On the Complexity of Deriving Schema Mappings from Database Instances

Pierre Senellart[1,2] Georg Gottlob[3]

1 INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE — INRIA FUTURS

2 UNIVERSITÉ PARIS-SUD 11

3 UNIVERSITY OF OXFORD

30 November 2007, *Gemo Seminar*

Different sources organize the same data differently

2007		
240	EE	Foto N. Afrati, Chen Li, Jeffrey D. Ullman: Using views to generate efficient evaluation plans for queries. J. Comput. Syst. Sci. 73(5): 703-724 (2007)
2005		
239	EE	Jeffrey D. Ullman: Gradiance On-Line Accelerated Learning. ACSC 2005: 3-6
238	EE	Serge Abiteboul, Rakesh Agrawal, Philip A. Bernstein, Michael J. Carey, Stefano Ceri, W. Bruce Croft, David J. DeWitt, Michael J. Franklin, Hector Garcia-Molina, Dieter Gawlick, Jim Gray, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, Martin L. Kersten, Michael J. Pazzani, Michael Lesk, David Maier, Jeffrey F. Naughton, Hans-Jörg Schek, Timos K. Sellis, Avi Silberschatz, Michael Stonebraker, Richard T. Snodgrass, Jeffrey D. Ullman, Gerhard Weikum, Jennifer Widom, Stanley B. Zdonik: The Lowell database research self-assessment. Commun. ACM 48(5): 111-118 (2005)
237	EE	Serge Abiteboul, Richard Hull, Victor Vianu, Sheila A. Greibach, Michael A. Harrison, Ellis Horowitz, Daniel J. Rosenkrantz, Jeffrey D. Ullman, Moshe Y. Vardi: In memory of Seymour Ginsburg 1928 - 2004. SIGMOD Record 34(1): 5-12 (2005)
2003		
236	EE	Jeffrey D. Ullman: A Survey of New Directions in Database System. DASFAA 2003: 3-
235	EE	Jeffrey D. Ullman: Improving the Efficiency of Database-System Teaching. SIGMOD Conference 2003: 1-3
234	EE	Jim Gray, Hans-Jörg Schek, Michael Stonebraker, Jeffrey D. Ullman: The Lowell Report. SIGMOD Conference 2003: 680
233	EE	Serge Abiteboul, Rakesh Agrawal, Philip A. Bernstein, Michael J. Carey, Stefano Ceri, W. Bruce Croft, David J. DeWitt, Michael J. Franklin, Hector Garcia-Molina, Dieter Gawlick, Jim Gray, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, Martin L. Kersten, Michael J. Pazzani, Michael Lesk, David Maier, Jeffrey F. Naughton, Hans-Jörg Schek, Timos K. Sellis, Avi Silberschatz, Michael Stonebraker, Richard T. Snodgrass, Jeffrey D. Ullman, Gerhard Weikum, Jennifer Widom, Stanley B. Zdonik: The Lowell Database Research Self Assessment CoRR cs.DB/0310006: (2003)

Different sources organize the same data differently

Querying websites using compact skeletons - all 11 versions »
A Rajaraman, **JD Ullman** - Journal of Computer and System Sciences, 2003 - Elsevier
Several commercial applications, such as online comparison shopping and process automation, require integrating information that is scattered across multiple websites or XML documents. Much research has been devoted to this problem, **...**
Cited by 13 - Related Articles - Web Search

[BOOK] Wprowadzenie do teorii automatów, jezyków i obliczen
JE Hopcroft, **JD Ullman**, B Konikowska - 2003 - Wydaw. Naukowe PWN
Cited by 15 - Related Articles - Web Search

Improving the efficiency of database-system teaching - all 3 versions »
JD Ullman - Proceedings of the 2003 ACM SIGMOD international conference …, 2003 - portal.acm.org
ABSTRACT The education industry has a very poor record of produc- tivity gains. In this brief article, I outline some of the ways the teaching of a college course in database systems could be made more ecient, and sta time used **...**
Cited by 4 - Related Articles - Web Search

A survey of new directions in database systems - all 5 versions »
JD Ullman - Database Systems for Advanced Applications, 2003.(DASFAA …, 2003 - ieeexplore.ieee.org
A survey of new directions in database systems. Ullman, JD Stanford University; This paper appears in: Database Systems for Advanced Applications, 2003. (DASFAA 2003). Proceedings. Eighth International **...**
Cited by 3 - Related Articles - Web Search

Motivation

Context

- Multiple data sources containing information about similar entities, with some redundancy (e.g., sources of the deep Web).
- Several different ways to present this information, i.e., several different schemata.
- No a priori information about (some of) these schemata.

How to know the relationships between these schemata, by just looking at the instances?

Other way to see this problem: Match operator on schema mappings, in the setting of data exchange.

Motivation

Context

- Multiple data sources containing information about similar entities, with some redundancy (e.g., sources of the deep Web).
- Several different ways to present this information, i.e., several different schemata.
- No a priori information about (some of) these schemata.

How to know the relationships between these schemata, by just looking at the instances?

Other way to see this problem: Match operator on schema mappings, in the setting of data exchange.

Motivation

Context

- Multiple data sources containing information about similar entities, with some redundancy (e.g., sources of the deep Web).
- Several different ways to present this information, i.e., several different schemata.
- No a priori information about (some of) these schemata.

How to know the relationships between these schemata, by just looking at the instances?

Other way to see this problem: Match operator on schema mappings, in the setting of data exchange.

Problem definition

Problem

Given two (relational) database instances I and J with different schemata, what is the optimal description Σ of J knowing I (with Σ a finite set of formulas in some logical language)?

What does optimal implies:

- Conciseness of description.
- Validity of facts predicted by I and Σ.
- All facts of J explained by I and Σ.

(Note the asymmetry between I and J; context of data exchange where J is computed from I and Σ).

Problem definition

Problem

Given two (relational) database instances I and J with different schemata, what is the optimal description Σ of J knowing I (with Σ a finite set of formulas in some logical language)?

What does optimal implies:

- Conciseness of description.
- Validity of facts predicted by I and Σ.
- All facts of J explained by I and Σ.

(Note the asymmetry between I and J; context of data exchange where J is computed from I and Σ).

Outline

Source-to-target tuple-generating dependencies

Definition (Source-to-target tgd)

First-order formula of the form:

$$\forall \mathbf{x}\, \varphi(x) \rightarrow \exists \mathbf{y}\, \psi(x, y)$$

with:

- φ conjunction of source relation atoms;
- ψ conjunction of target relation atoms;
- all variables of $\mathbf{x}$ bound in φ.

Example

$$\forall x_1 \forall x_2\, R_1(x_1, x_2) \wedge R_2(x_2) \rightarrow \exists y\, R'(x_1, y)$$

Particular tgds

We consider two ways of having simpler tgds:

- Disallow existential quantifiers on the right hand-side: full tgds.
- Disallow cycles on both left- and right-hand sides: acyclic tgds. (Classical notion of acyclicity on hypergraphs extending the basic notion of acyclicity on graphs.)

Examples

$\forall x_1 \forall x_2 \forall x_3\ R_1(x_1, x_2) \wedge R_2(x_2, x_3) \wedge R_3(x_3, x_1) \rightarrow R'(x_1)$ is cyclic (and full).

$\forall x_1 \forall x_2 \forall x_3\ R_1(x_1, x_2) \wedge R_2(x_2, x_3) \rightarrow R'(x_1)$ is acyclic (and full).

We then consider the languages:

$\mathcal{L}_{\text{tgd}}$: arbitrary source-to-target tgds;

$\mathcal{L}_{\text{full}}$: full tgds;

$\mathcal{L}_{\text{acyc}}$: acyclic tgds;

$\mathcal{L}_{\text{facyc}}$: full and acyclic tgds.

Particular tgds

We consider two ways of having simpler tgds:

- Disallow existential quantifiers on the right hand-side: full tgds.
- Disallow cycles on both left- and right-hand sides: acyclic tgds. (Classical notion of acyclicity on hypergraphs extending the basic notion of acyclicity on graphs.)

Examples

$\forall x_1 \forall x_2 \forall x_3\, R_1(x_1, x_2) \wedge R_2(x_2, x_3) \wedge R_3(x_3, x_1) \to R'(x_1)$ is cyclic (and full).
$\forall x_1 \forall x_2 \forall x_3\, R_1(x_1, x_2) \wedge R_2(x_2, x_3) \to R'(x_1)$ is acyclic (and full).

We then consider the languages:

$\mathcal{L}_{\text{tgd}}$: arbitrary source-to-target tgds;

$\mathcal{L}_{\text{full}}$: full tgds;

$\mathcal{L}_{\text{acyc}}$: acyclic tgds;

$\mathcal{L}_{\text{facyc}}$: full and acyclic tgds.

Particular tgds

We consider two ways of having simpler tgds:

- Disallow existential quantifiers on the right hand-side: full tgds.
- Disallow cycles on both left- and right-hand sides: acyclic tgds. (Classical notion of acyclicity on hypergraphs extending the basic notion of acyclicity on graphs.)

Examples

$\forall x_1 \forall x_2 \forall x_3\ R_1(x_1, x_2) \wedge R_2(x_2, x_3) \wedge R_3(x_3, x_1) \rightarrow R'(x_1)$ is cyclic (and full).

$\forall x_1 \forall x_2 \forall x_3\ R_1(x_1, x_2) \wedge R_2(x_2, x_3) \rightarrow R'(x_1)$ is acyclic (and full).

We then consider the languages:

$\mathcal{L}_{\text{tgd}}$: arbitrary source-to-target tgds;

$\mathcal{L}_{\text{full}}$: full tgds;

$\mathcal{L}_{\text{acyc}}$: acyclic tgds;

$\mathcal{L}_{\text{facyc}}$: full and acyclic tgds.

How to define the pertinence of a set of tgds?

Example

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

$\Sigma_0 = \varnothing$

$\Sigma_1 = \{\forall x \; R(x) \rightarrow R'(x,x)\}$

$\Sigma_2 = \{\forall x \; R(x) \rightarrow \exists y \; R'(x,y)\}$

$\Sigma_3 = \{\forall x \forall y \; R(x) \wedge R(y) \rightarrow R'(x,y)\}$

$\Sigma_4 = \{\exists x \exists y \; R'(x,y)\}$

How to define the pertinence of a set of tgds?

Example

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

$$\Sigma_0 = \varnothing$$

$$\Sigma_1 = \{\forall x\ R(x) \rightarrow R'(x, x)\}$$

$$\Sigma_2 = \{\forall x\ R(x) \rightarrow \exists y\ R'(x, y)\}$$

$$\Sigma_3 = \{\forall x \forall y\ R(x) \wedge R(y) \rightarrow R'(x, y)\}$$

$$\Sigma_4 = \{\exists x \exists y\ R'(x, y)\}$$

Idea

- Size of a formula: number of occurrences of variables and constants.
- Cost of a schema mapping Σ: Size of the minimum repair of Σ that is valid and explains all facts of J.
- Types of repairs considered:
  - "fix" a universal quantifier by adding conditions ($x = a$ or $x \neq a$);
  - "fix" an existential quantifier by giving corresponding constants ($\tau(\mathbf{x}) \rightarrow y = a$ with τ a conjunction of conditions on universally quantified variables);
  - add ground facts to the target instance.
- The problem is then to find a schema mapping of minimal cost.

Example of cost computation

Example

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

$$\forall x \; R(x) \rightarrow R'(x, x)$$

Predicted R'

a	a
b	b
c	c
d	d

Example of cost computation

Example

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

$$\forall x \; R(x) \wedge x \neq c \rightarrow R'(x, x)$$

Predicted R'

a	a
b	b
d	d

Example of cost computation

Example

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

$$\forall x \; R(x) \wedge x \neq c \rightarrow R'(x, x)$$

$$R(c, a)$$

Predicted R'

a	a
b	b
c	a
d	d

Example of cost computation

Example

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

$$\forall x \; R(x) \wedge x \neq c \rightarrow R'(x, x)$$

$$R(c, a)$$

$$R(g, h)$$

Predicted R'

a	a
b	b
c	c
d	d
g	h

Example of cost computation

Example

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

$$\forall x\ R(x) \land x \neq c \rightarrow R'(x,x)$$

$$\exists x \exists y\ R(x,y) \land x = c \land y = a$$

$$\exists x \exists y\ R(x,y) \land x = g \land y = h$$

Predicted R'

a	a
b	b
c	c
d	d
g	h

Example of cost computation

Example

R
a
b
c
d

R'	
a	a
b	b
c	a
d	d
g	h

$$\forall x\ R(x) \wedge x \neq c \rightarrow R'(x, x)$$
$$\exists x \exists y\ R(x, y) \wedge x = c \wedge y = a$$
$$\exists x \exists y\ R(x, y) \wedge x = g \wedge y = h$$

Cost: 17

Predicted R'

a	a
b	b
c	c
d	d
g	h

Problems considered

Decision problems of interest:

Cost	: Is the cost of a given schema mapping less than K?
Optimality	: Is a given schema mapping optimal?

Complexity? Algorithms?

Problems considered

Decision problems of interest:

Cost	:	Is the cost of a given schema mapping less than K?
Optimality	:	Is a given schema mapping optimal?

Complexity? Algorithms?

Outline

Behavior for simple operators

Consider the elementary operators of the relational algebra:

- Projection
- Intersection
- Selection (conjunction of atomic conditions)
- Cross Product
- Join (on a given attribute)

Theorem

For any elementary operator γ, the tgd naturally associated with γ is optimal with respect to $(I, \gamma(I))$ (or $(\gamma(J), J)$), under some basic assumptions.

Behavior for simple operators

Consider the elementary operators of the relational algebra:

- Projection
- Intersection
- Selection (conjunction of atomic conditions)
- Cross Product
- Join (on a given attribute)

Theorem

For any elementary operator γ, the tgd naturally associated with γ is optimal with respect to $(I, \gamma(I))$ (or $(\gamma(J), J)$), under some basic assumptions.

Examples of naturally associated tgds

Examples

	Condition	I and J	Optimal tgd
Projection	$I \neq \varnothing$	$J = \pi_1(I)$	$R(x,y) \to R'(x)$
	$\pi_1(J) \cap \pi_2(J) = \varnothing$, $\lvert\pi_1(J)\rvert \geqslant 2$	$I = \pi_1(J)$	$R(x) \to \exists y\ R'(x,y)$
Selection	$\lvert\sigma_\varphi(I)\rvert \geqslant \frac{\text{size}(\varphi)+2}{3}$	$J = \sigma_\varphi(I)$	$R(x) \to R'(x)$
	$\sigma_\varphi(J) \neq \varnothing$	$I = \sigma_\varphi(J)$	$R(x) \to R'(x)$
Product	$R_1^I \neq \varnothing$, $R_2^I \neq \varnothing$	$J = R_1^I \times R_2^I$	$R_1(x) \wedge R_2(y) \to R'(x,y)$
	${R_1'}^J \neq \varnothing$, ${R_2'}^J \neq \varnothing$	$I = {R_1'}^J \times {R_2'}^J$	$R(x,y) \to R_1'(x) \wedge R_2'(y)$

(Combined) Complexity Results

	$\mathcal{L}_{\text{tgd}}$	$\mathcal{L}_{\text{full}}$
Cost	Σ_3^P, Π_2^P-hard	Σ_2^P, (co)NP-hard
Optimality	Π_4^P, (co)NP-hard	Π_3^P, (co)NP-hard

	$\mathcal{L}_{\text{acyc}}$	$\mathcal{L}_{\text{facyc}}$
Cost	Σ_2^P, (co)NP-hard	NP-complete
Optimality	Π_3^P, (co)NP-hard	Π_2^P, (co)NP-hard

(Combined) Complexity Results

	$\mathcal{L}_{\text{tgd}}$	$\mathcal{L}_{\text{full}}$
Cost	Σ_3^P, Π_2^P-hard	Σ_2^P, (co)NP-hard
Optimality	Π_4^P, (co)NP-hard	Π_3^P, (co)NP-hard

	$\mathcal{L}_{\text{acyc}}$	$\mathcal{L}_{\text{facyc}}$
Cost	Σ_2^P, (co)NP-hard	NP-complete
Optimality	Π_3^P, (co)NP-hard	Π_2^P, (co)NP-hard

(Combined) Complexity Results

	$\mathcal{L}_{\text{tgd}}$	$\mathcal{L}_{\text{full}}$
Cost	$\Sigma_{\mathbf{3}}^{\boldsymbol{P}}$, $\Pi_{\mathbf{2}}^{\boldsymbol{P}}$-hard	$\Sigma_{\mathbf{2}}^{\boldsymbol{P}}$, **(co)NP**-hard
Optimality	$\Pi_{\mathbf{4}}^{\boldsymbol{P}}$, **(co)NP**-hard	$\Pi_{\mathbf{3}}^{\boldsymbol{P}}$, **(co)NP**-hard

	$\mathcal{L}_{\text{acyc}}$	$\mathcal{L}_{\text{facyc}}$
Cost	$\Sigma_{\mathbf{2}}^{\boldsymbol{P}}$, **(co)NP**-hard	**NP**-complete
Optimality	$\Pi_{\mathbf{3}}^{\boldsymbol{P}}$, **(co)NP**-hard	$\Pi_{\mathbf{2}}^{\boldsymbol{P}}$, **(co)NP**-hard

(Combined) Complexity Results

	$\mathcal{L}_{\text{tgd}}$	$\mathcal{L}_{\text{full}}$
Cost	Σ_3^P, Π_2^P-hard	Σ_2^P, (**co**)**NP**-hard
Optimality	Π_4^P, (**co**)**NP**-hard	Π_3^P, (**co**)**NP**-hard

	$\mathcal{L}_{\text{acyc}}$	$\mathcal{L}_{\text{facyc}}$
Cost	Σ_2^P, (**co**)**NP**-hard	**NP**-complete
Optimality	Π_3^P, (**co**)**NP**-hard	Π_2^P, (**co**)**NP**-hard

(Combined) Complexity Results

	$\mathcal{L}_{\text{tgd}}$	$\mathcal{L}_{\text{full}}$
Cost	Σ_3^P, Π_2^P-hard	Σ_2^P, (co)NP-hard
Optimality	Π_4^P, (co)NP-hard	Π_3^P, (co)NP-hard

	$\mathcal{L}_{\text{acyc}}$	$\mathcal{L}_{\text{facyc}}$
Cost	Σ_2^P, (co)NP-hard	NP-complete
Optimality	Π_3^P, (co)NP-hard	Π_2^P, (co)NP-hard

Vertex-Cover in r-partite r-uniform hypergraph

Vertex-Cover: find a set of vertices of minimal size that cover all (hyper)edges in a (hyper)graph.

- NP-complete for general (hyper)graphs.
- PTIME for bipartite graphs (Kőnig's theorem).

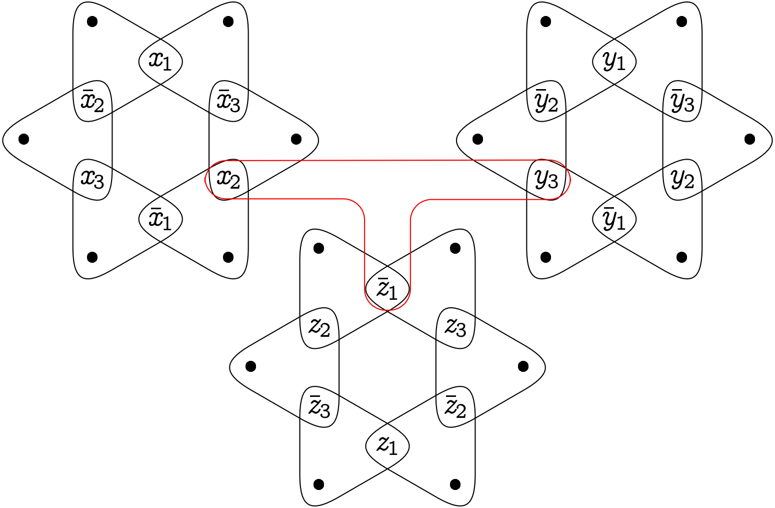
Lemma

Vertex-Cover is NP-complete for r-partite r-uniform hypergraphs for $r \geqslant 3$.

r-partite: partition of the set of vertices into r sets, with no hyperedge spanning vertices of two different sets.

r-uniform: every hyperedge spans r vertices.

Encoding of 3-SAT



$\neg z \vee x \vee y$

Cost is NP-hard for $\mathcal{L}_{\text{facyc}}$

Reduction from Vertex-Cover in 3-partite 3-uniform hypergraphs.

Without $x = a$ repairs on the left-hand side of a tgd:

- $R(x_1, x_2, x_3) \to R'(x_1)$
- Source instance: hypergraph
- Target instance: empty

Cost: size of the tgd plus twice the minimum size of a vertex cover.

With $x = a$ repairs: a little more difficult, but feasible!

Cost is NP-hard for $\mathcal{L}_{\text{facyc}}$

Reduction from Vertex-Cover in 3-partite 3-uniform hypergraphs.

Without $x = a$ repairs on the left-hand side of a tgd:

- $R(x_1, x_2, x_3) \rightarrow R'(x_1)$
- Source instance: hypergraph
- Target instance: empty

Cost: size of the tgd plus twice the minimum size of a vertex cover.

With $x = a$ repairs: a little more difficult, but feasible!

Cost is NP-hard for $\mathcal{L}_{\text{facyc}}$

Reduction from Vertex-Cover in 3-partite 3-uniform hypergraphs.

Without $x = a$ repairs on the left-hand side of a tgd:

- $R(x_1, x_2, x_3) \rightarrow R'(x_1)$
- Source instance: hypergraph
- Target instance: empty

Cost: size of the tgd plus twice the minimum size of a vertex cover.

With $x = a$ repairs: a little more difficult, but feasible!

Cost is NP-hard for $\mathcal{L}_{\text{facyc}}$

Reduction from Vertex-Cover in 3-partite 3-uniform hypergraphs.

Without $x = a$ repairs on the left-hand side of a tgd:

- $R(x_1, x_2, x_3) \to R'(x_1)$
- Source instance: hypergraph
- Target instance: empty

Cost: size of the tgd plus twice the minimum size of a vertex cover.

With $x = a$ repairs: a little more difficult, but feasible!

Outline

Extension to Relational Calculus

- Definition of repairs can be extended to relational calculus.
- Same definition of cost, optimality.
- Cost is not recursive (but co-r.e.).
- Computability of Optimality: open (!).

Other Cost Functions

Why not counting the number of tuples to add or remove in J?

. . . because it can be exponential in the size of the schema mapping!

Why not counting the number of tuples to add or remove in I or J?

. . . because selections are not captured!

Other Cost Functions

Why not counting the number of tuples to add or remove in J?

... because it can be exponential in the size of the schema mapping!

Why not counting the number of tuples to add or remove in I or J?

... because selections are not captured!

Other Cost Functions

Why not counting the number of tuples to add or remove in J?

... because it can be exponential in the size of the schema mapping!

Why not counting the number of tuples to add or remove in I or J?

... because selections are not captured!

Other Cost Functions

Why not counting the number of tuples to add or remove in J?

... because it can be exponential in the size of the schema mapping!

Why not counting the number of tuples to add or remove in I or J?

... because selections are not captured!

Outline

In summary...

- Formal framework for the discovery of symbolic relations between two data sources.
- High complexity (up to fourth level of $\mathbf{PH}$).

In summary...

- Formal framework for the discovery of symbolic relations between two data sources.
- High complexity (up to fourth level of **PH**).

- Link with Inductive Logic Programming?
- Heuristics?
- Generalization of acyclicity?

Merci.