Published in

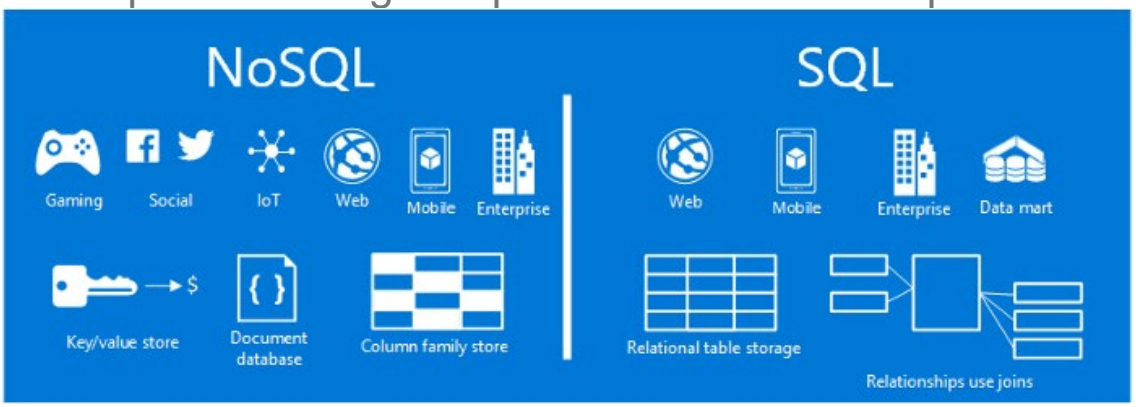
TechWo

[Marcela Sena](#)

Mar 28, 2017
·
7 min read

Como pasar de SQL a NoSQL sin sufrir

Conceptos de MongoDB para Devs — Primera parte



Introducción

Como developers es común que en alguna etapa de nuestra vida desarrollando software nos hayamos topado con conceptos de *bases de datos*, puesto que la información de los sistemas,

web apps, mobile apps, entre otros, que solemos desarrollar necesitan ser almacenados en algún lugar para su posterior consulta.

Los conceptos de bases de datos con los que estamos familiarizados son los *relacionales* pues fue de los primeros modelos que se hizo popular por su forma de organizar y consultar la información almacenada. Al usar bases de datos relacionales estamos también familiarizados con *SQL* (Structured Query Language) y la forma de poder usar este lenguaje para la definición de datos (DDL) y la consulta de estos mismos (DML) en sistemas de gestión de bases de datos ó DBMS (Database Management System).

Tablas, columnas, registros, restricciones, relaciones, consultas (queries), procedimientos almacenados, son términos que usamos para referirnos a los datos de nuestro sistema que deseamos manipular y mentalmente transformamos esos datos al modelo relacional.

No quiero generalizar, sé que muchos Devs de la nueva escuela tal vez ya tienen en mente otros conceptos y términos y ya mapean sus datos a otros modelos, es por eso que este post servirá para todos aquellos que quieren migrar de SQL a NoSQL sin sufrir.

MongoDB

MongoDB es una base de datos de documentos *open-source* que nos permite indexar registros de texto completo que se encuentra estructurado en un *Documento*. En MongoDB un documento es un registro compuesto por pares “campo : valor”. Estos documentos son muy parecidos a los objetos JSON.
User document

Los valores de los campos pueden ser otros documentos, arreglos y arreglos de documentos. Ejemplos de estos tipos de valores están en la línea 5 para arreglos, 6 para documentos y 7 para arreglo de documentos.

Las ventajas de tener este tipo de estructura de datos como registros es que principalmente podemos reducir las “relaciones” entre tablas que comúnmente usamos en bases de datos relacionales cuando hacemos consultas con los famosos “joins” o uniones de datos, ya que resulta muy costoso. Otra ventaja es que de cierta forma esta estructura se corresponde con tipos de datos en varios lenguajes de programación lo cual resulta conveniente para desarrollar código de forma más ágil. Además, también nos permite guardar grandes volúmenes de datos sin tanto costo.

MongoDB v3.4 promete un alto desempeño en tareas de I/O con el uso de modelos de datos embebidos e índices, además de alta disponibilidad haciendo réplicas y reduciendo la redundancia de datos; en MongoDB es posible escalar horizontalmente agregando clusters de maquinas donde los datos se distribuyen por medio de sharding.

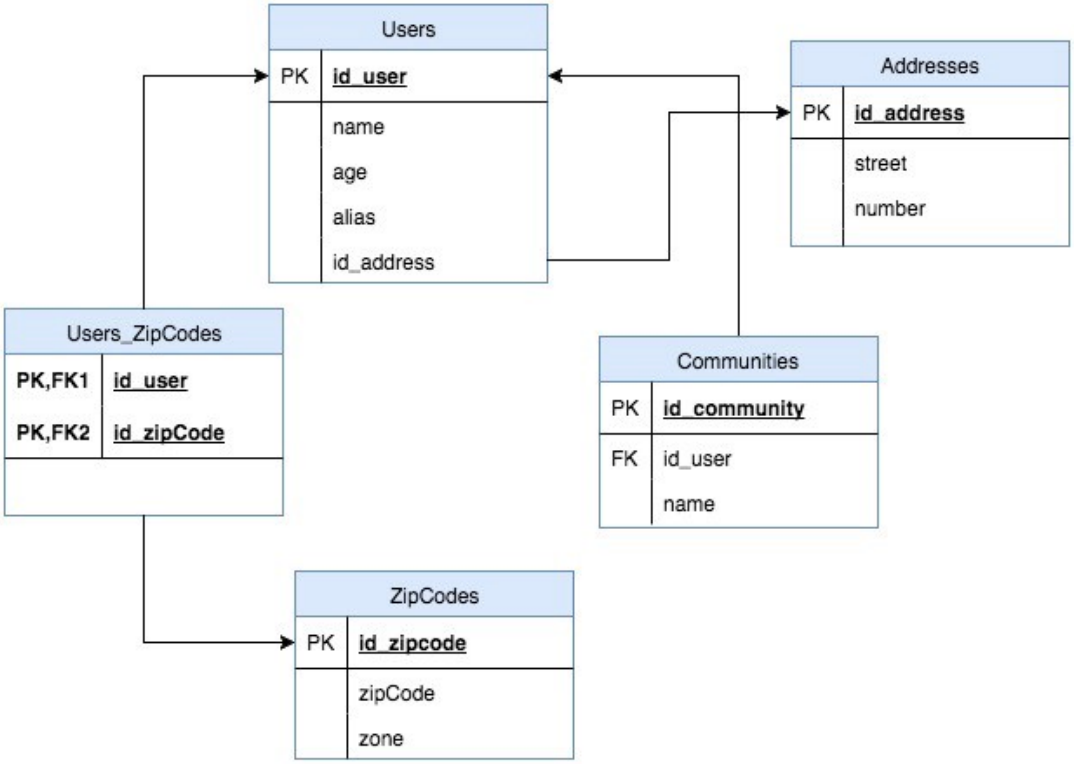
Y claro, MongoDB soporta un lenguaje de consultas enriquecido para hacer operaciones de escritura y lectura (CRUD), además de búsquedas de texto, consultas geoespaciales y agregación de datos.

How to install mongodb.

Migrando de modelo de datos

Ya vimos todo lo que MongoDB nos puede brindar para manejar nuestros datos, pero lo que queremos es saber como migrar nuestro modelo relacional a un modelo de documentos, y para esto necesitamos un ejemplo.

Usemos el ejemplo del documento anterior que representa a un usuario definiendo su nombre, edad, alias, las comunidades de las cuales es miembro, domicilio y las zonas donde se ubica. La representación del modelo relacional para almacenar estos datos sería la siguiente:



Users (modelo relacional)

Ahora veamos la representación en el modelo de documentos:

Users (modelo de documentos)

La diferencia sustancial sería la abstracción completa de todos los datos definidos en un solo documento y no en distintas tablas relacionadas. Las restricciones y la existencia de “ids” de otras tablas como parte de nuestros registros ya no son relevantes pues lo que nos interesa es la información como tal. Y aquí es donde podemos entrar en detalles sobre conceptos correspondientes entre ambos modelos.

El concepto de *database* en ambos modelos es el mismo, donde las cosas empiezan a cambiar es en la parte de definición de *tables/collections*. En MongoDB se le llama *collection* a la agrupación de *documents* (BSON — binary JSON) que se almacenan en una database. Por lo tanto, las operaciones de lectura y escritura y las consultas que haremos serán sobre una database y sus collections, definiendo así para nuestro ejemplo:

```
Mongo Shell statementspeopledb -> databaseuse <db>, crea una database si no existe y selecciona esta db para ser usada.> use peopledbusers -> collectiondb.createCollection(<name>, <options>), crea explícitamente una collection. Otra forma de hacerlo es cuando agregamos el primer registro a la collection con db.collectionName.insertOne(), si la collection no existe Mongo la crea e inserta el primer registro.> db.createCollection(“users”)O> db.users.insertOne()
```

Operaciones CRUD

A partir de aquí podemos comenzar a manipular datos con las operaciones comunes CREATE, READ, UPDATE, DELETE; y para migrar de SQL a NoSQL con Mongo, lo haremos paso a paso con una comparativa entre sentencias en uno y otro lenguaje.

Create y Alter, son sentencias (DML) que nos ayudan a crear estructuras de datos, y a modificarlas una vez que fueron definidas.

Los
métodos ***insertOne(***<document>***)*** e ***insertMany(***[<doc1>,<doc2>…]***)*** implícitamente crean una *collection* si no existe e insertan uno o muchos *documents* según sea el caso. Por si no se dieron cuenta en Mongo no tuvimos que especificar el "id" de nuestro *document*, si no definimos un campo "_id", Mongo lo crea automáticamente.

El método ***updateMany(***<filter>,<update>***)*** actualiza múltiples *documents* dentro de una *collection* basado en el filtro que recibe como argumento, pero además puede alterar los *documents* con operaciones de ***$set*** (establecer) y ***$unset*** (des-establecer) para agregar o borrar *fields*. Como vemos la forma de alterar información en Mongo no se hace a nivel de *collections* ya que no es una modificación estructural sino de *documents* (registros de nuestra colección).

Con **insert** nosotros podemos agregar registros a nuestra definición de datos. En Mongo ya vimos la forma de hacer estos inserts con los métodos *insertOne()* e *insertMany()*.

Update y **Delete**, son sentencias que nos ayudan a hacer modificaciones a la información, actualizar algún valor o a borrar algún registro.

En Mongo nosotros podemos usar el método ***updateMany(***<filter>,<update>***)*** para hacer modificaciones a los valores de un *document* en específico o a un rango de ellos en base a un filtro. Para nuestro ejemplo estamos modificando el valor del campo “alias” en base a una condición sobre el campo de “age”, esta condición determina cuáles serán los *documents* que van a ser modificados. Los operadores de consulta (query operators) ***$set*** y ***$gt*** son usados para establecer el nuevo valor en “alias” y para definir el rango mayor a 25 del campo “age”.

Para borrar *documents* usamos el método ***deleteMany(<filter>)*** que borra todos los *documents* que coinciden con el filtro.

Por último tenemos la sentencia **select** la cuál nos permite hacer consultas sobre la información que hemos almacenado. Como hemos notado en las anteriores comparativas sobre las sentencias de DML, Mongo no es muy diferente a SQL, se pueden hacer las mismas cosas con la principal diferencia de que en mongo usamos los mismo métodos pero con diferentes parámetros para modificar información ya que todo se hace a nivel de *documents* y en SQL debemos hacer diferencias entre conceptos de estructura como tables, columns, rows, y los valores que conforman los registros de esas estructuras.

Para mi, el manejo de DDL es donde está la gran diferencia entre SQL y NoSQL puesto que al no tener tablas relacionadas la

forma de hacer consultas cambia drásticamente. Veamos algunos ejemplos:

En Mongo nosotros podemos usar el método *find* para hacer búsquedas sobre documentos que no involucran a otros documentos. Para hacer funciones de agregación como GROUP BY y ORDER BY Mongo nos provee de un método *aggregate* el cuál nos ayuda a agrupar y ordenar documentos usando ***$group*** y ***$sort*** además de poder hacer uso de los campos que son parte de otros documentos dentro de nuestros documentos. En nuestro ejemplo "zipCodes" es un campo que su valor es un arreglo de documentos y haciendo uso de ***$unwind*** es como podemos separar estos documentos para poder hacer referencia a sus campos internos. En este método también podemos hacer uso de los operadores ***$sum***, ***$gt***, ***$lt*** y otros para hacer acumulaciones, comparaciones o cualquier otra operación permitida sobre los valores.

Conclusión

MongoDB es una buena opción para comenzar con las bases de datos del modelo de documentos, pero debemos tener mucho cuidado sobre qué datos deben ir como parte de un documento y qué datos deben ir por separado ya que podemos llegar a complicar nuestros queries si no tenemos un buen diseño de nuestros datos. En este post vimos sentencias básicas de un modelo y de otro y pudimos darnos cuenta que podemos representar nuestros data sets en una estructura más simple sin complicarnos con las relaciones del modelo relacional que

muchas veces hacen que los datos sean redundantes y que mantener tantas tablas puede ser complicado al momento de hacer queries. Si eres un developer que está migrando de modelo te darás cuenta que al usar este modelo con MongoDB te será más fácil modelar y manipular los datos de tu sistema, además de que es fácil de integrar con distintos lenguajes gracias a su amplia variedad de APIs.

En mi siguiente post crearemos un proyecto web y usaremos la API de Java para conectarnos a una database de MongoDB e implementaremos las operaciones CRUD sobre las collections que diseñemos para nuestros datos.

Referencias

SQL to MongoDB Mapping Chart - MongoDB Manual 3.4
Operational Segregation in MongoDB Deployments
docs.mongodb.com

SQL to Aggregation Mapping Chart - MongoDB Manual 3.4
The following table presents a quick reference of SQL aggregation statements and the corresponding MongoDB statements…
docs.mongodb.com

Thanks to Ana Betancourt

Comunidad formada por mujeres y hombres generadores de tecnología en busca del conocimiento y la inclusión de la mujer en TI.

Esther Moreno T.

·Mar 22, 2017

El síndrome de la bata blanca

When computing is symbolized as a complex social setting, it becomes a social object and the development and use of computer based services a social act. Kling and Schacechy, 1979 Hace algunos años de manera poco sistemática me han interesado los temas de usabilidad, interacción, diseño centrado en el usuario...

UX

4 min read

Matilde Rocha Aguilera

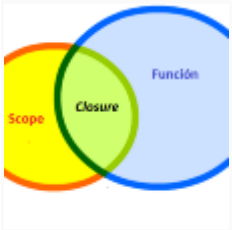
·Mar 14, 2017

Introducción a los “closures” en Javascript

Los “closures” (“cierre” o “clausura” en español) son funciones que se refieren a variables independientes. No te preocupes si esto suena confuso en este momento, a lo largo del artículo cobrará sentido y quedará más claro. Para explicarlo de forma más sencilla, los closures pueden considerarse un tipo especial de...

Closures

4 min read



Renee De La Torre

·Mar 7, 2017

Como elegir el lenguaje de programación adecuado

Si alguna vez te has hecho la pregunta de cuál es el mejor lenguaje que existe, quizás ya te hayas dado cuenta o permíteme decirte en este momento que no existe. Con esto me refiero a que hay diferentes “mejores” lenguajes adecuados para diferentes tipos de proyectos. La pregunta ideal...

Development

4 min read

Ivonne C

·Feb 28, 2017

Multi-arrendamiento

Rentando tu aplicación en la nube — Multi-arrendamiento (multi-tenancy en inglés) es una manera de desarrollar aplicaciones en cloud computing, pero teniendo una sola instancia para múltiples clientes. Para lograr esto es necesario entender algunos conceptos: Cloud computing Existen tres maneras de implementar cloud computing: Infraestructura como un Servicio, IaaS; Plataforma como un Servicio, PaaS; y Software como un...

Multitenancy

7 min read

Yizia

·Feb 21, 2017

Emprendiendo como mujer

Desde las clásicas barreras hasta el autosabotaje — La igualdad de género es un tema prioritario desde hace más de 40 años, aunque tenemos que aceptar que al momento de emprender, siempre se generarán oportunidades de gran impacto para el entorno, independientemente del género que lo provoque. ...

Women In Tech

7 min read

Marcela Sena

206 Followers

Antagonist. I’m a passionate woman in software engineering, interested in diversity and the balance of POO and FP. Love music, sports, and my dog.

Follow

More from Medium

Manikant Jha

SQL vs NoSQL, Which is better?

Pagorn Phusaisakul

Query Array JSON in PostgreSQL

D

Dina Ramadhani

Database Management System Fundamentals

Uditha Ekanayake

NoSQL basics with introduction to MongoDB