documentos

Cada registro en una base de datos relacional se ajusta a un esquema - con un número fijo de campos (columnas) cada uno con un propósito especificado y tipo de datos. La ventaja es que hay menos datos duplicados en la base de datos. La desventaja es

que un cambio en el esquema significa realizar varias declaraciones "alter table".

Con las bases de datos de documentos en el otro lado, cada documento puede tener una estructura completamente diferente de otros documentos. No se requiere ninguna gestión adicional en la base de datos para controlar los cambios de esquemas de documento2.

Una vez dicho lo anterior se centrar la atención en MongoDB, utilizando el paradigma basado en objetos, es una de las base de datos no relacionales más famosas.

MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Entonces, teniendo una colección de datos CSV o XML pueden convertirse al formato JSON, a partir de este último se puede generar las sentencias correspondientes para realizar inserciones de datos en nuestro nuevo entorno.

Como ejemplo se puede citar una herramienta ubicada en la red con la siguiente dirección, esta puede convertir texto en formato CSV que bien se puede obtener realizando una consulta a la base de datos de producción y guardando los resultados con extensión .csv para después convertirla al formato JSON más amigable con MongoDB. [2]

Este tipo de herramientas y/o programación ya sea en Phyton o JavaScript ayudarán a generar una mejor interfaz de migración de datos.

6. CONCLUSIONES

Regularmente lo nuevo es deslumbrante y el caso de bases de datos y su estructura no es una excepción, por el contrario, cuando se escucha hablar del modelado NoSQL se piensa en migrar, en tener más flexibilidad y estar a la vanguardia en aspectos de información y su almacenamiento; pues bien, en este caso no siempre lo más nuevo es la mejor opción, todo dependerá del tipo y volumen de información que se pretenda guardar.

Como se explicó ya en párrafos anteriores, la migración no es una tarea fácil, no es una decisión que se tome a la ligera, se deben considerar puntos esenciales, como el origen y destino de datos, mapeo de campos, conocer las diversas opciones que se tienen y los beneficios que ofrecen las nuevas estructuras así las facilidades que ofrecen para dar este gran salto.

Esta investigación pretende proporcionar una idea clara de lo que se debe considerar antes y durante la migración para hacer de esto una tarea más sencilla y fácil, tomando en cuenta un proceso específico y herramientas adecuadas para no abalanzarse a realizar un cambio que pudiera no ser necesario tomando considerando los beneficios que se obtendrían en relación con el costo que pudiera implicar.

7. REFERENCIAS

[1] Agile and Scalable, MongoDB. (May. 2014) , http://www.mongodb.org/

[2] CSV to JSON Converter, JSFIDDLE. (May. 2014), http://jsfiddle.net/sturtevant/AZFvQ/

[3] Migración de datos, Wikipedia. (Abril 2014). http://es.wikipedia.org/wiki/Migraci%C3%B3n_de_datos

[4] NoSQL. *Conocimiento con todos y para todos EcuRed.* (May. 2014), http://www.ecured.cu/index.php/NoSQL.

[5] NoSQL: clasificación de las bases de datos según el teorema CAP, GENBETA:dev. (Enero 2014). http://www.genbetadev.com/bases-de-datos/nosql-clasificacion-de-las-bases-de-datos-segun-el-teorema-cap

[6] Not Only SQL, NoSQL By Sutthinun Peauut. (May. 2014), http://www.kana45.com/Present/NoSQL.htm#/

[7] Transitioning from RDBMS to NoSQL. Interview with Couchbase's Dipti Borkar, InfoQ. (May. 2014), http://www.infoq.com/articles/Transition-RDBMS-NoSQL