

Diseño de una capa de persistencia para Bases de Datos Objeto-Relacionales en Oracle8i

B.Vela, E.Marcos, J.M.Vara
Grupo KYBELE
Universidad Rey Juan Carlos
{b.vela, e.marcos}@escet.urjc.es, jmvara@pantuflo.escet.urjc.es

Resumen

Las Bases de Datos Objeto-Relacionales (BDOR) se han posicionado claramente como la primera opción para la gestión de datos e interrelaciones complejos. Este tipo de bases de datos (BD) extienden el modelo relacional con nuevos tipos de datos, métodos, etc. Sin embargo, su utilización conlleva una fuerte dependencia del producto, ya que en la mayoría de los casos los productos han aparecido antes que el estándar, SQL:1999 y no se ajustan totalmente al mismo. Además, la migración desde cualquier modelo de persistencia a otro siempre es una tarea laboriosa, ya que los modelos no suelen ser compatibles. Por todo ello, suele existir una gran dependencia entre las aplicaciones y el modelo de persistencia que se elija inicialmente. Un modo de solucionar este problema es aislar las aplicaciones del modelo de persistencia elegido, mediante una capa de persistencia, que deberá permitir la manipulación de los datos, garantizando la integridad de la información. La capa de persistencia se ocupará, en definitiva, de ocultar a las aplicaciones el modelo de persistencia de los datos. En este artículo se describe una capa de persistencia para un producto concreto, Oracle8i.

Palabras Clave: Capa de persistencia, Arquitectura Multinivel, Mantenimiento de la integridad, Bases de Datos Objeto-Relacionales, Oracle8i

1 Introducción

La necesidad de dar soporte a tipos de datos e interrelaciones cada vez más complejos, ha dado lugar a la aparición de Sistemas de Gestión de Bases de Datos (SGBD) con modelos de datos semánticamente más ricos [7]. Este es el caso de las Bases de Datos Orientadas a Objetos (BDOO) [8,11] y de las Bases de Datos Objeto-Relacionales (BDOR) [12,15], que permiten soportar nuevos tipos de datos, interrelaciones complejas, herencia, datos multimedia y XML[10], etc. Parece cada vez más claro que, de estas dos tendencias, son las BDOR las que están teniendo un mayor ámbito de utilización y se espera que esta tecnología sea la que se utilice mayoritariamente en los próximos años [8]. Además, este tipo de Bases de Datos (BD) también está resultando ser especialmente apropiado para el tratamiento de datos en la Web.

Sin embargo, aunque el empleo de las extensiones de las BDOR ofrece numerosos beneficios, su utilización trae consigo ciertos problemas. En primer lugar, los productos no implementan completamente el actual estándar para BDOR, SQL:1999 [12], dado que su aprobación fue posterior a la salida al mercado de muchos de estos productos. Por otro lado, los modelos de persistencia no suelen ser compatibles entre sí y, por lo tanto, es muy difícil migrar de un modelo de datos a otro. Por todo ello, un problema de la utilización de los productos OR es la incompatibilidad entre distintos sistemas o, lo que es lo mismo, una total dependencia del producto que se elige.

Una forma de solucionar este problema es mediante la definición de una capa de persistencia genérica [16] que aisle aplicaciones del modelo de persistencia elegido. Se comenzará definiendo una capa de persistencia para BDOR en el producto Oracle8i [13,14] para, posteriormente, generalizar y obtener la capa de persistencia válida para cualquier Sistema Gestor de BDOR (SGBDOR). Esta capa de persistencia permitirá cualquier operación de manipulación de la BD conservando la integridad de la información. Una vez esté validada esta propuesta con el producto Oracle8i se pasará a generalizar la definición de la capa de persistencia con el fin de poder manipular y garantizar la integridad de los datos en cualquier SGBDOR.

Otros autores han visto la necesidad de crear una capa de persistencia de cara a aislar la aplicación del modelo de persistencia que existe por debajo. En [1,2,3,4,5,6] se propone la definición de una capa de persistencia con el fin de dar persistencia a una aplicación orientada a objetos en una base de datos relacional, haciendo la transformación de los objetos a tablas relacionales.

Las grandes ventajas de la utilización de una capa de persistencia genérica son, por un lado, la reutilización de la capa de persistencia en distintas aplicaciones y, por otro, la total independencia del modelo de persistencia que exista por debajo. En la Figura 1 se puede ver la arquitectura multinivel que aísla la aplicación del modelo de persistencia elegido, tendrá 3 niveles en el caso genérico y existirá una capa adicional en el caso de que el acceso sea vía Web.

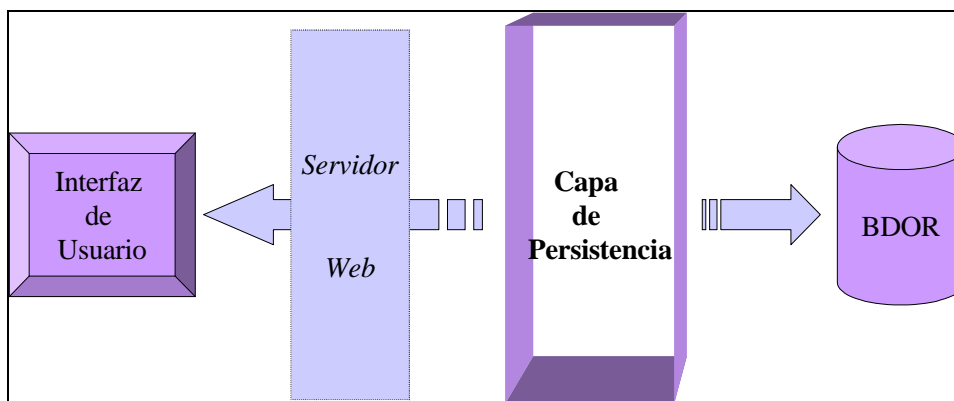


Figura 1. Arquitectura multinivel utilizada

Otro aspecto a tener en cuenta es que en las BDOR las relaciones se pueden implementar de distintas formas. En las BD Relacionales para mantener la integridad de la información se usan claves foráneas (*Foreign Keys*) y son los propios SGBD los que mantienen la integridad de la información. En las BDOR, sin embargo, ya no existe la necesidad de representar las interrelaciones existentes en el modelo conceptual mediante claves foráneas. Así, las relaciones pueden implementarse de diferentes formas usando tipos referencia, los nuevos tipos colección, etc., que ofrecen los productos objeto-relacionales, como Oracle8i [13,14]. Esto, unido a que los BDOR prescindir de la primera forma normal, permite almacenar en un atributo un conjunto de referencias, relacionando así una tupla de una tabla con un conjunto de tuplas de otra. Sin embargo, la utilización de estos tipos de datos para implementar las relaciones supone un coste adicional en el mantenimiento de la consistencia en la información, ya que no se realiza por el propio SGBD, dado que el usuario es el que decide cómo implementar las mismas y sólo en el caso de las claves foráneas se gestionará la integridad de forma automática por parte del SGBD. Por lo tanto, en la actualidad, el mantenimiento de la consistencia en las relaciones implementadas mediante los nuevos tipos de datos obliga a recurrir a la programación (rutinas, procedimientos almacenados y/o disparadores), que será específica del esquema para el que se desarrolla, es decir, una gran cantidad de software dependiente de la aplicación y no siempre reutilizable. La utilización de una capa de persistencia genérica garantizará la gestión de la información y el mantenimiento de la integridad en la BDOR. Además, para el desarrollador debe ser totalmente transparente el modelo de persistencia que exista por debajo. Así, la capa de persistencia nos aportará la independencia de la gestión de la información de la BDOR y del mantenimiento de las relaciones respecto a la BD y respecto a la interfaz de usuario de la aplicación en concreto que utilice la misma.

El resto del artículo se estructura de la siguiente forma: el apartado 2 describe una clasificación de los distintos tipos de relaciones que pueden existir en el SGBDOR Oracle8i. En el tercer apartado se verá el diseño de la capa de persistencia, incluyendo el modelo de análisis, el modelo de diseño y el de implementación. En el apartado 4 se describe la implementación parcial del diseño propuesto y su aplicación a un sitio Web. Por último, en el apartado 5 se exponen las principales conclusiones y se plantean futuros trabajos.

2 Caracterización de relaciones en Oracle8i

Como paso previo a la definición de la capa de persistencia es necesario realizar una clasificación de los distintos tipos de relaciones existentes en el producto Oracle8i. Esta clasificación tendrá en cuenta si se desea implementar la relación entre dos tablas de forma uni- o bidireccional y el tipo de las tablas que participan en la relación [16].

Para ello se presenta en la Figura 2 el modelo de los distintos tipos de datos contemplados por Oracle8i y cómo se relacionan y se forman los diferentes tipos de datos definidos por el usuario a partir de otros. El modelo parte de una jerarquía, cuya raíz es la clase *Tipo Dato*, que se especializa en *Tipo de Dato Predefinido*, *UDT* (tipo de dato definido por el usuario) y *Constructor de Tipo*. Las dos últimas utilizan otros tipos de dato para su definición. El Constructor de Tipo se especializa en *Tipo REF*, que es el que permite definir referencias a cualquier UDT, y *Tipo Colección*, empleado para almacenar un conjunto de valores en el cruce de una columna y fila de una tabla. Los tipos colección se dividen en *Tipo Varray* y *Tipo Nested Table*. Finalmente, las *Tablas Tipadas* se definen sobre un UDT y las *Nested Tables* (Tablas Anidadas) sobre un *Tipo Nested Table*. Por último, cabe destacar que una Tabla Tipada puede usar varias Nested Tables y varios Varrays.

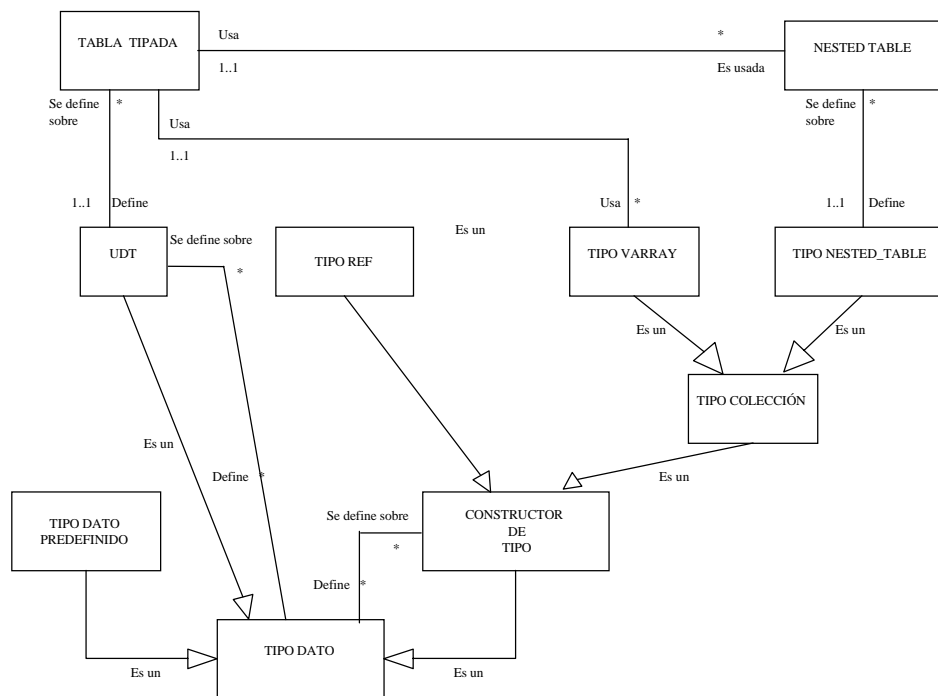


Figura 2. Tipos de Datos Oracle8i

Una vez vistos los tipos de datos existentes y sus interrelaciones en el producto Oracle8i, se verá la clasificación de los distintos tipos de relación, así como las distintas formas que existen de implementar las relaciones existentes en Oracle8i con sus constructores específicos.

Como primer paso se diferenciará entre Tablas Tipadas y Tablas no Tipadas, ya que las Tablas no Tipadas no pueden ser referenciadas y, por lo tanto, siempre que haya Tablas no Tipadas implicadas en una relación no se pueden crear colecciones de referencias para representar la misma. Así, las relaciones en las que el sentido de la misma lleve a una Tabla no Tipada sólo podrán representarse mediante claves foráneas.

Una vez hecha la distinción entre Tablas Tipadas y no Tipadas, se hace una clasificación entre relaciones unidireccionales y bidireccionales. En las relaciones bidireccionales cada uno de los sentidos de la navegación puede reflejarse empleando tipos de datos diferentes (REFs, VARRAYs o NESTED TABLEs). En este caso, las acciones necesarias para mantener la consistencia al realizar alguna operación de modificación serán distintas, según cuál sea la tabla en la que primero se realice la modificación (inserción, borrado o actualización).

Así, la clasificación está basada en tres criterios independientes: el tipo de la relación (unidireccional o bidireccional), la clase de tablas que participan en la misma (tipadas y no tipadas) y, finalmente, el tipo de datos (tipo ref, Varray o Nested Table) o, en su defecto, la restricción (clave foránea) empleada para representarla. En la Figura 3 se puede ver la clasificación de las relaciones de acuerdo con los tres criterios mencionados con anterioridad.

GÉNERO	TIPO TABLAS PARTICIPANTES	Navegación de la tabla A a la tabla B	Navegación de la tabla B a la tabla A
UNIDIRECCIONAL	TABLA No TIPADA - TABLA No TIPADA	Foreign Key	...
	TABLA TIPADA - TABLA No TIPADA	Foreign Key	...
	TABLA No TIPADA - TABLA TIPADA	Tipo Ref / Foreign Key	...
	TABLA TIPADA - TABLA TIPADA	Tipo Ref / Foreign Key	...
		Varray Nested Table	...
BIDIRECCIONAL	TABLA No TIPADA - TABLA No TIPADA	Foreign Key	Foreign Key
	TABLA TIPADA - TABLA No TIPADA	Foreign Key	Varray Nested Table
	TABLA No TIPADA - TABLA TIPADA	Tipo Ref / Foreign Key Varray Nested Table	Foreign Key
		Tipo Ref / Foreign Key	Tipo Ref / Foreign Key Varray Nested Table
	TABLA TIPADA - TABLA TIPADA	Varray	Tipo Ref / Foreign Key Varray Nested Table
		Nested Table	Tipo Ref / Foreign Key Varray Nested Table

Figura 3. Clasificación de relaciones en Oracle8i

Una vez vistos los distintos tipos de datos, así como la clasificación de los distintos tipos de relaciones de Oracle8i, se pasará a definir la capa de persistencia en Oracle8i definiendo en primer lugar el modelo conceptual de la misma y después el modelo de diseño. A partir de este modelo de diseño se ha implementado parcialmente la capa de persistencia en Oracle8i, con el fin de poder validar la misma y, posteriormente, poder definir una capa de persistencia genérica para cualquier SGBDOR.

3 Diseño de la Capa de Persistencia

En este apartado se realiza la definición de la capa de persistencia para el producto Oracle8i, para lo cual, en primer lugar, se define el modelo conceptual (o de análisis) de la capa de persistencia para, posteriormente, mostrar las modificaciones realizadas al mismo para la obtención del modelo de diseño de la capa de persistencia.

3.1 Modelo conceptual de la capa de persistencia

La Figura 4 muestra el modelo de análisis de la capa de persistencia.

La clase **Relación** (clase abstracta que permite referirse a cualquier tipo de relación de forma genérica) será la raíz de una jerarquía total y exclusiva en la que se definen dos subclases: **Unidireccional** y **Bidireccional**, que se emplean para representar los requisitos navegacionales de una relación concreta.

Se establecerá una relación de agregación que será clave en la descomposición del problema que nos ocupa. Toda relación entre dos tablas tipadas (no se contemplan las tablas no tipadas porque la funcionalidad aportada por la capa de persistencia que se está definiendo no tiene demasiada utilidad en el caso de tablas no tipadas, pues es el propio sistema el que se encarga del mantenimiento de la consistencia) contiene al menos un **Enlace**. Esta clase

es la que realmente permite abordar el problema desde una perspectiva mucho más sencilla. Las relaciones unidireccionales contendrán un enlace mientras que las bidireccionales contendrán dos enlaces. Ahora, el mantenimiento de la consistencia en la información referente a una relación, se reduce al mantenimiento de la consistencia en cada uno de los enlaces que contiene la relación. Esta afirmación permite simplificar en gran medida la parte más problemática del análisis, la que se refiere a las relaciones bidireccionales. Cada enlace contiene dos tablas tipadas, y puede ser de varios tipos, dependiendo de la forma elegida para representarlo, con lo que se define una jerarquía total y exclusiva, cuya raíz sea la clase **Enlace**, y cuyas subclases serán **Usa_Ref**, **Usa_Varray** y **Usa_NT**.

Se incluye también en el modelo la subclase **Usa_FK**, aunque este trabajo se centra más específicamente en las relaciones implementadas con los nuevos tipos de datos de Oracle8i, como se ha mencionado anteriormente. Sólo es necesario emplear claves foráneas cuando se trabaja con tablas no tipadas, ya que no se puede usar un tipo referencia para referenciar una tupla de una tabla no tipada. En estos casos el SGBD se ocupa de forma automática del mantenimiento de la consistencia, por lo que carecen de especial relevancia para el modelo propuesto. Por otro lado, también se pueden utilizar las claves foráneas para referenciar tuplas de tablas tipadas, pero por ser tablas tipadas, pueden usarse igualmente tipos referencia (que además potencian las características de orientación al objeto). Por lo tanto, estos casos ya son contemplados en el modelo incluyendo la clase **Usa_REF**. Por todo ello, se ha representado la clase **Usa_FK** con trazo discontinuo, indicando así dónde habría de ubicarse en el modelo, pero también que no es necesario contemplarla.

Las subclases de la clase **Enlace** serán todas clases instanciables, y serán las que aporten toda la funcionalidad de la capa de persistencia propuesta, redefiniendo los métodos abstractos **Insertar**, **Actualizar** y **Borrar**, heredados de la clase padre.

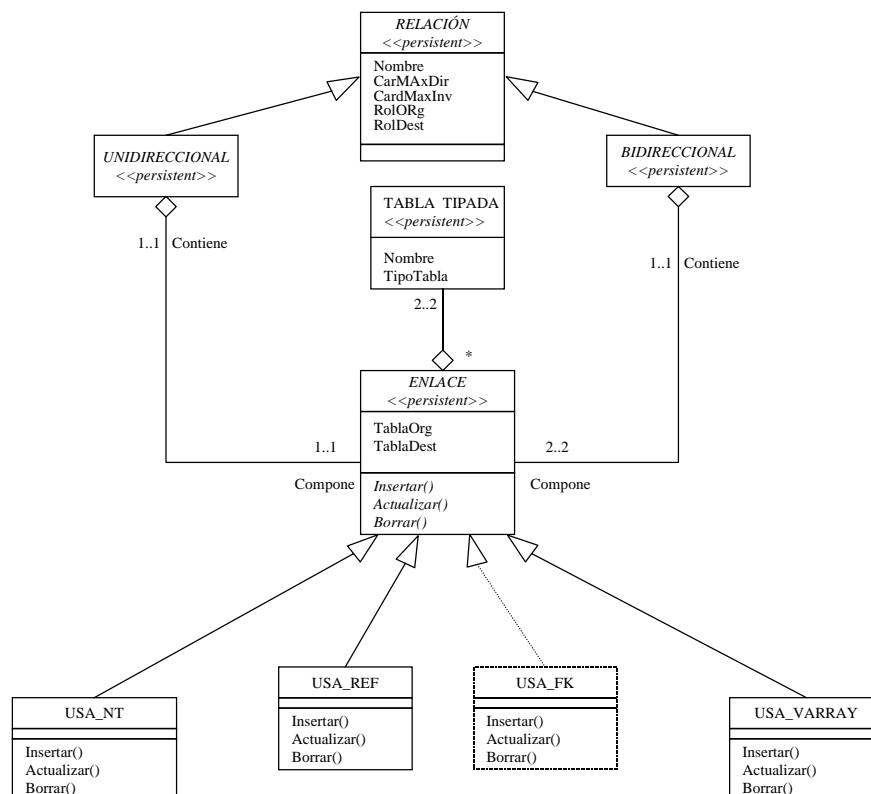


Figura 4. Esquema del modelo de análisis

3.2 Modelo de Diseño

Un refinamiento del modelo de análisis descrito en el apartado anterior, lleva al siguiente modelo de diseño. En éste se definen los atributos y las cabeceras de las operaciones de cada clase. Así mismo, se incluyen dos nuevas clases, *Parámetros* y *Opción* que serán necesarias para la fase de implementación como se puede ver en la Figura 5. La clase *Parámetros* se empleará para pasar los datos de la operación a realizar cuando se invoque a uno de los métodos de una instancia de cualquiera de las subclases de *Enlace*. En cuanto a la clase *Opción*, los objetos de esta clase serán instanciados para ser proporcionados como parámetros en las llamadas a las operaciones de modificación de las instancias de las clases *Usa_REF*, *Usa_Varray* y *Usa_NT*. Indicarán al objeto en cuestión la política a seguir en la operación de modificación (borrado o actualización) para la que se instancia. Esto es, si se quiere que la modificación sea en cascada, con puesta a nulos, a un valor por defecto, o cualquier otra opción que pudiera convenir a la aplicación. La inclusión de esta clase en el modelo permite adaptar mejor la capa de persistencia propuesta a las necesidades de cada aplicación, sin perder su carácter genérico.

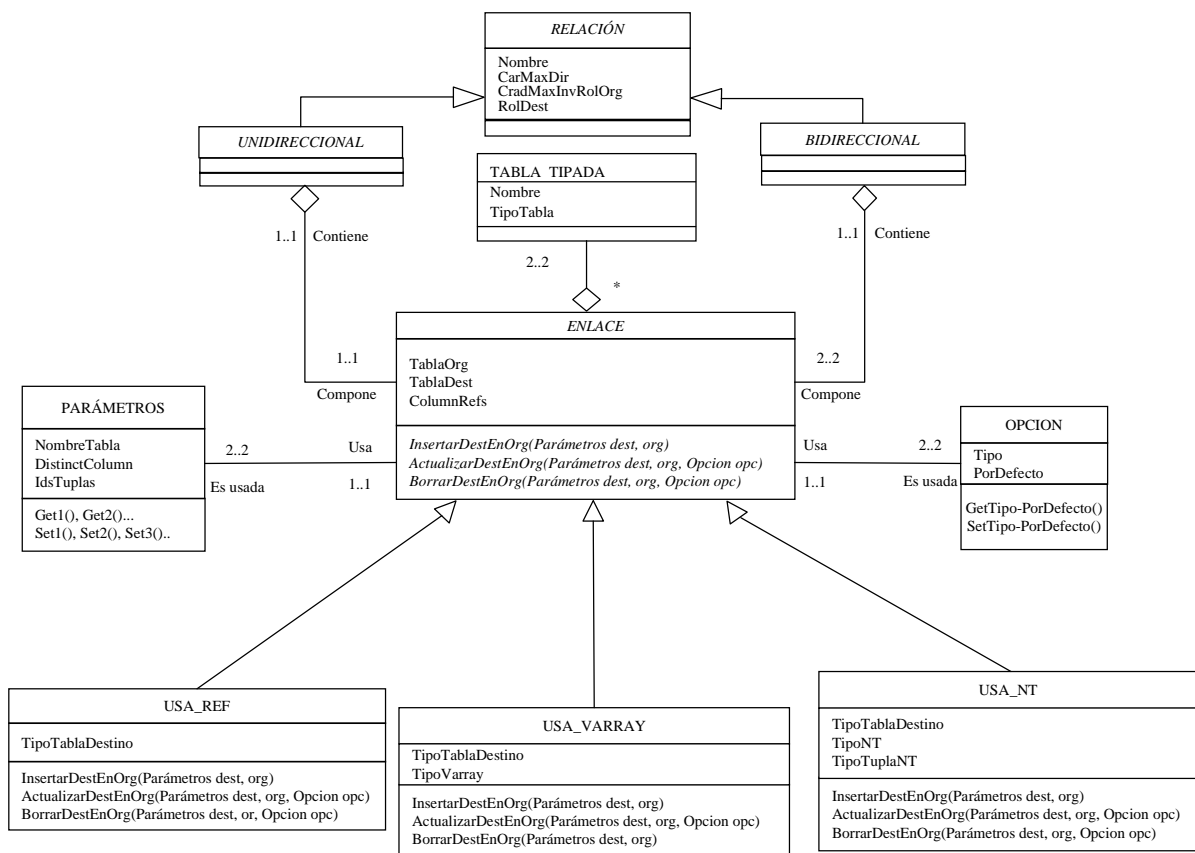


Figura 5. Esquema del modelo de diseño

3.3 Modelo de Implementación

Una vez definido el modelo de diseño de la capa de persistencia, se ha pasado a implementarlo parcialmente, con el fin de comprobar su validez. Se han empleado un conjunto de clases JAVA, haciendo corresponder las clases del modelo de diseño con las clases JAVA de la implementación. Se ha realizado una correspondencia directa entre el modelo de diseño y la implementación, es decir, los atributos y métodos de las clases de implementación son exactamente los mismos que aparecen en el modelo de diseño.

En la implementación realizada se han empleado un pequeño conjunto de tablas que almacenan el meta-esquema, constituido por las relaciones que existen en el esquema de BD. Este conjunto de tablas permite dar persistencia a

las clases del modelo de diseño que se definieron como persistentes. Así, se obtiene un modelo de implementación que difiere del de diseño, en tanto en cuanto parte del modelo de diseño se refleja ahora mediante el esquema lógico del meta-esquema de las relaciones presentes en la BD. En la Figura 6 se muestra el meta-esquema resultante.

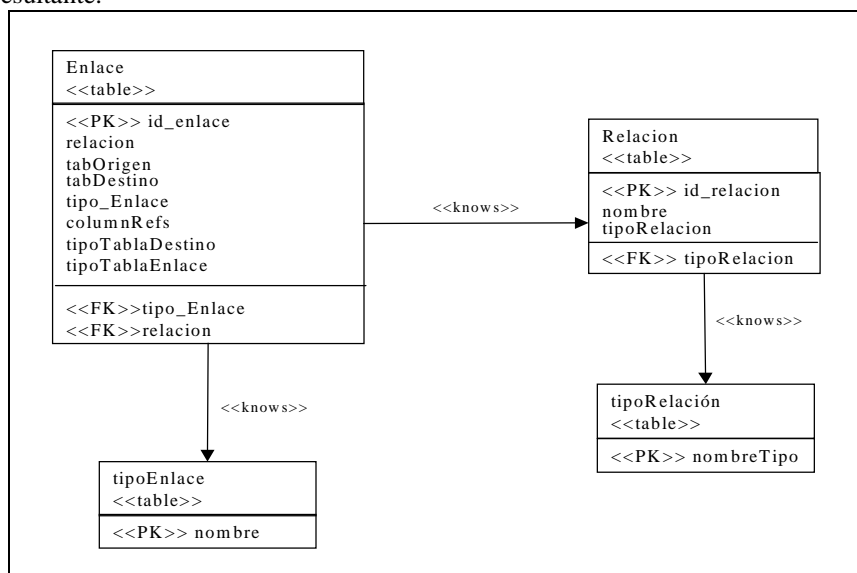


Figura 6. Esquema lógico del meta-esquema de las relaciones

La jerarquía del modelo de diseño cuya raíz es la clase *Relación*, y cuyas subclases son las clases *Unidireccional* y *Bidireccional* se ha representado en el modelo de implementación con las tablas *Relación* y *tipoRelación* en la BD. La tabla *tipoRelación* sirve para representar el dominio de los tipos de relación que existen, (unidireccional o bidireccional).

Para representar la clase *Enlace* se emplea la tabla *Enlace*, que contendrá una tupla por cada enlace entre dos tablas (una relación unidireccional contiene un enlace y una relación bidireccional contiene dos enlaces) que exista en el esquema de la BDOR sobre la que se trabaja. Además para representar el tipo de enlace (mediante Ref, Varray o Nested_Table) se utiliza la tabla *tipoEnlace*, que simula el dominio de los tipos de enlace.

No obstante la clase *Enlace* aparece en el modelo de implementación, porque es necesario codificarla, ya que las clases *Usa_REF*, *Usa_Varray* y *Usa_NT* heredan de la clase *Enlace*.

Conviene mencionar que esta necesidad de dotar de persistencia a algunas de las clases que constituyen la capa de persistencia, podría afrontarse en un futuro, integrando estas clases (y por tanto, sus extensiones, es decir, los objetos de la clase) en el mismo esquema de BD, sin emplear tablas adicionales.

La clase abstracta *Enlace* contiene los métodos *InsertarDestEnOrg*, *ActualizarDestEnOrg*, *BorrarDestEnOrg* que son sobre los que se sustenta la funcionalidad de la capa de persistencia.

En la Figura 7 se muestra el modelo de implementación.

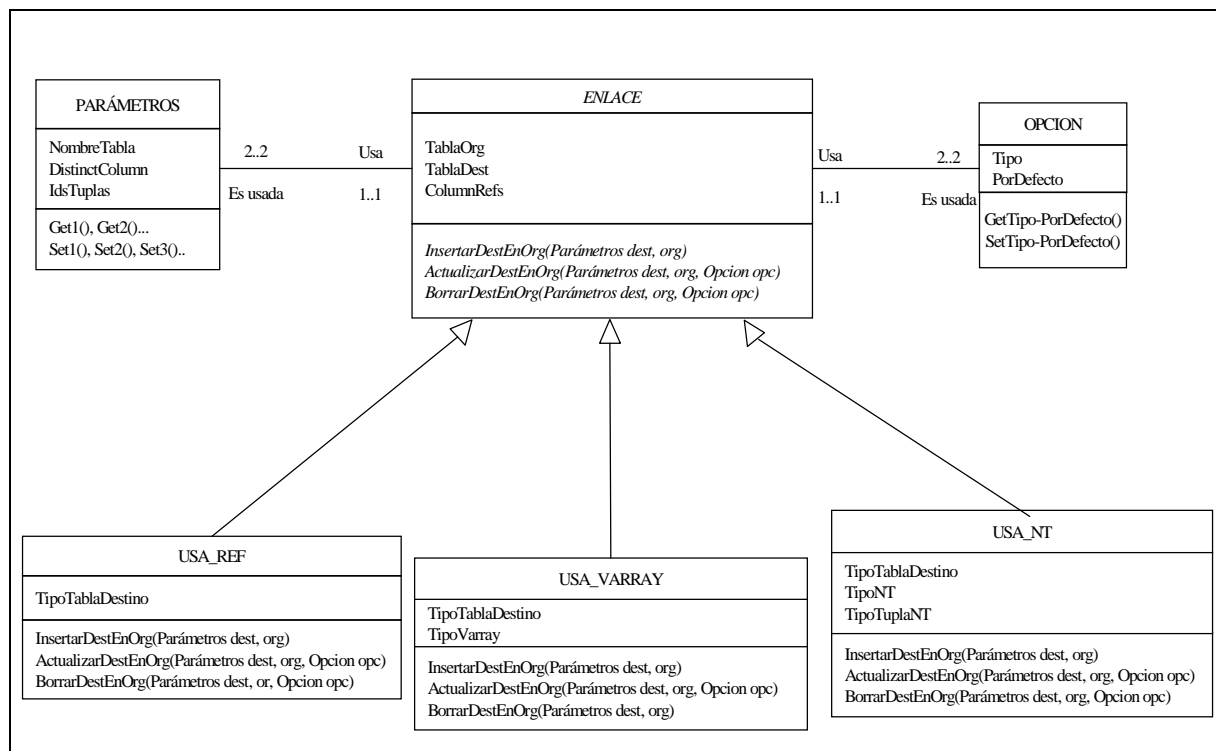


Figura 7. Modelo de Implementación

4 Caso de Prueba

Con el fin de mostrar toda la potencia de la propuesta de la capa de persistencia, se incluye la implementación parcial de la capa de persistencia diseñada y un ejemplo de su utilización para el caso concreto de una aplicación Web dinámica en explotación, permitiendo la gestión y el mantenimiento de la información de la BDOR por medio de la capa de persistencia.

Para esta implementación parcial, se ha optado por emplear JAVA como lenguaje de programación, porque es independiente de la plataforma sobre la que se ejecuta y por lo tanto, aporta aún más genericidad a la capa de persistencia que se presenta. Además, este lenguaje presenta una buena integración con el SGBDOR Oracle8i, ofreciendo potentes herramientas para interacción con bases de datos. Además es el lenguaje de programación orientado a objetos más potente y mayoritariamente extendido entre la comunidad de programadores y ofrece un muy buen soporte para el desarrollo de aplicaciones Web, especialmente recomendable por tanto, para la implementación del ejemplo, en el que se ha desarrollado una interfaz de usuario para una aplicación Web.

Además del código JAVA que implementa la capa de persistencia, se han empleado algunas tablas almacenadas en la misma BD con el fin de mantener el meta-esquema de las relaciones entre las tablas del esquema de la BD.

Por último, es necesario mencionar que el desarrollo de la capa de persistencia se ha realizado para el SGBDOR Oracle8i, porque es el SGBDOR más extendido y no se aleja demasiado del modelo objeto-relacional del estándar SQL:1999 [12], con lo que constituye un buen punto de partida desde el que afrontar en el futuro una capa de persistencia genérica, y por lo tanto, válida para cualquier SGBDOR.

En la siguiente Figura 8 se puede ver una página de la aplicación WEB que utiliza esta propuesta. La aplicación Web básicamente gestiona los miembros de un grupo de investigación y la docencia que se imparte en dicho grupo. Así, en este ejemplo, una vez insertados los miembros del grupo y las asignaturas que se imparten en esta página se seleccionan las asignaturas impartidas, los proyectos que dirige y la tesis doctoral realizada por el

miembro introducido en el formulario previo. La capa de persistencia será la encargada de garantizar la consistencia de la información contenida en dichas relaciones.



Figura 8. Ejemplo de inserción en relación a través de la capa de persistencia

5 Conclusiones y trabajos futuros

En este artículo se ha presentado una capa de persistencia para el SGBDOR Oracle8i. Esta capa de persistencia permite, por un lado, gestionar toda la información de la BDOR y, por otro, garantiza el mantenimiento de la integridad de la misma. Esta capa de persistencia se define sobre una arquitectura multinivel y, por lo tanto, aísla la gestión de la información de la BDOR de la aplicación en concreto. Por lo tanto, esta capa será totalmente genérica e independiente del esquema de la BDOR en concreto y de la aplicación que la utiliza.

Para diseñar la misma se ha realizado un modelo de análisis y un modelo de diseño de la capa de persistencia utilizando diagrama de clases de UML. Con el fin de validar el diseño se ha realizado un modelo de implementación en JAVA y se ha realizado una implementación parcial empleando clases JAVA.

Como futura línea de trabajo, se pretende completar la implementación de la capa de persistencia para el gestor de Oracle8i, con el fin de validar por completo el modelo de diseño de la misma. Una vez esté completamente validado el diseño de la capa de persistencia para este gestor en concreto, se pretende pasar a definir una capa de persistencia válida para cualquier SGBDOR.

Agradecimientos

Este trabajo se ha llevado a cabo dentro del proyecto coordinado DAWIS (TIC 2002-04050-C02) financiado por el MICYT.

Referencias

1. Ambler, S.W. (1998a) The Design of a Robust Persistence Layer For Relational Databases. Disponible en <http://www.ambysoft.com/persistenceLayer.pdf> , 1998.
2. Ambler, S.W. (1998b). Mapping Objects To Relational Databases: An AmbySoft Inc. White Paper. <http://www.ambysoft.com/mappingObjects.html>
3. Ambler, S.W. (1998c). Persistence Layer Requirements, Software Development, Enero 1998, pp 70-71.
4. Ambler, S.W. (1998d). Robust Persistence Layers, Software Development, Febrero 1998, pp 73-75.
5. Ambler, S.W. (1998e). Designing a Persistence Layer (Part 3 of 4), Software Development, Marzo 1998, pp 68-72.
6. Ambler, S.W. (1998f). Designing a Robust Persistence Layer (Part 4 of 4), Software Development, April 1998, pp73-75.
7. Bertino, E. y Marcos, E. (2000), Object Oriented Database Systems. En Advanced Databases: Technology and Design, O. Díaz y M. Piattini (Eds.). Artech House, 2000.
8. Bertino, E. y Martino, (1993), Object-Oriented Database Systems. Concepts and Architectures. Ed. Addison-Wesley.
9. Booch, Rumbaugh y Jacobson (1999), *The Unified Modelling Language User Guide*. Addison Wesley.
10. Bray, T., Paoli, J, Sperberg-McQueen, C. M. y Maler, E.(2000). Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation Retrieved from <http://www.w3.org/TR/2000/REC-xml-20001006>. 6 October 2000.
11. Leavit, N.(2000), "Whatever Happened to Object-Oriented Databases?". Computer, pp. 16-19, August 2000
12. Eisenberg, A. y Melton, J. (1999). SQL:1999, formerly known as SQL3. ACM SIGMOD Record, Vol. 28, No. 1, pp. 131-138, Marzo, 1999.
13. Oracle Corporation (1998), "Objects and SQL in Oracle8". Oracle Technical White paper. In: *Extended DataBase Technology conference (EDBT'98)*. Valencia (Madrid).
14. Oracle Corporation (2000), Oracle8i. *SQL Reference. Release 3 (8.1.7)*. In: www.oracle.com.
15. Stonebraker y Brown (1999). Object-Relational DBMSs. Traking the Next Great Wave. Morgan Kauffman, 1999.
16. Vara, J.M. (2002), Definición de una capa de una capa de persistencia para la gestión de una BDOR en Oracle8i. Proyecto Fin de Carrera. Universidad Rey Juan Carlos. Julio 2002.