Ensayo sobre las bases de datos orientadas a objetos

Autor: Luis Cabrera Benito
Página web: https://parzibyte.me/blog

Las bases de datos son una pieza fundamental en todos los lugares en donde se implementan sistemas de información. Desde empresas gigantes como Apple, Facebook y Google, hasta en pequeñas y medianas empresas.

Cada uno de nosotros hace uso de ellas sin darnos cuenta: cuando compramos algo en el supermercado, haciendo una compra digital con una tarjeta de crédito, reservando un viaje, visitando una biblioteca, rentando una película, usando el internet e incluso estudiando en una universidad. Han llegado para quedarse y estarán aquí por un largo tiempo, pues la información es y será una pieza fundamental en cualquier área.

Debido a que cada día el tamaño de la información crece, los administradores de bases de datos (así como los desarrolladores y expertos en la materia) buscan la forma de crear sistemas más rápidos, confiables, optimizados y sobre todo fáciles de usar. Así es como las bases de datos orientadas a objetos nacen, por la necesidad de la mejora continua en la informática. Si bien la mayor parte del mundo utiliza bases de datos relacionales o bases de datos NoSQL, la orientación a objetos es un tema que promete mucho para el mundo digital.

En el inicio de la revolución informática, los datos eran guardados en ficheros de texto o archivos binarios en donde los datos eran guardados con una velocidad aceptable. El problema vino más tarde, cuando había miles o millones de registros que disminuían el rendimiento.

No fue hasta el año 1970 cuando Edgar Frank Codd (un investigador de IBM) publicó su trabajo llamado *Un modelo relacional de datos para grandes bancos de datos compartidos* (el título original en inglés es *A Relational Model of Data for Large Shared Data Banks*) en el cual especificaba un modelo relacional para la administración de bases de datos. En el documento, Edgar incluía entre otras cosas una especificación, formas normales y además un lenguaje llamado *SEQUEL*, que más tarde se convertiría en lo que conocemos hoy como *SQL*. Más tarde nacieron algunos, nuevos paradigmas (p. ej. NoSQL) y características ACID. Así fue como se definió de manera formal a una base de datos

Una *base de datos* es un repositorio para datos; en otras palabras, se pueden almacenar montones de información en ellas. Por otro lado, una *base de datos relacional* es una base de datos especial que utiliza estructuras llamadas *tablas*, las cuales son enlazadas creando así lo que conocemos como *relaciones*.

Las ventajas que ofrecen estas bases de datos son que remueven la duplicidad de la información aplicando una técnica llamada normalización.

Los lenguajes de programación también evolucionaron. En ese punto recordamos primero al lenguaje ensamblador, en donde se tenía que programar dependiendo de las instrucciones que trajera incorporadas el procesador, ya fuera de Intel o de AMD (los primeros fabricantes).

Más tarde llegó el lenguaje *C* (claramente entre el ensamblador y él hubo muchos más) y su evolución: el lenguaje de programación *C*++. Aunque *Simula* fue el primer lenguaje de programación orientado a objetos, la popularidad de este paradigma se debe a su uso en los lenguajes de programación *Java*, *C*++ o *C#*. La programación orientada a objetos es un nuevo paradigma de programación que promueve la reutilización de código separando los conceptos en clases, a partir de las cuales se crean objetos. Los objetos se comunican con otros objetos a través de llamadas a funciones. Estos objetos también tienen propiedades que pueden definir por sí mismos o heredarlos de otros, haciendo uso de la herencia.

En este punto podemos observar la gran ventaja que tienen las bases de datos y la programación orientada a objetos. Si combinamos estas tecnologías podríamos crear un sistema gestor de bases de datos con las ventajas de: aislamiento, consistencia, durabilidad y atomicidad; además de guardar directamente objetos (y recuperarlos de igual manera) con las características de la POO como lo son el encapsulamiento, la herencia, el polimorfismo y la abstracción.

Una base de datos orientada a objetos (también llamada OODB por sus siglas en inglés) es manejada por un motor de bases de datos orientadas a objetos (llamado OODBMS) y el mismo debe ser capaz de proveer las características esenciales de una base de datos, identidad de objetos, encapsulamiento y objetos con un estado complejo. Todo ello a un nivel alto, facilitando su uso, pero siendo optimizado en los niveles más bajos.

En un escenario ideal, una base de datos orientada a objetos debería ser capaz de serializar los objetos para guardarlos ocupando el menor espacio posible, y hacer el proceso inverso al leerlos; cuidando que la operación realice el mínimo número de escrituras en memoria.

Un punto importante es que estas bases de datos deben tener las características que ya todos conocemos, como son la optimización de búsquedas a través de índices, gatillos, trabajos, copias de seguridad y restauración, vistas, procedimientos almacenados (aunque bien estos podrían ser hechos desde el lenguaje de programación), paginación (para limitar los resultados), agrupación y ordenamiento; sin mencionar cosas como la concurrencia o la replicación bidireccional.

Además, ni el programador ni el administrador deberían preocuparse por crear relaciones explícitas: cualquier clase podría heredar de otra clase, incluso de otras clases.
Mencionando otros aspectos, cada clase debería ser reutilizable y mantener el principio de responsabilidad única, el cual dice que cada clase debería encargarse de todo lo que tiene que ver con ella sin depender de alguien más.

No olvidemos los *getters* y los *setters*, ya que en las bases de datos relacionales seleccionamos las columnas que necesitamos, ahorrando almacenamiento, y en las bases de datos orientadas a objetos esto podría hacerse utilizando los métodos mencionados anteriormente para obtener únicamente algunas columnas. Algo muy importante es que un objeto (el cual se crea a través de una clase) no contiene únicamente propiedades, también tiene métodos que transforman a dichas propiedades. Hablando sobre objetos y miembros de los mismos, una base de datos de este tipo debería ser capaz de proteger a los miembros que le pertenecen dependiendo del clasificador, ya sea público o privado.

Lo más importante de todo esto es la seguridad, y para terminar pronto, un sistema gestor de este tipo debe tener una alta seguridad con usuarios y permisos de acceso. En niveles más bajos debe estar protegido contra desbordamientos de búferes, escalamiento de privilegios o accesos ilegales a la memoria.

Una ventaja que ofrecerían estos motores sería que ya no habría inyecciones SQL (tal vez habría incluso vulnerabilidades más peligrosas, pero ya no inyecciones) porque las consultas no serían simples concatenaciones, se utilizarían a los miembros de los objetos y se guardarían directamente sin ser procesados.

Desarrollar un motor de base de datos que sea capaz de abstraer objetos en lugar de relaciones es una tarea alcanzable, pero no por ello es fácil. Haciendo una comparación, una base de datos relacional puede traer datos en simples arreglos o vectores que fácilmente se pueden traducir a la mayoría de los lenguajes de programación. Sin embargo, la implementación de la orientación a objetos (aunque es un modelo para todos) dentro de las bases de datos cambiaría dependiendo del lenguaje que las consuman; tal vez un motor sería compatible únicamente con uno o pocos lenguajes.

Finalmente, en caso de conseguirlo debemos luchar con una cosa que todo administrador y programador debe tener en cuenta: el rendimiento. Como sabemos, un objeto es cargado en memoria cada que se consulta y debe crear relaciones en tiempo de ejecución, lo que disminuiría el tiempo de respuesta debido a la escritura en memoria. Todo ello hace que la orientación a objetos en las bases de datos sea un reto tanto de diseño como de optimización y rendimiento. Esperemos que, con la evolución de los microprocesadores, el estudio de la

computación y el aumento de velocidad en las redes algún día sea posible crear un estándar que nos permita abstraer la orientación a objetos dentro de una base de datos sin mucho esfuerzo.

Bibliografía

Connolly, T. &. (2005). *Database Systems.* Estados Unidos: Addison-Wesley.

Powell, G. (2006). *Beginning Database Design.* Canadá: Wiley Publishing.

Universidad técnica del norte. (1 de Enero de 2017). *Universidad técnica del norte: Pilares | Fundamentos de la Programación Orientada a Objetos*. Obtenido de http://www.utn.edu.ec/reduca/programacion/poo/pilares.html