PODS Special Event
Unlocking the Secrets: Bridging Theory and Practice

Renée J. Miller

University of Waterloo
Canada Excellence Research Chair
on Data Intelligence

Stories of Transformation: Story 1

- You've been promised:
  - first-hand accounts of projects that seamlessly transitioned between theory and practice, **igniting real-world impact**
- My first story:
  - first-hand account of a project that seamlessly transitioned from practice to theory, **igniting the theory community**

We begin summer 1998 visit to IBM Almaden

- Laura Haas: "You did a nice little academic thesis on data integration, but go down the hall and talk to our biologists and find out their **real** integration problems!"

From Data Integration to Data Exchange

Data integration:

Describe data in a global schema in terms of data in local schemas

Data exchange:

Describe data in a target schema in terms of data in a source schema, and actually produce the target database

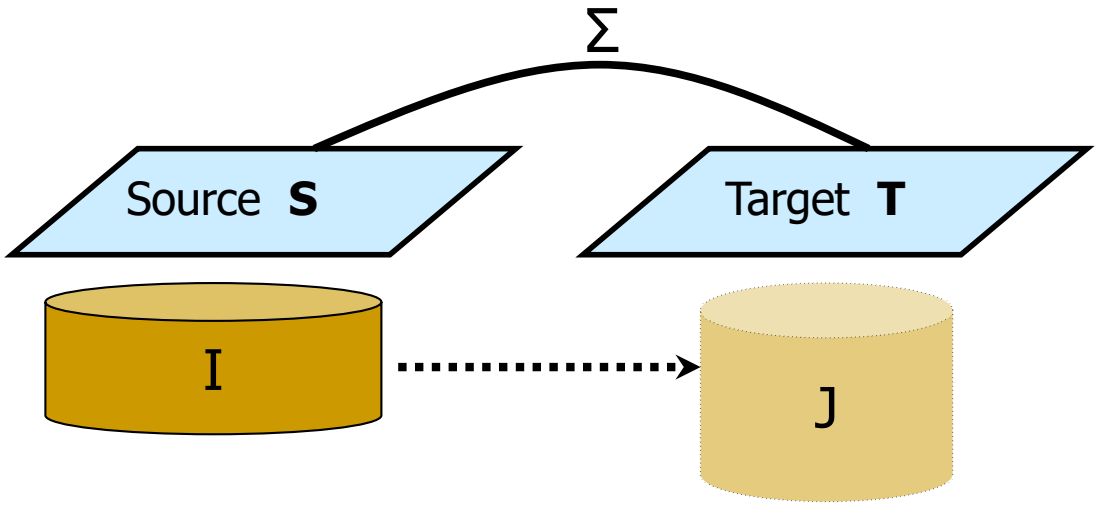
Onerous reporting requirements to federal agencies. Every time feds changed their schema, biologists had to spend hours debugging, testing and redeploying their low-level, procedural data exchange scripts.

We can help!!!

- Use declarative schema mapping language
- Semi-automatic generation of mappings
- Relational Solution
  - Miller, Haas, Hernández.
    Schema Mapping as Query Discovery. VLDB00
  - 2001 Haas moved to IBM product division to ***commercialize Clio*** on on the road to IBM Fellow
- Nested Relational Solution
  - Popa, Velegrakis, Miller, Hernández, Fagin.
    Translating Web Data. VLDB02

Clio

Schema Mappings & Data Exchange



- Schema Mapping **M** = (**S**, **T**, Σ)
  - Source schema **S**, Target schema **T**
  - High-level, declarative assertions Σ that specify the relationship between **S** and **T**
- Schema Mapping Σ Creation done via reasoning about schemas and data instances
  - Query discovery

Schema Mapping Specification Language

The relationship between source and target specified by *source-to-target tgds* (tuple-generating dependencies)

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

where

- $\varphi(\mathbf{x})$ is a conjunction of atoms over the source
- $\psi(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms over the target

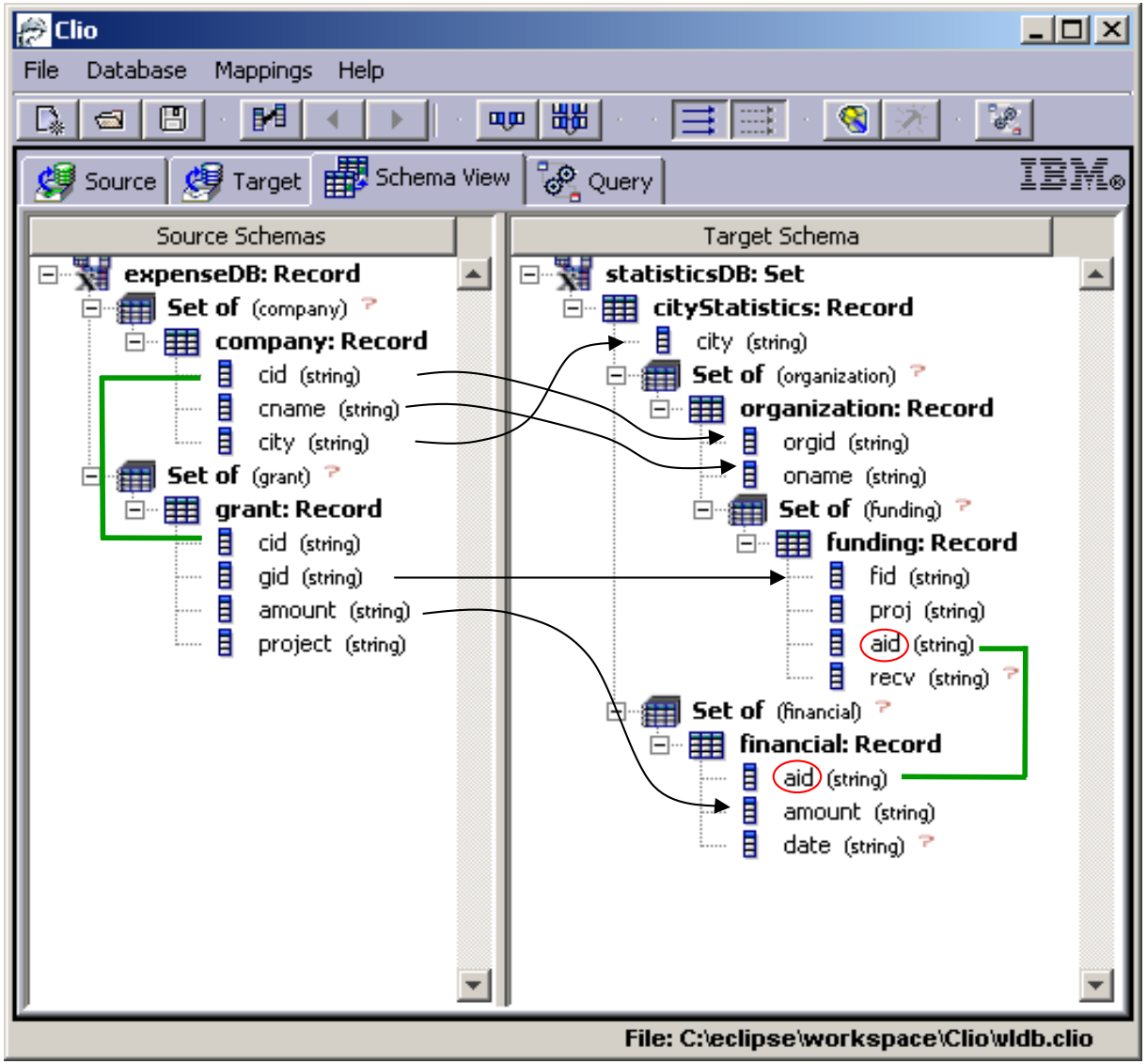
$$(\text{Student}(s) \wedge \text{Enrolls}(s,c)) \rightarrow$$
$$\exists t\ \exists g\ (\text{Teaches}(t,c) \wedge \text{Grade}(s,c,g))$$

There may also be target tgds and egds (equality-generating dependencies like functional dependencies):

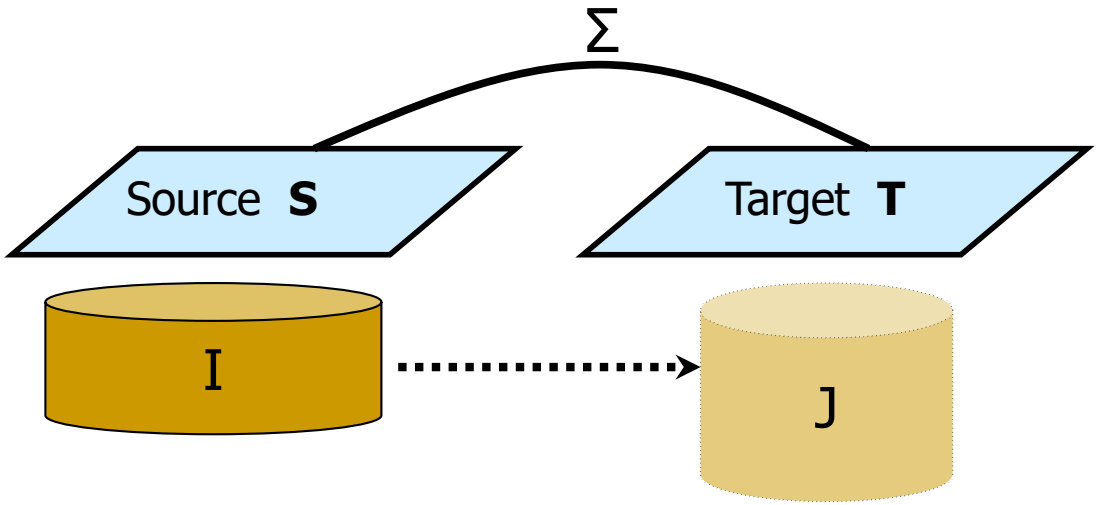
$$\text{Grade}(s,c,g)) \wedge \text{Grade}(s,c,g')) \rightarrow (g = g')$$

Some Features of Clio

- Supports nested structures
  - Nested Relational Model
  - Nested Constraints
- Automatic & semi-automatic discovery of attribute correspondence
- Interactive derivation of schema mappings
- Performs data exchange



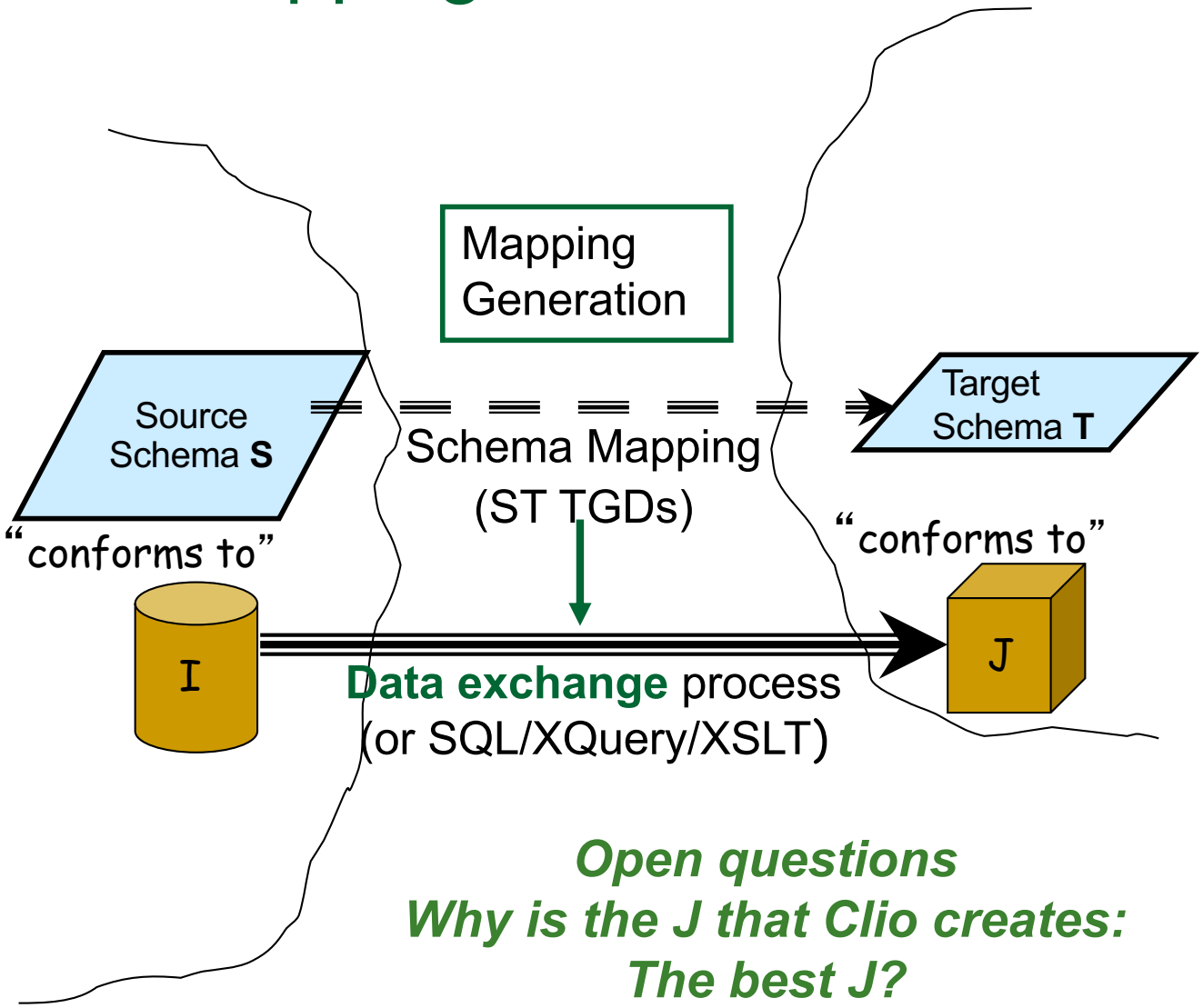
Schema Mappings & Data Exchange



- Schema Mapping **M** = (**S**, **T**, Σ)
  - Source schema **S**, Target schema **T**
  - High-level, declarative assertions Σ that specify the relationship between **S** and **T**
- Data Exchange via the schema mapping **M** = (**S**, **T**, Σ):

  Transform a given source instance I to a target instance J, so that <I, J> satisfy the specifications Σ of **M**

Schema Mappings in Clio



Open questions
Why is the J that Clio creates:
The best J?
A good J?

Solutions in Schema Mappings

Definition: Schema Mapping **M** = (**S**, **T**, Σ)

If I is a source instance, then a *solution* for I is a target instance J such that <I, J> satisfy Σ

Fact: In general, for a given source instance I,

- there may be no solutions at all

  or

- there may be multiple solutions; in fact there may be infinitely many solutions

What to exchange?

$$(\text{Student}(s) \wedge \text{Enrolls}(s,c)) \rightarrow \exists t\, \exists g\, (\text{Teaches}(t,c) \wedge \text{Grade}(s,c,g))$$

I

Student(Pat)

Student(Xiao)

Enrolls(Pat, DB101)

Enrolls(Xiao, Theory101)

J1

Teaches(*Floris*, DB101)

Teaches(*Floris,* Theory101)

Grade(Pat, DB101, *A*)

Grade(Xiao, Theory101, *A)*

- This **J1** is a correct solution (it satisfies the mapping), but is unsatisfying – it is too specific

What to exchange?

$(\text{Student}(s) \wedge \text{Enrolls}(s,c)) \rightarrow$
$\exists t\, \exists g\, (\text{Teaches}(t,c) \wedge \text{Grade}(s,c,g))$

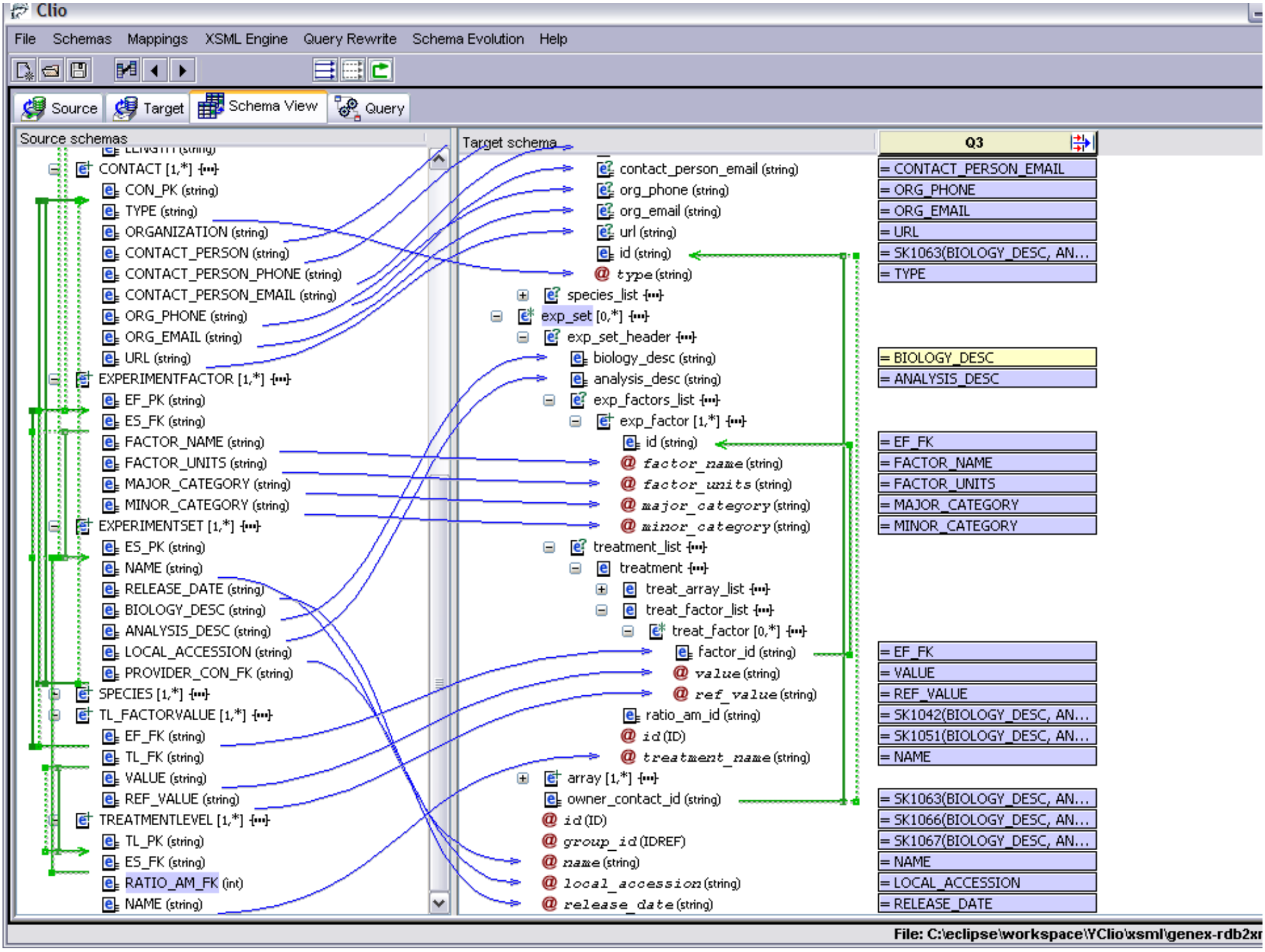
I

Student(Pat)
Student(Xiao)
Enrolls(Pat, DB101)
Enrolls(Xiao, Theory101)

J

Teaches(*t1*, DB101)
Teaches(*t2*, Theory101)
Grade(Pat, DB101, *g1*)
Grade(Xiao, Theory101, *g2*)

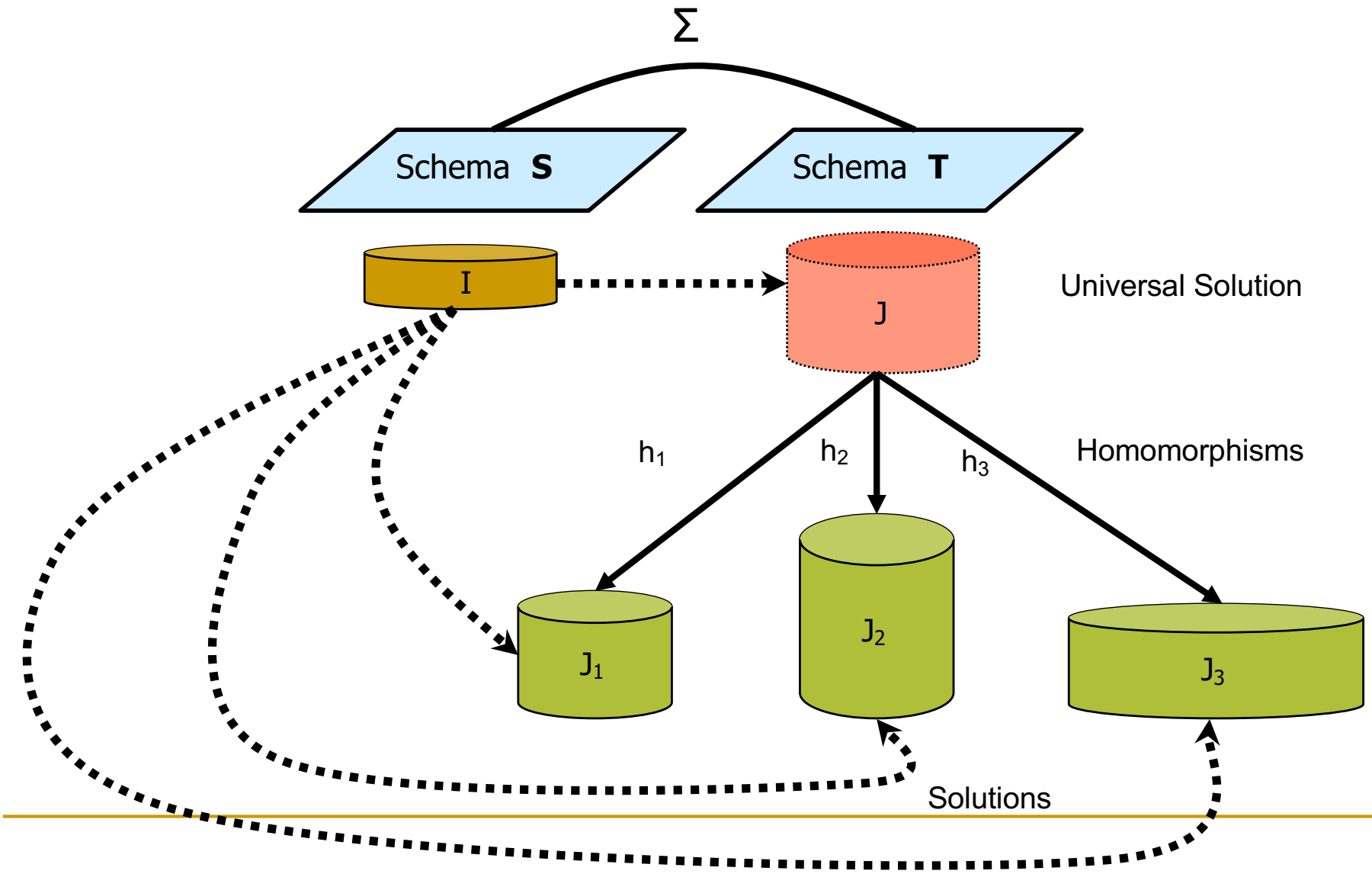
- Better solution **J** that uses labelled nulls for unknown values
  - Values *t1, t2, g1, g2*
- Clio created **J** & used Skolem terms for unknown values
  - Two different Skolems *may* have same value
  - The same Skolem term (in different relations) *must* have same value


Clio
File Schemas Mappings XSML Engine Query Rewrite Schema Evolution Help
Source Target Schema View Query
Source schemas
CONTACT [1,*]
CON_PK (string)
TYPE (string)
ORGANIZATION (string)
CONTACT_PERSON (string)
CONTACT_PERSON_PHONE (string)
CONTACT_PERSON_EMAIL (string)
ORG_PHONE (string)
ORG_EMAIL (string)
URL (string)
EXPERIMENTFACTOR [1,*]
EF_PK (string)
ES_FK (string)
FACTOR_NAME (string)
FACTOR_UNITS (string)
MAJOR_CATEGORY (string)
MINOR_CATEGORY (string)
EXPERIMENTSET [1,*]
ES_PK (string)
NAME (string)
RELEASE_DATE (string)
BIOLOGY_DESC (string)
ANALYSIS_DESC (string)
LOCAL_ACCESSION (string)
PROVIDER_CON_FK (string)
SPECIES [1,*]
TL_FACTORVALUE [1,*]
EF_FK (string)
TL_FK (string)
VALUE (string)
REF_VALUE (string)
TREATMENTLEVEL [1,*]
TL_PK (string)
ES_FK (string)
RATIO_AM_FK (int)
NAME (string)
Target schema
contact_person_email (string)
org_phone (string)
org_email (string)
url (string)
id (string)
type (string)
species_list
exp_set [0,*]
exp_set_header
biology_desc (string)
analysis_desc (string)
exp_factors_list
exp_factor [1,*]
id (string)
factor_name (string)
factor_units (string)
major_category (string)
minor_category (string)
treatment_list
treatment
treat_array_list
treat_factor_list
treat_factor [0,*]
factor_id (string)
value (string)
ref_value (string)
ratio_am_id (string)
id (ID)
treatment_name (string)
array [1,*]
owner_contact_id (string)
id (ID)
group_id (IDREF)
name (string)
local_accession (string)
release_date (string)
Q3
= CONTACT_PERSON_EMAIL
= ORG_PHONE
= ORG_EMAIL
= URL
= SK1063(BIOLOGY_DESC, AN...
= TYPE
= BIOLOGY_DESC
= ANALYSIS_DESC
= EF_FK
= FACTOR_NAME
= FACTOR_UNITS
= MAJOR_CATEGORY
= MINOR_CATEGORY
= EF_FK
= VALUE
= REF_VALUE
= SK1042(BIOLOGY_DESC, AN...
= SK1051(BIOLOGY_DESC, AN...
= NAME
= SK1063(BIOLOGY_DESC, AN...
= SK1066(BIOLOGY_DESC, AN...
= SK1067(BIOLOGY_DESC, AN...
= NAME
= LOCAL_ACCESSION
= RELEASE_DATE
File: C:\eclipse\workspace\YClio\xsml\genex-rdb2xr

Universal Solutions in Data Exchange

- [Fagin, Kolaitis, Miller, Popa – ICDT 2003] introduced 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): entries besides constants in target instances
  - Homomorphism h: $J_1 \rightarrow J_2$ between target instances:
    - h(c) = c, if c is a constant
    - If $P(a_1,...,a_m)$ is in J_1, then $P(h(a_1),...,h(a_m))$ is in J_2

Universal Solutions

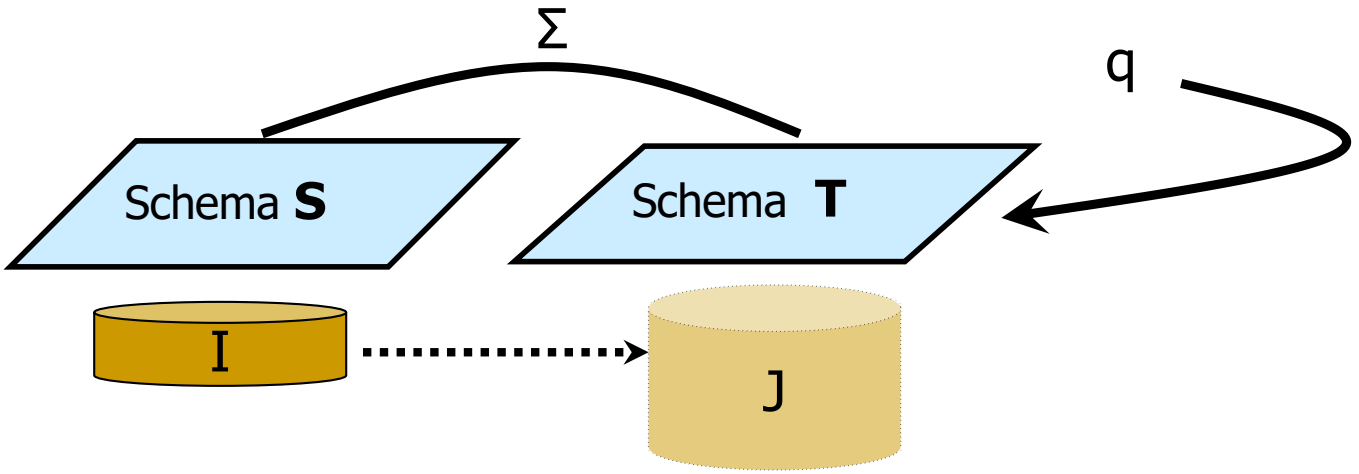


How to Obtain a Universal Solution?

- Answer: Use our old friend the chase!

  Theorem [Fagin, Kolaitis, Miller, Popa – ICDT 2003]:
  If there is a solution, then the chase produces a universal solution

Query Answering in Data Exchange



Question: What is the semantics of target query answering?

Definition: The certain answers of a query q over **T** on I

$$\textbf{certain}(q,I) = \bigcap \{\, q(J) : J \text{ is a solution for } I \,\}$$

Note: It is the standard semantics in data integration

Computing the Certain Answers

Theorem [Fagin, Kolaitis, Miller, Popa – ICDT 2003]:
Assume a standard schema mapping. Let q be a union of conjunctive queries over the target.

- If I is a source instance and J is a universal solution for I:

  certain(q,I) = the set of all "null-free" tuples in q(J).

- Hence, **certain**(q,I) is computable in polynomial time
  1. Compute a universal solution J, using the chase, in polynomial time
  2. Evaluate q(J) and remove tuples with nulls

Lessons Learned from Story 1

- Do your first sabbatical at IBM with Phokion Kolaitis
- Don't be afraid to ask Ron Fagin "easy" (stupid) questions
- Creating an intuitive, but useful solution may be more profound that you expected
  - No best paper award, but a 10-year ICDT test-of-time
  - Alonzo Church Award 2020
    - for Outstanding Contributions to Logic & Computation
    - Ronald Fagin, Phokion G.Kolaitis, Renée J. Miller, Lucian Popa, and Wang Chiew Tan
    - ground-breaking work on laying the logical foundations for data exchange
      - ICDT03 and PODS04 (Mapping Composition)

Crafting the Formula: Story 2

- Unveil the secret recipe for a successful theory-to-practice transition, straight from the experts' playbook.

- Here's my playbook:
  - Find some elegant and compelling theory
  - Even better if most results show that it is intractable and impractical
  - Make it practical

The ABC's of Inconsistent Query Answering

- Arenas, Bertossi, Chomicki PODS99: Consistent Query Answers in Inconsistent Databases. *aka* ABC99

Thanks to Marcello AMW17

What was in the ABC paper?

- Definition of consistent answer for arbitrary integrity constraints
  - Including a definition of repair
- A general methodology for computing consistent answers based on query rewriting
- A query rewriting algorithm for computing consistent answers (in some cases)

Inconsistent Databases

Integration is one of many reasons a DB may violate constraints

SATISFY custid KEY

Web

custid	…	income
Peter	…	40K
Paul	…	400K
Mary	…	110K

Sales

custid	…	income
Peter	…	200K
Paul	…	400K
Mary	…	130K

VIOLATES custid KEY

Inconsistent Integrated Database

custid	…	income
Peter	…	40K
Peter	…	200K
Paul	…	400K
Mary	…	110K
Mary	…	130K

Querying Inconsistent Databases

Example: Offering a Platinum credit card…

Query: "Get customers who make more than 100K"

Peter,Paul,Mary

Are we sure that we want to offer a card to Peter?

custid	***income***
Peter	40K
Peter	200K
Paul	400K
Mary	110K
Mary	130K

Formal Semantics Consistent Query Answers

- Related to semantics for querying incomplete data [Imielinski Lipski 84, Abiteboul Duschka 98]
  - Possible world: “complete” databases
- Consistent answers
  - Proposed by Arenas, Bertossi, and Chomicki in 1999
  - Possible world: “consistent” databases
    - Formalized as a repair
  - Consistent answer is one that is true in all consistent databases (repairs)

Repairs

Inconsistent database

custid	*income*
Peter	40K
Peter	200K
Paul	400K
Mary	110K
Mary	130K

Key: *custid*

Repairs

Peter	40K
Paul	400K
Mary	110K

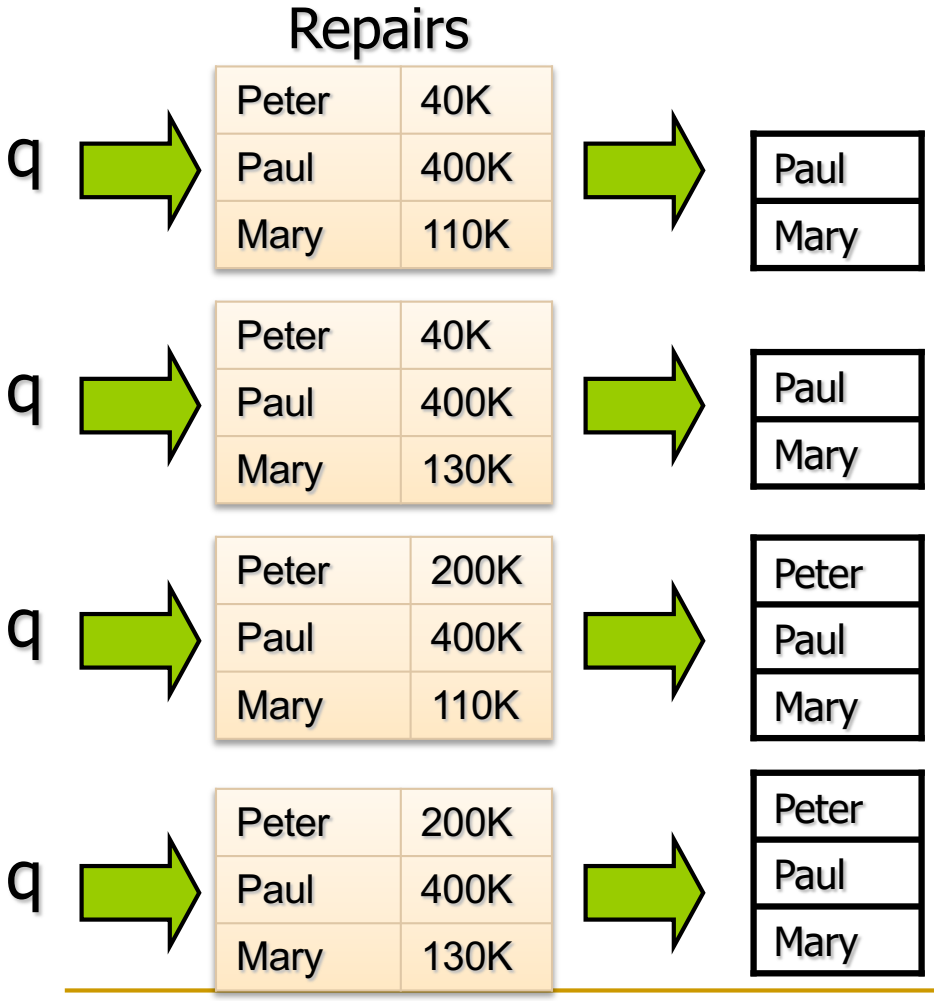
Peter	40K
Paul	400K
Mary	130K

Peter	200K
Paul	400K
Mary	110K

Peter	200K
Paul	400K
Mary	130K

Consistent Answers

Query=*"Get customers who make more than 100K"*



CONSISTENT ANSWERS
Answers obtained
no matter which repair
we choose

CONSISTENT ANSWER=
{Paul,Mary}

Simple semantics but intractable

- Schema
  - Orders (orderNo, custId) Key: orderNo
  - Invoices(invoiceNo, custId) Key: invoiceNo
- Query

  Are there orders and invoices for the same customer?

Computing consistent answers for this query is coNP-complete
[Chomicki and Marcinkowski 02]

- And this query is very simple
- But is it common?

Our Contributions – Theory

- Formal characterization of a broad class of queries
  - For which computing consistent answers is tractable under key constraints
  - That can be rewritten into first-order/SQL
- Extension of consistent query answering semantics for queries with grouping and aggregation
- Query rewriting algorithms for a class of
  - Select-Project-Join queries with set semantics
  - Select-Project-Join queries with bag semantics, grouping, and aggregation

Our Contributions – Practice

- Implementation of ConQuer
  - Designed to compute consistent answers efficiently
  - Used query rewriting within standard DBMS
  - Multiple rewriting strategies
- Experimental validation of efficiency and scalability
  - Representative queries from TPC-H
  - Large databases

SIGMOD Jim Gray Doctoral Dissertation Award 2008
Ariel Fuxman for his U. Toronto Ph.D. thesis
“Efficient Management of Inconsistent Databases”

Other examples of theory-practice playbook

- SIGMOD Tues 11am: **A Unified Approach for Resilience & Causal Responsibility with Integer Linear Programming (ILP) & LP Relaxations**
  Neha Makhija*, Wolfgang Gatterbauer
- PODS Wed 5pm: **Minimally Factorizing the Provenance of Self-join Free Conjunctive Queries**
  Neha Makhija*, Wolfgang Gatterbauer