JISBD2007-02: Model-driven reverse engineering for data warehouse design

J.-N. Mazón y J. Trujillo

***Abstract*— Data warehouses integrate several operational sources to provide a multidimensional analysis of data, thus improving the decision making process. Therefore, an in-depth analysis of these data sources is crucial for data warehouse development. Traditionally, this analysis has been based on a set of informal guidelines or heuristics to support the manually discovery of multidimensional elements on a well-known documentation. Therefore, this task may become highly tedious and prone to fail. In this paper, MDA (Model Driven Architecture) is used to design a reverse engineering process in which the following tasks are performed (i) obtain a logical representation of data sources (ii) mark this logical representation with multidimensional concepts, and (iii) derive a conceptual multidimensional model from the marked model.**

***Keywords*— MDA, reverse engineering, data warehouse, multidimensional modeling.**

I. Introducción

LOS almacenes de datos (AD) integran fuentes de datos heterogéneas en estructuras multidimensionales (MD), como hechos y dimensiones, con el fin de facilitar la información necesaria para el proceso de toma de decisiones de una organización [6,8].

En los últimos años, se han definido varias propuestas para acometer un análisis de las fuentes de datos que permita descubrir en ellas los diversos elementos MDs con el fin de (i) derivar directamente un modelo conceptual MD a partir de las fuentes de datos [4,5] o (ii) reconciliar las fuentes de datos con un modelo conceptual MD previamente definido a partir de las necesidades de información de los usuarios [3,10].

Sin embargo, todas estas propuestas sugieren mecanismos poco automatizados basados en guías de diseño informales para descubrir los elementos MD en las fuentes de datos, por lo que esta tarea resulta excesivamente tediosa y muy sensible a errores. Por otro lado, estas propuestas asumen la existencia de una documentación de las fuentes de datos (p.e. mediante diagramas Entidad-Relación) a partir de la cual se lleva a cabo el análisis de las mismas. Desafortunadamente, en proyectos de ADs reales, las fuentes de datos operacionales son verdaderos sistemas heredados que necesitan del concurso de técnicas de ingeniería inversa para su análisis, bien sea porque la documentación no existe, porque es excesivamente compleja o porque está obsoleta.

Teniendo en cuenta los problemas existentes, en este trabajo se propone el uso de MDA (*Model Driven Architecture*) [15] para realizar de manera automática las siguientes tareas de ingeniería inversa en el desarrollo del AD: (i) analizar las fuentes de datos existentes para obtener su representación mediante un modelo lógico, (ii) descubrir elementos MDs en este modelo, obteniendo así un modelo lógico cuyos elementos están marcados con conceptos MDs, (iii) utilizar el modelo marcado para obtener directamente un modelo conceptual MD o para reconciliar las fuentes de datos con un modelo conceptual MD previamente obtenido a partir de los requisitos de usuario. El modelo conceptual MD del AD guiará las siguientes etapas de diseño hasta obtener la implementación final del AD.

En este trabajo nos centramos en la obtención del modelo conceptual MD directamente a partir de las fuentes de datos operacionales. No obstante, el modelo lógico marcado de manera automática sirve para completar nuestro trabajo previo [10], donde se asumía que el modelo lógico de las fuentes de datos había sido marcado manualmente con conceptos MD. Por otra parte, el proceso de implementación del AD a partir de un modelo conceptual se propuso en [11,12,13].

El presente artículo queda estructurado de la siguiente manera: la parte II describe brevemente el estado de la cuestión en cuanto al descubrimiento de elementos MDs en las fuentes de datos operacionales; en la parte III se describen varios conceptos sobre ingeniería inversa y MDA; la parte IV presenta nuestra propuesta de uso de ingeniería inversa y MDA en ADs, mientras que en la última parte se comentan las conclusiones y se plantean trabajos futuros.

II. Trabajos relacionados

Existen varias propuestas para descubrir elementos MDs mediante el análisis de las fuentes de datos con el fin de derivar un modelo conceptual MD que guíe la implementación del AD. La mayoría de estas propuestas [4,5,21] sugieren el uso de algoritmos o guías de diseño para especificar un modelo conceptual MD a partir de unas fuentes operacionales

Este trabajo ha sido financiado por los proyectos METASIGN (TIN2004-00779) del Ministerio de Educación y Ciencia y DADS(PBC-05-012-2) de la Consejería de Educación y Ciencia de Castilla-La Mancha. J.-N. Mazón dispone de una beca FPU (AP2005-1360) del Ministerio de Educación y Ciencia.

Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante, Carretera de San Vicente del Raspeig s/n, 03690, Alicante. (Correos e.: jnmazon@dlsi.ua.es; jtrujillo@dlsi.ua.es).

descritas mediante un modelo Entidad-Relación (ER). Sin embargo, estas propuestas se aplican de manera manual para descubrir ciertos elementos MD y sólo el descubrimiento de jerarquías de dimensión estrictas se realiza automáticamente (navegando a través de las relaciones uno-a-muchos en el modelo ER). Esto hace que las propuestas actuales sean muy costosas de aplicar, sobre todo si, entre los desarrolladores, no se cuenta con expertos en el desarrollo de ADs. Además, estas propuestas asumen que se dispone de diagramas ER bien documentados, siendo imposible su aplicación si estos diagramas no existen o no se pueden obtener. Desafortunadamente, las fuentes de datos operacionales son auténticos sistemas heredados y la documentación no se encuentra generalmente disponible [1]. Además, si las fuentes de datos son muy complejas, aunque la documentación exista, puede que no sea fácilmente comprensible o esté obsoleta.

Por otro lado, existen dos propuestas interesantes que intentan mejorar estos problemas. En [7], se presenta un conjunto de algoritmos para descubrir estructuras MD en las fuentes de datos. Esta propuesta define una etapa de ingeniería inversa para obtener metadatos de las fuentes operacionales. Sin embargo, estos metadatos se deducen de las instancias de los datos, lo que es inviable cuando las fuentes de datos tienen un tamaño considerable o cuando éstas son muy complejas. Además, esta propuesta no se basa en estándares, lo que podría afectar a su aplicabilidad. La propuesta presentada en [22] automatiza la identificación en las fuentes de datos de conceptos MD relacionados con requisitos de usuario. No obstante, esta propuesta parte de la base de que los usuarios pueden expresar fácilmente sus requisitos de información como sentencias SQL, lo que puede no ser cierto cuando las fuentes operacionales son complejas.

Por tanto, la aplicación de las propuestas actuales es inviable en determinadas situaciones. Con el fin de superar estas limitaciones, en este artículo se propone el uso de MDA en el desarrollo del AD para describir una etapa de ingeniería inversa con el fin de (i) obtener un modelo lógico a partir de la implementación de las fuentes de datos, (ii) marcar automáticamente el modelo lógico de las fuentes de datos con conceptos MDs y (iii) derivar automáticamente un modelo conceptual MD del modelo lógico marcado.

III. MDA E INGENIERÍA INVERSA

MDA (*Model Driven Architecture*) [15] es un estándar del OMG (*Object Management Group*) dirigido al ciclo de vida completo de diseño, despliegue, integración y gestión de aplicaciones mediante el uso de modelos en el desarrollo del software. MDA separa la especificación de la funcionalidad de un sistema de la especificación de la implementación de dicha funcionalidad en un plataforma tecnológica determinada mediante la definición de diferentes modelos a diferentes niveles de abstracción. Por lo tanto, MDA fomenta la definición de un modelo independiente de la plataforma (*Platform Independent Model*, PIM) el cual no contiene información específica sobre ninguna plataforma o tecnología. Este PIM se transforma en un modelo específico de la plataforma (*Platform Specific Model*, PSM) con el fin de incluir información sobre una tecnología específica. Cada PSM se transforma en código para obtener la implementación final del sistema. Estos modelos suelen definirse usando metamodelos como como UML (*Unified Modeling Language*) [20] o CWM (*Common Warehouse Metamodel*) [17]. Las transformaciones entre modelos se desarrollan mediante lenguajes como QVT (*Query/View/Transformation*) [16].

Por otro lado, la ingeniería inversa consiste en el análisis de un sistema de información existente con el fin de obtener un modelo de dicho sistema a más alto nivel de abstracción, por ejemplo un modelo conceptual, que facilite la evaluación y mantenimiento de los sistemas de información existentes o el desarrollo y evolución de nuevos sistemas de información [2]. Por tanto, los conceptos de MDA se pueden usar para facilitar las tareas llevadas a cabo por la ingeniería inversa, ya que permiten el análisis automático de los sistemas de información existentes con el fin de obtener sus correspondientes modelos a diferentes niveles de abstracción. Es decir, se pueden usar mecanismos proporcionados por MDA para obtener automáticamente un PIM de un PSM que ha sido obtenido previamente de la implementación del sistema de información.

IV. MDA E INGENIERÍA INVERSA PARA ALMACENES DE DATOS

El desarrollo de un AD comienza cuando las fuentes de datos operacionales no pueden suministrar a los usuarios la información necesaria para el apoyo a la toma de decisiones. Es en ese momento cuando se plantea la construcción de un AD que facilite la ejecución de consultas MDs complejas.

Por tanto, el análisis de las fuentes de datos y su relación con elementos MDs juega un papel muy importante en el desarrollo del AD. Ésta es una tarea de ingeniería inversa que consiste en analizar las fuentes de datos disponibles para descubrir las estructuras MD y derivar un modelo conceptual MD que guíe las siguientes etapas de diseño hasta la implementación final del AD.

Esta tarea de ingeniería inversa se realiza mediante la aplicación de MDA, de tal manera que (i) las fuentes de datos se analizan y se especifica su representación lógica en un PSM, (ii) el PSM se marca con conceptos MDs mediante la asociación de cada uno de sus elementos con elementos MDs y (iii) se deriva un PIM (modelo conceptual MD) a partir del PSM marcado. Para llevar a cabo de manera automática tanto el descubrimiento de elementos MDs para marcar el PSM como la obtención del PIM, se han desarrollado un conjunto de relaciones QVT. La figura 1 resume nuestra aproximación.

La implementación de la propuesta se ha llevado a cabo en la herramienta CASE Borland Together Architect, basada en Eclipse, mediante la definición de los siguientes elementos : (i) un *plugin* para obtener el PSM a partir de la implementación de las fuentes operacionales mediante el uso del paquete *java.sql* de Java, (ii) un *plugin* de CWM para la definición del PSM, (iii) un *profile* para el modelado MD de AD que permite la especificación del PIM y (iv) el conjunto de relaciones QVT necesario para obtener el PSM marcado y el PIM. Todos estos elementos se comentan en las siguientes

secciones.

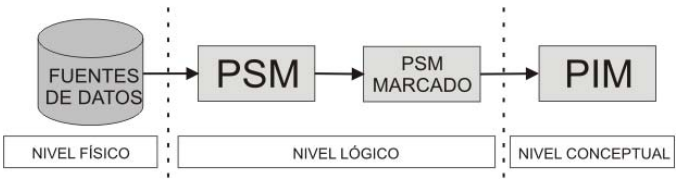

Fig. 1. Resumen de nuestra propuesta.

A. Obtención del PSM

En esta sección se describe como obtener un PSM de las fuentes de datos operacionales existentes. Normalmente, las fuentes de datos se implementan siguiendo una amplia variedad de técnicas, tales como bases de datos relacionales, esquemas XML o incluso ficheros de texto. En este trabajo nos ceñimos a un escenario concreto muy común, en el cual una base de datos relacional ha sido implementada en un sistema gestor de bases de datos (SGBD) relacional Oracle. Dentro de un SGBD relacional, toda la información sobre estructuras de datos se almacena en un diccionario de datos, a partir del cual se puede obtener un modelo lógico relacional de la base de datos implementada, es decir un PSM según una plataforma relacional. La definición de este PSM se realizará mediante el metamodelo relacional de CWM.

Además de este PSM, del diccionario de datos se obtienen varias medidas, las cuales se emplearán posteriormente en un conjunto de relaciones QVT para automatizar el marcado del PSM de manera correcta.

1) Diccionario de datos de Oracle

Un diccionario de datos contiene los metadatos de los elementos almacenados en una base de datos. Cada SGBD tiene su propio método para almacenar información en un diccionario de datos. Concretamente, el diccionario de datos de Oracle consiste en un conjunto de tablas en las cuales se almacenan los metadatos. El diccionario de Oracle tiene dos niveles: el nivel interno que contiene las propias tablas del diccionario de datos y el nivel externo que suministra una serie de vistas que permiten el acceso a los metadatos por parte del usuario. Por lo tanto, los metadatos de la base de datos se pueden extraer del diccionario de datos mediante la ejecución sobre el nivel externo de las consultas SQL requeridas.

El nivel externo del diccionario de datos de Oracle contiene tres tipos de vistas: *USER* (contiene metadatos de elementos que pertenecen al usuario actual), *ALL* (contiene metadatos de los elementos accesibles por el usuario actual) y *DBA* (contiene metadatos de todos los elementos de la base de datos). Con el fin de usar las vistas *USER*, se asume que las fuentes de datos operacionales pertenecen al usuario actual y que este usuario sólo posee los elementos contenidos en estas fuentes de datos. En nuestra propuesta se considera el siguiente subconjunto de vistas *USER*:

- *USER_TABLES*. Suministra datos sobre las tablas pertenecientes a cada usuario.
- *USER_TAB_COLUMNS*. Suministra datos sobre las columnas de cada tabla que pertenece a un determinado usuario.
- *USER_CONSTRAINTS*. Suministra datos sobre las restricciones que posee una tabla.
- *USER_CONS_COLUMNS*. Suministra datos sobre las restricciones que están relacionadas con cada columna de cada tabla.

Con el fin de derivar el PSM relacional de las fuentes de datos operacionales, se deben obtener sus metadatos mediante diversas consultas SQL sobre estas vistas del diccionario de datos. Cabe destacar que se asume que la base de datos existente fue desarrollada teniendo en cuenta la definición de cada restricción (tales como claves primarias o ajenas) necesaria para implementar un esquema en tercera forma normal. Existen diversos trabajos que permiten descubrir fácilmente estas restricciones (p.e. [23]), por lo que es posible detectarlas aun no estando definidas.

2) PSM relacional

El metamodelo relacional de CWM [17] es un estándar para representar la estructura de una base de datos relacional, permitiendo especificar tablas, columnas, claves primarias, claves ajenas, etc. Un extracto de este metamodelo relacional se muestra en la figura 2.

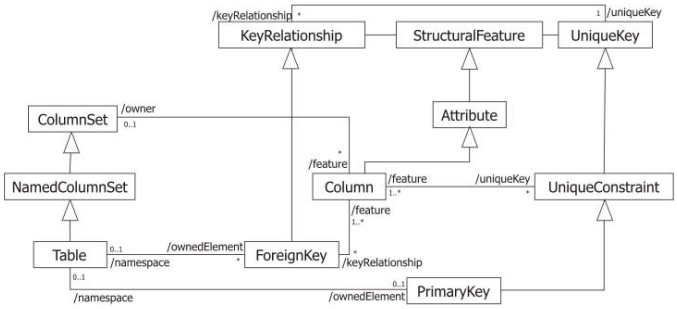

Fig. 2. Metamodelo relacional de CWM.

En nuestra aproximación, este metamodelo relacional se usa para especificar una representación relacional de las fuentes de datos (es decir, un PSM) a partir del diccionario de datos Oracle.

Junto con este PSM, se obtienen también, a partir de la base de datos existente, un conjunto de tres medidas que se usarán en las relaciones QVT destinadas a marcar el PSM como paso previo a la obtención del PIM. Concretamente, estas medidas se usan en un conjunto de heurísticas (definidas más adelante) para saber si un elemento de las fuentes de datos representa un hecho. Se obtienen las siguientes medidas:

- Número de instancias por tabla (NIT). Se obtiene fácilmente para cada tabla que está definida en la vista *USER_TABLES* del diccionario de datos mediante la ejecución de una sentencia SQL simple que haga un recuento de las instancias de cada tabla.
- Frecuencia de inserción de cada tabla (IFT). Es la media de inserciones en un periodo concreto de

tiempo (p.e. por día) en una tabla concreta. Se calcula dividiendo el número total de inserciones entre el tiempo (p.e. número de días) que lleva creada la tabla.

- Número de medidas de cada tabla (NMT). Es la cantidad de columnas de tipo numérico que tiene una tabla.

3) Del diccionario de datos al PSM

El proceso de obtención de un PSM relacional a partir del diccionario de datos de Oracle se ha implementado mediante el uso de Java dentro de la plataforma Eclipse. Concretamente, se ha usado la interface *java.sql.Connection* para conectar con la base de datos Oracle y ejecutar las sentencias SQL requeridas para obtener los metadatos necesarios del diccionario de datos. A partir de estos metadatos, el PSM correspondiente se obtiene usando un *plugin* de CWM desarrollado para Eclipse.

Para ilustrar nuestra aproximación, se muestra un caso de estudio extraído de un proyecto real de ADs llevado a cabo por el grupo de investigación IWAD de la Universidad de Alicante. Este proyecto está relacionado con el sistema de información de un hotel. Nos centramos en las fuentes operacionales que apoyan el proceso de negocio de las reservas. Dentro de este proceso, un cliente reserva una habitación en un hotel para una fecha concreta. Cada una de estas reservas contiene los siguientes datos: número de noches, precio por noche, descuento y coste total de una reserva. Del cliente se almacena información sobre su número de DNI, nombre, sexo, fecha de nacimiento, ciudad y país. Se sabe también el nombre y el número de habitantes de la ciudad y el nombre del país. Finalmente, una habitación del hotel se encuentra en una planta concreta y tiene una superficie. Una habitación puede ser de dos tipos, individual o doble. Para las habitaciones individuales se almacena el tamaño de la cama y para las habitaciones dobles, la posibilidad de contar con una cama supletoria.

La obtención de los metadatos de las fuentes de datos que apoyan a este proceso de negocio se realiza a partir de la consulta del diccionario de datos de Oracle. Después de obtener todos los metadatos, el PSM se deriva mediante el uso de las facilidades de modelado que proporciona Eclipse y de un *plugin* desarrollado para CWM. Se usa la herramienta Borland Together Architect que está basada en Eclipse. La figura 3 muestra el PSM derivado del sistema de información del hotel.

B. Obteniendo un PIM

Una vez que se ha obtenido un PSM (junto con su correspondiente conjunto de tres medidas), se debe derivar del mismo un PIM que represente el modelo conceptual MD. Este proceso de derivación consta de dos tareas: marcado del PSM y obtención del PIM a partir del modelo marcado. La primera tarea consiste en la asociación de cada elemento del PSM con un concepto MD con el fin de facilitar la segunda tarea donde se deriva el PIM teniendo en cuenta estructuras MD complejas. Con el fin de realizar ambas tareas de manera automática se han definido una serie de relaciones QVT.

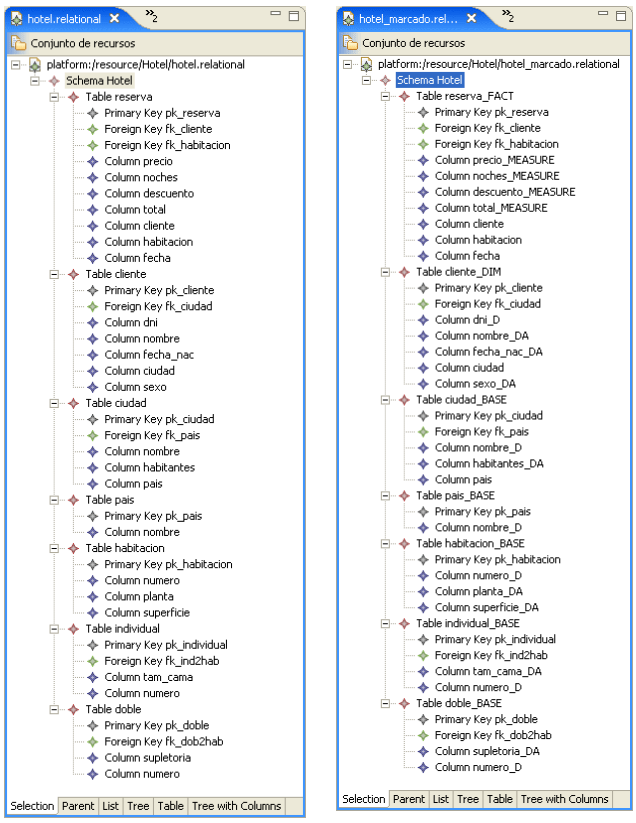

Fig. 3. PSMs sin marcar y marcado.

Llegados a este punto, cabe destacar que como paso final una vez obtenido el PIM, se debe añadir una dimensión temporal, aunque ésta no aparezca de manera explícita en las fuentes de datos operacionales, ya que un AD siempre guarda información histórica [8].

1) PIM multidimensional

El PIM para el AD se desarrolla siguiendo nuestro *profile* de UML para modelado conceptual MD [9]. En la figura 4 se presentan de manera general los principales estereotipos de nuestro *profile*. En concreto, las propiedades MDs se representan mediante un diagrama de clases en el cual la información se organiza en hechos y dimensiones, representados respectivamente por una clase estereotipada como *Fact* y una clase estereotipada como *Dimension*. Una clase *Fact* se define como una composición de *n* clases *Dimension* en una agregación compartida (*shared aggregation*). La cardinalidad mínima del rol de la clase *Dimension* es 1 para indicar que cada hecho debe relacionarse siempre con cada una de las dimensiones. Sin embargo, las relaciones muchos-a-muchos entre un hecho y una dimensión concreta se puede especificar mediante la cardinalidad 1..n en el rol de la clase *Dimension* correspondiente. Un hecho está compuesto de medidas, representadas como atributos del hecho con el estereotipo *FactAttribute*. Respecto a las dimensiones, cada nivel de jerarquía se especifica mediante una clase estereotipada como *Base*. Cada una de estas clases

Base puede contener varios atributos de dimensión representados mediante el estereotipo *DimensionAttribute* y deben contener una propiedad estereotipada como *Descriptor*. Una asociación entre clases *Base* con un estereotipo *Rolls-UpTo*, especifica la relación existente entre dos niveles de una jerarquía de dimensión. Además, se usan roles para determinar la manera en la que dos clases *Base* se relacionan: el rol *R* representa la dirección en la cual la jerarquía se agrega (*roll-up*), mientras que el rol *D* representa la dirección en la que la jerarquía se desagrega (*drill-down*). Gracias a la flexibilidad de UML, se pueden considerar también jerarquías no estrictas (un objeto de un nivel bajo de jerarquía pertenece a más de un objeto de alto nivel) o jerarquías de generalización (un nivel de jerarquía puede tener subtipos para modelar características especiales). Estas jerarquías se consideran respectivamente mediante la cardinalidad en los roles de las asociaciones y mediante la relación de generalización de UML.

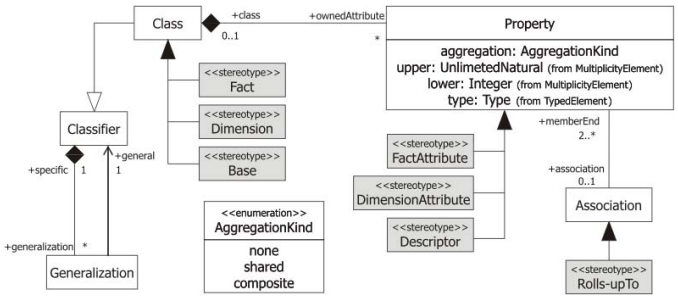

Fig. 4. Extensión de UML con estereotipos para el modelado MD de ADs.

En nuestro *profile* se usa OCL (*Object Constraint Language*) [18] con el fin de expresar reglas bien formadas para los nuevos elementos definidos, evitando así su uso arbitrario. Se remite al lector a [9] para una explicación más detallada de este *profile*.

2) QVT para marcar un PSM

Para reducir la complejidad de obtener un PIM para el modelado MD a partir del PSM, se necesita, como paso previo, relacionar cada elemento del PSM con conceptos MDs, mediante un marcado de los mismos. El marcado de modelos es una técnica que permite capturar información adicional en los modelos con el fin de guiar una transformación [14,15]. Una marca representa un concepto de un modelo que se puede aplicar a un elemento de otro modelo diferente.

En nuestra propuesta, el PSM se marca mediante la adición de un sufijo al nombre de cada elemento según los elementos MDs del *profile* UML descrito anteriormente (ver tabla I), con el fin de indicar la relación de cada elemento del PSM con cierto elemento MD.

Para realizar este marcado se ha definido una transformación QVT que identifica en el PSM los elementos MDs. Para que el marcado sea automático, se usan varias heurísticas que hacen uso de las medidas definidas en la sección IV.A.1. Esta transformación QVT ha sido implementada en Borland Together Architect haciendo uso del *plugin* para CWM. El PSM marcado resultante después de aplicar esta transformación QVT a nuestro caso de estudio se muestra en la figura 3.

TABLA I
MARCAS APLICADAS AL PSM

Marca	PSM	PIM
FACT	Table	Fact
DIM	Table	Dimension
BASE	Table	Base
MEASURE	Column	FactAttribute
DA	Column	DimensionAttribute
D	Column	Descriptor

A continuación se describe como descubrir los diferentes elementos MDs en el PSM: primero se descubren los hechos (y sus respectivos atributos), luego las dimensiones y sus jerarquías (también con sus respectivos atributos). Debido a restricciones de espacio solo se muestra un ejemplo de relación QVT: *DiscoverDimensions*.

a) Descubriendo hechos

Un hecho se relaciona con un proceso de negocio que se apoya en unas determinadas fuentes de datos operacionales [8]. Estas fuentes de datos almacenan medidas de los procesos de negocio útiles para facilitar el proceso de toma de decisiones, por lo que un hecho contiene todas las medidas de las fuentes de datos relacionadas con el proceso de negocio. Una tabla en el PSM se marca como un hecho según las heurísticas siguientes (basadas en las medidas descritas en la sección IV.A.2):

- El tamaño de la tabla es mucho mayor que el de otras tablas, debido a que almacena datos sobre actividades relacionadas con un proceso de negocio. Esta heurística se aplica mediante el uso de la medida NIT.
- La frecuencia de inserciones de la tabla es mayor que el de las otras tablas, lo que refleja que almacene datos de eventos dinámicos dentro del proceso de negocio. Esta heurística se aplica usando la medida IFT.
- La tabla almacena medidas relacionadas con el proceso de negocio. Esta heurística se aplica mediante el uso de la medida NMT.

Estas heurísticas se incluyen en las relaciones QVT para facilitar el marcado de elementos relacionales del PSM como hechos.

b) Descubriendo atributos del hecho

Cuando una tabla corresponde a un hecho, sus columnas pueden ser tanto claves ajenas, que hacen referencia a una tabla que corresponderá a una dimensión, como atributos del hecho, que representan medidas cuyos valores son analizados para apoyar el proceso de toma de decisión. Por lo tanto, una

columna que pertenece a una tabla marcada como hecho se marca como atributo de hecho siempre y cuando (i) su tipo sea numérico y (ii) no sea una clave ajena.

c) Descubriendo dimensiones

Las dimensiones representan el contexto de análisis de un hecho, determinando a las medidas que contiene. Por tanto, una tabla se marca como una dimensión si una tabla marcada como hecho tiene una columna que, a la vez, forma parte de la clave primaria y es una clave ajena a dicha tabla. Esto se define en la relación *DiscoverDimensions* (ver figura 5). Esta relación se ejecuta después de marcar los hechos, por lo que la tabla I se supone que está marcada como un hecho.

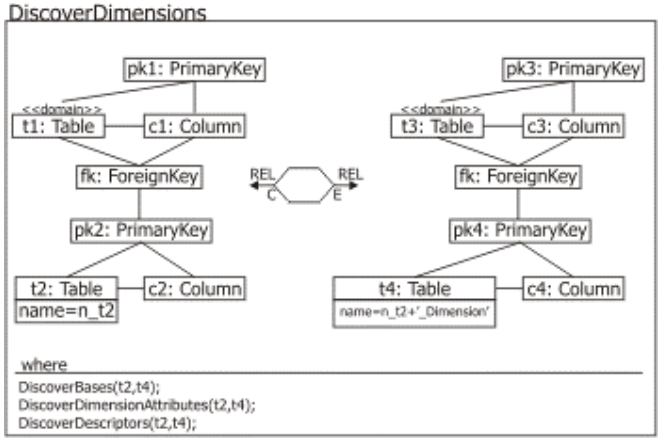

Fig. 5. Marcado de una tabla como dimensión.

d) Descubriendo bases

Cada tabla que no ha sido marcada como hecho o como dimensión, se marca como base. Más tarde, cuando se derive el PIM, las relaciones entre estas tablas serán útiles para obtener los diferentes tipos de jerarquías.

e) Descubriendo atributos de dimensión y descriptores

Las columnas de cada tabla marcada como dimensión o como base se consideran descriptores, si son parte de la clave primaria de la tabla, o atributos de dimensión, si no lo son.

3) QVT para obtener un PIM

Una vez que se obtiene el PSM marcado, se aplica un conjunto de relaciones QVT para derivar un PIM para modelado MD, mediante la obtención de diferentes estructuras MDs como hechos y sus relaciones con las dimensiones o diferentes tipos de jerarquías de dimensión. Estas relaciones QVT han sido implementadas en Borland Together Architect haciendo uso de un *plugin* para CWM y de la implementación del *profile* descrito en la sección IV.B.1. Después de aplicar estas relaciones QVT al PSM marcado de nuestro caso de estudio, se obtiene el PIM de la figura 6. En esta sección se muestra un ejemplo de relación QVT.

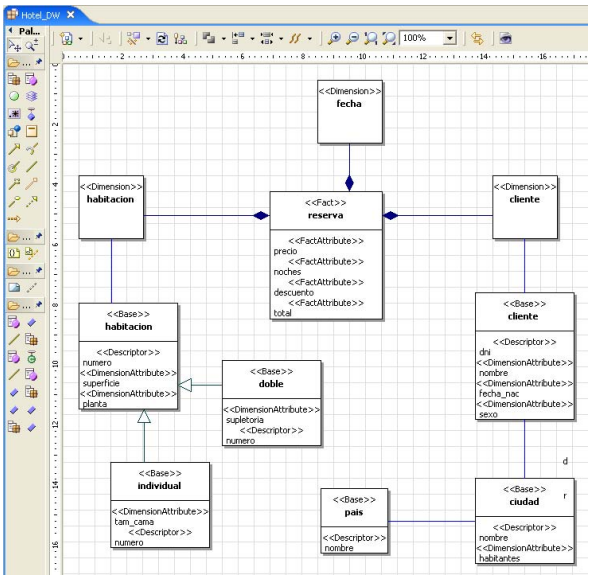

Fig. 6. PIM obtenido con nuestra aproximación.

a) Obtener hechos y dimensiones

A partir del PSM marcado, se obtienen fácilmente las clases hecho y dimensión (según nuestro *profile* de UML para modelado conceptual) y sus cardinalidades, para saber como están relacionados. Aparecen dos posibilidades:

- Relación muchos-a-uno: la cardinalidad mínima y máxima en el rol de la dimensión es 1 para indicar que el hecho debe estar relacionado con cada dimensión sólo una vez. Esta cardinalidad aparece en el PSM marcado cuando una tabla marcada como hecho tiene una clave ajena a una tabla marcada como dimensión.
- Relación muchos-a-muchos: se especifica mediante la cardinalidad 1..n en el rol de la dimensión correspondiente. Esta cardinalidad aparece en el PSM cuando una tabla marcada como hecho tiene dos claves ajenas: una que hace referencia a una tabla marcada como dimensión y otra que hace referencia a una tabla marcada como hecho. Las columnas que forman parte de esta clave ajena, forman, a su vez, la clave primaria de la tabla, representando una tabla puente [8].

b) Obteniendo jerarquías

Comenzando por una dimensión ya descubierta, se pueden descubrir los diferentes niveles de jerarquía navegando por las claves ajenas. Se han desarrollado diferentes relaciones QVT para derivar cada tipo de jerarquía. Debido a restricciones de espacio, nos centramos en las jerarquías de generalización. Estas jerarquías se describen en un PSM relacional marcado haciendo de la clave primaria del supertipo una clave ajena en las tablas que representan los subtipos tal y como se muestra en la relación *ObtainGeneralizedHierarchies* de la figura 7. Un ejemplo de este tipo de jerarquía se muestra en el caso de estudio (ver figura 3), donde la clave primaria de la tabla *"habitación"* que representa un supertipo es referenciada por

las tablas *"doble"* e *"individual"* que representan los subtipos. Esto se representa como una jerarquía de generalización en el PIM (figura 6).

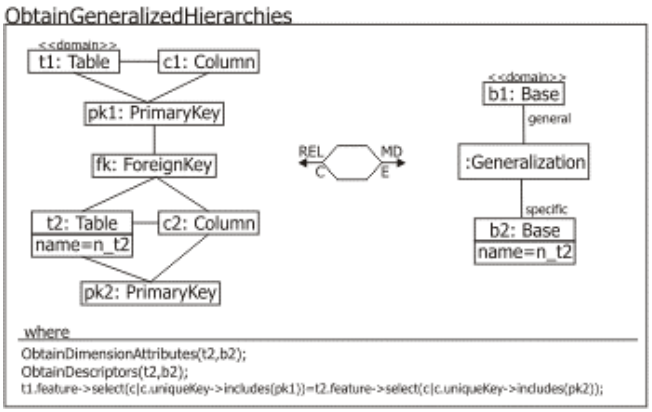

Fig. 7. Obtener jerarquías de generalización.

V. CONCLUSIONES

En este artículo se ha presentado una aproximación basada en ingeniería inversa dirigida por modelos para el desarrollo de ADs. Esta aproximación consiste en las siguientes tareas: (i) las fuentes de datos son analizadas y su representación lógica se especifica en un PSM, (ii) el PSM se marca con conceptos MDs y , finalmente, (iii) un PIM que representa un modelo conceptual MD se deriva del PSM marcado. Para definir los modelos requeridos se usan varios metamodelos como UML y CWM. Además, se ha definido un conjunto de relaciones QVT tanto para marcar el PSM como para obtener el PIM de manera automática. La implementación de la propuesta se ha llevado a cabo en la herramienta Borland Together Architect. Por otro lado, se pretende ampliar la propuesta presentada en este artículo con el fin de tener en cuenta la integración de diversas fuentes operacionales, obteniendo un único PIM. Además, recientemente la OMG ha propuesto una nueva especificación llamada *Architecture-Driven Modernization* (ADM) [19] que aboga por la definición de un conjunto de metamodelos para facilitar las tareas de ingeniería inversa, como el análisis y la identificación de diferentes tipos de elementos en los sistemas de información existentes. Nuestra intención es alinear la propuesta presentada en este trabajo con ADM, considerando el desarrollo del AD como un auténtico escenario de modernización del software.

REFERENCIAS

[1] Alhajj, R.: Extracting the extended entityrelationship model from a legacy relational database. Inf. Syst. 28(6) (2003) 597–618

[2] Fahrner, C., Vossen, G.: A survey of database design transformations based on the entity-relationship model. Data Knowl. Eng. 15(3) (1995) 213–250

[3] Giorgini, P., Rizzi, S., Garzetti, M.: Goal-oriented requirement analysis for data warehouse design. DOLAP 2005, 47–56

[4] Golfarelli, M., Maio, D., Rizzi, S.: The Dimensional Fact Model: A conceptual model for data warehouses. Int. J. Cooperative Inf. Syst. 7(2-3) (1998) 215–247

[5] Hüsemann, B., Lechtenbörger, J., Vossen, G.: Conceptual data warehouse modeling. DMDW 2000

[6] Inmon, W.: Building the Data Warehouse. Wiley & Sons (2002)

[7] Jensen, M.R., Holmgren, T., Pedersen, T.B.: Discovering multidimensional structure in relational data. DaWaK 2004. Vol. 3181 of LNCS, 138–148

[8] Kimball, R., Ross, M.: The DataWarehouse Toolkit. Wiley & Sons (2002)

[9] Luján-Mora, S., Trujillo, J., Song, I.Y.: A UML profile for multidimensional modeling in data warehouses. Data Knowl. Eng. 59(3) (2006) 725–769

[10] Mazón, J.N., Trujillo, J., Lechtenbörger, J.: A set of QVT relations to assure the correctness of data warehouses by using multidimensional normal forms. ER 2006. Vol. 4215 of LNCS, 385–398

[11] Mazón, J.N., Trujillo, J., Serrano, M., Piattini, M.: Applying MDA to the development of data warehouses. DOLAP 2005, 57–66

[12] Mazón, J.N., Pardillo, J., Trujillo, J.: Applying transformations to model driven data warehouses. DaWaK 2006. Vol. 4081 of LNCS, 13-22

[13] Mazón, J.N., Trujillo, J.: An MDA approach for the development of data warehouses. Decision Support Systems 45(1) (2008) 45–58

[14] Mellor, S., Scott, K., Uhl, A., Weise, D.: MDA distilled: principles of Model-Driven Architecture. Addison Wesley (2004)

[15] OMG: MDA Guide 1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01

[16] OMG: MOF 2.0 Query/View/ Transformation http://www.omg.org/cgi-bin/doc?ptc/2005-11-01

[17] OMG: Common Warehouse Metamodel Specification 1.1. http://www.omg.org/cgi-bin/doc?formal/03-03-02

[18] OMG: Object Constraint Language (OCL) specification 2.0 http://www.omg.org/cgi-bin/doc?ptc/03-10-14

[19] OMG: Architecture Driven Modernization (ADM) http://adm.omg.org/

[20] OMG, Unified Modeling Language Specification 2.0, http://www.omg.org/cgi-bin/doc?formal/05-07-04.

[21] Phipps, C., Davis, K.C.: Automating data warehouse conceptual schema design and evaluation. DMDW 2002, 23–32

[22] Romero, O., Abelló, A.: Multidimensional design by examples. DaWaK 2006. Vol. 4081 of LNCS, 85–94

[23] Soutou, C.: Relational database reverse engineering: Algorithms to extract cardinality constraints. Data Knowl. Eng. 28(2) (1998) 161–207

Jose-Norberto Mazón es estudiante de doctorado en la Universidad de Alicante (España). Actualmente dispone de una beca FPU del Ministerio de Educación y Ciencia de España. Es ingeniero informático por la Universidad de Alicante. Ha publicado varios trabajos en congresos nacionales e internacionales como DAWAK, ER, DOLAP, BNCOD, JISBD, etc. Sus líneas de investigación son: modelado de bases de datos, diseño conceptual de almacenes de datos, modelado multidimensional y desarrollo dirigido por modelos. Su dirección de correo electrónico es jnmazon@dlsi.ua.es.

Juan Trujillo es profesor en la Escuela de Informática de la Universidad de Alicante, España. Trujillo obtuvo su Doctorado en Informática en la Universidad de Alicante (España) el año 2001. Sus intereses de investigación incluyen modelado de bases de datos, diseño conceptual de almacenes de datos, bases de datos multidimensionales, OLAP, y análisis y diseño orientado a objetos con UML. Ha publicado artículos en conferencias internacionales y revistas tales como ER, UML, ADBIS, CaiSE, WAIM, Journal de Gestión de Bases de Datos (JDM) e IEEE Computer. Participa como miembro de Comité de Programa de varios talleres y conferencias tales como ER, DOLAP, DSS, y SCI. También ha participado como revisor de varias revistas tales como JDM, KAIS, ISOFT y JODS. Su correo es jtrujillo@dlsi.ua.es.