AUTORES: Lesly Ordenes Campos /Jorge Bustamante Valdés

MAIL: Lesly.ordenes@pat-traffic.cl / jorge_bv@tie.cl

ORM (Object Relational Mapping)

RESUMEN

El **ORM** (**Object-Relational mapping**) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia, en otras palabras nos permite vincular los objetos usados en nuestro modelo de la aplicación con la base de datos relacional. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto, da la posibilidad de usar características propias de la orientación a objetos (como por ejemplo, herencia y polimorfismo). El principal problema surge hoy en día, prácticamente todas las aplicaciones están diseñadas para usar la Orientación a Objetos (POO), mientras que las bases de datos más extendidas son del tipo relacional. Hay paquetes comerciales y de uso libre disponibles que desarrollan el mapeo relacional de objetos, aunque algunos programadores prefieren crear sus propias herramientas ORM. Uno de los componentes ORM más utilizado sino el más ampliamente es el Hibernate, surgido del ambiente Java y llevado al uso del framework .NET con la versión Nhibernate.

INTRODUCCION

El presente trabajo tiene como objetivo mostrar de forma resumida qué es y cuál es la funcionalidad de los ORM. Se explicará cómo se debe realizar su implementación y tener claro los diferentes Frameworks en los cuales se pueden implementar. Además de mostrar también sus desventajas.

DESARROLLO

En la programación orientada a objetos, las tareas de gestión de datos son implementadas generalmente por la manipulación de objetos, los cuales son casi siempre valores no escalares. Para ilustrarlo, considere el ejemplo de una entrada en una libreta de direcciones, que representa a una sola persona con cero o más números telefónicos y cero o más direcciones. En una implementación orientada a objetos, esto puede ser modelado por un "objeto persona" con "campos" que almacenan los datos de dicha entrada: el nombre de la persona, una lista de números telefónicos y una lista de direcciones. La lista de números telefónicos estaría compuesta por "objetos de números telefónicos" y así sucesivamente. La entrada de la libreta de direcciones es tratada como un valor único por el lenguaje de programación (puede ser referenciada por una sola variable, por ejemplo). Se pueden asociar varios métodos al objeto, como uno que devuelva el número telefónico preferido, la dirección de su casa, etc.

Sin embargo, muchos productos populares de base de datos, como los Sistemas de Gestión de Bases de Datos SQL, solamente pueden almacenar y manipular valores escalares como enteros y cadenas, organizados en tablas normalizadas. El programador debe convertir los valores de los objetos en grupos de valores simples para almacenarlos en la base de datos (y volverlos a convertir luego de recuperarlos de la base de datos), o usar sólo valores

escalares simples en el programa. El mapeo objeto-relacional es utilizado para implementar la primera aproximación.

El núcleo del problema reside en traducir estos objetos a formas que puedan ser almacenadas en la base de datos para recuperarlas fácilmente, mientras se preservan las propiedades de los objetos y sus relaciones; estos objetos se dice entonces que son persistentes.

IMPLEMENTACIONES

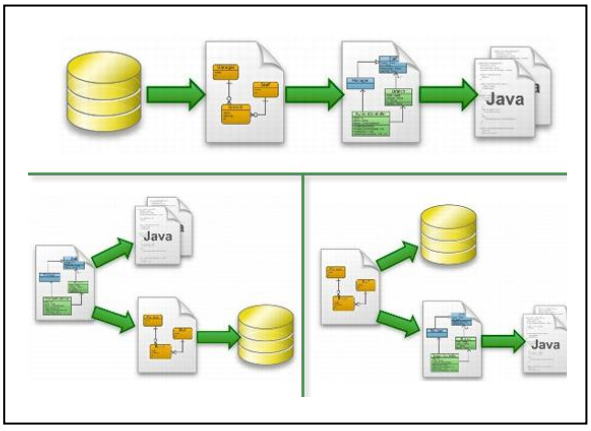
Los tipos de bases de datos usados mayormente son las bases de datos SQL, cuya aparición precedió al crecimiento de la programación orientada a objetos en los 1990s. Las bases de datos SQL usan una serie de tablas para organizar datos. Los datos en distintas tablas están asociados a través del uso de restricciones declarativas en lugar de punteros o enlaces explícitos. Los mismos datos que pueden almacenarse en un solo objeto podrían requerir ser almacenados a través de varias tablas.

Una implementación del mapeo relacional de objetos podría necesitar elegir de manera sistemática y predictiva qué tablas usar y generar las sentencias SQL necesarias.

Muchos paquetes han sido desarrollados para reducir el tedioso proceso de desarrollo de sistemas de mapeo relacional de objetos proveyendo bibliotecas de clases que son capaces de realizar mapeos automáticamente. Dada una lista de tablas en la base de datos, y objetos en el programa, ellos pueden automáticamente mapear solicitudes de un sentido a otro. Preguntar a un objeto persona por sus números telefónicos resultará en la creación y envío de la consulta apropiada, y los resultados son traducidos directamente en objetos de números telefónicos dentro del programa.

Desde el punto de vista de un programador, el sistema debe lucir como un almacén de objetos persistentes. Uno puede crear objetos y trabajar normalmente con ellos, los cambios que sufran terminarán siendo reflejados en la base de datos.

Sin embargo, en la práctica no es tan simple. Todos los sistemas ORM tienden a hacerse visibles en varias formas, reduciendo en cierto grado la capacidad de ignorar la base de datos. Peor aún, la capa de traducción puede ser lenta e ineficiente (comparada en términos de las sentencias SQL que escribe), provocando que el programa sea más lento y utilice más memoria que el código "escrito a mano".



Un buen número de sistemas de mapeo objeto-relacional se han desarrollado a lo largo de los años, pero su efectividad en el mercado ha sido diversa. NeXT's Enterprise Objects Framework (EOF) fue una de las primeras implementaciones, pero no tuvo éxito debido a que estaba estrechamente ligado a todo el kit de NeXT's, OpenStep .Fue integrado más tarde en NeXT's WebObjects, el primer servidor web de aplicaciones orientado a objetos. Desde que Apple compró NeXT's en 1997, EOF proveyó la tecnología detrás de los sitios web de comercio electrónico de Apple: los servicios .Mac y la tienda de música iTunes. Apple provee EOF en dos implementaciones: la implementación en Objective-C que viene con Apple Developers Tools y la implementación Pure Java que viene en WebObjects 5.2. Inspirado por EOF es el open source Apache Cayenne. Cayenne tiene metas similares a las de EOF e intenta estar acorde a los estándares JPA.

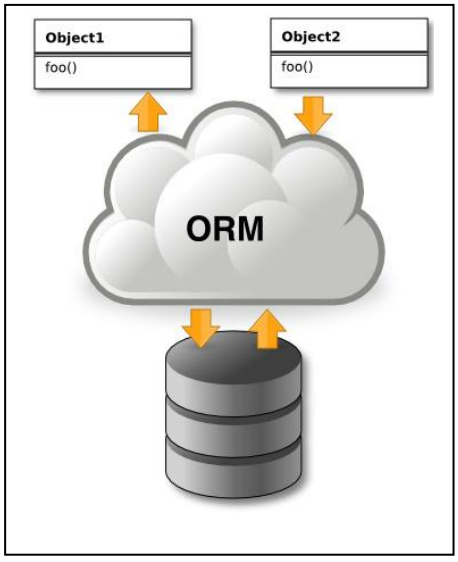
Una aproximación alternativa ha sido tomada por tecnologías como RDF y SPARQL, y el concepto de "triplestore". RDF es una serialización del concepto objeto-sujeto-predicado, RDF/XML es una representación en XML de aquello, SPARQL es un lenguaje de consulta similar al SQL, y un "triplestore" es una descripción general de una base de datos que trabaja con un tercer componente.

Más recientemente, un sistema similar ha comenzado a evolucionar en el mundo Java, conocido como Java Data Objects (JDO). A diferencia de EOF, JDO es un estándar, y muchas implementaciones están disponibles por parte de distintos distribuidores de software. La especificación 3.0 de Enterprise Java Beans (EJB) también cubre la misma área. Han existido algunos conflictos de estándares entre ambas especificaciones en términos de preeminencia. JDO tiene muchas implementaciones comerciales, mientras que EJB 3.0 está aún en desarrollo. Sin embargo, recientemente otro estándar ha sido anunciado por JCP para abarcar estos dos estándares de manera conjunta y lograr que el futuro estándar trabaje en diversas arquitecturas de Java. Otro ejemplo a mencionar es Hibernate, el framework de mapeo objeto-relacional más usado en Java que inspiró la especificación EJB 3.

En el framework de desarrollo web Ruby on Rails, el mapeo objeto-relacional juega un rol preponderante y es manejado por la herramienta ActiveRecord. Un rol similar es el que tiene el módulo DBIx:: Class para el framework basado en Perl Catalyst, aunque otras elecciones también son posibles.

BASES DE DATOS DISTINTAS A SQL

Las bases de datos como Caché no necesitan mapeo objeto-relacional manual. El acceso del SQL a los valores no escalares ya ha sido construido. Caché permite a los desarrolladores diseñar cualquier combinación de programación orientada a objetos y almacenamiento estructurado en tablas en la misma base de datos en lugar de depender de herramientas externas.



Otra solución puede ser el uso de un sistema de administración de base de datos orientada a objetos (OODBMS: Object-oriented database management system), lo cual, como el nombre lo sugiere, es una base de datos diseñada específicamente para trabajar con valores orientados a objetos. Usar un OODBMS puede eliminar la necesidad de convertir datos desde y hacia su forma SQL, y los datos pueden ser almacenados en su representación original como objetos.

Las bases de datos orientadas a objetos aún no han conseguido una alta aceptación y uso. Una de las principales limitaciones reside en que, cambiar de un sistema de administración de base de datos SQL a un sistema orientado totalmente a objetos implica que se pierde la capacidad de crear sentencias SQL, un método ya probado para obtener combinaciones específicas de datos. Por esta razón, muchos programadores se encuentran más a gusto trabajando con un sistema de mapeo de objetos y SQL, aún cuando la mayoría de las bases de datos comerciales orientadas a objetos son capaces de procesar consultas SQL de manera limitada.

DESVENTAJAS DE ORM

Curva de aprendizaje: Las herramientas (o librerías) ORM suelen ser muy amplias, por lo que llegar a explotar su máximo rendimiento costará tiempo. Antes de incluir esta tecnología en algún proyecto se debe estudiar en profundidad y hacer pruebas de la implementación de esta, sobre todo si el proyecto es entre mediano y grande.

Menor rendimiento: Está claro que tener una capa tan enorme entre tu código y el sistema de base de datos, ralentizará un poco la aplicación. Pongamos el ejemplo para una simple query, el sistema tendrá que: convertir la consulta al SQL del proveedor usado; enviar la consulta; leer registros, limpiarlos y convertirlos a objetos. Depende del encargado de proyecto decidir si este descenso en performance se verá compensado con la facilidad y velocidad de desarrollo.

Sistemas complejos: Normalmente la utilidad de ORM desciende con la mayor complejidad del sistema relacional. Es decir, si tienes una base de datos compleja, ORM también se te hará más complejo y perderás más tiempo adaptando tus clases que en un sistema de menor complejidad.

CONCLUSION

Después de haber realizado la investigación de lo antes expuesto, se puede resumir que el uso de los ORM es imprescindible para poder trabajar con base de datos relacionales y programación orientada a objetos, ya que es éste quien puede “enlazar” estas dos modalidades de trabajo. Se podrán crear todas las consultas de base de datos que se desean. El pensar que no se podrán hacer ciertas cosas con esta modalidad porque es limitante es incorrecto, más bien es todo lo contrario. Al decidir alguno de los Frameworks mencionados anteriormente, sólo bastará con estudiarlo y nos podremos dar cuenta cómo es más fácil de lo que parece.

BIBLIOGRAFIA

- http://www.tecnoretales.com/programacion/que-es-doctrine-orm/
- http://juliansomoza.com.ar/programadorphp/como-trabajo.html
- http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html
- http://es.wikipedia.org/wiki/Mapeo_objeto-relacional