

Resolución de la Integración en el Diseño del Data Warehouse

Adriana Marotta

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

Resumen

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 proponen soluciones para diseñar el DW con el enfoque de [Mar02] pero a partir de múltiples fuentes, resolviéndose los problemas de integración.

Palabras claves: Data Warehouse, 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 que se realiza la consulta. Además se puede ver como *Global As View (GAV)*, ya que se estaría manejando una vista global en función de las bases locales. 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, siguiendo un enfoque *eager* y *LAV*. En el trabajo presentado en [DoC00] se adapta la metodología de integración SIM a un contexto de DW con información proveniente de la Web.

En [Mar00, Mar02] 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 [Mar02] 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 presentamos los problemas con que nos encontramos al aplicar el enfoque elegido, en la Sección 3 presentamos la propuesta concreta para generar el DW a partir de múltiples fuentes, en la Sección 4 comentamos los conceptos sobre los que nos basaríamos para la resolución de integración de instancias, y en la Sección 5 presentamos las conclusiones.

2. Enfoque elegido y problemas en su aplicación

Se decide aplicar un enfoque GAV, en el cual 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 1** se muestra la arquitectura correspondiente a este enfoque.

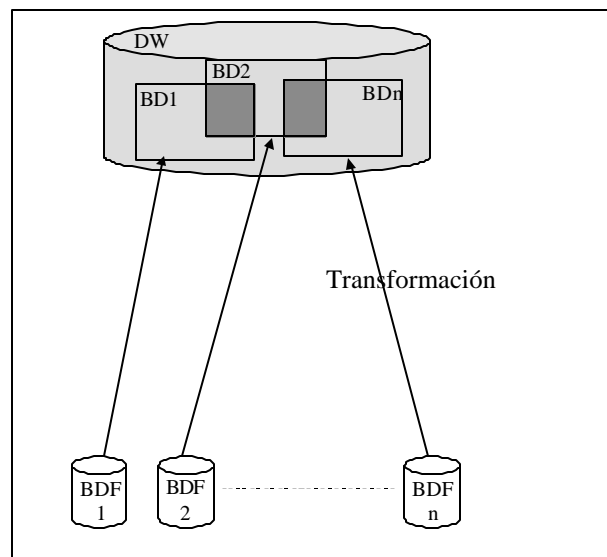


Figura 1: Arquitectura del proceso de diseño de DW. Enfoque GAV.

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 transformaciones, la cual deja una traza que nos permite cargar las instancias en el DW, (3) por último, se resuelve la integración de las instancias correspondientes a las intersecciones de los sub-esquemas.

A continuación mostramos, mediante un ejemplo, algunos problemas que nos encontramos con este enfoque.

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 cálculo para una etapa posterior de integración de instancias.

Para clarificar el problema que se expone se muestra un **ejemplo**:

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:

Se necesitan datos de BF2 para saber si el articulo es nacional, antes de agrupar.

..

BF2 → DW:

No tendría sentido llevar datos de BF2 al DW.

Como se ve en el ejemplo, 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.

3. 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, 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 transformaciones a las bases fuentes.

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 [Mar02] 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.

Los tipos de correspondencias semánticas utilizadas son las siguientes: de **equivalencia** (\circ), de **inclusión** (\hat{I}), de **solapamiento** (Q), de **partición vertical / horizontal** (**PV** / **PH**), de **agrupamiento** (\gg). Estos tipos de correspondencias se encuentran definidos y explicados en [Mot02]. Estas correspondencias ayudan a elegir qué transformaciones 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 el ejemplo de la Sección 2 se tendrían las siguientes correspondencias (*Notación*: $\{R1, \dots, Rn\}$ denota al sub-esquema formado por las relaciones $R1, \dots, Rn$):

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

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

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.

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.

Vemos en qué caso se encuentra el ejemplo que venimos manejando:

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*.

3.1. Modificación de transformaciones

En esta sección presentamos nuestra propuesta para darle a las transformaciones de esquemas para diseño de DW de [Mar02], la potencialidad necesaria para trabajar a partir de múltiples bases fuentes. Para esto realizamos dos modificaciones sobre el conjunto de las transformaciones: (1) modificamos la especificación de algunas transformaciones, y (2) creamos una nueva transformación 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 corresponden al mismo objeto del mundo real (equivalencia entre atributos es análogo a equivalencia entre sub-esquemas).

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 4.

Para la modificación (1) identificamos dentro del conjunto de transformaciones todas aquellas que, con mínimas modificaciones en su especificación, podrían involucrar más de una fuente. Entre estas están, por ejemplo, las transformaciones que agregan atributos calculados a una relación y dicho cálculo puede involucrar datos de otra relación. Mientras que, por ejemplo, una transformación que filtra atributos en una relación no estaría en este grupo.

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

T6 - DD-Adding

T8 - Hierarchy Roll-Up

T12 - Hierarchy Generation

T14 - New Dimension Crossing

A estas transformaciones modificadas les llamamos *Transformaciones Multifuente*. En la **Figura 2** se muestra como queda la especificación de una de ellas. Las modificaciones realizadas se encuentran resaltadas en negrita.

| <u>Transformation 6.2</u> | DD-ADDING N-1 |
|--|----------------------|
| Description: This transformation 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. | |
| Input: <ul style="list-style-type: none"> ▪ source schema : SDB₁.R₁ (A₁, ..., A_n), SDB₂.R₂ (B₁, ..., B_m) $\hat{\cap}$ Rel ▪ f (C₁, ..., C_k) / { C₁, ..., C_k } \subseteq { A₁, ..., A_n } \cup { B₁, ..., B_m }, where f is a user-defined function ▪ A / A \in { A₁, ..., A_n } \wedge B / B \in { B₁, ..., B_m }, join attributes ▪ is_fk , Boolean argument (declare A_{n+1} as a foreign key or not) ▪ R₃ \in Rel , relation to which A_{n+1} is a foreign key (optional) ▪ source instance : r₁, r₂ | |
| Resulting schema: <ul style="list-style-type: none"> ▪ R' ₁ (A₁, ..., A_n, A_{n+1}) \in Rel / A_{n+1} represents f (C₁, ..., C_k) \wedge if is_fk then A_{n+1} = Att_{FK}(R' ₁, R₃) | |
| Generated instance: <ul style="list-style-type: none"> ▪ r' ₁ = select A₁, ..., A_n, f (C₁, ..., C_k) from SDB₁.R₁ SDB₂.R₂ where II-Match (SDB₁.R₁.A, SDB₂.R₂.B) | |

Figura 2: Especificación de la Transformación DD-Adding N-1

Para la modificación (2) creamos una transformación **T15 – Relation Integration**.

Esta transformación recibe como argumentos: (a) 2 relaciones R1 y R2 provenientes de distintas fuentes, (b) un conjunto de triplas de conjuntos de atributos $\langle X, Y, Z \rangle$, 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 transformación es una relación cuyos atributos son la unión o intersección (según se haya pedido) de los atributos de las relaciones de entrada. La transformación además provee el pseudo-código de la transformación e integración de las instancias fuentes para la carga de la relación resultado.

En la **Figura 3** se presenta la nueva transformación.

| Transformation 15 | RELATION INTEGRATION |
|---|-----------------------------|
| Description: This transformation 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: <ul style="list-style-type: none"> source schema : $SDB_1.R_1 (A_1, \dots, A_n), SDB_2.R_2 (B_1, \dots, B_m) \in Rel$ $\{ \langle X_1, Y_1, Z_1 \rangle, \dots, \langle X_p, Y_p, Z_p \rangle \} / X_i \subseteq \{ A_1, \dots, A_n \} \wedge Y_i \subseteq \{ B_1, \dots, B_m \} \wedge X_i \equiv Y_i$ 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: <ul style="list-style-type: none"> $R' /$ 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: <ul style="list-style-type: none"> $r'_1 =$ if ai-type = "\cup" then if ii-type = "\cup" then $select (Att(SDB_1.R_1) - \cup_{i=1..p} X_i) \cup (Att(SDB_2.R_2) - \cup_{i=1..p} Y_i) \cup \{ II-Reconcile(X_1, Y_1), \dots, II-Reconcile(X_p, Y_p) \}$ | |

```

from SDB1.R1 SDB2.R2
where outer-join ( II-Match (SDB1.R1. X1, SDB2.R2. Y1) and
..... and II-Match (SDB1.R1. Xp, SDB2.R2. Yp) )
else if ii-type = “∩” then
select (Att(SDB1.R1) -  $\cup_{i=1..p} X_i$ )  $\cup$  (Att(SDB2.R2) -  $\cup_{i=1..p} Y_i$ )  $\cup$ 
{ II-Reconcile(X1, Y1), ....., II-Reconcile(Xp, Yp) }
from SDB1.R1 SDB2.R2
where II-Match (SDB1.R1. X1, SDB2.R2. Y1) and
..... and II-Match (SDB1.R1. Xp, SDB2.R2. Yp)
else if ai-type = “∩” then
if ii-type = “∪” then
select { II-Reconcile(X1, Y1), ....., II-Reconcile(Xp, Yp) }
from SDB1.R1 SDB2.R2
where outer-join ( II-Match (SDB1.R1. X1, SDB2.R2. Y1) and
..... and II-Match (SDB1.R1. Xp, SDB2.R2. Yp) )
else if ii-type = “∩” then
select { II-Reconcile(X1, Y1), ....., II-Reconcile(Xp, Yp) }
from SDB1.R1 SDB2.R2
where II-Match (SDB1.R1. X1, SDB2.R2. Y1) and
..... and II-Match (SDB1.R1. Xp, SDB2.R2. Yp)

```

Figura 3: Especificación de la Transformación Relation Integration

Ejemplo de una aplicación de la transformación Relation Integration:

Bases Fuentes: BDPersonal y BDContaduria

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

BDContaduria.Empleados (Nombre, Posición, Sueldo)

Aplicamos T15 con los siguientes parámetros:

X₁ = {Nombre, Apellido} Y₁ = {Nombre} Z₁ = {Nombre}

X₂ = {Cargo} Y₂ = {Posición} Z₂ = {Cargo}

ii-type = ∩

ai-type = ∪

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.

..

3.2. Re-definición de la traza de transformación

En [Mar02] se define una traza de transformación, la cual provee la información completa sobre las secuencias de transformaciones 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 se generaliza 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* (TT).

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

$$\mathbf{TT} = \{ \mathbf{T}_{BF1}, \dots, \mathbf{T}_{BFn}, \mathbf{T}_{e1}, \dots, \mathbf{T}_{em} \},$$

siendo: $\{BF1, \dots, BF_n\} \subseteq$ el conjunto de las bases fuentes

$$\{e1, \dots, em\} \subseteq \mathcal{P}_{\{BF1, \dots, BF_n\}} \text{ (conjunto potencia)}$$

Las trazas tipo \mathbf{T}_{BFi} , son trazas que mapean un sub-esquema del DW con una única fuente. Las trazas tipo \mathbf{T}_{ei} , son trazas que mapean un sub-esquema del DW con varias fuentes. La **Figura 4** ilustra esta idea.

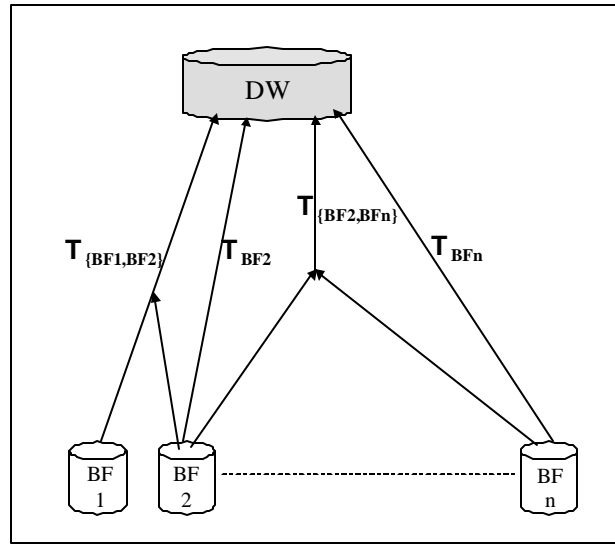


Figura 4: Trazas multifuente

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 T_{BF_i} son independientes y pueden ejecutarse en paralelo a los demás flujos de datos, mientras que los flujos de datos que siguen las trazas tipo T_{e_i} dependen de flujos provenientes de distintas fuentes. En este último 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 está disponible hasta algunas horas después.

4. 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 3 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 [Mar02], 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 *reconciliation*. La de *conversion* se utiliza para especificar que un

dato de una fuente puede ser convertido en un dato 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.

5. Conclusiones

En este trabajo se presenta una extensión a la propuesta de [Mar02] 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 transformaciones de esquemas.

Se adapta el conjunto de transformaciones de [Mar02] de forma de poder resolver los problemas de integración. Algunas transformaciones se generalizan permitiéndose que se apliquen a varias fuentes, y se propone una nueva transformación que permite la integración de dos relaciones. Esta transformación 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 transformaciones aplicar y cuándo aplicarlas.

Con respecto a la carga de datos, la traza dejada por las transformaciones 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 estos conflictos en la integración de instancias.

Consideramos como posibles trabajos futuros la validación del mecanismo propuesto, la cual podría ser abordada a través de aplicación a casos de estudio, y la profundización en la resolución de los problemas de conflictos en la integración de instancias.

Agradecimientos

Quisiera agradecer a la Profesora Regina Motz por valiosas discusiones y comentarios sobre el trabajo presentado en este artículo.

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.
- [Mar02] A. Marotta, R. Ruggia. *Data Warehouse Design: A Schema Transformation Approach*. Accepted paper. XXII Conferencia Internacional de la Sociedad Chilena de Ciencia de la Computación. 2002.
- [Mot02] R. Motz. *Dynamic Maintenance of an Integrated Schema*, PhD Thesis, Darmstadt University of Technology, Germany (forthcoming).