Integración de Bases de Conocimiento

Clase 5: Data Exchange

Profesores: Maria Vanina Martinez y Ricardo Rodriguez

Esquema

En esta primera parte veremos:

- Mapeos de esquema como modelo para la formalización y estudio de tareas de *interoperabilidad de datos*.
- Intercambio de datos y *soluciones*: soluciones universales y el "*core*" (núcleo).
- Respuesta a *consultas* en intercambio de datos.
- Incorporación de *ontologías*.

El desafío

- Hay aplicaciones en las cuales los datos residen en:
  - diferentes sitios *físicos*
  - diferentes *formatos* (tablas relacionales, XML, etc.).
- Las aplicaciones que manejan estos datos necesitan *acceder* y *procesarlos*.
- El *mercado* de las herramientas de integración de información a nivel empresa (*enterprise*) es muy grande:
  - A finales de 2014 (últimos números disponibles), el mercado movió USD 2.400 millones;
  - crecimiento del 6,9% con respecto a 2013;
  - Gartner proyecta USD 3.400 millones para 2019.

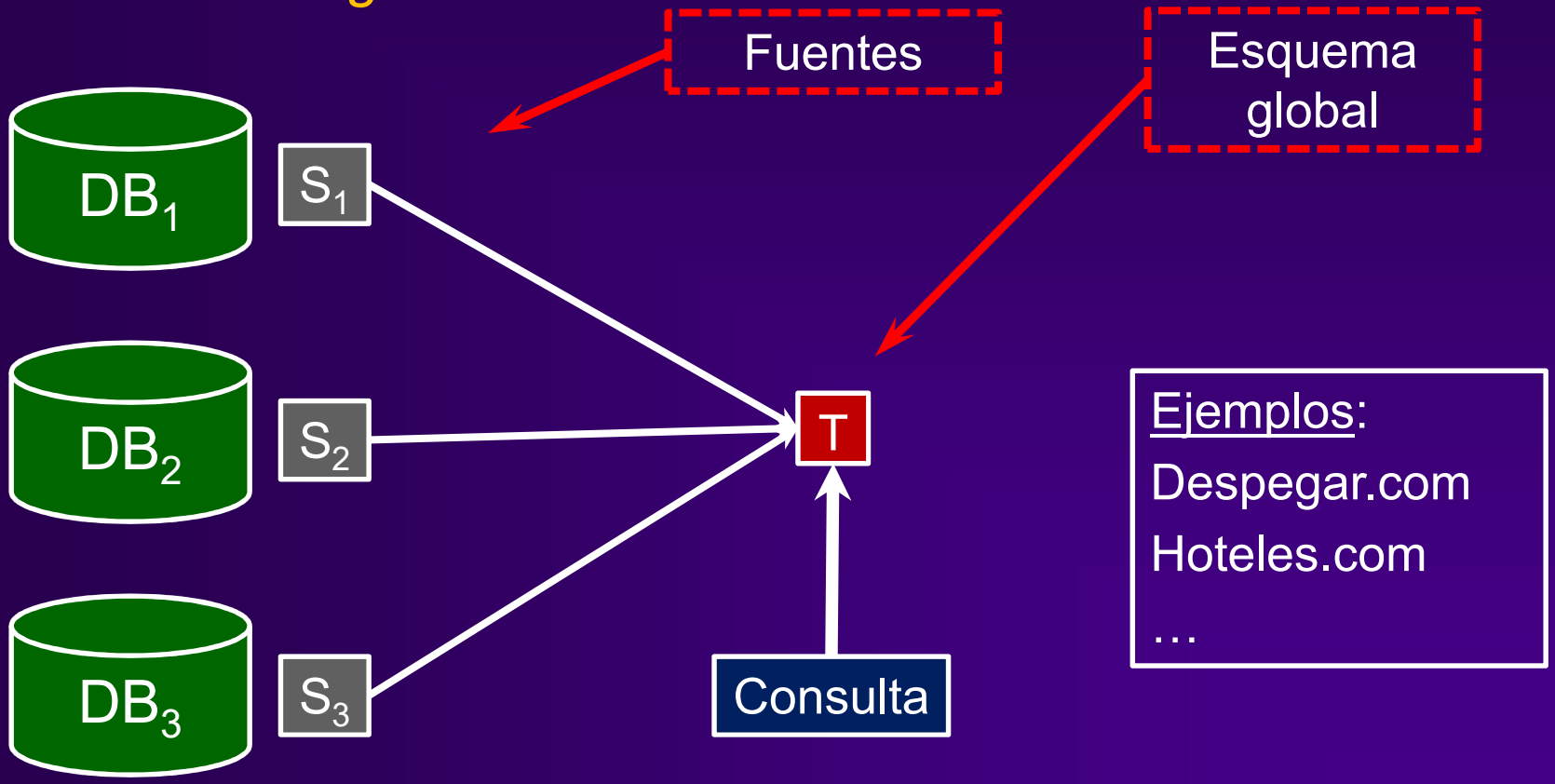
Reporte de Gartner (julio 2015)

Dos aspectos diferentes

- La comunidad de investigación ha estudiado dos facetas diferentes pero estrechamente relacionadas de la integración de información:
  - *Integración* de datos, también llamada Federación de datos ("*Data Integration*" y "*Data Federation*", en inglés)
  - *Intercambio* de datos, también llamada Traducción de datos ("*Data Exchange*" y "*Data Translation*", en inglés)
- Veamos cómo se caracteriza cada una...

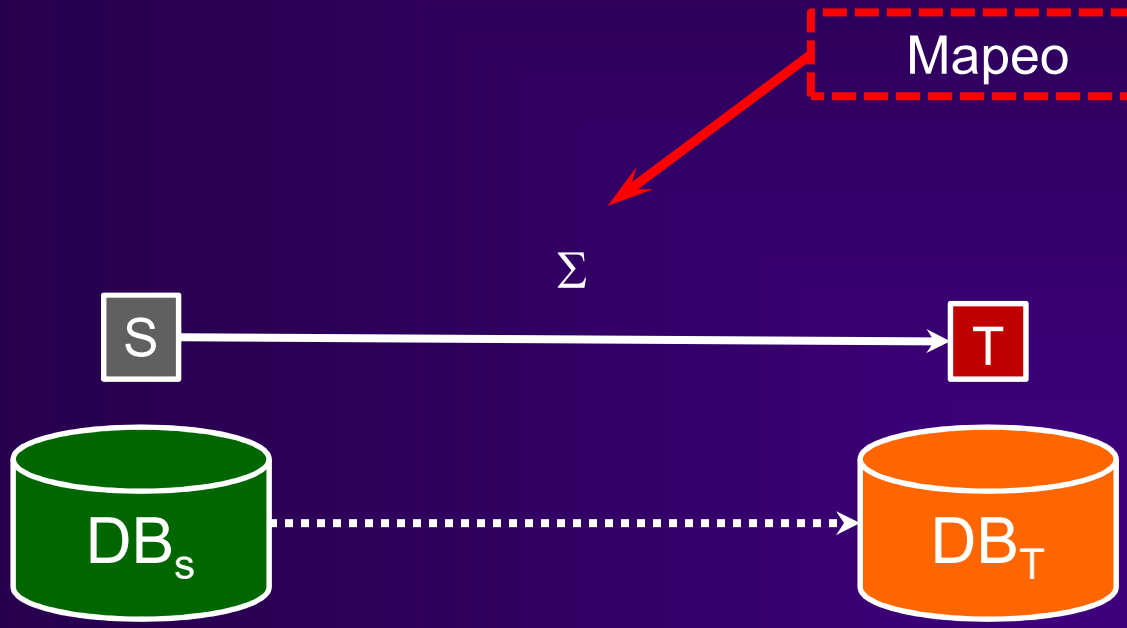
Integración de datos

Se consultan datos heterogéneos de diferentes fuentes vía un *esquema virtual global*:



Intercambio de datos

Se *transforman* datos estructurados bajo un esquema *origen* en datos estructurados bajo un esquema *destino* diferente:



Ejemplos:
Fusiones,
Adquisiciones,
…

Mapeos de esquema

Los *mapeos de esquema* son las piezas fundamentales para la formalización y estudio de ambos casos:

- Aserciones declarativas de alto nivel que especifican la *relación* entre dos esquemas de BD.
- Hacen que sea posible la separación del *diseño* de la relación entre esquemas y su *implementación*:
  - Son fáciles de generar y manejar (semi-)automáticamente
  - Pueden ser compilados en scripts SQL/XSLT

Mapeos de esquema

Mapeo $M = (S,\ T,\ \Sigma)$

- Esquema *origen* S, esquema *destino* T
- Mapeo Σ

¿Qué constituye un "buen" lenguaje de especificación de mapeos de esquema?

Lenguaje de mapeo de esquemas

Primera idea:

Usamos un lenguaje basado en lógica para especificar los mapeos de esquema. Por ejemplo, lógica de primer orden (FOL).

Advertencia:

El uso irrestricto de FOL como lenguaje de especificación da lugar a la *indecidibilidad* de los problemas básicos.

Lenguaje de mapeo de esquemas

Todo lenguaje de especificación de mapeos debería tener:

- *Copiado* (*Nicknaming*): Copiar cada tabla origen a una tabla destino, y renombrarla.
- *Proyección* (Borrado de columna): Formar una tabla destino borrando una o más columnas de una tabla origen.
- *Incorporación de columnas*: Formar una tabla destino sumando una o más columnas a una tabla origen.
- *Descomposición*: Descomponer una tabla origen en una o más tablas destino.
- *Join*: Formar una tabla destino haciendo join entre dos o más tablas origen.
- *Combinaciones* de estas operaciones.

Lenguaje de mapeo de esquemas

Veamos cómo éstas se pueden representar en FOL:

- Copiado (*Nicknaming*): $\forall X_1,\ldots,X_n \left(P(X_1,\ldots,X_n) \rightarrow R(X_1,\ldots,X_n)\right)$
- Proyección (Borrado de columna): $\forall X, Y,Z \left(P(X, Y,Z) \rightarrow R(X, Y)\right)$
- Agregado de columnas: $\forall X, Y \left(P(X, Y) \rightarrow \exists Z\, R(X, Y,Z)\right)$
- Descomposición: $\forall X, Y,Z \left(P(X, Y,Z) \rightarrow R(X, Y) \wedge T(Y,Z)\right)$
- Join: $\forall X, Y,Z \left(E(X,Z) \wedge F(Z, Y) \rightarrow R(X,Z, Y)\right)$
- Combinaciones de estas operaciones, por ejemplo: join + agregado + descomposición:

$$\forall X, Y,Z \left(E(X,Z) \wedge F(Z, Y) \rightarrow \exists W \left(R(X, Y) \wedge T(X, Y,Z, W)\right)\right)$$

TGDs

- Interesantemente, *todas* estas operaciones pueden ser especificadas mediante dependencias generadoras de tuplas (*tuple-generating dependencies*, TGDs).
- Informalmente, las TGDs son restricciones que se refieren a la necesidad de que *existan tuplas* cuando se cumple una condición de "disparo".
- Emergieron en la Teoría de Bases de Datos como una clase importante de restricciones dado su *balance* entre poder *expresivo* y propiedades *algorítmicas*.

Repaso de TGDs

- Su forma general es:

$$\forall \mathbf{X}\ \phi(\mathbf{X}) \rightarrow \exists \mathbf{Y}\ \psi(\mathbf{X}, \mathbf{Y})$$

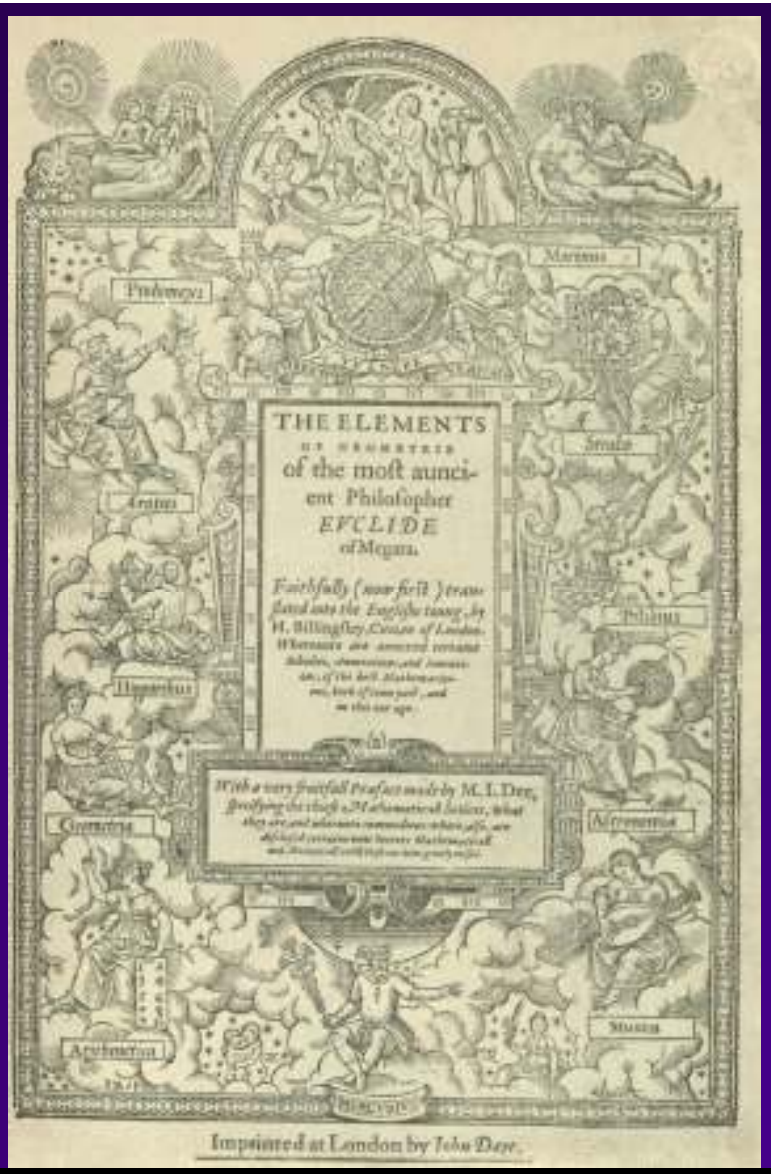
  donde $\mathbf{X}$ e $\mathbf{Y}$ son vectores de variables, y ϕ, ψ son *conjunciones* de fórmulas *atómicas*.

- Algunas subclases conocidas son:
  - Dependencias de *inclusión* (como las *claves foráneas*)
  - Dependencias *multivaluadas*

TGDs: Poder expresivo

J. Avigad, E. Dean y J. Mumma:
"*A Formal System for Euclid's Elements*".
The Review of Symbolic Logic vol. 2(4),
pp. 700–768, 2009.

Afirma que *todos los teoremas* en la colección "*Los Elementos de Euclides*" pueden ser *expresados mediante TGDs*.



s-t-TGDs

Para nuestros propósitos, alcanza con una subclase llamada *source-to-target TGDs* (s-t-TGDs), con forma:

$$\forall \mathbf{X}\ \phi(\mathbf{X}) \rightarrow \exists \mathbf{Y}\ \psi(\mathbf{X}, \mathbf{Y})$$

donde ϕ es una conjunción de átomos sobre el esquema *origen* y ψ es una conjunción de átomos sobre el *destino*.

Ejemplos:

$$\forall E\, \forall C\ estudiante(E) \wedge anotado(E,C) \rightarrow \exists N\ nota(E,C,N)$$

$$\forall E\, \forall C\ estudiante(E) \wedge anotado(E,C) \rightarrow$$

$$\exists N\, \exists P\ profesor(P,C) \wedge nota(E,C,N)$$

s-t-TGDs

- Las s-t-TGDs *generalizan* a dos tipos de restricciones:
  - *Local-as-view* (LAV): $\forall \mathbf{X}\ p(\mathbf{X}) \rightarrow \exists \mathbf{Y}\ \psi(\mathbf{X}, \mathbf{Y})$

    donde p es una relación del esquema *origen*.
  - *Global-as-view* (GAV): $\forall \mathbf{X}\ \phi(\mathbf{X}) \rightarrow \exists \mathbf{Y}\ r(\mathbf{X}, \mathbf{Y})$

    donde r es una relación del esquema *destino*.
- Por esta razón, también se las conoce como restricciones GLAV (*global-and-local-as-view)*.

El ejemplo anterior no es *ni LAV ni GAV*:

$$estudiante(E) \wedge anotado(E,C) \rightarrow \exists N\, \exists P\ profesor(P,C) \wedge nota(E,C,N)$$

Semántica de los mapeos de esquema

Mapeo $M = (S,\ T,\ \Sigma)$, donde Σ es un conjunto de s-t-TGDs.

Desde un punto de vista *semántico*, M puede verse como:

$$Inst(M) = \{(I,\ J) \mid I \text{ es una instancia sobre el esquema } \textit{origen},$$
$$J \text{ es una instancia sobre el esquema } \textit{destino}, \text{ y } (I,\ J) \models \Sigma\}$$

Una *solución* para una instancia origen I es una instancia destino J tal que $(I,\ J) \in Inst(M)$ (i.e., $(I,\ J) \models \Sigma$).

Semántica de los mapeos de esquema

Mapeo $M = (S,\ T,\ \Sigma)$, donde Σ es un conjunto de s-t-TGDs.

Informalmente, esta notación significa que J contiene todas las tuplas que surgen de aplicar las restricciones en Σ sobre tuplas en I.

Formalmente: $(I,\ J) \models \Sigma$

uede verse como:

tema *origen*,

ma *destino*, y $(I,\ J) \models \Sigma\}$

Una *solución* para una instancia origen I es una instancia destino J tal que $(I,\ J) \in \mathit{Inst}(M)$ (i.e., $(I,\ J) \models \Sigma$).

Intercambio de datos

- El problema del intercambio de datos a través del mapeo $M = (S, T, \Sigma)$ es entonces: *dada la instancia I, construir una solución J para I.*
- La *dificultad* de este problema radica en que:
  - Típicamente hay *muchas* soluciones posibles
  - ¿Cómo decidir *cuál de todas es la mejor*?

Intercambio de datos

Dos posibles escenarios “indeseados”:

- Una instancia origen puede no tener solución (sobre-especificación).
- Una instancia origen puede tener múltiples soluciones (sub-especificación).

Ejemplo: Esquemas *origen* $E(A, B)$, *destino* $H(A, B)$

$$\Sigma = E(X, Y) \rightarrow \exists Z \left(H(X, Z) \wedge H(Z, Y)\right)$$

Instancia origen: $I = \{E(a, b)\}$

- ¿Cuántas soluciones hay?

Intercambio de datos

Ejemplo: Esquemas origen $E(A,\ B)$, destino $H(A,\ B)$

$\Sigma = E(X,\ Y) \rightarrow \exists Z\,(H(X,\ Z) \wedge H(Z,\ Y))$

Instancia origen: $I = \{E(a,\ b)\}$

Constantes: $a,\ b,\ \ldots$ Nulls: $z_{1,}\ z_{2,\ \ldots}$

$J_1 = \{H(a,\ b),\ H(b,\ b)\}$

$J_2 = \{H(a,\ a),\ H(a,\ b)\}$

$J_3 = \{H(a,\ z_1),\ H(z_1,\ b)\}$

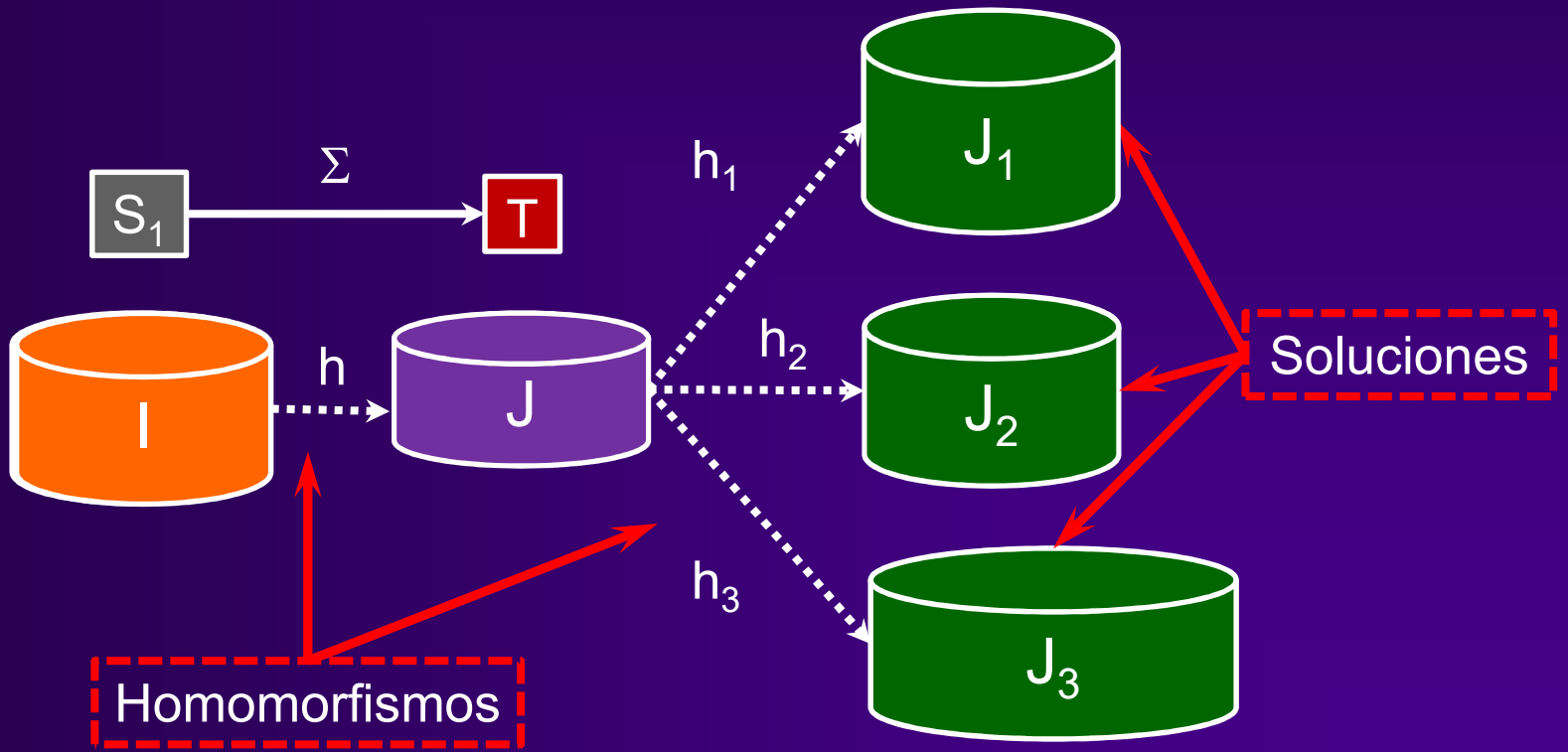
$J_4 = \{H(a,\ z_1),\ H(z_1,\ b),\ H(a,\ z_2),\ H(z_2,\ b)\}$

$J_5 = \{H(a,\ z_1),\ H(z_1,\ b),\ H(z_2,\ z_2)\}$

… *Soluciones*: ¡hay infinitas!

Soluciones universales

- Una solución J para I es *universal* si existen *homomorfismos* de J a todas las posibles soluciones para I.
- Intuitivamente, son las soluciones <u>más generales</u>:



Soluciones universales

Ejemplo: Esquemas origen $E(A,\ B)$, destino $H(A,\ B)$

$\Sigma = E(X,\ Y) \rightarrow \exists Z\,(H(X,\ Z) \wedge H(Z,\ Y))$

Instancia origen: $I = \{E(a,\ b)\}$

Soluciones:

$J_1 = \{H(a,\ b),\ H(b,\ b)\}$	*No* universal	Constantes: $a,\ b,\ \ldots$
$J_2 = \{H(a,\ a),\ H(a,\ b)\}$	*No* universal	Nulls: $z_1,\ z_2,\ \ldots$
$J_3 = \{H(a,\ z_1),\ H(z_1,\ b)\}$	Universal	
$J_4 = \{H(a,\ z_1),\ H(z_1,\ b),\ H(a,\ z_2),\ H(z_2,\ b)\}$		Universal
$J_5 = \{H(a,\ z_1),\ H(z_1,\ b),\ H(z_2,\ z_2)\}$		*No* universal

$\ldots$

Soluciones universales

- Las soluciones universales son similares en espíritu a los *unificadores más generales* en programación en lógica.
- Algunas propiedades:
  - *Unicidad modulo equivalencia homomórfica*: si J y J' son universales para I, entonces son equivalentes homomórficamente.
  - Si J es universal para I, y J' es universal para I', entonces las siguientes propiedades son *equivalentes*:
    - I e I' tienen el *mismo espacio* de soluciones.
    - J y J' son *homomórficamente equivalentes*.

El Chase

- Teorema: Sea $M = (S,\ T, \Sigma)$ un mapeo GLAV (i.e., Σ es un conjunto de s-t-TGDs). Entonces, para cada instancia origen I:
  - El procedimiento *chase* produce una *solución universal*, denotada con $chase_M(I)$.
  - El tiempo de ejecución para el chase está acotado por un *polinomio* en el tamaño de I (complejidad *data*).
- El chase puede ser *aumentado* también con dependencias sólo target (t-TGDs), y EGDs sólo target (t-EGDs).

Problemas en intercambio de datos

Dado un mapeo $M = (S, T, \Sigma)$ (donde Σ contiene s-t-TGDs, t-TGDs, y t-EGDs), los problemas fundamentales son:

- *Existencia* de soluciones $Sol(M)$
- *Construcción* de una solución universal

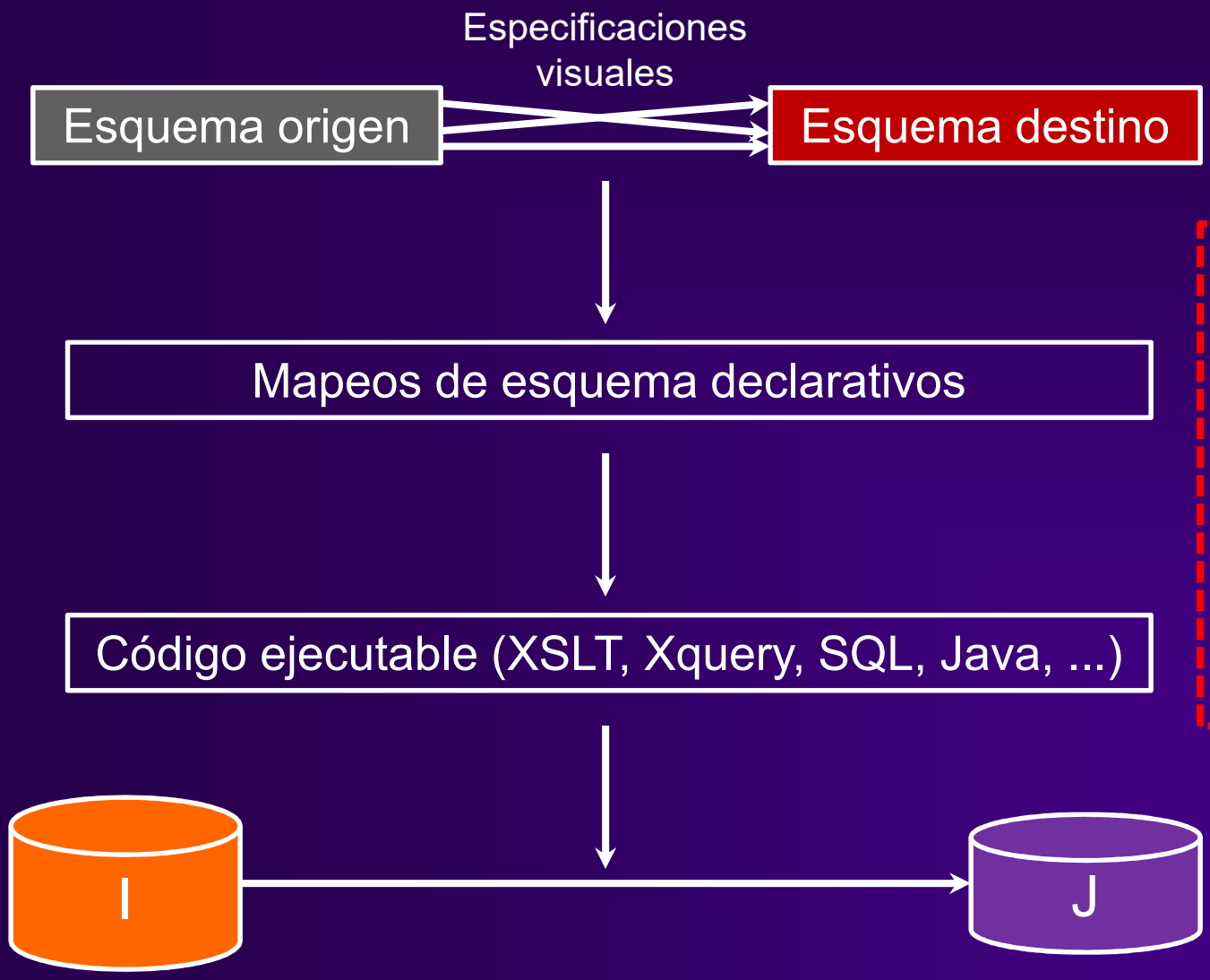
Complejidad: *Depende* de las t-TGDs… (*Ejercicio: ¿por qué?*)

- $Sol(M)$ puede ser *trivial* (siempre existen), soluciones universales pueden construirse en PTIME (LOGSPACE);
- $Sol(M)$ puede tener tamaño *polinomial* (construcción PTIME-completo), soluciones universales en PTIME; o
- La construcción de $Sol(M)$ puede ser *indecidible*, y pueden *no existir* soluciones universales (aunque existan soluciones).

Tratabilidad

- Idealmente, se buscan garantías de *tratabilidad computacional*:
  - Que la existencia de soluciones se pueda decidir en PTIME; y
  - que se puedan construir soluciones univ. en PTIME (si existen).
- Teorema: Dado $M = (S, T, \Sigma)$ (donde Σ contiene s-t-TGDs, t-TGDs de una clase *weakly-acyclic*, y t-EGDs):
  - Existen soluciones universales si y sólo si existen soluciones;
  - $Sol(M)$ se puede construir en PTIME; y
  - Se puede producir una solución universal canónica (si existe) en PTIME por medio del procedimiento chase.
- Este resultado se puede *generalizar* a otras clases de TGDs.

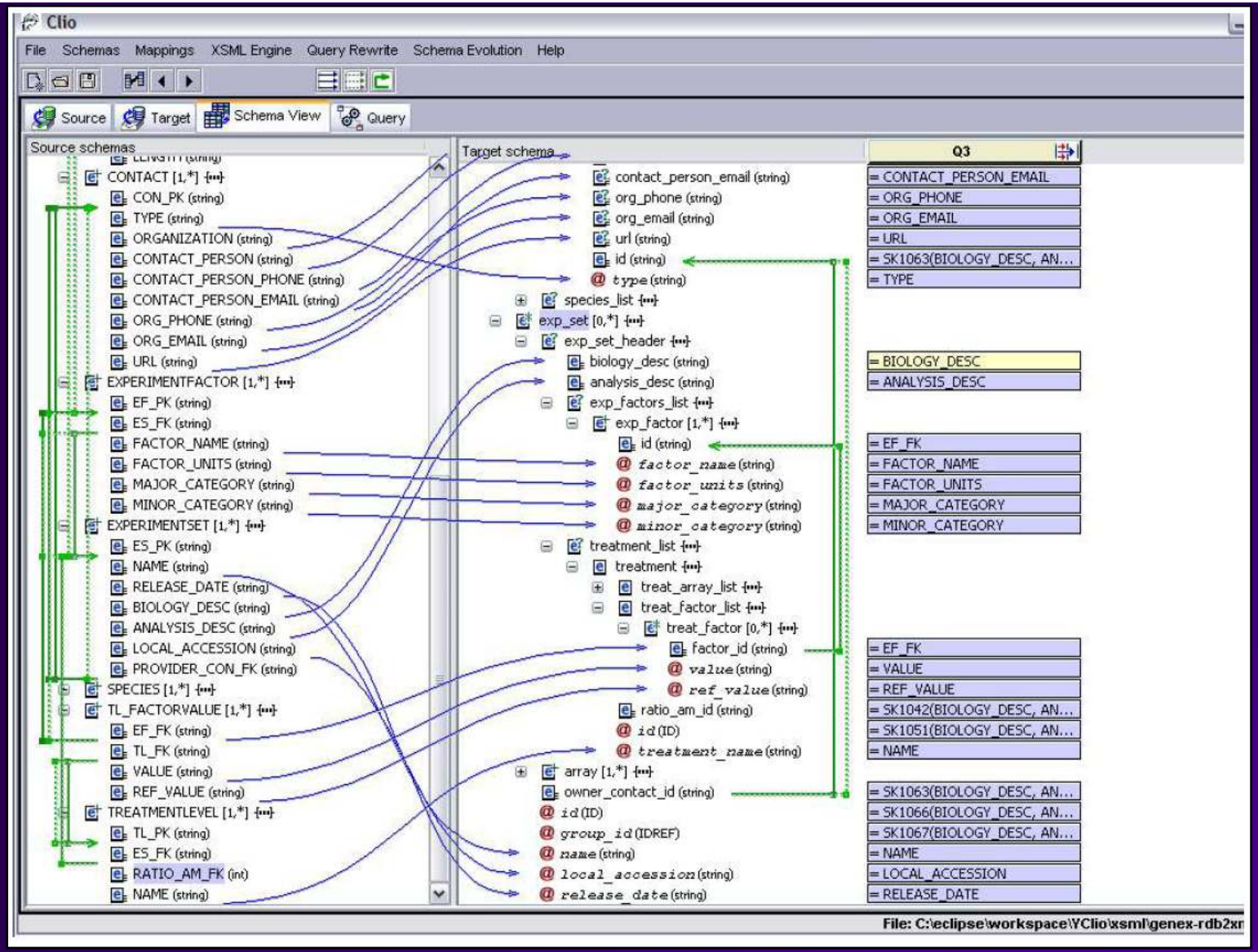
Sistemas en la práctica



Arquitectura genérica de sistemas tales como:

IBM Clio, HePToX, Altova MapForce, Stylus Studio, MS Biztalk Mapper, ...

Sistema *IBM Clio*



Solución universal mínima

- Las soluciones universales pueden *no ser únicas*.
- ¿Hay algún concepto de "*mejor*" solución universal?
- Una posibilidad es tomar la más *compacta*:

  Dada una instancia $\mathcal{J}$, la sub-instancia $\mathcal{J}'$ *más pequeña* (con respecto a inclusión) que es homomórficamente equivalente a $\mathcal{J}$ se denomina el *core* (núcleo) de $\mathcal{J}$.

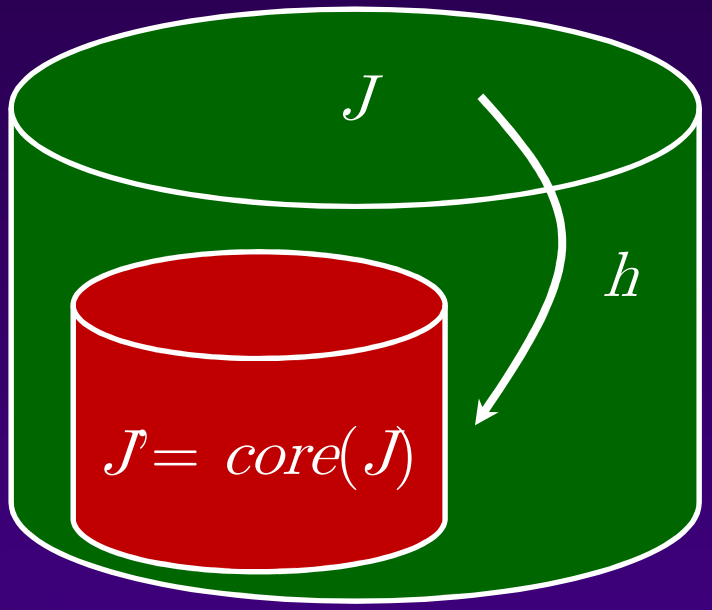
- Propiedades:
  - *Toda* estructura relacional finita tiene un core.
  - El core es *único*, módulo isomorfismo.

Solución universal mínima: *Core*

Definición:

J' es el *core* de J si:

- $J' \subseteq J$
- Existe un homomorfismo $h\colon J \to J'$
- No existe homomorfismo $g\colon J \to J''$ con $J'' \subset J'$.



Complejidad: el problema de *decidir* si un conjunto es el core de una instancia es *NP-hard*.

Solución universal mínima: *Core*

Ejemplo: Esquemas origen $E(A,\ B)$, destino $H(A,\ B)$

$\Sigma = E(X,\ Y) \rightarrow \exists Z\left(H(X,\ Z) \wedge H(Z,\ Y)\right)$

Instancia origen: $I = \{E(a,\ b)\}$

Soluciones: hay infinitas soluciones *universales*…

$J_3 = \left\{H(a,\ z_1),\ H(z_1,\ b)\right\}$ Es el *core* Constantes: $a,\ b,\ \ldots$

$J_4 = \left\{H(a,\ z_1),\ H(z_1,\ b),\right.$ Es universal, Nulls: $z_1,\ z_{2,\ \ldots}$

$\left.H(a,\ z_2),\ H(z_2,\ b)\right\}$ pero no core

…

Propiedades

- Teorema ([FKP03]): Sea $M = (S, T, \Sigma)$ un mapeo:
  - Todas las soluciones universales tienen el *mismo core*.
  - El core de las soluciones universales es la solución universal *más pequeña*.
  - Si toda restricción target es una EGD, entonces el core se puede computar en *tiempo polinomial*.
- Teorema ([GN06]): Dado $M = (S, T, \Sigma)$, si las restricciones target son o bien weakly-acyclic TGDs o t-EGDs, entonces el core se puede computar en tiempo polinomial.
- Nuevamente, se puede *generalizar* a otras clases de TGDs.

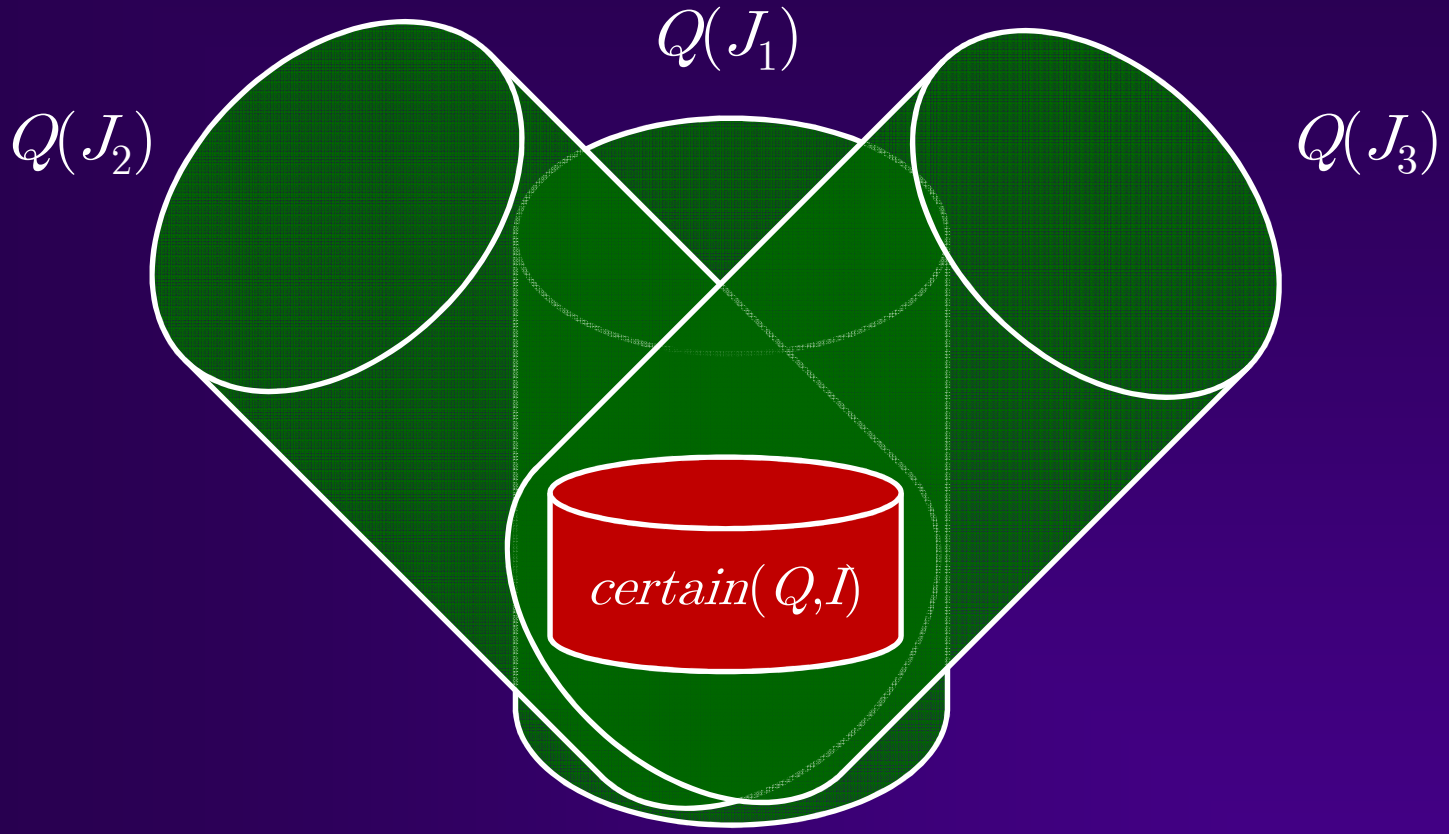
Consultas en intercambio de datos

- ¿Cuál es la *semántica* de la respuesta a consultas sobre el esquema destino?
- Definición: las *certain answers* (respuestas certeras) de una consulta Q sobre T a I se define:

$$certain(Q, I) = \bigcap \{ Q(J) \mid J \text{ es una solución para } I \}.$$

- Recordemos que se opera bajo la suposición de *mundo abierto*:
  - "El valor de verdad de una aserción no depende de que éste se sepa".
  - Es lo opuesto a la de *mundo cerrado* ("sólo lo que sabemos que es verdadero es verdadero").

Semántica de *certain answers*



$$certain(Q, I) = \bigcap \{ Q(J) \mid J \text{ es una solución para } I \}$$

Cómputo de *certain answers*

Si el conjunto de restricciones lo permite, es sencillo computar *certain answers* vía la construcción de *soluciones universales*:

- Computar una *solución universal* (por ejemplo, vía el chase).
- *Evaluar* la consulta y *descartar* toda tupla que contenga nulls, ya que éstas jamás pueden ser parte del resultado.

Agregando ontologías

Combinando lo visto hasta ahora con lo visto para la familia de lenguajes ontológicos *Datalog+/-*, podemos ver que:

- Ambos formalismos se basan en el mismo conjunto de *elementos básicos*: TGDs, EGDs, NCs, chase, …
- Las bases de datos pueden ser *generalizadas* a ontologías: pasamos de OBDA a OBDE (u *Ontological Data Exchange* – ODE).
- *Problema ODE* (o mapeo entre ontologías): $M = (S, T, \Sigma_s, \Sigma_t, \Sigma_{st})$
  - Una instancia destino J sobre T es una *solución* para una instancia origen I sobre S si y sólo si $(I \cup J) \models \Sigma_s \cup \Sigma_t \cup \Sigma_{st}$.
  - Igual que antes, $Sol(M)$ es el conjunto de estos pares (I, J).

Referencias

[Dong07] X. L. Dong, A. Y. Halevy, C. Yu: "*Data Integration with Uncertainty*". Proceedings of VLDB 2007, pp. 687–698.

[Gal09] A. Gal, M. V. Martinez, G. I. Simari, V. S. Subrahmanian: "*Aggregate Query Answering under Uncertain Schema Mappings*". Proc. ICDE 2009, pp. 940–951.

[Luk16] T. Lukasiewicz, M. V. Martinez, Livia Predoiu, G. I. Simari: "*Basic Probabilistic Ontological Data Exchange with Existential Rules*". Proc. AAAI 2016, pp. 1023–1029.

[FKP03] R. Fagin, P. G. Kolaitis, L. Popa: "*Data Exchange: Getting to the Core*". Proc. PODS 2003: 90–101.

[GN06] G. Gottlob, A. Nash: "*Data Exchange: Computing Cores in Polynomial Time*". Proc. PODS 2006: 40–49.

Referencias

Phokion G. Kolaitis: "*A Tutorial on Schema Mappings and Data Exchange*", dictado en DEIS 2010, Alemania, noviembre de 2010.

Parte del contenido de este curso está basado en trabajo de investigación realizado en colaboración con Thomas Lukasiewicz, Georg Gottlob, V.S. Subrahmanian, Avigdor Gal, Andreas Pieris, Giorgio Orsi, Livia Predoiu y Oana Tifrea-Marciuska.