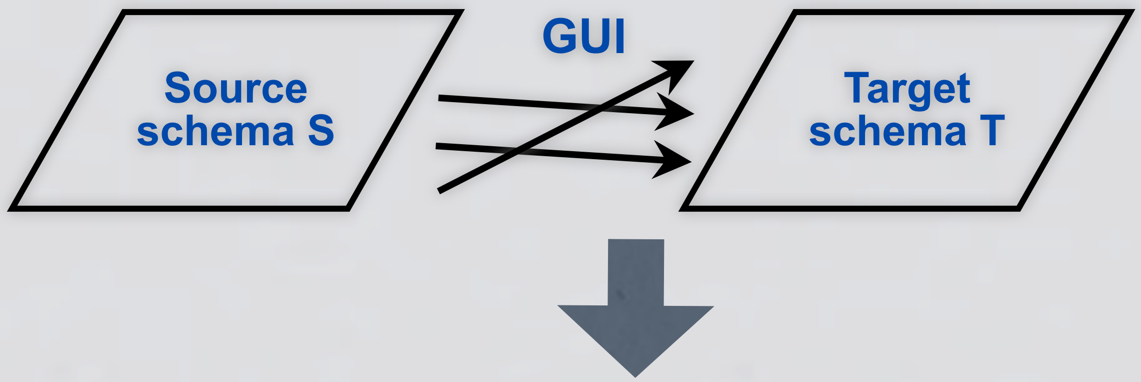
Emerging Applications for Schema Mappings

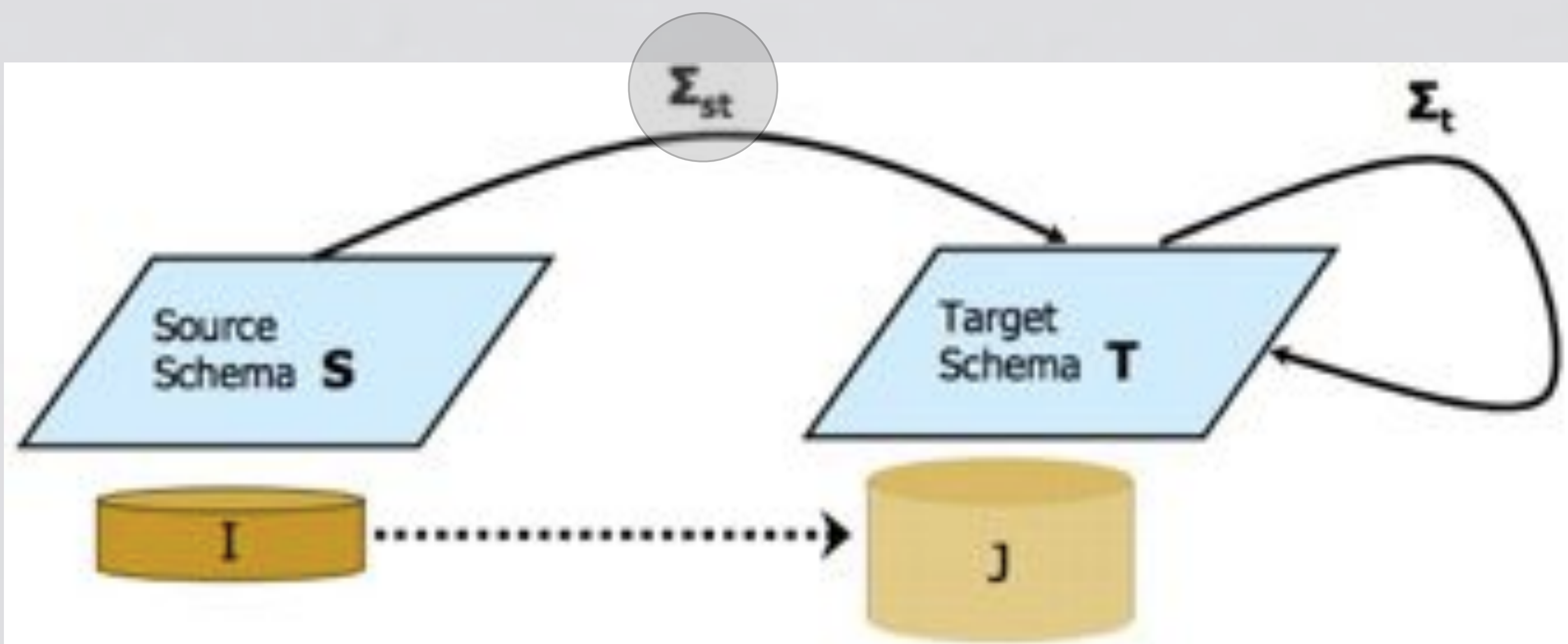
Paolo Papotti - Università Roma Tre

Maratea - SEBD 2011

A principled approach to information integration

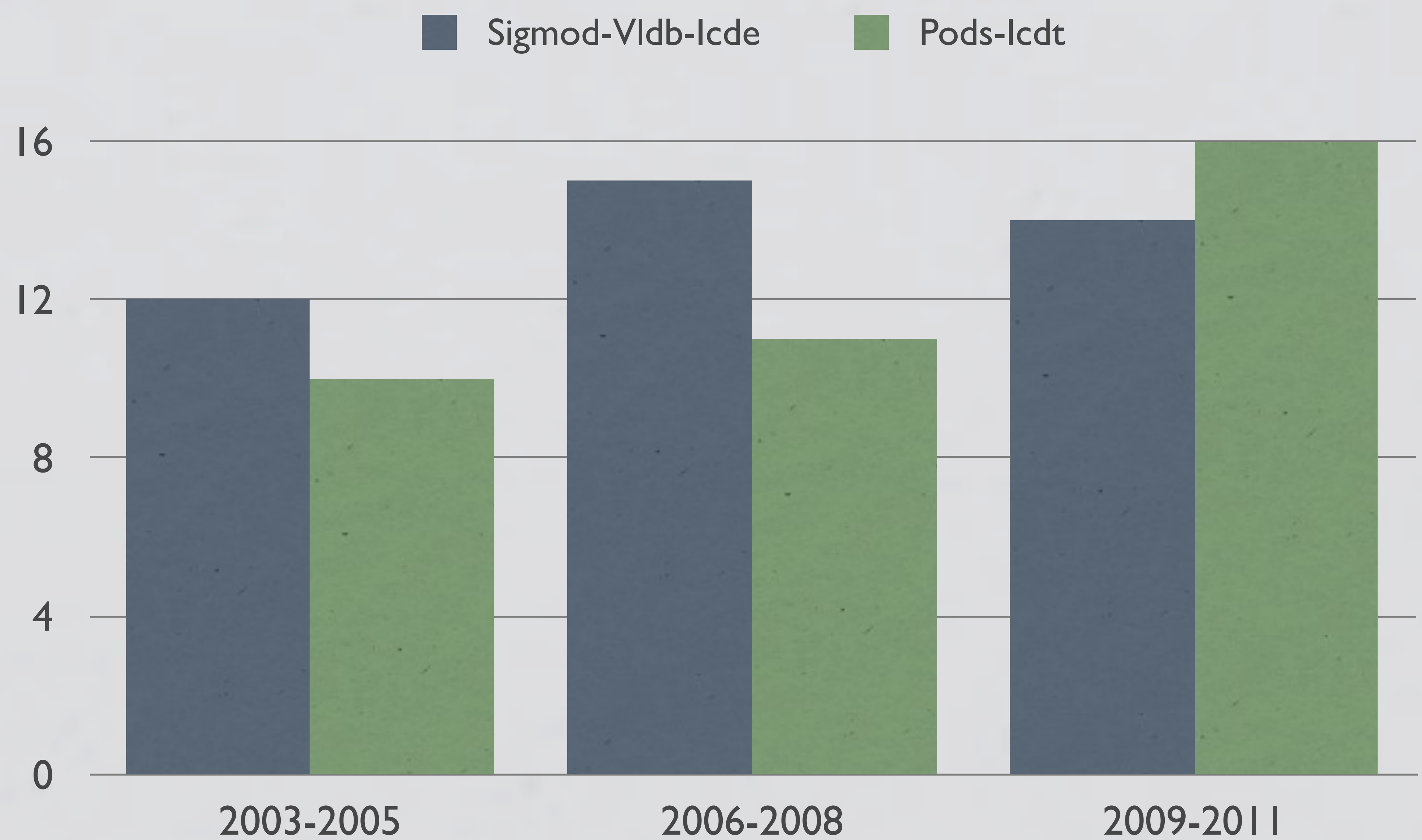


[Popa et al. Vldb'02]



[Fagin et al. Icdt'03]

Many good results



Something went wrong…

Notable expection: IBM InfoSphere Data Architect

...but what?

1. quality of the solutions produced by mapping systems
2. limited number of application scenarios
3. no schema mapping tools available to the community

http://www.flickr.com/photos/padesig/193865429/

This tutorial

- Goals
  - introduce recent advances in schema mappings and show how they can positively impact several data management problems
- Credits
  - feedback and comments from Gianni Mecca, Lucian Popa, and Mauricio Hernandez

Outline

- Background
  - schema mappings, data exchange
- Recent results
  - quality of solutions, larger class of scenarios, new tools available
- Emerging applications
  - data fusion, data cleaning, schema evolution, ETL

BACKGROUND

http://www.flickr.com/photos/doug88888/4492332051/

Moving Data

- Many possible approaches: procedural code (es: Java program), ad hoc script, ETL ...
- We need a more principled way to do that [Bernstein, Sigmod'07][Haas, Icdt'07]
- We want higher level of abstraction that makes it possible to <u>separate the design of the relationship between schemas from its implementation</u>
- A clean way to do it: by using logic

Schema mappings?

- High-level, declarative assertions that specify the relationship between two database schemas
- Building blocks in formalizing and studying data interoperability tasks, including data integration and data exchange
- Schema mappings help with the development of practical tools:
  - can be generated and managed automatically
  - can be compiled into SQL/XSLT/XQuery/... scripts automatically

Which language?

- Schema mappings should be
  - **expressive** enough to specify data interoperability tasks
  - **simple** enough to be efficiently manipulated by tools

- There is a tension between the two: increase in expressive power comes at the expense of efficiency

- <u>Unrestricted use of first-order logic</u> as a schema mapping specification language gives rise to <u>undecidability</u> of basic algorithmic problems about schema mappings

Example: Books

Source DB1: Internet Book Database

IBDBook [0..*]

title

title
The Hobbit
The Da Vinci Code
The Lord of the Rings

Source DB2: Library of Congress

LOC [0..*]

title

publisher

title	publisher
The Lord of the Rings	Houghton
The Catcher in the Rye	Lb Books

Source DB3: Internet Book List

IBLBook [0..*]

title

publisherId

title	pubId
The Hobbit	245
The Catcher in the Rye	776

IBLPublisher [0..*]

id

name

id	name
245	Ballantine
776	Lb Books

Target DB

Book [0..*]

title

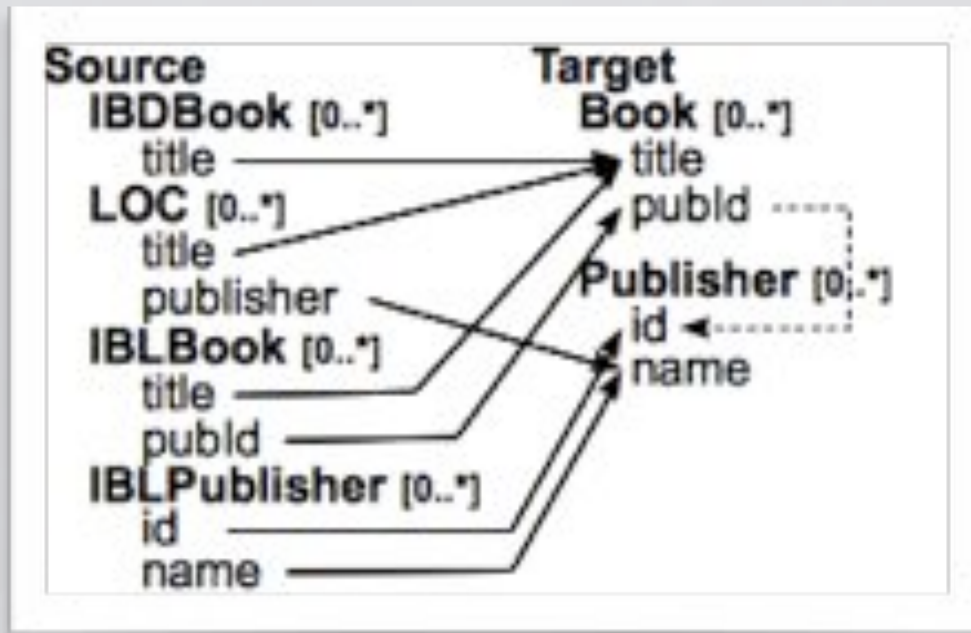
pubId

Publisher [0..*]

id

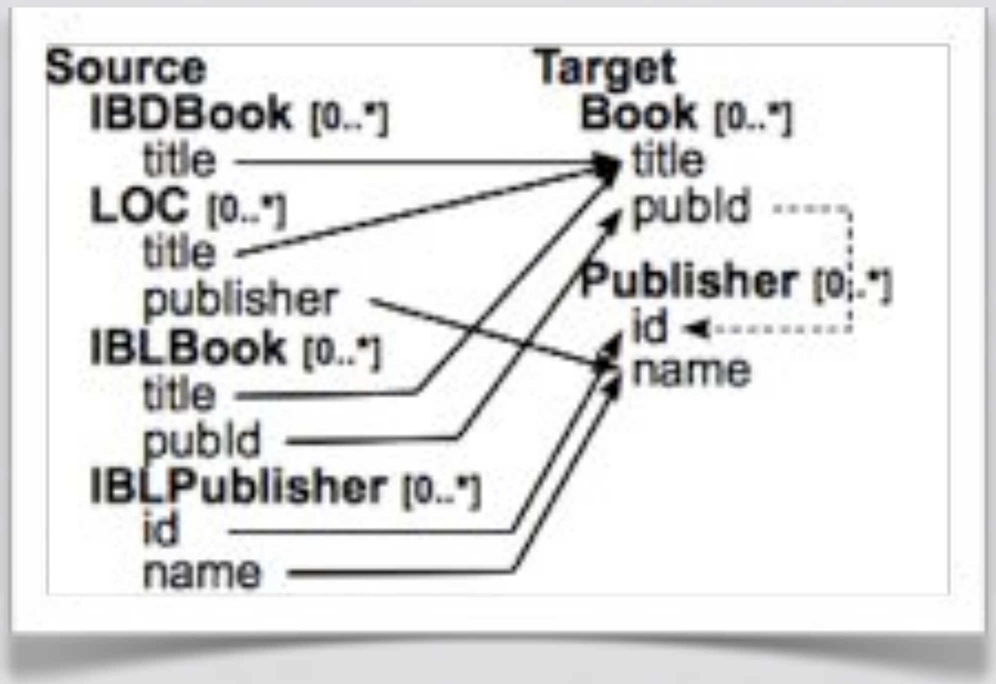
name

Source-to-target TGDs (Σ_{st})



- $m_1 : \forall\, t, i : IBLBook\,(t, i) \rightarrow Book(t, i)$
- $m_2 : \forall\, i, p : IBLPublisher(i, p) \rightarrow Publisher\,(i, p)$
- the result of a CQ over the source is contained in the result of a CQ over the target

Source-to-target TGDs (Σ_{st})



$m_3 : \forall\ t: \mathrm{IBDBook}(t) \rightarrow \exists N: \mathrm{Book}(t, N)$

$m_4 : \forall\ t, p: \mathrm{LOC}(t, p) \rightarrow \exists N: \mathrm{Book}(t, N)\ \mathrm{Publisher}\ (N,p)$

Labeled Nulls

- Labeled nulls N_1 N_2 … N_k handle existentially quantified variables
  - variables in the target instance to satisfy existential quantifiers
  - some are pure nulls, others correlate tuples
    e.g., LOC(t, p) → ∃N0: Book(t, N0) Publisher (N0, p)
- In practice, can be generated using Skolem functions
  - N0: sk(Book(A:**t**),Publisher(B:**p**))

Mapping language desiderata

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

How to model schema constraints (Σ_t)?

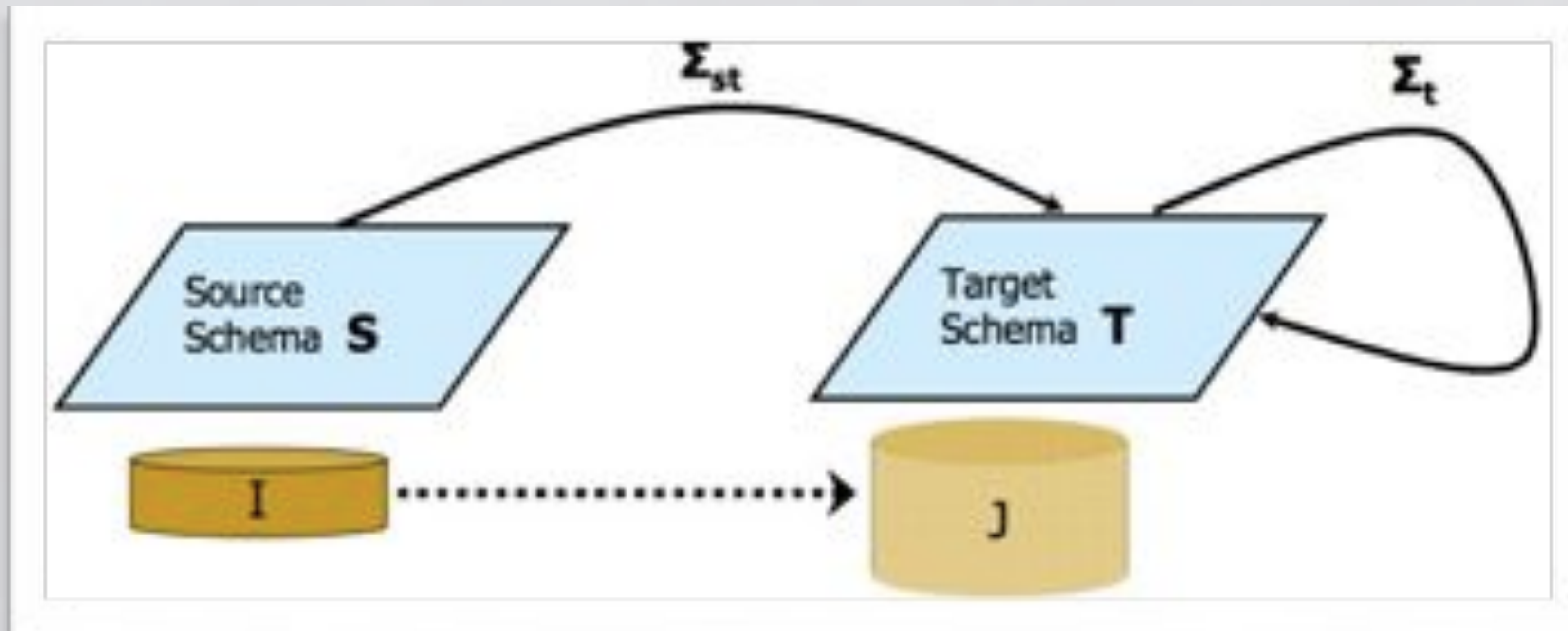
- **Target TGD**
  enforce inclusion constraints on the target schema. E.g.,

$$\forall\, t, i : Book(t, i) \rightarrow \exists N: \; Publisher(i, N)$$

- **Target EGD**
  enforce functional-dependencies on the target schema. E.g.,

$$\forall\, t, i, i' : Book(t, i), Book(t, i') \rightarrow (i = i')$$

Data Exchange



- DE Scenario: schema mapping M = <S, T, Σ_{st}, Σ_t >
- DE Problem: given M and I, generate J s.t. I and J satisfy the constraints in Σst and J satisfies the constraints in Σt
- Intuition: constraints are not used to check properties, but to generate or modify tuples

What are the problems?

- Problem 1: **To Generate Solutions**
  - given a data exchange scenario and a source instance, generate a solution (target instance)
- Problem 2: **To Generate the TGDs**
  - users are not willing to write down logical formulas
  - it is much more natural to provide a minimal, high-level specification of the mapping

Data Exchange Solutions

- A key observation: dependencies do not fully specify the solution (i.e., a scenario may have many solutions)
- Example: $\forall\, x : R(x) \rightarrow \exists\, y: S(x, y)$ with $I = \{ R(a) \}$
  - $J_0 = \{ S(a, N_1) \}$, $J_1 = \{ S(a, b) \}$, $J_2 = \{ S(a, N_1), T(c, d) \}$
- When more than one solution exist, which solutions are "better" than others?
  How do we compute the "best" solution?

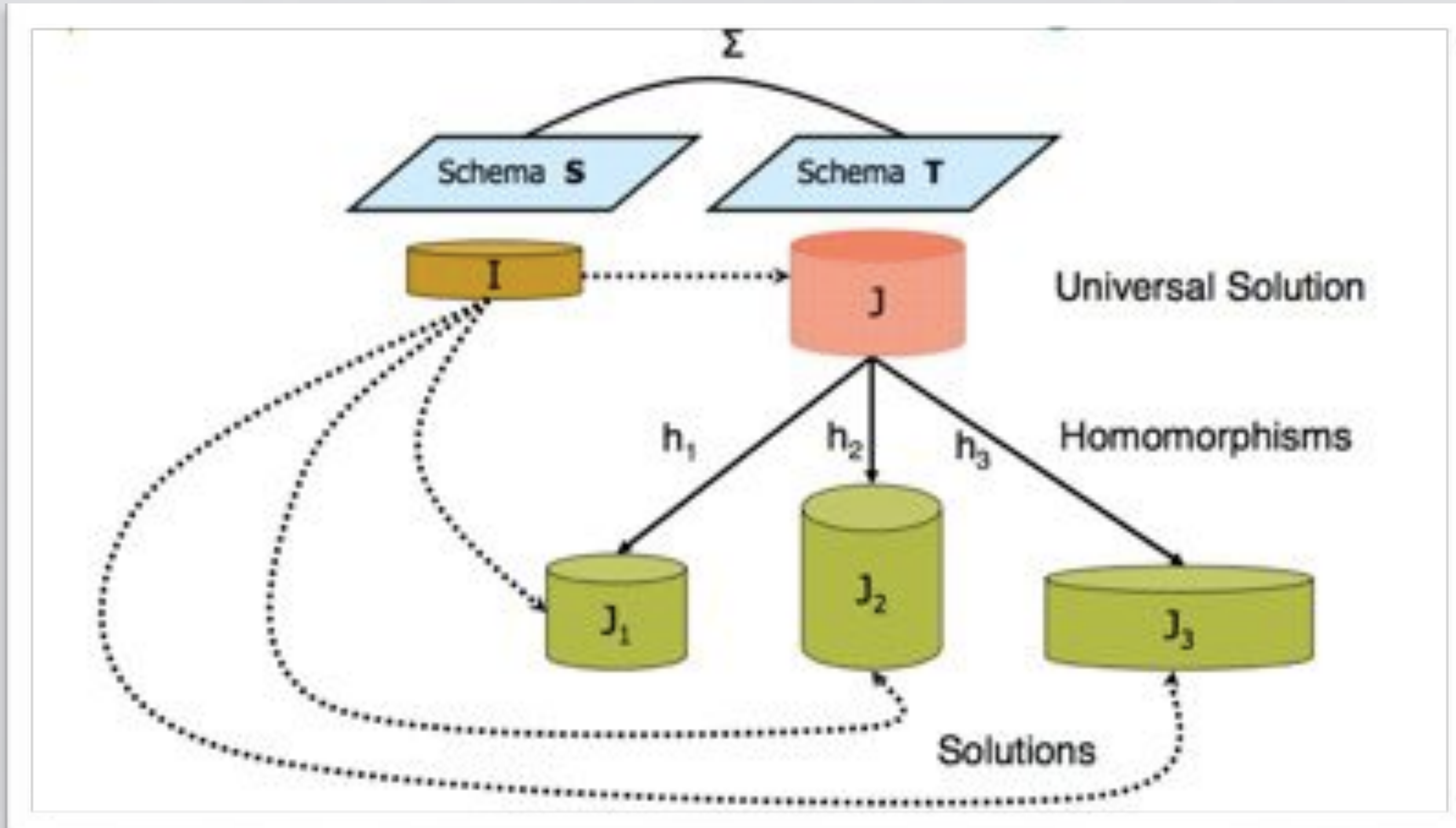
Homomorphisms

- A constant-preserving mapping of values
- Examples:
  1. from $J_0 = \{ S(a, N_1) \}$ to $J_1 = \{ S(a, b) \}$
  2. from $J_3 = \{ S(a, N_1), S(a, N_3) \}$ to $J_2 = \{ S(a, N_2) \}$
  3. from $J_4 = \{ S(a, N_1), S(b, N_1) \}$ to $J_5 = \{ S(a, b), S(b, c) \}$ X

Universal Solutions

* A good solution
  - contains sufficient information to satisfy the tgds
  - does not contain any extra information
  - unique up to homomorphic equivalence
* Universal solution [Fagin et al. Icdt'03]
  * a target instance J that is a solution for I and such that, for any other solution J' for I, there exists an homomorphism **h** : J → J'
* Examples: J_0 is universal, while J_1 and J_2 are not
  * $J_0 = \{ S(a, N_1) \}, J_1 = \{ S(a, b) \}, J_2 = \{ S(a, N_1), T(c, d) \}$

Universal Solutions



Good news

- [Fagin et al. Icdt'03] Given a schema mapping M s.t.:
  - Σst is a set of source-to-target tgds,
  - Σt is the union of a **weakly acyclic** set of target tgds with a set of target egds
- A canonical **universal** solution (if solutions exist) can be produced in **polynomial** time using the **chase** procedure
- the chase can be implemented in SQL + Skolem functions to generate nulls: **efficiency and portability**

Scripts generate universal solutions!

$m_1 : \forall\, t,i : \text{IBLBook}(t, i) \rightarrow \text{Book}(t, i)$

$m_2 : \forall\, i,p : \text{IBLPublisher}(i, p) \rightarrow \text{Publisher}(i, p)$

$m_3 : \forall\, t: \text{IBDBook}(t) \rightarrow \exists N: \text{Book}(t, N)$

$m_4 : \forall\, t, p: \text{LOC}(t, p) \rightarrow \exists N1: \text{Book}(t, N1)\ \text{Publisher}(N1, p)$

IBDBook

title
The Hobbit
The Da Vinci Code
The Lord of the Rings

LOC

title	publisher
The Lord of the Rings	Houghton
The Catcher in the Rye	Lb Books

IBLBook

Title	pubId
The Hobbit	245
The Catcher in the Rye	901

IBLPublisher

id	name
245	Ballantine
901	Lb Books

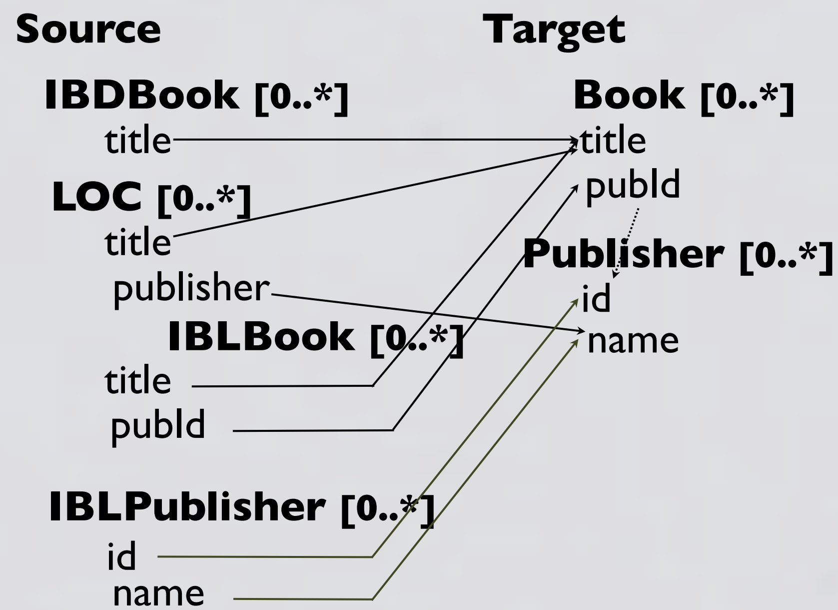
Target: Book

title	pubId
The Hobbit	NULL
The Da Vinci Code	NULL
The Lord of the Rings	NULL
The Lord of the Rings	I1
The Catcher in the Rye	I2
The Hobbit	245
The Catcher in the Rye	901

Target: Publisher

id	name
I1	Houghton
I2	Lb Books
245	Ballantine
901	Lb Books

But who wants to handwrite tgds?



IBDBoo

titlek
The Hobbit
The Da Vinci Code
The Lord of the Rings

LOC

title	publisher
The Lord of the Rings	Houghton
The Catcher in the Rye	Lb Books

IBLBook

Title	pubId
The Hobbit	245
The Catcher in the Rye	901

IBLPublisher

id	name
245	Ballantine
901	Lb Books

Target: Book

title	pubId
The Hobbit	NULL
The Da Vinci Code	NULL
The Lord of the Rings	NULL
The Lord of the Rings	I1
The Catcher in the Rye	I2
The Hobbit	245
The Catcher in the Rye	901

Target: Publisher

id	name
I1	Houghton
I2	Lb Books
245	Ballantine
901	Lb Books

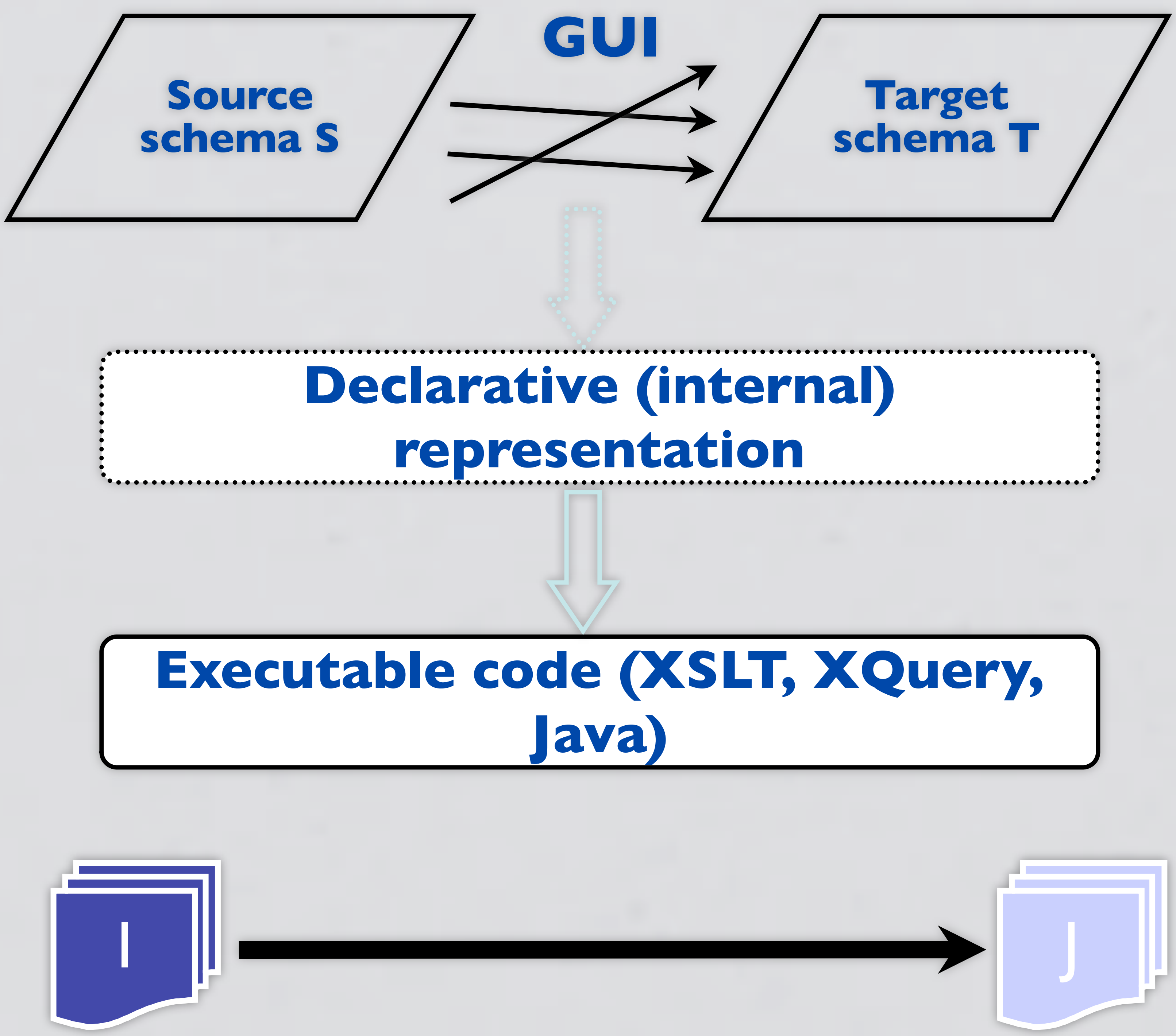
Problem 2: To Generate the TGDs

- A schema mapping system
  - takes as input an abstract specification of the mapping under the form of value correspondences among schema elements
  - generates the tgds and then executable transformations (SQL, XQuery, XSLT…) to run them

  - **notice**: schemas can be nested - can have FK constraints

- Problem 3: **Gathering Correspondences**
  - users may visually specify them as lines
  - or they may be suggested by a **schema matching** tool [Bernstein&Rahm, VLDBJ'01]

Mapping systems



IBM Clio [Vldb'02], +Spicy [Sigmod'09], Heptox [VldbJ'10]

MS ADO.net

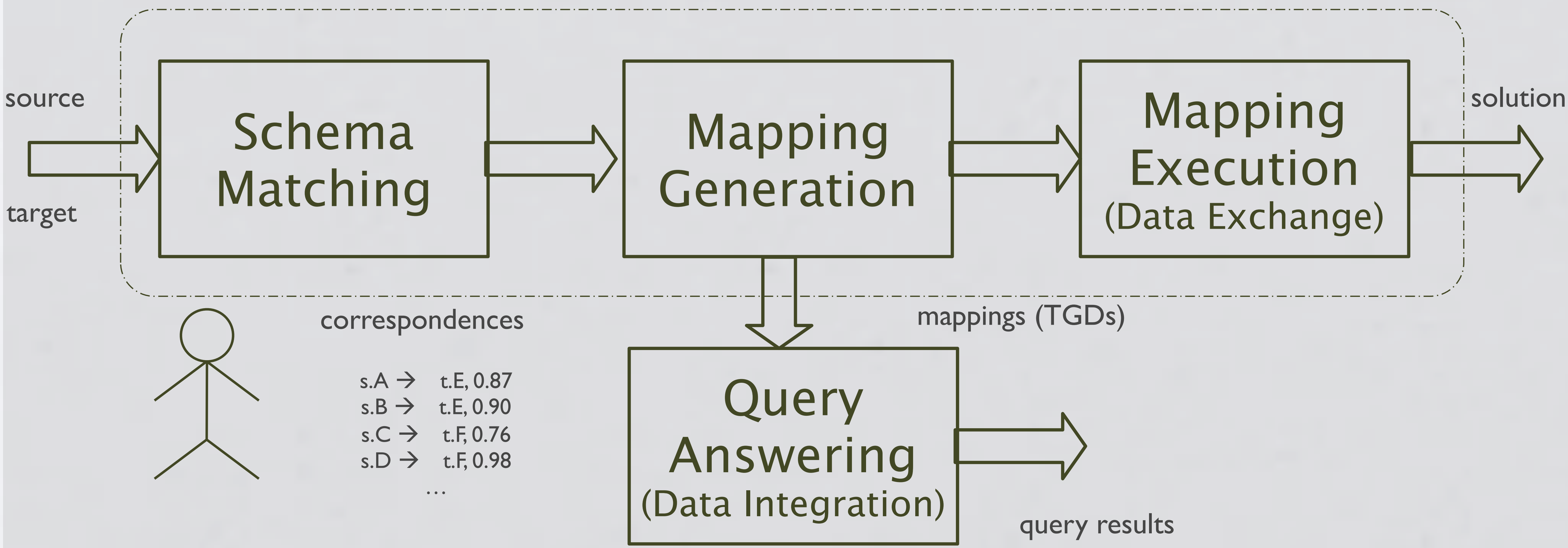
Altova MapForce

StylusStudio

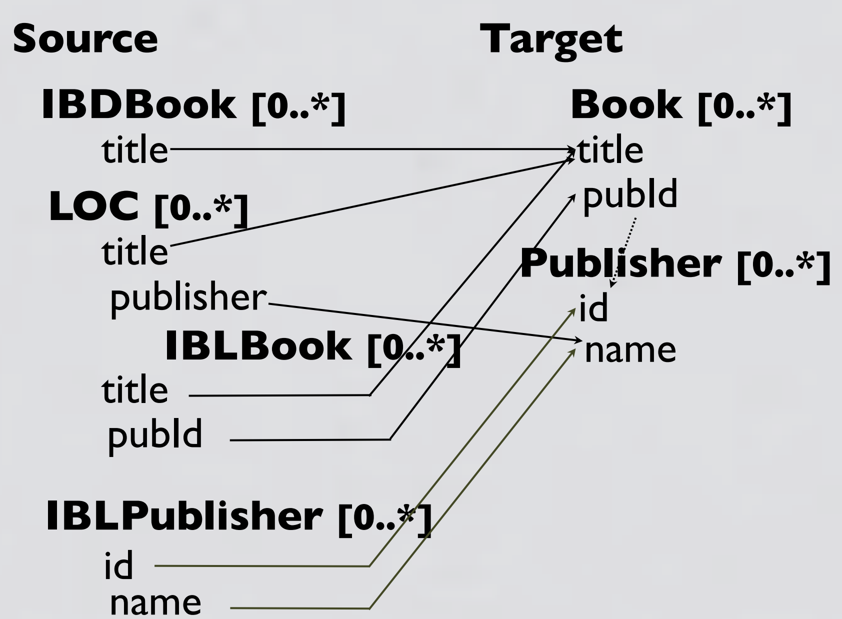
BEA Aqualogic

Data exchange

A reference architecure



Problems 1 and 2 are "solved"



m_1 : $\forall$ t,i : IBLBook (t, i) → Book(t, i)

m_2 : $\forall$ i,p : IBLPublisher(i, p) → Publisher (i, p)

m_3 : $\forall$ t: IBDBook(t) → $\exists$N: Book(t, N)

m_4 : $\forall$ t, p: LOC(t, p) → $\exists$N1: Book(t, N1) Publisher (N1, p)

IBDBoo

titlek
The Hobbit
The Da Vinci Code
The Lord of the Rings

LOC

title	publisher
The Lord of the Rings	Houghton
The Catcher in the Rye	Lb Books

IBLBook

Title	pubId
The Hobbit	245
The Catcher in the Rye	901

IBLPublisher

id	name
245	Ballantine
901	Lb Books

Target: Book

title	pubId
The Hobbit	NULL
The Da Vinci Code	NULL
The Lord of the Rings	NULL
The Lord of the Rings	I1
The Catcher in the Rye	I2
The Hobbit	245
The Catcher in the Rye	901

Target: Publisher

id	name
I1	Houghton
I2	Lb Books
245	Ballantine
901	Lb Books

1. Source semantics preserved in the target instance
2. Given a minimal abstract specification

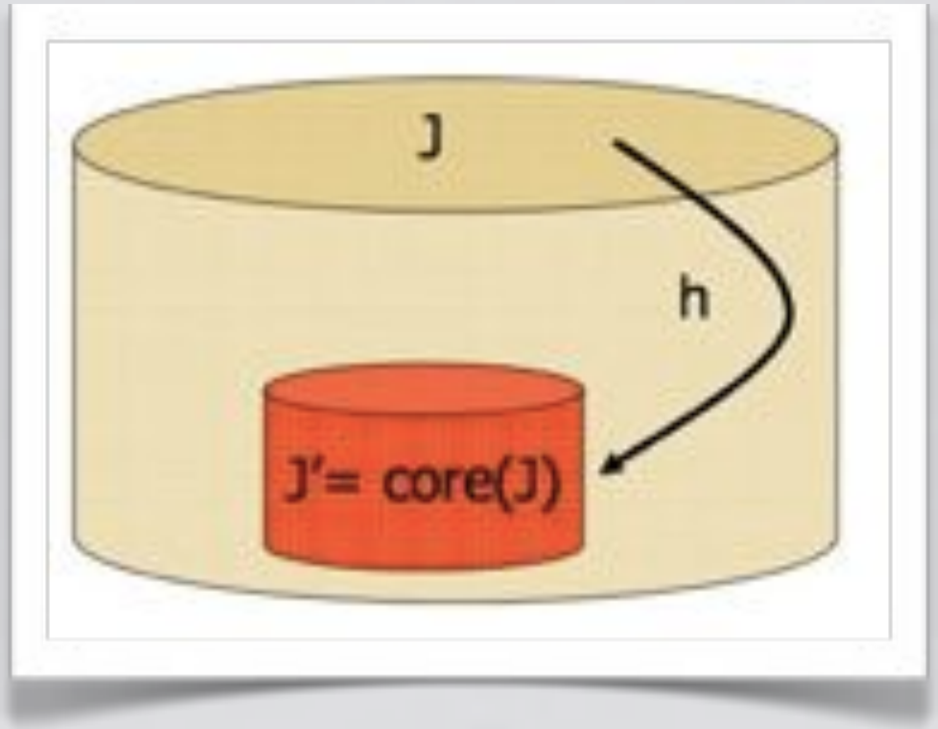
but looking at the target instance…

Redundancy

- What does redundancy mean?
  information that also appears elsewhere in the same instance
  - es: **Book**('The Hobbit', 245), **Book**('The Hobbit', N_1)
- A nice way to characterize this
  - any tuple **t,** such that there exists a tuple **t'** and an homomorphism **h** : t → t', is **redundant**
  - intuition: t' contains at least the same information
- Minimizing solutions: removing redundancy

The Core: smallest universal solution

- The **core** [Fagin et al, Pods'03]
  - originates in graph theory, exists for all finite structures
  - does not contain any proper subset that is also a universal solution
  - is unique (up to the renaming of nulls)

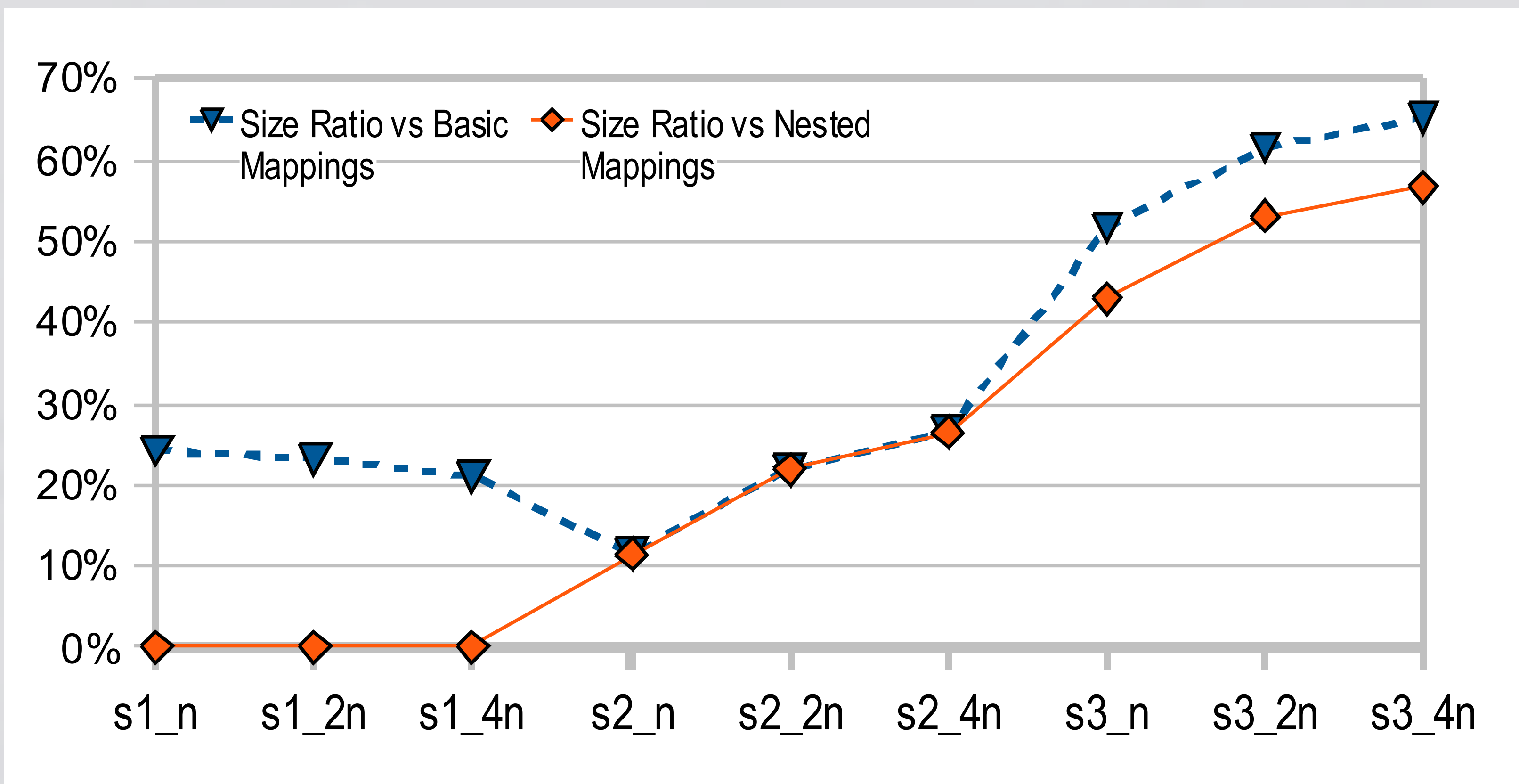


- clear notion of quality
  quality = minimality

Comparing Solutions

- First generation mapping systems generate core solutions only in special cases

- How far do they go from the core in general? [Mecca et al. Sigmod'09]



Getting to the Core (+)

- How much does it cost to find the core?
  - CORE-Identification for arbitrary instances is NP-hard
  - But for universal solutions, it is a polynomial problem [Fagin et al, Pods'03] [Gottlob&Nash, JACM'08]
- Intuition of the algorithm:
  - post-process the initial solution, look exhaustively for endomorphisms, and progressively remove nulls

Getting to the Core (-)

- The problem is "solved" for a very general settings (**weakly acyclic tgds + egds**)
- But in practice…
  - a simple scenario with 4 tables and 4 tgds
  - a small instance with 5000 source tuples
  - generating the canonical solution takes 1 sec
  - **computing the core takes 8 hours** *

* using an implementation [Pichler, Savenkov, LPAR 2008] of the algorithm in [Gottlob, Nash, JACM 2008] running on PostgreSQL

What went wrong?

1. quality of the solutions produced by mapping systems
   - core is good, but post processing **do not scale**
   - mapping systems scale, but produce **only univ. sol.**
2. limited number of application scenarios
   - **egd** are needed, but not supported by mapping systems
3. no schema mapping tools available

RECENT RESULTS

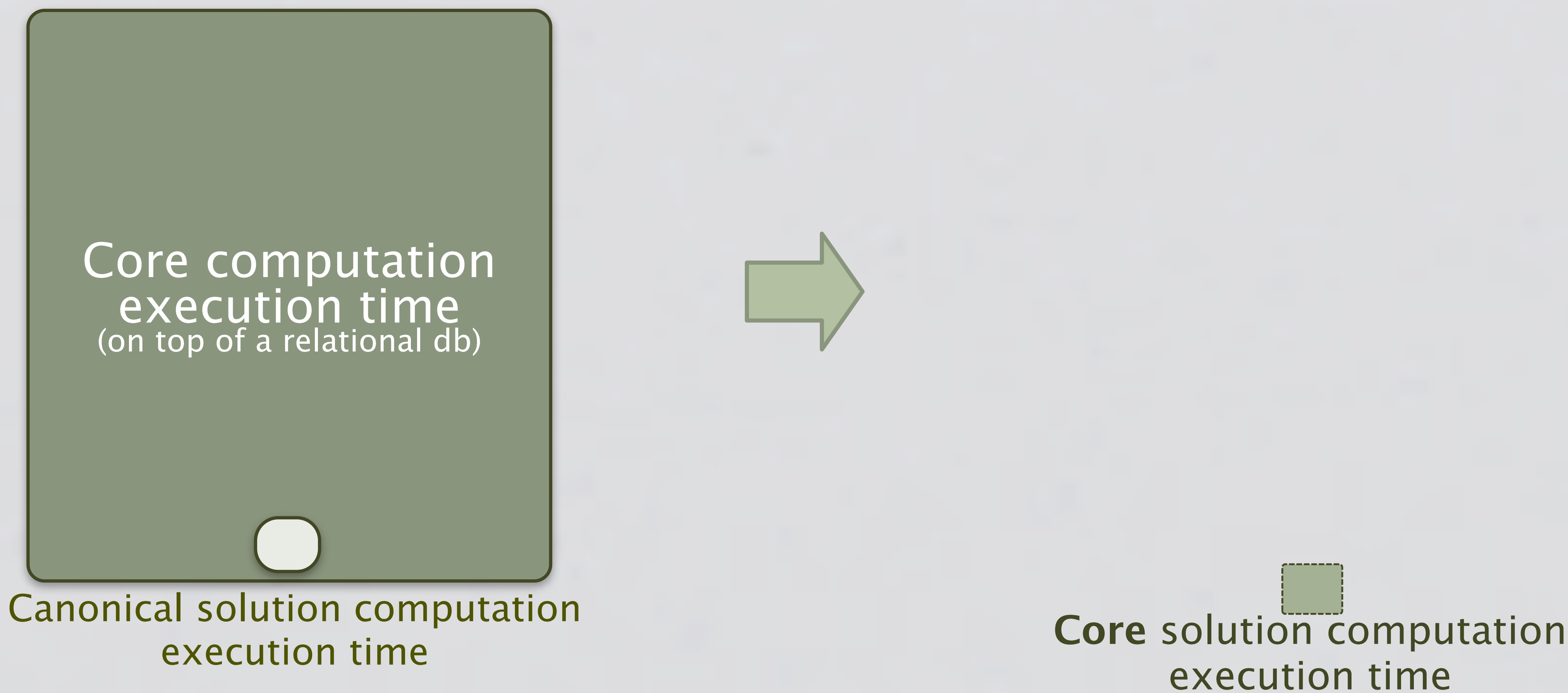
http://www.flickr.com/photos/mmpip/5858542140/

Outline

* Improving the quality of mapping systems solutions
* Enlarging the class of application scenarios
  * egds
  * optimization
  * mappings as operators
  * ...
* New tools available

Getting to the core (!)

- Can we compute a core solution using an executable script (e.g. SQL)?
- Advantages: efficiency, modularity, reuse



Good news: it is possible
(under proper restrictions)

- Given a mapping scenario M = <S, T, Σ_{st}>
- Generate a new scenario M' = <S, T, Σ'_{st}>
  - such that, for any source instance I, chasing Σ'_{st} yields **core solutions** for I under M
- Two independent algorithms that rewrite the original s-t tgds into core/laconic s-t tgds
  [Mecca et al. Sigmod'09] [ten Cate et al. Vldb'09]

The Key Intuition

- to prevent the generation of redundancy, i.e., of homomorphisms at tuple level
  - e.g., **Book**('The Hobbit', N_1) vs **Book**('The Hobbit', 245)
- look at tgd conclusions (i.e., structures of facts in the target) to identify **homorphisms at the formula level**
- and rewrite the tgds accordingly

Formula Homomorphism

- Mapping among variable occurrences that maps universal occurrences into universal occurrences and preserves tgd conclusions

- Example

  $m_1 : \forall\, t,i : IBLBook\,(t, i) \rightarrow Book(t, i)$

  $m_3 : \forall\, t': IBDBook(t') \rightarrow \exists N': Book(t', N')$

  $h: Book(t', N') \rightarrow Book(t, i)$

  $h(t') \rightarrow t$

  $h(N) \rightarrow i$

Tgd Rewriting Strategy

- For each tgd **m** such that there is a formula hom. into **m'**
  - fire m', the "more informative" mapping; then fire m only when m' does not fire for the same values
- In practice: negation in tgd premises, i.e., differences in the scripts
- Example:
  - $IBDBook(t') \wedge \neg(IBLBook(t, i) \wedge t = t') \rightarrow \exists N': Book(t', N')$

Experimental Results

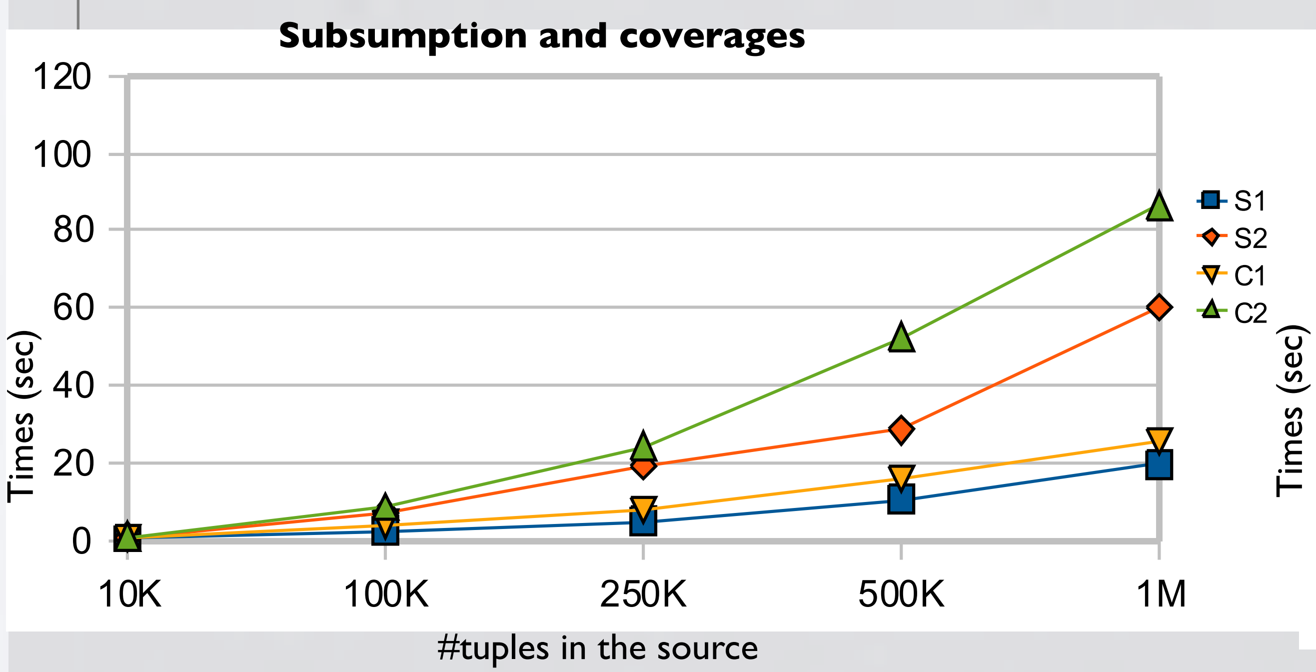
- Algorithms for Core schema mappings implemented in +**Spicy**
  http://www.db.unibas.it/projects/spicy/

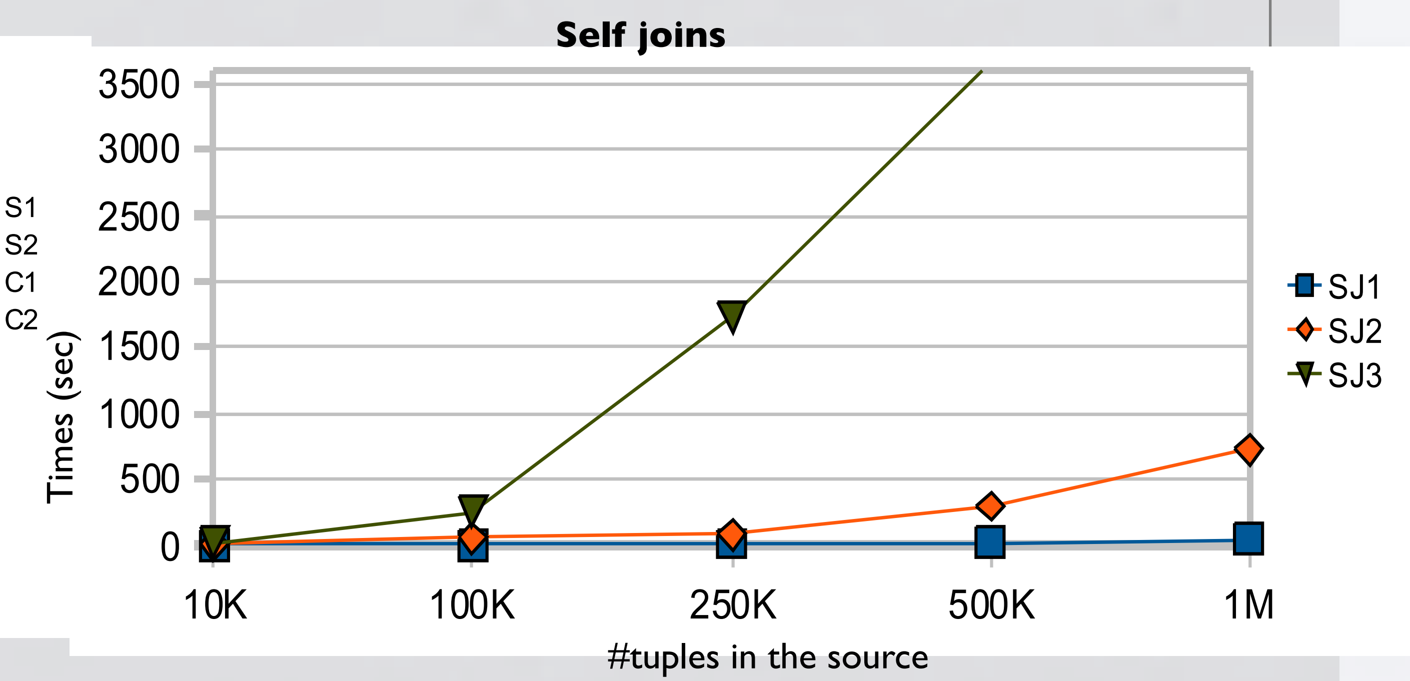
- Scripts in SQL (and XQuery)
  PostgreSQL 8.3 on a Intel CoreDuo 2.4Ghz/4GB Ram/Linux

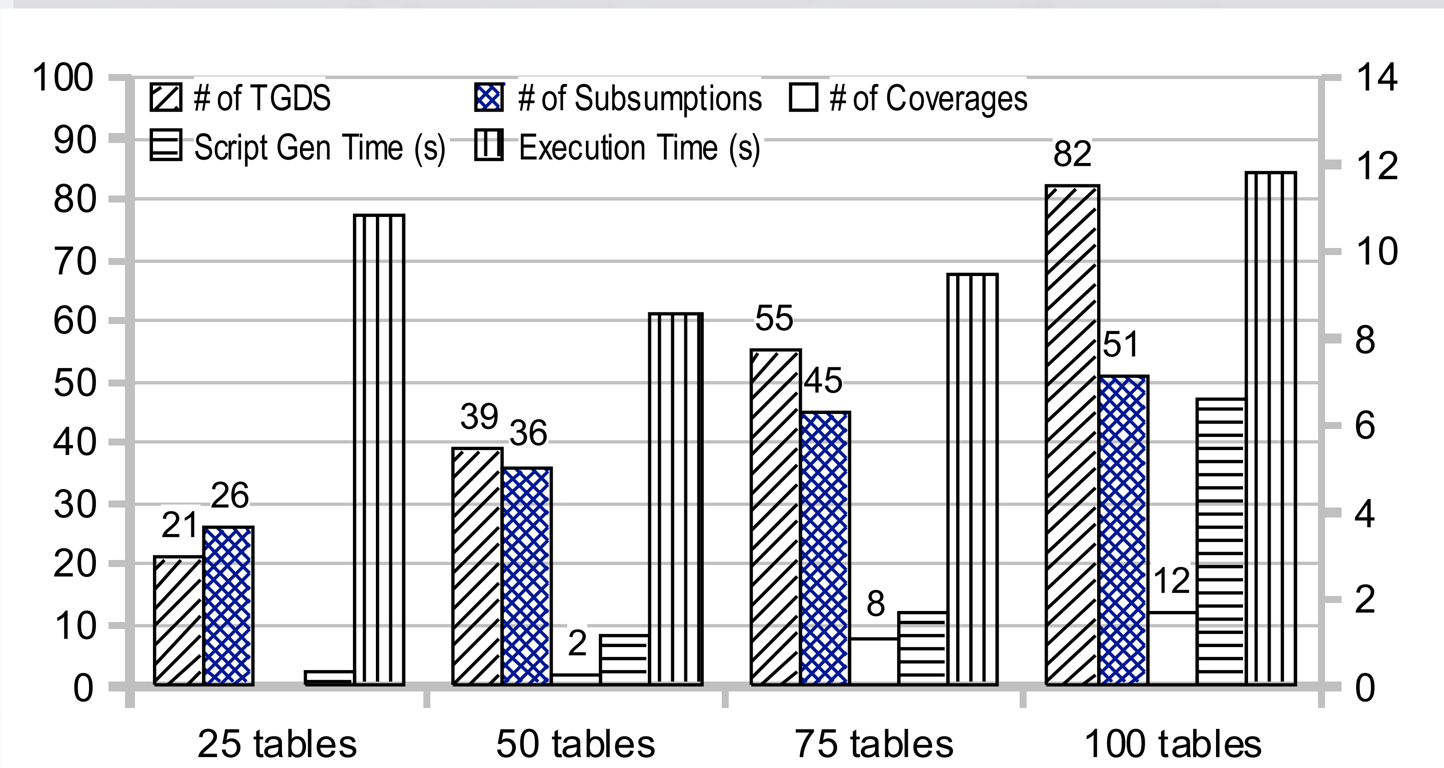
- Scenarios from the literature
  mostly from STBenchmark [Alexe et al. Vldb'08]

- Each SQL test
  - run with 10k, 100k, 250k, 500k, 1M tuples in the source
  - time limit = 1 hour
  - custom engine exceeded the time limit in all scenarios

Experiments results

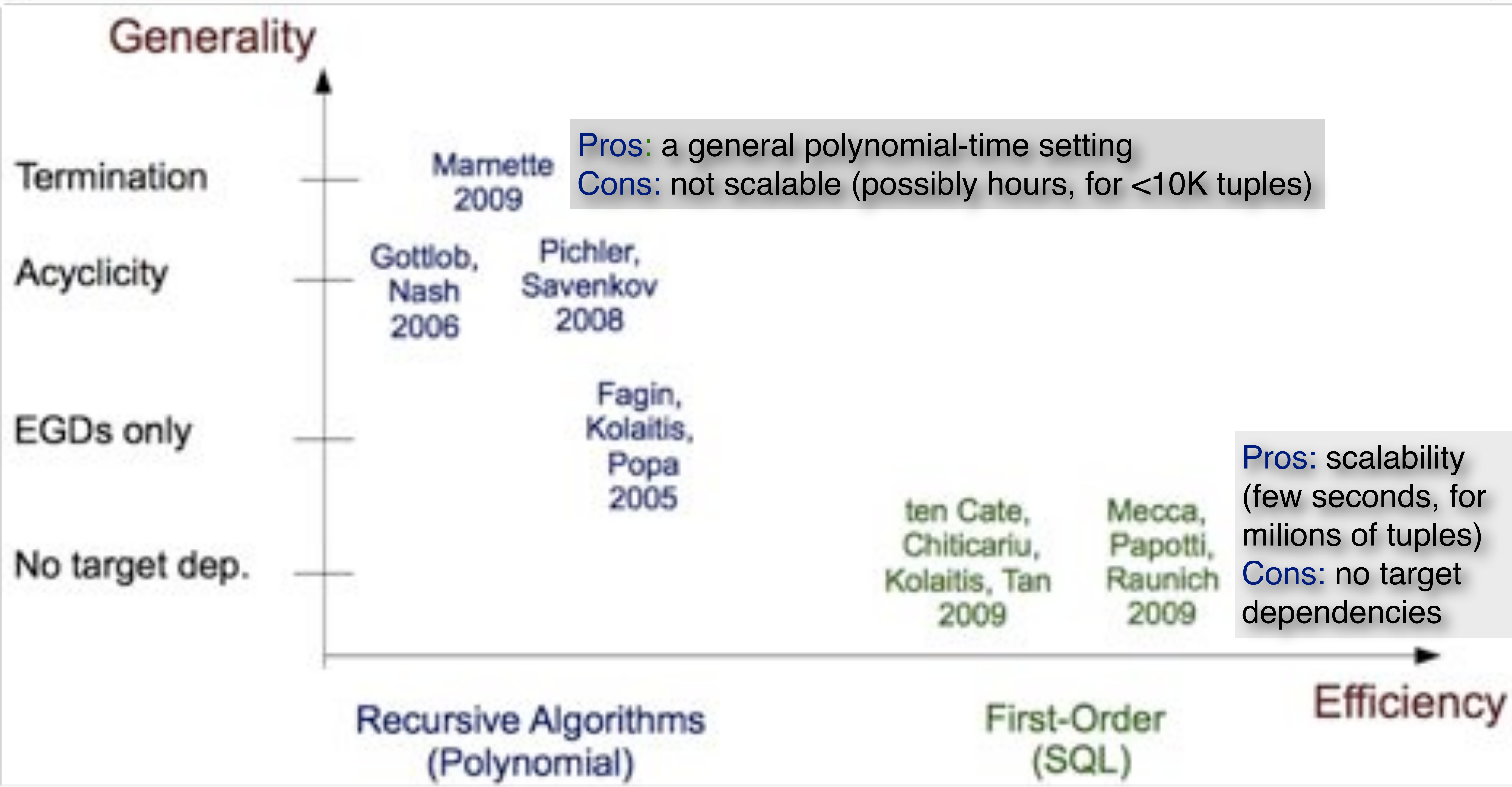






Scalability experiments with up to 100 tables (82 tgds, 51 subsuptions, 12 coverages): rewriting algorithm ran in 6 secs

Core computation landscape



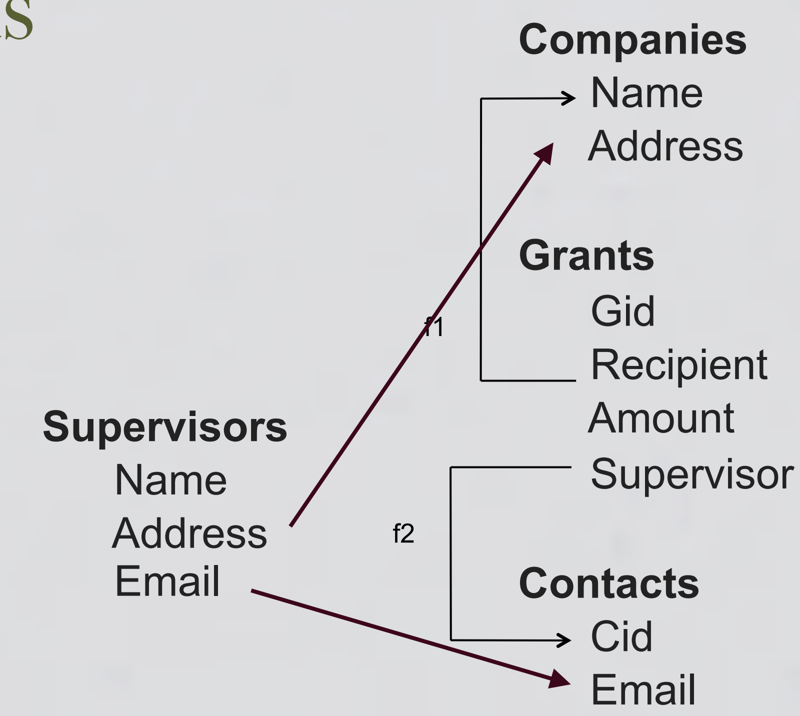
Assumptions for core mappings

- No arbitrary target constraints
  [Mecca et al. Sigmod'09] [ten Cate et al. Vldb'09]
  But....

- there is a workaround for **target tgds**: foreign key constraints can be rewritten into the s-t tgds
  [Popa et al. Vldb'02]

Supervisors (n,a,e) → Companies (N,a)
Supervisors (n,a,e) → Contacts(C,e) X

Supervisors (n,a,e) → Companies (**N**,a)
Grants (G, **N**, A, **C**) Contacts(**C**,e)



Assumptions for core mappings

- No arbitrary target constraints
  [Mecca et al. Sigmod'09] [ten Cate et al. Vldb'09]
  But….

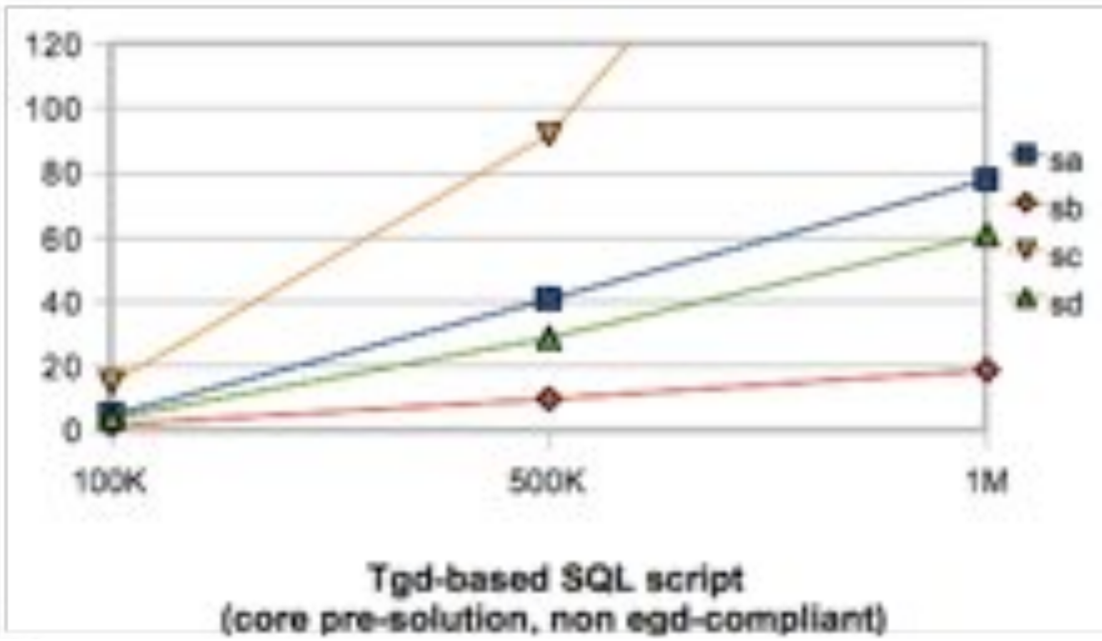
- there is a workaround for **target tgds**: foreign key constraints can be rewritten into the s-t tgds
  [Popa et al. Vldb'02]

- there is a best effort solution to rewrite also **target egds** into the s-t tgds
  [Marnette et al. Vldb'10]

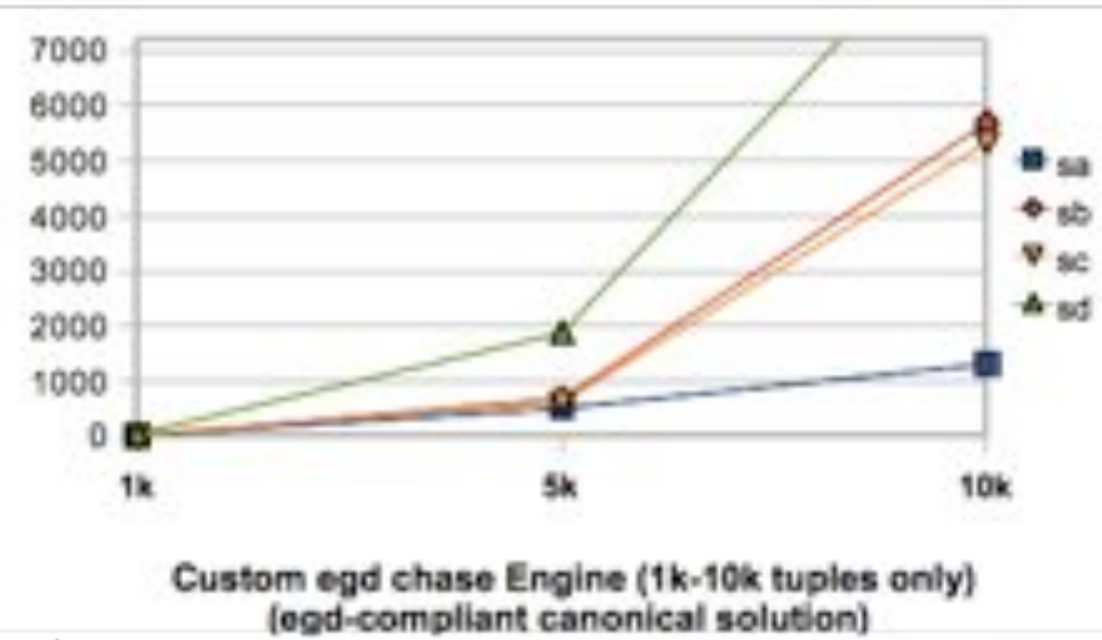
Target Functional Dependencies

- There are scenarios for which no correct SQL-script exists [Marnette et al, Vldb'10]
- s-t tgd: Friend(x,y) $\rightarrow \exists$g, Group(x,g) $\wedge$ Group(y,g)
- target FD: Group(x,g1) $\wedge$ Group(x,g2) $\rightarrow$ g1=g2
  - Friend (Anne,Bob) Friend (Bob,Ciad) [Friend (C,D) ...]
  - Recursion needed to compute the connected components of Friend
- No need to give up!
  - Many real-life scenarios remain in the scope of SQL
  - There are algorithms to recognize the bad cases

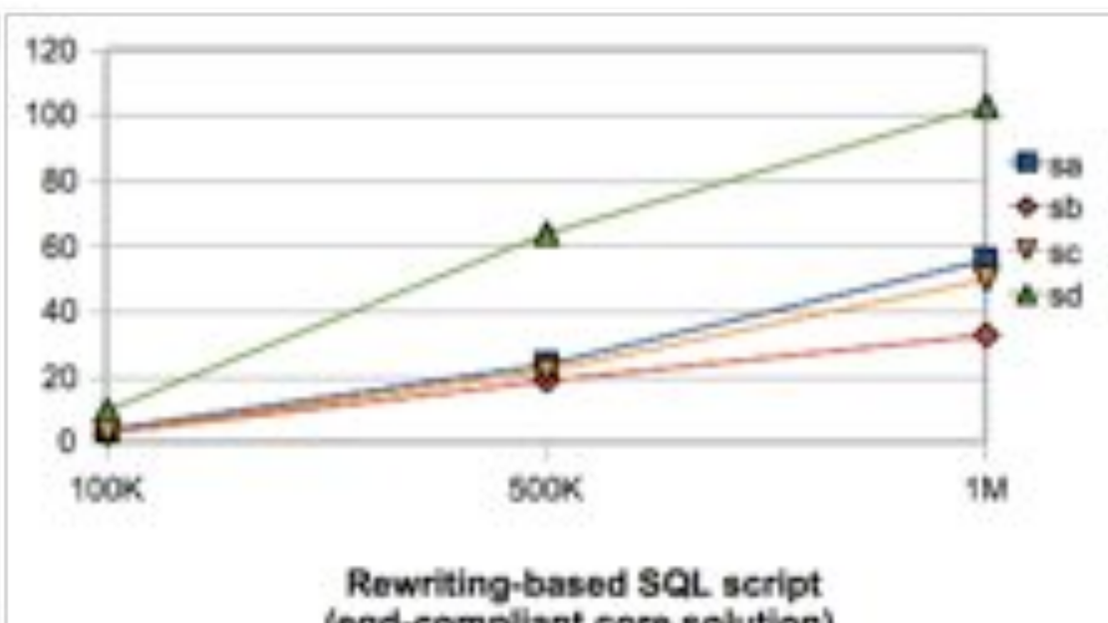
Experiments results



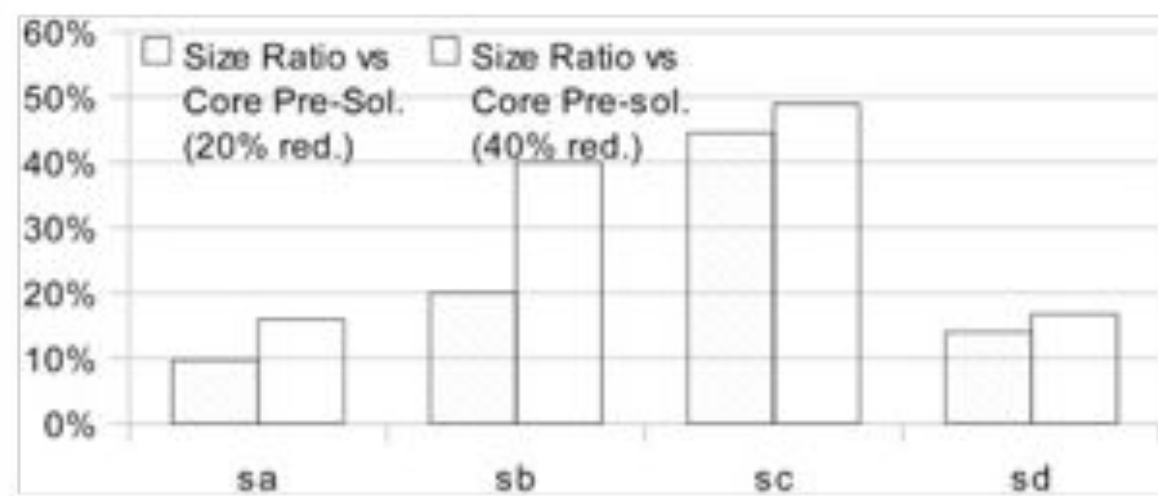
Tgd-based SQL script
(core pre-solution, non egd-compliant)



Custom egd chase Engine (1k-10k tuples only)
(egd-compliant canonical solution)



Rewriting-based SQL script
(egd-compliant core solution)



Benefits of s-t tgds

- Again: scalability and portability
- All these rewriting algorithms rely on the generation of s-t tgds as intermediate form, thus making possible the formal study of mapping properties and their optimization

Schema mapping optimization

- Consider $M = \langle S, T, \Sigma_{st} \rangle$ with

  S = Lecture (title, year, prof) Prof (name, area) Course (title, prof-area)
  T= MasterCourse (title,area)

- and (handwritten) Σ_{st} =

  $L(x1, x2, x3) \wedge L(x4, \text{‘3’}, x5) \wedge P(x5, x6) \rightarrow C(x4, x6)$
  $L(x1, \text{‘3’}, x2) \wedge P(x2, \text{‘db’}) \rightarrow C(x1, \text{‘db’})$

- Equivalent, simplified set of s-t tgds [Gottlob et al, Vldb’09]:

  $\Sigma_{st}' = \{ L(x4, \text{‘3’}, x5) \wedge P(x5, x6) \rightarrow C(x4, x6) \}$

Optimality Criteria
[Gottlob et al, Vldb'09]

- **Splitting** should be applied whenever possible
- Optimization goals:
  - **cardinality-minimality**: minimal number of st-tgds in Σ_{st}
  - **antecedent-minimality**: minimal total size of the antecedents
  - **conclusion-minimality**: minimal total size of the conclusions
  - **variable-minimality**: minimal total number of existentially quantified variables in the conclusions

Equivalence of schema mappings

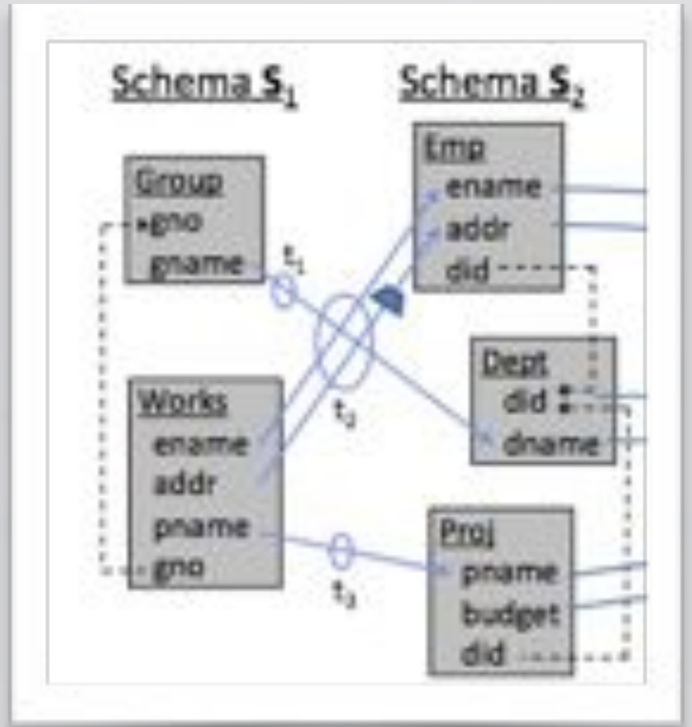
* Two relaxed notions of equivalence aside from standard logical equivalence (solutions coincide)
  [Fagin et al. Pods'08] [Pichler et al. Icdt'11]
  * data-exchange (DE) equivalence (univ. solutions coincide)
  * conjunctive-query (CQ) equivalence (core sol. coincide)
* DE and CQ equivalences coincide with logical equivalence when the mapping scenario is made only of s-t tgds (i.e., $\Sigma = \Sigma st$)
* Also optimization beyond equivalence [Calvanese et al. Icdt'11]

Model Management

* Generic approach, based on operators over schema mappings to solve problems of data programmability [Bernstein, Cidr'03]
* Semantic and algorithms for model management operators have been studied in recent time with mixed results

Model Management Operators

- **Confluence** operator, which describes the operation of merging two or more schema mappings, has been formalized and implemented
- MapMerge [Alexe et al. Vldb'10]
  - "divide-and-merge" paradigm (using SO tgds)
  - merge using schema constraints (in Clio spirit) and a heuristic to reuse mapping behavior from more general mappings (using Skolem)



Model Management Operators

- Formalization and implementation also for:
  - **Composition** of mappings [Madhavan et al. Vldb'03] [Fagin et al. Pods'04] [Yu&Popa. Vldb'05] [Bernstein et al. Vldb'06] [Arenas et al. Icdt'10]
  - **Merge** of schemas [Pottinger&Bernstein. Vldb'03] [Chiticariu et al. Sigmod'08]
  - **ModelGen** of schemas [Atzeni et al. VLDBJ'08]

  - Match; Diff; …
  [Bernstein. Cidr'07][Bernstein&Melnik, Sigmod'07]

Inverse operator

- On the contrary, for the **Inverse** operator, in general there are schema mappings for which <u>inversions that recover all the original data back do not exist</u> [Fagin. Pods'06]
  - projection $P(x,y) \rightarrow Q(y)$, union $P(x) \rightarrow Q(x)$ $R(x) \rightarrow Q(x)$, decomposition $P(x,y,z) \rightarrow Q(x,y) \wedge T(y,z)$

- Relaxed notions of invertibility follow a pragmatic approach: when an exact inverse does not exist, they recover the original source data as much as possible [Arenas et al. Pods'08-Vldb'09] [Fagin et al. Pods'07-Pods'09]

Inverse operator

* $Emp(x, y, z) \wedge y \neq z \wedge \neg DrivesWork(x) \rightarrow Shuttle(x)$

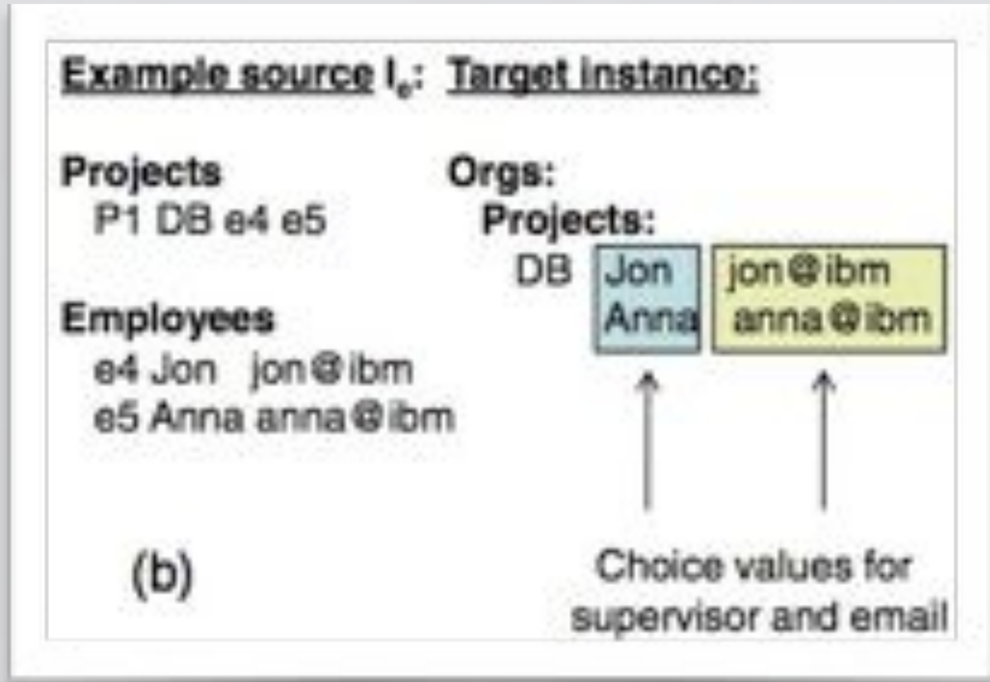
  [Arenas et al. Pods'08]

  - $Shuttle(x) \rightarrow \exists u \exists v\, Emp(x, u, v)$
  - $Shuttle(x) \rightarrow \exists u \exists v\, (Emp(x, u, v) \wedge u \neq v)$
  - $Shuttle(x) \rightarrow \exists u \exists v (Emp(x,u,v) \wedge u \neq v \wedge \neg DrivesWork(x))$

* Relaxed notions are still useful for practical applications, although so far they have been tested only in restricted settings [Curino et al. Vldb'08]

Instances

- Schema mappings can become complex in real-life applications: need for tools to support their understanding and design
  - use <u>data examples</u> to develop and illustrate schema mappings [Alexe et al. Icde'08]



- There exist schema mappings not characterized by any finite set of examples [Alexe et al, Pods'10]
  - novel notions of positive, negative, and universal examples

- Related: find a valid schema mapping given data examples only [Gottlob et al, Pods'08][Alexe et al, Sigmod'11]

XML Data Exchange

- XML is a more powerful data model
  - data organized into **trees** - queries expressed as **patterns**

	arbitrary DTDs	nested relational DTDs
CONS $(\Downarrow, \_)$	EXPTIME-complete	PTIME
CONS $(\Downarrow, \Rightarrow, \_)$	EXPTIME-complete	PSPACE-hard (even for CONS $(\Downarrow, \rightarrow)$)
CONS $(\Downarrow, =, \_)$	undecidable	NEXPTIME-complete (even with $\neq$)
CONS $(\Downarrow, \Rightarrow, =, \_)$	undecidable	undecidable
ABCONS $(\Downarrow, \_)$	in EXPSPACE; NEXPTIME-hard	PTIME for ABCONS$(\downarrow)$

- What is the tractable class for the XML world? [Arenas et al. Morgan & Claypool 2011]

Ontologies

* More powerful languages are usually needed for ontologies
  - DL-lite [Calvanese et al. AAAI'05]
  - Datalog± [Cali et al. Icdt'09]

* Query answering

  * Chase may be infinite, rewriting techniques alleviate the problem [Kontchakov et al. KR'10] [Cali et al. Vldb'10]

* Schema mappings among ontologies?

New tools available

- ++Spicy [Mecca et al. Vldb'11 (public release)]
  - matcher, mapping generation, core solutions, target FK+egds
- OpenII [Seligman et al. Sigmod'11]
  - matcher, mapping generation, schema repository, compare and merge schemas
- DEMo [Pichler et al. Vldb'09]
  - chase engine, core solutions, arbitrary target constraints
- ChaseT [Spezzano et al. Vldb'10 - Sebd'11 (public release)]
  - chase engine, termination, arbitrary target constraints
- MOMIS [Bergamaschi et al. IJCIS'02- Sebd'11 (public release)]
  - matcher, mappings, query rewriting

EMERGING APPLICATIONS

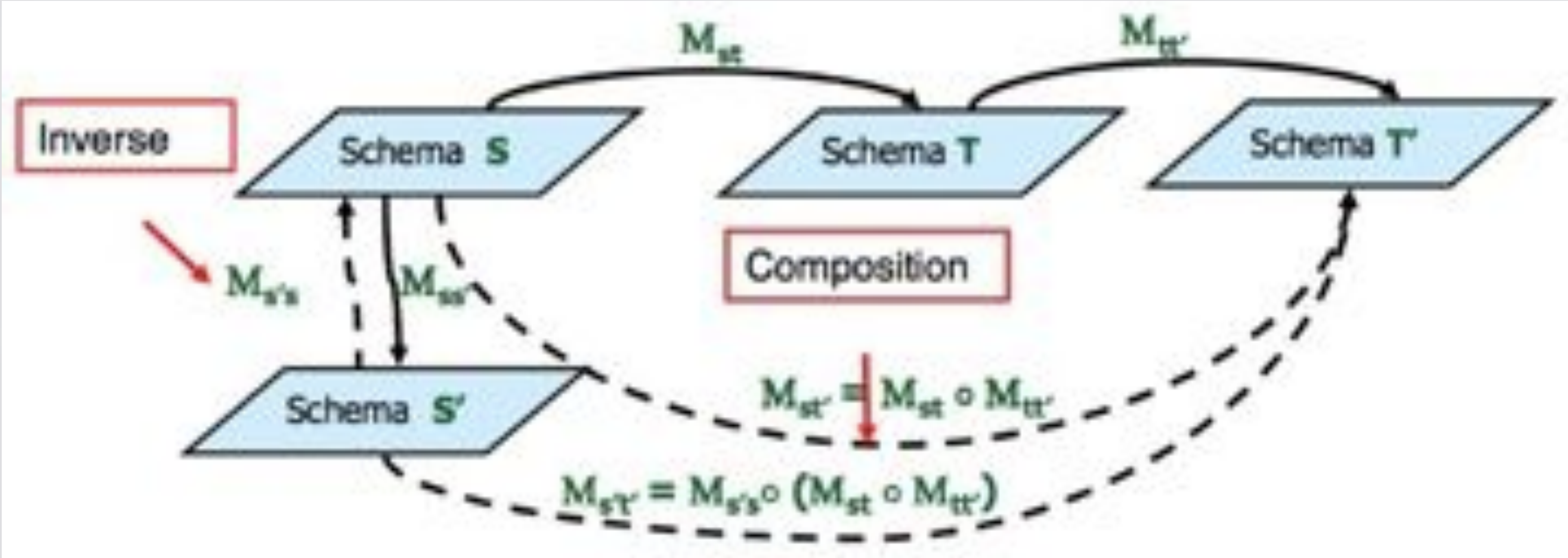
http://www.flickr.com/photos/elsonpro/5844650447/

Outline

- Schema evolution
- Data fusion
- Data cleaning

* ETL

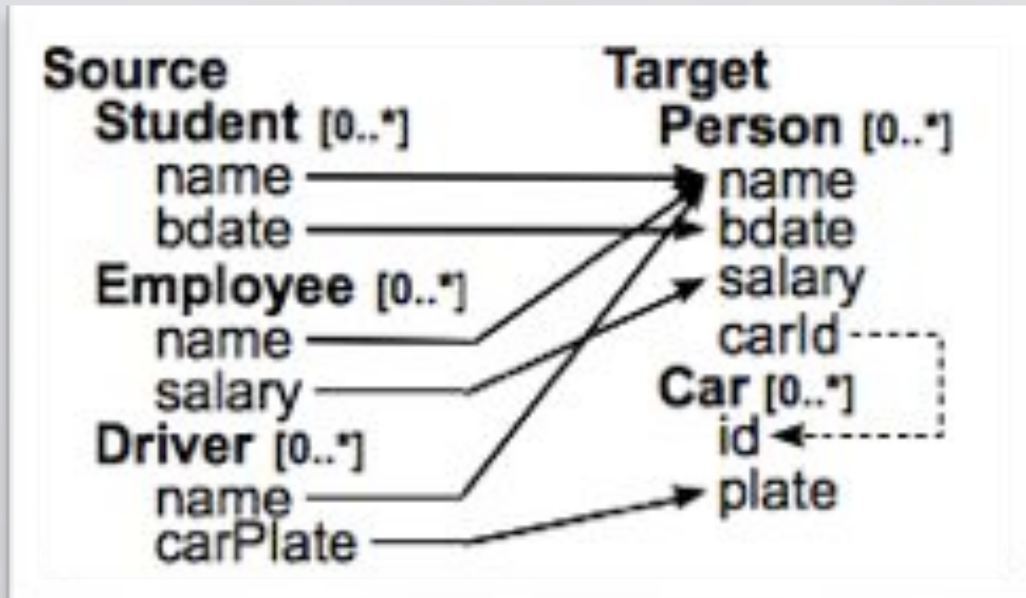
Schema evolution



- [Bernstein&Melnik. Sigmod'07] [Fagin et al. Schema Matching and Mapping, Springer'11]

- Lot of progress, but still missing an unifying schema mapping language that has (i) good algorithmic properties and (ii) is closed under both composition and the various flavors of inverses

Data fusion



a. Source Tables

Student

name	*bdate*
Jim	1980
Ray	1990

Employee

name	*salary*
Jim	25,000
Mike	17,000

Driver

name	*carPlate*
Jim	abc123
Joe	abc123
Mike	cde345

b. S-T Solution (does not enforce keys)

Person

name	*bdate*	*salary*	*carId*
Jim	1980	NULL	NULL
Jim	NULL	25,000	NULL
Jim	NULL	NULL	C1
Ray	1990	NULL	NULL
Mike	NULL	17,000	NULL
Mike	NULL	NULL	C3
Joe	NULL	NULL	C2

Car

id	*plate*
C1	abc123
C2	abc123
C3	cde345

c. Expected Solution

Person

name	*bdate*	*salary*	*carId*
Jim	1980	25,000	C1
Ray	1990	NULL	NULL
Mike	NULL	17,000	C3
Joe	NULL	NULL	C2

Car

id	*plate*
C1	abc123
C2	cde345

* [Marnette et al. Vldb'10]

Data cleaning

a. Employees

name	*age*	*salary*
Paul	1978	NULL
Paul	NULL	29,000
Paul	1979	NULL
Melanie	1990	NULL
Bob	NULL	37,000
Bob	1977	NULL
Charlie	1978	32,000

b. Employees

name	*age*	*salary*
Paul	?	29,000
Melanie	1990	NULL
Bob	1977	37,000
Charlie	1978	32,000

Paul.age = 1978 OR 1979 ?

- Support a principled approach to cleaning based on constraints [Galhardas et al, Vldb'01][Bertossi et al, Icdt'11]

Are mappings ready for the market?

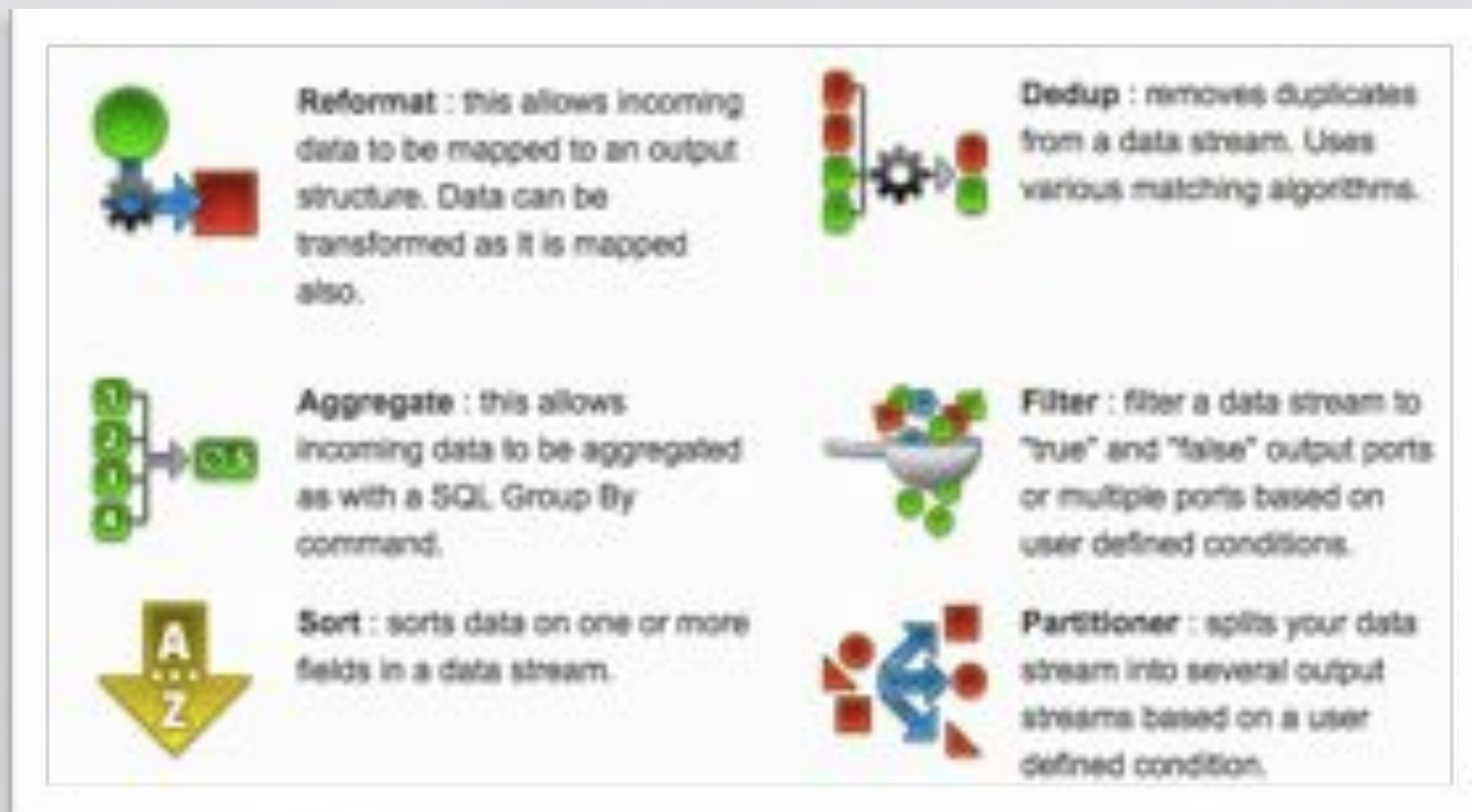
?

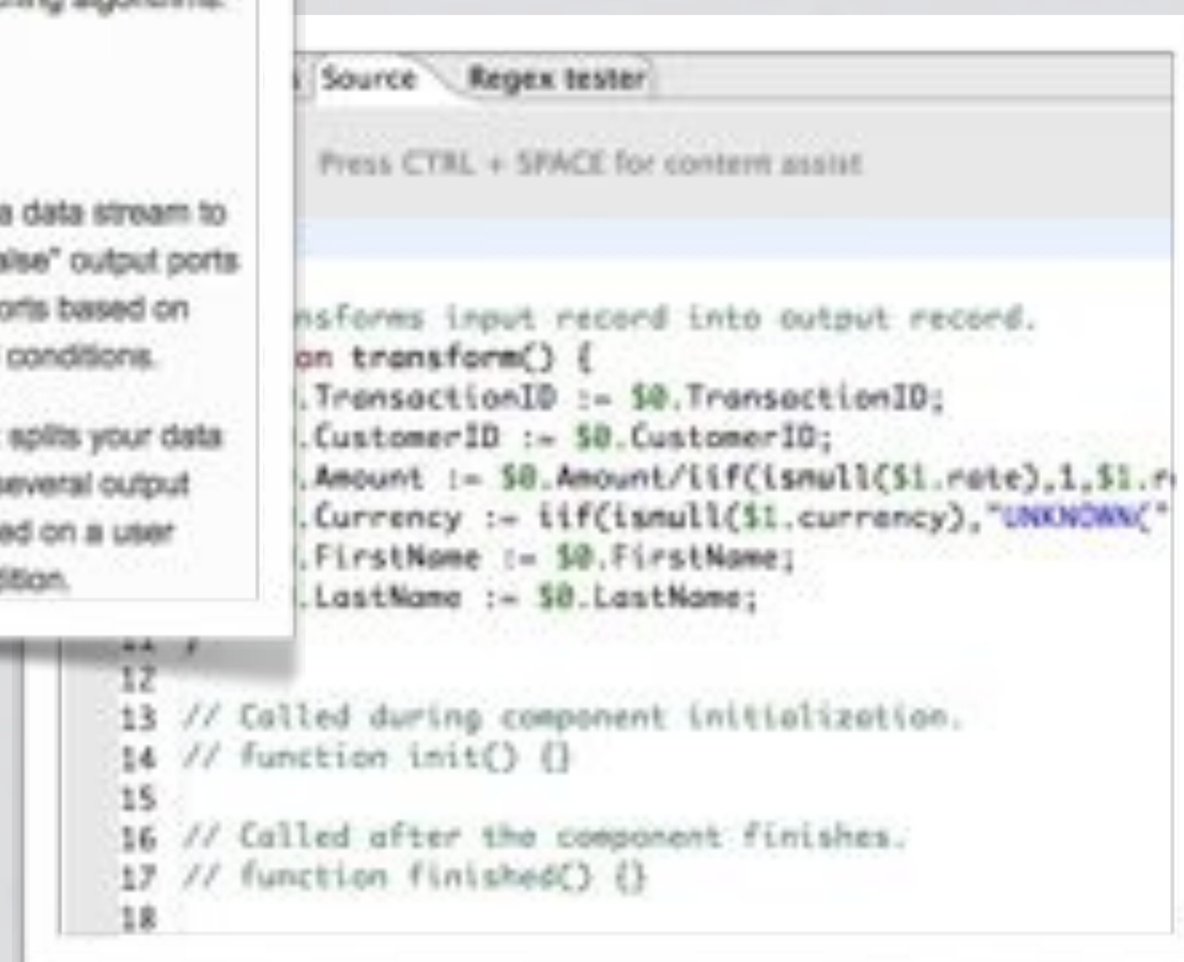
ETL tools

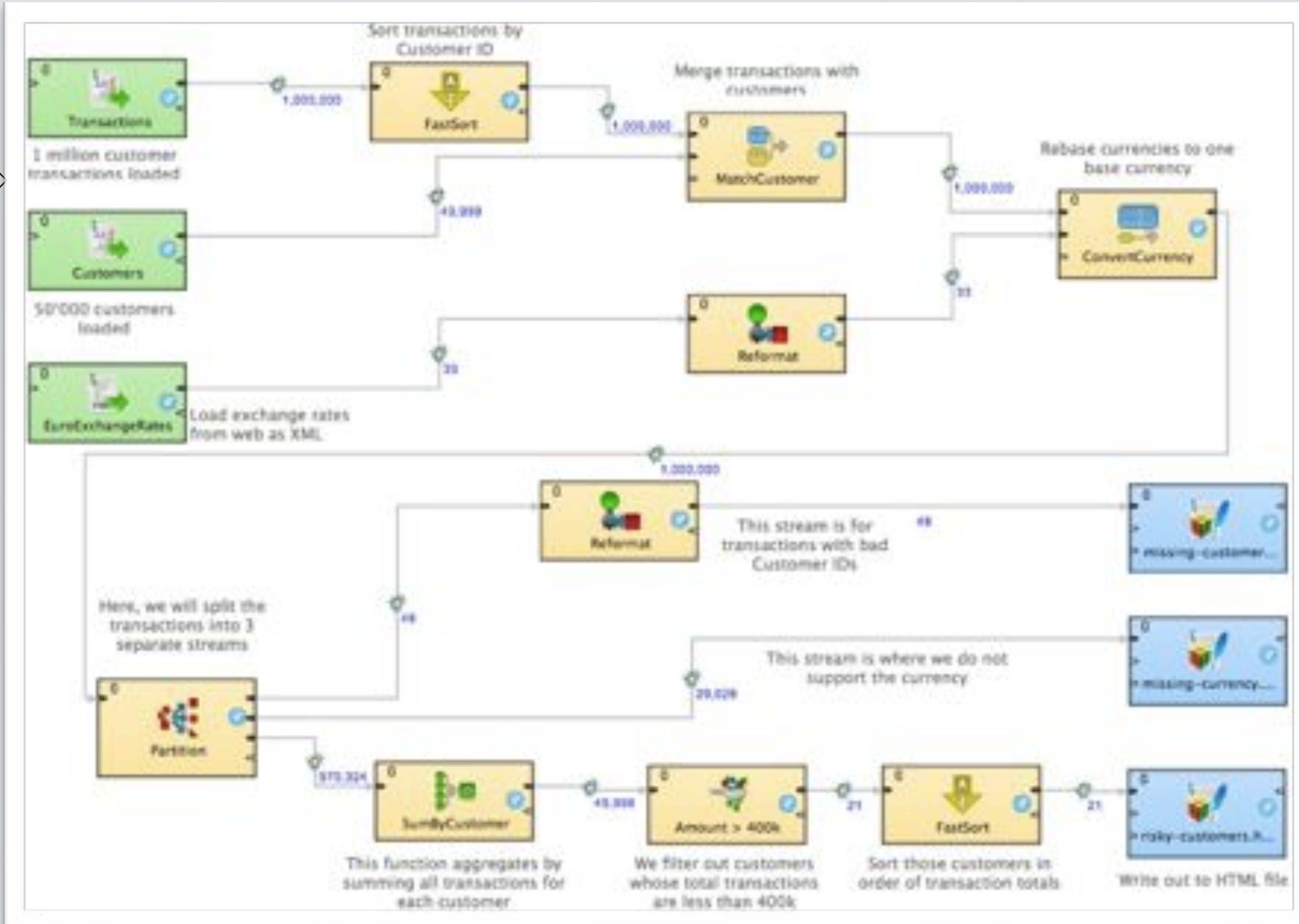
- **ETL** stands for Extract–Transform–Load
  - **Extract** (large volumes) data from multiple sources
  - **Transform** it so it is compatible with the schema
  - **Load** it into a database (warehouse)
- Most widely used systems in data warehousing environments

Data Transformation Graph from www.cloveretl.com

Components + script language





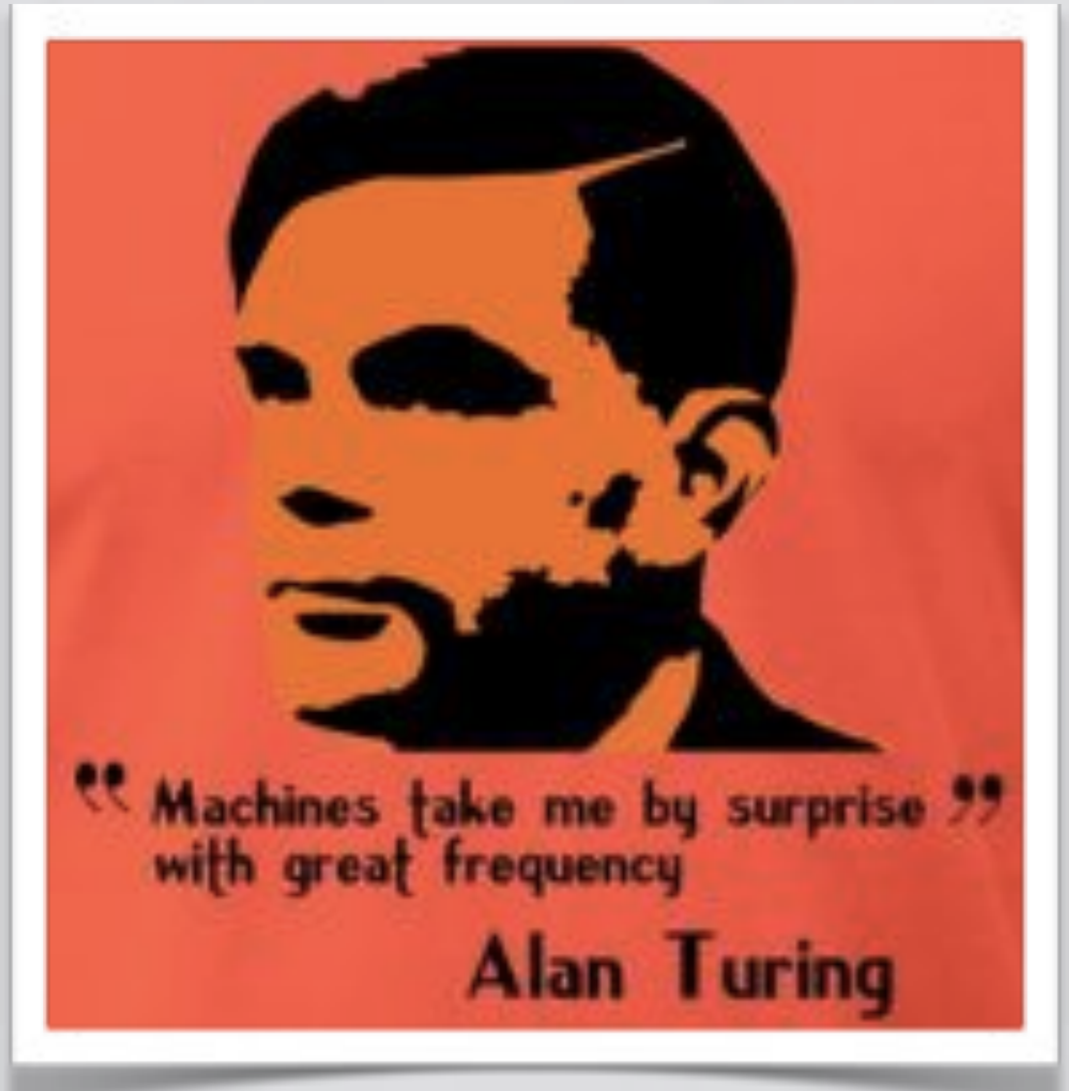

Sort transactions by Customer ID
Transactions
1,000,000
FastSort
1,000,000
Merge transactions with customers
MatchCustomer
1 million customer transactions loaded
49,999
1,000,000
Rebase currencies to one base currency
ConvertCurrency
Customers
50'000 customers loaded
Reformat
EuroExchangeRates
Load exchange rates from web as XML
1,000,000
Reformat
This stream is for transactions with bad Customer IDs
missing-customer...
Here, we will split the transactions into 3 separate streams
This stream is where we do not support the currency
missing-currency...
29,028
Partition
SumByCustomer
Amount > 400k
FastSort
risky-customers.h...
This function aggregates by summing all transactions for each customer
We filter out customers whose total transactions are less than 400k
Sort those customers in order of transaction totals
Write out to HTML file

Why ETL?

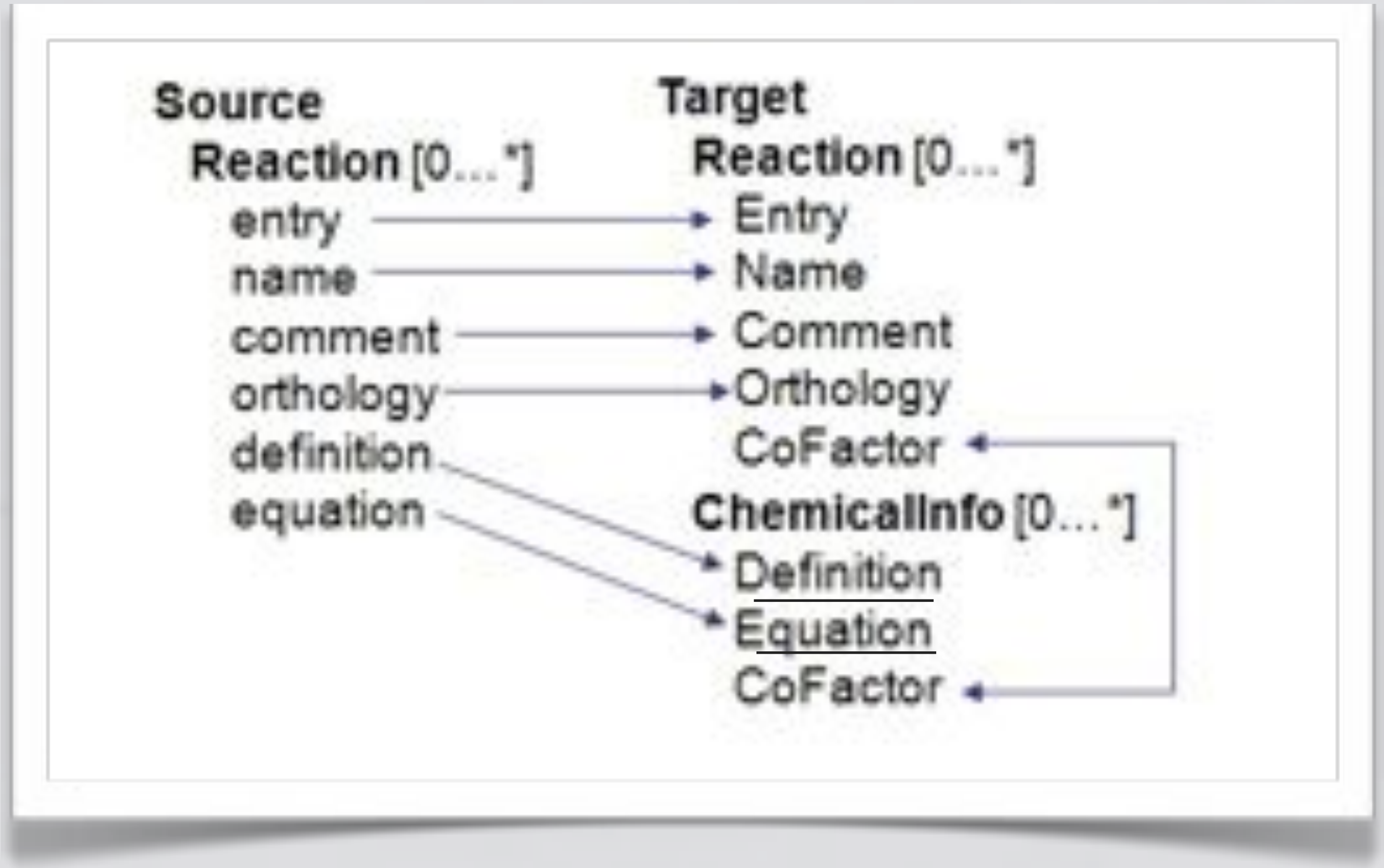
* **Procedural** fashion to design data exchange tasks.
  Focus on
  - **Data profiling**: samples, statistics, graphical tools to explore data
  - **Data cleaning**: e.g., Last Name vs LName; George St. vs George Street
  - **Simple transformations**: e.g., age = current year - ODB
  - **Performance/Scalability**
  - **Heavy emphasis on industry specific formats**
  e.g., Informatica has healthcare and financial services with support for specific formats: MS Word, Excel, PDF, UN/EDIFACT (Data interchange for admin., commerce and transport), RosettaNet, hospital forms, ...

Two good reasons for mappings

- Mapping research prototypes are more "intelligent"
  - have **clear semantics** (core solutions, target constraints)
  - require a **smaller user effort** for the same task

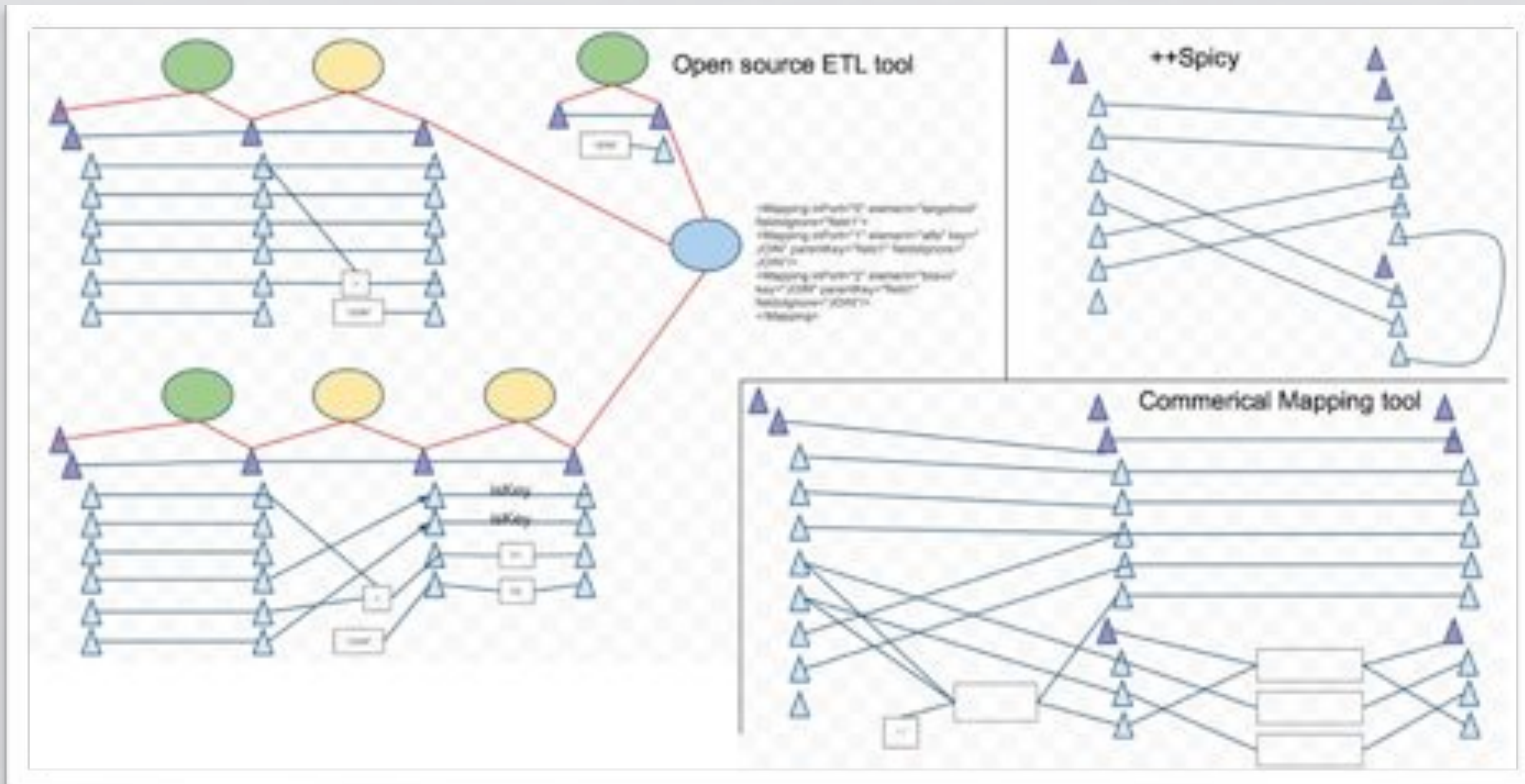


A simple example



[Alexe et al. Vldb'08 - www.stbenchmark.org]

Input graphs



So, why ETL?

- More popular than mapping systems because
  - they have a richer semantics, i.e., more operations [Dessloch et al. Icde'08] (but we have seen that something is going on)
  - the **declarative** nature of schema mapping tools become a limit with complex transformations when users have in mind <u>many intermediate steps</u>

Towards a “flow” of mappings

- Integration is complex: a mapping is often only **one piece** of a larger set of components (other mappings, transformations, black-box procedures) that need to be orchestrated together
- The designer may not know what the target is or how to get there: transformatios need to be built **incrementally**
- ETL (and data mashup systems) have nice “data flow” flavor but their level of abstraction is low (physical operators), with little opportunity for **automation, optimization and reuse**

Towards a “flow” of mappings

- “It may be easier to design a flow of small mappings that use intermediate results (small schemas) than a large complex mapping that goes directly from S to T” [Popa. INFINT’07]

- **Now** we have most of the pieces! [Mecca et al. Sigmod’09] [Marnette et al. Vldb’10] [Alexe et al. Vldb’10] [Alexe et al, Sigmod’11] ...

Open problems

* Mapping reuse [Wisnesky et al. Icdt'10] (e.g. Emp-Dept small mapping) + mapping repository
* Semi-automatic assembly of complex integration flows from existing mappings
* Labelled nulls in the source instances [Fagin et al. Icdt'09]
* Take new notions of inverse to the practice in data roundtrip scenarios (e.g., extending current object-to-relational systems [Melnik et al. Sigmod'07])

Questions?


QUESTIONS NOT EVEN 5+ YEARS OF GRAD SCHOOL WILL HELP YOU ANSWER
PAPER OR PLASTIC?
BUT, UNCLE, WHAT EXACTLY CAUSES GRAVITY?
um...
SO, IS MSG GOOD OR BAD FOR YOU?
SO, SON, THINK WE CAN FIX THIS OLD CAR?
um...
WHY WAR?
er...
PHD IN ENVIRONMENTAL ENGINEERING
PHD IN PHYSICS
PHD IN BIOLOGY
PHD IN MECHANICAL ENGINEERING
PHD IN POLITICAL SCIENCE
www.phdcomics.com

EXTRA

Connection with Data Exchange theory

- First generation mapping systems implement the chase
  - given correspondences between two schemas S, T, they generate a mapping scenario $M = (S, T, \Sigma st)$
  - given a source instance I of S, they generate a canonical universal solution J for M over I (**data exchange**)

- To do this, for a tgd $\forall X\ \Phi(X) \rightarrow \exists Y: \Psi(X, Y)$, they naively chase the tgd by running the following SQL statements:
  - a query $\Phi(I)$ over I to select all tuples that satisfy the premise,
  - a set of inserts - with proper Skolem terms -into J to satisfy $\Psi(X, Y)$

Mapping language desiderata

- Simple tasks that every schema mapping specification language should support:
  - Copy (Nicknaming), Projection, Column Augmentation, Decomposition, Join
  - Plus Combinations of the above (e.g., “join + column augmentation + …”)
- These simple tasks can be specified using tuple-generating dependencies (tgds) [Dependency theory in 70s and 80s]