Introducción a

Bases de datos y tipos

Las **bases de datos** son recopilaciones organizadas de datos estructurados, con el fin de conectarlos de forma lógica para resolver una problemática. Estas bases de datos no funcionan por cuenta propia, y por lo tanto dependen de un motor de bases de datos. Estos **motores de bases de datos** son herramientas internas que funcionan sobre estas bases, otorgando un número determinado de operaciones, y son los encargados de administrar los datos para que los mismos se mantengan seguros e íntegros. Funcionan como una especie de intermediarios entre las bases y las aplicaciones que las usan.
En la práctica, usamos softwares llamados **gestores de bases de datos** que nos dan una interfaz para crear, administrar e interactuar con las bases de datos. Estos gestores permiten a los usuarios y programadores una forma sistemática de crear, recuperar, actualizar y administrar su información.

Como era de esperarse, existen formas de calificar estas bases de datos. Pero la que nos concierne es categorizarlas entre relacionales y no relacionales:

- Las bases de datos SQL (acrónimo de Structured Query Language), también llamadas bases de datos relacionales están constituidas por un conjunto de tablas en las que los datos están clasificados por categorías y de forma lógica. Cada columna, presentes en estas tablas corresponde a un atributo, bien definido y tipado, el cuál expresa una cualidad del objeto que se guarda en dicha tabla.
  Estas tablas respetan generalmente el mismo esquema fijo, es decir que la forma de la tabla (cantidad de columnas, títulos, tipos de datos y eventualmente otras características) no suele cambiar una vez que su forma fue bien definida. A estas tablas, se les puede asociar con otras tablas que contienen otras informaciones, de aquí viene su calificación de relacionales.
- Por otra parte, las bases de datos NoSQL son, como su nombre lo indica, no relacionales. Estas bases de datos no necesitan un esquema fijo y son fácilmente modulares. El objetivo de este tipo de bases es recuperar los datos de un mismo lugar sin necesidad de pasar por las relaciones entre tablas. Esto lo convierte en herramientas muy rápidas, que tienen la capacidad de manejar grandes cantidades de datos.

Diferencia entre bases SQL y noSQL

Versatilidad (ventaja para noSQL)

- Debido a los constantes cambios que pueden surgir a partir del cliente o el mercado, el esquema flexible de las bases de datos noSQL permite que se puedan adaptar rápidamente sin realizar cambios extremos o complejos.
- SQL por su naturaleza en la cuál todos los objetos van a estar relacionados con los demás, hace que las nuevas adaptaciones requieren una gran análisis, ya que un cambio mal planificado puede generar comportamientos indeseables.

Escalabilidad (ventaja para noSQL):

- Su capacidad de ejecutarse en máquinas de bajo costo trae consigo otra ventaja. Ejecutarse, con pocos recursos en un dispositivo, permite que las bases de datos puedan ejecutarse en muchos dispositivos sin mucho trabajo, promoviendo así la escalabilidad horizontal. Esto permite que las bases noSQL estén mayormente libres para el uso y el consumo, pudiendo distribuir la información en forma de nodos en caso de que se necesite tener más disponibilidad.
- Por otra parte, las bases de datos SQL comienzan a bajar su rendimiento al momento en el cuál empiezan a crecer mucho. Debido a que son sistemas centralizados, la escalabilidad debe realizarse de forma vertical, lo cuál genera costos de implementación.

Costos (ventaja para noSQL)

- Las bases relacionales comerciales requieren un gran costo para mantenimiento y adquisición, ya que para funcionar necesitan una licencia, administradores de bases de datos entrenados, y grandes costos para escalar verticalmente.
- Las bases noSQL tienen la capacidad de correr sobre dispositivos de bajos recursos, convirtiéndolos en la mejor opción en este aspecto.

Atomicidad (ventaja de SQL)

- SQL garantiza que las operaciones se hagan con el objetivo de que cada operación realizada sobre los datos de la base mantengan su integridad. Esto quiere decir que si ocurre un problema a mitad de una operación importante que afecte la información, no se realizarán cambios para mantener el mismo estado anterior a cuando se ejecutó esa operación.
- noSQL no puede garantizar esta condición, e implementar algo similar requiere un poco más de trabajo.

Madurez y estandarización (ventaja de SQL)

- Dado que tiene muchos años en el mercado y aceptación de la comunidad, las bases de datos SQL cuentan con una gran cantidad de información y documentación que permite realizar cualquier operación sin buscar mucho sobre cómo resolver un problema. Creación de tablas y elementos, consultas, y la sintaxis son cosas bien conocidas hoy en día.
- Hay muchas bases noSQL y no muchos estándares a cómo se deberían hacer las cosas, por lo que no terminan siendo muy homogéneas. Esto se complejiza teniendo en cuenta que las opciones noSQL es una alternativa relativamente nueva, por lo

que la documentación y las operaciones que requieren más complejidad suelen ser más escasas.

¿Cuándo usar cada uno entonces?
No hay una respuesta obvia, ya que la colusión va a depender de las circunstancias.

- Cuando los datos necesiten mantenerse íntegros, y que los datos estén relacionados entonces es aconsejable una base SQL. Más teniendo en cuenta que la información no crezca en grandes cantidades, sean más centralizadas, y se tenga sobre todo una definición clara de los objetos que se van a crear.

- Si los requerimientos son cambiantes, y la información no necesita tener una estructura fija es recomendable una base noSQL. Esto trae como beneficio que vamos a poder manejar grandes cantidades de información, y tener en disponibilidad la misma cuanto ésta crezca exponencialmente, sin muchos costos aparentes.

MongoDB

MongoDB es una alternativa a la familia de bases de datos no relacionales, cuyo funcionamiento se basa en guardar los datos en documentos para lectura y escritura. Estos documentos son la forma en la que Mongo va a almacenar la información que esté dentro de nuestra base de datos. Los documentos tienen una forma de organizar y almacenar información con un conjunto de pares clave-valor, de forma similar a lo que sería la estructura de un archivo JSON. Esto permite al mismo tiempo que el esquema que en el cuál presenta la información sea mucho más flexible y permita presentar o no campos, acorde a si estos mismos están o no presentes.

Las colecciones son la forma en la que guardamos esos documentos y que normalmente comparten datos entre sí, de forma que una colección de documentos tiene como objetivo guardarlos en una forma cohesiva y que sean similares entre sí. Para ello, MongoDB almacena documentos en una colección, usualmente con campos comunes entre sí.
Un ejemplo podría ser una colección llamada Usuarios que contengan todos los documentos de los usuarios de nuestra aplicación, junto con su información personal; teniendo en cuenta esto, y para hacer un paralelismo, serían algo similar a las tablas en bases de datos relacionales.

JSON VS BSON
Json, es un formato amigable ya que permite organizar fácilmente la información y es comprensible para las personas con poca experiencia en su uso. Además es fácil de leer con su formato clave y valor, presente en muchas estructuras dentro de lo que es programación. Este formato tiene, sin embargo, un par de desventajas. Principalmente, el hecho de que se base en formato de texto hace que sea muy pesado al agregar más caracteres, por lo que una consulta a una gran fuente de datos puede hacer el sistema en el cuál lo implementamos un poco lento ya que consume mucho espacio. Por otra parte, está limitado a la cantidad de tipos que puede mostrar, ya que soporta numéricos, strings, booleanos y arreglos.

BSON por otra parte representa los datos en forma binaria, lo cuál hace que no consuma tanto espacio como su contraparte, haciendo así de alto rendimiento. A diferencia de JSON, también tiene un repertorio más grande de tipo, soportando otros como diferentes tipos numéricos, data, raw, entre otros. Como desventaja, este formato no es estándar ya que es propio de Mongo, y tampoco va a ser cómodo de leer ya que es propio de la máquina.

Una de las razones por las cuales elegimos MongoDB en vez de otros es por su gran y sólida comunidad en comparación a otros noSQL, la cuál constantemente promueve soluciones a problemas frecuentes, y expanden (mediante extensiones y otros softwares) el funcionamiento de esta base. MongoDB ha podido crear su ecosistema de servicios en donde se pueden ofrecer bases de datos con planes gratis con Mongo Atlas, como también herramientas como Mongo Compass, que sirve como una interfaz visual para ver los datos de nuestra base. También existe una extensión para VSCode para realizar operaciones y conexiones, y Mongo puede ser contenido en Docker.
Existen otras herramientas útiles como el Aggregation Framework que sirve para generar pipelines de agregación de datos, conexiones, y sirve en grandes rasgos para procesar, potencialmente, grandes cantidades de datos.

Ejemplos de dominio:

- Los productos de un ecommerce.
- Las clases de un curso.
- Catálogo de productos.
- Aplicaciones móviles con requisitos cambiantes.
- Redes sociales.

Las bases de datos NoSQL tienen la característica de tener gran facilidad de escalamiento.

- Escalamiento vertical: Se refiere a las características de esa misma máquina donde nuestra base de datos corre, en cuanto a recursos (+RAM, +CPU, +almacenamiento). Este escalamiento con el tiempo termina siendo más costoso.
- Escalamiento horizontal: Tener una máquina (o nodo) y “copiarla” (teniendo ahora varios nodos o máquinas con las mismas características), asegurando alta disponibilidad, sistemas de replicación o tener un conjunto que responda en simultáneo. Esto te libera de la necesidad de escalar de forma vertical. Las bases de datos NoSQL aprovechan muy bien el Escalamiento horizontal ya que podes armar un cluster de información en el que los datos se van a conectar entre si (varios nodos). Esto hace que las bases de datos NoSQL sean más fácilmente escalables o replicables que una SQL. Este escalamiento tiene un costo inicial que se mantiene incluso después de distribuir nodos.

¿Que es Replicación en ese caso? Es una técnica en la que, una vez distribuimos la base de datos en varios nodos (EH) mediante un load balancer, que asigna las peticiones y consultas a cada uno de los nodos en una forma ordenada. Esto permite procesamiento en paralelo y también alta disponibilidad (si una réplica falla, habrá otra disponible). Esto no es posible en el escalamiento vertical.

Información extra

Tipos de Bases NoSQL

- Documentales: Se empareja cada clave con una estructura de datos compleja que se denomina documento, existen otras bases de datos documentales como Cloud Firebase o Couchbase.
- Grafos: Se utilizan para almacenar información sobre redes de datos, como las conexiones sociales. Neo4j es uno de los ejemplos más populares de bases datos de este tipo.
- Clave - Valor: Son las bases de datos NoSQL más simples. Cada elemento de la base de datos se almacena como un nombre de atributo (o “clave”), junto con su valor para acceso de memoria rápida. Un ejemplo de estas bases de datos es Redis.
- Orientada a columnas: Estas bases de datos, como Cassandra (la más popular) o Hbase, permiten realizar consultas en grandes conjuntos y almacenan los datos en columnas, en lugar de filas.