

Apéndice A

EFECTOS DE LAS OPERACIONES DE LA TAXONOMÍA DE EVOLUCION SOBRE LAS PRIMITIVAS

Versión 2.0

1- Introducción

El objetivo de este documento es estudiar los posibles efectos que puedan surgir de aplicar la evolución en la DataWarehouse (DW). Luego de haber usado las primitivas para diseñar la misma pueden ocurrir cambios en la base fuente que produzcan inconsistencias en la DW. A continuación se hace un estudio de los efectos que podrían generar los posibles cambios. Se efectúa un estudio por primitiva.

2- Primitivas

A continuación se enumeran las primitivas sobre las cuales se hará el estudio. Si bien estas son las primitivas actualmente disponibles el diseño permite que se agreguen nuevas.

- P1- Identity
- P2- Data Filter
- P3- Temporalization
- P4- Group Key Generalization
 - P4.1- Version Digits
 - P4.2- Key Extension
- P5- Foreign Key Update
- P6- Group DD-Adding
 - P6.1- DD-Adding1-1
 - P6.2- DD-Addingn-1
 - P6.3- DD-Addingn-n
- P7- Attribute Adding
- P8- Hierarchy Roll Up
- P9- Agregate Generation
- P10- Data Array Creation
- P11- Group Partition By Stability
 - P11.1- Vertical Partition
 - P11.2- Horizontal Partition
- P12- Group Hierarchy Generation
 - P12.1- De-Normalized
 - P12.2- SnowFlake
 - P12.3- Free Decomposition
- P13- Minidimension Break Off
- P14- New Dimension Crossing

3- Operaciones de la taxonomía

La taxonomía seleccionada para representar los posibles cambios al esquema fuente es la siguiente:

- 1- Renombrar Atributo
- 2- Agregar Atributo
- 3- Eliminar Atributo (el atributo no puede ser clave primaria)

- 4- Cambiar la clave de una relación
- 5- Renombrar Relación
- 6- Agregar Relación
- 7- Borrar Relación

El diseño también está pensado para que se puedan agregar operaciones. Si bien aquí se listan todas las operaciones disponibles, sólo algunas podrán causar inconsistencias en la DW. Por ejemplo la operación Renombrar Atributo difícilmente pueda generar alguna inconsistencia ya que al renombrar un atributo en la base fuente al propagar los cambios a lo sumo se renombrará algún Atributo más en las Relaciones intermedias, la DW ni siquiera se ve afectada ya que la traza absorbe los cambios. Sin embargo al borrar un Atributo se debe tener más cuidado en evitar efectos secundarios que puedan hacer que se pierda la semántica de alguna primitiva o que la aplicación de la misma ya no tenga sentido o quizás haya que eliminarla porque ni siquiera es posible que la primitiva sea aplicable después de realizar los cambios. Por ejemplo, es posible que se borre un Atributo que sea usado como único parámetro en una función de cálculo. Es posible que esa función sólo pueda ser calculada con ese atributo como parámetro y sea imposible sustituir el atributo que es parámetro de entrada de la función por otro. Si el atributo se borra, la función no puede aplicarse más y es posible que la primitiva deje de tener sentido.

Las operaciones 5, 6 y 7 se aplican sobre Relaciones y no se estudian. Al aplicar alguna de estas operaciones se podrán agregar o borrar primitivas pero nunca se producirá algún efecto inesperado.

4- Estudio de las posibles inconsistencias

A continuación se efectúa el estudio por primitiva y se marcan con negrita las operaciones de la taxonomía que podrían causar problemas. También estudiamos otros casos que si bien no producen efectos inesperados consideramos que vale la pena comentarlos. Estos casos se estudian aparte pero no se resaltan en negrita.

4.1 Identity

En ésta primitiva no encontramos ninguna operación básica que produzca efectos inesperados ya que ésta primitiva hace una copia exacta de la Relación origen y cualquier cambio que se produzca en ésta puede ser reflejado en forma idéntica en la Relación destino sin que aparezcan inconsistencias.

4.2 Data Filter

En las operaciones Renombrar Atributo y Agregar Atributo no encontramos ningún tipo de efecto inesperado.

Remover atributo. Si aplicamos la operación Remover atributo en la relación de entrada de una instancia de Data Filter pueden pasar dos cosas: 1- que el atributo eliminado de la relación de entrada haya sido filtrado por la primitiva en cuyo caso la relación de salida no se ve afectada y 2- que el atributo eliminado no haya sido filtrado en cuyo caso se elimina el mismo atributo de la relación de salida, en ese caso la primitiva seguiría actuando de la misma forma. Podrían encontrarse casos de borde donde por

ejemplo el atributo a eliminar sea el único que no se filtra, al eliminarse éste, la Relación de salida queda vacía y como consecuencia se elimina la primitiva y posiblemente varios caminos en la traza.

Cambiar la clave. Si cambiamos la clave en una Relación que es entrada de Data Filter habría que tener especial cuidado si la clave se cambia por algún Atributo que fue filtrado en la DW (si bien la finalidad de Data Filter es eliminar Atributos puramente operacionales, ésta primitiva no tiene forma de saber que es lo que se filtra, y un mal diseño podría llevar a que se cambie la clave a algún Atributo filtrado). En éste caso se aplica la regla R6 (que lo que hace es agregar en la relación de la DW los atributos que van a ser nueva clave en la relación del esquema fuente) y como el Atributo que pasa a ser clave fue filtrado en la DW éste se agrega en la misma. Podría quererse filtrar la clave vieja al agregar la nueva, esto no lo hace la regla de propagación.

4.3 Temporalization

En las operaciones Remove Atributo, Renombrar Atributo y Agregar Atributo no encontramos ningún tipo de efecto inesperado.

Cambiar la clave. La Primitiva Temporalization nos da la opción de extender la clave agregando el nuevo atributo de tiempo a la clave primaria. En este caso podrían surgir problemas al cambiar la clave que no estén contemplados en ninguna regla de propagación. Para ilustrar se muestra un ejemplo muy simple (aunque poco real): Supongamos que tenemos la relación R1 con los siguientes atributos:

$R1 = \{ \text{Nombre, Apellido, Teléfono} \}$ PK = [Nombre]

y que queremos aplicarle a esta relación la primitiva Temporalization agregando un nuevo atributo llamado Time e incluirlo como parte de la clave. La relación de salida la llamamos R2 y tiene los siguientes atributos:

$R2 = \{ \text{Nombre, Apellido, Teléfono, Time} \}$ PK = [Nombre, Time]

Supongamos que queremos ahora cambiar la clave de R1 de Nombre a Apellido. El atributo Nombre es copiado en R2 y la regla de propagación nos pregunta si el atributo Nombre es clave primaria o foránea en R2 y vemos que ni es clave primaria ni foránea ya que es parte de la clave primaria pero no es clave porque se agregó el atributo Time a la clave. No tenemos en este caso ninguna regla de propagación.

4.4 Group Key Generalization

4.4.1 Version Digits

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados

4.4.2 Key Extension

En las operaciones Renombrar Atributo, Agregar Atributo y Remover Atributo no encontramos ningún tipo de efecto inesperado.

Cambiar la clave. Si cambiamos la clave en una relación que sea entrada de una primitiva Key Extension debemos aplicar la regla de propagación R6 ya que todos los Atributos de la Relación de salida son copiados de la relación de entrada. En ésta regla se plantean dos acciones diferentes: una si el Atributo de la DW es clave de la Relación en el esquema fuente y otra si es clave foránea en el mismo. Éste caso no se encuentra dentro de ninguno de esos dos, en la DW la clave a original es parte de la clave actual ya que la finalidad de la primitiva es justamente extender la clave. Este problema es similar al que puede ocurrir en Temporalization.

4.5 Foreign Key Update

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados

4.6 Group DD-Adding

4.6.1 DD-Adding1-1

En las operaciones Renombrar Atributo, Agregar Atributo y Cambiar la Clave no encontramos ningún tipo de efecto inesperado.

Remover Atributo. En éste caso al Remover un Atributo se aplican las reglas de propagación R3 y R4. La aplicación de R3 es trivial por lo que se discute la aplicación de R4. Los Atributos de la base fuente son usados en el cálculo del Atributo a agregar por lo que si se borra uno, en el caso que éste participe en la función de cálculo el usuario debe decidir si cambia la función de cálculo o elimina el Atributo calculado. Cambiar la función de cálculo no siempre es posible por lo que habría casos en que el Atributo agregado necesariamente debería ser borrado. Esto es consistente con la regla de propagación pero sin embargo si se borrara el Atributo agregado por DD-Adding la primitiva aplicada dejaría de ser ésta y pasaría a tener la semántica de la identidad ya que el único atributo que se agregó ahora se borra. En este caso podría ser preferible eliminar la primitiva ya que su aplicación puede dejar de tener sentido.

4.6.2 DD-Addingn-1

En las operaciones Renombrar Atributo, Agregar Atributo y Cambiar la Clave no encontramos ningún tipo de efecto inesperado.

Remover Atributo. Si se borra un Atributo que participa en la función de cálculo y no es posible quitar éste Atributo de la función estamos en las mismas condiciones que en el caso 4.6.1. Si se borra el Atributo de join entre las Relaciones se aplica la regla de propagación R5 y la primitiva entera debe eliminarse ya que no es posible su aplicación sin un Atributo de join entre las Relaciones.

4.6.3 DD-Addingn-n

Éste caso es análogo al anterior

4.7 Attribute Adding

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados

4.8 Hierarchy Roll Up

En las operaciones Renombrar Atributo y Agregar Atributo no encontramos ningún tipo de efecto inesperado.

Remove Atributo. En éste caso puede haber problemas si el atributo que se borra es calculado en la DW y ninguna de las dos acciones a tomar en las reglas de propagación R4 es adecuada (Remove el atributo en la DW o cambiar la función de cálculo).

Ejemplo:

En el capítulo 3 de la tesis se ilustra un ejemplo donde se hace un Hierarchy Roll Up desde DATE hasta MONTH (se escalan dos niveles en la jerarquía). El Atributo MONTH si bien se copia igual de la Relación origen es un Atributo calculado usando como función de cálculo la igualdad, no se copia como Att_cpy. Esta diferencia radica en que se requieren mas Atributos (DATE en este caso) y no es sino la operación Att_calc que permite éste tipo de cálculos. Si deseáramos borrar el Atributo MONTH del esquema fuente aplicamos entonces la regla de propagación R4. En las acciones nos da la opción de borrar el Atributo correspondiente en la DW o cambiar la función de cálculo.

Si decidiéramos optar por borrar el Atributo correspondiente en la DW deberíamos borrar el Atributo MONTH y su camino desde el esquema fuente. Esta decisión no es correcta porque la DW quedaría con la Relación pero el Atributo MONTH desaparece que es un Atributo fundamental de la Relación porque es justamente el que se eligió para hacer el Roll Up. En éste caso la primitiva permanecería pero su resultado no sería el esperado y la aplicación de la primitiva dejaría de tener sentido.

La segunda opción tampoco sería correcta porque la única función adecuada para función de cálculo es la igualdad y hay Atributos requeridos que no permiten que se aplique la operación básica Att_cpy.

Cambiar la clave. Podría suceder en algunos casos que se agregue un nuevo Atributo a una Relación y que luego se cambie la clave a éste nuevo Atributo. En caso de que la antigua clave sea requerida para el cálculo de un Atributo de la DW se nos presentan dos opciones, eliminar el Atributo de la DW o cambiar el Atributo requerido. Puede haber casos en donde ninguna de estas alternativas sea la adecuada.

Ejemplo:

Usando el mismo ejemplo anterior supongamos que se agrega un nuevo Atributo Time_Key en la Relación TIME y luego se cambia la clave desde DATE a Time_Key. Al hacer éste cambio se debe aplicar la regla de propagación R7 ya que DATE es requerido para el cálculo de MONTH en la DW como veíamos en el ejemplo anterior. Ésta regla permite los dos tipos de acciones mencionados anteriormente pero resulta que ninguno de ellos sirve ya no podemos borrar el Atributo MONTH o quedaríamos en las

mismas condiciones del ejemplo anterior. Tampoco podríamos cambiar el Atributo requerido en la función de cálculo ya que Time_Key (el nuevo Atributo agregado) no existe en la Relación MONTH_SALES de la DW y si observamos los Atributos requeridos en la DW vemos que también se necesitaría que el Atributo Time_Key figure en MONTH_SALES (ver la descomposición de Hierarchy Roll Up en operaciones básicas).

En caso que se dé esta situación tenemos dos alternativas: o bien no se cambia la clave o en el caso de que se decida efectuar el cambio se debe eliminar toda la primitiva ya que esta pasa a un estado inconsistente. Además de eliminarse la primitiva se debe eliminar todo lo que involucre a esta ya que luego pueden venir más primitivas y estas probablemente tampoco puedan aplicarse.

4.9 Aggregate Generation

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados

4.10 Data Array Creation

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados

4.11 Group Partition by Stability

4.11.a Vertical Partition

En ésta primitiva los cambios en la base fuente no producen resultados inesperados en la DW.

4.11.b Horizontal Partition

En ésta primitiva los cambios en la base fuente no producen resultados inesperados en la DW.

4.12 Group Hierarchy Generation

4.12.1 De-Normalized Hierarchy Generation

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados

4.12.2 Snowflake Hierarchy Generation

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados

4.12.3 Free Decomposition Hierarchy Generation

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados

4.13 Minidimension Break Off

En ésta primitiva no encontramos ningún cambio que produzca efectos inesperados.

4.14 New Dimension Crossing

En las operaciones Renombrar Atributo y Agregar Atributo no encontramos ningún tipo de efecto inesperado.

Remover Atributo. Para ser aplicada ésta primitiva se exige que las Relaciones de entrada tengan un Atributo en común. Si éste Atributo se elimina de dejarían de cumplir los requerimientos para la aplicación de la primitiva y ésta debería eliminarse. Las reglas de propagación no son aplicables en éste caso.

Ejemplo:

En el capítulo 3 de la tesis se aplica New Dimension Crossing con dos Relaciones cuyo único Atributo en común es COURSE, si éste Atributo se elimina de alguna de las dos Relaciones, el Atributo Z que se exige que esté en ambas Relaciones ya no existe y por lo tanto la aplicación de la primitiva ya no es posible. Éste Atributo COURSE no es ni calculado, ni requerido para el cálculo ni copiado por lo que lo clasificaríamos como Atributo sin dependencias en la DW y por lo tanto no se aplicaría ninguna regla de propagación.

Cambiar la clave. Al cambiar la clave hay que tener en cuenta lo que hace la primitiva, definir una clave en la nueva Relación que es la unión de las claves de las Relaciones del esquema fuente. En éste caso correspondería aplicar la regla de propagación R6, sin embargo habría que ver cuales son los pasos a seguir si el Atributo en la DW es parte de la clave y no la clave entera. Esto se podría dar al aplicar New Dimension Crossing porque se define como clave de la Relación resultante a la unión de claves de las Relaciones originales y en la regla R6 se estudia si el Atributo es clave primaria o clave foránea en la DW. Habría que adaptar las acciones cuando el Atributo es parte de la clave primaria que no es ninguno de estos dos casos. Esta situación es análoga a Cambiar la clave en la primitiva Key Extension (4.4.2) y Temporalization (4.3).

5- Soluciones

En el punto anterior se introdujeron varias posibles inconsistencias que se pueden dar en la DW al aplicar cambios en el esquema fuente. El próximo paso luego de encontradas estas inconsistencias es encontrar la solución mas adecuada.

Podríamos pensar en una primera posible solución para evitar que surjan inconsistencias, impedir al usuario que realice los cambios en el esquema fuente si sabemos que se está en alguna de las situaciones críticas. Esta posibilidad asegura que no se producen inconsistencias en la DW, sin embargo es una solución demasiado restrictiva para el usuario el cual quizás no tenga mas remedio que hacer cambios en la base fuente.

Una segunda alternativa podría ser que el usuario tenga total libertad para hacer los cambios que desee, si se observa que se produce alguna inconsistencia, todas las

Relaciones que queden en ese estado son eliminadas, además de eliminar también de la traza los caminos que ya no se necesitan.

Si bien aplicando ésta última solución el usuario tiene total libertad sobre el esquema fuente, éste podría no ser conciente de los problemas que puedan ocurrir ya que algunos problemas son poco intuitivos y difíciles de ver por lo que aplicar éste enfoque podría ser peligroso por lo que optamos por usar un híbrido de las posibles soluciones mencionadas anteriormente. Si el usuario desea hacer algún cambio, éste tiene toda la libertad de hacerlo pero si el cambio que va a efectuar puede llevar a que ocurra alguna inconsistencia se le avisa al usuario el problema que puede ocurrir y éste decide lo mas conveniente en cada caso. Si de todas maneras decide hacer el cambio, se elimina todo lo que pueda quedar inconsistente, sin embargo en éste caso el usuario fue avisado y él decidió hacer el cambio de todas maneras. De ésta forma el usuario no necesita pensar en los posibles efectos que puedan ocurrir, simplemente en caso que ocurra alguno éste se entera y toma la decisión mas adecuada de acuerdo a la situación.

6- Conclusión

Como se vio anteriormente la evolución de la DW es un tema complejo en el cual pueden surgir efectos no esperados. Las primitivas mas problemáticas resultaron ser Hierarchy Roll Up y New Dimension Crossing. Sin embargo en algunas de las otras primitivas hay efectos a tener en cuenta. La mejor forma de atacar estos problemas es dejar que el usuario tome las decisiones que le parezcan adecuadas pero advirtiéndole en todo momento las consecuencias que sus cambios puedan traer.