UNIVERSIDAD PRIVADA DE TACNA

INGENIERIA DE SISTEMAS

TEMA:

Mapeo Objeto Relacional

CURSO:

BASE DE DATOS II

DOCENTE(ING):

Patrick Jose Cuadros Quiroga

Integrantes:

Marko Antonio RIVAS RIOS	(2016055461)
Jorge Luis MAMANI MAQUERA	(2016055236)
Andree Ludwed VELASCO SUCAPUCA	(2016055286)
Yofer Nain CATARI CABRERA	(2017059289)
Adnner Sleyder ESPERILLA RUIZ	(2015050543)
Jesus ESCALANTE ALANOCA	(2015050641)

Índice

1. abstract

marko

2. Introducción

ORM es el mapeo objeto-relacional (más conocido por su nombre en inglés, Object- Relational mapping), consiste en una técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional utilizado en el desarrollo de aplicaciones. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo). Hay paquetes comerciales y de uso libre disponibles que desarrollan el mapeo relacional de objetos, aunque algunos programadores prefieren crear sus propias herramientas ORM.

Entre estos paquetes comerciales tenemos una lista alfabética de los principales motores de mapeo objeto relacional, tales como: ColdFusion, Common Lisp, Java, JavaScript, .NET, Perl, PHP, Python, Ruby, Smalltalk, C++.

El problema que surge, porque hoy en día prácticamente todas las aplicaciones están diseñadas para usar la Programación Orientación a Objetos (POO), mientras que las bases de datos más extendidas son del tipo relacional y estas solo permiten guardar tipos de datos primitivos (enteros, cadenas de texto) por lo que no se puede 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.

En este punto es que se muestra la importancia del ORM, ya que este se encarga de forma automática, de convertir los objetos en registros y viceversa, simulando así tener una base de datos orientada a objetos.

3. Objetivos

El desarrollo de software busca mejorar la productividad de las empresas por medio de la automatización y el uso de herramientas, las empresas que se dedican al desarrollo de software tienen como objetivo ayudar a otras empresas en la automatización y desarrollo de herramientas para este fin, pero muchas veces las mismas olvidan su propia productividad. Las Herramientas ORM (Object Relational Mapping) se han ideado con este fin, al evitar repetir muchas líneas de programación en la capa de abstracción, pero es necesario considerar que cada una tiene su estándar o su propio lenguaje por lo cual se requeriría tiempo adicional para ocuparlas.

El desarrollo de este ORM tiene como objetivo proveer una herramienta acorde a las necesidades y el estándar de programación de la empresa Interfaces, logrando que durante el desarrollo de software, este se centre en otros aspectos más relevantes, como el diseño de la base de datos, interfaz de usuario y la capa de negocios, olvidándonos casi por completo de la capa de abstracción. Con el uso de la herramienta ORM se logró reducir el tiempo de desarrollo de la aplicación de prueba.

Realizar el modelo y modo de implementación del ORM en un java-object.
Implementar la recuperación de datos automáticamente de los java-objects.
Implementar la conectividad y las consultas autogeneradas a la Base de Datos a partir de los modelos heredados del modelo base del ORM.

Implementar la conversión automática de los resultados a los modelos java-object.
Realizar pruebas de validación al ORM.

4. Marco Teorico

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.

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

Además en el modelorelacional 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

Mapeo Objeto/Relacional

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 relacional. Esto posibilita el uso de las características propias de la orientación a objetos (esencialmente la herencia y el polimorfismo).

5. Análisis (apreciaciones)

Comencemos analizando JDBC(API de Java)como solución. El mayor beneficio es el rendimiento dado que programado adecuadamente, con store procedures y al no tener capas intermedias que pasar, el rendimiento aumenta a sacrificio de muchas características que son de suma importancia en una arquitectura empresarial.
JDBC es una API de bajo nivel y sus prestaciones son para funciones de bajo nivel.
Cuando el modelo de datos es simple la solución más fácil es JDBC
Los frameworks ORM ya están consolidados, probados en el mercado y existen muchas herramientas para desarrollar con estos frameworks. La mayoría de ellos permiten la definición de las entidades generalmente a través de ficheros XML o anotaciones. A partir de esa definición los ORM son capaces de extraer la definición de esquema de la base de datos necesaria para representar ese modelo de objetos. Los ORM permiten el mapeo de estos esquemas de base de datos a objetos de modo que a partir de las tablas de base de datos se pueden generar las clases Java, necesarias para modelar dicho esquema con todas las relaciones de jerarquía que estén presentes en el mismo.
Obviamente esto supone una ventaja grandísima ya que la cantidad de código necesaria para realizar todas esas funciones es realmente considerable

6. Conclusiones

El uso de un ORM es una alternativa sumamente efectiva a la hora de trasladar el modelo conceptual (orientado a objetos) al esquema relacional nativo de las bases de datos SQL. Evita la inclusión de sentencias SQL embebidas en el código de la aplicación, lo que a su vez facilita la migración hacia otro sistema gestor de bases de datos. Incorpora una capa de abstracción entre el modelo relacional físico y la capa de negocios de la aplicación. Al ser realizado, en esta capa, de manera automática la conversión de instrucciones orientadas a objetos, a sentencias SQL, minimiza la ocurrencia de errores humanos.

De cualquier modo utilizar un ORM no debe ser considerado una panacea, sino que debe usarse a discreción; teniendo en cuenta las particularidades de cada problema a modelar. En determinados casos no es recomendable el uso de un ORM, sobre todo cuando se imponen tiempos de respuesta mínimos o se requiere una menor sobrecarga. En estos casos lo más conveniente es el uso de un microORM; evitando siempre que sea posible las inyecciones de SQL Inline.

Lo anteriormente expuesto libera a los desarrolladores de aplicaciones de la responsabilidad de conocer las múltiples variantes de SQL que existen en función del gestor de bases de datos que se utilice. No obstante, en escenarios en que se necesite hacer un uso más eficiente del sistema de almacenamiento de información, personalizado de acuerdo a las necesidades de la aplicación, y se escoja como gestor de bases de datos una variante NoSQL no es necesaria la utilización de un ORM.

En resumen, en dependencia del gestor de bases de datos a emplear, se recomienda siempre que sea posible y sea factible la utilización de un ORM

7. Bibliografía

http://revistatelematica.cujae.edu.cu/index.php/tele/article/view/23/21