



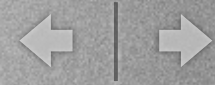
# Master I IF DBDM

---

ENS Lyon, A.Y. 2015-2016

Angela Bonifati

Foundations of Databases: Data Integration



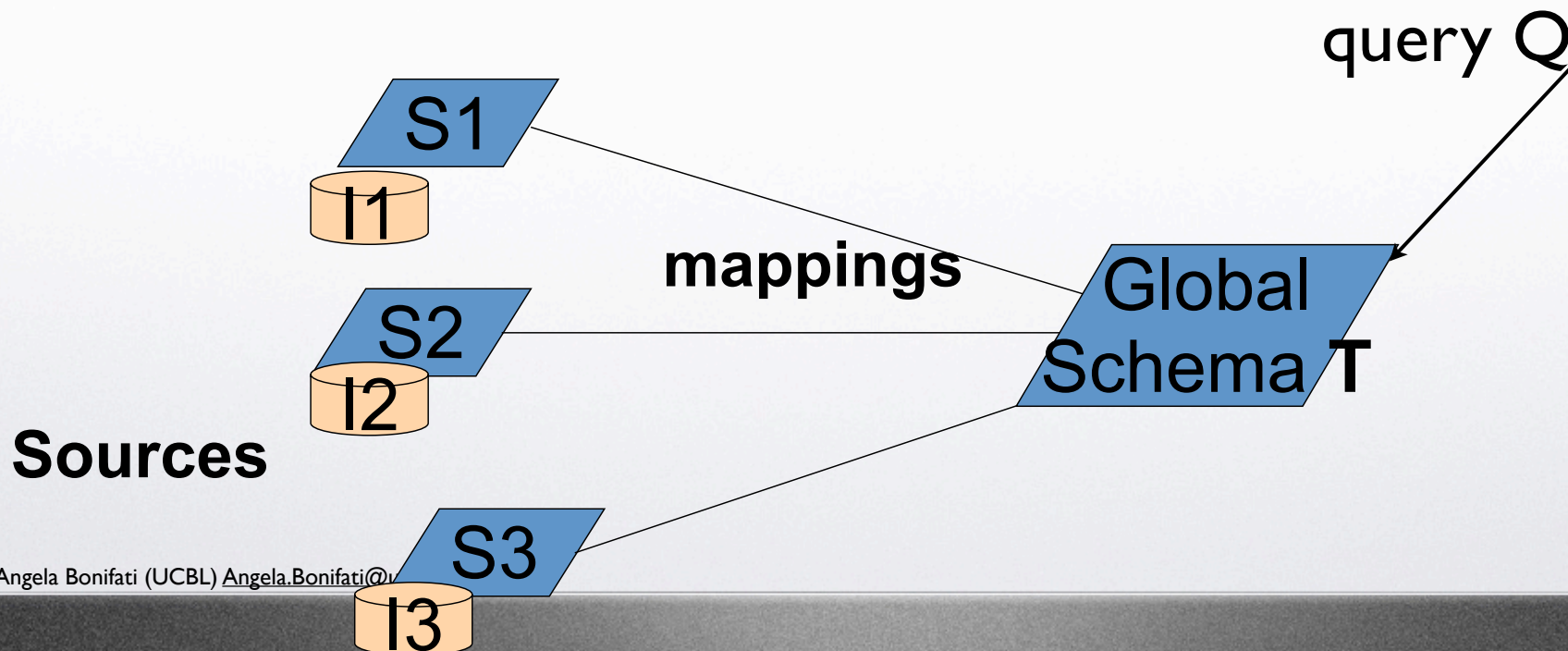
# Introduction

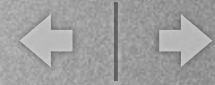
- Data is inherently heterogeneous
  - Due to the explosion of online data repositories
  - Due to the variety of users, who develop a wealth of applications
    - At different time
    - With disparate requirements in their mind
- A fundamental requirement is to translate data across different formats and to ensure data interoperability
  - Data Integration, Data Exchange are two facets of the same problem
  - Schema Integration and Schema Evolution are also important



# Data Integration

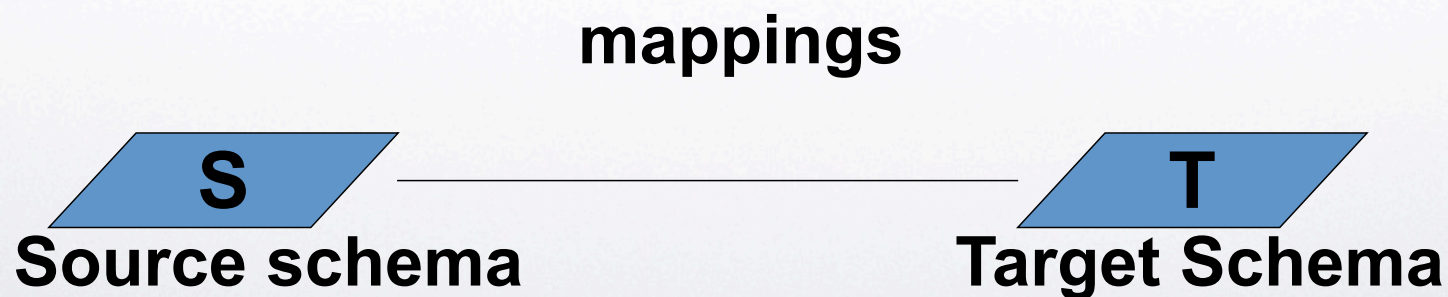
- Data integration [Lenzerini 2002]
  - Query heterogeneous data in different **sources** via a virtual **global** schema



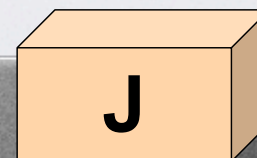
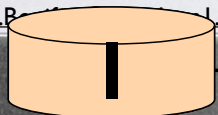


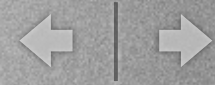
# Data exchange

- Data exchange [Fagin et al. 2005]
  - Transform data structured under a **source** schema into data structured under a different **target** schema.



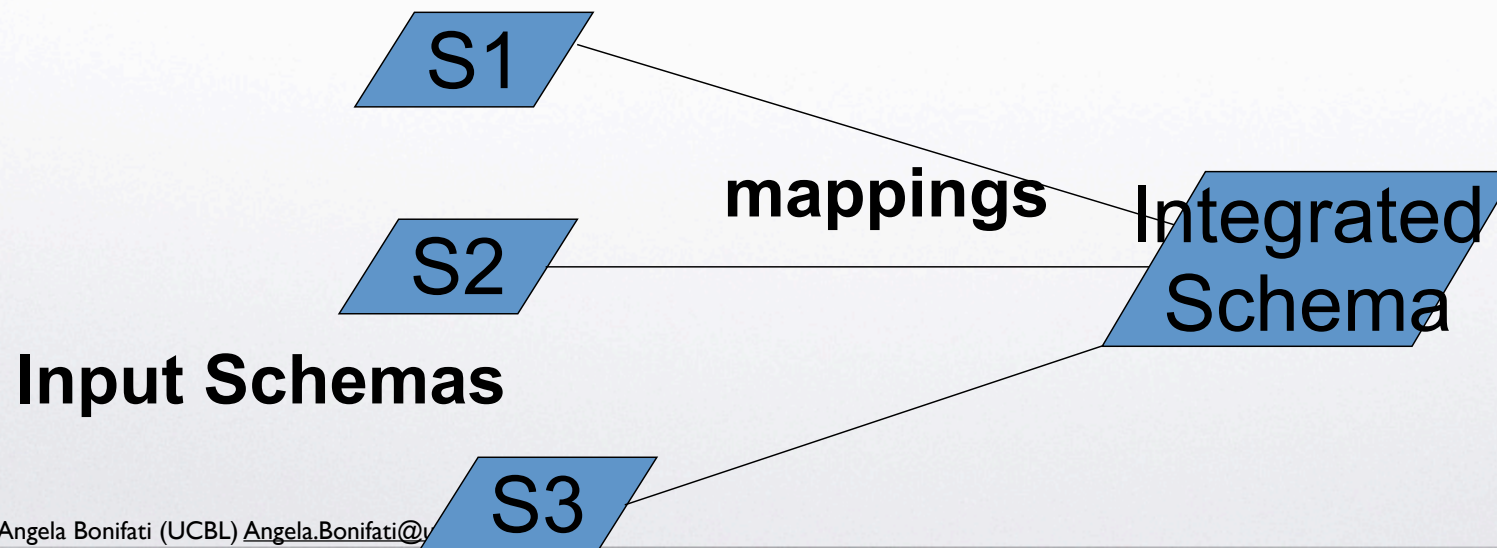
Angela Bonifati (UCBL) Angela Bonifati, University of  
Lfr





# Schema Integration

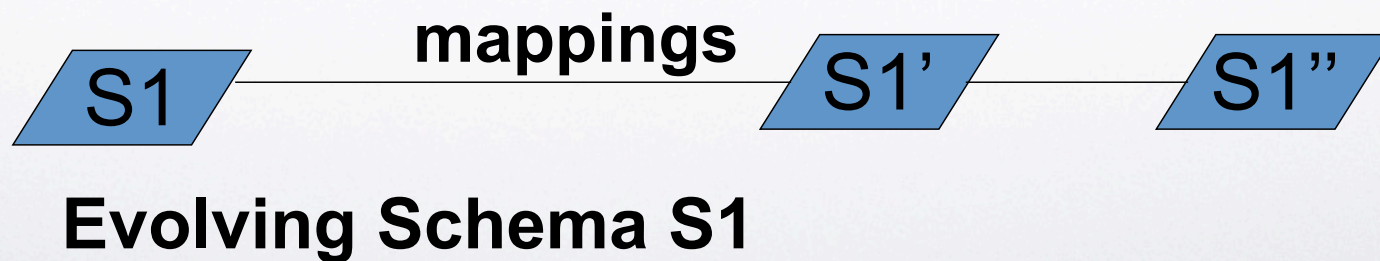
- Schema integration [Batini et al. 1986]
- A set of source schemas need to be integrated into one mediated schema





# Schema Evolution

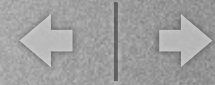
- Schema evolution [Lerner 2000]
- An original schema  $S1$  evolves into subsequent versions  $S1'$ ,  $S1''$  etc.





# Data Exchange

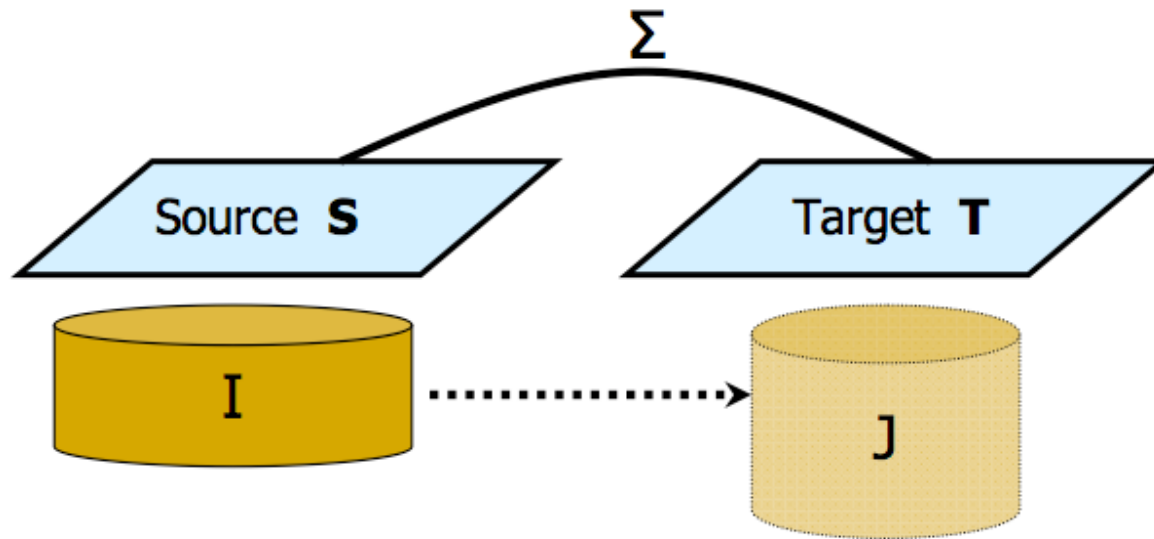
- Data Exchange is an old, but recurrent, database problem
- Phil Bernstein, Microsoft – 2003 “Data exchange is the oldest database problem”
- **EXPRESS**: IBM San Jose Research Lab – 1977  
**EX**traction, **P**rocessing, and **RES**tructuring **S**ystem for transforming data between hierarchical databases.
- Data Exchange underlies: Data Warehousing, ETL (Extract-Transform-Load) tasks; XML Publishing, XML Storage; more recently, exporting relational data to RDF.



# Schema mappings

- Schema mappings: high-level, declarative assertions that specify the relationship between two schemas.
- Ideally, schema mappings should be
  - **expressive** enough to specify data interoperability tasks;  
**simple** enough to be efficiently manipulated by tools.
- Schema mappings constitute the essential **building blocks** in formalizing **data integration** and **data exchange**.
- Schema mappings help with the development of tools: are easier to generate and manage (semi)-automatically; can be compiled into SQL/XSLT scripts automatically.





- Schema Mapping  $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$
- Source schema  $\mathbf{S}$ , Target schema  $\mathbf{T}$
- High-level, declarative assertions  $\Sigma$  that specify the relationship between  $\mathbf{S}$  and  $\mathbf{T}$ .
- Data Exchange via the schema mapping  $\mathbf{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ 
  - Transform a given source instance  $I$  to a target instance  $J$ , so that  $\langle I, J \rangle$  satisfy the specifications  $\Sigma$  of  $\mathbf{M}$ .



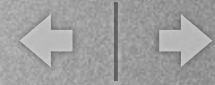
# Solutions in schema mappings

- **Definition:** Schema Mapping  $M = (S, T, \Sigma)$  If  $I$  is a source instance, then a *solution* for  $I$  is a target instance  $J$  such that  $(I, J)$  satisfy  $\Sigma$ .
- **Fact:** In general, for a given source instance  $I$ ,
- *No* solution for  $I$  may exist or
- *Multiple* solutions for  $I$  may exist; in fact, infinitely many solutions for  $I$  may exist.



# Schema mapping specification languages

- **Question:** How are schema mappings specified?
- **Answer:** Use logic. In particular, it is natural to try to use first-order logic as a specification language for schema mappings.
- **Fact:** There is a fixed first-order sentence specifying a schema mapping  $M^*$  such that  $\text{Sol}(M^*)$  is undecidable.
  - Reason: undecidability of validity in FOL
- Hence, we need to restrict ourselves to well-behaved fragments of first-order logic.



# Embedded dependencies

- **Dependency Theory:** extensive study of constraints in relational databases in the 1970s and 1980s.
- **Embedded Implicational Dependencies:** R. Fagin, C. Beeri and M. Vardi, ...
  - Class of constraints with a balance between high expressive power and good algorithmic properties:
  - **Tuple-generating dependencies (tgds)** Inclusion and multi-valued dependencies are a special case.
  - **Equality-generating dependencies (egds)** Functional dependencies are a special case.



## specification language

- The relationship between source and target is given by formulas of first-order logic, called **Source-to-Target Tuple Generating Dependencies (s-t tgds)**
- $\varphi(x) \rightarrow \exists y \psi(x, y)$ , where
- $\varphi(x)$  is a conjunction of atoms over the source;  $\psi(x, y)$  is a conjunction of atoms over the target.
- Example:
- $(\text{Student}(s) \wedge \text{Enrolls}(s,c)) \rightarrow \exists t \exists g (\text{Teaches}(t,c) \wedge \text{Grade}(s,c,g))$



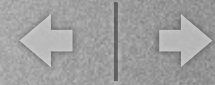
## specification language

- s-t tgds assert that: some SPJ source query is contained in some other SPJ target query
- $(\text{Student}(s) \wedge \text{Enrolls}(s,c)) \rightarrow \exists t \exists g (\text{Teaches}(t,c) \wedge \text{Grade}(s,c,g))$
- s-t tgds generalize the main specifications used in data integration:
  - They generalize LAV (local-as-view) specifications:
    - $P(x) \rightarrow \exists y \psi(x, y)$ , where  $P$  is a source schema.
  - They generalize GAV (global-as-view) specifications:
    - $\varphi(x) \rightarrow R(x)$ , where  $R$  is a target schema
    - They are equivalent full tgds:  $\varphi(x) \rightarrow \psi(x)$ , where  $\varphi(x)$  and  $\psi(x)$  are conjunctions of atoms



# Examples of simple mapping tasks

- Let us consider some simple tasks that a schema mapping specification language should support:
  - Copy (Nicknaming): Copy each source table to a target table and rename it.
  - Projection: Form a target table by projecting on one or more columns of a source table.
  - Decomposition: Decompose a source table into two or more target tables.
  - Column Augmentation: Form a target table by adding one or more columns to a source table.
  - Join: Form a target table by joining two or more source tables.
  - Combinations of the above (e.g., “join + column augmentation”)



# Examples of simple mapping tasks

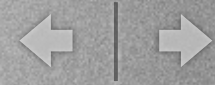
- Copy (Nicknaming):
  - $\forall x_1, \dots, x_n (P(x_1, \dots, x_n) \rightarrow R(x_1, \dots, x_n))$
- Projection:
  - $\forall x, y, z (P(x, y, z) \rightarrow R(x, y))$
- Decomposition:
  - $\forall x, y, z (P(x, y, z) \rightarrow R(x, y) \wedge T(y, z))$
- Column Augmentation:
  - $\forall x, y (P(x, y) \rightarrow \exists z R(x, y, z))$
- Join:
  - $\forall x, y, z (E(x, z) \wedge F(z, y) \rightarrow R(x, y, z))$
- Combinations of the above (e.g., "join + column augmentation")
  - $\forall x, y, z (E(x, z) \wedge F(z, y) \rightarrow \exists w T(x, y, z, w))$



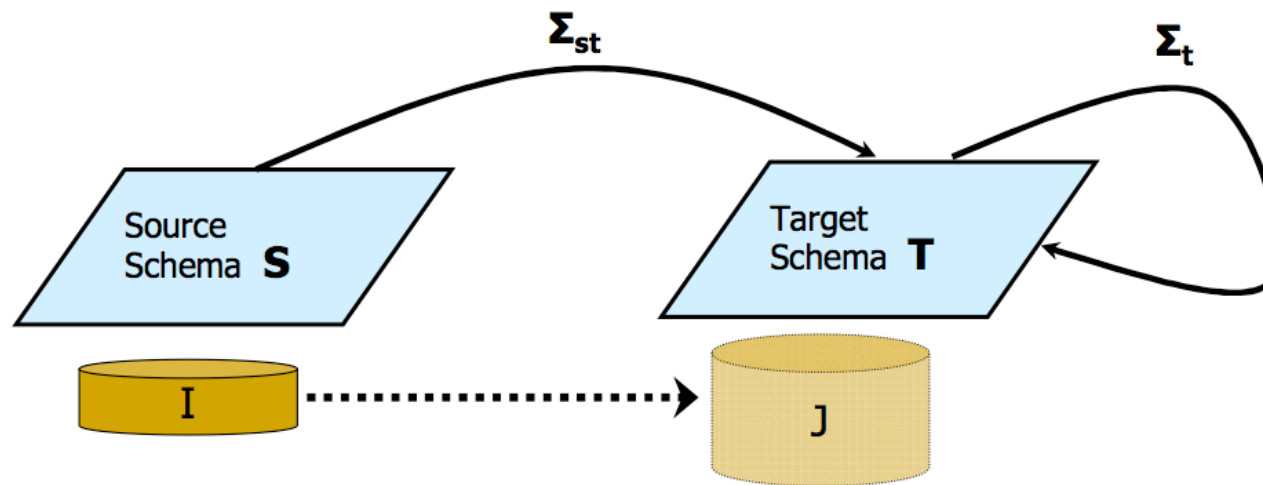


# Target dependencies

- In addition to source-to-target dependencies, we also consider target dependencies:
  - **Target Tgds** :  $\varphi_T(x) \rightarrow \exists y \psi_T(x, y)$
  - Dept (did, dname, mgr\_id, mgr\_name)  $\rightarrow$  Mgr (mgr\_id, did) (a target inclusion dependency constraint)
  - **Target Egds** (Equality Generating Dependencies):  $\varphi_T(x) \rightarrow (x_1 = x_2)$
  - (Mgr(e, d1)  $\wedge$  Mgr(e, d2))  $\rightarrow$  (d1 = d2) (a target key constraint)



# Data exchange framework



- Schema Mapping  $M = (S, T, \Sigma_{st}, \Sigma_t)$ , where
  - $\Sigma_{st}$  is a set of source-to-target tgds
  - $\Sigma_t$  is a set of target tgds and target egds

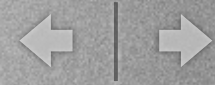


# Multiple solutions

- Fact: Given a source instance, multiple solutions may exist.
- Example: Source relation  $E(A,B)$ , target relation  $H(A,B)$
- $\Sigma: E(x,y) \rightarrow \exists z (H(x,z) \wedge H(z,y))$ 
  - Source instance  $I = \{E(a,b)\}$
  - Solutions: Infinitely many solutions exist



- Consider a set of source-to-target dependencies  $\Sigma_{st}$  :
  - (d1)  $\text{EmpCity}(e, c) \rightarrow \exists H \text{ Home}(e, H)$ ,
  - (d2)  $\text{EmpCity}(e, c) \rightarrow \exists D (\text{EmpDept}(e, D) \wedge \text{DeptCity}(D, c))$ ,
  - (d3)  $\text{LivesIn}(e, h) \rightarrow \text{Home}(e, h)$ ,
  - (d4)  $\text{LivesIn}(e, h) \rightarrow \exists D \exists C (\text{EmpDept}(e, D) \wedge \text{DeptCity}(D, C))$ ,
- and a source instance  $I$  such that:
  - $I = \{\text{EmpCity}(\text{Alice}, \text{SJ}), \text{EmpCity}(\text{Bob}, \text{SD}), \text{LivesIn}(\text{Alice}, \text{SF}), \text{LivesIn}(\text{Bob}, \text{LA})\}$ .
- **Which possible solutions  $J$  do exist?**



# Possible solutions $J_0, J, J_0'$

- $J_0 = \{\text{Home}(\text{Alice}, \text{SF}), \text{Home}(\text{Bob}, \text{LA}), \text{EmpDept}(\text{Alice}, \text{D1}), \text{EmpDept}(\text{Bob}, \text{D2}), \text{DeptCity}(\text{D1}, \text{SJ}), \text{DeptCity}(\text{D2}, \text{SD})\}$ ,
- $J = \{\text{Home}(\text{Alice}, \text{SF}), \text{Home}(\text{Bob}, \text{LA}), \text{Home}(\text{Alice}, \text{H1}), \text{Home}(\text{Bob}, \text{H2}), \text{EmpDept}(\text{Alice}, \text{D1}), \text{EmpDept}(\text{Bob}, \text{D2}), \text{DeptCity}(\text{D1}, \text{SJ}), \text{DeptCity}(\text{D2}, \text{SD})\}$ ,
- $J_0' = \{\text{Home}(\text{Alice}, \text{SF}), \text{Home}(\text{Bob}, \text{LA}), \text{EmpDept}(\text{Alice}, \text{D}), \text{EmpDept}(\text{Bob}, \text{D}), \text{DeptCity}(\text{D}, \text{SJ}), \text{DeptCity}(\text{D}, \text{SD})\}$ .



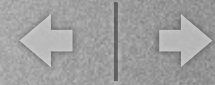
# Exercise I

- Say whether/why the next J is a solution for the schema mapping  $\Sigma: E(x,y) \rightarrow \exists z (H(x,z) \wedge H(z,y))$  with source instance  $I = \{E(a,b)\}$ 
  - 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:  $J5 = \{H(a,X), H(X,b), H(Y,Y)\}$



# Main issues in data exchange

- For a given source instance, there may be multiple target instances satisfying the specifications of the schema mapping.
- When more than one solution exist, which solutions are “better” than others?
- How do we compute a “best” solution?
- In other words, what is the “right” semantics of data exchange?



# Universal solutions in data exchange

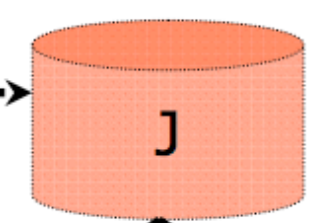
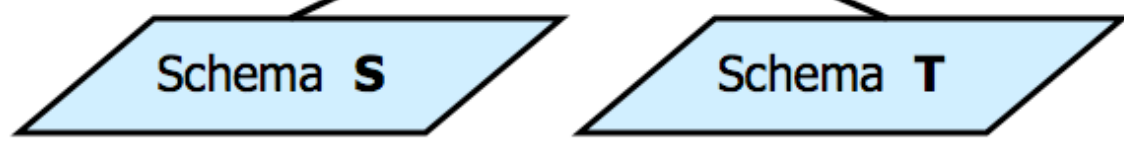
- We introduce the notion of universal solutions as the “best” solutions in data exchange.
- By definition, a solution is universal if it has homomorphisms to all other solutions (thus, it is a “most general” solution).
- **Constants**: entries in source instances;
- **Variables (labeled nulls)**: invented entries in target instances
- Homomorphism  $h: J_1 \rightarrow J_2$  between target instances:
  - $h(c) = c$ , for every constant  $c$  in  $J_1$
  - For every fact  $P(a_1, \dots, a_m)$  in  $J_1$ , then we have that  $P(h(a_1), \dots, h(a_m))$  is a fact in  $J_2$



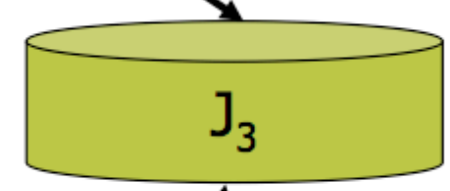
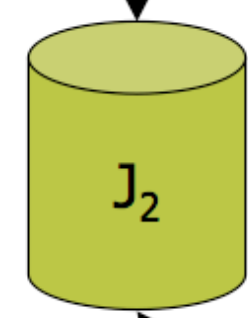
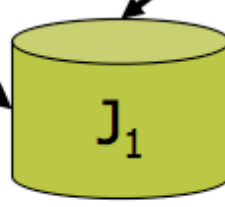


# Universal solutions in data exchange

$\Sigma$



Universal Solution



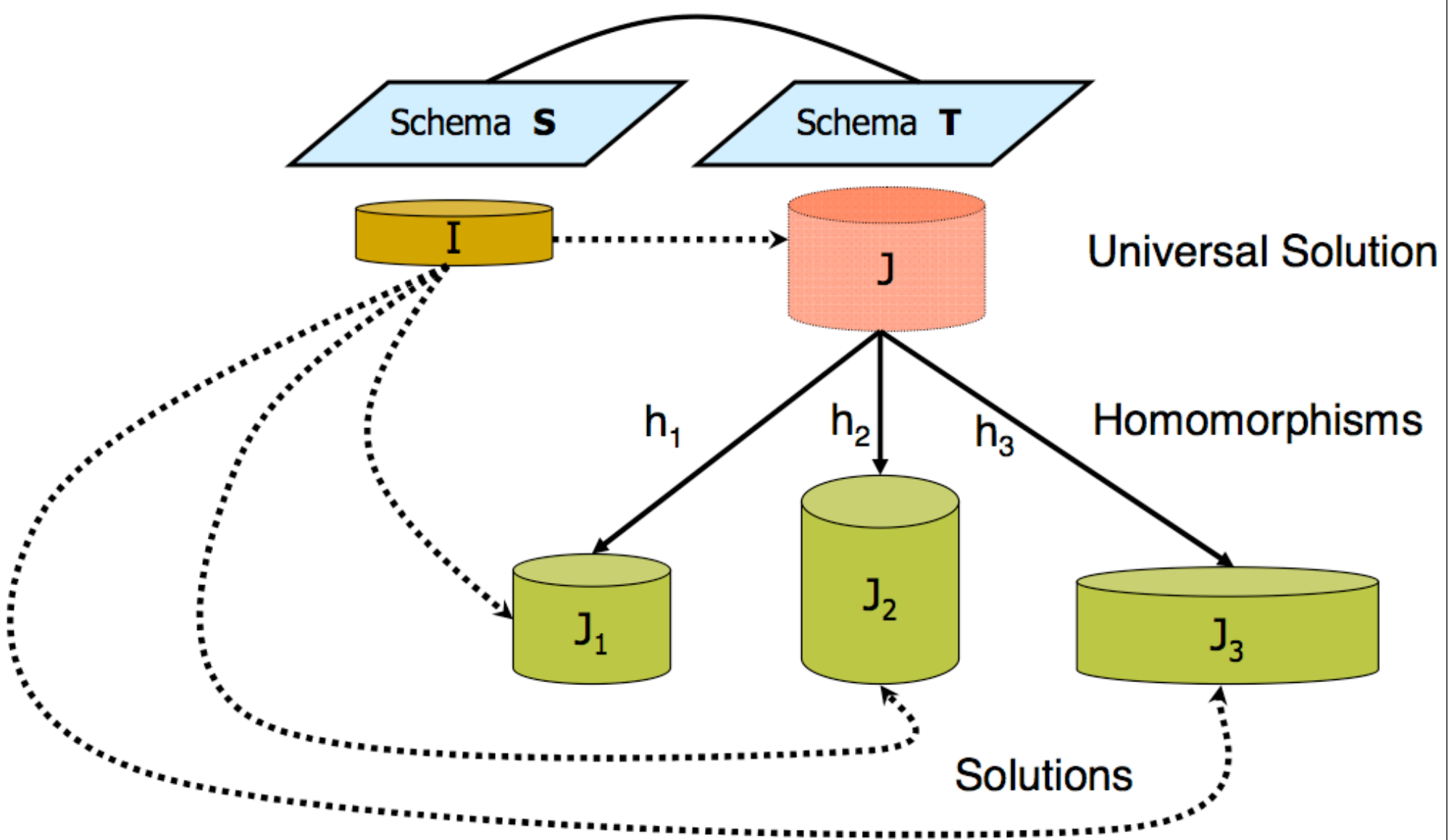
Homomorphisms

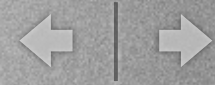
$h_1$

$h_2$

$h_3$

Solutions





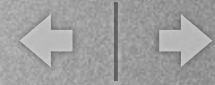
# Exercise I (cont'd)

- Say whether/why the next  $J$  is a 'universal' solution
  - Q1.6:  $J1 = \{H(a,b), H(b,b)\}$
  - Q1.7:  $J2 = \{H(a,a), H(a,b)\}$
  - Q1.8:  $J3 = \{H(a,X), H(X,b)\}$
  - Q1.9:  $J4 = \{H(a,X), H(X,b), H(a,Y), H(Y,b)\}$
  - Q1.10:  $J5 = \{H(a,X), H(X,b), H(Y,Y)\}$



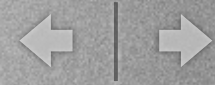
# Structural Properties of Universal Solutions

- Universal solutions are analogous to most general unifiers in logic programming.
- Uniqueness up to homomorphic equivalence: If  $J$  and  $J'$  are universal for  $I$ , then they are homomorphically equivalent.
- Representation of the entire space of solutions: Assume that  $J$  is universal for  $I$ , and  $J'$  is universal for  $I'$ . Then the following are equivalent:
  - $I$  and  $I'$  have the same space of solutions.
  - $J$  and  $J'$  are homomorphically equivalent.



## Existence-of-solutions problem

- Question: What can we say about the existence-of-solutions problem  $\text{Sol}(M)$  for a fixed schema mapping  $M = (S, T, \Sigma_{st}, \Sigma_t)$  specified by s-t tgds and target tgds and egds?
- Answer: Depending on the target constraints in  $\Sigma_t$ :
  - $\text{Sol}(M)$  can be trivial (solutions always exist).
  - $\text{Sol}(M)$  can be in PTIME
  - $\text{Sol}(M)$  can be undecidable.



# Existence-of-solutions problem

- Proposition: If  $M = (S, T, \Sigma_{st}, \Sigma_t)$  is a schema mapping such that  $\Sigma_t$  is a set of full target tgds, then:
  - Solutions always exist; hence,  $\text{Sol}(M)$  is trivial.
  - There is a Datalog program  $\pi$  over the target  $T$  that can be used to compute universal solutions as follows: Given a source instance  $I$ ,
    - Compute a universal solution  $J^*$  for  $I$  w.r.t. the schema mapping  $M^* = (S, T, \Sigma_{st})$  using the naïve chase algorithm.
    - Run the Datalog program  $\pi$  on  $J^*$  to obtain a universal solution  $J$  for  $I$  w.r.t.  $M$ .
  - Consequently, universal solutions can be computed in polynomial



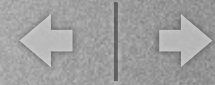
# The Chase Algorithm

- **Naïve Chase Algorithm** for  $M^* = (S, T, \Sigma_{st})$  : given a source instance  $I$ , build a target instance  $J^*$  that satisfies each s-t tgd in  $\Sigma_{st}$  by introducing new facts in  $J$  as dictated by the RHS of the s-t tgd and by introducing new values (variables) in  $J$  each time existential quantifiers need witnesses.
- Example:  $M = (S, T, \Sigma_{st}, \Sigma_t)$ 
  - $\Sigma_{st}: E(x,y) \rightarrow \exists z(F(x,z) \wedge F(z,y))$
  - $\Sigma_t: F(u,w) \wedge F(w,v) \rightarrow F(u,v)$
- The naïve chase returns a relation  $F^*$  obtained from  $E$  by adding a new node between every edge of  $E$ .
- The Datalog program  $\pi$  computes the transitive closure of  $F^*$ .



# Algorithmic properties of universal solutions

- Theorem (FKMP 2003): Schema mapping  $M = (S, T, \Sigma_{st}, \Sigma_t)$  such that:
  - $\Sigma_{st}$  is a set of source-to-target tgds;
  - $\Sigma_t$  is the union of a weakly acyclic set of target tgds with a set of target egds.
- Then: Universal solutions exist if and only if solutions exist.
- $\text{Sol}(M)$  is in PTIME.
- If a solution exists, then a universal solution can be produced in polynomial time using the chase procedure.



# Weakly acyclic sets of tgds

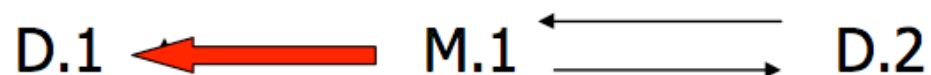
- Position graph of a set  $\Sigma$  of target tgds:
- Nodes:  $R.A$ , with  $R$  relation symbol,  $A$  attribute of  $R$
- Edges: for every  $\varphi(\mathbf{x}) \rightarrow \exists y \psi(\mathbf{x}, y)$  in  $\Sigma$ , for every  $x$  in  $\mathbf{x}$  occurring in  $\psi$ , for every occurrence of  $x$  in  $\varphi$  in position  $R.A$ :
  - For every occurrence of  $x$  in  $\psi$  in position  $S.B$ , add an edge  $R.A \rightarrow S.B$
  - In addition, for every existentially quantified  $y$  that occurs in  $\psi$  in position  $T.C$ , add a special edge  $R.A \rightarrow T.C$
- $\Sigma$  is weakly acyclic if the position graph has no cycle containing a special edge.
- A tgd  $\theta$  is weakly acyclic if so is the singleton set  $\{\theta\}$ .



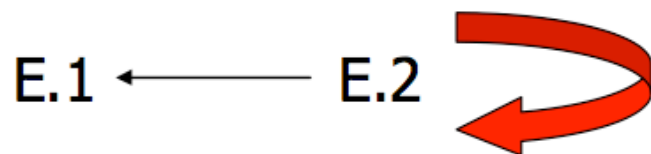


## Weakly acyclic sets of tgds: examples

- **Example 1:**  $\{ D(e,m) \rightarrow M(m), M(m) \rightarrow \exists e D(e,m) \}$  is weakly acyclic, but cyclic.



- **Example 2:**  $\{ E(x,y) \rightarrow \exists z E(y,z) \}$  is not weakly acyclic.





## Exercise 2



- Homework I
- (Q2.1) Build the dependency graph for the following schema mapping and say whether/why the set of tgds is weakly acyclic.

$$\Sigma_{st} = \{ \text{DeptEmp}(d, n, e) \rightarrow \exists M (\text{Dept}(d, M, n) \wedge \text{Emp}(e, d)) \},$$

$$\Sigma_t = \{ \text{Dept}(d, m, n) \rightarrow \exists D \text{Emp}(m, D), \\ \text{Emp}(e, d) \rightarrow \exists M \exists N \text{Dept}(d, M, N) \}.$$



## Exercise 2



- Homework I
- (Q2.2) Build the dependency graph for the following schema mapping and say whether/why the set of tgds is weakly acyclic

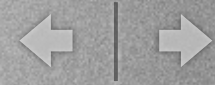
$$\Sigma_{st} = \{ \text{DeptEmp}(d, n, e) \rightarrow \exists M (\text{Dept}(d, M, n) \wedge \text{Emp}(e, d)) \},$$

$$\Sigma'_t = \{ \text{Dept}(d, m, n) \rightarrow \text{Emp}(m, d), \\ \text{Emp}(e, d) \rightarrow \exists M \exists N \text{Dept}(d, M, N) \}.$$



# Data Exchange with weakly acyclic sets of tgds

- Theorem: Schema mapping  $M = (S, T, \Sigma_{st}, \Sigma_t)$  such that:
  - $\Sigma_{st}$  is a set of source-to-target tgds;
  - $\Sigma_t$  is the union of a weakly acyclic set of target tgds with a set of target egds.
- There is an algorithm, based on the chase procedure, so that:
  - Given a source instance  $I$ , the algorithm determines if a solution for  $I$  exists; if so, it produces a universal solution for  $I$ .
  - The running time of the algorithm is polynomial in the size of  $I$ .
  - Hence, the existence-of-solutions problem  $Sol(M)$  for  $M$ , is in **PTIME**.



# Chase Procedure for Tgds and Egds

- Given a source instance  $I$ ,
- 1. Use the naïve chase to chase  $I$  with  $\Sigma_{st}$  and obtain a target instance  $J^*$ .
- 2. Chase  $J^*$  with the target tgds and the target egds in  $\Sigma_t$  to obtain a target instance  $J$  as follows:
  - 2.1. For target tgds introduce new facts in  $J$  as dictated by the RHS of the s-t tgd and introduce new values (variables) in  $J$  each time existential quantifiers need witnesses.
  - 2.2. For target egds  $\varphi(x) \rightarrow x_1 = x_2$ 
    - 2.2.1. If a variable is equated to a constant, replace the variable by that constant;
    - 2.2.2. If one variable is equated to another variable, replace one variable by the other variable.
    - 2.2.3 If one constant is equated to a different constant, stop and report “failure”.

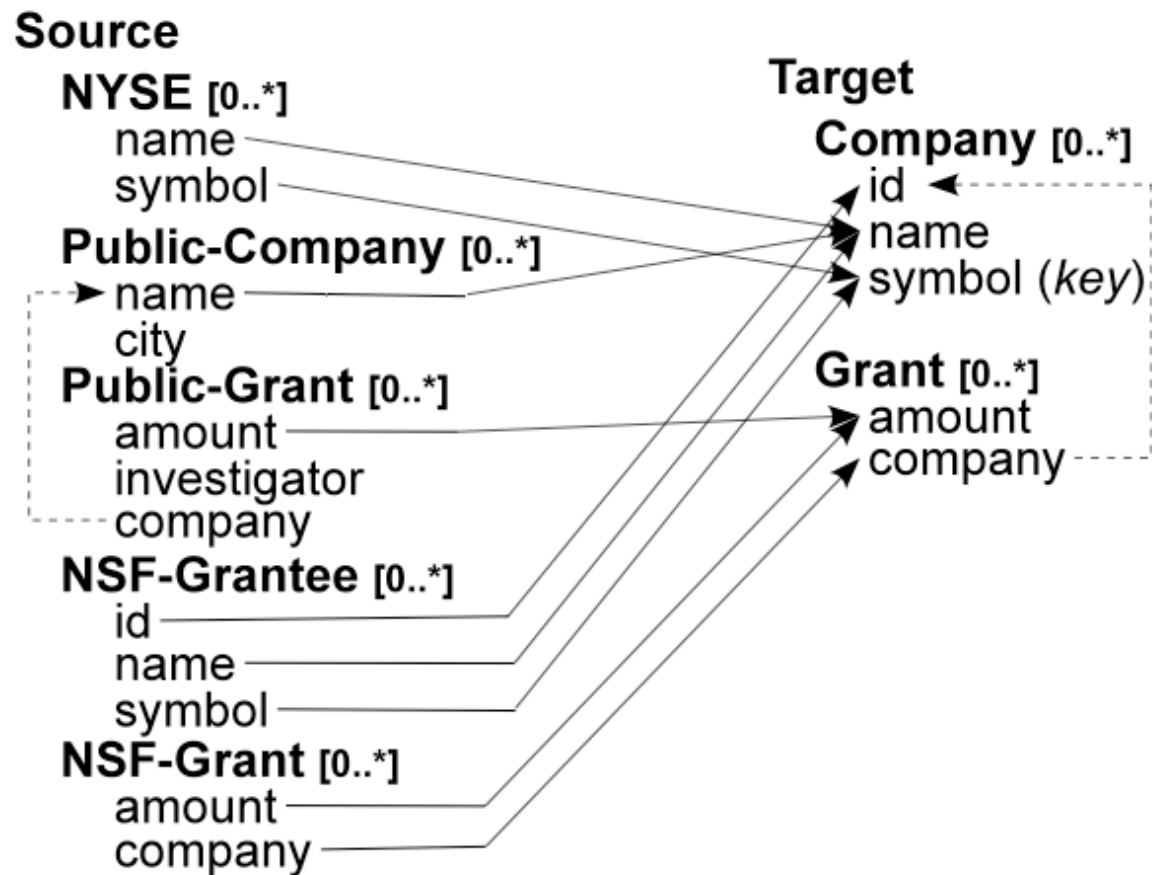


# The existence-of-solutions problem

- **Summary:** The existence-of-solutions problem
  - is undecidable for schema mappings in which the target dependencies are arbitrary tgds and egds;
  - is in PTIME for schema mappings in which the set of the target dependencies is the union of a weakly acyclic set of tgds and a set of egds.



- Assume you have the following S, T





# Exercise 3



- and the corresponding tgds:

## SOURCE-TO-TARGET TGDS

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

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

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

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

## TARGET TGDS

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

## TARGET EGDS

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





# Exercise 3



- Assume you have the following source instance I:

## NYSE

name	symbol
Google	GOOG
Yahoo!	YHOO

## Public-Company

name	city
Apple	Cup
Adobe	SJ

## Public-Grant

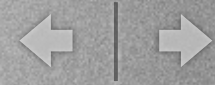
company	investigator	amount
Apple	Mike B.	25,000
Adobe	Anne C.	50,000

## NSF-Grantee

id	name	symbol
23	Yahoo!	YHOO
25	Adobe	ADBE

## NSF-Grant

company	amount
23	18,000
25	50,000



# Homework I - Exo 3

- Q3.1: Say whether the following solution is

## 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

a universal solution? a non-universal solution? other?



# Homework I - Exo 3

- Q3.2: Say whether the following solution is

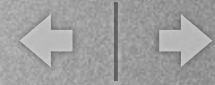
## Company

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

- a universal solution? a non-universal solution? other?



# Homework I - Exo 3

- Q3.3: Say whether the following solution is

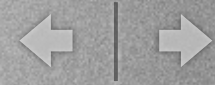
## Company

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

- a universal solution? a non-universal solution? other?



# Exercise 3

- Q3.4: Say whether the following solution is

## Company

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

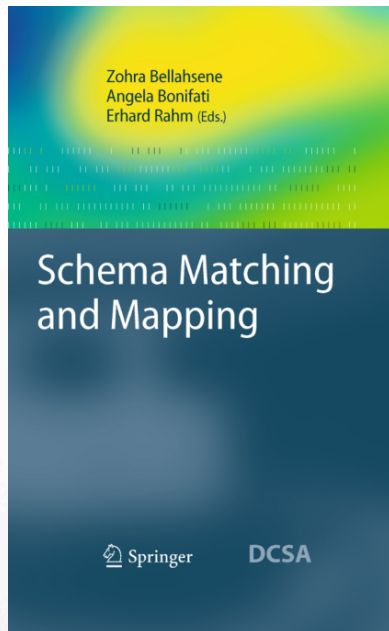
## Grant

amount	company
25,000	I1
18,000	I2
50,000	25
80,000	N1

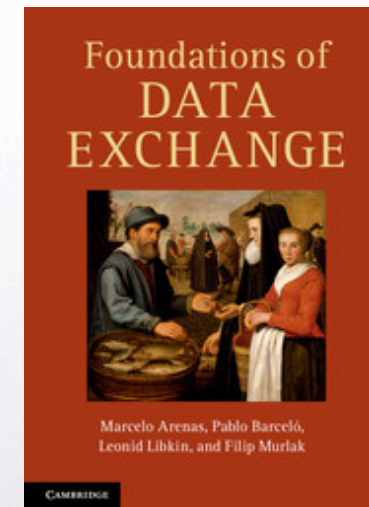
- a universal solution? a non-universal solution? other?



# Books



[Bellahsene et al. 2011] Z. Bellahsene, A. Bonifati and E. Rahm, "*Schema Matching and Mapping*", Springer-Verlag, 2011

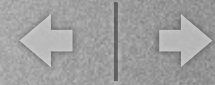


[Arenas et al. 2014] M. Arenas et al., "*Foundations of Data Exchange*", Cambridge University Press, 2014



# List of References

- [Batini et al. 1986] Batini C, Lenzerini M, Navathe SB (1986) A Comparative Analysis of Methodologies for Database Schema Integration. ACM Comp. Surv. 18(4):323-364
- [Lenzerini 2002] Lenzerini Maurizio “Data Integration: A Theoretical Perspective”. In: PODS, pp 233-246
- [Fagin et al. 2005] Fagin R, Kolaitis PG, Miller RJ, Popa L (2005) Data exchange: semantics and query answering. Theoretical Computer Science 336(1):89-124
- [Lerner 2000] Lerner BS (2000) A Model for Compound Type Changes Encountered in Schema Evolution. TPCTC 25(1):83–127
- Extra slides:
  - A. Bonifati and Y. Velegrakis tutorial at EDBT 2011 <http://www.lifl.fr/~bonifati/pubs/EDBT11MappingBenchmarkTutorial.pdf>



# List of References

- [Batini et al. 1986] Batini C, Lenzerini M, Navathe SB (1986) A Comparative Analysis of Methodologies for Database Schema Integration. ACM Comp. Surv. 18(4):323-364
- [Lenzerini 2002] Lenzerini Maurizio “Data Integration: A Theoretical Perspective”. In: PODS, pp 233-246
- [Fagin et al. 2005] Fagin R, Kolaitis PG, Miller RJ, Popa L (2005) Data exchange: semantics and query answering. Theoretical Computer Science 336(1):89-124
- [Lerner 2000] Lerner BS (2000) A Model for Compound Type Changes Encountered in Schema Evolution. TPCTC 25(1):83–127
- Slides:
  - P. Kolaitis talk at PODS 2005 <http://users.soe.ucsc.edu/~kolaitis/talks/datexch-pods05.pdf>
  - A. Bonifati and Y. Velegrakis tutorial at EDBT 2011 <http://www.lifl.fr/~bonifati/pubs/EDBT11MappingBenchmarkTutorial.pdf>