Data Exchange in the Relational and RDF Worlds

Marcelo Arenas

Department of Computer Science
Pontificia Universidad Católica de Chile

This is joint work with Jorge Pérez, Juan Reutter, Cristian Riveros and Juan Sequeda

The problem of data exchange

Given: A source schema **S**, a target schema **T** and a specification $\Sigma_{\mathbf{ST}}$ of the relationship between these schemas

Data exchange: Problem of materializing an instance of **T** given an instance of **S**

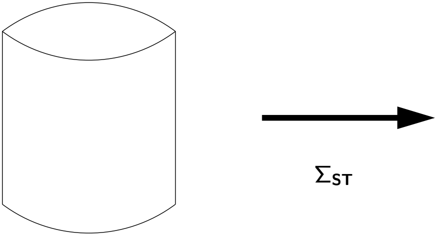
- Target instance should reflect the source data as accurately as possible, given the constraints imposed by $\Sigma_{\mathbf{ST}}$ and **T**
- It should be efficiently computable
- It should allow one to evaluate queries on the target in a way that is *semantically consistent* with the source data

Data exchange in a picture

$$\xrightarrow[\Sigma_{\mathbf{ST}}]{}$$

Schema **S** Schema **T**

Data exchange in a picture

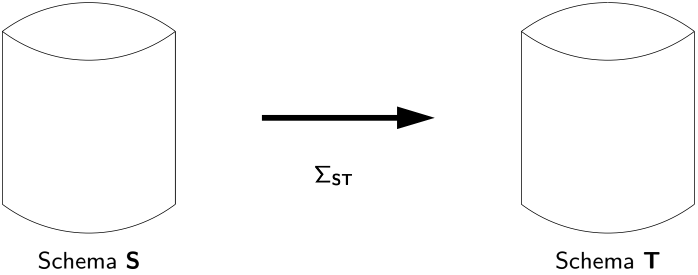


$\Sigma_{\mathbf{ST}}$

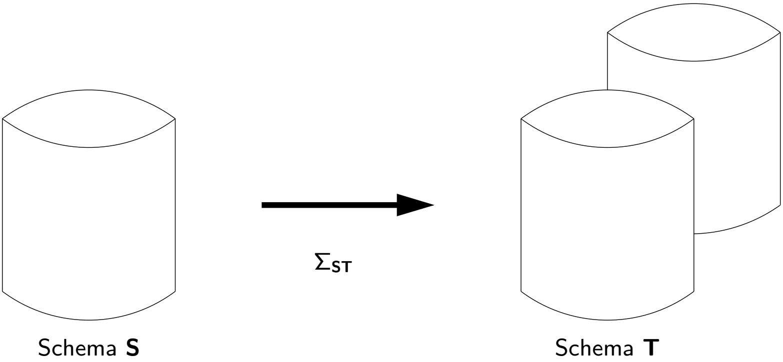
Schema **S**

Schema **T**

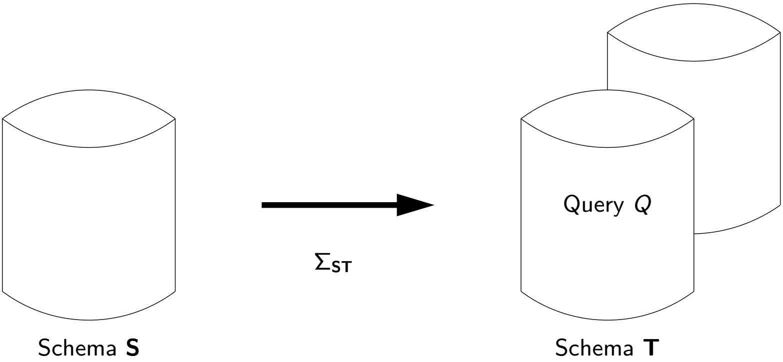
Data exchange in a picture



Data exchange in a picture



Data exchange in a picture



Data exchange: Some fundamental questions

Why is data exchange an interesting problem?

- Is it a difficult problem?

What are the challenges in the area?

- What is a good language for specifying the relationship between source and target data?
- What is a good instance to materialize? Why is it good?
- What does it mean to answer a queries over target data?
- How do we answer queries over target data? Can we do this efficiently?

Outline of the talk

- Relational data exchange
- Translating relational data into RDF
- Metadata management
  - Composition, inverse
- Concluding remarks

Outline of the talk

- Relational data exchange
- Translating relational data into RDF
- Metadata management
  - Composition, inverse
- Concluding remarks

Data exchange in relational databases

It has been extensively studied in the relational world.

- It has also been implemented: IBM Clio

Relational data exchange setting:

- Source and target schemas: Relational schemas
- Relationship between source and target schemas: Source-to-target tuple-generating dependencies (st-tgds)

Semantics of data exchange has been precisely defined.

- Efficient algorithms for materializing target instances and for answering queries over the target schema have been developed

Schema mapping: The key component in relational data exchange

Schema mapping: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$

- **S** and **T** are disjoint relational schemas
- $\Sigma_{\mathbf{ST}}$ is a finite set of st-tgds:

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

$\varphi(\bar{x}, \bar{y})$: conjunction of relational atomic formulas over **S**

$\psi(\bar{x}, \bar{z})$: conjunction of relational atomic formulas over **T**

Relational schema mappings: An example

Example

- **S**: *book*(*title*, *author_name*, *affiliation*)
- **T**: *writer*(*name*, *book_title*, *year*)
- $\Sigma_{\mathbf{ST}}$:

$$\forall x_1 \forall x_2 \forall y_1 \, (book(x_1, x_2, y_1) \rightarrow \exists z_1 \, writer(x_2, x_1, z_1))$$

Relational schema mappings: An example

Example

- **S**: *book*(*title*, *author_name*, *affiliation*)
- **T**: *writer*(*name*, *book_title*, *year*)
- $\Sigma_{\mathbf{ST}}$:

$$\forall x_1 \forall x_2 \forall y_1 \, (book(x_1, x_2, y_1) \rightarrow \exists z_1 \, writer(x_2, x_1, z_1))$$

Note

We omit universal quantifiers in st-tgds:

$$book(x_1, x_2, y_1) \rightarrow \exists z_1 \, writer(x_2, x_1, z_1)$$

Relational data exchange problem

Fixed: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$

Problem: Given instance I of $\mathbf{S}$, find an instance J of $\mathbf{T}$ such that (I, J) satisfies $\Sigma_{\mathbf{ST}}$

- (I, J) satisfies $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ if whenever I satisfies $\varphi(\bar{a}, \bar{b})$, there is a tuple $\bar{c}$ such that J satisfies $\psi(\bar{a}, \bar{c})$

Relational data exchange problem

Fixed: $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$

Problem: Given instance I of $\mathbf{S}$, find an instance J of $\mathbf{T}$ such that (I, J) satisfies $\Sigma_{\mathbf{ST}}$

- (I, J) satisfies $\varphi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ if whenever I satisfies $\varphi(\bar{a}, \bar{b})$, there is a tuple $\bar{c}$ such that J satisfies $\psi(\bar{a}, \bar{c})$

Notation

J is a solution for I under $\mathcal{M}$

- $\mathrm{Sol}_{\mathcal{M}}(I)$: Set of solutions for I under $\mathcal{M}$

The notion of solution: First example

Example

Consider mapping $\mathcal{M}$ specified by:

$$book(x_1, x_2, y_1) \rightarrow \exists z_1\, writer(x_2, x_1, z_1)$$

Given I:

book	*title*	*author_name*	*affiliation*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

The notion of solution: First example

Example

Consider mapping $\mathcal{M}$ specified by:

$$book(x_1, x_2, y_1) \rightarrow \exists z_1\, writer(x_2, x_1, z_1)$$

Given I:

book	title	author_name	affiliation
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

Solution J_1:

writer	name	book_title	year
	Hungerford	Algebra	1974
	Royden	Real Analysis	1988

The notion of solution: First example

Example

Consider mapping $\mathcal{M}$ specified by:

$$book(x_1, x_2, y_1) \rightarrow \exists z_1\, writer(x_2, x_1, z_1)$$

Given I:

book	*title*	*author_name*	*affiliation*
	Algebra	Hungerford	U. Washington
	Real Analysis	Royden	Stanford

Solution J_1:

writer	*name*	*book_title*	*year*
	Hungerford	Algebra	1974
	Royden	Real Analysis	1988

Solution J_2:

writer	*name*	*book_title*	*year*
	Hungerford	Algebra	n_1
	Royden	Real Analysis	n_2

The notion of solution: Second example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

The notion of solution: Second example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: $dept(Peter, 1)$

The notion of solution: Second example

Example

- **S**: *employee*(*name*)
- **T**: *dept*(*name*, *number*)
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: $dept(Peter, 1)$

J_2: $dept(Peter, 1)$, $dept(Peter, 2)$

The notion of solution: Second example

Example

- $\mathbf{S}$: $employee(name)$
- $\mathbf{T}$: $dept(name,\ number)$
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y\, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: $dept(Peter{,}1)$

J_2: $dept(Peter{,}1)$, $dept(Peter{,}2)$

J_3: $dept(Peter{,}1)$, $dept(John{,}1)$

The notion of solution: Second example

Example

- $\mathbf{S}$: $\mathit{employee}(\mathit{name})$
- $\mathbf{T}$: $\mathit{dept}(\mathit{name}, \mathit{number})$
- $\Sigma_{\mathbf{ST}}$: $\mathit{employee}(x) \rightarrow \exists y\, \mathit{dept}(x, y)$

Solutions for $I = \{\mathit{employee}(\mathit{Peter})\}$:

J_1: $\mathit{dept}(\mathit{Peter},1)$

J_2: $\mathit{dept}(\mathit{Peter},1)$, $\mathit{dept}(\mathit{Peter},2)$

J_3: $\mathit{dept}(\mathit{Peter},1)$, $\mathit{dept}(\mathit{John},1)$

J_4: $\mathit{dept}(\mathit{Peter},n_1)$

The notion of solution: Second example

Example

- $\mathbf{S}$: $employee(name)$
- $\mathbf{T}$: $dept(name, number)$
- $\Sigma_{\mathbf{ST}}$: $employee(x) \rightarrow \exists y \, dept(x, y)$

Solutions for $I = \{employee(Peter)\}$:

J_1: $dept(Peter, 1)$

J_2: $dept(Peter, 1)$, $dept(Peter, 2)$

J_3: $dept(Peter, 1)$, $dept(John, 1)$

J_4: $dept(Peter, n_1)$

J_5: $dept(Peter, n_1)$, $dept(Peter, n_2)$

Canonical universal solution

Question

What is a good instance to materialize?

Canonical universal solution

Question

What is a good instance to materialize?

Algorithm

Input : $(\mathbf{S}, \mathbf{T}, \Sigma_{\mathbf{ST}})$ and an instance I of $\mathbf{S}$
Output : Canonical universal solution $J^\star$ for I under $\mathcal{M}$

let $J^\star :=$ empty instance of $\mathbf{T}$
for every $\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z})$ in $\Sigma_{\mathbf{ST}}$ **do**
  for every $\bar{a}$, $\bar{b}$ such that I satisfies $\varphi(\bar{a}, \bar{b})$ **do**
    create a fresh tuple $\bar{n}$ of pairwise distinct null values
    insert $\psi(\bar{a}, \bar{n})$ into $J^\star$

Canonical universal solution: Example

Example

Consider mapping $\mathcal{M}$ specified by dependency:

$$employee(x) \quad \rightarrow \quad \exists y \; dept(x, y)$$

Canonical universal solution for $I = \{employee(Peter), employee(John)\}$:

- For $a = Peter$ do
  - Create a fresh null value n_1
  - Insert $dept(Peter, n_1)$ into $J^\star$
- For $a = John$ do
  - Create a fresh null value n_2
  - Insert $dept(John, n_2)$ into $J^\star$

Result: $J^\star = \{dept(Peter, n_1),\ dept(John, n_2)\}$

Query answering in data exchange

Given: Mapping $\mathcal{M}$, source instance I and query Q over the target schema

- What does it mean to answer Q?

Query answering in data exchange

Given: Mapping $\mathcal{M}$, source instance I and query Q over the target schema

- What does it mean to answer Q?

Definition (Certain answers)

$$\text{certain}_{\mathcal{M}}(Q, I) \;=\; \bigcap_{J \text{ is a solution for } I \text{ under } \mathcal{M}} Q(J)$$

Certain answers: Example

Example

Consider mapping $\mathcal{M}$ specified by:

$$employee(x) \quad \rightarrow \quad \exists y \, dept(x, y)$$

Given instance $I = \{employee(Peter)\}$:

$$\begin{array}{lcl} \text{certain}_{\mathcal{M}}(\exists y \, dept(x, y), I) & = & \{Peter\} \\ \text{certain}_{\mathcal{M}}(dept(x, y), I) & = & \emptyset \end{array}$$

How can we compute certain answers?

- Naïve algorithm does not work: infinitely many solutions

How can we compute certain answers?

- Naïve algorithm does not work: infinitely many solutions

Approach proposed in [FKMP03]: **Query Rewriting**

Given a mapping $\mathcal{M}$ and a target query Q, compute a query $Q^\star$ such that for every source instance I with canonical universal solution $J^\star$:

$$\text{certain}_{\mathcal{M}}(Q, I) \quad = \quad Q^\star(J^\star)$$

Query rewriting over the canonical universal solution

Theorem (FKMP03)

Given a mapping $\mathcal{M}$ specified by st-tgds and a union of conjunctive queries Q, there exists a query $Q^\star$ such that for every source instance I with canonical universal solution $J^\star$:

$$\text{certain}_{\mathcal{M}}(Q, I) \quad = \quad Q^\star(J^\star)$$

Query rewriting over the canonical universal solution

Theorem (FKMP03)

Given a mapping $\mathcal{M}$ specified by st-tgds and a union of conjunctive queries Q, there exists a query $Q^\star$ such that for every source instance I with canonical universal solution $J^\star$:

$$\text{certain}_{\mathcal{M}}(Q, I) \quad = \quad Q^\star(J^\star)$$

Proof idea: Assume that $\mathbf{C}(a)$ holds whenever a is a constant.

Then:

$$Q^\star(x_1, \ldots, x_m) \quad = \quad \mathbf{C}(x_1) \wedge \cdots \wedge \mathbf{C}(x_m) \wedge Q(x_1, \ldots, x_m)$$

Query rewriting over the canonical solution: Example

Example

Let $\mathcal{M}$ be specified by:

$$employee(x) \quad \rightarrow \quad \exists y\, dept(x, y)$$

Let $Q_1(x) = \exists y\, dept(x, y)$ and $Q_2(x, y) = dept(x, y)$:

$$\begin{aligned} Q_1^\star(x) &= \mathbf{C}(x) \wedge \exists y\, dept(x, y) \\ Q_2^\star(x, y) &= \mathbf{C}(x) \wedge \mathbf{C}(y) \wedge dept(x, y) \end{aligned}$$

Let $I = \{employee(Peter), employee(John)\}$:

$$J^\star \quad = \quad \{dept(Peter, n_1), dept(John, n_2)\}$$

Then:

$$\begin{aligned} \text{certain}_{\mathcal{M}}(Q_1, I) &= \{Peter, John\} & \qquad Q_1^\star(J^\star) &= \{Peter, John\} \\ \text{certain}_{\mathcal{M}}(Q_2, I) &= \emptyset & \qquad Q_2^\star(J^\star) &= \emptyset \end{aligned}$$

Computing certain answers: Complexity

Data complexity: Data exchange setting and query are considered to be fixed.

- Is this a reasonable assumption?

Corollary (FKMP03)

For mappings given by st-tgds, certain answers for **UCQ** *can be computed in polynomial time (data complexity)*

Relational data exchange: Some lessons learned

Key steps in the development of the area:

- Definition of schema mappings: Precise syntax and semantics
  - Definition of the notion of solution
- Identification of good solutions
- Polynomial time algorithms for materializing good solutions
- Definition of target queries: Precise semantics
- Polynomial time algorithms for computing certain answers for **UCQ**

Outline of the talk

- Relational data exchange
- Translating relational data into RDF
- Metadata management
  - Composition, inverse
- Concluding remarks

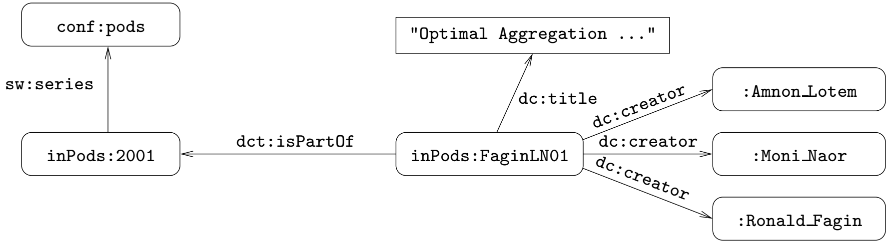
RDF in a nutshell

RDF is the W3C proposed framework for representing information in the Web:

- URI vocabulary
  - A URI is an atomic piece of data, and it identifies an abstract resource
- Syntax based on directed labeled graphs
  - URIs are used as node labels and edge labels
- Schema definition language (RDFS): Define new vocabulary
  - Typing, inheritance of classes and properties, ...
- Formal semantics

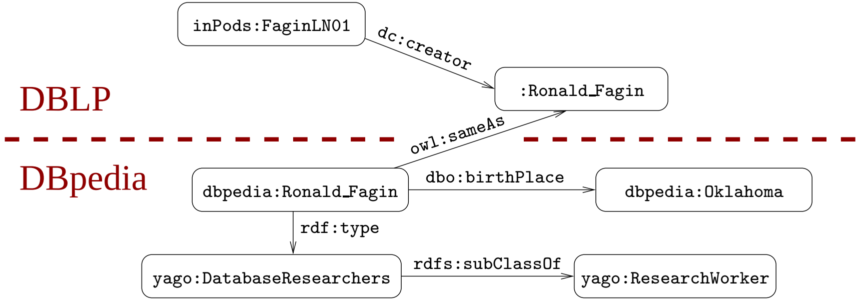
An example of an RDF graph: DBLP

```
      : <http://dblp.l3s.de/d2r/resource/authors/>
   conf: <http://dblp.l3s.de/d2r/resource/conferences/>
 inPods: <http://dblp.l3s.de/d2r/resource/publications/conf/pods/>
     sw: <http://swrc.ontoware.org/ontology#>
     dc: <http://purl.org/dc/elements/1.1/>
    dct: <http://purl.org/dc/terms/>
```

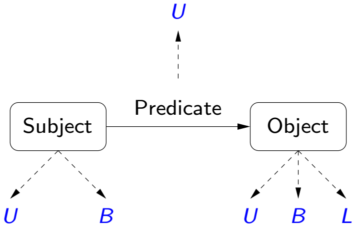


A second example of an RDF graph: DBpedia

```
       : <http://dblp.l3s.de/d2r/resource/authors/>
dbpedia: <http://dbpedia.org/resource/>
    rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
   rdfs: <http://www.w3.org/2000/01/rdf-schema#>
    owl: <http://www.w3.org/2002/07/owl#>
   yago: <http://dbpedia.org/class/yago>
    dbo: <http://dbpedia.org/ontology/>
```



RDF formal model

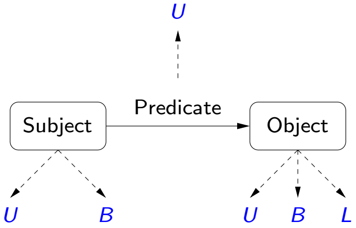


U : set of URIs

B : set of blank nodes

L : set of literals

RDF formal model



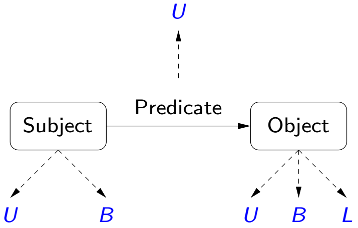
U : set of URIs

B : set of blank nodes

L : set of literals

$(s, p, o) \in (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{B} \cup \mathbf{L})$ is called an RDF triple

RDF formal model



U : set of URIs

B : set of blank nodes

L : set of literals

$(s, p, o) \in (\mathbf{U} \cup \mathbf{B}) \times \mathbf{U} \times (\mathbf{U} \cup \mathbf{B} \cup \mathbf{L})$ is called an RDF triple

A set of RDF triples is called an RDF graph

Data exchange in the RDF world

We have witnessed a constant growth in the amount of RDF data available on the Web

- Also in the number of applications for this data

This has generated an increasing interest in publishing relational data as RDF

- Resulted in the creation of the W3C RDB2RDF Working Group

Data exchange in the RDF world

The problem of translating relational data into RDF can be seen as a data exchange problem

- Schema mappings can be used to describe how the relational data is to be mapped into RDF

We will explore this connection.

- We start by formalizing one of the proposals of the W3C

Data exchange in the RDF world

Some interesting consequences of our study:

- It gives us a mapping language that can be easily extended to deal with RDF-to-RDF data exchange tasks
- It help us in recognizing new problems that should be studied in the area of data exchange

The direct mapping

This mapping is defined in:

> A Direct Mapping of Relational Data to RDF. W3C Working Draft. Editors: M. Arenas, E. Prud'hommeaux and J. Sequeda

The direct mapping defines a default way to translate relational databases into RDF.

- We provide a formalization of this mapping

The direct mapping

Input: A relational schema and a database instance of this schema

Output: An RDF graph

The direct mapping

Input: A relational schema and a database instance of this schema

Output: An RDF graph

We start by describing how the input is specified

Translating relational data into RDF: Running example

Consider the following relational schema:

- *person*(*ssn*, *name*): *ssn* is the primary key
- *student*(*number*, *degree*, *ssn*): *number* is the primary key, *ssn* is a foreign key to *ssn* in *person*

Consider the following instance:

person	*ssn*	*name*
	123	Peter Smith
	456	John Brown
	789	George Taylor

student	*number*	*degree*	*ssn*
	1	CS	123
	2	Math	456

Input: Relational schema

These predicates are used to store a relational schema:

- $\textsc{Rel}(r)$: r is a relation name

Input: Relational schema

These predicates are used to store a relational schema:

- $\text{REL}(r)$: r is a relation name

  Example: $\text{REL}(\texttt{person})$, $\text{REL}(\texttt{student})$

Input: Relational schema

These predicates are used to store a relational schema:

- $\textsc{Rel}(r)$: r is a relation name

  Example: $\textsc{Rel}(\texttt{person})$, $\textsc{Rel}(\texttt{student})$

- $\textsc{Attr}(a, r)$: a is an attribute of relation r

Input: Relational schema

These predicates are used to store a relational schema:

- $\textsc{Rel}(r)$: r is a relation name

  Example: $\textsc{Rel}(\texttt{person})$, $\textsc{Rel}(\texttt{student})$

- $\textsc{Attr}(a, r)$: a is an attribute of relation r

  Example: $\textsc{Attr}(\texttt{ssn}, \texttt{person})$, $\textsc{Attr}(\texttt{name}, \texttt{person})$

Input: Relational schema

- $\mathrm{PK}_n(a_1, \ldots, a_n, r)$: $(a_1, \ldots, a_n)$ $(n \geq 1)$ is a primary key in r

Input: Relational schema

- $\mathrm{PK}_n(a_1, \ldots, a_n, r)$: $(a_1, \ldots, a_n)$ $(n \geq 1)$ is a primary key in r

  Example: $\mathrm{PK}_1(\mathtt{ssn}, \mathtt{person})$

Input: Relational schema

- $\mathrm{PK}_n(a_1, \ldots, a_n, r)$: $(a_1, \ldots, a_n)$ $(n \geq 1)$ is a primary key in r

  Example: $\mathrm{PK}_1(\mathtt{ssn}, \mathtt{person})$

- $\mathrm{FK}_n(a_1, \ldots, a_n, r, b_1, \ldots, b_n, s)$: $(a_1, \ldots, a_n)$ $(n \geq 1)$ is a foreign key in relation r that references to $(b_1, \ldots, b_n)$ in relation s

Input: Relational schema

- $\mathrm{PK}_n(a_1, \ldots, a_n, r)$: $(a_1, \ldots, a_n)$ $(n \geq 1)$ is a primary key in r

  Example: $\mathrm{PK}_1(\mathtt{ssn}, \mathtt{person})$

- $\mathrm{FK}_n(a_1, \ldots, a_n, r, b_1, \ldots, b_n, s)$: $(a_1, \ldots, a_n)$ $(n \geq 1)$ is a foreign key in relation r that references to $(b_1, \ldots, b_n)$ in relation s

  Example: $\mathrm{FK}_1(\mathtt{ssn}, \mathtt{student}, \mathtt{ssn}, \mathtt{person})$

Input: A database instance

This predicate is used to store the tuples in a database instance:

- $\textsc{Value}(v, a, t, r)$: v is the value of attribute a in a tuple with identifier t in relation r

Input: A database instance

This predicate is used to store the tuples in a database instance:

- $\textsc{Value}(v, a, t, r)$: v is the value of attribute a in a tuple with identifier t in relation r

For example, the following relation:

student	*number*	*degree*	*ssn*
	1	CS	123
	2	Math	456

is stored by using the following facts:

$$\begin{array}{l}
\textsc{Value}(1, \texttt{number}, \texttt{t1}, \texttt{student}) \\
\textsc{Value}(\texttt{CS}, \texttt{degree}, \texttt{t1}, \texttt{student}) \\
\textsc{Value}(123, \texttt{ssn}, \texttt{t1}, \texttt{student}) \\
\textsc{Value}(2, \texttt{number}, \texttt{t2}, \texttt{student}) \\
\textsc{Value}(\texttt{Math}, \texttt{degree}, \texttt{t2}, \texttt{student}) \\
\textsc{Value}(456, \texttt{ssn}, \texttt{t2}, \texttt{student})
\end{array}$$

Generating IRIs

IRIs are an essential component of RDF graphs.

A way to generate IRIs for the produced RDF triples has to be provided.

- IRIs should be generated for relations, attributes and tuples

Generating IRIs

IRIs are an essential component of RDF graphs.

A way to generate IRIs for the produced RDF triples has to be provided.

- IRIs should be generated for relations, attributes and tuples

Assume given a base IRI (`http://exa.org/`), and the following family of built-in predicates ($n \geq 2$):

- $\textsc{Concat}_n(s_1, \ldots, s_n, s)$ holds if s is the concatenation of the strings s_1, …, s_n

Generating IRIs

IRIs are an essential component of RDF graphs.

A way to generate IRIs for the produced RDF triples has to be provided.

- IRIs should be generated for relations, attributes and tuples

Assume given a base IRI (`http://exa.org/`), and the following family of built-in predicates ($n \geq 2$):

- $\textsc{Concat}_n(s_1, \ldots, s_n, s)$ holds if s is the concatenation of the strings s_1, ..., s_n
- It can be defined by using the usual $\textsc{Concat}(\cdot, \cdot, \cdot)$

Generating IRIs for relations

This rule generates IRIs for relations:

$$\text{RelationIRI}(X, Y) \quad \leftarrow \quad \text{Rel}(X), \text{Concat}_2(\texttt{http://exa.org/}, X, Y)$$

Generating IRIs for relations

This rule generates IRIs for relations:

$$\textsc{RelationIRI}(X, Y) \quad \leftarrow \quad \textsc{Rel}(X), \textsc{Concat}_2(\texttt{http://exa.org/}, X, Y)$$

Example

http://exa.org/person and http://exa.org/student

Generating IRIs for attributes

The following family of rules generates IRIs for attributes ($n \geq 1$):

$$\begin{aligned}
&\textsc{AttrIRI}_n(X_1, \ldots, X_n, Y, Z) \leftarrow \\
&\qquad\qquad \textsc{Rel}(Y), \textsc{Attr}(X_1, Y), \ldots, \textsc{Attr}(X_n, Y), \\
&\qquad\qquad \textsc{Concat}_{2+2n}(\texttt{http://exa.org/}, Y, \texttt{"\#"}, X_1, \texttt{","}, \\
&\qquad\qquad\qquad\qquad X_2, \texttt{","}, \ldots, \texttt{","}, X_n, Z)
\end{aligned}$$

Generating IRIs for attributes

The following family of rules generates IRIs for attributes ($n \geq 1$):

$$
\begin{aligned}
\textsc{AttrIRI}_n(X_1, \ldots, X_n, Y, Z) \leftarrow \\
& \textsc{Rel}(Y), \textsc{Attr}(X_1, Y), \ldots, \textsc{Attr}(X_n, Y), \\
& \textsc{Concat}_{2+2n}(\texttt{http://exa.org/}, Y, \texttt{"\#"}, X_1, \texttt{","}, \\
& \qquad X_2, \texttt{","}, \ldots, \texttt{","}, X_n, Z)
\end{aligned}
$$

Example

- `http://example.org/student#number` is generated for attribute `number` in relation `student`
- `http://example.org/student#number,degree,ssn` is generated for attributes `number`, `degree`, `ssn` in relation `student`

Generating IRIs for tuples

The following family of rules generates IRIs for tuples ($n \geq 1$):

$$
\begin{aligned}
\textsc{TupleID}(X, Y, Z) \ \leftarrow\ & \\
& \textsc{Rel}(Y), \mathrm{PK}_n(X_1, \ldots, X_n, Y), \\
& \textsc{Value}(V_1, X_1, X, Y), \ldots, \textsc{Value}(V_n, X_n, X, Y), \\
& \textsc{Concat}_{2+4n}(\texttt{http://exa.org/}, Y, \texttt{"\#"}, X_1, \texttt{"="}, V_1, \texttt{","}, \\
& \qquad X_2, \texttt{"="}, V_2, \ldots, \texttt{","}, X_n, \texttt{"="}, V_n, Z)
\end{aligned}
$$

Generating IRIs for tuples

The following family of rules generates IRIs for tuples ($n \geq 1$):

$$
\begin{aligned}
\text{TupleID}(X, Y, Z) \leftarrow\ & \\
& \text{Rel}(Y), \text{PK}_n(X_1, \ldots, X_n, Y), \\
& \text{Value}(V_1, X_1, X, Y), \ldots, \text{Value}(V_n, X_n, X, Y), \\
& \text{Concat}_{2+4n}(\texttt{http://exa.org/}, Y, \texttt{"\#"}, X_1, \texttt{"="}, V_1, \texttt{","}, \\
& \qquad X_2, \texttt{"="}, V_2, \ldots, \texttt{","}, X_n, \texttt{"="}, V_n, Z)
\end{aligned}
$$

Example

- `http://exa.org/student#number=1` is generated for tuple `t1` in relation student
  - Recall that $\text{PK}_1(\texttt{number}, \texttt{student})$ and $\text{Value}(1, \texttt{number}, \texttt{t1}, \texttt{student})$ hold in our running example

Generating IRIs for tuples

One extra case need to be considered: Some relations may not have a primary key.

$$
\begin{array}{rcl}
\textsc{HasPK}(X) & \leftarrow & \mathrm{PK}_n(X_1, \ldots, X_n, X) \qquad (n \geq 1) \\
\textsc{TupleID}(X, Y, Z) & \leftarrow & \textsc{Rel}(Y), \textsc{Value}(V, A, X, Y), \neg\textsc{HasPK}(X), \\
 & & \textsc{Concat}_3(\_:, Y, \_, X, Z)
\end{array}
$$

Generating IRIs for tuples

One extra case need to be considered: Some relations may not have a primary key.

$$
\begin{array}{rcl}
\textsc{HasPK}(X) & \leftarrow & \mathrm{PK}_n(X_1, \ldots, X_n, X) \qquad (n \geq 1) \\
\textsc{TupleID}(X, Y, Z) & \leftarrow & \textsc{Rel}(Y), \textsc{Value}(V, A, X, Y), \neg\textsc{HasPK}(X), \\
 & & \textsc{Concat}_3(\_:, Y, \_, X, Z)
\end{array}
$$

Example

If `student` does not have a primary key, then the following blank node would be the identifier of tuple `t1`:

`_:student_t1`

Defining the mapping

We have the necessary ingredients to introduce the rules that define the direct mapping

The mapping generates three types of triples.

- Table triples: For each relation, store the tuples that belong to it
- Literal triples: For each tuple, store the values in each of its attributes
- Reference triples: Store the references generated by foreign keys

Generating table triples

This rule generates table triples:

$$\begin{aligned}\textsc{Triple}(S, \texttt{rdf:type}, O) \ \leftarrow \\ &\textsc{Rel}(X), \textsc{Value}(V, A, Y, X), \\ &\textsc{TupleID}(Y, X, S), \textsc{RelationIRI}(X, O)\end{aligned}$$

Generating table triples

This rule generates table triples:

$$\begin{aligned}\textsc{Triple}(S, \mathtt{rdf\!:\!type}, O) \ \leftarrow\ & \\ & \textsc{Rel}(X), \textsc{Value}(V, A, Y, X), \\ & \textsc{TupleID}(Y, X, S), \textsc{RelationIRI}(X, O)\end{aligned}$$

Example

The following triples are generated for relation `student`:

$$\begin{aligned}&\textsc{Triple}(\mathtt{http://exa.org/student\#number\!=\!1},\ \mathtt{rdf\!:\!type}, \\ &\qquad \mathtt{http://exa.org/student}) \\ &\textsc{Triple}(\mathtt{http://exa.org/student\#number\!=\!2},\ \mathtt{rdf\!:\!type}, \\ &\qquad \mathtt{http://exa.org/student})\end{aligned}$$

Generating literal triples

The following rule generates literal triples:

$$
\begin{aligned}
\textsc{Triple}(S, P, O) \ \leftarrow& \\
& \textsc{Rel}(X), \textsc{Value}(O, A, Y, X), \\
& \textsc{TupleID}(Y, X, S), \textsc{AttrIRI}(A, X, P)
\end{aligned}
$$

Generating literal triples

The following rule generates literal triples:

$$
\begin{aligned}
\textsc{Triple}(S, P, O) \;\leftarrow\; & \\
& \textsc{Rel}(X), \textsc{Value}(O, A, Y, X), \\
& \textsc{TupleID}(Y, X, S), \textsc{AttrIRI}(A, X, P)
\end{aligned}
$$

Example

The following triples are generated from facts $\textsc{Value}(1, \texttt{number}, \texttt{t1}, \texttt{student})$ and $\textsc{Value}(\texttt{CS}, \texttt{degree}, \texttt{t1}, \texttt{student})$:

$$
\begin{aligned}
& \textsc{Triple}(\texttt{http://exa.org/student\#number=1}, \\
& \qquad \texttt{http://exa.org/student\#number},\ \texttt{1}) \\
& \textsc{Triple}(\texttt{http://exa.org/student\#number=1}, \\
& \qquad \texttt{http://exa.org/student\#degree},\ \texttt{CS})
\end{aligned}
$$

Generating literal triples: A modification

Relational databases with null values have to be consider.

Recall that $\mathbf{C}(a)$ holds if a is a constant

- $\mathbf{C}(a)$ holds if a is not the null value

Generating literal triples: A modification

Relational databases with null values have to be consider.

Recall that $\mathbf{C}(a)$ holds if a is a constant

- $\mathbf{C}(a)$ holds if a is not the null value

The following is the actual rule used to generate literal triples:

$$\begin{aligned} \textsc{Triple}(S, P, O) \ \leftarrow \\ & \textsc{Rel}(X), \textsc{Value}(O, A, Y, X), \mathbf{C}(O), \\ & \textsc{TupleID}(Y, X, S), \textsc{AttrIRI}(A, X, P) \end{aligned}$$

Generating reference triples

This family of rules is used to generate reference triples ($n \geq 1$):

$$
\begin{aligned}
\textsc{Triple}(S, P, O) \ \leftarrow & \\
& \mathrm{FK}_n(X_1, \ldots, X_n, X, Y_1, \ldots, Y_n, Y), \\
& \textsc{Value}(V_1, X_1, U, X), \ldots, \textsc{Value}(V_n, X_n, U, X), \\
& \mathbf{C}(V_1), \ldots, \mathbf{C}(V_n), \\
& \textsc{Value}(V_1, Y_1, W, Y), \ldots, \textsc{Value}(V_n, Y_n, W, Y), \\
& \textsc{TupleID}(U, X, S), \\
& \textsc{AttrIRI}(X_1, \ldots, X_n, X, P), \\
& \textsc{TupleID}(W, Y, O)
\end{aligned}
$$

Generating reference triples

Example

Recall that attribute ssn is a foreign key in relation student that references the attribute ssn in relation person.

Then from the facts $\text{VALUE}(123, \texttt{ssn}, \texttt{t1}, \texttt{student})$ and $\text{VALUE}(123, \texttt{ssn}, \texttt{t3}, \texttt{person})$, the following triple is generated:

$$\text{TRIPLE}(\texttt{http://exa.org/student\#number=1}, \texttt{http://exa.org/student\#ssn}, \texttt{http://exa.org/person\#ssn=123})$$

What we have learned?

We have a mapping language that:

- can be used to encode the direct mapping proposed by the W3C
- can be used by a user to express her/his own rules for translating relational data into RDF
- can be easily extended to deal with RDF-to-RDF data exchange tasks

What we have learned?

We have a mapping language that:

- can be used to encode the direct mapping proposed by the W3C
- can be used by a user to express her/his own rules for translating relational data into RDF
- can be easily extended to deal with RDF-to-RDF data exchange tasks

What we have learned?

We have a mapping language that:

- can be used to encode the direct mapping proposed by the W3C
- can be used by a user to express her/his own rules for translating relational data into RDF
- can be easily extended to deal with RDF-to-RDF data exchange tasks

What we have learned?

We have a mapping language that:

- can be used to encode the direct mapping proposed by the W3C
- can be used by a user to express her/his own rules for translating relational data into RDF
- can be easily extended to deal with RDF-to-RDF data exchange tasks

The direct mapping from a relational data exchange point of view

Semantics of the translation process can be defined as in the relational case.

- This is appropriate for the open-world semantics of RDF
- We are just interested in materializing the canonical universal solution

The direct mapping from a relational data exchange point of view

Semantics of the translation process can be defined as in the relational case.

- This is appropriate for the open-world semantics of RDF
- We are just interested in materializing the canonical universal solution

But some new problems need to be addressed.

Relational data exchange: Some issues to consider

Raised issues that we know how to address:

- Mapping rules may contain constants
- Mapping rules may need to use negation in the left-hand side
- Mapping rules may need to generate fresh identifiers (IRIs)
  - Second-order tuple-generating dependencies

Relational data exchange: Some issues to consider

Raised issues that we know how to address:

- Mapping rules may contain constants
- Mapping rules may need to use negation in the left-hand side
- Mapping rules may need to generate fresh identifiers (IRIs)
  - Second-order tuple-generating dependencies

Relational data exchange: Some issues to consider

Raised issues that we know how to address:

- Mapping rules may contain constants
- Mapping rules may need to use negation in the left-hand side
- Mapping rules may need to generate fresh identifiers (IRIs)
  - Second-order tuple-generating dependencies

Relational data exchange: Some issues to consider

Raised issues that we know how to address:

- Mapping rules may contain constants
- Mapping rules may need to use negation in the left-hand side
- Mapping rules may need to generate fresh identifiers (IRIs)
  - Second-order tuple-generating dependencies

Relational data exchange: Some issues to consider

Raised issues that we know how to address:

- Mapping rules may contain constants
- Mapping rules may need to use negation in the left-hand side
- Mapping rules may need to generate fresh identifiers (IRIs)
  - Second-order tuple-generating dependencies

Relational data exchange: Some issues to consider

Raised issues that are more complex:

- Mapping rules may need to use built-in predicates
- Source instances may contain null values
  - What is the semantics of null values in a relational database? There is a value but it is not known, or there is no value
  - How null values should be treated in a data exchange system? See [APR11] for the relational case
- Keys and foreign keys have to be translated

Relational data exchange: Some issues to consider

Raised issues that are more complex:

- Mapping rules may need to use built-in predicates
- Source instances may contain null values
  - What is the semantics of null values in a relational database? There is a value but it is not known, or there is no value
  - How null values should be treated in a data exchange system? See [APR11] for the relational case
- Keys and foreign keys have to be translated

Relational data exchange: Some issues to consider

Raised issues that are more complex:

- Mapping rules may need to use built-in predicates
- Source instances may contain null values
  - What is the semantics of null values in a relational database? There is a value but it is not known, or there is no value
  - How null values should be treated in a data exchange system? See [APR11] for the relational case
- Keys and foreign keys have to be translated

Relational data exchange: Some issues to consider

Raised issues that are more complex:

- Mapping rules may need to use built-in predicates
- Source instances may contain null values
  - What is the semantics of null values in a relational database?
    There is a value but it is not known, or there is no value
  - How null values should be treated in a data exchange system?
    See [APR11] for the relational case
- Keys and foreign keys have to be translated

Relational data exchange: Some issues to consider

Raised issues that are more complex:

- Mapping rules may need to use built-in predicates
- Source instances may contain null values
  - What is the semantics of null values in a relational database? There is a value but it is not known, or there is no value
  - How null values should be treated in a data exchange system? See [APR11] for the relational case
- Keys and foreign keys have to be translated

Relational data exchange: Some issues to consider

Raised issues that are more complex:

- Mapping rules may need to use built-in predicates
- Source instances may contain null values
  - What is the semantics of null values in a relational database? There is a value but it is not known, or there is no value
  - How null values should be treated in a data exchange system? See [APR11] for the relational case
- Keys and foreign keys have to be translated

Relational data exchange: Some issues to consider

Raised issues that are more complex:

- Mapping rules may need to use built-in predicates
- Source instances may contain null values
  - What is the semantics of null values in a relational database? There is a value but it is not known, or there is no value
  - How null values should be treated in a data exchange system? See [APR11] for the relational case
- Keys and foreign keys have to be translated

Relational data exchange: Some issues to consider

Raised issues that are more complex:

- Mapping rules may need to use built-in predicates
- Source instances may contain null values
  - What is the semantics of null values in a relational database? There is a value but it is not known, or there is no value
  - How null values should be treated in a data exchange system? See [APR11] for the relational case
- Keys and foreign keys have to be translated

An important question about the direct mapping

Is the direct mapping information preserving?

More generally: Is a mapping defined in the language just presented information preserving?

- How much of the initial information is preserved?
- How much of the initial instance can be reconstructed?

This fundamental issue has been studied in the context of relational data exchange.

- We will give a brief introduction to the theory that can help to answer this type of questions: Metadata management

An important question about the direct mapping

Is the direct mapping information preserving?

More generally: Is a mapping defined in the language just presented information preserving?

- How much of the initial information is preserved?
- How much of the initial instance can be reconstructed?

This fundamental issue has been studied in the context of relational data exchange.

- We will give a brief introduction to the theory that can help to answer this type of questions: Metadata management

An important question about the direct mapping

Is the direct mapping information preserving?

More generally: Is a mapping defined in the language just presented information preserving?

- How much of the initial information is preserved?
- How much of the initial instance can be reconstructed?

This fundamental issue has been studied in the context of relational data exchange.

- We will give a brief introduction to the theory that can help to answer this type of questions: Metadata management

An important question about the direct mapping

Is the direct mapping information preserving?

More generally: Is a mapping defined in the language just presented information preserving?

- How much of the initial information is preserved?
- How much of the initial instance can be reconstructed?

This fundamental issue has been studied in the context of relational data exchange.

- We will give a brief introduction to the theory that can help to answer this type of questions: Metadata management

An important question about the direct mapping

Is the direct mapping information preserving?

More generally: Is a mapping defined in the language just presented information preserving?

- How much of the initial information is preserved?
- How much of the initial instance can be reconstructed?

This fundamental issue has been studied in the context of relational data exchange.

- We will give a brief introduction to the theory that can help to answer this type of questions: Metadata management

An important question about the direct mapping

Is the direct mapping information preserving?

More generally: Is a mapping defined in the language just presented information preserving?

- How much of the initial information is preserved?
- How much of the initial instance can be reconstructed?

This fundamental issue has been studied in the context of relational data exchange.

- We will give a brief introduction to the theory that can help to answer this type of questions: Metadata management

Outline of the talk

- Relational data exchange
- Translating relational data into RDF
- Metadata management
  - Composition, inverse
- Concluding remarks

Relational data exchange: Some practical lesson learned

Creating schema mappings is a time consuming and expensive process

- Manual or semi-automatic process in general

Relational data exchange: Some practical lesson learned

Creating schema mappings is a time consuming and expensive process

- Manual or semi-automatic process in general

Key question: Can we reuse existing schema mapping?

Can we reuse schema mappings?

$$\mathbf{S} \xrightarrow[\Sigma_{\mathbf{ST}}]{} \mathbf{T}$$

Can we reuse schema mappings?

$$\mathbf{S} \xrightarrow[\Sigma_{\mathbf{ST}}]{} \mathbf{T} \xrightarrow[\Sigma_{\mathbf{TU}}]{} \mathbf{U}$$

Can we reuse schema mappings?

$$
\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[rr, bend right, "\Sigma_{\mathbf{SU}}\,?"'] & \mathbf{T} \arrow[r, "\Sigma_{\mathbf{TU}}"'] & \mathbf{U}
\end{tikzcd}
$$

Can we reuse schema mappings?

$$\begin{tikzcd}[column sep=huge] \mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[rr, bend right, "\Sigma_{\mathbf{SU}}\,?"'] & \mathbf{T} \arrow[r, "\Sigma_{\mathbf{TU}}"'] & \mathbf{U} \end{tikzcd}$$

We need some operators for schema mappings

Can we reuse schema mappings?

$$
\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[rr, bend right] & \mathbf{T} \arrow[r, "\Sigma_{\mathbf{TU}}"'] & \mathbf{U}
\end{tikzcd}
$$

$$\Sigma_{\mathbf{SU}} = \Sigma_{\mathbf{ST}} \circ \Sigma_{\mathbf{TU}}$$

We need some operators for schema mappings

- **Composition** in the above case

Metadata management

Contributions mentioned in the previous slides are just a first step towards the development of a general framework for data exchange.

In fact, as pointed in [B03],

> many information system problems involve not only the design and integration of complex application artifacts, but also their subsequent manipulation.

Metadata management

This has motivated the need for the development of a general infrastructure for managing schema mappings.

The problem of managing schema mappings is called **metadata management.**

High-level algebraic operators, such as compose, are used to manipulate mappings.

- What other operators are needed?

More operators are needed

$$\mathbf{S} \xrightarrow[\Sigma_{\mathbf{ST}}]{} \mathbf{T}$$

More operators are needed

$$\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & \mathbf{T} \\
\mathbf{V} &
\end{tikzcd}$$

More operators are needed

$$\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & \mathbf{T} \\
\mathbf{V} \arrow[u, bend left, "\Sigma_{\mathbf{VS}}?"]
\end{tikzcd}$$

More operators are needed

$$
\begin{tikzcd}[row sep=huge, column sep=huge]
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & \mathbf{T} \\
\mathbf{V} \arrow[u, bend left, "\Sigma_{\mathbf{VS}} = \Sigma_{\mathbf{SV}}^{-1}"]
\end{tikzcd}
$$

An **inverse** operator is needed in this case

More operators are needed

$$
\begin{tikzcd}
\mathbf{S} \arrow[rr, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & & \mathbf{T} \\
\mathbf{V} \arrow[u, bend left, "\Sigma_{\mathbf{VS}} = \Sigma_{\mathbf{SV}}^{-1}"] \arrow[urr, bend right, "\Sigma_{\mathbf{SV}}^{-1} \circ \Sigma_{\mathbf{ST}}"'] & &
\end{tikzcd}
$$

An **inverse** operator is needed in this case

- Combined with the composition operator

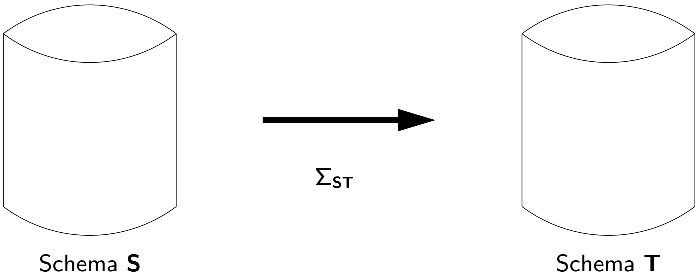
More operators are needed

$$
\begin{tikzcd}
\mathbf{S} \arrow[r, "\Sigma_{\mathbf{ST}}"'] \arrow[d, "\Sigma_{\mathbf{SV}}"] & \mathbf{T} \arrow[r, "\Sigma_{\mathbf{TU}}"'] & \mathbf{U} \\
\mathbf{V} \arrow[u, bend left, "\Sigma_{\mathbf{VS}} = \Sigma_{\mathbf{SV}}^{-1}"] \arrow[ur, bend right, "\Sigma_{\mathbf{SV}}^{-1} \circ \Sigma_{\mathbf{ST}}"'] \arrow[urr, bend right, "(\Sigma_{\mathbf{SV}}^{-1} \circ \Sigma_{\mathbf{ST}}) \circ \Sigma_{\mathbf{TU}}"'] & &
\end{tikzcd}
$$

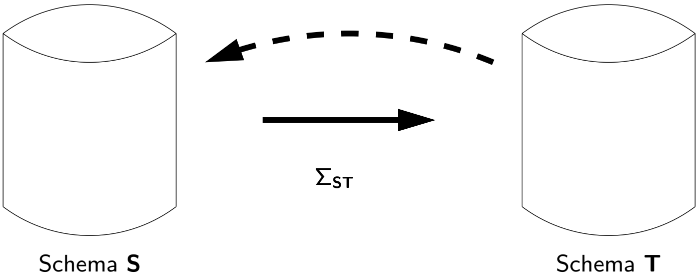
An **inverse** operator is needed in this case

- Combined with the composition operator

The inverse operator: How much of the initial instance can be reconstructed?



The inverse operator: How much of the initial instance can be reconstructed?



Defining the inverse operator: The composition operator

Notation

We can view a mapping $\mathcal{M}$ as a set of pairs:

$$(I, J) \in \mathcal{M} \quad \text{iff} \quad J \in \mathsf{Sol}_{\mathcal{M}}(I)$$

Defining the inverse operator: The composition operator

Notation

We can view a mapping $\mathcal{M}$ as a set of pairs:

$$(I, J) \in \mathcal{M} \quad \text{iff} \quad J \in \mathrm{Sol}_{\mathcal{M}}(I)$$

Definition (FKPT04)

Let $\mathcal{M}_{12}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $\mathcal{M}_{23}$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_3$:

$$\begin{aligned} \mathcal{M}_{12} \circ \mathcal{M}_{23} \ = \ \{(I_1, I_3) \mid \\ \exists I_2 : (I_1, I_2) \in \mathcal{M}_{12} \text{ and } (I_2, I_3) \in \mathcal{M}_{23}\} \end{aligned}$$

The inverse operator

Question

What is the semantics of the inverse operator?

This turns out to be a very difficult question.

We consider three notions of inverse here:

- Fagin-inverse
- Quasi-inverse
- Maximum recovery

The notion of Fagin-inverse

Intuition: A mapping composed with its inverse should be equal to the identity mapping

The notion of Fagin-inverse

Intuition: A mapping composed with its inverse should be equal to the identity mapping

What is the identity mapping?

- $\mathsf{Id}_{\mathbf{S}} \; = \; \{(I, I) \mid I \text{ is an instance of } \mathbf{S}\}$?

The notion of Fagin-inverse

Intuition: A mapping composed with its inverse should be equal to the identity mapping

What is the identity mapping?

- $\text{Id}_{\mathbf{S}} \ = \ \{(I, I) \mid I \text{ is an instance of } \mathbf{S}\}$?

For mapping specified by st-tgds, $\text{Id}_{\mathbf{S}}$ is not the right notion.

- $\overline{\text{Id}}_{\mathbf{S}} \ = \ \{(I_1, I_2) \mid I_1, I_2 \text{ are instances of } \mathbf{S} \text{ and } I_1 \subseteq I_2\}$

The notion of Fagin-inverse: Formal definition

Definition (F06)

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $\mathcal{M}^\star$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{M}^\star$ is a Fagin-inverse of $\mathcal{M}$ if:

$$\mathcal{M} \circ \mathcal{M}^\star \;=\; \overline{\mathsf{Id}}_{\mathbf{S}_1}$$

The notion of Fagin-inverse: Formal definition

Definition (F06)

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$, and $\mathcal{M}^\star$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{M}^\star$ is a Fagin-inverse of $\mathcal{M}$ if:

$$\mathcal{M} \circ \mathcal{M}^\star \quad = \quad \overline{\mathsf{Id}}_{\mathbf{S}_1}$$

Example

Consider mapping $\mathcal{M}$ specified by:

$$A(x) \quad \rightarrow \quad R(x) \wedge \exists y\, S(x, y)$$

Then the following are Fagin-inverses of $\mathcal{M}$:

$$\begin{array}{lcl} \mathcal{M}_1^\star & : & R(x) \rightarrow A(x) \\ \mathcal{M}_2^\star & : & S(x, y) \rightarrow A(x) \end{array}$$

Is Fagin-inverse the right notion of inverse for mappings?

On the positive side: It is a natural notion

- With good computational properties

On the negative side: A mapping specified by st-tgds is not guaranteed to admit a Fagin-inverse

- For example: Mapping specified by $A(x, y) \rightarrow R(x)$ does not admit a Fagin-inverse

In fact: This notion turns out to be rather restrictive, as it is rare that a schema mapping possesses a Fagin-inverse.

Is Fagin-inverse the right notion of inverse for mappings?

The notion of quasi-inverse was introduced in [FKPT07] to overcome this limitation.

- The idea is to relax the notion of Fagin-inverse by not differentiating between source instances that are equivalent for data exchange purposes

Numerous non-Fagin-invertible mappings possess natural and useful quasi-inverses.

- But there are still simple mappings specified by st-tgds that have no quasi-inverse

The notion of maximum recovery overcome this limitation.

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\mathcal{M}_1^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \exists v \; emp(x, u, v)$$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\mathcal{M}_1^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \exists v\, emp(x, u, v) \qquad \checkmark$$

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\begin{array}{llcl} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v \, emp(x, u, v) \quad \checkmark \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, u, z) \end{array}$$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\mathcal{M}_1^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \exists v \, emp(x, u, v) \quad \checkmark$$
$$\mathcal{M}_2^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \, emp(x, u, z) \quad \checkmark$$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\begin{array}{llcll} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) & \checkmark \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) & \checkmark \\ \mathcal{M}_3^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, z, u) & \end{array}$$

Recovery: specifies how to recover sound information

Data may be lost in the exchange through a mapping $\mathcal{M}$

- We would like to find a mapping $\mathcal{M}^\star$ that at least recovers sound data w.r.t. $\mathcal{M}$
  - $\mathcal{M}^\star$ is called a recovery of $\mathcal{M}$

Example

Consider a mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

What mappings are recoveries of $\mathcal{M}$?

$$\begin{array}{llcll}
\mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v \, emp(x, u, v) & \checkmark \\
\mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, u, z) & \checkmark \\
\mathcal{M}_3^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, z, u) & \times
\end{array}$$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{llcl} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \end{array}$$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{llcl} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v\, emp(x, u, v) \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u\, emp(x, u, z) \end{array}$$

Intuitively: $\mathcal{M}_2^\star$ is better than $\mathcal{M}_1^\star$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{llcl} \mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v \, emp(x, u, v) \\ \mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, u, z) \\ \mathcal{M}_4^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, u, z) \wedge u \neq z \end{array}$$

Intuitively: $\mathcal{M}_2^\star$ is better than $\mathcal{M}_1^\star$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{llcl}
\mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v \, emp(x, u, v) \\
\mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, u, z) \\
\mathcal{M}_4^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, u, z) \wedge u \neq z
\end{array}$$

Intuitively: $\mathcal{M}_2^\star$ is better than $\mathcal{M}_1^\star$
$\mathcal{M}_4^\star$ is better than $\mathcal{M}_2^\star$ and $\mathcal{M}_1^\star$

Maximum recovery: The *most informative* recovery

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

These mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{llll}
\mathcal{M}_1^\star: & shuttle(x, z) & \rightarrow & \exists u \exists v \, emp(x, u, v) \\
\mathcal{M}_2^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, u, z) \\
\mathcal{M}_4^\star: & shuttle(x, z) & \rightarrow & \exists u \, emp(x, u, z) \wedge u \neq z
\end{array}$$

Intuitively: $\mathcal{M}_2^\star$ is better than $\mathcal{M}_1^\star$
$\mathcal{M}_4^\star$ is better than $\mathcal{M}_2^\star$ and $\mathcal{M}_1^\star$

We would like to find a recovery of $\mathcal{M}$ that is better than any other recovery: Maximum recovery

The notion of recovery: Formalization

Definition (APR08)

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$ and $\mathcal{M}^\star$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{M}^\star$ is a recovery of $\mathcal{M}$ if:

for every instance I of $\mathbf{S}_1$: $(I, I) \in \mathcal{M} \circ \mathcal{M}^\star$

The notion of recovery: Formalization

Definition (APR08)

Let $\mathcal{M}$ be a mapping from $\mathbf{S}_1$ to $\mathbf{S}_2$ and $\mathcal{M}^\star$ a mapping from $\mathbf{S}_2$ to $\mathbf{S}_1$. Then $\mathcal{M}^\star$ is a recovery of $\mathcal{M}$ if:

for every instance I of $\mathbf{S}_1$: $(I, I) \in \mathcal{M} \circ \mathcal{M}^\star$

Example

Consider again mapping $\mathcal{M}$ specified by:

$$emp(x, y, z) \wedge y \neq z \quad \rightarrow \quad shuttle(x, z)$$

This mapping is not a recovery of $\mathcal{M}$:

$$\mathcal{M}_3^\star: \quad shuttle(x, z) \quad \rightarrow \quad \exists u \, emp(x, z, u)$$

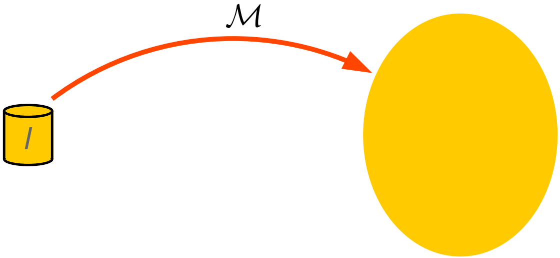
The notion of recovery: Formalization

Example (Cont'd)

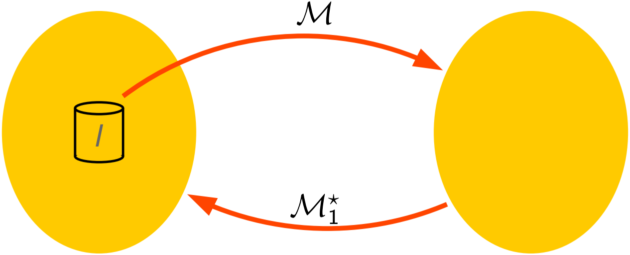
On the other hand, these mappings are recoveries of $\mathcal{M}$:

$$\begin{array}{llcl}
\mathcal{M}_1^\star: & \mathit{shuttle}(x, z) & \rightarrow & \exists u \exists v\, \mathit{emp}(x, u, v) \\
\mathcal{M}_2^\star: & \mathit{shuttle}(x, z) & \rightarrow & \exists u\, \mathit{emp}(x, u, z) \\
\mathcal{M}_4^\star: & \mathit{shuttle}(x, z) & \rightarrow & \exists u\, \mathit{emp}(x, u, z) \wedge u \neq z
\end{array}$$

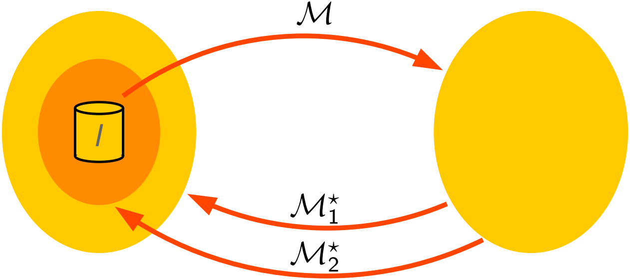
The notion of maximum recovery



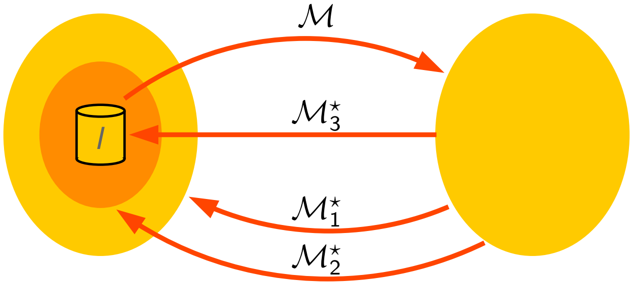
The notion of maximum recovery



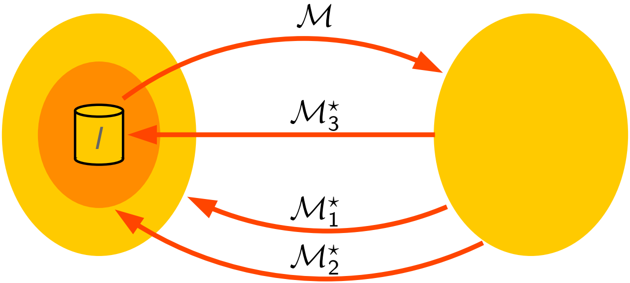
The notion of maximum recovery



The notion of maximum recovery



The notion of maximum recovery



Definition (APR08)

$\mathcal{M}^\star$ is a maximum recovery of $\mathcal{M}$ if:

- $\mathcal{M}^\star$ is a recovery of $\mathcal{M}$
- for every recovery $\mathcal{M}'$ of $\mathcal{M}$: $\mathcal{M} \circ \mathcal{M}^\star \subseteq \mathcal{M} \circ \mathcal{M}'$

On the existence of maximum recoveries

Maximum recoveries overcome one of the limitations of Fagin-inverses and quasi-inverses.

On the existence of maximum recoveries

Maximum recoveries overcome one of the limitations of Fagin-inverses and quasi-inverses.

Theorem (APR08)

Every mapping specified by st-tgds has a maximum recovery.

On the existence of maximum recoveries

Maximum recoveries overcome one of the limitations of Fagin-inverses and quasi-inverses.

Theorem (APR08)

Every mapping specified by st-tgds has a maximum recovery.

Example

Consider a mapping $\mathcal{M}$ specified by:

$$P(x, y) \wedge P(y, z) \quad \rightarrow \quad R(x, z) \wedge T(y)$$

$\mathcal{M}$ has neither an inverse nor a quasi-inverse [FKPT07]. A maximum recovery of $\mathcal{M}$ is specified by:

$$\begin{aligned} R(x, z) \quad &\rightarrow \quad \exists y \, P(x, y) \wedge P(y, z) \\ T(y) \quad &\rightarrow \quad \exists x \exists z \, P(x, y) \wedge P(y, z) \end{aligned}$$

Outline of the talk

- Relational data exchange
- Translating relational data into RDF
- Metadata management
  - Composition, inverse
- Concluding remarks

Concluding remarks

- The problem of exchanging relational data has been extensively studied
- There is an increasing interest in publishing relational data as RDF
- The problem of translating relational data into RDF can be seen as a data exchange problem

Concluding remarks

- We present a mapping language that can be used to formalize the direct mapping proposed by the W3C
  - Can be used by a user to express her/his own rules for translating relational data into RDF
  - Can also be used in RDF-to-RDF data exchange tasks
- This study help us in recognizing some new problems that should be addressed in the area of relational data exchange
- Some results in the area of metadata management can be useful in the study of some fundamental properties of the mapping languages for RDF

Bibliography

[APR08] M. Arenas, J. Pérez, C. Riveros. The recovery of a schema mapping: bringing exchanged data back. PODS 2008: 13-22

[APR11] M. Arenas, J. Pérez, J. Reutter. Data Exchange beyond Complete Data. To appear in PODS 2011.

[B03] P. A. Bernstein. Applying Model Management to Classical Meta Data Problems. CIDR 2003

[FKMP03] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa. Data Exchange: Semantics and Query Answering. ICDT 2003: 207-224

[FKPT04] R. Fagin, P. G. Kolaitis, L. Popa, W.-C. Tan. Composing Schema Mappings: Second-Order Dependencies to the Rescue. PODS 2004: 83-94

[F06] R. Fagin. Inverting schema mappings. PODS 2006: 50-59

[FKPT07] R. Fagin, P. G. Kolaitis, L. Popa, W.-C. Tan. Quasi-inverses of schema mappings. PODS 2007: 123-132