Resolución de la Integración en el Diseño del Data Warehouse ‡

Adriana Marotta
Instituto de Computación. Facultad de Ingeniería.
adriana@fing.edu.uy

Abstract

Un Data Warehouse (en adelante DW) es una base de datos orientada a la toma de decisiones cuya información proviene de múltiples fuentes. En un sistema de DW puede existir heterogeneidad en las fuentes de tipo semántica (diferencias en cuanto a qué objetos del mundo real se representan), sintáctica (diferencias en el esquema) y conflictos de datos (inconsistencia entre datos de las distintas bases, que se corresponden). Por lo tanto, en el proceso de diseño del DW y en el de carga de los datos al mismo, es necesario resolver problemas de integración. En este trabajo se agrega a la propuesta de [Mar00] para diseño de DW, la posibilidad de diseñar el DW a partir de múltiples fuentes, resolviéndose los problemas de integración.

1. Introducción

Un DW es una base de datos cuya información proviene de múltiples fuentes y es resultado de transformaciones que la hacen útil para el análisis orientado a la toma de decisiones. En un sistema de DW las fuentes de datos pueden ser heterogéneas y además contener semántica en común. Puede existir heterogeneidad semántica (diferencias en cuanto a qué objetos del mundo real se representan), sintáctica (diferencias en el esquema) y conflictos de datos (inconsistencia entre datos de las distintas bases, que se corresponden). Por lo tanto, en el proceso de diseño del DW y en el de carga de los datos al mismo, es necesario resolver problemas de integración.

Existen algunas propuestas para integración en un contexto de DW. En [Cal99] se trabaja en el problema de integración de datos, con un enfoque *Local As View (LAV)* y apoyándose sobre un modelo conceptual de datos corporativos. Cada tabla de las fuentes y del DW se define como una vista del modelo global de los datos corporativos. Se presenta una metodología que se apoya en una definición de correspondencias inter-esquema. Se puede decir que en esta propuesta se sigue un enfoque *eager*, ya que la integración se realiza a priori obteniendo una vista materializada. En [Gin00] se propone un lenguaje llamado *nd-SQL* que es capaz de expresar consultas sobre una federación de bases relacionales resolviendo conflictos entre los esquemas componentes, y expresar consultas OLAP involucrando agregaciones de granularidad múltiple. Esta propuesta se puede considerar en un enfoque *lazy*, ya que la integración se ejecuta en el momento en

‡ Este trabajo fue financiado por Comisión Sectorial de Investigación Científica de la Universidad de la República, Montevideo, Uruguay.

que se realiza la consulta. La metodología SIM [Fan97][Mot02] resuelve la integración en un contexto de bases de datos federadas basándose en un conjunto de correspondencias entre sub-esquemas de los esquemas a integrar. Es declarativa y semi-automatica. En el trabajo presentado en [DoC00] se adapta la metodología de integración SIM a un contexto de DW con información proveniente de la Web. Se incorporan nuevos tipos de correspondencias y criterios de integración de instancias.

En [Mar00] se propone una herramienta para diseño de DW a través de transformaciones de esquema. Con este mecanismo, el diseño del DW deja una traza de transformación que es útil principalmente para la construcción de los procesos de carga y para la evolución del DW a partir de la evolución del esquema fuente. En esta propuesta se diseña el DW a partir de una única base de datos fuente.

Nuestro objetivo en este trabajo es agregarle a la propuesta de [Mar00] la posibilidad de diseñar el DW a partir de múltiples fuentes. Este diseño dejaría como resultado el esquema del DW y ayudas para la especificación de los procesos de carga del mismo.

En la Sección 2 realizamos un análisis de los enfoques posibles para lograr nuestro objetivo, en la Sección 3 presentamos los problemas con que nos encontramos al aplicar el enfoque elegido, en la Sección 4 presentamos la propuesta concreta para generar el DW a partir de múltiples fuentes, en la Sección 5 comentamos los conceptos sobre los que nos basaríamos para la resolución de integración de instancias, y en la Sección 6 presentamos las conclusiones.

2. Análisis de enfoques posibles

En esta sección discutimos los posibles enfoques que podemos tomar para la generalización del mecanismo de diseño de [Mar00] a un contexto de múltiples fuentes, resolviendo la integración de éstas.

2.1. Enfoque LAV

En este enfoque se separa la transformación de las fuentes al DW en 2 etapas, una de integración y otra de diseño. En la **Figura 1** se muestra la arquitectura correspondiente a este enfoque.

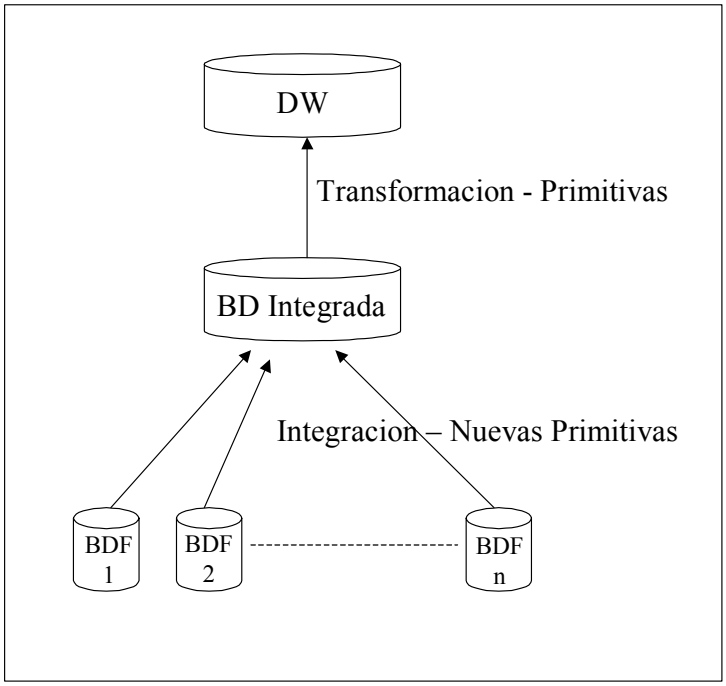

Figura 1: Arquitectura del proceso de diseño de DW. Enfoque LAV.

En la primera etapa se integran las fuentes utilizando técnicas de integración que no tienen en cuenta los requerimientos del DW ni un esquema deseado para el mismo. Se realiza integración de esquemas e integración de instancias. Se obtiene una base integrada que es resultado de la integración de las bases fuentes completas. Para esto seguiríamos la propuesta de [Mot02], donde por medio de operaciones llamadas *aumentaciones* se transforma cada esquema fuente en uno común y se realiza la integración de las instancias. Para utilizar esta propuesta sería necesario adaptarla al modelo relacional, y luego construir primitivas que implementen las aumentaciones y la resolución de conflictos de datos.

En la segunda etapa se transforma la base integrada en el DW. Al comienzo de este proceso ya se tiene el esquema deseado para el DW, o se cuenta con los requerimientos del DW o el modelo conceptual del mismo. Se utilizaría la propuesta de [Mar00] en donde se transforma el esquema de la base fuente (que es única) en el esquema del DW mediante la aplicación de *primitivas de transformación*. Esta aplicación dejaría una traza que es aprovechable para la carga de los datos desde la base integrada al DW.

Decimos que este enfoque es LAV (Local as View) porque cada una de las bases fuentes puede definirse como una vista de la base integrada. También se dice que es un enfoque bottom-up, ya que el proceso se realiza de abajo hacia arriba.

Las ventajas que presenta este enfoque son las siguientes: (1) se tratan por separado los problemas de integración y los problemas de diseño de DW. De esta manera en cada etapa hay un objetivo único, quedando claras las decisiones que se toman. (2) No es necesario contar a priori con el esquema lógico del DW.

Las desventajas que encontramos en este enfoque son: (1) se integran por completo las bases fuentes sin tener en cuenta si realmente toda esa información va a ser utilizada por el DW, y (2) es necesario definir un nuevo grupo de primitivas de transformación orientadas exclusivamente a la integración de bases de datos.

2.2. Enfoque GAV

En este enfoque se realiza la transformación de cada base fuente directamente hacia el DW. Estas transformaciones resuelven a la vez integración de las bases fuentes y diseño del DW. Es necesario contar con el esquema lógico del DW (esquema deseado) a priori, ya que todas las bases fuentes deben transformarse en sub-esquemas, no necesariamente disjuntos, del *mismo* esquema. En la **Figura 2** se muestra la arquitectura correspondiente a este enfoque.

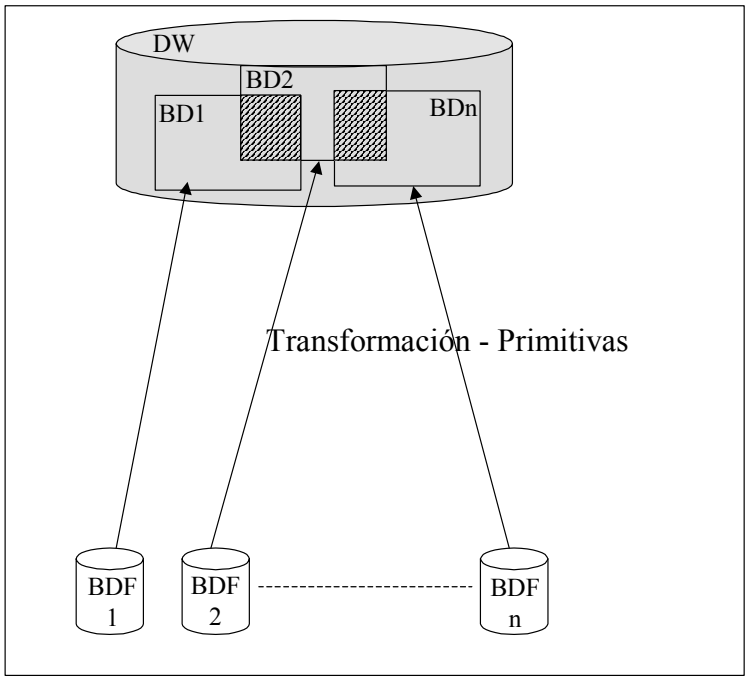

Figura 2: Arquitectura del proceso de diseño de DW. Enfoque LAV.

Decimos que este enfoque es GAV (Global as View) porque la base integrada (que coincide con el DW) es una vista de las bases fuentes. También se dice que es un enfoque top-down, ya que el proceso comienza "arriba" con la definición del esquema del DW.

Para realizar el proceso de transformación de las bases fuentes hacia el DW se siguen los siguientes pasos: (1) identificar los distintos sub-esquemas del esquema de DW, que serán resultado de la transformación de cada una de las bases fuentes, (2) obtener a partir de cada base fuente el sub-esquema deseado, mediante la aplicación de primitivas de transformación. La aplicación de las primitivas deja una traza que nos permite cargar las instancias en el DW resolviendo los problemas de integración de instancias.

Las ventajas que presenta este enfoque son las siguientes: (1) se integra de las bases fuentes solamente lo que se necesita para el DW, y (2) no es necesario definir un nuevo grupo de primitivas de transformación con un objetivo diferente al ya existente.

Algo que puede resultar una desventaja en este enfoque es que es necesario contar con el esquema del DW a priori.

Otra posibilidad es tener un enfoque GAV pero con la arquitectura de 2 etapas planteada para el enfoque LAV. Sería GAV porque a partir del esquema objetivo de DW o de los requerimientos se acotaría lo que se integra de las fuentes. El problema es que no habría un mecanismo bien determinado para identificar qué partes de las fuentes se deben integrar, ya que las correspondencias se establecerían entre el esquema de DW y el esquema integrado y no con las fuentes directamente. Por otro lado la propuesta de [Mot02] no trabaja a partir de correspondencias entre el esquema integrado y las fuentes.

3. Enfoque elegido y problemas en su aplicación

A partir del análisis de la sección anterior se decide aplicar el enfoque GAV realizándose la transformación de cada base fuente directamente hacia el DW como lo mostraba la **Figura 2**. En esta sección mostramos, mediante ejemplos, algunos problemas que traería su aplicación directa.

Cada proceso de transformación de una base fuente en uno de los sub-esquemas del DW se podría ver como un problema normal de diseño de DW partiendo de una única fuente. En este caso el paso de transformación de esquemas estaría resuelto. Sin embargo, esa visión es incorrecta. En nuestro contexto de múltiples fuentes que deben transformarse e integrarse a la vez en un DW, la transformación de las fuentes no se puede resolver como la suma de varias transformaciones independientes de una única fuente en un esquema objetivo. Existen casos en que la transformación de un dato de una fuente en uno del DW puede involucrar un cálculo en el cual intervenga un dato de otra fuente, no pudiendo posponerse este calculo para la etapa de integración de instancias.

Para clarificar el problema que se expone se muestran algunos ejemplos. En el **Ejemplo 1** se muestra un caso en el cual no habría problema en resolver las transformaciones de cada base fuente por separado y luego integrar.

Ejemplo 1.

Bases fuentes:

BF1:

Ventas-Montevideo

Vendedor	Articulo	Fecha	Cant
V1	A1	20/2/01	20
V2	A1	14/2/01	5
V3	A2	3/3/01	30

BF2:

Ventas-Interior

Vendedor	Articulo	Fecha	Cant
V4	A1	4/2/01	10
V5	A2	5/3/01	20
V6	A2	10/3/01	4

Se desea que el DW contenga la siguiente información:

DW:

Ventas

Articulo	Mes	Cantidad
A1	2/01	**35**
A2	3/01	**54**

Aplicando las transformaciones independientemente se obtendría:

BF1 → DW:

Articulo	Mes	Cantidad
A1	2/01	25
A2	3/01	30

BF2 → DW:

Articulo	Mes	Cantidad
A1	2/01	10
A2	3/01	24

Integrando,

DW:

Ventas

Articulo	Mes	Cantidad
A1	2/01	**35**
A2	3/01	**54**

♦

En el **Ejemplo 2** y el **Ejemplo 3** se muestran casos en los cuales no se puede realizar las transformaciones de cada base fuente en forma independiente y luego integrar.

Ejemplo 2.

Bases fuentes:

BF1:

Temperaturas1

Ciudad	Fecha	Temp
Montevideo	2/2/01	20
Montevideo	7/2/01	15
Durazno	3/3/01	30

BF2:

Temperaturas2

Ciudad	Fecha	Temp
Montevideo	20/2/01	25
Durazno	7/3/01	28
Durazno	20/3/01	30

Se desea que el DW contenga la siguiente información:

DW:

Temperaturas

Ciudad	Mes	Temperatura
Montevideo	2/01	**20**
Durazno	3/01	**29,3**

donde para cada ciudad y mes se tiene el promedio de temperaturas.

Aplicando las transformaciones independientemente se obtendría:

BF1 → DW:

Ciudad	Mes	Temperatura
Montevideo	2/01	17,5
Durazno	3/01	30

BF2 → DW:

Ciudad	Mes	Temperatura
Montevideo	2/01	25
Durazno	3/01	29

Integrando,

DW:

Temperaturas

Ciudad	Mes	Temperatura
Montevideo	2/01	**21,25**
Durazno	3/01	**29,5**

♦

Como se puede ver en el Ejemplo 2, las temperaturas promedio que se obtienen transformando independientemente las bases fuentes y luego integrando, no son las esperadas.

Ejemplo 3.

Bases fuentes:

BF1:

Ventas

Articulo	Fecha	Cant
A1	5/2/02	100
A1	7/2/02	50
A2	1/3/02	130
A3	2/3/02	80
A4	3/3/02	40

Articulos

Articulo	Tipo-art
A1	T1
A2	T1
A3	T2
A4	T2

BF2:

Articulos

Articulo	Nac-o-Imp
A1	N
A2	I
A3	N
A4	N

Se desea que el DW contenga la siguiente información:

DW:

Vtas-art-nacionales

Tipo-art	Mes	Cantidad
T1	2/02	**150**
T2	3/02	**120**

donde para cada tipo de articulo y mes se tiene la cantidad vendida, pero solo para artículos nacionales.

Intentamos aplicar las transformaciones independientemente:

BF1 → DW:	**BF2 → DW:**
Se necesitan datos de BF2 para saber si el articulo es nacional, antes de agrupar.	No tendría sentido llevar datos de BF2 al DW.

♦

Como se ve en los ejemplos 2 y 3, existen casos en que es necesario integrar los esquemas y/o datos provenientes de distintas fuentes dentro del proceso de transformación de las fuentes al DW. En otras palabras, no siempre se puede dejar el problema de integración para después de terminadas las transformaciones de las fuentes al DW.

Los casos en que es necesario resolver la integración durante el proceso de transformación se pueden agrupar de la siguiente forma: (a) los casos en que para llegar a una relación del DW se transforma un sub-esquema de una única fuente, pero en cierto paso de la transformación necesita la "contribución" de otra fuente (caso del ejemplo 3), y (b) los casos en que para llegar a una relación del DW se transforman sub-esquemas de varias fuentes, hay una operación de agregación de datos y existen medidas para las cuales no es lo mismo integrarlas y luego agregarlas que agregarlas y luego integrarlas (caso del ejemplo 2).

4. Generación del DW a partir de múltiples fuentes de datos

Nuestro objetivo es dar una propuesta para que se pueda generar el DW a partir de múltiples bases fuentes, con el enfoque GAV explicado en la Sección 2.2, solucionando los problemas mostrados en la sección anterior.

El proceso de generación del DW que proponemos parte de un esquema lógico objetivo de DW y consta de las siguientes 2 etapas: (1) establecimiento de correspondencias semánticas entre el esquema de DW y los esquemas de las bases de datos fuentes (esto es necesario para la determinación de los sub-esquemas de DW que serán resultado de la transformación de cada una de las bases fuentes) y (2) aplicación de primitivas de transformación a las bases fuentes. La aplicación de las primitivas debería dejar una traza que nos permita cargar las instancias en el DW resolviendo los problemas de integración de instancias.

Para hacer posible el proceso planteado es necesario: definir qué tipos de correspondencias semánticas se pueden establecer entre el DW y las fuentes, y adaptar la propuesta de [Mar00] de forma que contemple la existencia de varias fuentes y los distintos casos en que hay que resolver integración de varias fuentes durante el proceso de transformación.

4.1. Correspondencias semánticas

Es necesario establecer correspondencias semánticas entre sub-esquemas del DW y sub-esquemas de las bases fuente.

Se utilizan los siguientes tipos de correspondencias:

- de **equivalencia (≡)**
  Dos sub-esquemas son equivalentes cuando se puede definir una función biyectiva entre los respectivos conjuntos de instancias posibles.

- de **inclusión (⊂)**
  Un sub-esquema S_1 tiene una correspondencia de inclusión hacia otro sub-esquema S_2 cuando se puede definir: (a) una función inyectiva pero no subyectiva desde $I(S_1)$ a $I(S_2)$, o (b) una función biyectiva desde un subconjunto de $I(S_2)$ a $I(S_1)$ y una condición que indique qué elementos de $I(S_2)$ pertenecen a ese subconjunto. I(S) denota el conjunto de las instancias posibles del sub-esquema S.

- de **solapamiento (Θ)**
  Dos sub-esquemas S_1 y S_2 están en correspondencia de solapamiento cuando se puede definir una función biyectiva entre un subconjunto de $I(S_1)$ y un subconjunto de $I(S_2)$.

- de **partición vertical / horizontal (PV / PH)**
  Un sub-esquema S_1 tiene una correspondencia de partición vertical hacia otro sub-esquema S_2 cuando existe una partición de S_2 en 2 sub-esquemas S_{21} y S_{22}, tal que se pueden definir dos funciones f_1: $I(S_1) \rightarrow I(S_{21})$ y f_2: $I(S_1) \rightarrow I(S_{22})$, de manera que su "pairing" $<f_1,f_2>$: $I(S_1) \rightarrow I(S_{21}) \times I(S_{22})$ es inyectiva.

  Un sub-esquema S_1 tiene una correspondencia de partición horizontal hacia otro sub-esquema S_2 cuando existe una partición de $I(S_2)$ en n subconjuntos $I_1(S_2)$ ... $I_n(S_2)$ tal que se puede definir una función biyectiva f: $I(S_1) \rightarrow \cup_{i:1..n} I_i(S_2)$, siendo $\cup$ una unión disjunta.

- de **agrupamiento (>>)**
  Un sub-esquema S_1 tiene una correspondencia de agrupamiento hacia otro sub-esquema S_2 cuando se puede definir una función inyectiva f: $I(S_1) \rightarrow \mathcal{P}_{fin}(I(S_2))$.

Estos tipos de correspondencias se encuentran definidos y explicados con más detalle en [Mot02]. Extendemos la definición de la correspondencia de inclusión, ya que en el contexto de este trabajo es necesario que la función definida pueda mapear elementos del conjunto más grande al más pequeño.

Estas correspondencias ayudan a elegir qué primitivas de transformación aplicar a las fuentes y en particular permiten deducir si en un sub-esquema del DW participan varias fuentes y como es su participación.

En los ejemplos de la Sección 3 se tendrían las siguientes correspondencias:

Notación: {R1, ..., Rn} denota al sub-esquema formado por las relaciones R1, ..., Rn.

Ejemplo 1:

DW.Ventas >> BF1.Ventas-Montevideo
DW.Ventas $\supset$ BF1.Ventas-Montevideo

DW.Ventas >> BF2.Ventas-Interior
DW.Ventas $\supset$ BF2.Ventas-Interior

DW.Ventas PH {BF1.Ventas-Montevideo, BF2.Ventas-Interior}

Ejemplo 2:

Las correspondencias serían análogas a las del ejemplo anterior.

Ejemplo 3:

DW.Vtas-art-nacionales >> BF1.{Ventas, Articulos}

DW.Vtas-art-nacionales $\subset$ BF1.{Ventas, Articulos}

siendo: f: A $\rightarrow$ I(DW.Vtas-art-nacionales)
A = { e $\in$ I(BF1.{Ventas, Articulos}) / g(e).Nac-o-imp = "N" }
g: I(BF1.{Ventas, Articulos}) $\rightarrow$ I(BF2.Articulos)

♦

4.2. Casos

Nos interesa diferenciar los sub-esquemas del DW según la intervención de las fuentes para la generación de sus relaciones. Considerando las correspondencias establecidas, distinguimos los siguientes casos:

Caso 1 - Sus relaciones son generadas a partir de una única fuente, es decir todas las correspondencias son con relaciones de la misma fuente.

Caso 2 - Sus relaciones son generadas a partir de una fuente con contribución de otra/s. Todas las correspondencias son con relaciones de una misma fuente, pero para algunos cálculos se necesita la intervención de otra/s fuente/s.

Caso 3 - Sus relaciones son generadas a partir de varias fuentes. Esto significa que para una misma relación existen correspondencias con relaciones de distintas fuentes.

Continuando con los ejemplos de la Sección 3, vemos en qué casos estaría cada uno:

Ejemplo 1 y Ejemplo 2:

DW.Ventas tiene correspondencias con BF1 y con BF2.

Clasificamos a {DW.Ventas} como *Caso 3*.

Ejemplo 3:

DW.Vtas-art-nacionales tiene correspondencias solamente con BF1, pero en la función f que mapea las instancias posibles interviene la fuente BF2.

Clasificamos a {DW.Vtas-art-nacionales} como *Caso 2*.

♦

A la vez en el *Caso 3* nos interesa distinguir los casos en los cuales alguna de las transformaciones involucra una operación de agregación y la medida que se agrega debe ser integrada antes de ser agregada. Esto significa que las instancias provenientes de las distintas fuentes deben ser integradas antes de realizarse la transformación de agregación. El **Ejemplo 2** estaría en este caso.

4.3. Modificación de primitivas

En esta sección presentamos nuestra propuesta para darle a las primitivas de transformación de esquemas para diseño de DW de [Mar00], la potencialidad necesaria para trabajar a partir de múltiples bases fuentes. Para esto realizamos dos modificaciones sobre el conjunto de las primitivas: (1) modificamos la especificación de algunas primitivas, y (2) creamos una nueva primitiva para integración de dos relaciones semánticamente correspondientes.

En ambos casos se utilizan funciones de integración de instancias (*II-Match, II-Reconcile*) cuyo objetivo es resolver conflictos de datos. *II-Match* indica si 2 instancias de 2 conjuntos de atributos equivalentes[1] corresponden al mismo objeto del mundo real. *II_Reconcile* devuelve una instancia que es el resultado de reconciliar 2 instancias diferentes. Las funciones de integración de instancias se comentan con más profundidad en la Sección 5.

Para la modificación (1) identificamos dentro del conjunto de primitivas todas aquellas que realizan transformaciones que, con mínimas modificaciones en su especificación, podrían involucrar más de una fuente. Entre estas están, por ejemplo, las primitivas que agregan atributos calculados a una relación y dicho cálculo puede involucrar datos de otra

[1] El concepto de equivalencia de atributos es el mismo que el de equivalencia de sub-esquemas presentado en la Sección 4.1.

relación. Mientras que, por ejemplo, una primitiva que filtra atributos en una relación no estaría en este grupo.

Las primitivas de transformación que modificamos, permitiendo que tomen como argumentos elementos de distintas fuentes, son las siguientes:

P6 - DD-Adding
P8 - Hierarchy Roll-Up
P12 - Hierarchy Generation
P14 - New Dimension Crossing

A estas primitivas modificadas les llamamos *Primitivas Multifuente*. En las Figuras 3 y 4 se muestra como queda la especificación de algunas de ellas. Las modificaciones realizadas se encuentran resaltadas en negrita.

Primitive 6.2 **DD-ADDING N-1**

Description:

This primitive adds to a relation an attribute that is derived from some attributes from the same relation and others from other relation. **The relations may belong to different source databases.** The calculation function works over only one tuple of the relations. This tuple must be uniquely obtained through a join operation. Besides, the derived attribute can be defined as a foreign key to another relation.

Note: This primitive works with only two relations. If participation of more than two relations is required, additional steps must be applied.

Input:

- source schema : $\mathbf{SDB_1.R_1 (A_1, ..., A_n), SDB_2.R_2 (B_1, ..., B_m) \in \mathit{Rel}}$
- f (C_1, ..., C_k) / { C_1, ..., C_k } $\subseteq$ { A_1, ..., A_n }$\cup${ B_1, ..., B_m }, where f is a user-defined function
- A / A $\in$ { A_1, ..., A_n } $\wedge$ B / B $\in$ { B_1, ..., B_m } , join attributes
- is_fk , Boolean argument (declare A_{n+1} as a foreign key or not)
- $R_3 \in \mathit{Rel}$, relation to which A_{n+1} is a foreign key (optional)
- source instance : r_1, r_2

Resulting schema:

- R'_1 (A_1, ..., A_n, A_{n+1}) $\in \mathit{Rel}$ / A_{n+1} represents f (C_1, ..., C_k) $\wedge$
  if is_fk then $A_{n+1} = Att_{FK}(R'_1, R_3)$

Generated instance:

- **r'_1 = select A_1, ..., A_n, f (C_1, ..., C_k)**
  from $SDB_1.R_1$ $SDB_2.R_2$
  where II-Match ($SDB_1.R_1.A$, $SDB_2.R_2.B$)

Figura 3: Especificación de la Primitiva DD-Adding N-1

Primitive 8 **HIERARCHY ROLL UP**

Description:

Given a measure relation R_1 and a hierarchy relation R_2, this primitive does a roll up to R_1 by one of its attributes following the hierarchy in R_2 (by a foreign key that must exist from R_1 to R_2). Besides, it can generate another hierarchy relation with the corresponding grain. **The relations R_1 and R_2 may belong to different source databases.**

Input:

- source schema :
  - **$SDB_1.R_1$ (A_1, ..., A_n) $\in$ Rel_M**
  - **$SDB_2.R_2$ (B_1, ..., B_n) $\in$ Rel_J**
  - **$\exists$ A $\in$ {A_1, ..., A_n} $\wedge$ $\exists$ B $\in$ {B_1, ..., B_n} $\wedge$ {B} $\in$ $Att_K(R_2)$ $\wedge$ II-Match ({A}, {B})**
- Z set of attributes / card(Z) = k (measures)
- B / B $\in$ { B_1, ..., B_n } $\wedge$ B $\in$ $Att_D(R_2)$ (chosen hierarchy level)
- { e_1, ..., e_k } , aggregate expressions
- X / X $\subset$ { A_1, ..., A_n } $\wedge$ X $\subset$ ($Att_D(R_1) \cup Att_M(R_1)$) (they have a lower grain)
- Y / Y $\subset$ { B_1, ..., B_n } $\wedge$ Y $\subset$ $Att_D(R_2)$ (they have a lower grain)
- agg_h , Boolean argument (generate a new hierarchy or not)
- source instance : r_1, r_2

Resulting schema:

- R'_1 (A'_1, ..., A'_m) $\in$ Rel_M / { A'_1, ..., A'_m } = sust [A, B, { A_1, ..., A_n } – X]
- If agg_h then

$$R'_2 (B'_1, ..., B'_m) \in Rel_J / \{ B'_1, ..., B'_m \} = \{ B_1, ..., B_n \} - Y \wedge \{B\} \in Att_K(R'_2) \wedge Att_{FK}(R'_1, R'_2) = \{B\}$$

Note: Note that the original hierarchy relation is not part of the resulting schema in any case of application of this primitive.

Generated instance:

- **r'_1 = select $SDB_1.R_1.(\{A_1, ..., A_n\}-\{A\}-X-Z \cup \{ e_1, ..., e_k \}) \cup \{SDB_2.R_2.B\}$**
  from $SDB_1.R_1$ $SDB_2.R_2$
  where II-Match ($SDB_1.R_1.A$, $SDB_2.R_2.B$)
  group by $SDB_1.R_1.(\{A_1, ..., A_n\}-\{A\}-X-Z) \cup \{SDB_2.R_2.B\}$

Figura 4: Especificación de la Primitiva Hierarchy Roll Up

Para la modificación (2) creamos una primitiva de transformación **P15 – Relation Integration**.

Esta primitiva recibe como argumentos: (a) 2 relaciones R1 y R2 provenientes de distintas fuentes, (b) un conjunto de triplas de conjuntos de atributos <X,Y,Z>, donde para cada tripla se cumple: (i) los conjuntos X e Y son semánticamente equivalentes, (ii) $X \in R1$, $Y \in R2$, (iii) Z es el resultado de la integración de X e Y, y (c) 2 parámetros que indican si la integración debe ser por unión o por intersección de atributos y si el resultado contendrá la unión o la intersección de las instancias.

El esquema resultado de la aplicación de la primitiva es una relación cuyos atributos son la unión o intersección (según se haya pedido) de los atributos de las relaciones de entrada. La primitiva además provee el pseudo-código de la transformación e integración de las instancias fuentes para la carga de la relación resultado.

En la Figura 5 se presenta la nueva primitiva.

Primitive 15 **RELATION INTEGRATION**

Description:

This primitive generates a relation that is the integration of two relations that come from different source databases and are semantically correspondent. It can integrate making the union or the intersection of the relations' attributes. It can integrate making the union or the intersection of the relations' instances.

Input:

- source schema : $SDB_1.R_1 (A_1, ..., A_n), SDB_2.R_2 (B_1, ..., B_m) \in Rel$
- $\{<X_1,Y_1,Z_1>, ..., <X_p,Y_p,Z_p>\} / X_i \subseteq \{ A_1, ..., A_n \} \wedge Y_i \subseteq \{ B_1, ..., B_m \} \wedge X_i \equiv Y_i$
- ii-type , Argument that indicates instance integration type ($\cup$ or $\cap$)
- ai-type, Argument that indicates attribute integration type ($\cup$ or $\cap$)
- source instance : r_1, r_2

Resulting schema:

- R' / if ai-type = "$\cup$" then

  $Att(R') = (Att(SDB_1.R_1) - \cup_{I=1..p}X_i) \cup (Att(SDB_2.R_2) - \cup_{I=1..p}Y_i) \cup (\cup_{I=1..p}Z_i)$

  else if ai-type = "$\cap$" then

  $Att(R') = \cup_{I=1..p}Z_i$

Generated instance:

- r'_1 =

if ai-type = "$\cup$" then

if ii-type = "$\cup$" then

select (Att($SDB_1.R_1$) - $\cup_{l=1..p}X_i$) $\cup$ (Att($SDB_2.R_2$) - $\cup_{l=1..p}Y_i$) $\cup$
{ II-Reconcile(X_1,Y_1), …., II-Reconcile(X_p,Y_p)}
from $SDB_1.R_1$ $SDB_2.R_2$
where outer-join (II-Match ($SDB_1.R_1.X_1$, $SDB_2.R_2.Y_1$) and
…… and II-Match ($SDB_1.R_1.X_p$, $SDB_2.R_2.Y_p$))

else if ii-type = "$\cap$" then

select (Att($SDB_1.R_1$) - $\cup_{l=1..p}X_i$) $\cup$ (Att($SDB_2.R_2$) - $\cup_{l=1..p}Y_i$) $\cup$
{ II-Reconcile(X_1,Y_1), …., II-Reconcile(X_p,Y_p)}
from $SDB_1.R_1$ $SDB_2.R_2$
where II-Match ($SDB_1.R_1.X_1$, $SDB_2.R_2.Y_1$) and
…… and II-Match ($SDB_1.R_1.X_p$, $SDB_2.R_2.Y_p$)

else if ai-type = "$\cap$" then

if ii-type = "$\cup$" then

select { II-Reconcile(X_1,Y_1), …., II-Reconcile(X_p,Y_p)}
from $SDB_1.R_1$ $SDB_2.R_2$
where outer-join (II-Match ($SDB_1.R_1.X_1$, $SDB_2.R_2.Y_1$) and
…… and II-Match ($SDB_1.R_1.X_p$, $SDB_2.R_2.Y_p$))

else if ii-type = "$\cap$" then

select { II-Reconcile(X_1,Y_1), …., II-Reconcile(X_p,Y_p)}
from $SDB_1.R_1$ $SDB_2.R_2$
where II-Match ($SDB_1.R_1.X_1$, $SDB_2.R_2.Y_1$) and
…… and II-Match ($SDB_1.R_1.X_p$, $SDB_2.R_2.Y_p$)

Figura 5: Especificación de la Primitiva Relation Integration

Ejemplo de una aplicación de la primitiva Relation Integration:

Bases Fuentes: BDPersonal y BDContaduria

BDPersonal.Empleados (Nombre, Apellido, Cargo, Telefono)

BDContaduria.Empleados (Nombre, Posición, Sueldo)

Aplicamos P15 con los siguientes parámetros:

X_1 = {Nombre, Apellido} Y_1 = {Nombre} Z_1 = (Nombre}
X_2 = {Cargo} Y_2 = {Posición} Z_2 = {Cargo}
ii-type = $\cap$
ai-type = $\cup$

Resultado:

DW-Empleados (Telefono, Sueldo, Nombre, Cargo)

Carga de datos:

SELECT A.telefono, B.sueldo, II-Reconcile(A.{Nombre, Apellido}, B.{Nombre}),
II-Reconcile(A.{Cargo}, B.{Posición})

FROM BDPersonal.Empleados A, BDContaduria.Empleados B

WHERE II-Match(A.{Nombre, Apellido}, B.{Nombre}) AND
II-Match(A.{Cargo}, B.{Posición})

Si las instancias fueran las siguientes:

BDPersonal.Empleados

Nombre	Apellido	Cargo	Telefono
Juan	Perez	C1	5050402
Silvia	Gonzalez	C4	2002120
Maria	Pena	C2	3210406

BDContaduria.Empleados

Nombre	Posición	Sueldo
Juan Perez	CC1	20000
S. Gonzalez	CC4	5000

Quedaría:

DW.Empleados

Nombre	Cargo	Telefono	Sueldo
Juan Perez	C1	5050402	20000
Silvia Gonzalez	C4	2002120	5000

En este ejemplo tomamos:

- II-Reconcile devuelve siempre los valores de la fuente BDPersonal (selecciona por preferencia)
- II-Match consulta en una tabla que contiene los valores que se corresponden.

♦

4.4. Re-definición de la traza de transformación

En [Mar00] se define una traza de transformación, la cual provee la información completa sobre las secuencias de primitivas que fueron aplicadas (componiéndose) a cada elemento del esquema fuente. Esta traza puede verse como un mapeo entre los elementos del esquema del DW y los elementos del esquema fuente.

En el contexto de este trabajo es necesario generalizar la definición de la traza de forma que mapee los elementos del esquema del DW con elementos de múltiples esquemas fuente. A la nueva traza le llamamos *traza multifuente* ($\mathcal{TT}$) y su definición es la misma que la de la traza $\mathcal{T}$, presentada en [Mar00], excepto que en la gramática los términos *Rel_Name* y *Att_Name* ahora representan relación y atributo perteneciente a determinada fuente.

En algunos casos nos interesa ver a la traza multifuente como una familia de trazas. Para esto la particionamos de la siguiente manera:

$\mathcal{TT} = \{ \mathcal{T}_{BF1}, ..., \mathcal{T}_{BFn}, \mathcal{T}_{e1}, ..., \mathcal{T}_{em} \}$,

siendo: {BF1, ..., BFn} $\subseteq$ el conjunto de las bases fuentes
{e1, ..., em} $\subseteq \mathcal{P}_{\{BF1, ..., BFn\}}$ (conjunto potencia)

Las trazas tipo $\mathcal{T}_{BFi}$, son trazas que mapean un sub-esquema del DW con una única fuente. Las trazas tipo $\mathcal{T}_{ei}$, son trazas que mapean un sub-esquema del DW con varias fuentes. La Figura 6 ilustra esta idea.

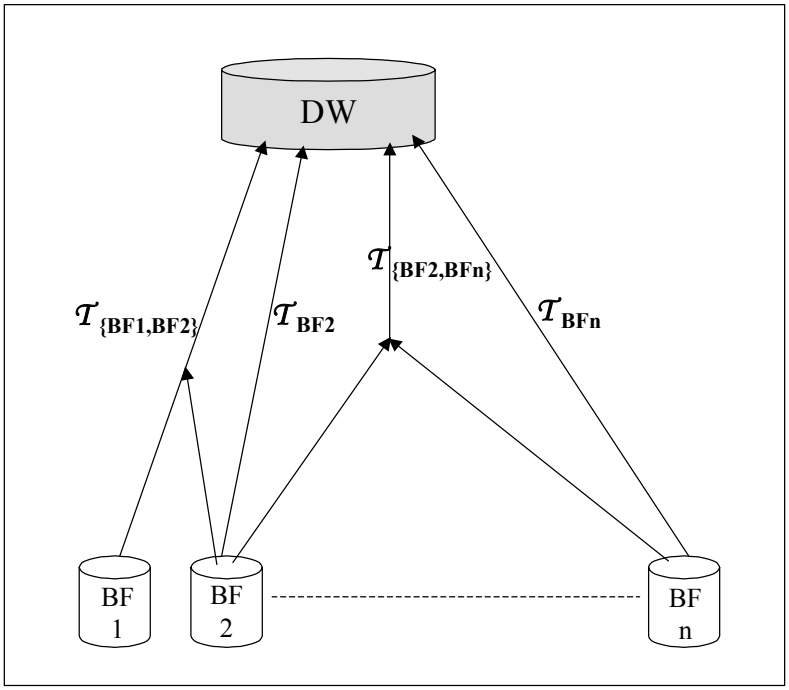

Figura 6: Traza multifuente

La visión de la traza multifuente de esta manera es de utilidad a la hora de configurar el workflow de carga de los datos al DW. Los flujos de datos que siguen las trazas tipo $\mathcal{T}_{BFi}$ son independientes y pueden ejecutarse en paralelo a los demás flujos de datos, mientras que los flujos de datos que siguen las trazas tipo $\mathcal{T}_{ei}$ dependen de flujos provenientes de distintas fuentes. En este ultimo caso puede darse, por ejemplo, que la transformación de

datos provenientes de una base fuente quede detenida esperando por datos que provienen de otra base que no esta disponible hasta algunas horas después.

5. Integración de Instancias

Cuando se quiere integrar datos provenientes de distintas fuentes es necesario resolver conflictos de datos. Considerando los casos presentados en la Sección 4.2 vemos que esto es necesario en los *Casos 2* y *3*. En el *Caso 2* siempre se debe hacer por lo menos un join entre datos de distintas fuentes (ya que para relacionar datos de una fuente con datos de otra se hace a través de datos en común) y puede suceder que entre los valores de los atributos que participan en el join haya conflictos de representación. Entonces es necesario determinar qué valores se corresponden ("matchean"). En el *Caso 3* sucede lo mismo que en el anterior pero además es necesario reconciliar datos, es decir a partir de dos valores provenientes de distintas fuentes que sabemos que se corresponden, obtener otro valor como resultado de la integración de ambos.

Nuestra propuesta para resolver los conflictos de datos se basa en 2 ideas: (a) una *clasificación dimensional* de los elementos de los esquemas [Mar00], y (b) las funciones *II-Match*, *II-Convert*, e *II-Reconcile*, que son una adaptación de las propuestas en [Cal99].

La *clasificación dimensional* permite tomar decisiones sobre como resolver la integración y en particular como resolver los conflictos de datos. A continuación damos algunos ejemplos de esto. Si estamos integrando dos atributos descriptivos de una dimensión, puede interesar que el resultado contenga los valores de ambos atributos (por ej., nros. telefónicos). Si estamos integrando dos atributos que son del tipo medida debería haber reconciliación entre sus valores, por ejemplo con un criterio de preferencia. Existen casos, cuando las medidas están a distinto nivel de granularidad, en que reconciliar los valores es muy difícil. Si lo que estamos integrando son atributos determinantes de un nivel de una jerarquía también deberían reconciliarse los valores eligiendo uno de los dos (por ej., si es un atributo que determina la empresa dueña de una sucursal).

Las funciones *II-Match*, *II-Convert*, e *II-Reconcile* se basan en la idea de las correspondencias propuestas en [Cal99]. En dicho trabajo se presentan *correspondencias inter-esquema* que permiten especificar declarativamente correspondencias entre los datos de distintos esquemas. Distinguen tres tipos de correspondencias: *conversion*, *matching* y *reconcilication*. La de *conversion* se utiliza para especificar que un dato de una fuente puede ser convertido en un datos de otra fuente o del DW. La de *matching* se usa para especificar como datos de diferentes fuentes pueden corresponderse ("matchear"). La de *reconciliation* se usa para establecer como datos de diferentes fuentes pueden reconciliarse en datos del DW.

6. Conclusiones

En este trabajo se presenta una extensión a la propuesta de [Mar00] para resolver el diseño de DW a partir de múltiples bases fuentes. Se propone un mecanismo en el cual se parte de un esquema de DW objetivo, se establecen correspondencias semánticas entre

este esquema y los esquemas fuentes, y luego se aplican primitivas de transformación de esquemas.

Se adapta el conjunto de primitivas de transformación de [Mar00] de forma de poder resolver los problemas de integración. Algunas primitivas se generalizan permitiéndose que se apliquen a varias fuentes, y se propone una nueva primitiva que permite la integración de dos relaciones. Esta primitiva permite, por medio de parámetros, elegir la modalidad en que se integran los esquemas y las instancias. Además se re-define la traza de transformación adecuándola al contexto de varias fuentes.

Se propone una clasificación de los sub-esquemas del DW en distintos casos según como sea la participación de las fuentes, la cual permite elegir qué primitivas aplicar y cuándo aplicarlas.

Con respecto a la carga de datos, la traza dejada por las primitivas provee el mapeo de los datos de cada fuente hacia el DW, quedando encapsulada mediante funciones la resolución de conflictos de datos.

Por último se comentan soluciones posibles para la resolución de conflictos en la integración de instancias. Vemos como posible trabajo futuro la profundización en la resolución de estos problemas.

Bibliografía

[Cal99] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati. *A principled approach to data integration and reconciliation in Data Warehouses.* International Workshop on Design and Management of Data Warehouses (DMDW'99), 1999.

[DoC00] A. Do Carmo. *Aplicando Integración de Esquemas en un contexto DW-Web.* Master thesis. Universidad de la República. Uruguay. 2000.

[Fan97] P. Fankhauser. *A Methodology for Knowledge-Based Schema Integration*. PHD-Thesis, Technical University of Vienna, December 1997.

[Gin00] Frédéric Gingras, Laks V. S. Lakshmanan: nD-SQL: A Multi-Dimensional Language for Interoperability and OLAP. VLDB 1998: 134-145.

[Mar00] A. Marotta. *Data Warehouse Design and Maintenance through schema transformations.* Master thesis. Universidad de la República. Uruguay. 2000.

[Mot02] R.Motz. *Dynamic Mantainance of an Integrated Schema,* PhD Thesis, Darmstadt University of Technology, Germany (forthcomming).