Un Escenario para Diseño Lógico de Data Warehouses

Verónika Peralta
Instituto de Computación, Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
vperalta@fing.edu.uy

Resumen: Los Data Warehouses (DW) son bases de datos cuyo cometido es dar soporte a la toma de decisiones. Debido a sus diferencias con las bases de datos operacionales, las técnicas clásicas de diseño relacional no pueden ser aplicadas para el diseño de DWs relacionales. En este artículo se trata el problema de diseño lógico de DWs. Se presenta un escenario de diseño donde el diseñador especifica un esquema conceptual, correspondencias entre dicho esquema y la base de datos fuente y criterios de diseño que desea aplicar. Estos elementos son la entrada a un proceso de diseño que mediante transformaciones y reglas de alto nivel construye el esquema lógico del DW obteniendo (a) trazabilidad del diseño, (b) mapeo entre el esquema fuente y esquema del DW, y (c) facilidades para diseño de estructuras complejas de DW.

Palabras claves: Diseño de Data Warehouses.

1. Introducción

Los Data Warehouses (DW) son bases de datos cuyo cometido es dar soporte a la toma de decisiones. Usualmente la información de estas bases de datos es analizada por el usuario final con una perspectiva multidimensional [9], utilizando herramientas especializadas en facilitar este tipo de análisis (herramientas OLAP). Los DWs tienen características que los distinguen de las bases de datos operacionales, fundamentalmente con respecto a la información que almacenan y al tipo de operaciones que se realiza sobre ellos. La información contenida en un DW es resultado de transformaciones de datos provenientes de las bases operacionales. Las operaciones que debe soportar un DW son en general consultas complejas y no operaciones del tipo transaccional (OLTP), ya que el mantenimiento se hace normalmente en forma "batch" y periódicamente. Debido a las diferencias mencionadas anteriormente las técnicas clásicas de diseño de bases de datos relacionales no pueden ser aplicadas para el diseño de DWs relacionales. Para diseñar este tipo de bases de datos se suele buscar estructuras que optimicen las consultas, teniendo en cuenta el modelo dimensional [10].

En este artículo se abordan los problemas de diseño lógico de DWs, tomando como entrada un esquema conceptual multidimensional [9][3]. La realización previa de un esquema conceptual es importante porque permite la construcción de una descripción abstracta y completa del problema, cercana a la visión del usuario.

Para las bases de datos operacionales existen varias propuestas para construir un esquema relacional a partir de un esquema conceptual, particularmente un esquema Entidad Relación (E/R) [11][18][8], pero para DWs se carece de metodologías suficientemente generales. De los trabajos existentes en diseño de DWs algunos proponen construir el esquema lógico a partir de requerimientos sin realizar un diseño conceptual [14][10], mientras que otros proponen la realización de un esquema conceptual y a partir de éste generar un esquema lógico basado en un tipo particular de diseño (generalmente estrella) [2][6][1].

Nuestra propuesta consiste en un escenario de diseño para la construcción de un esquema lógico relacional para el DW tomando como entrada un esquema conceptual multidimensional y una base de datos fuente previamente integrada. En este escenario, el esquema lógico del DW se construye aplicando transformaciones al esquema lógico de la base de datos fuente [12][13]. Estas transformaciones se utilizan como bloques de diseño y embeben en su semántica técnicas usuales de diseño de DWs, permitiendo por ejemplo, agregar atributos calculados a una tabla, o realizar totalizaciones a un conjunto de atributos. Además de las transformaciones se provee un conjunto de *reglas* que embeben estrategias de diseño de más alto nivel, orientadas a encarar globalmente los distintos problemas de diseño. Estas reglas determinan qué transformaciones aplicar para resolver cada problema a partir de información provista por el diseñador, la cual consiste en los requerimientos del DW, la relación de éstos con las bases de datos fuentes, y los criterios de diseño que desea aplicar.

El ambiente permite al diseñador especificar la información de entrada de las reglas mediante tres construcciones: (1) un modelo conceptual multidimensional [3], (2) *mapeos*, los cuales especifican las correspondencias entre los elementos del esquema conceptual y los elementos del esquema fuente, y (3) *lineamientos*, que dan información sobre criterios de diseño relacional de DW.

Con este encare se cubren algunos aspectos que consideramos importantes en cualquier proyecto de construcción de un DW: (a) trazabilidad del diseño, (b) mapeo entre esquema lógico fuente y esquema lógico de DW, y (c) facilidades para diseño de estructuras complejas de DW (como estructuras para manejo de datos históricos, versionamiento de dimensiones, datos calculados, generalización de claves).

Resumiendo, la contribución principal de este artículo es la presentación de un escenario de diseño lógico de DW y la especificación de construcciones que permiten al diseñador: (a) especificar las correspondencias entre el esquema conceptual del DW y el esquema fuente, (b) especificar criterios de diseño lógico a aplicar, y (c) obtener un conjunto de transformaciones a aplicar para resolver los distintos problemas de diseño.

El resto del artículo se organiza de la siguiente manera: En la sección 2 describimos el escenario de diseño de DWs. En la sección 3 presentamos los lineamientos de diseño y los mapeos entre el esquema conceptual y el esquema fuente. En la sección 4 describimos el proceso de diseño usando transformaciones y reglas y en la sección 5 presentamos las conclusiones.

2. Un Escenario para Diseño Lógico de DW

La Figura 1 ilustra el escenario de diseño para construir un esquema lógico relacional de un DW tomando como entrada un esquema conceptual multidimensional y una base de datos fuente previamente integrada.

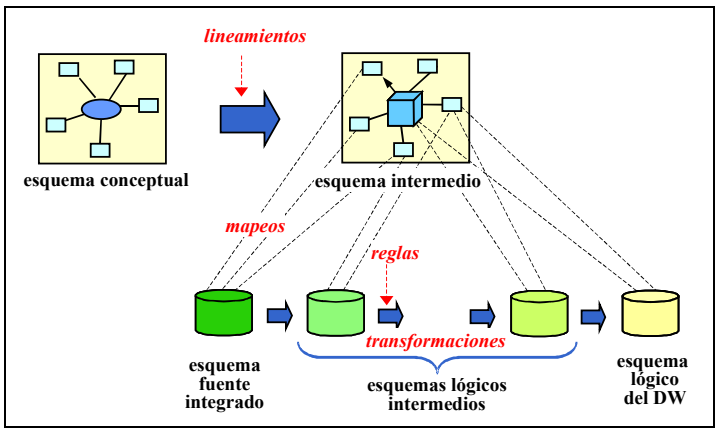

Figura 1 – Escenario de diseño lógico de DWs

Según nuestro enfoque, la tarea de diseñar un DW comienza con la construcción de un esquema conceptual multidimensional. Dicho esquema conceptual se complementa con lineamientos de diseño que abstraen estrategias de diseño lógico de DWs y restricciones de performance y almacenamiento para el mismo. Como ejemplo de lineamientos, el diseñador puede indicar cómo fragmentar datos históricos o cuándo normalizar o denormalizar dimensiones. Los lineamientos permiten al diseñador indicar las estrategias a aplicar según los requerimientos de su problema concreto, constituyendo un mecanismo sencillo y flexible para que el diseñador pueda expresar las propiedades que desea para el DW. El esquema conceptual más los lineamientos conforman el esquema intermedio.

Luego se especifican mapeos entre las estructuras del esquema intermedio y la base de datos fuente, indicando cómo obtener a partir de las fuentes los diferentes elementos conceptuales. El establecimiento de correspondencias o mapeos entre diferentes esquemas es ampliamente utilizado en el diseño de bases de datos [17][4][19]. En el contexto de DWs, además de correspondencias de equivalencia se necesita expresar cálculos y funciones sobre los elementos de la base de datos fuente.

La construcción del esquema lógico del DW se lleva a cabo mediante transformaciones aplicadas al esquema lógico de la base de datos fuente. El enfoque se basa en un conjunto de transformaciones de esquema predefinidas [12][13], que tienen embebida en su semántica las técnicas usuales de diseño de DW. Estas transformaciones se aplican a sub-esquemas relacionales, comenzando por el esquema fuente, generando como resultado el esquema

lógico del DW. Como forma de apoyar al diseñador en el proceso de diseño se presenta un conjunto de reglas de alto nivel que utilizan el esquema conceptual, los lineamientos y los mapeos como base para elegir qué transformaciones se deben aplicar y en qué orden.

3. Lineamientos y Mapeos

El diseño lógico del DW se realiza aplicando las transformaciones de esquemas, pero la decisión de qué transformaciones aplicar, y en qué orden hacerlo dependerá de los requerimientos, de la base de datos fuente y de los criterios de diseño del propio diseñador. Los requerimientos son modelados usando un modelo conceptual multidimensional [3], que representa los objetos del negocio en términos de dimensiones y cruzamientos entre ellas. Los diferentes criterios de diseño para el esquema lógico relacional pueden conducir a esquemas más o menos normalizados [14] y fragmentados [5]. Estos criterios de diseño tienen un papel fundamental en la elección de las transformaciones a aplicar. Por otro lado, dado que las transformaciones se aplican al esquema fuente, es necesario vincular a éste con el esquema conceptual, indicando donde se encuentran en la base de datos fuente los diferentes conceptos multidimensionales. En nuestra propuesta permitimos al diseñador especificar esta información mediante un *esquema conceptual*, *lineamientos, y mapeos*.

A continuación presentamos los lineamientos y los mapeos valiéndonos de un ejemplo para su ilustración.

3.1. Un ejemplo

El ejemplo trata de una empresa que brinda atención telefónica a sus clientes y quiere contabilizar el tiempo invertido en atención de llamadas por mes y por cliente, y totales anuales, tanto por ciudad como por departamento (división política del país) de residencia del cliente.

La Figura 2 muestra un esquema multidimensional para el problema, modelado con CMDM [3]. Se identifican tres dimensiones: *clientes, fechas* y *llamadas* (medida). La dimensión clientes tiene 3 niveles: *departamento, ciudad* y *cliente*; y la dimensión fechas tiene 2 niveles: *año* y *mes*. Para resolver los requerimientos se define la relación dimensional *atención*, que cruza *clientes y fechas*, tomando *llamadas* como medida.

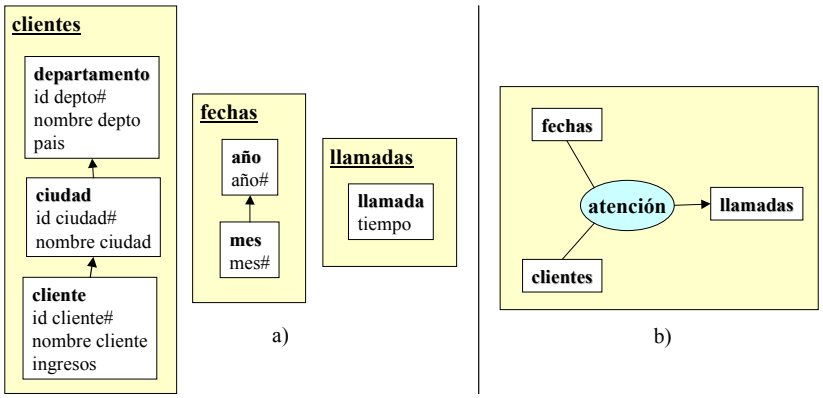

Figura 2 – Esquema conceptual: a) dimensiones, y b) relación dimensional. En a) los cuadros de texto son los niveles de la dimensión; sus nombres aparecen en negrita y debajo aparecen los atributos llamados *ítems*. Los ítems seguidos por un numeral (#) identifican al nivel. Las jerarquías de niveles se representan por flechas entre los niveles. En b) el óvalo central indica el nombre de la relación, las líneas indican las dimensiones que se cruzan y la flecha indica la medida.

Se quiere construir un esquema relacional a partir del esquema conceptual. Se tienen varias alternativas para definir tanto las tablas de dimensión como las tablas de hechos. Por ejemplo, para la dimensión *clientes*, se puede optar por almacenar cada nivel en una tabla, con los ítems del nivel y la clave del nivel superior en la jerarquía para poder realizar el join (Figura 3a). Otra opción sería mantener una única tabla para la dimensión que contenga todos los ítems (Figura 3b), o seguir estrategias intermedias. El diseñador debe decidir qué niveles almacenar juntos en una misma tabla, basándose en criterios de performance, redundancia y ocupación de disco.

De la misma forma se debe definir qué cruzamientos (llamados cubos [3]) se van a implementar, obteniendo tablas de hechos. El cubo con más detalle para la relación *atención* es por *mes* y por *cliente* (Figura 3c) pero también se puede querer almacenar totales, por ejemplo, por *ciudad* y *año* (Figura 3d). El diseñador debe decidir qué cubos materializar, tratando de lograr un equilibrio entre performance y espacio de almacenamiento.

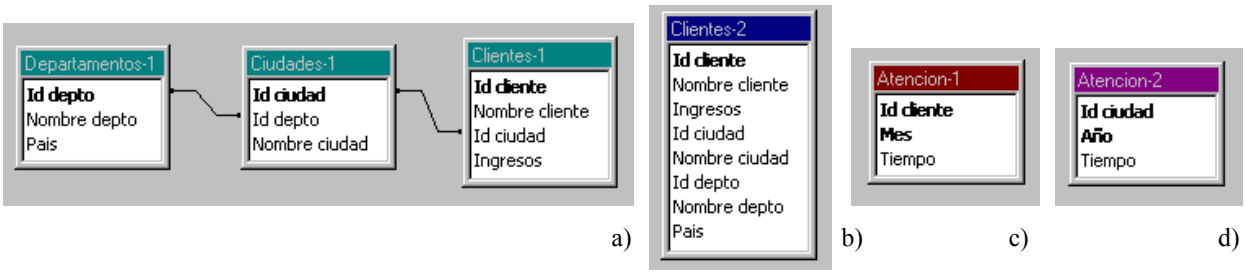

Figura 3 – Diferentes estrategias de generación de estructuras relacionales. Los cuadros de texto representan tablas en el modelo relacional. En la parte superior aparece el nombre de la tabla, y debajo aparecen los atributos. Los atributos en negrita conforman la clave primaria y las líneas entre tablas indican los atributos por los que debe realizarse el join (igualando atributos).

Las tablas de hechos se pueden fragmentar horizontalmente [15] de manera de mantener tablas con menos registros que mejoren la performance de las consultas de los usuarios. Se puede, por ejemplo, tener una tabla con los datos de los últimos dos años y tener los datos históricos en otra tabla. El diseñador debe decidir mediante qué criterios fragmentar horizontalmente las tablas de hechos.

Estos tipos de decisiones que debe tomar el diseñador se especifican mediante los *lineamientos*. Se trata de información que complementa al esquema conceptual con estrategias de diseño. Al definir los lineamientos el diseñador está dando pautas de alto nivel, de cómo debe ser el esquema lógico del DW.

Una vez definidos los lineamientos, se debe relacionar el esquema conceptual con la base de datos fuente. La Figura 4 muestra una posible base de datos de la empresa con tres maestros: de clientes, ciudades y departamentos, una tabla con los sueldos de los clientes y una tabla donde se registran sus llamados.

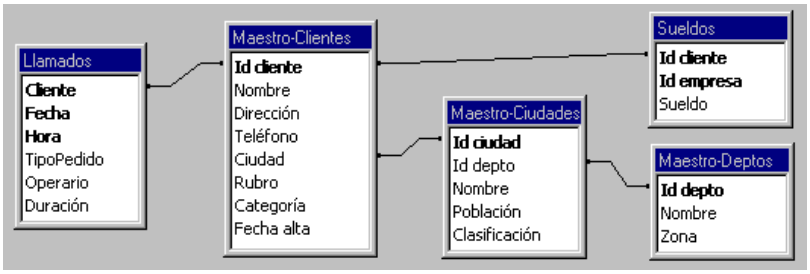

Figura 4 – Base de datos fuente

Dadas dos tablas interesa reflejar cómo se vinculan, es decir, cómo se debe realizar el join entre ellas. A un vínculo de este tipo se le llama *link*. La vinculación entre tablas puede expresarse mediante un grafo, donde los nodos son las tablas y las aristas son los links rotulados con predicados de join. Gráficamente se omiten los rótulos para los predicados de igualdad de atributos (mayoría de los casos) y se dibujan líneas que unen los atributos involucrados. En la Figura 4 pueden visualizarse los links entre las tablas.

El diseñador debe indicar de qué datos fuentes (tablas y atributos) se tomarán los datos para cargar el DW. Para ello indica como se corresponden los ítems del esquema conceptual con los atributos de la base de datos fuente. Esto se especifica mediante los *mapeos*. Por ejemplo, puede indicar que los ítems del nivel ciudad se mapean a los atributos *Id ciudad* y *Nombre* de la tabla *Maestro-Ciudades*, y los ítems *mes* y *año* se calculan a partir del atributo *Fecha* de la tabla *Llamados*.

En las siguientes secciones se formalizan estos conceptos.

3.2. Lineamientos de Diseño

A través de los lineamientos el diseñador define el estilo de diseño para el DW (snowflakes, estrella o alguna estrategia intermedia [14]) e indica requerimientos de performance y almacenamiento.

Existe un conjunto de lineamientos que el diseñador debe especificar. Este contiene lineamientos para *Materialización de Cubos*, *Fragmentación Vertical de Dimensiones* y *Fragmentación Horizontal de Cubos.* A continuación se presenta la especificación de un lineamiento; el conjunto completo puede consultarse en [16].

Fragmentación Vertical de Dimensiones

Mediante este lineamiento, el diseñador puede indicar el grado de normalización que quiere lograr al generar estructuras relacionales para cada dimensión. Por ejemplo, le puede interesar un esquema estrella, denormalizando todas las dimensiones. Por el contrario, le puede interesar un esquema de snowflakes, normalizando todas [14]. También le puede interesar tratar diferente cada dimensión, indicando para cada una si normaliza, denormaliza o efectúa una estrategia intermedia, indicando en este último caso, qué niveles quedan en una misma tabla.

Para especificar este lineamiento, el diseñador debe indicar para cada dimensión qué niveles desea almacenar juntos. Para ello define por extensión una función que a cada dimensión le hace corresponder una fragmentación de sus niveles. A esa función se le llama SchDFragmentation.

Para que una fragmentación tenga sentido los niveles de cada fragmento deben estar relacionados jerárquicamente. Es decir, si se considera el grafo que representa a la jerarquía de la dimensión, el sub-grafo inducido por los niveles del fragmento debe ser conexo. Si dos niveles de un mismo fragmento no están relacionados, ni directa ni transitivamente, entonces conforman un cruzamiento y se pierde la relación jerárquica de la dimensión. La fragmentación además debe ser completa, es decir que todos los niveles deben estar en al menos un fragmento para no perder información. Si no se quiere duplicar el almacenamiento de información de cada nivel, los fragmentos deben ser disjuntos, pero no es una exigencia. El diseñador decide cuando duplicar información de acuerdo a su estrategia de diseño.

En la Definición 1 se define un *fragmento* de una dimensión (Fragments(D)) como un sub-conjunto del conjunto de los niveles de la dimensión (D.L), los cuales conforman un sub-grafo conexo de su jerarquía (D.Po es el orden parcial de niveles de la dimensión que conforman sus jerarquías [3]). Luego se define un *conjunto completo de fragmentos* (CompleteFragmentSets (D)) como un conjunto de fragmentos en el que todo nivel de la dimensión (D.L) está en algún fragmento. Por último se define una *fragmentación de dimensiones* (SchDFragmentation) como una función que a cada dimensión le hace corresponder un conjunto completo de fragmentos (SchDimensions es el conjunto de dimensiones del esquema conceptual [16]).

Gráficamente, la fragmentación de una dimensión se representa recuadrando del mismo color o estilo los niveles de cada fragmento. La Figura 5 muestra la representación gráfica de dos fragmentos para la dimensión clientes, el primero con los niveles *departamento* y *ciudad* (línea corrida), y el segundo con el nivel *cliente* (línea punteada).

- $\text{FRAGMENTS}(D) \equiv \{F / F \subseteq D.L \wedge \forall A,B \in F . ($
  $<A,B> \in D.PO \vee$
  $<B,A> \in D.PO \vee$
  $\exists C \in F . (<A,C> \in D.PO \wedge <B,C> \in D.PO)) \}$
- $\text{COMPLETEFRAGMENTSETS}(D) \equiv \{Fs / Fs \subseteq \text{FRAGMENTS}(D) \wedge$
  $\forall L \in D.L . (\exists F \in Fs . (L \in F)) \}$
- $\text{SCHDFRAGMENTATION} \in \{D / D \in \text{SCHDIMENSIONS}\} \rightarrow$
  $\text{COMPLETEFRAGMENTSETS}(D) \}$

Definición 1 – SchDFragmentation

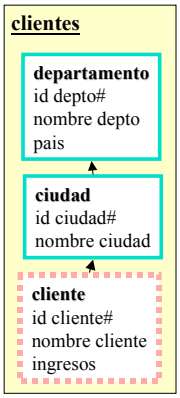

Figura 5 – Fragmentación

3.3. Mapeos entre el Esquema Conceptual y la Base de Datos Fuente

Los mapeos indican dónde se encuentran en el esquema lógico fuente los diferentes elementos del esquema conceptual y los lineamientos. Son funciones que asocian a cada elemento del esquema conceptual una expresión construida en base a las tablas y atributos del esquema fuente. Son definidas por el diseñador en forma explícita.

Antes de definir las funciones de mapeo se presenta una caracterización de las expresiones involucradas.

Expresiones de Mapeo

En una función de mapeo, a cada ítem del esquema conceptual le corresponde una expresión construida en base a atributos de las tablas fuentes, a la que se llama expresión de mapeo (*MapExpr*).

Una expresión de mapeo puede ser un atributo de una tabla fuente (DirectME), o un cálculo que involucra varios atributos de una tupla (1calcME), o un resumen o total que involucra varios atributos de varias tuplas (NcalcME) o un valor externo a la fuente como una constante, una marca de tiempo o dígitos de versión (ExternME).

A modo de ejemplo, la Definición 2 presenta la definición de las expresiones de resumen (NcalcME), las restantes definiciones pueden consultarse en [16]. Una expresión de resumen es una tripla que contiene: una tabla fuente: Tab (SchTables es el conjunto de tablas fuentes), un conjunto de atributos de la tabla que intervienen en el cálculo: Patts (Tab.As son los atributos de la tabla TAB) y una expresión de cálculo: Expr (RollUpExpressions(Patts) es el conjunto de expresiones tipadas de resumen que se pueden construir con los elementos del conjunto Patts [16])

- NCALCME ≡ { <EXPR, TAB, PATTS > / TAB ∈ SCHTABLES
  ∧ PATTS ⊆ TAB.AS ∧ EXPR ∈ ROLLUPEXPRESSIONS(PATTS) }

Definición 2 – NcalcME

El conjunto de las expresiones de mapeo (MAPEXPR) es la unión de las expresiones de los distintos tipos.

Funciones de Mapeo

Definido MapExpr, se pueden definir funciones de mapeo (Definición 3). Una función de mapeo, definida para un conjunto de ítems, le hace corresponder a cada ítem, una expresión de mapeo, controlando que coincidan los tipos de los ítems (I.Type) y de la expresión (F(I).Expr.Type).

- MAPPINGS(ITS) ≡ { F / F ∈ ITS → MAPEXPRS ∧ ∀ I ∈ ITS. (I.TYPE = F(I).EXPR.TYPE) }

Definición 3 – Mappings

Las funciones de mapeo se utilizan en 2 contextos: para vincular fragmentos de dimensiones a las tablas fuentes (mapeos de fragmentos), y vincular cubos a las tablas fuentes (mapeos de cubos). Esta vinculación se realiza definiendo una función de mapeo que haga corresponder los ítems del fragmento o cubo con las tablas fuentes.

Gráficamente una función de mapeo se representa por flechas de los ítems del esquema conceptual a los atributos de las tablas fuentes. Cuando el mapeo es directo se representa con una línea corrida, cuando es un cálculo se representa con una línea cortada a cada atributo que interviene en el cálculo y se adjunta la definición del cálculo, y cuando es externo no se utilizan líneas pero se adjunta la expresión de mapeo. En la Figura 6 se muestra una función de mapeo para la dimensión *clientes* (único fragmento). El ítem *país* tiene un mapeo externo (constante "Uruguay"), el ítem *ingresos* tiene un mapeo calculado en base al atributo *sueldo* de la tabla *sueldos*. Los demás ítems tienen mapeos directos.

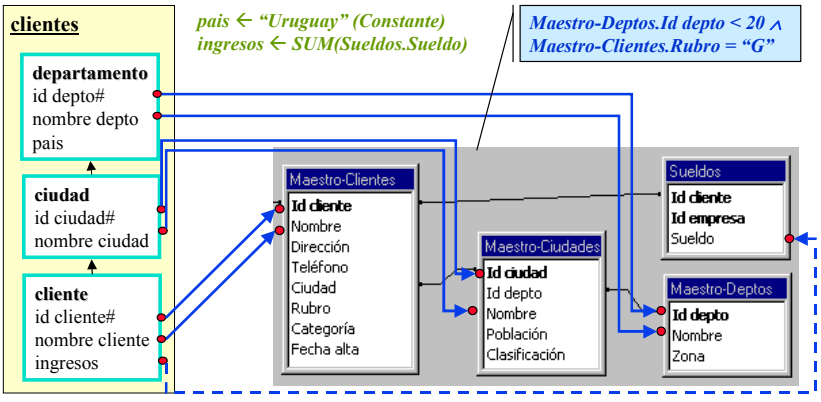

Figura 6 – Representación gráfica de una función de mapeo

Cada función de mapeo representa una vista o expresión SQL que tiene en el cabezal (SELECT) las expresiones de mapeo y obtiene los datos (FROM) de las tablas referenciadas en dichas expresiones, imponiendo los links entre las tablas como condición de join (WHERE) y agrupando por las expresiones que no son de agregación. Las tablas involucradas deben tener links definidos, no tienen sentido los productos cartesianos.

En el ejemplo de la Figura 6, se asocia al fragmento de la dimensión *clientes* la siguiente vista SQL:

```
SELECT  Maestro-Deptos.Id_depto as id_depto,  Maestro-Deptos.Nombre as nombre_depto,
        "Uruguay" as pais, Maestro-Ciudades.Id_ciudad as id_ciudad,
        Maestro-Ciudades.Nombre as nombre_ciudad,
        Maestro-Clientes.Id_cliente as id_cliente,
        Maestro-Clientes.Nombre as nombre_cliente, SUM(Sueldos.Sueldo) as ingresos,
FROM    Maestro-Deptos, Maestro-Ciudades, Maestro-Clientes, Sueldos
WHERE   Maestro-Clientes.Ciudad = Maestro-Ciudades.Id_ciudad
AND     Maestro-Ciudades.Id_depto = Maestro-Deptos.Id_depto
AND     Maestro-Clientes.Id_cliente = Sueldos.Id_cliente
GROUP BY Maestro-Deptos.Id_depto, Maestro-Deptos.Nombre, Maestro-Ciudades.Id_ciudad,
        Maestro-Ciudades.Nombre, Maestro-Clientes.Id_cliente, Maestro-Clientes.Nombre
```

En la vinculación de un fragmento o un cubo pueden existir condiciones, por ejemplo que los valores de un atributo se encuentren en determinado rango. Estas condiciones pueden deberse a restricciones en el esquema conceptual o a restricciones que deseen aplicarse a la base de datos fuente.

En el ejemplo de la Figura 6, el nivel *departamento* puede tener una restricción indicando que sólo interesan los departamentos con id menor a 20, que pueden ser por ejemplo, los departamentos uruguayos. Esta es una restricción del esquema conceptual. También puede ocurrir que en la tabla *Maestro-Clientes* se almacenen clientes de varios sistemas o secciones, y que para el DW sólo interesen los de un determinado rubro (por ejemplo: G - Gastronomía). Esta es una restricción respecto a la fuente.

Ambas restricciones deben tenerse en cuenta al establecer los mapeos, e incorporar una condición tipo:

$$Maestro\text{-}Deptos.Id\_depto < 20 \wedge Maestro\text{-}Clientes.Rubro = \text{“}G\text{”}$$

Estas restricciones pueden verse como condiciones adicionales de la vista SQL.

Gráficamente las condiciones se representan como llamadas (callouts). La Figura 6 muestra las condiciones anteriores. Formalmente, las condiciones son predicados sobre atributos de las tablas fuentes [16].

Para mapear un fragmento debe definirse una función de mapeo y opcionalmente una condición. Para mapear un cubo, además de una función de mapeo deben definirse otros elementos, como las operaciones de roll-up de las medidas. Omitimos estas definiciones por restricciones de espacio de este artículo; las mismas pueden ser consultadas en [16].

4. Diseño de DW Mediante Transformación de Esquemas

En esta sección comentamos brevemente la propuesta para diseño lógico de DW a través de transformación de esquemas y cómo esas transformaciones son utilizadas en las reglas de diseño.

4.1. Las Transformaciones

Se propone un mecanismo donde el esquema de DW es generado por aplicación sucesiva de transformaciones al esquema fuente. Durante el proceso las transformaciones se componen para obtener el esquema objetivo, dejando como resultado una traza de diseño. La Figura 7 muestra la arquitectura básica de la transformación.

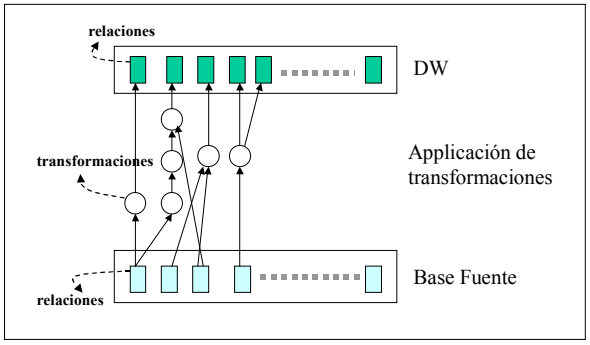

Figura 7 - Arquitectura de la transformación

Las transformaciones propuestas tienen como entrada un sub-esquema y su salida es otro sub-esquema y un esbozo de la transformación de la instancia correspondiente.

Las transformaciones pre-definidas realizan operaciones tales como: partición de una tabla, combinación de tablas, agregado de atributos calculados, y cambios en claves primarias y foráneas. Algunas las agrupamos en familias. Las transformaciones pertenecen a una misma familia cuando resuelven el mismo problema siguiendo distintas alternativas de diseño.

Los siguientes son algunos ejemplos que muestran su utilidad.

> *Muchas tablas en los sistemas operacionales no mantienen una noción temporal. Por ejemplo, las tablas de stock suelen tener los datos del stock actual, actualizándolo con cada movimiento de los productos. Sin embargo, en el DW la mayoría de las tablas necesitan incluir un elemento temporal, de manera que puedan mantener información histórica. Para este propósito existe una transformación llamada Temporalization que agrega un atributo de tiempo al conjunto de atributos de una tabla.*
>
> *En los sistemas de producción, usualmente los datos se calculan a partir de otros datos en el momento de las consultas, a pesar de la complejidad de algunas funciones de cálculo, para prevenir cualquier clase de redundancia. Por ejemplo, los precios de productos expresados en dólares son calculados a partir de los precios expresados en otra moneda y una tabla conteniendo el valor del dólar. En un sistema de DW, a veces es conveniente mantener esta clase de datos pre-calculados, por razones de performance. Definimos una familia de transformaciones, cuyo nombre es DD-Adding, que agrega a una tabla un atributo derivado a partir de otros.*

La Figura 8 muestra la lista de transformaciones predefinidas propuestas, la especificación de las mismas puede encontrarse en [12].

T1	Identity	T8	Hierarchy Roll Up
T2	Data Filter	T9	Aggregate Generation
T3	Temporalization	T10	Data Array Creation
T4	Key Generalization (family)	T11	Partition by Stability (family)
T5	Foreign Key Update	T12	Hierarchy Generation (family)
T6	DD-Adding (family)	T13	Minidimension Break off
T7	Attribute Adding	T14	New Dimension Crossing

Figura 8 – Las transformaciones

4.2. Reglas

Para construir manualmente y paso a paso un DW el diseñador toma una serie de decisiones, y en cada paso va transformando el esquema relacional del DW. Para ello se basa en características que observa en los lineamientos y mapeos; es decir, cuando se cumple determinada característica o condición, realiza cierta transformación en el DW. Proponemos un conjunto de reglas de diseño, cuyo objetivo es automatizar esa serie de decisiones, sustituyendo las decisiones del diseñador. Las reglas controlan que se verifiquen una serie de condiciones, y determinan qué transformaciones deben aplicarse al esquema.

La Figura 9 muestra una tabla con las reglas y un breve comentario de su utilidad (los * representan familias de reglas orientadas a resolver un mismo problema). La especificación puede encontrarse en [16].

	Regla	Descripción
R1	Join	Combina las dos tablas generando una nueva tabla. Se usa cuando un fragmento o cubo mapea a más de una tabla.
R2	Rename	Renombra los atributos mapeados en forma directa, utilizando los nombres de los ítems. Se usa cuando los nombres de los ítems no coinciden con los de los atributos a los que mapean.
R3	Calculate *	Agrega a una tabla un atributo que materializa un cálculo. Se usa cuando un ítem mapea a una expresión de cálculo.
R4	Extern *	Agrega a la tabla un atributo que materializa un valor externo. Se usa cuando un ítem mapea a una expresión externa.
R5	Group	Elimina de una tabla los atributos no mapeados por ningún ítem, agrupando por los demás y resumiendo (roll-up) las medidas. Se usa para eliminar atributos no mapeados.
R6	Drill Up *	Reduce el nivel de detalle de una tabla de hechos, realizando un drill-up respecto a una dimensión.
R7	Primary Key	Modifica la clave primaria de una tabla. Se usa para hacer coincidir los ítems clave de un fragmento o cubo, con los atributos que los mapean.
R8	Filter	Elimina de una tabla las tuplas que no cumplan una condición. Se usa para imponer condiciones en los mapeos, y para fragmentar horizontalmente un cubo.

Figura 9 – Reglas de diseño

A continuación ilustramos mediante un ejemplo la aplicación de la regla Aggregate Calculate de la familia Calculate (R3).

El objetivo de esta regla es materializar un cálculo. Se utiliza cuando se cumplen las siguientes condiciones: (i) un ítem de un fragmento o cubo se corresponde (por la función de mapeo) con una expresión de mapeo de tipo resumen (NcalcME), y (ii) las expresiones de mapeo de tipo directo o cálculo simple (directME y 1calcME) que se corresponden con los otros ítems, involucran una única tabla (tabla base).

Esta regla da como resultado la aplicación de la transformación DD-Adding N-N. La transformación genera una nueva tabla, con todos los atributos de la tabla base y un nuevo atributo que materializa el cálculo.

Ejemplo:

La Figura 10a muestra la dimensión *clientes* (con un único fragmento) y su función mapeo a las tablas *Clientes* y *Sueldos*. Al ítem *ingresos* le corresponde una expresión de mapeo de tipo NcalcME.

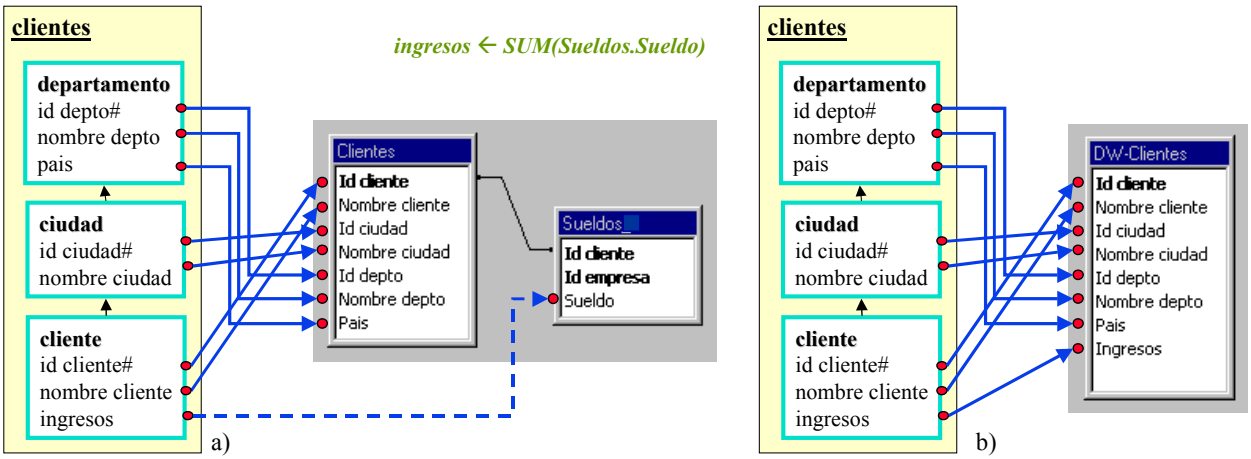

Figura 10 – Función de mapeo del fragmento de la dimensión clientes: a) antes y, b) después de aplicar la regla

Se calcula un nuevo atributo, de nombre *Ingresos,* como *SUM(Sueldos.Sueldo)*. Esto es, se construye una nueva tabla (*DW-Clientes*) agregando dicho atributo a los atributos de la tabla *Clientes*.

La aplicación de la regla actualiza la función de mapeo para que referencie a los atributos de la nueva tabla cuando corresponda. La Figura 10b muestra la función de mapeo actualizada.

5. Hacia la Automatización del Proceso de Diseño Lógico

Las diferentes metodologías de diseño de DWs [7][10][1][14][6] dividen el problema global de diseño en sub-problemas más pequeños y proponen algoritmos (o simplemente secuencias de pasos) para ordenar y atacar esos sub-problemas.

El escenario propuesto consiste en utilizar las reglas para la resolución de esos problemas concretos. En cada paso se aplicarían diferentes reglas, y podría intercalarse la aplicación de alguna transformación aislada. La incorporación de nuevos problemas a resolver puede demandar la definición de nuevas reglas, nuevos lineamientos e incluso nuevas transformaciones.

En este escenario el diseñador del DW trabaja de la siguiente manera: Comienza por definir un esquema conceptual multidimensional (en CMDM). Luego define los lineamientos y los mapeos. A continuación aplica las reglas que le dicen qué transformaciones debe aplicar al esquema fuente, y lo complementa si es necesario, con la aplicación de otras transformaciones.

Este escenario constituye un paso importante hacia la automatización del proceso de diseño. Actualmente se está trabajando en la definición de un algoritmo que engloba los pasos de metodologías existentes y genera automáticamente el esquema lógico del DW eligiendo el orden de aplicación de las reglas.

6. Conclusiones

Este artículo presenta un escenario para construir el esquema relacional de un DW tomando como entrada el esquema conceptual y la base de datos fuente. La información de entrada se complementa con un conjunto de lineamientos que permiten indicar qué propiedades debe cumplir el esquema lógico del DW, y un conjunto de mapeos que relacionan el esquema conceptual (enriquecido con los lineamientos) con la base de datos fuente.

En este escenario el esquema lógico del DW es construido mediante aplicación sucesiva de transformaciones al esquema fuente. Como ayuda en la automatización de las tareas del diseñador, se proveen reglas que dan como resultado la aplicación de las transformaciones adecuadas para los distintos problemas de diseño teniendo en cuenta el esquema conceptual, lineamientos y mapeos. Cada regla está orientada a resolver un problema particular, cubriendo los problemas más frecuentes en diseño de DWs relacionales.

Con este trabajo apuntamos a obtener un mecanismo semiautomático de diseño lógico de DWs, donde a partir de las especificaciones del diseñador mencionadas anteriormente, se genere el esquema del DW. Este mecanismo tendría la importante ventaja de permitir aplicar distintos criterios de diseño, por medio de la especificación de los lineamientos. Pero a la vez permite definir estrategias por defecto que automaticen la definición de los lineamientos (por ejemplo, una estrategia posible es construir siempre un esquema estrella denormalizando todas las dimensiones). Podrían estudiarse funciones de costos y heurísticas que automaticen la definición de lineamientos de acuerdo a distintos factores: limitaciones de espacio físico, performance en las consultas, afinidad, cantidad de datos.

Se ha desarrollado un prototipo de una herramienta CASE que soporta la generación del esquema relacional del DW a partir del esquema conceptual [16]. El prototipo incluye una interfaz para la definición de lineamientos, la definición de mapeos, y la aplicación automática de las reglas para la generación del esquema lógico.

Referencias

[1] Ballard, C.; Herreman, D.; Schau, D.; Bell, R.; Kim, E.; Valncic, A.: “Data Modeling Techniques for Data Warehousing”. SG24-2238-00. IBM Red Book. ISBN number 0738402451. 1998.

[2] Cabibbo, L. Torlone, R.:"A Logical Approach to Multidimensional Databases", EDBT, 1998.

[3] Carpani, F. Ruggia, R.: “An Integrity Constraints Language for a Conceptual Multidimensional Data Model”. SEKE’01, Argentina, 2001.

[4] Fankhauser, P.: “A Methodology for Knowledge-Based Schema Integration”. PhD-Thesis, Technical University of Vienna, 1997.

[5] Golfarelli, M. Maio, D. Rizzi, S.:”Applying Vertical Fragmentation Techniques in Logical Design of Multidimensional Databases”. DAWAK’00, UK, 2000.

[6] Golfarelli, M. Rizzi, S.:”Methodological Framework for Data Warehouse Design.", DOLAP’98, USA, 1998.

[7] Inmon, W.: “Building the Data Warehouse”. John Wiley & Sons, Inc. 1996.

[8] Jajodia, S. Ng, P. Springsteel, F.: “The problem of equivalence for entity-relationship diagrams”, IEEE Trans. on Software Engineering SEO,5. September 1983.

[9] Kenan Technologies:"An Introduction to Multidimensional Databases”. White Paper, Kenan Technologies, 1996.

[10] Kimball, R.:"The Datawarehouse Toolkit". John Wiley & Son, Inc., 1996.

[11] Markowitz, V. Shoshani, A.: “On the Correctness of Representing Extended Entity-Relationship Structures in the Relational Model”. SIGMOD’89, USA, 1989.

[12] Marotta, A.: “Data Warehouse Design and Maintenance through Schema Transformations”. Master Thesis. InCo - Pedeciba, Universidad de la República, Uruguay, 2000.

[13] Marotta, A. Ruggia, R.: “Data Warehouse Design: A schema-transformation approach”. Accepted paper in SCCC’2002. Chile. 2002.

[14] Moody, D.; Kortnik, M.: “From Enterprise Models to Dimensionals Models: A Methodology for Data Warehouse and Data Mart Design”. DMDW’00, Sweden, 2000.

[15] Ozsu, M.T. Valduriez, P.: “Principles of Distributed Database Systems”. Prentice-Hall Int. Editors. 1991.

[16] Peralta, V.: “Diseño lógico de Data Warehouses a partir de Esquemas Conceptuales Multidimensionales”. Master Thesis. InCo - Pedeciba, Universidad de la República, Uruguay. 2001.

[17] Spaccapietra, S. Parent, C.: “View integration: A step forward in solving structural conflicts”. TKDE’94,Vol 6,No.2, 1994.

[18] Teorey, T. Yang, D. Fry, J.: “A logical design methodology for relational databases using: the extended entity-relationship model”, Cornput&Surveys 18,2. June 1986.

[19] Yang, L. Miller, R. Haas, L. Fagin, R.: “Data-Driven Understanding and Refinement of Schema Mappings”. ACM SIGMOD’01, USA, 2001.