SURVEYS

# Sistemas de intercambio de información manejo de metadatos

Gentileza Javier Velasco

*Jorge Pérez*
*Estudiante de Doctorado en Ciencia de la Computación, Pontificia Universidad Católica de Chile. Ingeniero Civil en Computación y Magíster en Ciencia de la Computación, de la misma Universidad.*
*jperez@ing.puc.cl*

Muchas de las tareas actuales en el área de manejo de información suponen no sólo crear, modificar o consultar datos, sino también establecer correspondencias entre fuentes de datos. A pesar de que las correspondencias no son consideradas datos al nivel de los almacenados en cada fuente, sí son consideradas metadatos, es decir, información acerca de cómo se relacionan los datos que se encuentran almacenados independientemente. Estos son un *input* crucial para tareas como intercambio, migración o integración de datos.

La creación de metadatos para relacionar fuentes independientes supone un trabajo considerable por parte de un experto (ingeniero, administrador de sistema, etc.), que debe conocer la semántica de cada componente de las fuentes que participan. Los metadatos resumen entonces parte del conocimiento del experto y podrían, en un principio, ser reutilizados en tareas incluso más allá de las que involucran las fuentes de datos para las que fueron inicialmente creados.

Un ejemplo conceptualmente simple de manipulación y reutilización de metadatos es la composición [9,6]. Considere tres fuentes de datos A, B y C y suponga que se cuenta con reglas de correspondencia entre A y B, llamémoslas R(A,B), y reglas de correspondencia R(B,C) entre B y C. Asuma ahora que una nueva aplicación necesita trasladar datos desde A hacia C. Crear un conjunto de metadatos de correspondencia entre A y C desde cero podría implicar un esfuerzo considerable. Quisiéramos entonces reutilizar los metadatos en R(A,B) y R(B,C); quisiéramos usar R(A,B) y R(B,C) para construir las correspondencias entre A y C de manera automática. Este proceso debiera resultar de una operación

abstracta de composición.De inmediato surgen muchas preguntas técnicas. Dado que las correspondencias entre esquemas generalmente se especifican en lenguajes lógicos (los veremos más adelante) ¿qué significa exactamente componer este tipo de especificaciones? Una vez que tenemos claro el significado, otras preguntas incluyen determinar si existe un algoritmo para calcular tal composición o qué tipo de especificación se necesita para expresar la composición. Además de ésta se han identificado operaciones como inversión, mezcla y diferencia [4,9]. Y para todas ellas surgen las mismas interrogantes.

En este artículo daremos una visión breve e introductoria al manejo de metadatos en el contexto de sistemas de intercambio de información, principalmente al de operaciones algebraicas sobre correspondencias entre esquemas, centrándonos en la composición y la inversa. Cabe destacar que este tema es relativamente nuevo en la comunidad de base de datos, por lo que los principales avances han estado en el lado más fundamental del área. En este artículo nos enfocaremos en los aspectos teóricos del tema desarrollados en los últimos cinco años.

## METADATOS Y CORRESPONDENCIAS ENTRE ESQUEMAS

La Figura 1 muestra un ejemplo simple de correspondencia entre los datos de dos tablas CliA y CliB y una tercera independiente CliAB. En dicha figura las correspondencias están plasmadas en la forma de flechas entre los esquemas de las bases de datos, en particular, entre los campos (o atributos) de las tablas. Las flechas indican que, por ejemplo, el campo name en la tabla CliA corresponde al campo client en la tabla CliAB y el campo amount en la tabla CliB corresponde al atributo balance en la tabla CliAB. La figura muestra también cómo estas correspondencias pueden ser usadas para migrar datos desde las tablas CliA y CliB hacia la tabla CliAB. Note cómo en la migración hay datos que no quedan completamente determinados por las correspondencias (por ejemplo, el campo account en la primera fila de la tabla CliAB).

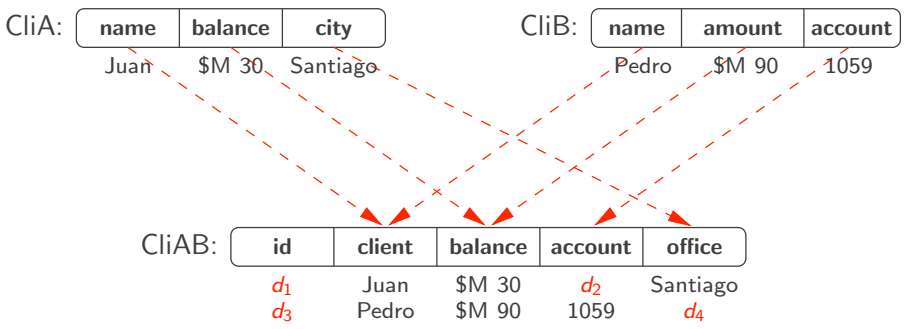
En la práctica estas correspondencias pueden estar especificadas con relaciones complejas entre los esquemas de las bases de datos. En la literatura ha sido típico el uso de lenguajes lógicos para la especificación formal de estas relaciones. Por ejemplo, la relación entre CliA y CliAB puede especificarse usando la siguiente implicación en lógica de primer orden:

$$\forall x \forall y \forall z \left(\mathrm{CliA}(x,y,z) \rightarrow \exists U \exists V \ \mathrm{CliAB}(U,x,y,V,z)\right) \quad (1)$$

La fórmula está especificando de manera formal que si existe una tupla *(x, y, z)* en la tabla CliA, entonces deben existir valores *U* y *V* tales que *(U, x, y, V, z)* es una tupla en la tabla CliAB. Note que la fórmula no especifica completamente los valores de las primera y cuarta componentes en CliAB, sólo fuerza a que estos valores existan. Entonces, si se están migrando datos, es tarea de la aplicación de migración el decidir qué valor debe agregar cuando las componentes no están totalmente especificadas. Esta cuantificación existencial, y la subsiguiente indeterminación, es uno de los puntos que hace al intercambio de datos un problema técnicamente desafiante [5].

**Figura 1**

Correspondencias simples entre esquemas de bases de datos




## COMPOSICIÓN DE CORRESPONDENCIAS ENTRE ESQUEMAS

La composición fue identificada como una de las primeras operaciones fundamentales a nivel de metadatos de correspondencias entre esquemas [9,6]. Dadas correspondencias R(A,B) entre las fuentes A y B, y R(B,C) entre las fuentes B y C, su composición debiera ser un conjunto de correspondencias R(A,C) entre A y C que sea semánticamente consistente con los metadatos en R(A,B) y R(B,C).

La semántica de la composición de correspondencias está basada en el uso de ellas para el intercambio de datos. Básicamente R(A,C) es considerada la composición de R(A,B) y R(B,C) si el resultado de migrar datos entre A y C usando R(A,C) es exactamente el mismo que el de migrar datos entre A y B usando R(A,B), seguido de migrar los datos entre B y C usando R(B,C)[1].

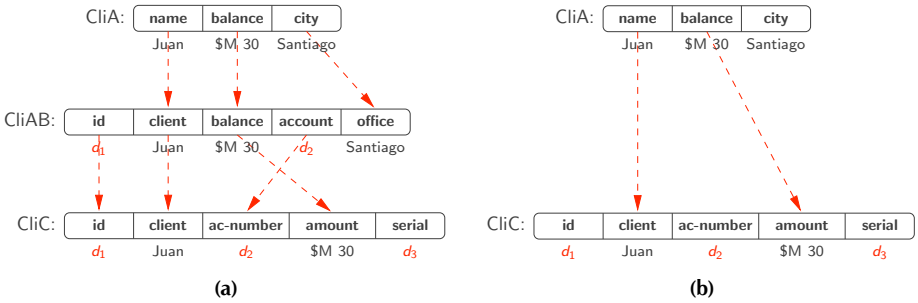
Hay casos en que la composición resulta muy simple. Por ejemplo, suponga que la tabla CliAB en la Figura 1 se relaciona con otra tabla CliC como se muestra en la Figura 2(a). En la Figura también se muestran las correspondencias antiguas entre CliA y CliAB y el resultado de un intercambio de datos usando ambos conjuntos de correspondencias. Las correspondencias entre CliAB y CliC pueden expresarse con la siguiente fórmula:

$$\forall x \forall y \forall u \forall v \left(\exists Z \ \mathrm{CliAB}(u,x,y,v,Z) \rightarrow \exists W \ \mathrm{CliC}(u,x,v,y,W)\right) \quad (2)$$

La Figura 2(b) muestra el resultado de la composición de ambas correspondencias. En este caso la composición es muy simple; sólo debemos seguir las flechas desde CliA

1 La definición formal está basada en la noción algebraica de composición de relaciones binarias. El lector interesado puede ver el trabajo de Fagin et al. [6] al respecto.

**Figura 2 (a) y (b)**
Resultado de la composición de correspondencias simples




hasta CliC pasando por CliAB. La misma figura muestra también como una migración de datos usando las nuevas correspondencias entre CliA y CliC da exactamente el mismo resultado que en la Figura 2(a). Al nivel de fórmulas, las correspondencias resultantes pueden expresarse como:

$$\forall x \forall y \, (\exists Z \, \mathrm{CliA}(x,y,Z) \rightarrow \exists U \exists V \exists W \, \mathrm{CliC}(U,x,V,y,W)) \quad (3)$$

Note que la fórmula (3) resulta de reemplazar el lado derecho de la implicación de la fórmula (1), por el lado derecho de la fórmula (2), agregando cuantificación existencial sobre las variables que no se mencionan en ambos lados de la fórmula resultante (*Z,U,V y W* en este caso).

Fagin et al. [6] fueron los primeros en mostrar que la composición de correspondencias es, en general, un problema técnicamente desafiante y no siempre tan simple como en el anterior ejemplo. En particular, fórmulas que utilizan joins entre tablas más cuantificación existencial, complican sustancialmente el proceso. Considere el ejemplo de correspondencias entre tres esquemas de fuentes de datos para alumnos y cursos que se muestra en la Figura 3. El esquema intermedio tiene dos tablas, Student y Assgn. Las correspondencias entre el primer y segundo esquema son simples y pueden expresarse con las siguientes fórmulas:

$$\forall x \, (\exists Y \, \mathrm{Takes}(x,Y) \rightarrow \exists Z \, \mathrm{Student}(Z,x))$$

$$\forall x \forall y \, (\mathrm{Takes}(x,y) \rightarrow \mathrm{Assgn}(x,y))$$

La correspondencia entre el segundo y tercer esquema es un poco más complicada ya que usan un join entre las tablas Student y Assgn para establecer la relación con la tabla Enrollment. Esto puede expresarse con la fórmula:

$$\forall y \forall z \, (\exists X \, \mathrm{Student}(z,X) \, \& \, \mathrm{Assgn}(X,y) \rightarrow \mathrm{Enrollment}(z,y))$$

La Figura 4(a) muestra el resultado de hacer la composición simplemente siguiendo las flechas en la Figura 3. Estas correspondencias pueden expresarse con la fórmula:

$$\forall y \, (\exists X \, \mathrm{Takes}(X,y) \rightarrow \exists Z \, \mathrm{Enrollment}(Z,y)) \quad (4)$$

No es difícil notar que la fórmula (4) no captura exactamente la composición. El problema principal está en los valores que se obtienen en el campo s-id en la tabla Enrollment al intercambiar datos con estas correspondencias. Note que en la Figura 3, a pesar de que no tenemos seguridad del valor exacto del dato en el campo s-id, sabemos que en ambas tuplas este valor debe ser el mismo ($d_1$ en la Figura 3). Sin embargo, cuando se intercambian datos usando la fórmula (4) nada fuerza a que los valores en el campo s-id de la tabla Enrollment coincidan ($d_1$ y $d_2$ en la Figura 4(a)). Para lidiar con este problema, Fagin et al. [6] introducen el uso de funciones al especificar correspondencias entre esquemas. La Figura 4(b) muestra un conjunto de correspondencias que captura la composición. Note que en el campo s-id se usa el dato *f*(Juan) para modelar un dato que existe aunque no sepamos exactamente su valor. De esta forma, el dato puede repetirse en ambas tuplas a pesar de que su valor no esté completamente especificado. Fagin et al. [6] usan la siguiente fórmula (en lógica de segundo orden) para expresar la composición:

$$\exists f \, (\forall x \forall y \, (\mathrm{Takes}(x,y) \rightarrow \mathrm{Enrollment}(f(x),y)))$$

Fagin et al. [6] no sólo introducen el uso de funciones para especificar correspondencias, sino que muestran que estas son imprescindibles para expresar la composición. Entre otros resultados, entregan un algoritmo para computar de manera automática la composición y estudian la complejidad computacional de los problemas asociados a la composición de correspondencias.

**Figura 3**
Correspondencias entre tres esquemas con información de alumnos

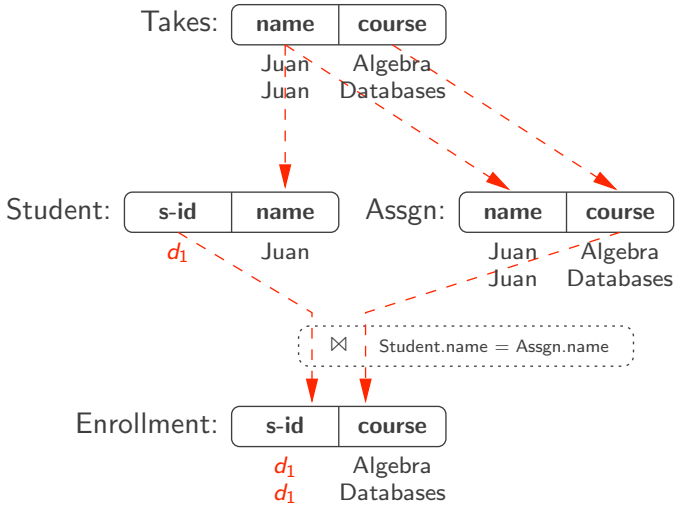
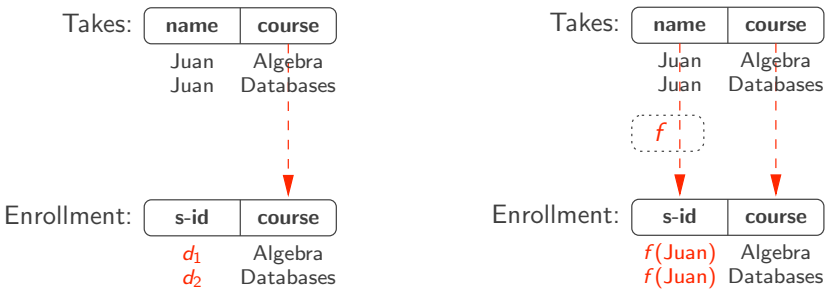


**Figura 4**

Dos posibles composiciones para las correspondencias en la Figura 3




## INVIRTIENDO CORRESPONDENCIAS ENTRE ESQUEMAS

Una operación sobre correspondencias entre esquemas que ha recibido considerable atención es la inversa [7,8,1,2]. Si R(A,B) es un conjunto de correspondencias para intercambiar datos entre A y B, su inversa debiera ser un conjunto de correspondencias que, intuitivamente, especifique como migrar los datos de regreso desde B hacia A.

Invertir correspondencias entre esquemas resultó ser un problema sumamente desafiante. Incluso establecer la semántica formal ha probado ser un problema no trivial.

**Figura 5**

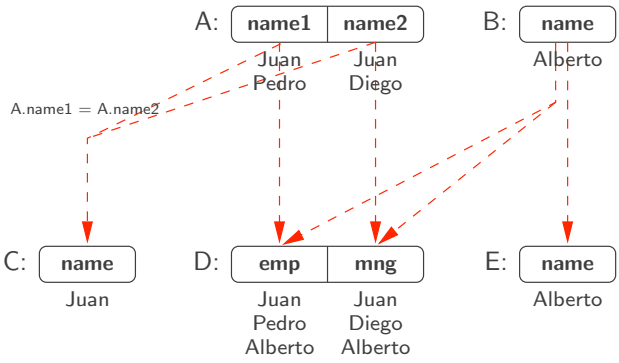
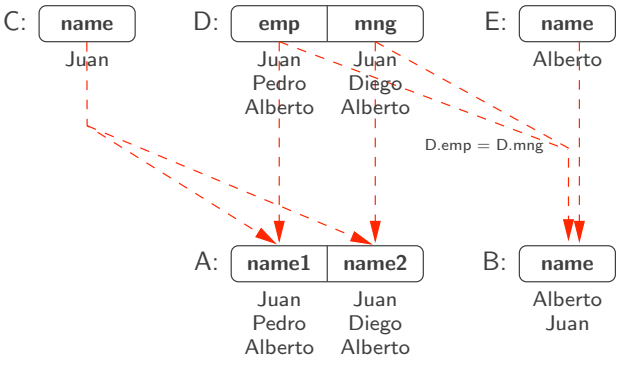



**Figura 6**

Resultado de invertir las flechas de la Figura 5




Problemas como cuándo un conjunto de correspondencias es invertible, qué lenguaje es necesario para expresar la inversa y diseñar algoritmos para computarla, han sido el tema principal de diversos trabajos [7, 8, 1].

La primera solución que uno podría entregar como inversa de un conjunto de correspondencias es simplemente invertir el sentido de las flechas que especifican las correspondencias. Sin embargo esta solución no entrega un resultado de acuerdo a la intuición de la inversa. A modo de ejemplo, considere las correspondencias entre esquemas en la Figura 5.

Estas correspondencias pueden expresarse con el conjunto de fórmulas:

$$\forall x \forall y\,(A(x,y) \rightarrow D(x,y)) \qquad \forall x\,(A(x,x) \rightarrow C(x))$$
$$\forall x\,(B(x) \rightarrow D(x,x)) \qquad \forall x\,(B(x) \rightarrow E(x))$$

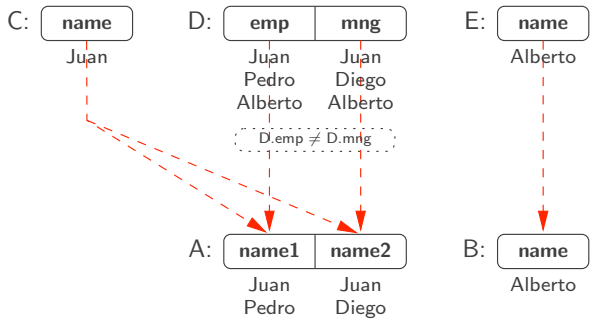
Invertir el sentido de las flechas en la Figura 5 implica simplemente invertir la dirección de la implicación en cada una de las anteriores fórmulas. La Figura 6 muestra un esquema de las correspondencias resultantes.

Esta Figura muestra claramente por qué invertir el sentido de las flechas no es una buena solución para la inversa de un conjunto de correspondencias. En la Figura 5 podemos ver que la tabla A inicialmente tiene las tuplas (Juan, Juan) y (Pedro, Diego) mientras que la tabla B tiene la tupla (Alberto) como único dato. En la Figura 6 se están migrando los datos de vuelta usando las correspondencias que resultan de invertir el sentido de las flechas. Note como tanto en la tabla A como en la B aparecen datos que no existían inicialmente, a saber, la tupla (Alberto, Alberto) en la tabla A y la tupla (Juan) en la tabla B. Esto ocurre porque al hacer el intercambio inicial (Figura 5), ambas tablas A y B aportan con datos para la D. En particular, si en la tabla D aparece una tupla con valores repetidos, como (Juan, Juan), no tenemos seguridad de si esta tupla vino de la tabla A o de la B.

La Figura 7 muestra una mejor solución para la inversa de las correspondencias en la Figura 5. De hecho, note que el intercambio de datos de vuelta hacia las tablas A y B genera exactamente los datos iniciales. Note cómo sólo las tuplas de la forma *(a,b)* con

**Figura 7**

Inversa de las correspondencias en la Figura 5




$a \neq b$ se traspasan desde la tabla D a la tabla A. Los otros datos de A se obtienen desde la tabla C. Similarmente los datos de B se obtienen desde la tabla E.

Las correspondencias de la Figura 7 puede expresarse con las fórmulas:

$$\forall x \forall y \, (D(x,y) \,\&\, x \neq y \rightarrow A(x,y))$$

$$\forall x \, (C(x) \rightarrow A(x,x))$$

$$\forall x \, (E(x) \rightarrow B(x))$$

Fagin [7] y Fagin et al. [8] estudiaron el problema de invertir correspondencias dando una semántica formal. Diseñaron un algoritmo para computar la inversa e identificaron el lenguaje necesario para expresar inversas. En particular, demostraron que el uso de desigualdades ($\neq$) era imprescindible para expresar inversas.

En la práctica, cuando se intercambian datos usando correspondencias desde una fuente A hacia una fuente B, muchos de los datos de A no son efectivamente migrados. Por ejemplo, en las correspondencias en la Figura 2(b), el dato "Santiago" simplemente se pierde luego de la migración. Conjuntos de correspondencias que pierden datos en una migración son inmediatamente no invertibles según la noción propuesta por Fagin [7]. Esto hace que casi cualquier ejemplo práctico de correspondencias sea no invertible. Con esta motivación Arenas et al. [1] proponen la idea de recuperar el máximo de información posible como semántica para la inversa de un conjunto de correspondencias. Esta nueva noción resulta ser una generalización de la noción de Fagin y con mejores propiedades en cuanto a los casos que puede cubrir. Arenas et al. [1] también estudiaron problemas de expresividad y complejidad y diseñaron un algoritmo para computar esta nueva noción de inversa. En particular, para el caso de correspondencias que no usan cuantificación existencial, el algoritmo de Arenas et al. [1] funciona en tiempo cuadrático, lo que es una mejora sustancial con respecto a los algoritmos propuestos por Fagin et al. [7,8] que funcionan en tiempo exponencial. En un subsiguiente trabajo, Arenas et al. [2] proponen refinamientos para las nociones de inversa con buenas propiedades con respecto a la expresividad necesaria para representar las inversas.

## CONCLUSIONES

El manejo de metadatos de correspondencias entre esquemas, en particular poder operar algebraicamente estas correspondencias, es un problema fundamental y de alto interés tanto teórico como práctico. Si bien los principales avances han estado en el lado formal, grandes compañías como Microsoft e IBM están invirtiendo recursos para diseñar herramientas que puedan implementar algunas de estas ideas, dando una solución de alto nivel a los problemas de intercambio e integración de datos [10,11]. Un estudio de las operaciones necesarias para dar una solución integral al manejo de metadatos se puede encontrar en los trabajos de Bernstein [4] y Melnik [9].

Muchos problemas interesantes relacionados con el tema de operaciones sobre correspondencias permanecen abiertos y poco explorados. En particular, preguntas usuales que aparecen en cualquier álgebra están en un estado muy inicial de investigación. Por ejemplo, la interacción entre las operaciones de composición e inverso no es para nada clara. La existencia de sistemas de representación de correspondencias (lenguajes lógicos), que sean cerrados bajo estas operaciones, es otro problema fundamental que permanece abierto. Recientemente Arenas et al. [3] han propuesto un marco teórico general para comparar correspondencias entre esquemas, que puede ser el punto de partida para estudiar algunos de los problemas fundamentales del área. BITS

## REFERENCIAS

[1] M. Arenas, J. Pérez, and C. Riveros. The Recovery of a Schema Mapping: Bringing Exchanged Data Back. ACM Transactions on Database Systems, 34(4), 2009.
[2] M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Inverting Schema Mappings: Bridging the Gap between Theory and Practice. In VLDB, pages 1018–1029, 2009.
[3] M. Arenas, J. Pérez, J. L. Reutter and C. Riveros. Foundations of Schema Mapping Management. To appear in PODS 2010.
[4] P. A. Bernstein. Applying Model Management to Classical Metadata Problems. In CIDR, 2003.
[5] R. Fagin, P. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. Theoretical Computer Science 336(1):89–124, 2005.
[6] R. Fagin, P. Kolaitis, L. Popa, and W.-C. Tan. Composing Schema Mappings: Second-Order Dependencies to the Rescue. ACM Transactions on Database Systems, 30(4):994–1055, 2005.
[7] R. Fagin. Inverting Schema Mappings. ACM Transactions on Database Systems, 32(4), 2007.
[8] R. Fagin, P. Kolaitis, L. Popa, and W.-C. Tan. Quasi-Inverses of Schema Mappings. ACM Transactions on Database Systems, 33(2), 2008.
[9] S. Melnik. Generic Model Management: Concepts and Algorithms. Volume 2967 of LNCS, Springer, 2004.
[10] MS ADO.NET Entity Framework, http://msdn.microsoft.com/en-us/library/bb399572.aspx
[11] IBM Clio System, http://www.almaden.ibm.com/cs/projects/criollo/