OR/M (Object Relational Mapping)

Robles Flores, Anthony Richard (2016056192), Porlles Carrillo, Diego Armando (2015050948), Sandoval Blas, Jesus Enrique (2016054467), Quispe Mamani, José Luis (2015053235)

Tacna, Perú

Abstract

Developing applications following the object-oriented paradigm makes programmers face, among other challenges, the problem of data persistence, due to differences between the relational model and objectoriented model. This problem most of the cases is solved with the help of certain tools that handle automatically data access generation, abstracting the programmer from this problem.

1. Resumen

Al desarrollar una aplicación siguiendo el paradigma orientado a objetos, los programadores se enfrentan, entre otros desafíos, al problema de la persistencia de los datos, debido a las diferencias que existen entre el modelo relacional y el modelo orientado a objetos. Este problema la mayoría de los casos es solucionado con la ayuda de ciertas herramientas que se encargan de generar de manera automática el acceso a datos, abstrayendo al programador de este problema.

2. Introducción

El Modelo Relacional es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos para la gestión de una base de datos. Siguiendo este modelo se puede construir una base de datos relacional que no es más que un conjunto de una o más tablas estructuradas en registros (filas) y campos (columnas), que se vinculan entre sí por un campo en común. Sin embargo, en el Modelo Orientado a Objetos en una única entidad denominada objeto, se combinan las estructuras de datos con sus comportamientos.

En este modelo se destacan conceptos básicos tales como objetos, clases y herencia.

Entre estos dos modelos existe una brecha denominada desajuste por impedancia dada por las diferencias entre uno y otro. Una de las diferencias se debe a que en los sistemas de bases de datos relacionales, los datos siempre se manejan en forma de tablas, formadas por un conjunto de filas o tuplas; mientras que en los entornos orientados a objetos los datos son manipulados como objetos, formados a su vez por objetos y tipos elementales. La gran mayoría de los lenguajes de programación como java o C, tienen un modelo de manejo de datos basado en leer, escribir o modificar registros de uno en uno. Por ello, cuando se invoca el lenguaje de consulta SQL (Standard Query Language) desde un lenguaje de programación es necesario un mecanismo de vinculación que permita recorrer las filas de una consulta a la base de datos y acceder de forma individual a cada una de ellas.[1]

Además en el modelo relacional no se puede modelar la herencia que aparece en el modelo orientado a objetos y existen también desajustes en los tipos de datos, ya que los tipos y denotaciones de tipos asumidos por las consultas y lenguajes de programación difieren. Esto concierne a tipos atómicos como integer, real, boolean, etc. La representación de tipos atómicos en lenguajes de programación y en bases de datos pueden ser significativamente diferentes, incluso si los tipos son denotados por la misma palabra reservada, ej.: integer. Esto ocurre también con tipos complejos como las tablas, un tipo de datos básico en SQL ausente en los lenguajes de programación.

Para atenuar los efectos del desajuste por impedancia entre ambos modelos existen varias técnicas y prácticas como los Objetos de Acceso a Datos (Data Acces Objects o DAOs), marcos de trabajo de persistencia (Persistence Frameworks), mapeadores Objeto/Relacionales (Object/Relational Mappers u ORM), consultas nativas (Native Queries), lenguajes integrados como PL-SQL de Oracle y T-SQL de SQL Server; mediadores, repositorios virtuales y bases de datos orientadas a objetos.

Las soluciones al problema de la impedancia mencionadas anteriormente presentan ventajas y desventajas que deberán ser evaluadas según las características y requisitos del sistema a desarrollar. En el presente artículo se propone el uso de las herramientas de mapeo Objeto/Relacional como una

alternativa a este problema.

3. Marco Teórico

3.1. Definición

El mapeo objeto-relacional es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos al utilizado en una base de datos relacional. En la práctica esto crea una base de datos virtual orientada a objetos sobre la base de datos Osmel Yanes Enriquez, Hansel Gracia del Busto 3 Revista Telem@tica. Vol. 10. No. 3, septiembre-diciembre, 2011. ISSN 1729-3804 relacional. Esto posibilita el uso de las características propias de la orientación a objetos (esencialmente la herencia y el polimorfismo). Las bases de datos relacionales solo permiten guardar tipos de datos primitivos (enteros, cadenas de texto, etc.) por lo que no se pueden guardar de forma directa los objetos de la aplicación en las tablas, sino que estos se deben de convertir antes en registros, que por lo general afectan a varias tablas. En el momento de volver a recuperar los datos, hay que hacer el proceso contrario, se deben convertir los registros en objetos. Es entonces cuando ORM cobra importancia, ya que se encarga de forma automática de convertir los objetos en registros y viceversa, simulando así tener una base de datos orientada a objetos [2]. Entre las ventajas que ofrecen los ORM se encuentran: rapidez en el desarrollo, abstracción de la base de datos, reutilización, seguridad, mantenimiento del código, lenguaje propio para realizar las consultas. No obstante los ORM traen consigo algunas desventajas como el tiempo invertido en el aprendizaje. Este tipo de herramientas suelen ser complejas por lo que su correcta utilización requiere un espacio de tiempo a emplear en conocer su funcionamiento adecuado para posteriormente aprovechar todo el partido que se le puede sacar. Otra desventaja es que las aplicaciones suelen ser algo más lentas. Esto es debido a que todas las consultas que se hagan sobre la base de datos, el sistema primero deberá transformarlas al lenguaje propio de la herramienta, luego leer los registros y por último crear los objetos. En el mapeo objeto-relacional encontramos el uso de algunos patrones de diseño como el Repository y el Active Record. [3]

3.2. ¿Qué es Agile?

La metodología ágil es también una metodología de desarrollo de software que surgió alrededor del año 2001, cuando se presentó el manifiesto ágil.

Emplea cuatro valores y doce principios que ayudan a construir una cultura de desarrollo de software "ágil".

En términos generales, ágil fomenta la adopción y una mentalidad de liderazgo que promueve el trabajo en equipo, la autoorganización y la responsabilidad. Más importante aún, el enfoque ágil se enfoca más en alinear continuamente el desarrollo con las necesidades y tendencias del cliente, incluso cuando esas necesidades y tendencias cambian al final del proceso de desarrollo.

Agile incorpora un conjunto de principios que ayudan a individuos, equipos y unidades más grandes a trabajar juntos. La "mentalidad ágil"se enfoca más en las personas que en los procesos y herramientas. Una organización ágil se adapta y aprende sobre el cambio constante que les permite identificar nuevas oportunidades y añadir más valor para los clientes. "SQL-92.º "SQL2".[1]

3.3. Requisitos para implementar una cultura DevOps

Lo primero es comunicar que la cultura DevOps se va a implementar, mencionando los beneficios y haciendo que el equipo se sienta parte del cambio. Luego explicar las acciones que se van a tomar, no importa que no sean desarrolladores.

La comunicación y colaboración para tener una cultura DevOps son cruciales, es un trabajo entre los desarrolladores y los equipos encargados de la infraestructura de los servidores.

La tarea principal será la automatización, reducir los tiempos de despliegue del producto y mantener una calidad de desarrollo y estabilidad para el usuario. Debes tener claro que esto será un proceso sin fin, por lo que cada mejora te dará el tiempo que dedicarás para innovar el proceso y hacerlo cada vez mejor.[3]

3.4. Patrón Repository

AGREGAR CONTENIDO [5]

3.5. Patrón Active Record

AGREGAR CONTENIDO.[5]

3.6. SubSonic y NHibernate

AGREGAR CONTENIDO[2]
AGREGAR CONTENIDO.[3]

3.7. ALGO ADICIONAL

AGREGAR CONTENIDO.[4]

4. Analisis

AGREGAR CONTENIDO

5. Conclusiones

5.1. Conclusión

AGREGAR CONTENIDO

Referencias

[1] 2008, V. S. (ne). The microsoft developer network. Recuperado de http://cort.as/-QSJ3. Accedido 18-08-2019.

[2] Cristina, J. R. (ne). Qué es devops 2015. Recuperado de http://cort.as/-P7oQ . Accedido 28-08-2019.

[3] Guardado, I. (2010). Introducción a object-relational mapping. Recuperado de http://cort.as/-QSHx. Accedido 18-08-2019.

[4] IntelDig (ne). ¿por qué se usa devops?,¿cuándo no adoptar devops?,principios de devops. Recuperado de http://cort.as/-P7Ne. Accedido 31-08-2019.

[5] Pariseau, B. (ne). Guía esencial: Las bases de datos dan soporte a las tendencias de ti 2017. Recuperado de http://cort.as/-P7o9 . Accedido 28-08-2019.