Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Tecnologías de base de datos
Recibido: 09/09/2017 | Aceptado: 05/04/2018

Reparación Compatible en Almacenes de Datos Inconsistentes

Compatible Repair in Inconsistent Data Warehouses

Juan-José Ramírez[1*], Raúl Arredondo[2], Cristian Vallejos[2]

[1] Universidad de Los Lagos. Av. Fuchslocher 1305, Osorno, Chile. juan.ramirez@ulagos.cl

[2] Universidad de Los Lagos. Chinquihue km 6, Puerto Montt, Chile. {raul.arredondo, cristian.vallejos}@ulagos.cl

[*] **Autor para correspondencia**: juan.ramirez@ulagos.cl

Resumen

Una dimensión, en un almacén de datos, es un concepto abstracto que agrupa datos que comparten un significado semántico común. Las dimensiones se modelan mediante un esquema jerárquico de categorías. Una dimensión es llamada estricta si cada elemento de cada categoría tiene exactamente un ancestro en cada categoría superior, y homogénea si cada elemento de una categoría tiene por lo menos un ancestro en cada categoría superior. Si una dimensión es estricta y homogénea se pueden utilizar consultas pre-computadas en los niveles inferiores para obtener respuestas en los niveles superiores. Sin embargo, cuando las dimensiones no son estrictas/homogéneas debemos conocer sus restricciones para obtener un resultado correcto. En el mundo real, las dimensiones pueden no satisfacer estas restricciones, y en estos casos, es importante corregir estas dimensiones o encontrar formas de obtener respuestas correctas a las preguntas planteadas en las dimensiones inconsistentes. Una reparación minimal es una nueva dimensión que satisface las restricciones estrictas y homogéneas, y que se obtiene a partir de la dimensión original a través de un número mínimo de cambios, la cual tiene un costo computacional NP-duro. Para mejorar esto, se define la dimensión extendida y se propone la reparación compatible que obtiene una nueva dimensión consistente, manteniendo la esencia de la dimensión original. En este último punto se centra la experimentación, elaborando y evaluando un programa que genere esta reparación compatible y obtenga resultados en tiempo polinomial.

Palabras clave: almacén de datos, experimentos, inconsistencia, restricción estricta, restricción homogénea.

Abstract

A dimension in a data warehouse is an abstract concept that groups data that shares a common semantic meaning. Dimensions are modeled by a hierarchical scheme of categories. A dimension is called strict if each element of each

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

category has a unique ancestor in each higher category, and homogenous if each element of a category has at least one ancestor in each higher category. If a dimension is strict and homogenous, precomputed queries can be used at the lower levels to obtain answers at higher levels. However, when the dimensions are not strict/homogenous their constraints should be known in order to obtain a correct result. In the real world, dimensions may not meet these constraints, and in these cases, it is important to correct these dimensions or find ways to get correct answers to questions posed in inconsistent dimensions. A minimal repair is a new dimension that satisfies the strict and homogenous constraints, and is obtained from the original dimension through a minimum number of changes, which has an NP-hard computational cost. To improve this, the extended dimension is defined and the compatible repair is proposed, which obtains a new consistent dimension, maintaining the essence of the original dimension. The experimentation is focused in this last point, to elaborate and evaluate a program that generates this compatible repair and obtains results in polynomial time.

Keywords: *data warehouse, experiments, homogenous constrains, inconsistency, strict constrains.*

Introducción

Los almacenes de datos integran información de múltiples fuentes, y la almacenan de forma histórica para el análisis de datos y apoyo a la toma de decisiones a nivel estratégico, en empresas, organizaciones o instituciones, aportando a la inteligencia de negocios (Dediz y Stanier, 2016) representándose mediante modelos multidimensionales y se componen de dimensiones y hechos. Las dimensiones se modelan como jerarquías de elementos, entregando el orden jerárquico donde cada elemento pertenece a una categoría (también conocido como niveles) (Chaudhuri y Dayal, 1997) y estas categorías también se organizan a través de los esquemas de jerarquía. Los hechos corresponden a eventos que se asocian generalmente a valores numéricos conocidos como medidas, y hacen referencia a elementos de la dimensión, estos elementos se pueden apreciar en el Ejemplo 1.

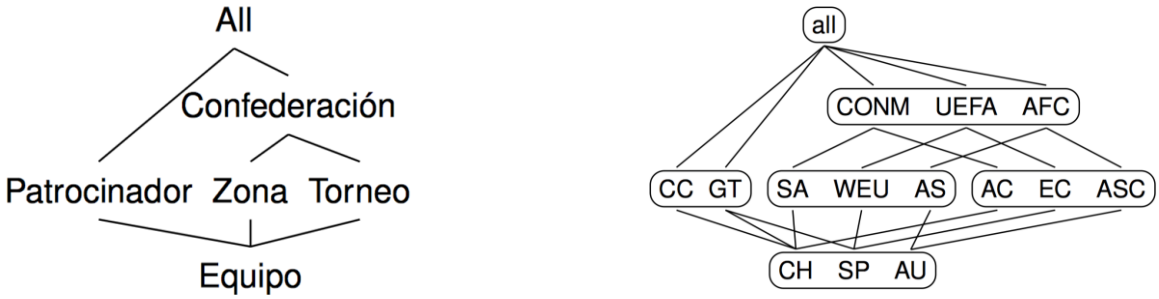
Ejemplo 1 La FIFA tiene un almacén de datos para administrar la información sobre las selecciones nacionales de futbol. Este usa la dimensión Tiempo que está organizada jerárquicamente por las categorías Día, Mes, Año y All, y por la Dimensión Selecciones que se puede ver en la Figura 1a En la Figura 1b se pueden apreciar los elementos pertenecientes a cada categoría de la dimensión Equipos y las relaciones *rollups*[1] entre cada una de ellas. Por ejemplo: CH (selección chilena), SP (selección española), AU (selección australiana) son elementos de la categoría Equipos y

[1] Relación hijo-padre entre los elementos de diferentes categorías unidos mediante el esquema de jerarquía

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

SA (América del sur), WEU (Europa del este), y AS (Asia) son elementos de la categoría Zona. La relación *rollup* entre estas categorías son los pares (CH, SA), (SP, WEU) y (AU, AS)[2]. En la Tabla 1 se muestra la tabla de hechos Ingresos (Equipo, Fecha, Total), la cual almacena las ganancias por equipo en una fecha determinada. La estructura multidimensional permite computar consultas en diferentes niveles de granularidad. Por ejemplo, es fácil computar el total de ingresos agrupados por zona y mes, o el total de ingresos por confederación en un año específico, entre otros.

Las dimensiones son consideradas la parte estática de los almacenes de datos, mientras que los hechos se consideran la parte dinámica, en el sentido de que las operaciones de actualización afectan principalmente a las tablas de hechos (Hurtado et al., 1999). La estructura multidimensional de un almacén de datos permite a los usuarios formular consultas en diferentes niveles de granularidad.



(a) Esquema de jerarquía de la dimensión

(b) Elementos y relaciones *rollups* de la dimensión

Figura 1. Ejemplo de Almacén de datos Selecciones \cite{Ramirez2013}

Tabla 1. Tabla de Hechos Ingresos para la dimensión Selecciones

Ingresos		
Equipo	**Fecha**	**Ingresos**
CH	01-01-2014	$90.000
SP	01-07-2014	$130.000
CH	01-01-2015	$110.000
SP	01-07-2015	$150.000

Las respuestas consistentes a consultas en almacenes de datos se basan en el uso de consultas pre-computadas en categorías inferiores para calcular resultados en los niveles más altos de la jerarquía. En una dimensión consistente, cada medida en la tabla de hechos se debe agregar como máximo una vez y para asegurar esa condición, la dimensión debe cumplir ciertas restricciones de integridad (RI), las cuales se dividen en *estrictas* y homogéneas (Rafanelli y Shoshani, 1990; Lenz y Shoshani, 1997; Hurtado et al., 2005; Mazón et al., 2009) agrupados en el conjunto Σ. Una dimensión es estricta, si todas sus relaciones *rollup* (directas o indirectas en el esquema jerárquico) entre elementos de

[2] La FIFA considera a Australia como parte de Asia.

las categorías son relaciones muchos a uno. La restricción homogénea indica que los *rollups* son obligatorios entre elementos de dos categorías que estén conectados en el esquema jerárquico.

Lo ideal sería que todas las relaciones dentro de una categoría satisfagan estas propiedades para asegurarse un cómputo eficiente al usar consultas pre-computadas. Sin embargo, cuando se modela una determinada dimensión, podría darse el caso de que algunas categorías no cumplan alguna de las RI impuestas en ella, como se aprecia en el Ejemplo 2.

Ejemplo 2 En la Figura 1b se puede apreciar que el *rollup* entre las categorías Equipos y Patrocinador no es estricta, ya que hay elementos en Equipos que tienen más de un Patrocinador como es el caso de *CH*. Así mismo, esta relación tampoco es homogénea ya que *AU* no hace *rollup* a ningún elemento en Patrocinador. Sin embargo, si la dimensión completa es estricta/homogénea dependerá de las RI que se hayan impuesto.

Una dimensión se dice que es consistente respecto a Σ si satisface todas las restricciones definidas en ella, de lo contrario, la dimensión es inconsistente y verificar que una dimensión satisface sus RI puede hacerse en tiempo polinomial (Caniupán et al., 2012).

Una dimensión usualmente se vuelve inconsistente luego de una actualización (Caniupán y Vaisman, 2011) y si se quieren obtener respuestas desde una dimensión inconsistente, es necesario repararla o corregirla. Para repararla se han definido algunas técnicas como la reparación minimal, una de las primeras aproximaciones a este tipo de reparación la cual, realiza cambios de arcos entre las relaciones que no cumplen alguna de las RI impuestas para la dimensión, pero con un mínimo número de cambios respecto a la dimensión original (Bertossi et al., 2009; Caniupán et al., 2012).

Como se menciona en (Caniupán et al., 2012), el problema de este método es que encontrar las reparaciones minimales de una dimensión inconsistente puede ser exponencial, además que la complejidad computacional es NP-duro y decidir si una dimensión *D*' es una reparación minimal con respecto a *D* es co-NP-completo. Sin embargo, se propuso un programa en *Datalog* basado en semántica de modelos estables, que permite obtener las reparaciones minimales de una dimensión, aunque este programa en lógica no es capaz de obtener todas las posibles reparaciones minimales de una dimensión *D* respecto a un conjunto de restricciones Σ.

En el marco de definir una respuesta aproximada, utilizando el conjunto de reparaciones minimales obtenidas de una dimensión inconsistente, en (Bertossi et al., 2009) se propone la reparación canónica que es una nueva dimensión que agrupa en una única dimensión todas las reparaciones minimales obtenidas para aplicar las consultas pre-computadas.

Trabajo Previo

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

Para obtener una dimensión consistente respecto Σ, se han desarrollado dos métodos. El primero se basa en el modelo de dimensión canónica presentado en (Bertossi et al., 2009), el cual, agrupa todas las reparaciones minimales generando una nueva dimensión que contiene conjuntos de elementos en sus categorías y el segundo método es propuesto en (Caniupán et al., 2015) el cual, repara una dimensión que se vuelve inconsistente con respecto a sus restricciones estrictas, después de una actualización en la dimensión y se realiza para dimensiones que tengan un nivel de conflicto (Caniupán y Vaisman, 2011).

En (Ramírez et al. 2013) se define la dimensión extendida, la cual se crea para poder insertar conjuntos de elementos en sus categorías (como se aprecia en la Figura 2b y poder dar respuesta a consultas de agregación sin necesidad de computar las reparaciones minimales y generar la dimensión canónica.

(a) Dimensión inconsistente *D* (b) Reparación compatible *X*

Figura 2. Ejemplo de dimensión inconsistente y su reparación compatible

Con la definición de la dimensión extendida, se desarrolló un método de reparación que se llama reparación compatible, el cual retorna una dimensión que permita responder a consultas de agregación (Bertossi et al., 2009). Dada la dimensión original $D=(H_D, E_D, Elem_D, <_D)$ que es inconsistente respecto a un conjunto de RI Σ, la dimensión $X=(H_X, E_X, CElem_X, <<_X)$ es una reparación compatible de *D* respecto a Σ si se aplica el Algoritmo 1 propuesto en (Ramírez et al., 2013).

El algoritmo de la reparación compatible obtiene todos los elementos que son inconsistentes respecto a Σ para el conjunto de RI homogéneas (línea 1), luego busca evidencias para evaluar el mejor cambio de arcos (línea 6), en caso de haber más de un elemento al que se podría hacer el cambio de arcos, se decide por el primero que se encuentra (línea 7), en caso de no existir evidencia, realiza un cambio de arcos al primer elemento de la categoría padre con el fin de devolver la consistencia a las RI homogéneas, sin embargo, estos últimos casos pueden generar nuevas inconsistencias con respecto a sus restricciones estrictas.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

Posteriormente se evalúan todos los elementos que son inconsistentes respecto a las RI estrictas, para buscar alguna evidencia de hacia qué elemento padre debe hacer *rollup* cada elemento conflictivo (línea 11) y si hay evidencia a un único elemento, se hace el cambio de arco correspondiente. En caso de que no existan evidencias o haya más de una, se genera un elemento conjunto que posea todos los elementos posibles en la categoría padre a los que el elemento conflictivo hace *rollup* para posteriormente hacer el cambio de arcos a ese nuevo elemento (línea 12). Este proceso se repite hasta que no existan elementos inconsistentes (Ramírez et al., 2013) y cada cambio de arcos no debe generar nuevas inconsistencias (Caniupán y Vasiman, 2011). En el Ejemplo 3 se puede apreciar el trabajo del Algoritmo 1 en una dimensión *D* inconsistente.

Ejemplo 3 Considere ahora el esquema de jerarquía de la Figura 1ay la dimensión de la Figura 2a la cual es inconsistente respecto a la RI homogénea Torneo⇒Confederación por el elemento t_5 y también es inconsistente respecto a la RI estricta Equipo→Confederación por los elementos e_2, e_4 y e_6. El algoritmo presentado en (Caniupán et al., 2015) no puede reparar esta dimensión y si se computa esta dimensión en el programa en DLV propuesto en (Caniupán et al., 2012) para obtener las reparaciones minimales se obtienen 17 reparaciones minimales con 7 cambios para cada una (4 inserciones y 3 eliminaciones de arcos).

Si se computa esta misma dimensión en el Algoritmo 1 se obtiene la reparación compatible *X* de la figura 2b, la cual es una dimensión extendida que es consistente respecto a Σ. Esta dimensión se obtiene con 9 cambios de arcos (5 inserciones y 4 eliminaciones de arcos) y al compararlo con el programa en DLV, es más efectivo ya que solo se procesa una sola reparación con 9 cambios y no 17 reparaciones con 7 cambios como el programa en lógica. El Algoritmo 1 genera un elemento conjunto llamado c_4, c_5 insertándolo en la categoría Confederación al cual hace *rollup* el elemento e_4, por medio de sus categorías superiores z_3 y t_4. Estos, al no tener otros elementos que les hagan *rollup*, no genera nuevas inconsistencias por lo que no hay problema en hacer el cambio de arcos.

Complejidad del Algoritmo

Sea *n* es el número máximo de elementos de una categoría, *r* es el número máximo de las relaciones del *rollup* entre dos categorías, m_c es el número de restricciones homogéneas y m_s es el número de restricciones estrictas. El costo de verificar si una restricción homogénea $c_i \Rightarrow c_j$ se viola es $O(n^2)$ ya que se debe que comprobar si hay una relación *rollup* entre cada elemento en c_i y un elemento en c_j. El costo de comprobar si una restricción estricta $c_i \rightarrow c_j$ se viola es $O(r^2)$ ya que la relación $r_{c_i}^{c_j}$ es de tamaño $O(r)$ y se requiere para comparar cada tupla en ella con todas sus tuplas. Por lo tanto, el costo de obtener C, es $O((m_c * n^2)+(m_s * r^2))$. A pesar de que en el peor de los casos *r* es $O(n^2)$, se espera que *r* sea

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

O(n) ya que en general el número de violaciones de una restricción es pequeño, y por lo tanto la mayoría de los elementos en una categoría se resumen en un único elemento de una categoría superior. Dada una restricción *ic* de la forma $c_i \rightarrow c_j$ o $c_i \Rightarrow c_j$, BuscaEvidencia calcula las rutas entre un elemento *e'* que pertenece a la categoría c_a con $c_a \nearrow c_i$, tal que (*e'*,*e*) con $e \in c_i$, y los elementos en el nivel de conflicto de *D*. El número de rutas y el costo de computarla, en el peor de los casos es $O(n^{|C|})$, pero en la mayoría de los casos, en que el número de inconsistencias es bajo, es de *O(|C|)*. Por último, las funciones *Reparar_C* y *Reparar_S* poseen un costo de *O(|C|)*. Por lo tanto, se ha demostrado que computar una dimensión inconsistente respecto a un conjunto Σ de RI por medio de el Algoritmo 1 en el peor de los casos tiene un costo computacional de $O(m_c(r^2+n^{|c|}) + m_s(r^2+n^{|c|}))$ y en el caso esperado tiene un orden de $O(n^2(m_c+m_s))$ (Ramírez et al., 2013).

Algoritmo 1. Reparación Compatible

Entrada: Dimensión *D* y un conjunto de restricciones Σ
Salida: Reparación Compatible X

```
1.  C ← conjunto de tuplas de la forma (ic,inc) donde ic es una
    restricción que se viola en Σ e inc es el conjunto de elementos que
    participan en esa inconsistencia ic.
2.  X ← D
3.  for (ic, inc) ∈ C do
4.     if ic es una restricción homogénea then
5.        for e ∈inc do
6.           Evidencia ← BuscarEvidencia(D,ic,e)
7.           X ← Reparar_C(X, ic, e, Evidencia)
8.  for (ic, inc) ∈ C do
9.     if ic es una restricción estricta then
10.       for e ∈inc do
11.          Evidencia ← BuscarEvidencia(D,ic,e)
12.          X ← Reparar_S(X, ic, e, Evidencia)
13. Return X
```

Experimentos

Para evaluar la factibilidad de obtener una reparación compatible obtenida desde un almacén de datos inconsistente se implementó el Algoritmo 1 en lenguaje C utilizando listas ligadas por punteros. La prueba preliminar de este algoritmo se realizó en la dimensión *D* de la Figura 3a, donde la categoría Equipo está conectado con Zona (Zona Geográfica) y ésta a su vez con Confederación. También, la categoría Equipo está conectado a la categoría Torneo y ésta a su vez con Confederación. La categoría más alta es *All* a la cual llega Confederación.

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

Las restricciones de integridad que debe cumplir la dimensión D_s vienen dadas en la Tabla 2 y como se puede apreciar en la Figura 3a los arcos con líneas continuas representan los elementos inconsistentes respecto a Σ. La dimensión D_s posee 4 elementos que no participan de la violación de ninguna restricción (e_2, e_6, e_9 y e_{10}).

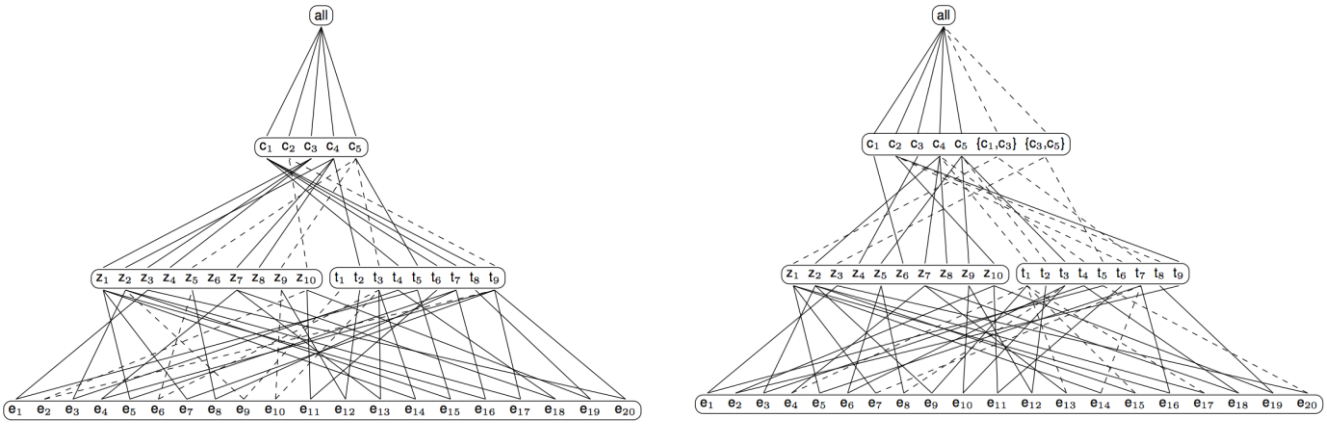
Tabla 2. Conjunto Σ de Restricciones de integridad de la dimensión *Selecciones* D_s

Restricciones Estrictas	**Restricciones Homogéneas**
Equipo→Zona	Equipo⇒Zona
Equipo→Torneo	Equipo⇒Torneo
Equipo→Confederación	Zona⇒Confederación
Zona→Confederación	Torneo⇒Confederación
Torneo→Confederación	

En el caso de la dimensión D_s cuando se viola la RI estricta Equipo→Confederación se provoca que un equipo pertenezca a más de una confederación, esto se puede apreciar en la Tabla 3, por ejemplo, que el equipo e_3 pertenece a la confederación c_1 y c_4.

Tabla 3. *Rollup* entre Equipo y Confederación de la Figura 3[a]

$R_{D_S\ Equipo}^{Confederación}$	
Confederación	**Equipo**
c_1	e_3, e_5, e_8, e_9, e_{11}, e_{16}, e_{18}, e_{19}
c_2	e_2, e_4, e_7, e_{11}, e_{17}, e_{19}, e_{20}
c_3	e_1, e_5, e_7, e_{14}, e_{16}, e_{20}
c_4	e_3, e_4, e_{12}, e_{13}, e_{15}, e_{17}, e_{18}
c_5	e_1, e_6, e_8, e_{10}, e_{12}, e_{13}, e_{14}, e_{15}



Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

(a) Dimensión *Selecciones* D_s Inconsistente

(b) Reparación Compatible X_s obtenida desde la Dimensión D_s Inconsistente y el Algoritmo 1

Figura 3. Caso de estudio de la dimensión inconsistente y su reparación compatible

Si se computa esta dimensión (D_s) en el Algoritmo 1 se obtiene la reparación compatible de la Figura 3b, la cual es una dimensión extendida X_s que es consistente respecto al conjunto Σ presentado en la Tabla 2. El resultado final es la dimensión X_s de la Figura 3b, la cual indica, con las líneas segmentadas, los arcos que sufrieron cambios a diferencia de la dimensión original. Además, en la Tabla 4 se pueden apreciar a qué confederación finalmente pertenece cada equipo en la dimensión X_s.

Los nuevos elementos conjuntos, son elementos que permitirán usarlos para obtener respuestas aproximadas a consultas pre-computadas (Ramírez et al., 2013). Por último, indicar que la reparación compatible X_s (Figura 3b) se obtuvo en un tiempo de 0.193 segundos.

Experimento en Escenario Real

Esta implementación del Algoritmo 1 es una primera versión, para el cual se ha estudiado su comportamiento en una base de datos realista tomando el caso de los Teléfonos de Chile, una dimensión propuesta originalmente en (Bertossi et al., 2009), para computar la dimensión canónica. Dada esta misma dimensión, se estudió el tiempo que demora en generar la reparación compatible en distintos porcentajes de inconsistencia para dimensiones que tienen millones de datos.

Tabla 4. *Rollups* entre Equipo y Confederación de la dimensión extendida X_s

$R_{X_S\ Equipo}^{Confederación}$	
Confederación	**Equipo**
c_1	ϕ
c_2	e_2, e_{11}, e_{19}
c_3	ϕ
c_4	e_3, e_4, e_9, e_{13}, e_{15}, e_{17}, e_{18}
c_5	e_6, e_8, e_{10}, e_{12}
$\{c_1, c_3\}$	e_5, e_7, e_{14}, e_{16}
$\{c_3, c_5\}$	e_1, e_{20}

La dimensión Teléfonos está basada en el país de Chile, el cual está divido políticamente en comunas que en la actualidad ascienden a 346. Cada comuna tiene un cierto número de teléfonos (de acuerdo con su población), los cuales

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

están asociados con un código de área. Ese código de área se relaciona con una única región, lo mismo sucede para una comuna. El esquema jerárquico que modela dicha situación es el que se presenta en la Figura 4. Cabe señalar que se incluye la cantidad de elementos que pertenecen a cada categoría en forma de cardinalidad (# número).

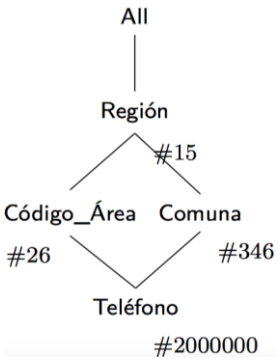


Figura 4. Esquema jerárquico de la dimensión Teléfonos y cantidad de elementos por categoría

Los resultados obtenidos para el código implementado en la dimensión Teléfonos con distintos porcentajes de inconsistencias, son los que se muestran en la Tabla 5, además, para una mejor interpretación, se proporciona el gráfico de Tiempo (Figura 5) el cual muestra el tiempo de ejecución requerido para obtener la reparación compatible, dados distintos niveles de inconsistencia.

Tabla 5. Resultados de la reparación compatible a la dimensión Teléfonos

Elementos inconsistentes en $D_{Teléfonos}$	**% de inconsistencias en $D_{Teléfonos}$**	**Tiempo $X_{Teléfonos}$ (hh:mm:ss)**
500	0,03 %	00:00:15,290
12.120	0,61%	00:04:11,687
31.720	1,59%	00:04:10,345
46.940	2,35%	00:18:04,076
115.420	5,79%	02:05:49,978

En la Tabla 5 se presenta la cantidad de elementos de la categoría inferior (Teléfono) que son inconsistentes respecto a Σ (ver columna 1) luego, el porcentaje de inconsistencia de la dimensión para cada caso (ver columna 2) y, por último, los tiempos de generación de la reparación compatible para los distintos escenarios (ver columna 3).

Analizando el gráfico de la Figura 5 se puede apreciar que en los primeros experimentos el tiempo de ejecución del algoritmo es más o menos estable, realizando cambios en categorías superiores (involucrando pocos cambios de arcos), sin embargo, el último caso es atípico, debido a la cantidad de elementos que participan de la inconsistencia, la gran

Revista Cubana de Ciencias Informáticas
Vol. 12, No. 2, Abril-Junio, 2018
ISSN: 2227-1899 | RNPS: 2301
Pág. 15-29
http://rcci.uci.cu

cantidad de cambios computados para devolver la consistencia a la dimensión y estos fueron realizados entre las categorías inferiores. Basado en los resultados anteriores se puede mencionar que la reparación compatible es obtenida con éxito en todos los casos (eliminando las inconsistencias), con el cual se obtiene una dimensión extendida que es consistente respecto a sus RI.

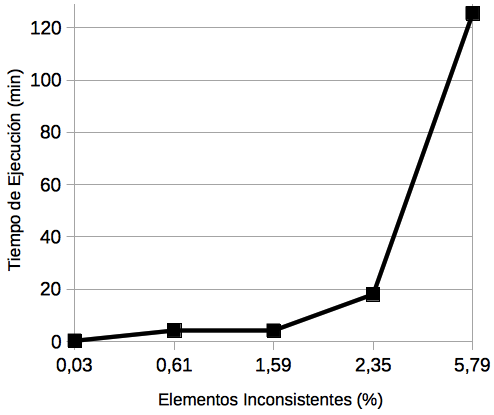


Figura 5. Tiempos de ejecución para computar la reparación compatible en la dimensión Teléfonos

Equipo de pruebas y obtención de datos

El equipo donde se realizaron las pruebas posee un procesador Intel® Core™ i5-2430M CPU @ 2.40GHz x 4 con 7,7Gb de memoria RAM y 10 Gb de memoria de intercambio, a su vez el sistema operativo instalado es un *Debian* Versión 8.2 (Jessie) de 64-bit con Núcleo Linux 3.16.0-4-amd64 y entorno gráfico GNOME 3.14.0. Los softwares con los que se trabajó para obtener los resultados fueron: PostgresSQL versión 9.4.4 para manejo de Bases de Datos, gcc versión 4.9.2-10 como compilador, *Chocolat* 3.3 como editor de texto para programación en lenguaje C, DLV versión x86-64-linux-elf-unixodbc (2012-12-17). Para calcular el tiempo de ejecución del programa se ha utilizado el comando *time* de Linux, este comando permite obtener el tiempo total de ejecución de un programa. Su especificación se puede ver en el manual correspondiente.

Conclusiones y Trabajo Futuro

En (Ramírez et al., 2013) se estudia el problema de las dimensiones inconsistentes y proponen una técnica para la reparación de ellas basadas en el uso de la dimensión extendida, que permite la inserción de elementos conjuntos en una categoría. La incertidumbre y la imprecisión se han analizado en el contexto de modelos multidimensionales en (Pedersen et al., 1999). En particular, en (Burdick et al., 2007) la imprecisión está permitida en las dimensiones, y la incertidumbre en los valores de la tabla de hechos.

La reparación compatible es una dimensión ampliada, obtenida a partir de la dimensión inconsistente que satisface las restricciones estrictas y homogéneas. Esta reparación permite hacer frente a las ambigüedades que surgen de las inconsistencias y es posible obtenerla en tiempo polinomial, no así la reparación canónica propuesta en (Bertossi et al., 2009) ya que requiere obtener el conjunto de reparaciones minimales.

En este artículo se presentan los resultados obtenidos por la implementación en lenguaje C del Algoritmo 1 propuesto en (Ramírez et al., 2013), donde se busca la consistencia de los elementos inconsistentes en base a la evidencia que estos elementos entregan. El programa implementado no altera información que es consistente en la dimensión original, ya que repara en base a evidencias y solo cuando estas no existen se generan elementos conjuntos.

Mencionado esto, se puede decir que una gran parte de la dimensión original está contenida en la reparación compatible, lo cual permite mantener la semántica de estos elementos a diferencia de métodos como las reparaciones minimales (Caniupán et al., 2012) o la misma reparación canónica (Bertossi et al., 2009), las cuales buscan obtener una reparación con el mínimo número de cambios, pero sin intentar mantener la semántica de los datos. En el peor de los casos, si la dimensión es totalmente inconsistente, al principio del proceso se irán generando elementos conjuntos, pero a medida que avance el proceso de reparación se encontrarán evidencias, lo que la respuesta aproximada obtenida de la reparación compatible, como se explica en (Ramírez et al., 2013) seguirá siendo mejor que la respuesta obtenida de la dimensión inconsistente.

La reparación compatible puede ser considerada en la categoría de limpieza de bases de datos, basado en (Bertossi y Bravo, 2013; Fan, 2008; Fan et al., 2008). Así mismo, la reparación compatible podría ser de gran valor para el administrador del almacén de datos ya que al ser consistente puede ser utilizada para realizar consultas de agregación.

El algoritmo implementado, permite obtener la reparación compatible en un número finito de pasos, determinado principalmente por el número de elementos inconsistentes de la dimensión original. Sin embargo, cuando existen millones de elementos pueden surgir miles de evidencias y el algoritmo requiera procesar más opciones, el tiempo para obtener una respuesta es mayor, pero a pesar de este inconveniente, el algoritmo entrega una dimensión consistente y en un tiempo aceptable.

Como trabajo futuro queda mejorar el proceso de generación de la reparación compatible, utilizando técnicas algorítmicas que permitan optimizar el consumo de memoria temporal y mejorar los tiempos de proceso, además de incorporar grados de prioridad a la hora de intentar hacer cambios de arcos para garantizar la semántica de los datos (Arredondo y Muñoz, 2017).

Agradecimientos

Juan-Jose Ramírez y Raúl Arredondo agradecen a la Dirección de Investigación de la Universidad de Los Lagos, a través de los proyectos internos de investigación R18/17 y R12/15, respectivamente. Adicionalmente, los autores agradecen a Mónica Caniupán por insertarlos en el área de investigación en temáticas relevantes en Ciencias de la Computación.

Referencias

ARREDONDO, R. y MUÑOZ, L. Reparación de Data Warehouses con sentido semántico. Revista Cubana de Ciencias Informáticas, 2017, 11: p. 73 – 86.

BERTOSSI, L. y BRAVO, L. Generic and Declarative Approaches to Data Cleaning: Some Recent Developments. En: Handbook of Data Quality – Research and Practice. Sptinger-Verlag Berlin Heidelberg: S Sadiq, 2013.

BERTOSSI, L.; BRAVO, L. y CANIUPÁN, M. Consistent Query Answering in Data Warehouses. En: Proceedings of the III Alberto Mendelzon International Workshop on Foundation of Data Management (AMW), 2009.

BURDICK, D.; DESHPANDE, PM.; JAYRAM, TS.; RAMAKRISHNAN, R. y VAITHYANATHAN, S. OLAP over uncertain and imprecise data. The VLDB Journal, 2007, 16: p. 123 – 144.

CANIUPÁN, M. y VAISMAN, A. Repairing dimension hierarchies under inconsistent reclassification. En: Proceedings of the 30th international conference on Advances in conceptual modeling: recent developments and new directions, ER'11. Berlin, Heidelberg, 2011, p. 75 – 85.

CANIUPÁN, M.; BRAVO, L. y HURTADO, C. Repairing inconsistent dimensions in data warehouses. Data & Knowledge Engineering, 2012, 79-80: p. 17 – 39.

CANIUPÁN, M.; VAISMAN, A. y ARREDONDO, R. Efficient repair of dimension hierarchies under inconsistent reclassification. Data & Knowledge Engineering, 2015, 95: p. 1 – 22.

CHAUDHURI, S. y DAYAL, U. An Overview of Data Warehousing and OLAP Technology. SIGMOD Record, 1997, 26(1): p. 65 – 74.

DEDIŹ, N y STANIER, C. An evaluation of the challenges of multilingualism in data warehouse deve- lopment. En: Proceedings of the 18th International Conference on Enterprise Information Systems, ICEIS 2016. Portugal: SCITEPRESS - Science and Technology Publications, Lda., 2016, p. 196 – 206.

FAN, W. Extending Dependencies with Conditions for Data Cleaning. En: The 8th IEEE International Conference on Computer and Information Technology. Sydney, Australia: IEEE, 2008, p. 185 – 190.

FAN, W.; FAN, F.; JIA, X. y KEMENTSIETSIDIS, A. Conditional Functional Dependencies for Capturing Data Inconsistencies. ACM Transactions on Database Systems, 2008, 33(2): p. 6:1 – 6:48.

HURTADO, C.; MENDELZON, A. y VAISMAN, A. Updating OLAP Dimensions. En: Proceedings of the 2Nd ACM International Workshop on Data Warehousing and OLAP. New York, USA: ACM, 1999, p. 60 – 66.

HURTADO, C.; GUTIERREZ, C. y MENDELZON, A. Capturing Summarizability with Integrity Constraints in OLAP. ACM Transactions on Database Systems, 2005, 30(3): p. 854 – 886.

LENZ, HJ. y SHOSHANI, A. Summarizability in OLAP and Statistical Data Bases. En: Proceedings of the Ninth International Conference on Scientific and Statistical Database Management. SSDBM '97. Washington, DC, USA: IEEE Computer Society, 1997, p. 132 – 143.

MAZÓN, JM.; LECHTENBÖRGER, J. y TRUJILLO, J. A Survey on Summarizability Issues in Multidimensional Modeling. Data & Knowledge Engineering, 2009, 68(12): p. 1452 – 1469.

PEDERSEN, TB.; JENSEN, C. y DYRESON, C. Supporting Imprecision in Multidimensional Databases Using Granularities. En: Proceedings of the 11th International Conference on Scientific and Statistical Database Management. SSDBM'99. Washington, DC, USA: IEEE Computer Society, 1999, p. 90.

RAFANELLI, M. y SHOSHANI, A. STORM: A Statistical Object Representation Model. En: Proceedings of the Fifth International Conference on Statistical and Scientific Database Management. SSDBM V. New York, NY, USA: Springer-Verlag New York, Inc., 1990, p. 14 – 29.

RAMÍREZ, J.; BRAVO, L. y CANIUPÁN, M. Extended Dimensions for Cleaning and Querying Inconsistent Data Warehouses. En: Proceedings of the Sixteenth International Workshop on Data Warehousing and OLAP. DOLAP '13. New York, NY, USA: ACM, 2013, p. 39 – 46.