# Transformación de Modelos para el Desarrollo de Bases de Datos Objeto-Relacionales

Juan M. Vara, Belén Vela, José Mª Cavero y Esperanza Marcos, *Universidad Rey Juan Carlos*

***Resumen*-- En este trabajo presentamos una aproximación para el desarrollo de Bases de Datos (BD) Objeto-Relacionales (OR) en el marco de MIDAS, una metodología dirigida por modelos para el desarrollo de Sistemas de Información Web. En nuestra propuesta, el Modelo Independiente de la Plataforma (PIM, *Platform Independent Model*) será el modelo conceptual de datos y a partir de éste obtendremos el Modelo Específico de Plataforma (PSM, *Platform Specific Model*), esto es el modelo OR. Dado que la notación utilizada en ambos modelos es UML, la propuesta incluye también los correspondientes perfiles UML para la elaboración del modelo OR (para el estándar SQL:2003 y para un producto concreto, Oracle10*g*). Este artículo se centra en la formalización de los *mappings* de PIM a PSM. Así, definimos las transformaciones de modelos para obtener el esquema de la BD (el modelo OR) a partir del modelo conceptual de datos (PIM), inicialmente como reglas expresadas en lenguaje natural, para posteriormente expresarlas mediante gramáticas de grafos.**

***Palabras Clave*-- Desarrollo Bases de Datos Objeto-Relacionales, Gramáticas de Grafos, Transformaciones de Modelos, MDA, UML**

## I. Introducción

A pesar de su enorme aceptación a lo largo de las últimas décadas, las Bases de Datos (BD) relacionales presentan algunas limitaciones a la hora de satisfacer las necesidades de algunas de las aplicaciones actuales. Así, aparecen aplicaciones cada vez más sofisticadas que incluyen objetos y relaciones cada vez más complejos/as. Para poder representar estos objetos y sus relaciones en el modelo relacional sería necesario descomponerlas en un elevado número de tuplas, lo que resultaría en la necesidad de hacer un importante número de *joins* para poder recuperar un objeto y la consiguiente pérdida de rendimiento [1].

La nueva generación de BD orientadas a objetos, que incluyen las BD Objeto-Relacionales (OR) [2], aparecieron para resolver este tipo de problemas. Esta nueva tecnología, basada en estándares [3] e incorporada ya en muchos productos comerciales [4][5][6], resulta ideal para almacenar y recuperar datos complejos porque ofrece soporte para la definición de tipos de datos complejos y el mantenimiento de sus relaciones, datos multimedia, herencia, etc.

No obstante, las mejoras tecnológicas no bastan para aprovechar toda la potencia de estas extensiones, son necesarias además, metodologías para el desarrollo de BDOR, igual que lo eran para el desarrollo de BD relacionales. Estas metodologías [7][8] adoptaban una *aproximación dirigida por modelos*: se definían diferentes modelos para los diferentes niveles de abstracción, tecnologías y fases del proceso de desarrollo, así como las reglas de transformación entre dichos modelos. En realidad, esta aproximación no dista mucho de una de las tendencias de mayor auge a día de hoy en el desarrollo software: la Arquitectura Dirigida por Modelos (MDA, *Model Driven Arquitecture*) [9] propuesta por el OMG. MDA es un marco de trabajo para el desarrollo software cuya principal característica es la definición de los modelos como elementos de primer orden en el diseño, desarrollo e implementación del software y la definición de las transformaciones entre dichos modelos. En función del nivel de abstracción, MDA considera diferentes tipos de modelos: los requisitos del sistema son recogidos en los Modelos Independientes de Computación (CIM, *Computation Independent Models*); los Modelos Independientes de la Plataforma (PIM, *Platform Independent Models*) representan la funcionalidad del sistema abstrayéndose de la plataforma final y los Modelos Específicos de la Plataforma (PSM, *Platform Specific Models*) combinan las especificaciones contenidas en un PIM, con los detalles de la plataforma elegida. A partir de los distintos PSM se pueden generar automáticamente distintas implementaciones del mismo sistema.

En este trabajo presentamos una aproximación metodológica basada en MDA, siendo esta su aportación principal frente a otras propuestas [11][12], para el desarrollo de BDOR en el marco de MIDAS [10], una metodología dirigida por modelos para el desarrollo de Sistemas de Información Web (SIW). MIDAS propone una arquitectura dirigida por modelos basada en MDA y considera el modelado del sistema de acuerdo a los aspectos de contenido, hipertexto y comportamiento en los niveles CIM, PIM y PSM. La figura 1 muestra la arquitectura simplificada de MIDAS, donde se incluyen los CIM comunes a todo el sistema y los PIMs y PSMs necesarios para representar los aspectos de contenido, hipertexto y comportamiento. En este trabajo nos centramos en el aspecto de contenido, que se corresponde con el concepto tradicional de BD, donde el PIM es el modelo conceptual de datos y el PSM será el modelo OR, ambos representados en UML. Alternativamente, en los casos en que se opte por utilizar

Este trabajo se ha llevado a cabo en el marco de los proyectos: GOLD (TIN2005-00010/) financiado por el Ministerio de Educación y Ciencia FoMDAs (URJC-CM-2006-CET-0387) cofinanciado por la Universidad Rey Juan Carlos y la Comunidad de Madrid.

Juan M. Vara, Belén Vela, José Mª Cavero y Esperanza Marcos pertenecen al grupo de investigación Kybele de la Universidad Rey Juan Carlos, C/ Tulipán S/N, 28933, Móstoles (MADRID)
(e-mail: {juanmanuel.vara, belen.vela, josemaria.cavero, esperanza.marcos} @urjc.es)

tecnología XML para el aspecto de contenido, el PSM será el modelo del XML Schema correspondiente; en [13] abordamos esta alternativa utilizando una BD XML. En el caso de utilizar tecnología OR consideramos dos PSMs diferentes: el primero representa el modelo OR para el estándar SQL:2003 [3] y el segundo representa el modelo OR para un producto concreto, Oracle10*g* [6]. Dado que MIDAS propone la utilización de UML como notación única para todo el modelado del sistema, ha sido necesario definir sendos perfiles UML para BDOR que soportarán la semántica asociada a los dos modelos OR contemplados.

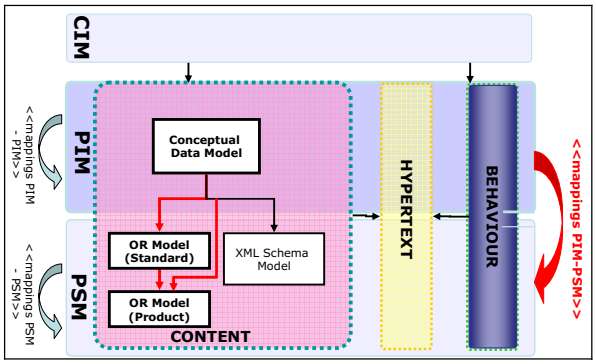


Fig. 1. Arquitectura de MIDAS simplificada

En este trabajo completamos la definición de dichos perfiles y nos centramos en la definición formal de las transformaciones esbozadas en [14]. Para ello definimos primero las reglas de transformación en lenguaje natural y luego las formalizamos utilizando gramáticas de grafos, un tipo especial de transformaciones basadas en reglas representadas mediante diagramas [15]. Por lo tanto y, dado que ya habíamos definido las transformaciones en forma de reglas, resulta un paso natural el uso de grafos para formalizar dichos *mappings*. Con la definición de las transformaciones entre modelos concluimos la definición de la metodología y obtenemos finalmente una **metodología completa** para el desarrollo de BDOR basada en **MDA**.

## II. Desarrollo de BDOR

Nuestra propuesta para el desarrollo de BDOR parte del modelo conceptual de datos (PIM) representado mediante un diagrama de clases UML y aplica sobre este modelo una serie de *mappings,* que se detallarán en la siguiente sección, para obtener el esquema de la BD (PSM). En esta sección presentamos los perfiles UML para BDOR que se utilizan para elaborar los modelos PSM (para el estándar SQL:2003 y el producto Oracle10*g*) que permiten representar el esquema de la BD en UML extendido. Los dos perfiles que se presentan se basan en el perfil para el modelado de BD relacionales presentado en [16].

### A. Perfil SQL:2003 - OR

El perfil OR para el estándar SQL:2003 es un perfil UML que soporta el modelado de un esquema OR basado en el estándar SQL:2003, usando un diagrama de clases UML extendido. El metamodelo que resulta de aplicar dicho perfil se muestra en la figura 2. Dicho diagrama representa sólo las características de objetos del metamodelo OR del estándar SQL:2003, es decir, los artefactos correspondientes al metamodelo relacional no se incluyen. Los tipos de datos se clasifican en *ARRAY*, *MULTISET*, *ROW*, referencia y tipos estructurados. Una tabla tipada está basada en un tipo estructurado. Un tipo estructurado puede contener atributos y métodos (para más detalles véase [14]).

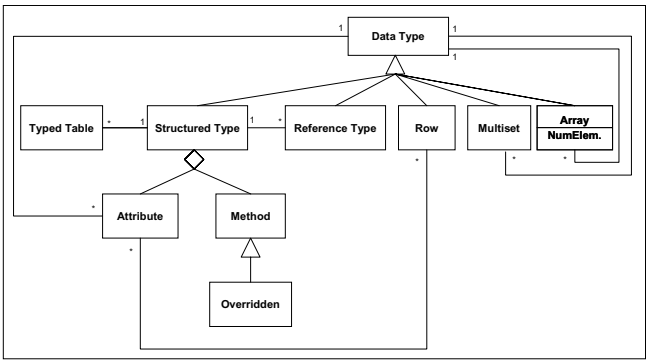


Fig. 2. Metamodelo OR del SQL:2003

### B. Perfil Oracle - OR

En esta sección se presenta el perfil UML para definir esquemas de BDOR para un Sistema Gestor de BD (SGBD) concreto, Oracle10*g*. En realidad no dista mucho del presentado para el estándar SQL:2003, pero dado que los diferentes SGBD comerciales optan por diferentes implementaciones del estándar, es necesario dar soporte al modelado de esas diferencias.

De este modo, posibilitamos la definición de distintos PSMs para los diferentes productos y posibilitamos la generación automática del código que implementa la BD modelada para cada producto. La figura 3 muestra el metamodelo que representa las características del modelo OR de Oracle10*g*. Las principales diferencias con respecto al metamodelo del estándar son que Oracle no soporta el tipo *ROW*, soporta las tablas anidadas (*NESTED TABLEs*) en vez de *MULTISETs* (a pesar de que representan el mismo concepto) y no soporta la herencia de tablas, aunque sí la de tipos.

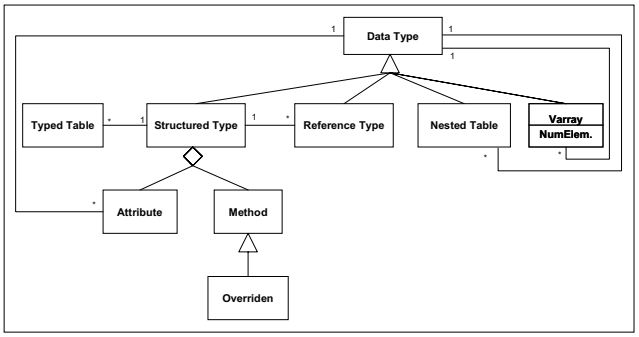


Fig. 3. Metamodelo OR de Oracle10*g*

## III. Transformaciones de modelos para BDOR

Como ya se ha mencionado, nuestra propuesta para el desarrollo de BDOR es una metodología dirigida por modelos, es decir, los modelos son considerados elementos de primer orden a lo largo de todo el proceso de desarrollo. Cada paso

del proceso consiste básicamente en la generación de un modelo de salida a partir de uno o más modelos de entrada sobre los que se aplican un conjunto de transformaciones. En los siguientes pasos del proceso, éste modelo de salida actuará como uno de los modelos de entrada. Así, podemos definir el proceso como el conjunto de tareas a completar para generar cada uno de esos modelos. Es evidente entonces la importancia de los *mappings* en el proceso de desarrollo. De hecho, podemos identificar el conjunto de tareas con el conjunto de *mappings* que debemos realizar. En este trabajo, seguimos una aproximación basada en gramáticas de grafos para formalizar los *mappings* definidos en el proceso de desarrollo de BDOR.

### *A. Reglas de Transformación entre Modelos*

De acuerdo con [9], "la descripción de los *mappings* puede realizarse en lenguaje natural, un algoritmo en un *action language* o un modelo en un lenguaje de transformación". En nuestro caso, y como una primera aproximación a las transformaciones de modelos para el desarrollo de BDOR, hemos optado por describir las reglas de transformación en lenguaje natural, para después expresarlas por medio de gramáticas de grafos. Por cuestiones de espacio, resumimos dichas reglas en la tabla 1.

TABLA I
REGLAS DE TRANSFORMACIÓN PIM – PSM (MODELO CONCEPTUAL – MODELO OR)

| PIM (Modelo Conceptual) | | PSM estándar (SQL:2003) | PSM de producto (Oracle10*g*) |
|---|---|---|---|
| Clase | | Tipo Estructurado | Tipo Objeto |
| Extensión de la Clase | | Tabla Tipada | Tabla de Tipo Objeto |
| Atributo | Multivaluado | Array/Multiset | Varray/Nested Table |
| | Compuesto | ROW/Columna de Tipo Estructurado | Columna de Tipo Objeto |
| | Calculado | Trigger/Método | Trigger/Método |
| Asociación | 1 : 1 | Ref/[Ref] | Ref/[Ref] |
| | 1 : N | [Ref]/[Multiset/Array] | [Ref]/[Nested Table/Varray] |
| | N : M | Multiset/Multiset o Array/Array | Nested Table/Nested Table o Varray/Varray |
| | Agregación | Multiset/Array | Nested Table/Varray de Referencias |
| | Composición | Multiset/Array | Nested Table/Varray de Objetos |
| | Generalización | Tipos/Tablas Tipadas | Tipos/Tablas Tipadas |

### *B. Transformaciones basadas en grafos para el desarrollo de BDOR*

De acuerdo con los principios de MDA, las transformaciones entre modelos incluidas en nuestra propuesta para el desarrollo de BDOR deben ser automatizadas, al menos en cierta medida. Para cumplir con esta premisa hemos optado por una aproximación basada en grafos [15], dado que proyectos como el *Attribute Graph Grammar System* (AGG) [17] proporcionan la funcionalidad necesaria para automatizar transformaciones entre modelos expresadas con grafos. El término *Graph Transformation* es utilizado para referirse a un tipo especial de transformaciones basadas en reglas que se representan por medio de diagramas. Así, puesto que ya habíamos formalizado las reglas de mapeo en un conjunto de reglas expresadas en lenguaje natural, parece conveniente trasladar estas definiciones a reglas de transformación basadas en grafos. Finalmente y desde un punto de vista puramente matemático, podemos contemplar los modelos de tipo UML como grafos: un grafo tiene arcos y nodos mientras que un modelo UML tiene clases y asociaciones entre dichas clases.

La expresión de transformaciones de modelos como gramáticas de grafos se basa en la definición de un conjunto de reglas que siguen la estructura *LHS:= RHS (Left Hand Side:= Right Hand Side)*. En la parte izquierda (*LHS*) se define un (sub)grafo patrón y en la parte derecha (*RHS*) se define un (sub)grafo de sustitución. Cada vez que se encuentra una correspondencia con el patrón en el modelo origen, se sustituye en el modelo destino por el grafo de sustitución. En este trabajo hemos usado la propuesta de [18] para definir las reglas de grafos que recogen en la tabla 1, caracterizada por:

- Los nodos en el LHS se identifican con números consecutivos, lo que hace posible su identificación a sus respectivos nodos en el RHS.
- La referencia a un nodo del LHS, en el RHS, se realiza a través de la expresión match(x), donde x es el número del nodo del LHS.
- Todas las propiedades de los diferentes nodos siempre tendrán asignado un valor inicial. Para indicar que es válido cualquier valor, se utiliza "???".
- Para referenciar a un atributo de un nodo LHS, se usará un ".", por ejemplo match(x).name
- Igual que los nodos pertenecientes al LHS, los nodos pertenecientes al RHS también han de ser numerados. Hay que tener en consideración las siguientes situaciones:
  - Si el número de nodo del LHS aparece en el RHS, el tipo de nodo de un lado es igual al otro.
  - Si un número de nodo aparece en el LHS pero no en el RHS, el nodo del LHS deberá ser eliminado así como las relaciones en las que participa.
  - Si un número de nodo aparece en el RHS, que no estaba en el LHS, hay que añadir este nuevo nodo.
  - Si un número de nodo que aparece en el LHS aparece en el RHS con apóstrofe (x'), significa que el tipo de nodo en el RHS es diferente, aunque se preservan las conexiones con el resto de elementos.

De acuerdo con estas directrices, hemos definido un

conjunto de reglas de transformación basadas en grafos para las transformaciones entre modelos comprendidas en nuestra propuesta para el desarrollo de BDOR. A continuación presentaremos estas reglas. Conviene mencionar que sólo se presentarán las reglas para obtener el modelo OR correspondiente al estándar (SQL:2003). No obstante, estas reglas son igualmente válidas para obtener el PSM para el producto, Oracle10*g*, incluyendo modificaciones mínimas que se mencionan expresamente allí donde deben ser incluidas.

- **Transformación de Clases *Persistent***

En la figura 4 mostramos la regla para transformar las clases persistentes incluidas en el modelo conceptual, esto es las clases del PIM, a objetos del esquema de la BD, es decir, clases en el PSM. Por cada clase UML incluida en el PIM y estereotipada como *persistent* (①−③) se añade en el PSM un nuevo tipo estructurado (tipo de objeto) y una tabla tipada que usa dicho tipo (①'), de manera que la tabla tipada es la extensión del tipo de objeto. Cada atributo o propiedad de la clase UML (②→②') se recoge en el PSM añadiendo un atributo al tipo estructurado correspondiente. Respecto al PSM para Oracle10*g*, la única diferencia estriba las peculiaridades del SQL del producto concreto. Así, es necesario incluir la cláusula *AS OBJECT* en el momento de implementar el diseño resultante en Oracle10*g*.

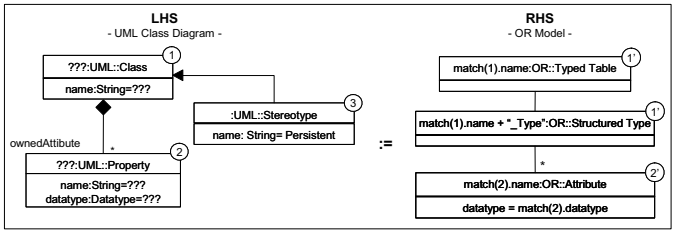


Fig. 4. Transformación de una clase *persistent*

En la figura 5 se muestra como puede instanciarse esta regla. En la parte izquierda de la figura tenemos el PIM: un modelo UML muy sencillo, con una única clase que tiene un único atributo. Dicho modelo se representa en la parte inferior izquierda como una instancia del metamodelo de UML. En la parte derecha se muestra el resultado de aplicar la transformación: un tipo de objeto con un único atributo y una tabla tipada que utiliza dicho tipo.

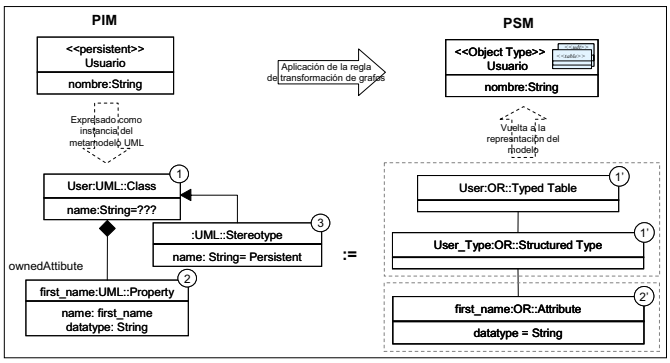


Fig. 5. Ejemplo de aplicación de la regla para clases *persistent*

- **Transformación de Atributos**

Como se indicaba en la tabla 1, la transformación de los atributos de una clase dependerá de la naturaleza del atributo a transformar en cada caso. Así, se han definido distintas reglas para cada tipo de atributo.

Los **atributos multivaluados** del modelo conceptual se corresponden con tipos colección en el esquema de la BDOR: *ARRAY* y *MULTISET* para SQL:2003 y *VARRAY* y *NESTED TABLE* para Oracle10*g*. En la figura 6 se muestra la regla de transformación de grafos a aplicar en el caso de encontrar atributos multivaluados para el caso en que se opta por utilizar el tipo *MULTISET*: la clase UML (①) posee un atributo multivaluado, tal y como indica el valor *true* del atributo *isMultivalued* de la propiedad UML (②). Por tanto, el tipo de objeto correspondiente a la clase UML (①') posee un atributo de tipo *MULTISET* (②'−③'). La misma regla es válida para el caso en que se utilicen *ARRAYs* (SQL:2003) o *VARRAYS/NESTED TABLES* (Oracle10*g*) sin más que reemplazar la palabra clave *MULTISET* por el tipo colección correspondiente.

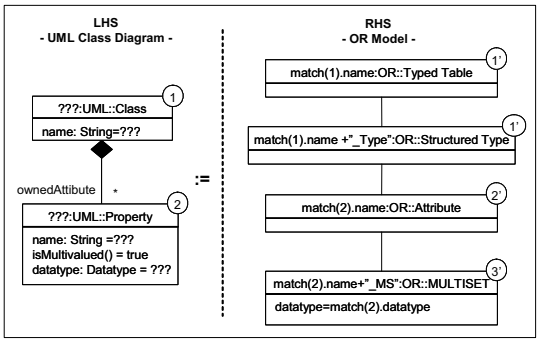


Fig. 6. Transformación de atributos multivaluados

Los **atributos compuestos** de las clases incluidas en el PIM se recogen como tipos *ROW* o tipos estructurados en SQL:2003. Pero al contrario que ocurría con la transformación de las clases *persistent*, este tipo de objeto no da lugar a una tabla tipada, porque su extensión estará contenida en la tabla tipada correspondiente a la clase del PIM que posee el atributo compuesto. La Figura 7 muestra la regla correspondiente.

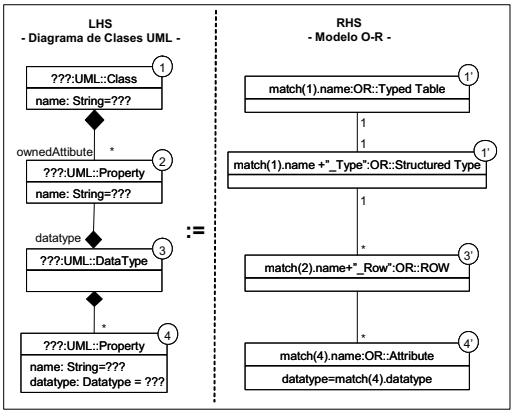


Fig.7. Transformación de atributos compuestos

La clase UML posee una propiedad (①−②) cuyo tipo es a su vez otro tipo de dato con sus propias propiedades

(③−④). Este tipo de datos UML se recoge en el PSM por medio de un tipo *ROW* (③→③'), y cada propiedad del tipo de datos se recoge como un atributo del correspondiente tipo *ROW* (④→④'). Finalmente, el tipo *ROW* sirve como tipo de datos para uno de los atributos del tipo estructurado que representa la clase *persistent* (①'−③').

Los **atributos derivados** son el último tipo de atributos que consideramos. Estos atributos son recogidos en el PSM añadiendo un método o un *trigger* al tipo estructurado que corresponde a la clase *persistent* que incluye el atributo, de manera que el valor del atributo es calculado por dicho subprograma. Esto se plasma gráficamente en la inclusión de la declaración del subprograma en el tipo estructurado.

La figura 8 muestra la regla de transformación correspondiente. En el nivel PIM, el valor *true* para el atributo *isDerived* de la propiedad UML (②) nos dice que se trata de un atributo derivado. En el nivel PSM, el atributo derivado se corresponde con el subprograma asociado al tipo estructurado (①'−②') que representa la clase UML que posee el atributo derivado.

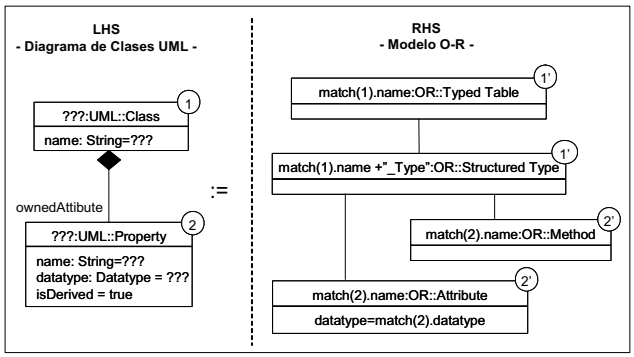


Fig. 8. Transformación de atributos derivados

- **Transformación de Asociaciones**

Las asociaciones UML que encontramos en el nivel PIM son transformadas a relaciones uni- o bidireccionales en el esquema de la BD que obtenemos en el nivel PSM, teniendo en cuenta al coste adicional asociado al uso de relaciones bidireccionales, ya que el sistema no se ocupa de mantener la consistencia de este tipo de asociaciones. Para discernir entre los dos tipos de relaciones, nos fijaremos en la clase de consultas que serán formuladas sobre la asociación en cuestión. Así, en caso de que en la mayoría de los casos la asociación sea navegada en un solo sentido, utilizaremos una relación unidireccional, mientras que si los dos sentidos son navegados por igual, optaremos por una relación bidireccional. Centrándonos en la transformación de asociaciones del PIM al PSM (del modelo conceptual al esquema de la BDOR), la cardinalidad de la asociación es la clave. Así, proponemos diferentes reglas de transformación en función de la multiplicidad máxima de las clases implicadas en la asociación. Para facilitar la comprensión de las reglas definidas, se consideran aquí sólo los casos en los que las asociaciones UML se transforman en relaciones unidireccionales. Téngase presente que lo único que habría que hacer si se quisieran utilizar relaciones bidireccionales es repetir la misma construcción en la parte derecha de la regla. De nuevo, presentamos las reglas de transformación para SQL:2003 e indicamos, cuando sea preciso, las diferencias con la transformación para Oracle10*g*.

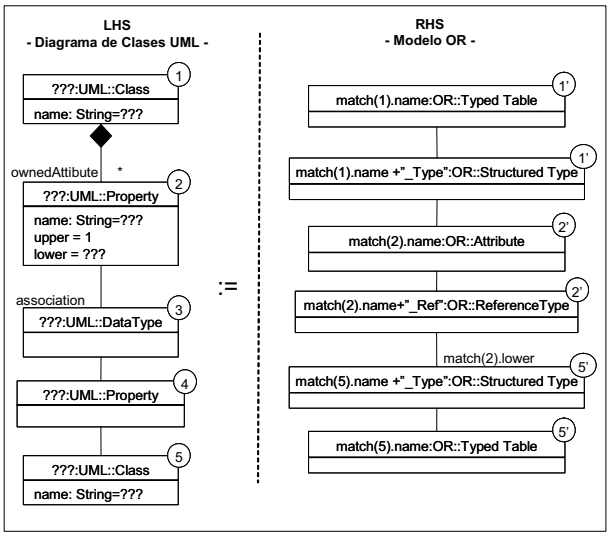


Fig. 9. Transformación de asociaciones con multiplicidad 1

Tal y como muestra la figura 9, las asociaciones de **multiplicidad máxima 1** se identifican en el modelo conceptual por el valor *1* de la propiedad UML (②) perteneciente a la clase origen de la asociación (①). La correspondiente relación unidireccional en el esquema de la BD se construye sobre un atributo del tipo estructurado que mapea la clase origen (①'−②'). Este atributo de tipo *REF* será un puntero a la extensión del tipo estructurado que mapea la clase fin de la asociación (②'−⑤'). Como se puede comprobar en la figura 10, la única diferencia con la transformación de asociaciones de **multiplicidad máxima K** o **N** estriba en el tipo del atributo incluido en el tipo estructurado correspondiente a la clase origen: en el primer caso se utiliza un *ARRAY* de K elementos, mientras que en el segundo se utiliza un *MULTISET*, dado que desconocemos la cantidad exacta de elementos que compondrán esta colección. Estas reglas son igualmente válidas para Oracle10*g* sin más que utilizar *VARRAYS* en lugar de *ARRAYS* y *NESTED TABLES* por *MULTISETS*. Al igual que hemos definido transformaciones para los distintos tipos de atributos que podemos encontrar a nivel PIM, definimos distintas reglas de transformación para los diferentes tipos de asociación UML: generalización, agregación y composición.

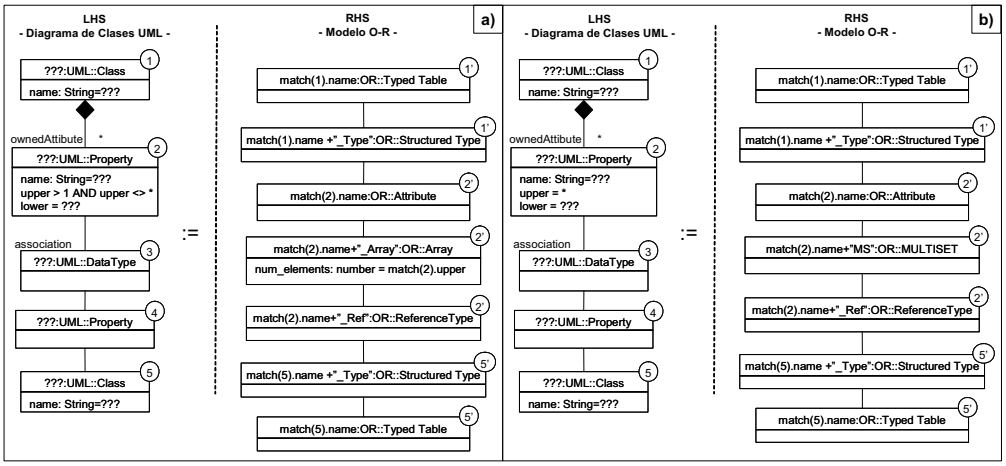


Fig. 10. Transformación de asociaciones (a) con multiplicidad K y (b) multiplicidad N

- **Transformación de Generalizaciones**

Mientras que el estándar SQL:2003 soporta la herencia tanto de tipos como de tablas, Oracle10*g* sólo soporta la herencia de tipos. Así, en Oracle10*g* se pueden definir dos tablas tipadas que compartan la misma estructura: atributos, métodos y tipos de datos para esos atributos y/o métodos, sin más que definir los tipos de dichas tablas como subtipos de un mismo tipo padre. Sin embargo, dado que Oracle no soporta la herencia de tablas, la única forma de que dichas tablas incluyesen las mismas restricciones sería repetir la definición de dichas restricciones sobre cada una de las tablas. Esta desventaja hace que raramente se utilice la generalización cuando se trabaja con la funcionalidad OR de Oracle.

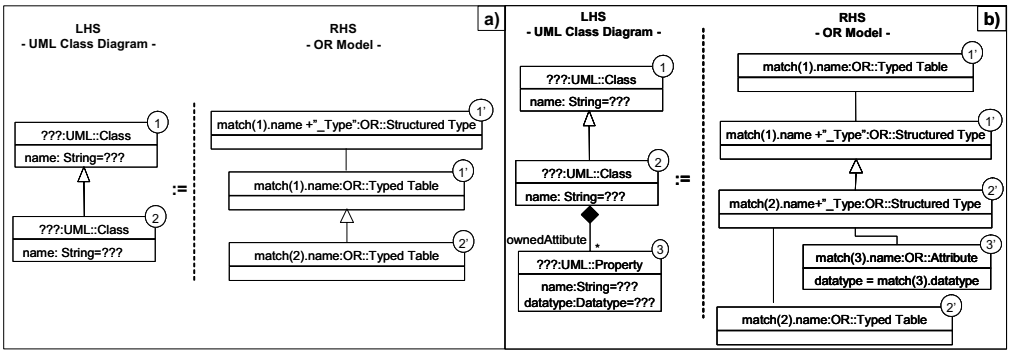


Fig. 11. Transformación de generalizaciones en SQL:2003: (a) generalización simple (b) características adicionales en la clase hija

Centrándonos en la transformación de generalizaciones en SQL:2003, distinguimos dos situaciones diferentes: la primera y más simple, en la que la subclase no añade nuevas características a las ya incluidas en la clase padre (figura 11(a)) y que corresponde a la situación típica en la que necesitamos distinguir entre dos conceptos que comparten la misma estructura, por ejemplo: los tipos cuadrado, rectángulo y rombo son todos subtipos del tipo cuadrilátero. La segunda situación (figura 11(b)), la más compleja, es aquella en la que las subclases incluyen características adicionales a las que heredan del padre en forma de atributos o métodos. Un ejemplo típico podría ser la jerarquía *medios de transporte*, en la que un avión tendría un determinado número de alas, un coche un determinado número de ruedas, etc.

A la hora de transformar una generalización del primer tipo, bastará incluir en el esquema de la BD una tabla tipada que descienda directamente de la tabla que corresponde a la extensión de la clase padre del modelo conceptual. Así, todas las restricciones definidas sobre la tabla padre serán preservadas en la tabla hija. La segunda situación es más complicada puesto que para recoger las características adicionales que aporta la clase hija, es preciso definir un nuevo tipo estructurado en el esquema de la BD. Este tipo será un subtipo del tipo que corresponde a la clase padre del modelo conceptual y incluirá los nuevos atributos o métodos que se definieron en la clase hija.

- **Transformación de Agregaciones/Composiciones**

En ambos casos el paso de agregaciones o composiciones del PIM al PSM (del modelo conceptual UML al esquema de la BDOR) sigue la misma aproximación: la inclusión de un atributo de tipo colección en el tipo estructurado al que se mapea la clase UML que actúa como *todo* en la relación de agregación/composición. En la figura 12 se muestran las reglas de grafos para la transformación de agregaciones (a) y composiciones (b).

En el caso de la agregación, denotada por el valor *shared* que toma el atributo *aggregation* de la propiedad UML (②), dado que las *partes*, esto es, las clases UML asociadas con el *todo*, tienen existencia propia independientemente del *todo*, el tipo colección utilizado en el PSM será una colección de referencias a los objetos que componen la colección, almacenados en su propia tabla tipada (②'–⑤'). Por el contrario, en el caso de la composición, las *partes* sólo tienen existencia dentro del *todo*, así que en este caso el tipo

colección del PSM será una colección de objetos en lugar de una colección de referencias a objetos. Así, en la figura 12(b), que define la regla de transformación para la composición desaparece la segunda tabla tipada y el tipo referencia que se usaba para definir el tipo colección en el caso de la agregación: las *partes* serán incluidas directamente en el objeto *todo*, es decir, en la tabla tipada que mapea la clase UML que actúa como *todo* en la composición (①'–②'–⑤') mientras que en la agregación, estos objetos eran referenciados desde la tabla correspondiente al *todo*.

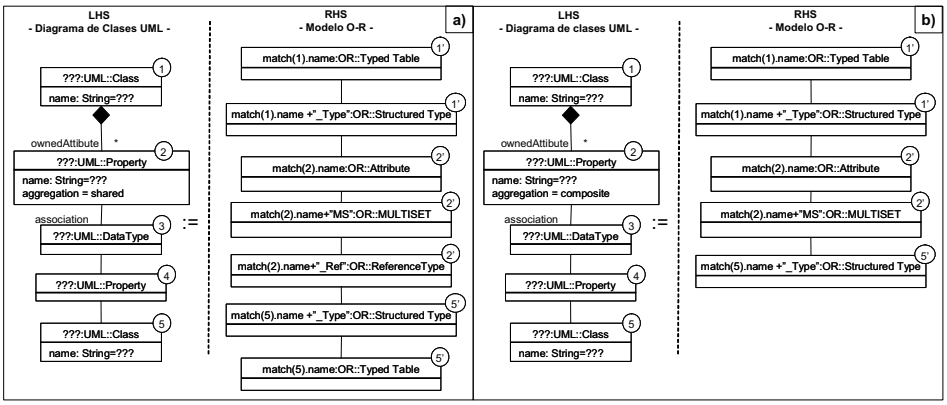


Fig. 12. Transformación de asociación de Agregación (a) y Composición (b)

## IV. CONCLUSIONES

En este artículo se completa la propuesta de desarrollo de BDOR dirigida por modelos en el marco de MIDAS. Para ello se han definido las transformaciones necesarias para pasar del PIM al PSM, es decir, generar el esquema de la BD (PSM) partiendo del modelo conceptual de datos (PIM), representado mediante un diagrama de clases UML. Se han propuesto dos PSM diferentes: el primero para representar el esquema de la BD en el estándar SQL:2003 y el segundo para un producto concreto, Oracle10*g*. Para obtener un proceso de desarrollo completo, se han presentado los metamodelos y los perfiles UML necesarios para elaborar los modelos OR contemplados en el proceso y la definición de las reglas para transformar el PIM en los PSM de datos. En primer lugar, se han resumido las reglas de transformación de una forma declarativa para posteriormente formalizarlas con reglas de grafos y automatizarlas usando alguna de las facilidades existentes para la automatización de transformaciones basadas en grafos.

Este trabajo recalca la importancia de las transformaciones de modelos en el desarrollo software: las transformaciones entre modelos presentadas en este trabajo completan la definición del proceso de desarrollo de BDOR de MIDAS, una propuesta contrastada y publicada que encuentra en la transformación de modelos la pieza que faltaba para convertirse en una metodología completa y factible.

Actualmente estamos trabajando en la integración del proceso de desarrollo de BDOR en una herramienta MDA que integra todas las técnicas propuestas en MIDAS para la generación semiautomática de SIW y se está desarrollando en nuestro grupo de investigación. En trabajos previos [19] ya han sido presentadas tanto su arquitectura como algunas de sus funcionalidades. Igualmente se está abordando la automatización de las transformaciones usando algunas de las tecnologías existentes como AGG.

## V. REFERENCIAS

[1] Bertino, E. y Marcos, E. Object Oriented Database Systems. En: Advanced Databases: Technology and Design, O. Díaz y M. Piattini (Eds.). Artech House, 2000.

[2] Stonebraker, M. y Brown, P. Object-Relational DBMSs. Tracking the Next Great Wave. Morgan Kauffman, 1999.

[3] ISO / IEC 9075 Standard, Information Technology – Database Languages – SQL:2003, International Organization for Standardization, 2003

[4] IBM DB2 Universal Database. En: http://www-306.ibm.com/software/data/db2/.

[5] Microsoft SQL Server. En: http://www.microsoft.org/sql/.

[6] Oracle Corporation. Oracle10*g* Release 2 (10.2). En: www.oracle.com, 2005.

[7] Atzeni, P., Ceri, S., Paraboschi, S. y Torlone R., Database Systems. Concepts, Languages and Architectures. McGraw-Hill, 1999.

[8] Chen, P.P. The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, No. 1. Marzo 1976, pp. 9-36, 1976.

[9] OMG. MDA Guide Version 1.0. Document number omg/2003-05-01. Ed.: Miller, J. y Mukerji, J.http://www.omg.com/mda, 2003.

[10] Marcos, E. Vela, B., Cáceres, P. y Cavero, J.M. MIDAS/DB: a Methodological Framework for Web Database Design. DASWIS 2001. LNCS 2465, Springer-Verlag, pp. 227-238, septiembre, 2002.

[11] Muller R. Database Design for Smarties. Morgan Kaufmann, 1999

[12] Naiburg, E.J. y Maksimchuk R.A. UML for Database Design. Addison-Wesley, 2001.

[13] Vela, B., Acuña, C. y Marcos, E. A Model Driven Approach for XML Database Development, ER2004. Springer Verlag, LNCS 3288, pp. 780-794. 2004.

[14] Marcos, E., Vela, B. y Cavero J.M. Methodological Approach for Object-Relational Database Design using UML. Journal on Software and Systems Modeling (SoSyM). Springer-Verlag. Ed.: R. France y B. Rumpe. Vol. SoSyM 2, pp.59-72, 2003.

[15] Tratt, L. Model transformations and tool integration, Software and Systems Modeling, Vol. 4, Issue 2, mayo 2005, Pp. 112 – 122, 2005.

[16] Ambler, S. Agile Database Techniques. Wiley, 2003.

[17] Buttner, F. y Gogolla, M. Realizing UML Metamodel Transformations with AGG, Proceedings of GT-VMT. Electronic Notes in Theoretical Computer Science. Vol. 109, diciembre 2004, pp. 31-42. 2004.

[18] Muñoz, J. Una Gramática de Grafos para la Transformación de Relaciones de Asociación desde Modelos de Análisis hacia Modelos de Diseño. Technical Report, DSIC-UPV 2004.

[19] Vara, J.M., De Castro, V. y Marcos, E. WSDL automatic generation from UML models in a MDA framework In International Journal of Web Services Practices. Vol.1 – Issue 1 & 2, pp.1–12. Noviembre 2005.

## VI. BIOGRAFÍAS

**Juan Manuel Vara Mesa** es Ingeniero en Informática por la Universidad Rey Juan Carlos, donde realiza actualmente su Tesis Doctoral en el área del Desarrollo Software Dirigido por Modelos. Ha trabajado durante un año en el Departamento de Sistemas de la empresa Viva Tours y actualmente es profesor ayudante del departamento de Lenguajes y Sistemas Informáticos de la Universidad Rey Juan Carlos, donde su docencia se centra en temas relacionados con la Ingeniería del Software y las Bases de Datos. Igualmente participado como docente en varios masters y cursos de especialización y en la actualidad es profesor del Máster de Sistemas de Información y Comunicación para la Defensa de la Escuela de Informática para el Ejército. Es coautor de numerosas publicaciones en congresos nacionales e internacionales y ha participado en numerosos proyectos de investigación a nivel local, regional, nacional y europeo.

**Belén Vela Sánchez** es doctora por la Universidad Rey Juan Carlos e Ingeniera Informática por la Universidad Carlos III de Madrid. Actualmente es profesora colaboradora en el departamento de Lenguajes y Sistemas Informáticos de la Universidad Rey Juan Carlos. Ha sido profesora invitada en la Fachbereich Informatik de la Fachhochschule de Hannover (Alemania). Ha trabajado durante 2 años como consultora en la empresa privada (Cronos Ibérica S.A. y PriceWaterhouseCoopers).

Ha impartido clases en numerosos cursos y masters de especialización. Actualmente es profesora del Máster en Ingeniería de la Decisión impartido en la Universidad Rey Juan Carlos y el Máster de Sistemas de Información y Comunicación para la Defensa de la Escuela de Informática para el Ejército. Es coautora del libro "Tecnología y Diseño de Bases de Datos" (Ra-ma, 2006) y de numerosas publicaciones en revistas y congresos nacionales e internacionales. A su vez, también ha participado en numerosos proyectos de investigación, siendo la investigadora principal en dos de ellos.

**José María Cavero Barca** es licenciado en Informática por la Facultad de Informática de la Universidad Politécnica de Madrid y doctor por la Universidad Rey Juan Carlos. Actualmente es profesor titular de escuela universitaria en el departamento de Lenguajes y Sistemas Informáticos de la Universidad Rey Juan Carlos.

Ha impartido clases en numerosos cursos y masters de especialización. Actualmente es profesor del Máster en Ingeniería de la Decisión y el Máster en Tecnologías de la Información y Sistemas Informáticos de la Universidad Rey Juan Carlos y el Máster de Sistemas de Información y Comunicación para la Defensa de la Escuela de Informática del Ejército. Es autor de numerosas publicaciones en revistas y congresos nacionales e internacionales, y ha participado en numerosos proyectos de investigación. Sus temas de investigación incluyen: modelado conceptual, diseño de bases de datos y ontologías.

**Esperanza Marcos Martínez** es Ingeniera y Doctora en Informática por la Universidad Politécnica de Madrid y Diplomada en Informática por la Universidad de Valladolid. Durante cinco años ha sido profesora del Departamento de Informática de la Universidad Carlos III de Madrid. Ha sido representante de AENOR en los comités internacionales del ISO/IEC JTC1/SC21 WG3 DBL sobre estandarización del lenguaje SQL.

Actualmente es Profesora Titular de la Escuela Superior de Ciencias Experimentales y Tecnología (ESCET) de la Universidad Rey Juan Carlos donde coordina el Grupo Kybele cuya investigación se centra en Ingeniería de Sistemas de Información, abordando temas de desarrollo dirigido por modelos, Sistemas de Información Web, etc. Ha impartido clases en numerosos cursos y Masters de especialización; en la actualidad colabora en el Máster en Ingeniería de la Decisión impartido en la Universidad Rey Juan Carlos y es coordinadora académica del Máster de Sistemas de Información y Comunicaciones para la Defensa de la Escuela de Informática del Ejército.

Es coautora de numerosos libros y capítulos de libros, así como de más de un centenar de artículos en congresos y revistas nacionales e internacionales. Ha participado y dirigido numerosos proyectos de investigación.