


Master IF DBDM

ENS Lyon, A.Y. 2015-2016

Angela Bonifati

Foundations of Databases: Data Integration




Weakly acyclic sets of tgds

- Weakly acyclic sets of tgds contain as special cases:
  - Sets of full target tgds where $\varphi_T(x)$ and $\psi_T(x)$ are conjunctions of target atoms.
  - $\varphi_T(x) \rightarrow \psi_T(x)$,
  - Example: $H(x,z) \wedge H(z,y) \rightarrow H(x,y) \wedge C(z)$
- Full target tgds express containment between relational joins.

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr

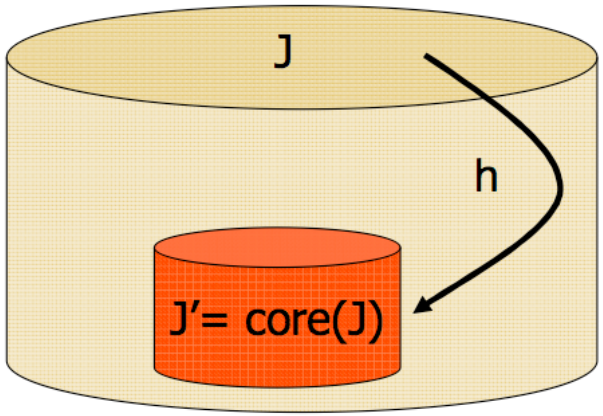



The smallest universal solution

- Fact: Universal solutions need not be unique.
  Question: Is there a "best" universal solution?
  - "small is beautiful" approach: There is a smallest universal solution (if solutions exist); hence, the most compact one to materialize.
  - Definition: The core of an instance J is the smallest subinstance J' that is homomorphically equivalent to J.
  - Fact: Every finite relational structure has a core. The core is unique up to isomorphism.

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr

The core (universal) solution



Definition: J′ is the core of J if

- J′ ⊆ J
- there is a homomorphism h:J→J′
- there is no homomorphism g:J→J″, where J″ ⊂ J′.

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




Example: which one is the core?

- Say whether/why the next J is a *core* solution
  - Q1.1: J1 = {H(a,b), H(b,b)}
  - Q1.2: J2 = {H(a,a), H(a,b)}
  - Q1.3: J3 = {H(a,X), H(X,b)}
  - Q1.4: J4 = {H(a,X), H(X,b), H(a,Y), H(Y,b)}
  - Q1.5: J5 = {H(a,X), H(X,b), H(Y,Y)}

Exercise 1 (homework)

- Exo1. Find a core solution for the following set of s-t tgds:
  - L(x1, x2, x3) -> \exists y C(x1,y)
  - L(x1, x2, x3), L(x4, x2, x5) -> E (x1, x4)
  - first with I = { L(DB,2012,Ross), L(OOP,2012,Stuart},L(DB, 2013,Barth)}
  - and then with I' = { L(DB,2012,Ross), L(OOP,2012,Stuart},L(DB, 2012,Barth)}

Core computation

- Theorem (FKP'05): M = (S, T, Σst , Σt) a schema mapping:
  - All universal solutions have the same core.
  - The core of the universal solutions is the smallest universal solution.
  - If every target constraint is an egd, then the core is polynomial-time computable.
- Theorem (Got'05): M = (S, T, Σst , Σt)
  - If every target constraint is an egd or a full tgd, then the core is polynomial-time computable.
- Theorem (GN'08) : M = (S, T, Σst , Σt)
  - If every target constraint is an egd or a weakly acyclic set of tgds (for which the chase is known to terminate), then the core is polynomial-time computable.




Core computation: the algorithm

- The chase algorithm followed by a minimization procedure:
  - first, *chase* the s-t tgds in Σst on I to generate a canonical pre-solution, J_{pre};
  - then, *chase* the target constraints (target tgds and especially egds) on J_{pre}, to generate a canonical universal solution, J;
  - *minimize* J by looking for endomorphic subsets that are still universal solutions to generate the core universal solution, J_0




Jpre, J and J0 in our example

Company

id	name	symbol
N1	Google	GOOG
N2	Yahoo	YHOO
I1	Apple	S1
I2	Adobe	S2
23	Yahoo!	YHOO
25	Adobe	ADBE

Grant

amount	company
25,000	I1
50,000	I2
18,000	23
50,000	25

Company

id	name	symbol
N1	Google	GOOG
I1	Apple	S1
I2	Adobe	S2
23	Yahoo!	YHOO
25	Adobe	ADBE

Grant

amount	company
25,000	I1
50,000	I2
18,000	23
50,000	25

Company

id	name	symbol
N1	Google	GOOG
I1	Apple	NULL
23	Yahoo!	YHOO
25	Adobe	ADBE

Grant

amount	company
25,000	I1
18,000	23
50,000	25

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr

TGDs

SOURCE-TO-TARGET TGDS

$m_1.\ \forall s,n: NYSE(s,n) \rightarrow \exists I: Company(I,n,s)$

$m_2.\ \forall n,c,a,pi: Public\text{-}Company(n,c) \wedge Public\text{-}Grant(a,pi,n) \rightarrow \exists I,S: Company(I,n,S) \wedge Grant(a,I)$

$m_3.\ \forall i,n,s: NSF\text{-}Grantee(i,n,s) \rightarrow Company(i,n,s)$

$m_4.\ \forall a,c: NSF\text{-}Grant(a,c) \rightarrow Grant(a,c)$

TARGET TGDS

$t_1.\ \forall a,c: Grant(a,c) \rightarrow \exists N,S: Company(c,N,S)$

TARGET EGDS

$e_1.\ \forall n,n',i,i',s: Company(i,n,s) \wedge Company(i',n',s) \rightarrow (i=i') \wedge (n=n')$

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




Chasing with SQL

- The chase algorithm can be implemented in SQL:
  - Given a tgd $\varphi(x) \rightarrow \exists y(\psi(x,y))$, in order to chase it over I we may see $\varphi(x)$ as a first-order query $Q\varphi$ with free variables x over S.
  - We may execute $Q\varphi$ (I) using SQL in order to find all vectors of constants that satisfy the premise.
  - We now need to insert the appropriate tuples into the target instance to satisfy $\psi(x,y)$.

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




OID Invention/Skolemization

- Given a vector of k universally quantified variables x, a Skolem term over x is a term of the form f(x) where f is a function symbol of arity k.
- Skolem terms are used to create fresh labeled nulls on the target.
- Given an assignment of values a for x, with the Skolem term above we (injectively) associate a labeled null $N_{f(a(x))}$.
- For each tgd m : $\varphi(x) \rightarrow \exists y(\psi(x,y))$, we use a different Skolem function $f_{m,yi}$ for each variable yi $\in$ y, and take as argument all universal variables that appear in the conclusion.

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




Chasing with SQL in our example

$$m_2.\ \forall n,c,a,pi,n\colon \textit{Public-Company}(n,c) \wedge \textit{Public-Grant}(a,pi,n) \rightarrow \exists I,S\colon \textit{Company}(I,n,S) \wedge \textit{Grant}(a,I)$$

In order to implement the chase in SQL, the tgd is first rewritten using Skolem terms as follows:

$$m_2'.\ \forall n,c,a,pi,n\colon \textit{Public-Company}(n,c) \wedge \textit{Public-Grant}(a,pi,n) \rightarrow \exists I,S\colon \textit{Company}(f_I(n,a),n,f_S(n,a)) \wedge \textit{Grant}(a,f_I(n,a))$$

As an example, we show below one of the two SQL statements to which m_2' is translated (we omit the second on *Grant* for space reasons):

```
INSERT into Company
   SELECT append(`fI(',c.name, `,', g.amount, `)'), c.n,
          append(`fS(',c.name, `,', g.amount, `)')
   FROM Public-Company c, Public-Grant g
   WHERE c.name = g.company
```




Pros/cons of canonical universal solutions

- Advantage: the canonical universal solution is easy to compute
- A schema mapping specified by s-t tgds can be compiled into a list of SQL queries computing the canonical universal solution.
- We can compute the canonical universal solution using any RDBMS.
- Disadvantage: the canonical universal solution may contain more null values and more facts than strictly necessary.




Redundancy of canonical solutions

- Canonical universal solutions contain redundant tuples (we ignore the egd e1 in the example)
- Example (grey tuples are redundant):

Company

id	name	symbol
N1	Google	GOOG
N2	Yahoo	YHOO
I1	Apple	S1
I2	Adobe	S2
23	Yahoo!	YHOO
25	Adobe	ADBE
23	M1	S3
25	M2	S4

Grant

amount	company
25,000	I1
50,000	I2
18,000	23
50,000	25

a. Canonical universal solution

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




How to eliminate the redundancy?

- Getting to the core: two methods
  - (1st) apply a post-processing step
    - generate a solution and trim redundant (grey) tuples
  - (2nd) avoid generating redundant (grey) tuples at all
- We focus on the detailed description of both methods in the remainder

Company

id	name	symbol
N1	Google	GOOG
I1	Apple	NULL
23	Yahoo!	YHOO
25	Adobe	ADBE

Grant

amount	company
25,000	I1
18,000	23
50,000	25

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




1st method

- Several algorithms have been proposed:
- Greedy [FKP'05], Blocks [FKP'05], Gottlob-Nash [GN'08] all these algorithms involve recursively performing two steps
  - 1. testing for the existence of a homomorphism from J into a proper subinstance of J, and
  - 2. removing facts from J (which goes beyond SQL, i.e., out-of-the-box data transformation).




2nd method

- Apply an algorithm for transforming every schema mapping specified by s-t tgds into an equivalent laconic schema mapping (assuming a linear order on the source domain).
- Algorithm for computing core universal solution:
  - Transform schema mapping M into equivalent laconic M'.
  - Compile M' into SQL queries computing canonical universal solution
  - Apply SQL queries to the source instance to obtain core universal solution
- Thus, yes we can compute the core universal solution with SQL queries using any RDBMS.




Computing the core: 1st method - Blocks

- Blocks are connected subsets of a variable graph (Gaifman graph):
  - vertices <=> variables
  - edges <=> shared tuples
- Example of blocks:
  - P(X,Y), Q(Y, Z), P(V,W), S(U), Q(T,U)
- Blocks can be mapped independently of each other.
- A homomorphism h:A → C exists iff there exist homomorphisms h_i:B_i →C for each block B_i of A.




Complexity of computing the core

[Fagin, Kolaitis, Popa, PODS 2003]:
– Core computation is NP-hard in general
– It is tractable if the block size b is bounded (in case of s-t tgds and egds):

core(I) can be computed in time

$$n \cdot O(|dom(I)|^{b+2}) = O(n^{b+3})$$

In absence of target dependencies, block size is bounded (every chase step introduces independent blocks).
Target dependencies ruin the idyll:

$$P(X,Y) \wedge Q(V,W) \rightarrow R(Y,W)$$

now X,Y,V,W are in the same block




The FindCore Algorithm (target TGDs)

[Gottlob and Nash, PODS 2006] describe a beautiful algorithm that computes cores in polynomial time.

The crux: chosen a variable **x** and a term **y** from *dom*(J), spot a part of an instance J **with bounded blocksize** to test if there is an endomorphism e s.t. e(**x**) = e(**y**)... and

iterate, iterate, iterate, iterate, iterate, iterate, iterate at most $|dom(J)|^2$ times,

until no more variables can be eliminated

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




FindCore (egds)

In presence of target EGDs, it is not possible to spot the characteristic part of an instance with fixed blocksize, to test existence of endomorphisms on:

– EGDs can unify variables anywhere in the instance and thus merge blocks in an unpredictable way

Solution of the original FindCore algorithm: simulate EGDs by TGDs.

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr

FindCore (egds)

Introducing **a new relation E/2** to hold equal terms.

- Substitute each EGD by a number of TGDs :
  instead of equating two terms, add them to E, i.e. replace x=y with E(x,y)

Add constraints enforcing equality axioms

- **E(x,y) → E(y,x) symmetry**
- **E(x,y) E(y,z) → E(x,z) transitivity**
- **R(x_1, ... x_k) → $x_i = x_i$ reflexivity for every relation R/k and i ∈{1...k}**

Add consistency constraints:

R(x_1, ... x_k) E(x_i,y) → R(x_1, ..., y, ..., x_k) for every tuple of every relation R/k and i ∈ {1...k}

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




FindCore (egds)

Generated TGDs may be **NOT weakly-acyclic**.

Define a special "nice" chase order ensuring termination of chase.

Disadvantages:

Equalities effect only after core computation → **chase no longer produces a solution**.

Performance overhead: add lots of redundant tuples.

"Nice" chase is slower than its don't-care-nondeterministic classical version




Example

- Imagine an instance after the source-to-target chase: **J = {R(x,y), P(y,x)}**

  Let the egd be **R(z,v)∧P(v,z) → z = v**
  The set of simulating TGDs is:

- R(z, v)∧P (v, z) → E (z ,v);
- E(x, y) → E (y, x); E (x, y) ∧E(y, z) → E (x , z);
- R(x, y) → E (x, x); R(x, y) → E (y, y);
- P(x, y) → E(x, x); P(x, y) → E(y, y);
- R(x, y)∧E(x, z) → R(z, y); R(x, y)∧E(y, z) → R(x, z);
- P(x, y)∧E(x, z) → P (z, y); P(x, y)∧E (y, z) → P(x,z)




Example (cont'd)

The solution satisfying the simulating TGDs is { R(x, y), R(x, x), R(y, x), R(y, y), P(y, x), P(y, y), P(x, y), P(x, x) } ∪ { E(x, x), E (x, y), E(y, x), E (y, y) }

Only after computing the core, we get the instance satisfying the initial EGD: {R(x,x), P(x,x)} (we can ignore the auxiliary relation E)

If EGDs are enforced as usual, we get this solution just after a single chase step.




2nd method: Laconic schema mappings

- Why creating redundant tuples in the first place?
  - Compute the core directly (without any minimization)
- Formal Definition A schema mapping M is *laconic* if for every source instance I, the canonical universal solution of I with respect to M is the core universal solution of I with respect to M.
- Definition Schema mapping is laconic, if chasing it (naively) produces a core. Naive chase: fire each st tgd for each distinct tuple satisfying its antecedent.
- Proposition If a schema mapping (S, T, Σ) specified by s-t tgds is laconic, then for each s-t tgd $\forall x(\varphi \rightarrow \exists y\, \psi) \in \Sigma st$, the canonical instance of ψ is a core.

Example

S: PTStudent(age,name)
GradStudent(age,name)

T: Advised(sname,facid)
WorksWith(sname,facid)

Σ: PTStudent(x,y) → ∃z. Advised(y,z)
GradStudent(x,y) → ∃z.(Advised(y,z) ∧ WorksWith(y,z))

Source instance

PTStudent

age	name
32	John
30	Ann

GradStudent

age	name
27	Bob
30	Ann

Canonical univ. sol.

Advised

sname	facid
John	N1
Ann	N2
Bob	N3
Ann	N4

WorksWith

sname	facid
Bob	N3
Ann	N4

Core univ.sol.

Advised

sname	facid
John	N1
Bob	N3
Ann	N4

WorksWith

sname	facid
Bob	N3
Ann	N4

Example (cont'd)

S: PTStudent(age,name)
GradStudent(age,name)

T: Advised(sname,facid)
WorksWith(sname,facid)

Σ: PTStudent(x,y) ∧ ¬∃u.Gradstudent(u,y) → ∃z. Advised(y,z)
GradStudent(x,y) → ∃z.(Advised(y,z) ∧ WorksWith(y,z))

Source instance

PTStudent

age	name
32	John
30	Ann

GradStudent

age	name
27	Bob
30	Ann

Canonical univ. sol.

Advised

sname	facid
John	N1
~~Ann~~	~~N2~~
Bob	N3
Ann	N4

WorksWith

sname	facid
Bob	N3
Ann	N4

Core univ.sol.

Advised

sname	facid
John	N1
Bob	N3
Ann	N4

WorksWith

sname	facid
Bob	N3
Ann	N4




Other examples

- $P(x) \rightarrow \exists y \exists z\, R(x, y) \wedge R(x, z)$
  - (*laconic*) $P(x) \rightarrow \exists y\, R(x, y)$
- $P(x) \rightarrow \exists y\, R(x, y)$ $P(x) \rightarrow R(x, x)$
  - (*laconic*) $P(x) \rightarrow R(x,x)$
- $R(x, y) \rightarrow S(x, y)$ $P(x) \rightarrow \exists y\, S(x, y)$
  - (*laconic*) $R(x,y) \rightarrow S(x,y)$ $P(x) \wedge \neg \exists y\, R(x, y) \rightarrow \exists y\, S(x, y)$




More examples (Ten Cate et al. 2009)

- No self-joins in the conclusion of tgds
  - $S1(x, y) \rightarrow (\exists z)\ P(x, z) \wedge Q(z, y)$
  - $S2(x, v) \rightarrow P(x, v) \qquad S3(v, y) \rightarrow Q(v, y)$
  - Laconic variant of the first tgd: $S1(x, y) \wedge \neg S2(x, v) \wedge \neg S3(v, y) \rightarrow (\exists z)\ P(x, z) \wedge Q(z, y)$
- Tgds with self-joins in the conclusion
  - $R(x, y) \rightarrow (\exists z)\ S(x, z) \wedge S(y, z)$
  - Laconic variant: $(R(x, y) \vee R(y, x)) \wedge x \leq y \rightarrow (\exists z)\ S(x, z) \wedge S(y, z)$




More examples (Ten Cate et al. 2009)

- Both negation and order on the source domain are necessary.
- Rewritten mappings can be exponential in the number of dependencies of the original, non-laconic mapping.
- Skolemized form, suitable for SQL implementation
- $S(x1, x2, x3) \rightarrow \exists y\ R(x1, y)$
  - $S(x1, x2, x3) \rightarrow R(x1, f\ (x1, x2, x3))$
  - $S(1, 3, 4) \Rightarrow R(1, \text{'f(1,3,4)'})$




Embracing target constraints in practice

- No complete solution, unless target constraints can be fully "captured" by the st tgds. (E.g.: bounded chase property.)
- *Best-effort* approaches are available and can be helpful in practice. Target functional dependencies [Marnette, Mecca, and Papotti, 2010]
  - A FO implementation Σ_{st}^{FO} of the mapping $M = \{S, T, \Sigma st \cup \Sigma t\}$ where Σt consists of FDs, is a set of st tgds (having UCQs with negation in the antecedents).
  - If chase$(I, \Sigma_{st}^{FO}) \models \Sigma t$, then Σ_{st}^{FO} succeeds on I, and fails otherwise.
    - Soundness: If Σ_{st}^{FO} succeeds on I, then chase(I, Σ_{st}^{FO}) is a universal solution. E.g., Σ_{st}^{FO} does not "invent" target artefacts.
    - Completeness: Σ_{st}^{FO} succeeds on I iff M has solutions on I.

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr

Compute the laconic schema mappings for the following s-t tgds:

SOURCE-TO-TARGET TGDS

m_1. $\forall s, n: NYSE(s,n) \rightarrow \exists I: Company(I,n,s)$

m_2. $\forall n, c, a, pi: Public\text{-}Company(n,c) \wedge Public\text{-}Grant(a,pi,n) \rightarrow \exists I, S: Company(I,n,S) \wedge Grant(a,I)$

m_3. $\forall i, n, s: NSF\text{-}Grantee(i,n,s) \rightarrow Company(i,n,s)$

m_4. $\forall a, c: NSF\text{-}Grant(a,c) \rightarrow Grant(a,c)$

Exercise 3 (Homework)

The following tgds are in the laconic and non laconic form; explain how the laconic form lets directly derive the core solution

E3.1 $R(x, y) \rightarrow \exists z\, S(x, y, z)$ $\quad R(x, x) \rightarrow S(x, x, x)$

(*laconic*) $R(x,y) \wedge x!=y \rightarrow \exists z\, S(x,y,z)$ $\quad R(x, x) \rightarrow S(x, x, x)$

E3.2 $P(x) \rightarrow \exists y\, R1(x, y)$

$Q(x) \rightarrow \exists yzu\, (R2(x, y) \wedge R2(z, y) \wedge R1(z,u))$

(*laconic*) $P(x) \rightarrow \exists y\, R1(x, y)$

$Q(x) \wedge \neg P(x) \rightarrow \exists yzu\, (R2(x, y) \wedge R2(z, y) \wedge R1(z, u))$

$Q(x) \wedge P(x) \rightarrow \exists y\, (R2(x, y))$

Angela Bonifati (UCBL), Angela.Bonifati@univ-lyon1.fr




List of References

- [FPK'05] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa: Data exchange: getting to the core. ACM Trans. Database Syst. 30(1): 174-210 (2005)
- [FPM'05] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, Lucian Popa: Data exchange: semantics and query answering. Theor. Comput. Sci. 336(1): 89-124 (2005)
- [GN'08] Georg Gottlob, Alan Nash: Efficient core computation in data exchange. J. ACM 55(2): (2008)
- [Got'05] Georg Gottlob: Computing cores for data exchange: new algorithms and practical solutions. PODS 2005: 148-159
- [HN'92] Pavol Hell, Jaroslav Nesetril: The core of a graph. Discrete Mathematics 109(1-3): 117-126 (1992)
- [CCK'09] Balder ten Cate, Laura Chiticariu, Phokion G. Kolaitis, Wang Chiew Tan: Laconic Schema Mappings: Computing the Core with SQL Queries. PVLDB 2(1): 1006-1017 (2009)
- [MMP'10] Bruno Marnette, Giansalvatore Mecca, Paolo Papotti: Scalable Data Exchange with Functional Dependencies. PVLDB 3(1): 105-116 (2010)