Diseño de Data Warehouses:

Un enfoque basado en transformación de esquemas

Adriana Marotta, Verónika Peralta
Facultad de Ingeniería – In.Co. – Universidad de la República - Uruguay
adriana@fing.edu.uy

Resumen

Los Data Warehouses (DW) son bases de datos que almacenan información para la toma de decisiones. Aunque los DW generalmente se implementan sobre DBMSs relacionales, los criterios de diseño tradicionales no son siempre válidos para el diseño del DW. Los trabajos existentes sobre diseño de DW se centran en sub-modelos (star, snowflake, etc.) y pierden generalidad en el tratamiento de técnicas de diseño.

Este artículo trata sobre los problemas de diseño de DW y propone un enfoque basado en transformaciones para construir el esquema del DW a partir de los esquemas fuente. Para ello se definen un conjunto de primitivas de diseño. Estas primitivas materializan el conocimiento sobre criterios de diseño y proveen un mecanismo para llevar la traza del diseño. Además se comportan como building blocks, que pueden ser compuestos para la construcción del esquema final. Este artículo presenta también un conjunto de reglas que aseguran la consistencia del resultado obtenido, y varias estrategias que proveen diferentes soluciones a los problemas típicos que deben ser resueltos durante el diseño de un DW.

1. Introducción

Los Data Warehouses (DW) son bases de datos que sirven de base para la toma de decisiones. Tienen algunas características clave: (i) las inserciones y actualizaciones de datos en el DW se hacen mediante procesos batch en lugar de OLTP, (ii) los datos del DW son generalmente el resultado de transformaciones de historización y control de calidad, (iii) el DW tiene que procesar consultas complejas (sumarizaciones, agregaciones, cruzamiento de datos), con performance razonable.

Las características del DW hacen que el proceso de diseño y las estrategias que se aplican para el mismo, generalmente sean diferentes a las tradicionales para bases de datos relacionales. Un ejemplo típico es la *redundancia de datos*, la cual a menudo es necesaria para mejorar la performance en el procesamiento de consultas complejas.

Este artículo trata el problema de diseñar un DW con un enfoque de transformación de esquemas. Más concretamente, con primitivas que abstraen y generalizan las técnicas de diseño de DW, para construir las relaciones del esquema del DW a partir de las relaciones del esquema fuente.

La definición de primitivas de diseño es interesante por dos motivos: primero, porque las primitivas materializan el conocimiento en criterios de diseño, y segundo, porque proveen una manera de llevar la traza del diseño. Además, aumentan la productividad del diseñador, ya que se comportan como building blocks de diseño, que pueden ser compuestos para llegar al esquema final. Este trabajo puede ser considerado como un paso importante hacia la introducción de las características de diseño de DW en una herramienta CASE.

El uso de primitivas de transformación de esquemas es una herramienta conceptual clásica en el área de bases de datos. En [1], se presentan primitivas de diseño y estrategias como building blocks de una metodología de diseño conceptual. En [2] y [3] se analiza el concepto de transformación de esquemas

El presente trabajo propone un conjunto de primitivas para el diseño de DW relacionales.

En general, el trabajo existente sobre diseño de DW consiste principalmente de técnicas para sub-modelos específicos (como star o snowflake) y patrones de diseño para áreas específicas. Algunos de

esos trabajos son [4], [5], [6], [7] y [8]. Aunque esos trabajos constituyen una preciada base de conocimiento para el diseño de DW, su aplicación práctica no es directa. Para hacerlo, los diseñadores deben incorporar este conocimiento, abstraer las reglas y estrategias de diseño, y luego aplicarlas en casos particulares.

El presente trabajo intenta abstraer y estructurar las técnicas y estrategias de diseño, en un conjunto de primitivas de transformación de esquemas. Además, incluye algunos lineamientos para su aplicación.

La contribución principal de este trabajo es la propuesta de un conjunto de primitivas. Estas son aplicadas sobre los esquemas fuente, más específicamente, a la integración de los mismos. Junto con cada primitiva, se provee de la especificación de las transformaciones que deben ser aplicadas sobre las instancias del esquema fuente, para poblar el DW generado.

Para la utilización de las primitivas, se proveen 2 tipos de lineamiento: un conjunto de reglas de consistencia que siempre deben ser aplicadas para asegurar la consistencia del resultado obtenido, y algunas estrategias que proveen diferentes soluciones a los problemas típicos que deben ser encarados durante el diseño de un DW.

El resto del artículo está organizado de la siguiente manera: La sección 2 presenta una visión general de las primitivas. La sección 3 provee las definiciones básicas: el modelo utilizado y un conjunto de invariantes del esquema. La sección 4 muestra las primitivas implementadas. La sección 5 es una visión general de algunas reglas y estrategias para la aplicación de las primitivas y la sección 6 concluye.

2. Las primitivas de transformación del esquema – Visión general

En nuestro enfoque, el diseño de un DW es un proceso que comienza con el esquema fuente y termina con el esquema resultado, el cual corresponde al esquema del DW. Este se genera por la aplicación de primitivas al esquema fuente y a los sub-esquemas intermedios[1] que son generados durante el proceso. Entonces, todos los elementos que constituyen el esquema final: relaciones y atributos, son el resultado de aplicar las primitivas al esquema fuente.

Arquitectura de la transformación

Figure 2.1 muestra la arquitectura básica de la transformación de un esquema fuente en un esquema de DW, a través de la aplicación de primitivas.

Las primitivas que diseñamos son transformaciones básicas que son aplicadas a un esquema relacional fuente para obtener un esquema relacional de DW. Concretamente, toman como entrada un sub-esquema y su salida es otro sub-esquema. Se presenta un esbozo de las transformaciones a ser aplicadas sobre una instancia fuente.

Agrupamos algunas de las primitivas en familias porque en algunos casos son alternativas para resolver el mismo problema, o más de un estilo de diseño que puede ser aplicado.

Las primitivas no conducen a una estrategia o metodología en particular. Es más, su aplicación sin criterios de diseño bien definidos, puede llevar a resultados no deseados, a veces, con problemas de consistencia. Para ayudar al diseñador en este aspecto proveemos lo siguiente: (i) la definición de invariantes del esquema de DW (Sección 3), (ii) estrategias para resolver los problemas típicos que aparecen en el diseño (Sección 5.1), y (iii) reglas de consistencia para la aplicación de las primitivas (sección 5.2).

[1] Consideramos un sub-esquema como un conjunto de relaciones que son parte de un esquema.

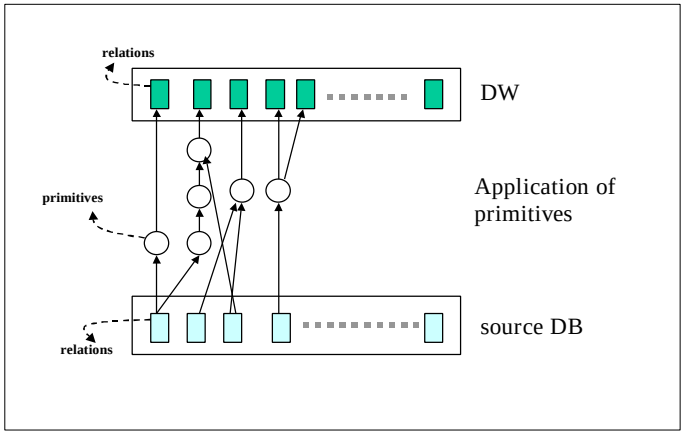

Figure 2.1

3. Definiciones básicas

El modelo subyacente para las primitivas de transformación propuestas es el Modelo Relacional. Además, los elementos relacionales (relaciones y atributos) están clasificados en diferentes conjuntos, de acuerdo a su comportamiento en el contexto del DW.

Algunos de los elementos clasificados son: relaciones de dimensión, relaciones de medida, atributos descriptivos, atributos de medida.

La clasificación permite a las primitivas realizar un tratamiento más refinado de las diferentes situaciones en el diseño del DW.

Los Conjuntos definidos sobre el Modelo Relacional

Conjuntos de Relaciones[2]:

Rel – Conjunto de todas las relaciones (cualquier clase de relación).

Rel_D – Conjunto de relaciones de "dimensión". Son las relaciones que representan información descriptiva sobre los sujetos del mundo real.

Rel_M – Conjunto de relaciones de "medida". Son las relaciones que representan relacionamientos o combinaciones entre los elementos de un grupo de dimensiones. Usualmente contienen atributos que representan medidas para las combinaciones.

Rel_C – Conjunto de relaciones de "cruzamiento". Son relaciones de medida que no tienen ningún atributo de medida.

Rel_J – Conjunto de relaciones de "jerarquía". Son relaciones de dimensión que contienen un conjunto de atributos que constituyen una jerarquía. El hecho de que exista una jerarquía entre un conjunto de atributos, sólo puede ser determinado por la semántica de los mismos.

$Rel_H(R)$ – Conjunto de relaciones de "historia". Son relaciones que tienen información histórica que corresponde a información de la relación R.

Estos conjuntos verifican las siguientes propiedades:

- $Rel_C \subset Rel_M$
- $Rel_J \subset Rel_D$
- $Rel_H \subset (Rel_D \cup Rel_M)$

[2] Usamos la palabra relación como un sinónimo de esquema relación.

Conjuntos de Atributos:

$Att(R)$ – Conjunto de todos los atributos de la relación R.

$Att_M(R)$ – Conjunto de atributos de medida de la relación R.

$Att_D(R)$ – Conjunto de atributos descriptivos de la relación R.

$Att_C(R)$ – Conjunto de atributos derivados (calculados) de relación R.

Att_J – Conjunto de conjuntos de atributos que representan una jerarquía.

$Att_K(R)$ – Conjunto de conjuntos de atributos que son clave de la relación R.

$Att_{FK}(R)$ – Conjunto de conjuntos de atributos que son clave externa de la relación R.

$Att_{FK}(R_1, R_2)$ – Conjunto de atributos que son clave externa de la relación R_1 respecto a la relación R_2.

Estos conjuntos verifican las siguientes propiedades:

- $Att_M(R) \cup Att_D(R) \cup Att_C(R) = Att(R)$
- $\forall X / X \in Att_J, X \subset \cup_{R \in Rel} Att_D(R)$
- $Att_{FK}(R) = \{ e / e = Att_{FK}(R, R_i) \}$, i=1..n, donde n es el número de relaciones con respecto al cual R tiene clave externa.
- $\forall A / A \in X$ y $X \in (Att_K(R) \cup Att_{FK}(R))$, $A \in Att_D(R)$
- Si $X \in Att_K(R)$ y $Y \in Att_{FK}(R)$, debe ser: $X \cap Y \neq \emptyset$

Invariantes del Esquema de DW

Los invariantes del esquema son un conjunto de propiedades que deben ser satisfechas por un esquema relacional de DW para ser consistente. Los invariantes son:

I1 - Integridad Referencial:

Cada clave externa declarada debe tener una clave primaria correspondiente en las relaciones a las que referencia. Además se debe referenciar a todas las relaciones con esa clave primaria que representen los mismos sujetos de mundo real.

$\forall X, R_1, R_2 / X = Att_{FK}(R_1, R_2)$, se cumple:

$X = Att_K(R_2) \wedge$

$\forall R / Att_K(R) = X \wedge$ R representa los mismos sujetos, $X = Att_{FK}(R_1, R)$

I2 - Datos Derivados:

Todos los atributos que son necesarios para calcular o derivar a los atributos derivados, deben existir en el esquema.

$\forall A, R / A \in Att_C(R) \wedge A = f(B_1, ..., B_n), \{B_1, ..., B_n\} \subset \cup_{R \in Rel} Att(R)$

I3 - Jerarquías:

Dado un conjunto de atributos X que representa una jerarquía, debe existir una dependencia funcional entre cada atributo de X y todos los atributos de X que corresponden a niveles más altos en la jerarquía.

Sea $X / X \in Att_J \wedge X = \{A_1, ..., A_n\} \wedge$

$A_1 < A_2 < .... < A_n$, donde a<b significa que b está en un nivel más alto en la jerarquía que a

Se cumple que:

$A_1 \rightarrow A_2, A_2 \rightarrow A_3, ..., A_{n-1} \rightarrow A_n$

I4 - Relaciones de historia :

Una relación de historia que corresponde a una relación con datos actuales, debe incluir una clave externa referenciando a la correspondiente relación actual.

Sea R_H / $R_H \in Rel_H(R)$, se cumple que: $\exists$ X / X = $Att_{FK}(R_H,R)$

I5 - Relaciones de Medida :

- Si una relación de medida tiene un atributo de alguna relación de dimensión, entonces debe tener una clave externa relativa a esa relación.

  Sea R_D, R_M / $R_D \in Rel_D \wedge R_M \in Rel_M$
  si $\exists$ A / A $\in Att(R_D) \wedge$ A $\in Att(R_M) \Rightarrow \exists$ X / X = $Att_{FK}(R_M,R_D)$

- Las relaciones de medida deben tener una dependencia funcional, cuyo lado izquierdo sea un conjunto de atributos que son clave externa de dimensiones y cuyo lado derecho son el resto de los atributos.

  Sea R_M, X / $R_M \in Rel_M \wedge$ X = $Att_{FK}(R_M)$, se cumple que X $\rightarrow$ ($Att(R_M)$ – X)

4. Las primitivas

La siguiente figura muestra una tabla que contiene el conjunto completo de las primitivas propuestas.

Primitiva		Descripción
P1	**Identity**	Dada una relación, genera otra que es exactamente igual a la original.
P2	**Data Filter**	Dad una relación, genera otra donde sólo algunos atributos son preservados. El objetivo es eliminar los atributos puramente operacionales.
P3	**Temporalization**	Agrega un elemento de tiempo al conjunto de atributos de una relación.
P4	**Key Generalization ***	Estas primitivas generalizan la clave primaria de una relación de dimensión, para poder almacenar más de una tupla de cada elemento de la relación.
P5	**Foreign Key Update**	A través de esta primitiva, una clave externa y sus referencias pueden ser cambiadas en una relación. Es útil cuando las claves primarias son modificadas.
P6	**DD-Adding ***	Estas primitivas agregan a una relación, un atributo que es derivado de otros.
P7	**Attribute Adding**	Agrega atributos a una relación de dimensión. Puede ser usada para mantener en la misma tupla más de una versión de un atributo.
P8	**Hierarchy Roll Up**	Esta primitiva efectúa el roll up por uno de los atributos de una relación siguiendo una jerarquía. Además, puede generar otra relación de jerarquía con el correspondiente nivel de detalle.
P9	**Aggregate Generation**	Dada una relación de medida, genera otra relación de medida, donde los datos están resumidos (o agrupados) por un conjunto de atributos dado.
P10	**Data Array Creation**	Dada una relación que contiene un atributo de medida y un atributo que representa un conjunto pre-determinado de valores, esta primitiva genera una relación con una estructura de array.
P11	**Partition by Stability ***	Estas primitivas particionan una relación, organizando el almacenamiento de datos históricos. Se puede aplicar Vertical Partition o Horizontal Partition, dependiendo del criterio de diseño utilizado.
P12	**Hierarchy Generation ***	Es una familia de primitivas que genera una jerarquía de relaciones, teniendo como entrada, las relaciones que incluyen parte de la jerarquía.
P13	**Minidimension Break off**	Esta primitiva elimina un conjunto de atributos de una relación de dimensión, construyendo una nueva relación con ellos.
P14	**New Dimension Crossing**	Esta primitiva permite materializar un cruzamiento de dimensión en una nueva relación. También es útil para de-normalizar relaciones, lo que mejora la performance de las consultas.

Las primitivas que proponemos son operaciones de transformación que pueden ser aplicadas a un esquema para hacerlo más adecuado para las consultas que se le harán.

Los siguientes ejemplos ilustran su utilidad.

Muchas relaciones en los sistemas operacionales no mantienen una noción temporal. Por ejemplo, las relaciones 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 relaciones necesitan incluir un elemento temporal, de manera que puedan mantener información histórica. Para este propósito existe una primitiva llamada **Temporalization** que agrega un elemento de tiempo al conjunto de atributos de una relación.

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 calculados, por razones de performance. Tenemos un grupo de primitivas, cuyo nombre es **DD-Adding**, que agrega a una relación un atributo derivado a partir de otros.

La figura 4.1 muestra una tabla que contiene el conjunto completo de las primitivas propuestas.

Ejemplo de especificación de una primitiva

En esta sección mostramos la especificación de una de las primitivas. La especificación del conjunto completo de primitivas se puede encontrar en [9].

Primitiva P9 – Aggregate Generación

Descripción:

Dada una relación de medida, la primitiva genera otra relación de medida, donde los datos son resumidos (o agrupados) por un conjunto dado de atributos.

Entrada:

- esquema fuente: $R (A_1, ..., A_n) \in elm$
- Z , conjunto de atributos / card(Z) = k (medidas)
- $\{ e_1, ..., e_k \}$, expresiones de agregación
- $Y / Y \subset \{ A_1, ..., A_n \} \wedge Y \subset (Att_D(R) \cup Att_M(R))$, atributos a ser borrados
- instancia fuente: r

Esquema Resultado :

$R' (A'_1, ..., A'_m) \in Rel_M$ / $\{ A'_1, ..., A'_m \} = \{ A_1, ..., A_n \} - Y \cup Z$

Instancia Generada:

r' = select ($\{ A'_1, ..., A'_m \} - Z$) $\cup$ $\{ e_1, ..., e_k \}$
from R
group by $\{ A'_1, ..., A'_m \} - Z$

Ejemplo:

Se tiene una relación con la cantidad vendida por cliente, vendedor, mes, producto y ciudad.

VENTAS_MENSUALES

CLIENTE	VENDEDOR	MES	PROD	CIUDAD	CANTIDAD
Juan	Pedro	1/98	25	Montevideo	5
Juan	Pedro	1/98	7	Colonia	7
Juan	Maria	2/98	4	Montevideo	1
Juan	Laura	2/98	4	Maldonado	5
Luis	Pedro	1/98	100	Montevideo	2
Luis	Laura	1/98	100	Montevideo	6
Luis	Laura	4/98	100	Canelones	3

Ahora queremos almacenar la cantidad vendida para cada cliente, en cada mes, de cada producto. Entonces se agrupará por CLIENTE, MES, PRODUCTO.

Aplicamos la primitiva **P9**, donde la entrada es:

- R = **VENTAS_MENSUALES**
- Z = { CANTIDAD }, card(Z) = k = 1, la medida que queremos
- { e_1, ..., e_k } = { sum(CANTIDAD) }
- Y = { VENDEDOR, CIUDAD }
- r = tuplas de VENTAS_MENSUALES

Resultado:

VENTAS_CLIENTE_MES_PROD

CLIENTE	MES	PROD	CANTIDAD
Juan	1/98	25	5
Juan	1/98	7	7
Juan	2/98	4	6
Luis	1/98	100	8
Luis	4/98	100	3

5. Diseñando el DW a través de las primitivas: estrategias y reglas para aplicarlas

Cuando diseñamos un esquema de DW a través de las primitivas, debemos considerar dos aspectos (relacionados al esquema obtenido): (i) la consistencia del esquema y (ii) como se ajusta el esquema a los requerimientos del DW. Consideramos que un esquema es consistente si satisface los invariantes definidos en la sección 3. Los requirementos del DW, los cuales usualmente consisten de consultas complejas que implican largos volúmenes de datos, son los que determinan las estructuras de datos más convenientes para el esquema del DW. Un buen diseño debe estructurar los datos de manera que las consultas puedan ser satisfechas lo más eficientemente posible.

Para ayudar al diseñador a hacer un diseño que se ajuste a los requerimientos de su DW, proveemos un conjunto de estrategias, algunas de las cuales son presentadas en la siguiente sección. Al final, mostramos algunas reglas cuyo objetivo es asegurar la consistencia del esquema generado.

5.1. Estrategias de Diseño del DW

Las estrategias para la aplicación de primitivas fueron diseñadas teniendo en cuenta algunos problemas típico de DW y deberían ser útiles para solucionarlos.

Las estrategias propuestas resuelven problemas de diseño relativos a: versionamiento de dimensiones, versionamiento de relaciones de dimensión N:1, sumarización y cruzamiento de datos, manejo de jerarquías, y datos derivados. Debido a limitaciones de espacio, sólo presentamos un grupo de estrategias. (Para las otras estrategias propuestas referirse a [9]).

S1 - Dimension versioning

Los sujetos del mundo real representados en las dimensiones, usualmente evolucionan a través del tiempo. Por ejemplo, un cliente puede cambiar su dirección, un producto puede cambiar su descripción o tamaño. A veces se necesita mantener la historia de esos cambios en el DW. En algunos de esos casos es necesario almacenar todas las versiones del elemento de manera que la historia completa sea mantenida. En otros casos, sólo un número fijo de valores de ciertos atributos deben ser almacenados. Por ejemplo, puede ser útil mantener el valor actual de un atributo y el anterior, o el valor actual y el original.

Un problema usual que los diseñadores tienen que enfrentar es como manejar el versionamiento de dimensiones. Esto se refiere a como estructurar la información de una dimensión cuando se necesita mantener la historia. La idea es mantener versiones de la información de cada sujeto del mundo real.

Se proveen algunas alternativas. En todas ellas se genera una nueva relación de dimensión, donde se mantiene los datos históricos sobre los sujetos.

Las siguientes son posibles estrategias para aplicar:

1) Aplicar la primitiva **Temporalization** (P3), de manera que el atributo de tiempo pertenezca a la clave de la relación.
2) Generalizar la clave de la relación de dimensión a través de una las primitivas de la familia de **Key Generalization** (P4).
3) Agregar nuevos atributos, de manera que un número pequeño de versiones de ciertos datos pueda ser mantenido. Se logra aplicando la primitiva **Attribute Adding** (P7).
4) Generalizar la clave de la relación siguiendo las alternativas **2.1** o **2.2**, y agregar un atributo de tiempo que no pertenezca a la clave (P4.1, P3 o P4.2, P3).
5) Particionar la relación de acuerdo a su estabilidad con una de las primitivas de **Partition by Stability** (P11).

El siguiente es un ejemplo que muestra la aplicación de algunas de esas alternativas.

CLIENTES

SSN	NOMBRE	EDAD	SUELDO	DIRECCION	SEXO	CIUDAD	EC
276052	R. Mendez	20	10000	Bvar. Artigas 3	F	Montevideo	S
342587	S. Nunez	30	15000	J. Herrera y Ob	M	Montevideo	C
431222	M. Garcia	20	10000	Garzon 2125	F	Salto	S
213438	L. Lopez	50	5000	18 de Julio 643	M	Colonia	C

2 opciones

CLIENTES_1

GRAL_SSN	NOMBRE	.....
01276052	R. Mendez	.....
01342587	S. Nunez	.....
01431222	M. Garcia	.....
01213438	L. Lopez	.....

CLIENTES_2

SSN	FECHA	NOMBRE	.....
276052	1/1/93	R. Mendez	.....
342587	23/4/97	S. Nunez	.....
431222	5/2/98	M. Garcia	.....
213438	3/3/99	L. Lopez	.....

5.2. Reglas de Consistencia

Estas son algunas reglas que deben ser aplicadas siempre, cuando un esquema de DW es construido a través de la aplicación de las primitivas.

Las reglas consideran los diferentes casos de inconsistencias que pueden ser generados por la aplicación de primitivas y propone las acciones correctoras.

R1 – Actualización de la clave externa

R1.1 –

AL APLICAR: *Temporalization* o *Key Generalization* a R, donde X= clave vieja,Y= clave nueva

APLICAR: *Foreign Key Update* a todas las R_i / $Att_{FK}(R_i,R)$ = X, obteniendo $Att_{FK}(R_i,R)$ = Y

R1.2 –

AL APLICAR: *Vertical Partition* a R con clave X, obteniendo R_1, R_2, R_3, clave X en cada caso

APLICAR: *Foreign Key Update* a todas las R_i / $Att_{FK}(R_i,R)$ = X, obteniendo $Att_{FK}(R_i,R_1)$ = X, $Att_{FK}(R_i,R_2)$ = X, $Att_{FK}(R_i,R_3)$ = X

R2 – Attribute recovering

AL APLICAR: *Data Filter*, *Hierarchy Roll Up*, *Aggregate Generation* o *New Dimention Crossing* a R, borrando A $\in$ *Att(R)*

CUANDO: $\exists$ B , $\exists$ R' / B $\in$ $Att_C(R')$ $\wedge$ A es usado para el cálculo de B

APLICAR: *Identity* a R

R3 – History relations update

AL APLICAR: *DD-Adding*, *Attribute Adding*, *Hierarchy Generation*, *Aggregate Generation* o *Data Array Creation* a R, agregando A / A $\in$ *Att(R)*

CUANDO: $\exists$ R' / R' $\in$ $Rel_H(R)$ $\wedge$ A se desea mantener en R'

APLICAR: *Attribute Adding* a R', obteniendo A $\in$ *Att(R')*

6. Conclusión

Este artículo presenta un conjunto de primitivas de transformación de esquemas, así como algunas estrategias y reglas para su aplicación práctica. Estas primitivas permiten diseñar un DW desde un esquema fuente actuando como building blocks de diseño que tienen embebido en su semántica el conocimiento de diseño de DW.

Las primitivas proveen una traza de diseño, la cual es de gran importancia para la documentación y la administración del proceso de diseño. Además, las primitivas permiten repercutir la evolución del esquema fuente en un contexto de DW. Específicamente, en DW con gran evolución de las bases fuente (por ejemplo, con datos extraídos del Web [10],[11]), permiten la repercusión de los cambios del esquema fuente al DW.

En nuestra propuesta la consistencia del esquema es manejada a través de las *invariantes* y *reglas* del esquema de DW. Mientras los *invariantes* especifican las condiciones de consistencia que los esquemas de DW deben satisfacer, las *reglas* proveen transformaciones adicionales al esquema para mantener el esquema de DW en un estado consistente.

La principal contribución de este artículo es la propuesta de un conjunto de primitivas de diseño complementado con estrategias y reglas para su aplicación. Creemos que es directa la definición de metodologías bien estructuradas para el diseño de DW y su implementación en una herramienta CASE.

En lo concerniente al alcance de las primitivas, las estrategias de diseño presentadas muestran como un gran espectro de los problemas de diseño pueden ser solucionados a través de la aplicación de las primitivas.

El trabajo futuro cubre diferentes tópicos: experimentación con las primitivas aplicándolas en el desarrollo de DW reales [12], focalizando en el uso de las primitivas como traza del diseño y evaluar la repercusión de la evolución del esquema fuente, y la especificación de las transformaciones de la instancia.

Actualmente, estamos implementando las primitivas en una herramienta de diseño de DW que permita al diseñador aplicar las primitivas través de una interface gráfica [13]. Además la herramienta debería incluir lineamientos de diseño basándose en estrategias definidas e implementar las reglas de corrección.

La versión actual de las primitivas no incluye las facilidades de integración de esquemas. Consideramos que ese es un problema en si mismo, el cual involucra aspectos específicos como la especificación de correspondencias de conceptos, resolución de conflictos, mezcla de esquemas, etc. De todos modos creemos que las primitivas permiten de algún modo implementar la integración de esquemas.

7. Referencias

[1] Batini, Ceri, Navathe. *Conceptual Database Design. An Entity-Relationship Approach.* The Benjamin/Cummings Publishing Company, Inc. 1992

[2] J. L. Hainaut. *Entity-Generating schema transformations for Entity-Relacioneship models.* ER 1991: 643 – 670.

[3] B. Staudt Lerner, A. Nico Habermann. *Beyond Schema Evolution to Database Reorganization.* ECOOP/OOPSLA 1990 Proceedings.

[4] R. Kimball. *The Data Warehouse Toolkit.* J. Wiley & Sons, Inc. 1996

[5] R. Kimball. *The Data Warehouse Lifecycle Toolkit.* J. Wiley & Sons, Inc. 1998

[6] R. Kimball. *Data Warehouse Architect.* Column in DBMS online magazine. 1997

[7] C. Adamson, M. Venerable. *Data Warehouse Design Solutions.* J. Wiley & Sons, Inc. 1998

[8] L. Silverston, W. H. Inmon, K. Graziano. *The Data Model Resource Book.* J. Wiley & Sons, Inc. 1997

[9] A. Marotta. *A transformations based approach for designing the Data Warehouse.* Technical Report. In.Co. Facultad de Ingeniería - Montevideo – Uruguay.

[10] R. Hackathorn. *Reaping the Web for your Data Warehousing.* DBMS, August 1998.

[11] R. D. Hackathorn. *Web Farming for the Data Warehouse.* Morgan Kaufmann Publishers, Inc. San Francisco, California. 1999

[12] R. Abella, L. Coppola, D. Olave. *A Data warehouse for the Engineering Faculty.* Graduate Project. In.Co. Facultad de Ingeniería - Montevideo – Uruguay. 1998.

[13] P. Garbusi, F. Piedrabuena, G. Vazquez. *Disign and Implementation of a schema transformation based DW design tool.* Graduate Project. In.Co. Facultad de Ingeniería - Montevideo – Uruguay. On going work 1999.