Bases de Datos No-SQL

Programación Orientada a Objetos

2CM1

Humberto Alejandro Ortega Alcocer

Tarea 3

Opción #2 Bases de Datos No-SQL

22 de enero de 2021

Una de las complejidades más grandes en el mundo de los sistemas informáticos es aquella ligada a la información que debemos almacenar, consumir, procesar y resguardar para un correcto funcionamiento. Uno de los pilares del almacenamiento de la información son los archivos, los cuales corresponden a un conjunto de bytes almacenados en un medio de almacenamiento físico como lo puede ser un disco duro, una cinta magnética, un disco compacto, un diskette o medios de almacenamiento basados en celdas de memoria NAND como lo puede ser una memoria USB, un disco SSD o la misma memoria RAM de una computadora. Los archivos cumplen su función de almacenar la información adecuadamente, nos permiten tener índices de posicionamiento y nos extienden, como programadores, la capacidad de almacenar información secuencia y acceder a ella de forma nativa dentro de una computadora.

No es sino hasta que comienzan a surgir y a popularizarse los sistemas de cómputo distribuidos que los archivos comienzan a llegar a sus límites funcionales para las demandas de trabajo y las necesidades que se tienen dentro de un sistema con grandes flujos de información. Es por estas complicaciones que empiezan a surgir en la década de los 70's los primeros Gestores de Bases de Datos, y con ellos el paradigma de la estructuración y características relaciónales de los mismos. En esencia, los programas gestores de bases de datos, emplean archivos en formato binario para almacenar la información pero presentan una interfaz programática (API) capaz de recibir cientos de miles de instrucciones de consulta de forma secuencial y en paralelo apoyándose en un motor relacional interno para poder establecer "relaciones" entre los datos de distintos tipos. Estas relaciones, nos permiten obtener información sin importar si los datos se encuentran en dos o más archivos distintos, en realidad el concepto de archivo comienza a ser irrelevante pues, dado que los gestores modernos de bases de datos abstraen la lógica detrás

del almacenamiento final de la información, como usuarios únicamente nos es relevante el concepto de una tabla y los datos que hay en ella.
Con el paso del tiempo, los gestores de bases de datos relacionales se posicionaron como la forma predilecta para el desarrollo de sistemas informáticos. El lenguaje SQL (Lenguaje Estructurado de Consulta) fue aceptado por la gran mayoría y, salvo ligeras variaciones para los tipos de datos o funciones particulares, se volvió un estándar con el cual podemos acceder a la información almacenada por estos programas de forma rápida y eficiente.
Si bien, con SQL uno puede desarrollar *casi* cualquier aplicación que se pueda uno imaginar, existen casos de uso dónde la estructuración y relación de los datos puede resultar ineficiente e incompatible con el caso de uso que pudiéramos tener. Una de las situaciones dónde las bases de datos relacionales comenzaron a mostrar sus limitantes, fue en el almacenamiento de *Big Data*, en dónde se tratan de ingestar al sistema millones de datos por hora, por minuto e inclusive por segundo, y las bases de datos relacionales convencionales pueden sufrir serios problemas de escalabilidad si debemos añadir y eliminar índices constantemente, modificando la información en la mismas, o si, simplemente, cada consulta de información que realiza nuestro sistema final depende de la carga de múltiples relaciones entre datos para poder arrojar una agregación adecuada.
Estas complejidades dieron paso a un esquema de modelado de información que no cumple con los principios de SQL en dónde, para que la información pueda ser ligada apropiadamente en una consulta, se deben establecer las relaciones previamente por lo que los desarrolladores deben tener un total conocimiento y control sobre los distintos datos que pueden entrar al sistema así como de el debido manejo de la relación de los datos para poder obtener los beneficios de las misma. En un contexto dónde se requieren añadir nuevos datos, o modificar los ya existentes dentro una base de datos, cuando se tienen altos volúmenes de

información las tareas para ajustar la estructura de la base de datos puede resultar en labores que pueden tomar días o meses en completarse apropiadamente. Para evitar estos problemas, existen bases de datos llamadas “No-SQL” o “No Relacionales” dónde la información se almacena de forma unitaria, es decir, todos los datos que puedan estar relacionados con alguna entidad se almacenan en dicha entidad directamente. Al no estructurar la información, existen riesgos de cohesión en cuanto a la información almacenada, pero esto es un beneficio a su vez ya que en caso de que se requiera añadir nueva información a alguna entidad no habrá problema alguno en añadirla.

Este tipo de bases de datos son excelentes para tareas con millones de datos ya que nos permiten *ingestar* los datos rápidamente al sistema y comenzar a consumirlos sin requerir establecer las relaciones entre los mismos. Otro beneficio es en el tiempo de respuesta ya que, a pesar de que los gestores de bases de datos modernos como MySQL, PostgreSQL, Microsoft SQL, entre otros, son extraordinariamente eficientes, consultas dónde se requieran ligar millones de datos entre sí para realizar comparaciones y ofrecer como resultado una agregación de varias tablas con varias columnas puede resultar un proceso costoso sin importar qué tan eficiente sea el gestor en si. En un esquema no relacional este tipo de operaciones se vuelven mucho menos costosas en procesamiento y memoria ya que la información está contenida en una sola entidad, por lo que con tan solo una operación de lectura el gestor es capaz de obtener todos los datos que pueda necesitar.

Otro beneficio al emplear bases de datos no relacionales es en la fase de prototipado de un proyecto o empresa ya que, a pesar de que la robustez en la estructuración de la información representa un beneficio para la interacción programática con la misma, durante esta etapa puede no ser muy clara la

información a almacenar o puede ser muy variante en cuanto a los campos que pueden ser relevantes para el desarrollo del proyecto.

Durante sus inicios, el paradigma de bases de datos no relacionales no fue bien recibido por la industria ni el gremio de la programación, inclusive fue motivo de burlas y comentarios minimizando sus objetivos. Sin embargo, grandes empresas como MongoDB han salido a delante y demostrado en el mundo que su modelo de almacenamiento de información tiene grandes beneficios y múltiples casos de uso.

El hecho de que su almacenamiento sea fundamentalmente la idea de "*de-normalizar*" la información que se tiene, permite que grandes volúmenes de información puedan ser procesados rápidamente, lo cual es una gran ventaja en nuestro mundo moderno globalizado dónde la instantaneidad de los análisis computacionales con grandes volúmenes de información son el camino para ofrecer a los usuarios experiencias personalizadas y de valor. Existen casos de uso para todo, y es de sabios saber dónde conviene y dónde no conviene utilizar cierta tecnología o procedimiento para la resolución de algún problema.

Y no es que en una base de datos no relacional no se pueda realizar una agregación entre distintos "esquemas" (tablas), sino que la relación no se considera *dura* en el sentido de que el propio gestor de bases de datos se asegure de mantener una consistencia en la misma. Uno puede realizar una relación tan simple como decir, este examen pertenece al alumno Humberto, pero el gestor no tratará de verificar que dicho alumno exista o que el alumno en cuestión tenga ciertas características previas para realizar la operación, simplemente almacena el dato y cuando se desea consumir el programador deberá manejar esas complicaciones desde su código para prevenir errores.

Es un intercambio entre complejidad para sistemas donde los datos evolucionan a través del tiempo y eficiencia para consumir dichos datos. Inclusive, existen casos documentados dónde grandes entidades financieras han optado por migrar a

esquemas no relacionales para almacenar toda su información operativa. Tal es el caso de HSBC que en el 2019 decidió migrar todas sus múltiples bases de datos a una sola base de datos no relacional dónde almacenan toda su información. Este paso, aunque criticado por muchos, es un paso natural en las instituciones financieras para poder centralizar su información y permitir la realización de análisis más complejos de forma mucho más rápida y eficiente que lo que jamás se podría realizar con una base de datos relacional.

Si bien, las bases de datos relacionales siempre serán nuestra primer opción como desarrolladores para almacenar información y consumirla, las bases de datos no relacionales llegaron para quedarse y son las que están potenciando la innovación tecnológica en áreas de grandes volúmenes de información como en el sector financiero o en el internet de las cosas.

Dentro de los distintos modelados de información disponible, podemos asegurarnos que las bases de datos no relacionales nos ofrecen la flexibilidad que requerimos para crecer exponencialmente la información que almacenamos, ofreciéndonos de forma rápida, un modelo ideal para plantear las soluciones tecnológicas del futuro, hoy.

Referencias

- Amazon Web Services. (n.d.). *Databases on AWS: The Right Tool for the Right Job*. Amazon Web Services, Inc. Retrieved January 16, 2021, from https://aws.amazon.com/nosql/
- colaboradores de Wikipedia. (2021, January 6). *NoSQL*. Wikipedia, La Enciclopedia Libre. https://es.wikipedia.org/wiki/NoSQL
- Microsoft Azure. (n.d.). *Base de datos NoSQL: ¿qué es NoSQL? | Microsoft Azure*. Retrieved January 16, 2021, from https://azure.microsoft.com/es-mx/overview/nosql-database/
- MongoDB. (n.d.). *What is NoSQL? NoSQL Databases Explained*. Retrieved January 16, 2021, from https://www.mongodb.com/nosql-explained