RDF and SPARQL: Database Foundations

Marcelo Arenas, Claudio Gutierrez, Jorge Perez

Department of Computer Science
Pontificia Universidad Católica de Chile
Universidad de Chile

Center for Web Research
http://www.cwr.cl/

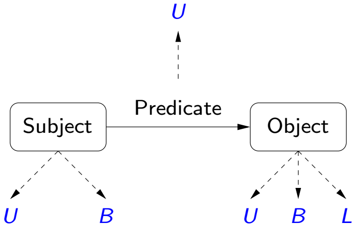
Outline

- Part I: The RDF data model
- Part II: Querying RDF Data
  - Querying: The simple and the ideal
  - Querying: Semantics and Complexity
- Part III: Querying Data with SPARQL
  - Decisions taken
  - Decisions to be taken
- Conclusions

RDF in a nutshell

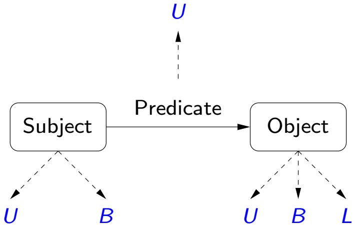
- RDF is the W3C proposal framework for representing information in the Web.
- Abstract syntax based on directed labeled graph.
- Schema definition language (RDFS): Define new vocabulary (typing, inheritance of classes and properties).
- Extensible URI-based vocabulary.
- Support use of XML schema datatypes.
- Formal semantics.

RDF formal model



$$
\begin{aligned}
U &= \text{set of Uris} \\
B &= \text{set of Blank nodes} \\
L &= \text{set of Literals}
\end{aligned}
$$

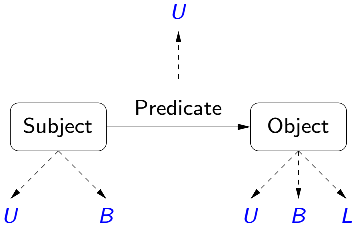
RDF formal model



$$U = \text{set of Uris}$$
$$B = \text{set of Blank nodes}$$
$$L = \text{set of Literals}$$

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

RDF formal model



U = set of Uris

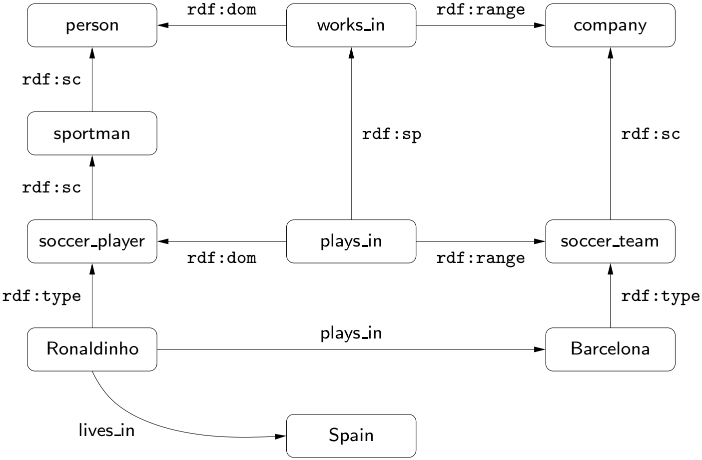
B = set of Blank nodes

L = set of Literals

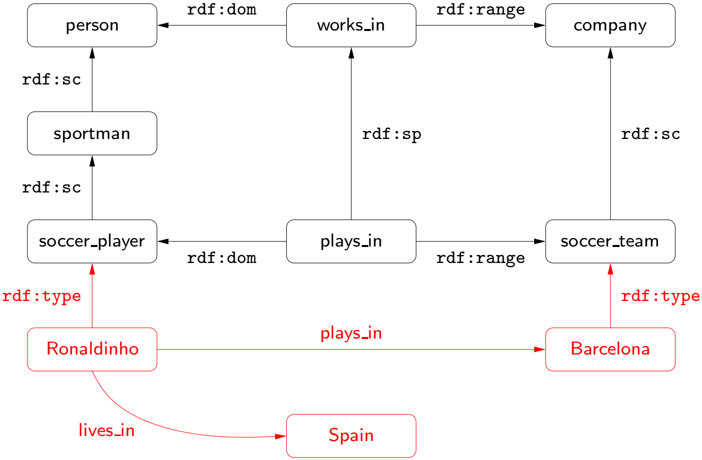
$(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an RDF triple

A set of RDF triples is called an RDF graph

RDFS: An example



RDFS: An example



RDF model

Some difficulties:

- Existential variables as datavalues
- Built-in vocabulary with fixed semantics (RDFS)
- Graph model where nodes may also be edge labels

RDF data processing can take advantage of database techniques:

- Query processing
- Storing
- Indexing

RDF model

Some difficulties:

- Existential variables as datavalues
- Built-in vocabulary with fixed semantics (RDFS)
- Graph model where nodes may also be edge labels

RDF data processing can take advantage of database techniques:

- Query processing
- Storing
- Indexing

Entailment of RDF graphs

Entailment of RDF graphs:

Entailment of RDF graphs

Entailment of RDF graphs:

- Can be defined in terms of classical notions such model, interpretation, etc

Entailment of RDF graphs

Entailment of RDF graphs:

- Can be defined in terms of classical notions such model, interpretation, etc
  - As for the case of first order logic

Entailment of RDF graphs

Entailment of RDF graphs:

- Can be defined in terms of classical notions such model, interpretation, etc
  - As for the case of first order logic
- Has a graph characterization via homomorphisms.

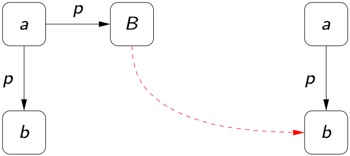
Homomorphism

A function $h : U \cup B \cup L \to U \cup B \cup L$ is a homomorphism h from G_1 to G_2 if:

- $h(c) = c$ for every $c \in U \cup L$;
- for every $(a, b, c) \in G_1$, $(h(a), h(b), h(c)) \in G_2$

Notation: $G_1 \to G_2$

Example: $h = \{B \mapsto b\}$



Entailment

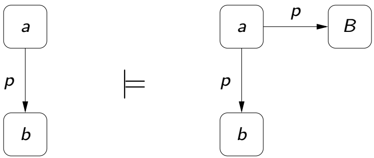
Theorem (CM77)

$G_1 \models G_2$ *if and only if there is a homomorphism* $G_2 \rightarrow G_1$.

Entailment

Theorem (CM77)

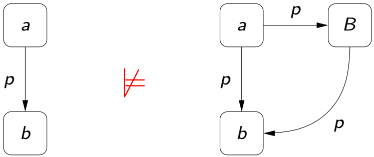
$G_1 \models G_2$ if and only if there is a homomorphism $G_2 \to G_1$.



Entailment

Theorem (CM77)

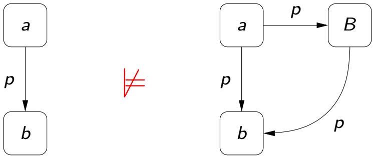
$G_1 \models G_2$ if and only if there is a homomorphism $G_2 \to G_1$.



Entailment

Theorem (CM77)

$G_1 \models G_2$ if and only if there is a homomorphism $G_2 \to G_1$.



Complexity

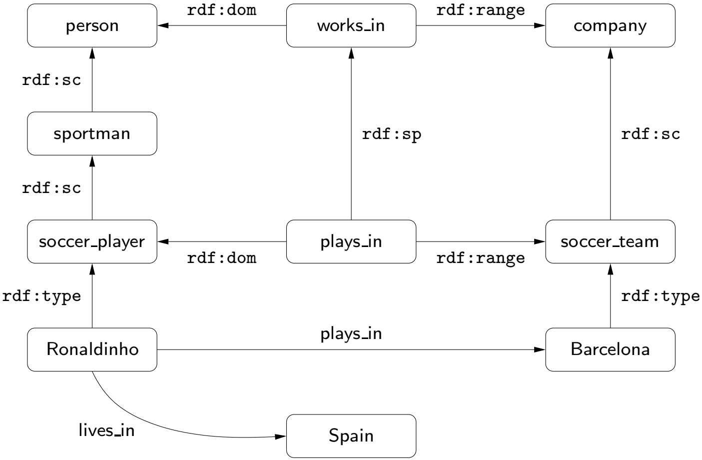
Entailment for RDF is NP-complete

Graphs with RDFS vocabulary

Previous characterization of entailment is not enough to deal with RDFS vocabulary:

Graphs with RDFS vocabulary

Previous characterization of entailment is not enough to deal with RDFS vocabulary: $(\mathtt{Ronaldinho}, \mathtt{rdf:type}, \mathtt{person})$



Graphs with RDFS vocabulary

Built-in predicates have pre-defined semantics:

Graphs with RDFS vocabulary

Built-in predicates have pre-defined semantics:

`rdf:sc`: transitive

Graphs with RDFS vocabulary

Built-in predicates have pre-defined semantics:

- `rdf:sc`: transitive
- `rdf:sp`: transitive

Graphs with RDFS vocabulary

Built-in predicates have pre-defined semantics:

- `rdf:sc`: transitive
- `rdf:sp`: transitive
- More complicated interactions: $\dfrac{(p, \texttt{rdf:dom}, c) \quad (a, p, b)}{(a, \texttt{rdf:type}, c)}$

Built-in predicates have pre-defined semantics:

$\texttt{rdf:sc}$: transitive

$\texttt{rdf:sp}$: transitive

More complicated interactions: $\dfrac{(p, \texttt{rdf:dom}, c) \quad (a, p, b)}{(a, \texttt{rdf:type}, c)}$

RDFS-entailment can be characterized by a set of rules

- An Existential rule
- Subproperty rules
- Subclass rules
- Typing rules
- Implicit typing

Graphs with RDFS vocabulary: Inference rules

Inference system in [MPG07] has 14 rules:

Existential rule :

Subproperty rules :

Subclass rules :

Typing rules :

Implicit typing :

Graphs with RDFS vocabulary: Inference rules

Inference system in [MPG07] has 14 rules:

Existential rule : $\frac{G_1}{G_2}$ if $G_2 \rightarrow G_1$

Subproperty rules :

Subclass rules :

Typing rules :

Implicit typing :

Graphs with RDFS vocabulary: Inference rules

Inference system in [MPG07] has 14 rules:

Existential rule : $\frac{G_1}{G_2}$ if $G_2 \rightarrow G_1$

Subproperty rules : $\frac{(p, \texttt{rdf:sp}, q) \quad (a, p, b)}{(a, q, b)}$

Subclass rules :

Typing rules :

Implicit typing :

Inference system in [MPG07] has 14 rules:

Existential rule : $\frac{G_1}{G_2}$ if $G_2 \rightarrow G_1$

Subproperty rules : $\frac{(p, \texttt{rdf:sp}, q) \quad (a, p, b)}{(a, q, b)}$

Subclass rules : $\frac{(a, \texttt{rdf:sc}, b) \quad (b, \texttt{rdf:sc}, c)}{(a, \texttt{rdf:sc}, c)}$

Typing rules :

Implicit typing :

Inference system in [MPG07] has 14 rules:

Existential rule : $\frac{G_1}{G_2}$ if $G_2 \rightarrow G_1$

Subproperty rules : $\frac{(p, \texttt{rdf:sp}, q) \quad (a, p, b)}{(a, q, b)}$

Subclass rules : $\frac{(a, \texttt{rdf:sc}, b) \quad (b, \texttt{rdf:sc}, c)}{(a, \texttt{rdf:sc}, c)}$

Typing rules : $\frac{(p, \texttt{rdf:dom}, c) \quad (a, p, b)}{(a, \texttt{rdf:type}, c)}$

Implicit typing :

Graphs with RDFS vocabulary: Inference rules

Inference system in [MPG07] has 14 rules:

Existential rule : $\frac{G_1}{G_2}$ if $G_2 \rightarrow G_1$

Subproperty rules : $\dfrac{(p, \texttt{rdf:sp}, q) \quad (a, p, b)}{(a, q, b)}$

Subclass rules : $\dfrac{(a, \texttt{rdf:sc}, b) \quad (b, \texttt{rdf:sc}, c)}{(a, \texttt{rdf:sc}, c)}$

Typing rules : $\dfrac{(p, \texttt{rdf:dom}, c) \quad (a, p, b)}{(a, \texttt{rdf:type}, c)}$

Implicit typing : $\dfrac{(q, \texttt{rdf:dom}, a) \quad (p, \texttt{rdf:sp}, q) \quad (b, p, c)}{(b, \texttt{rdf:type}, a)}$

Inference system in [MPG07] has 14 rules:

Existential rule :

Subproperty rules : $\dfrac{(p, \texttt{rdf:sp}, q) \quad (a, p, b)}{(a, q, b)}$

Subclass rules :

Typing rules : $\dfrac{(p, \texttt{rdf:dom}, c) \quad (a, p, b)}{(a, \texttt{rdf:type}, c)}$

Implicit typing : $\dfrac{(q, \texttt{rdf:dom}, a) \quad (p, \texttt{rdf:sp}, q) \quad (b, p, c)}{(b, \texttt{rdf:type}, a)}$

Graphs with RDFS vocabulary: Inference rules

Inference system in [MPG07] has 14 rules:

Existential rule :

Subproperty rules : $\dfrac{(p, \texttt{rdf:sp}, q) \quad (a, p, b)}{(a, q, b)}$

Subclass rules :

Typing rules : $\dfrac{(p, \texttt{rdf:dom}, c) \quad (a, p, b)}{(a, \texttt{rdf:type}, c)}$

Implicit typing : $\dfrac{(B, \texttt{rdf:dom}, a) \quad (p, \texttt{rdf:sp}, B) \quad (b, p, c)}{(b, \texttt{rdf:type}, a)}$

RDFS Entailment

Theorem (H04,GHM04,MPG07)

$G_1 \models G_2$ iff there is a proof of G_2 from G_1 using the system of 14 inference rules.

Complexity

RDFS-entailment is NP-complete.

Proof idea

Membership in NP: If $G_1 \models G_2$, then there exists a polynomial-size proof of this fact.

Closure of an RDF Graph

Notation:

ground(G)	:	Graph obtained by replacing every blank B in G by a constant c_B.
ground^{-1}(G)	:	Graph obtained by replacing every constant c_B in G by B.

Closure of an RDF graph G (denoted by closure(G)):

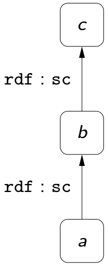
Notation:

ground(G)	:	Graph obtained by replacing every blank B in G by a constant c_B.
ground^{-1}(G)	:	Graph obtained by replacing every constant c_B in G by B.

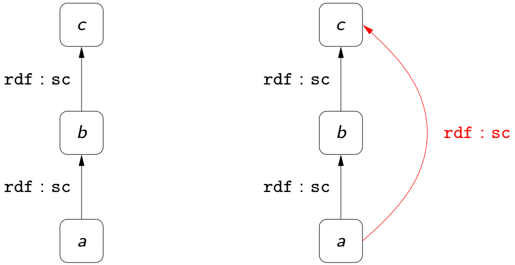
Closure of an RDF graph G (denoted by closure(G)):

$$G \cup \{t \in (U \cup B) \times U \times (U \cup B \cup L) \mid \text{there exists a ground tuple } t' \text{ such that } \text{ground}(G) \models t' \text{ and } t = \text{ground}^{-1}(t')\}$$

Closure of an RDF Graph: Example



Closure of an RDF Graph: Example



Closure of an RDF graph: complexity

Proposition (H04,GHM04,MPG07)

$G_1 \models G_2$ *iff* $G_2 \rightarrow \mathit{closure}(G_1)$

Complexity

The closure of G can be computed in time $O(|G|^4 \cdot \log |G|)$.

Closure of an RDF graph: complexity

Proposition (H04,GHM04,MPG07)

$G_1 \models G_2$ *iff* $G_2 \rightarrow \mathit{closure}(G_1)$

Complexity

The closure of G can be computed in time $O(|G|^4 \cdot \log |G|)$.

Can the closure be used in practice?

Closure of an RDF graph: complexity

Proposition (H04,GHM04,MPG07)

$G_1 \models G_2$ *iff* $G_2 \rightarrow \mathit{closure}(G_1)$

Complexity

The closure of G can be computed in time $O(|G|^4 \cdot \log |G|)$.

Can the closure be used in practice?

- Can we use an alternative materialization?

Closure of an RDF graph: complexity

Proposition (H04,GHM04,MPG07)

$G_1 \models G_2$ *iff* $G_2 \rightarrow \mathit{closure}(G_1)$

Complexity

The closure of G *can be computed in time* $O(|G|^4 \cdot \log |G|)$.

Can the closure be used in practice?

- Can we use an alternative materialization?
- Can we materialize a small part of the closure?

Core of an RDF Graph

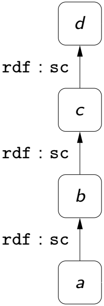
An RDF Graph G is a *core* if there is no homomorphism from G to a proper subgraph of it.

Theorem (HN92,FKP03,GHM04)

- *Each RDF graph G has a unique core (denoted by $core(G)$).*
- *Deciding if G is a core is coNP-complete.*
- *Deciding if $G = core(G')$ is DP-complete.*

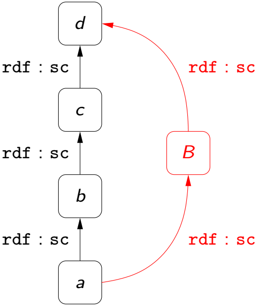
Core and RDFS

For RDF graphs with RDFS vocabulary, the core of G may contain redundant information:



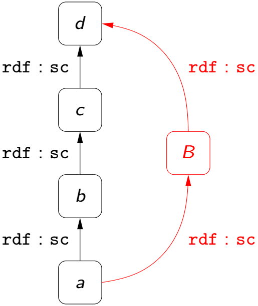
Core and RDFS

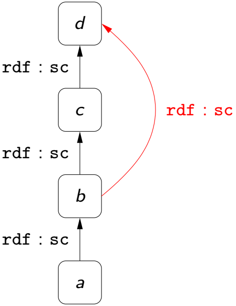
For RDF graphs with RDFS vocabulary, the core of G may contain redundant information:



Core and RDFS

For RDF graphs with RDFS vocabulary, the core of G may contain redundant information:





A normal form for RDF graphs

To reduce the size of the materialization, we can combine both core and closure.

A normal form for RDF graphs

To reduce the size of the materialization, we can combine both core and closure.

- $\mathrm{nf}(G) = \mathrm{core}(\mathrm{closure}(G))$

A normal form for RDF graphs

To reduce the size of the materialization, we can combine both core and closure.

- $\text{nf}(G) = \text{core}(\text{closure}(G))$

Theorem (GHM04)

- *G_1 is equivalent to G_2 iff $nf(G_1) \cong nf(G_2)$.*
- *$G_1 \models G_2$ iff $G_2 \to nf(G_1)$*

A normal form for RDF graphs

To reduce the size of the materialization, we can combine both core and closure.

- $\text{nf}(G) = \text{core}(\text{closure}(G))$

Theorem (GHM04)

- *G_1 is equivalent to G_2 iff $nf(G_1) \cong nf(G_2)$.*
- *$G_1 \models G_2$ iff $G_2 \rightarrow nf(G_1)$*

Complexity

The problem of deciding if $G_1 = nf(G_2)$ is DP-complete.

Querying RDF data: Desiderata

Let D be a database, Q a query, and $Q(D)$ the answer.

- Outputs should belong to the same family of objects as inputs
- If $D \equiv D'$, then $Q(D) = Q(D')$
  (Weaker) If $D \equiv D'$, then $Q(D) \cong Q(D')$
- $Q(D)$ should have no (or minimal) redundancies
- The framework should be extensible to RDFS
  (Should the framework be extensible to OWL?)
- Incorporate to the framework the notion of entailment

Querying RDF data: Desiderata

Outputs should belong to the same family of objects as inputs

- Allows compositionality of queries
- Allows defining views
- Allows rewriting

In RDF, the natural objects of input/output are RDF graphs.

Querying RDF data: Desiderata

If $D \equiv D'$, then $Q(D) = Q(D')$
(Weaker) If $D \equiv D'$, then $Q(D) \cong Q(D')$

- Outputs are syntactic or semantic objects?
- Need a notion of "equivalent" databases ($\equiv$)
  (In RDF, there is a standard notion of logical equivalence)
- One could just ask logical equivalence in the output
- In RDF there is an intermediate notion: graph isomorphism

Querying RDF data: Desiderata

$Q(D)$ should have no (or minimal) redundancies

- Desirable to avoid inconsistencies
- Desirable to improve processing time and space
- Standard requirement for exchange information

Querying RDF data: Desiderata

The framework should be extensible to RDFS
(Should the framework be extensible to OWL?)

- A basic requirement of the Semantic Web Architecture
- Extension to OWL are not trivial because of the known mismatch
- Not necessarily related to the type of semantics given (logical framework, graph matching, etc.)

Querying RDF data: Desiderata

Incorporate to the framework the notion of entailment

- RDF graphs are not purely syntactic objects
- Would like to incorporate KB framework
- Beware of the complexity issues! RDF navigates on the Web
- Find the good compromise

Querying RDF data: Definitions

A conjunctive query Q is a pair of RDF graphs H, B where some resources have been replaced by variables $\bar{X}, \bar{Y}$ in V.

$$Q: \quad H(\bar{X}) \leftarrow B(\bar{X}, \bar{Y})$$

Issues:

- Free variables in B (projection)
- Treatment of blank nodes in B
- Treatment of blank nodes in H

Querying RDF data: Definitions (cont.)

A valuation is a function $v : V \to U \cup B \cup L$

A matching of a graph B in the database D is a valuation v such that $v(B) \subseteq D$.

A pre-answer to Q over D is the set

$$preans(Q, D) = \{v(H) : \ v \text{ is a matching of } B \text{ in } D\ \}$$

A single answer is an element of $preans(Q, D)$

Querying RDF data: Two semantics

Union: answer $Q(D)$ is the union of all single answers

$$ans_U(Q, D) = \bigcup preans(Q, D)$$

Merge: answer $Q(D)$ is the merge of all single answers

$$ans_M(Q, D) = \biguplus preans(Q, D)$$

Proposition

1. *For both semantics, if $D \models D'$ then $ans(Q, D) \models ans(Q, D')$*
2. *For all D, $ans_U(Q, D) \models ans_M(Q, D)$*
3. *With merge semantics, we cannot represent the identity query*

Problem

Two non-isomorphic datasets D, D' give different answers to the same query.

A slightly refined semantics:

1. Normalize D before querying
2. Then query as usual over $nf(D)$

Good News: if $D \equiv D'$ then $Q(D) \cong Q(D')$
Bad News: computing $nf(D)$ is hard

The news as formal results:

Theorem (MPG07)

Do not need to compute the normal form.

Theorem (FG06)

If a query language has the following two properties:

1. *for all Q, if $D \equiv D'$ then $Q(D) = Q(D')$,*
2. *can represent the identity query,*

then the complexity of evaluation (in data complexity) is as hard as the evaluation of $\equiv$.

Querying RDF data: Containment

A query Q' contains a query Q, denoted $Q \sqsubseteq Q'$ iff $ans(Q', D)$ comprises all the information of $ans(Q, D)$.

In classical DB: $ans(Q, D) \subseteq ans(Q', D)$

In our setting we have two versions:

- $ans(Q, D) \subseteq ans(Q', D) \quad (Q \sqsubseteq_p Q')$
- $preans(Q, D) \subseteq preans(Q', D)$ (modulo iso) $(Q \sqsubseteq_m Q')$

For ground RDF both notions coincide.

Querying RDF data: Complexity

Query complexity version: The evaluation problem is NP-complete

Data complexity version: The evaluation problem is polynomial

Querying with SPARQL

- SPARQL is the W3C candidate recommendation query language for RDF.
- SPARQL is a graph-matching query language.
- A SPARQL query consists of three parts:
  - Pattern matching: optional, union, nesting, filtering.
  - Solution modifiers: projection, distinct, order, limit, offset.
  - Output part: construction of new triples, ...

Recall the formalization from Unit-2

Syntax:

- Triple patterns: RDF triple + variables (no bnodes)
- Operators between triple patterns: AND, UNION, OPT.
- Filtering of solutions: FILTER.
- A full parenthesized algebra.

Recall the formalization from Unit-2

Semantics:

- Based on mappings, partial functions from variables to terms.
- A mapping μ is a solution of triple pattern t in G iff
  - $\mu(t) \in G$
  - $\mathrm{dom}(\mu) = \mathrm{var}(t)$.
- $[\![t]\!]_G$ is the evaluation of t in G, the set of solutions.

Example

G	t	$[\![t]\!]_G$
$(R_1$, name, john$)$ $(R_1$, email, J@ed.ex$)$ $(R_2$, name, paul$)$	$(?X$, name, $?Y)$	see below

	$?X$	$?Y$
μ_1:	R_1	john
μ_2:	R_2	paul

Compatible mappings

Definition

Two mappings are compatible if they agree in their shared variables.

Example

	$?X$	$?Y$	$?Z$	$?V$
μ_1 :	R_1	john		
μ_2 :	R_1		J@edu.ex	
μ_3 :			P@edu.ex	R_2

Compatible mappings

Definition

Two mappings are compatible if they agree in their shared variables.

Example

	$?X$	$?Y$	$?Z$	$?V$
μ_1 :	R_1	john		
μ_2 :	R_1		J@edu.ex	
μ_3 :			P@edu.ex	R_2

Compatible mappings

Definition

Two mappings are compatible if they agree in their shared variables.

Example

	?X	?Y	?Z	?V
μ_1 :	R_1	john		
μ_2 :	R_1		J@edu.ex	
μ_3 :			P@edu.ex	R_2
$\mu_1 \cup \mu_2$:	R_1	john	J@edu.ex	

Compatible mappings

Definition

Two mappings are compatible if they agree in their shared variables.

Example

	$?X$	$?Y$	$?Z$	$?V$
μ_1 :	R_1	john		
μ_2 :	R_1		J@edu.ex	
μ_3 :			P@edu.ex	R_2
$\mu_1 \cup \mu_2$:	R_1	john	J@edu.ex	

Compatible mappings

Definition

Two mappings are compatible if they agree in their shared variables.

Example

	$?X$	$?Y$	$?Z$	$?V$
μ_1 :	R_1	john		
μ_2 :	R_1		J@edu.ex	
μ_3 :			P@edu.ex	R_2
$\mu_1 \cup \mu_2$:	R_1	john	J@edu.ex	
$\mu_1 \cup \mu_3$:	R_1	john	P@edu.ex	R_2

Compatible mappings

Definition

Two mappings are compatible if they agree in their shared variables.

Example

	$?X$	$?Y$	$?Z$	$?V$
μ_1 :	R_1	john		
μ_2 :	R_1		J@edu.ex	
μ_3 :			P@edu.ex	R_2
$\mu_1 \cup \mu_2$:	R_1	john	J@edu.ex	
$\mu_1 \cup \mu_3$:	R_1	john	P@edu.ex	R_2

- μ_2 and μ_3 are not compatible

Sets of mappings and operations

Let M_1 and M_2 be sets of mappings:

Definition

Let M_1 and M_2 be sets of mappings:

Definition

Join: $M_1 \bowtie M_2$

- extending mappings in M_1 with compatible mappings in M_2

Sets of mappings and operations

Let M_1 and M_2 be sets of mappings:

Definition

Join: $M_1 \bowtie M_2$

- extending mappings in M_1 with compatible mappings in M_2

Difference: $M_1 \setminus M_2$

- mappings in M_1 that cannot be extended with mappings in M_2

Sets of mappings and operations

Let M_1 and M_2 be sets of mappings:

Definition

Join: $M_1 \bowtie M_2$

- extending mappings in M_1 with compatible mappings in M_2

Difference: $M_1 \smallsetminus M_2$

- mappings in M_1 that cannot be extended with mappings in M_2

Union: $M_1 \cup M_2$

- mappings in M_1 plus mappings in M_2 (set theoretical union)

Sets of mappings and operations

Let M_1 and M_2 be sets of mappings:

Definition

Join: $M_1 \bowtie M_2$

- extending mappings in M_1 with compatible mappings in M_2

Difference: $M_1 \smallsetminus M_2$

- mappings in M_1 that cannot be extended with mappings in M_2

Union: $M_1 \cup M_2$

- mappings in M_1 plus mappings in M_2 (set theoretical union)

Definition

Left Outer Join: $M_1 ⟕ M_2 = (M_1 \bowtie M_2) \cup (M_1 \smallsetminus M_2)$

Semantics of general graph patterns

Definition

Given a graph G the evaluation of a pattern is recursively defined

- $[\![(P_1 \text{ AND } P_2)]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$
- $[\![(P_1 \text{ UNION } P_2)]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$
- $[\![(P_1 \text{ OPT } P_2)]\!]_G = [\![P_1]\!]_G ⟕ [\![P_2]\!]_G$
- $[\![(P \text{ FILTER } R)]\!]_G = \{\mu \in [\![P]\!]_G \mid \mu \text{ satisfies } R\}$

Differences with Relational Algebra / SQL

- Not a fixed output schema
  - mappings instead of tables
  - schema is implicit in the domain of mappings
- Too many NULLs
  - mappings with disjoint domains can be joined
  - mappings with distinct domains in output solutions
- SPARQL-to-SQL translations experience these issues
  - need of `IS NULL`/`IS NOT NULL` in join/outerjoin conditions
  - need of `COALESCE` in constructing output schema

SPARQL complexity: the evaluation problem

Input:

A mapping μ, a graph pattern P, and an RDF graph G.

Question:

Is the mapping in the evaluation of the pattern against the graph?

$$\mu \in [\![P]\!]_G ?$$

Evaluation of AND-FILTER patterns is polynomial.

Theorem (PAG06)

For patterns using only AND and FILTER operators, the evaluation problem is polynomial:

$$O(|P| \times |G|).$$

Evaluation of AND-FILTER patterns is polynomial.

Theorem (PAG06)

For patterns using only AND *and* FILTER *operators, the evaluation problem is polynomial:*

$$O(|P| \times |G|).$$

Proof idea

- *Check that the mapping makes every triple to match.*
- *Then check that the mapping satisfies the* FILTERs.

Evaluation including UNION is NP-complete.

Theorem (PAG06)

For patterns using AND, FILTER *and* UNION *operators, the evaluation problem is NP-complete.*

Evaluation including UNION is NP-complete.

Theorem (PAG06)

For patterns using AND, FILTER and UNION operators, the evaluation problem is NP-complete.

Proof idea

- *Reduction from 3SAT.*
- *A pattern encodes the propositional formula.*
- $\neg$ bound *is used to encode negation.*

Evaluation including UNION is NP-complete.

Theorem (PAG06)

For patterns using AND, FILTER *and* UNION *operators, the evaluation problem is NP-complete.*

Proof idea

- *Reduction from 3SAT.*
- *A pattern encodes the propositional formula.*
- $\neg$ bound *is used to encode negation.*

Evaluation including OPT is PSPACE-complete.

Theorem (PAG06)

For patterns using AND, FILTER and OPT operators, the evaluation problem is PSPACE-complete.

Evaluation including OPT is PSPACE-complete.

Theorem (PAG06)

For patterns using AND, FILTER and OPT operators, the evaluation problem is PSPACE-complete.

Proof idea

- *Reduction from QBF*
- *A pattern encodes a quantified propositional formula:*

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \cdots \psi.$$

- *nested OPTs are used to encode quantifier alternation.*
  (This time, we do not need $\neg$ *bound.)*

Evaluation including OPT is PSPACE-complete.

Theorem (PAG06)

For patterns using AND, FILTER *and* OPT *operators, the evaluation problem is PSPACE-complete.*

Proof idea

- *Reduction from QBF*
- *A pattern encodes a quantified propositional formula:*

$$\forall x_1 \exists y_1 \forall x_2 \exists y_2 \cdots \psi.$$

- *nested* OPT*s are used to encode quantifier alternation.*

  (This time, we do not need $\neg$ bound.*)*

PSPACE-hardness: A closer look

Assume $\varphi = \forall x_1 \exists y_1\, \psi$, where $\psi = (x_1 \vee \neg y_1) \wedge (\neg x_1 \vee y_1)$.

We generate G, P_φ and μ_0 such that μ_0 belongs to the answer of P_φ over G iff φ is valid:

G :

P_ψ :

P_φ :

μ_0 :

PSPACE-hardness: A closer look

Assume $\varphi = \forall x_1 \exists y_1\, \psi$, where $\psi = (x_1 \vee \neg y_1) \wedge (\neg x_1 \vee y_1)$.

We generate G, P_φ and μ_0 such that μ_0 belongs to the answer of P_φ over G iff φ is valid:

$G \quad : \quad \{(a, \mathtt{tv}, 0),\ (a, \mathtt{tv}, 1),\ (a, \mathtt{false}, 0),\ (a, \mathtt{true}, 1)\}$

$P_\psi \quad :$

$P_\varphi \quad :$

$\mu_0 \quad :$

PSPACE-hardness: A closer look

Assume $\varphi = \forall x_1 \exists y_1\, \psi$, where $\psi = (x_1 \vee \neg y_1) \wedge (\neg x_1 \vee y_1)$.

We generate G, P_φ and μ_0 such that μ_0 belongs to the answer of P_φ over G iff φ is valid:

G : $\{(a, \texttt{tv}, 0),\ (a, \texttt{tv}, 1),\ (a, \texttt{false}, 0),\ (a, \texttt{true}, 1)\}$

P_ψ : $((a, \texttt{tv}, ?X_1)\ \text{AND}\ (a, \texttt{tv}, ?Y_1))\ \text{FILTER}$
$((?X_1 = 1 \vee ?Y_1 = 0) \wedge (?X_1 = 0 \vee ?Y_1 = 1))$

P_φ :

μ_0 :

PSPACE-hardness: A closer look

Assume $\varphi = \forall x_1 \exists y_1\, \psi$, where $\psi = (x_1 \vee \neg y_1) \wedge (\neg x_1 \vee y_1)$.

We generate G, P_φ and μ_0 such that μ_0 belongs to the answer of P_φ over G iff φ is valid:

$$
\begin{array}{lcl}
G & : & \{(a, \mathtt{tv}, 0),\ (a, \mathtt{tv}, 1),\ (a, \mathtt{false}, 0),\ (a, \mathtt{true}, 1)\} \\
 & & \\
P_\psi & : & ((a, \mathtt{tv}, ?X_1)\ \text{AND}\ (a, \mathtt{tv}, ?Y_1))\ \text{FILTER} \\
 & & ((?X_1 = 1 \vee ?Y_1 = 0) \wedge (?X_1 = 0 \vee ?Y_1 = 1)) \\
P_\varphi & : & (a, \mathtt{true}, ?B_0)\ \text{OPT}\ (P_1\ \text{OPT}\ (Q_1\ \text{AND}\ P_\psi)) \\
 & & \\
\mu_0 & : &
\end{array}
$$

PSPACE-hardness: A closer look

Assume $\varphi = \forall x_1 \exists y_1\, \psi$, where $\psi = (x_1 \vee \neg y_1) \wedge (\neg x_1 \vee y_1)$.

We generate G, P_φ and μ_0 such that μ_0 belongs to the answer of P_φ over G iff φ is valid:

$$
\begin{array}{lcl}
G & : & \{(a, \mathtt{tv}, 0),\ (a, \mathtt{tv}, 1),\ (a, \mathtt{false}, 0),\ (a, \mathtt{true}, 1)\} \\[1em]
P_\psi & : & ((a, \mathtt{tv}, ?X_1)\ \text{AND}\ (a, \mathtt{tv}, ?Y_1))\ \text{FILTER} \\
 & & ((?X_1 = 1 \vee ?Y_1 = 0) \wedge (?X_1 = 0 \vee ?Y_1 = 1)) \\
P_\varphi & : & (a, \mathtt{true}, ?B_0)\ \text{OPT}\ (P_1\ \text{OPT}\ (Q_1\ \text{AND}\ P_\psi)) \\[1em]
\mu_0 & : & \{?B_0 \mapsto 1\}
\end{array}
$$

PSPACE-hardness: A closer look

$$
\begin{array}{lcl}
P_\varphi & : & (a, \texttt{true}, ?B_0) \text{ OPT } (P_1 \text{ OPT } (Q_1 \text{ AND } P_\psi)) \\
P_1 & : & (a, \texttt{tv}, ?X_1) \\
Q_1 & : & (a, \texttt{tv}, ?X_1) \text{ AND } (a, \texttt{tv}, ?Y_1) \text{ AND } (a, \texttt{false}, ?B_0)
\end{array}
$$

PSPACE-hardness: A closer look

$$
\begin{array}{lcl}
P_\varphi & : & (a, \mathtt{true}, ?B_0) \ \text{OPT} \ (P_1 \ \text{OPT} \ (Q_1 \ \text{AND} \ P_\psi)) \\
P_1 & : & (a, \mathtt{tv}, ?X_1) \\
Q_1 & : & (a, \mathtt{tv}, ?X_1) \ \text{AND} \ (a, \mathtt{tv}, ?Y_1) \ \text{AND} \ (a, \mathtt{false}, ?B_0)
\end{array}
$$

$?B_0 \mapsto 1$

PSPACE-hardness: A closer look

$$
\begin{array}{lcl}
P_\varphi & : & (a, \texttt{true}, ?B_0) \text{ OPT } (P_1 \text{ OPT } (Q_1 \text{ AND } P_\psi)) \\
P_1 & : & (a, \texttt{tv}, ?X_1) \\
Q_1 & : & (a, \texttt{tv}, ?X_1) \text{ AND } (a, \texttt{tv}, ?Y_1) \text{ AND } (a, \texttt{false}, ?B_0)
\end{array}
$$

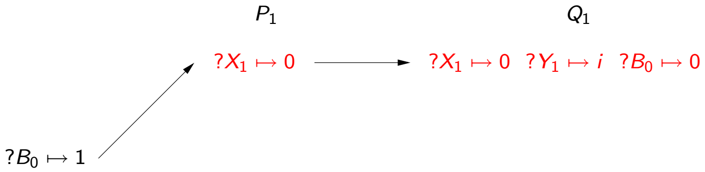
P_1

$?X_1 \mapsto 0$

$?B_0 \mapsto 1$

PSPACE-hardness: A closer look

$$
\begin{array}{lcl}
P_\varphi & : & (a, \texttt{true}, ?B_0) \ \text{OPT} \ (P_1 \ \text{OPT} \ (Q_1 \ \text{AND} \ P_\psi)) \\
P_1 & : & (a, \texttt{tv}, ?X_1) \\
Q_1 & : & (a, \texttt{tv}, ?X_1) \ \text{AND} \ (a, \texttt{tv}, ?Y_1) \ \text{AND} \ (a, \texttt{false}, ?B_0)
\end{array}
$$



PSPACE-hardness: A closer look

$$
\begin{array}{lcl}
P_\varphi & : & (a, \texttt{true}, ?B_0) \text{ OPT } (P_1 \text{ OPT } (Q_1 \text{ AND } P_\psi)) \\
P_1 & : & (a, \texttt{tv}, ?X_1) \\
Q_1 & : & (a, \texttt{tv}, ?X_1) \text{ AND } (a, \texttt{tv}, ?Y_1) \text{ AND } (a, \texttt{false}, ?B_0)
\end{array}
$$

$$
\begin{array}{ccccc}
 & & P_1 & & Q_1 \\
 & \nearrow & ?X_1 \mapsto 0 & \longrightarrow & ?X_1 \mapsto 0 \;\; ?Y_1 \mapsto i \;\; ?B_0 \mapsto 0 \\
?B_0 \mapsto 1 & & & & \\
 & \searrow & \color{red}{?X_1 \mapsto 1} & &
\end{array}
$$

PSPACE-hardness: A closer look

$$
\begin{array}{lcl}
P_\varphi & : & (a, \texttt{true}, ?B_0) \ \text{OPT} \ (P_1 \ \text{OPT} \ (Q_1 \ \text{AND} \ P_\psi)) \\
P_1 & : & (a, \texttt{tv}, ?X_1) \\
Q_1 & : & (a, \texttt{tv}, ?X_1) \ \text{AND} \ (a, \texttt{tv}, ?Y_1) \ \text{AND} \ (a, \texttt{false}, ?B_0)
\end{array}
$$

$$
\begin{tikzcd}
& P_1 & Q_1 \\
& ?X_1 \mapsto 0 \arrow[r] & ?X_1 \mapsto 0 \quad ?Y_1 \mapsto i \quad ?B_0 \mapsto 0 \\
?B_0 \mapsto 1 \arrow[ur] \arrow[dr] & & \\
& ?X_1 \mapsto 1 \arrow[r] & ?X_1 \mapsto 1 \quad ?Y_1 \mapsto j \quad ?B_0 \mapsto 0
\end{tikzcd}
$$

Data–complexity is polynomial

Theorem (PAG06)

When patterns are consider to be fixed (data complexity), the evaluation problem is in LOGSPACE.

Data–complexity is polynomial

Theorem (PAG06)

When patterns are consider to be fixed (data complexity), the evaluation problem is in LOGSPACE.

Proof idea

From data–complexity of first–order logic.

SPARQL reordering/optimization: a simple normal from

- AND and UNION are commutative and associative.
- AND, OPT, and FILTER distribute over UNION.

Theorem (UNION Normal Form)

Every graph pattern is equivalent to one of the form

$$P_1 \ \text{UNION} \ P_2 \ \text{UNION} \ \cdots \ \text{UNION} \ P_n$$

where each P_i is UNION*–free.*

We concentrate in UNION-free patterns.

Well–designed patterns

Definition

A graph pattern is well–designed iff for every OPT in the pattern

$$(\cdots\cdots\cdots\cdots \quad (\quad A \quad \mathrm{OPT} \quad B \quad) \quad \cdots\cdots\cdots\cdots)$$

if a variable occurs inside B and anywhere outside the OPT, then the variable must also occur inside A.

Well–designed patterns

Definition

A graph pattern is well–designed iff for every OPT in the pattern

$$(\cdots\cdots\cdots\cdots \quad (\quad A \quad \mathrm{OPT} \quad \underset{\uparrow}{B} \quad) \quad \cdots\cdots\cdots\cdots)$$

if a variable occurs inside B and anywhere outside the OPT, then the variable must also occur inside A.

Well–designed patterns

Definition

A graph pattern is well–designed iff for every OPT in the pattern

$$(\cdots\cdots\cdots\cdots \quad (\quad A \quad \text{OPT} \quad B \quad) \quad \cdots\cdots\cdots\cdots)$$
$$\uparrow \qquad\qquad\qquad\qquad \uparrow \qquad\qquad\qquad \uparrow$$

if a variable occurs inside B and anywhere outside the OPT, then the variable must also occur inside A.

Well–designed patterns

Definition

A graph pattern is well–designed iff for every OPT in the pattern

$$(\cdots\cdots\cdots\cdots \quad (\quad A \quad \mathrm{OPT} \quad B \quad) \quad \cdots\cdots\cdots\cdots)$$
$$\uparrow \qquad \uparrow \qquad \uparrow \qquad \uparrow$$

if a variable occurs inside B and anywhere outside the OPT, then the variable must also occur inside A.

Well–designed patterns

Definition

A graph pattern is well–designed iff for every OPT in the pattern

$$(\cdots\cdots\cdots\cdots \quad (\quad A \quad \text{OPT} \quad B \quad) \quad \cdots\cdots\cdots\cdots)$$
$$\uparrow \qquad\qquad \uparrow \qquad\qquad \uparrow \qquad\qquad \uparrow$$

if a variable occurs inside B and anywhere outside the OPT, then the variable must also occur inside A.

Example

$(\; (\; (?Y, \text{name}, \text{paul}) \text{ OPT } (?X, \text{email}, ?Z) \;) \quad \text{AND} \quad (?X, \text{name}, \text{john}) \;)$

Well–designed patterns

Definition

A graph pattern is well–designed iff for every OPT in the pattern

$$(\cdots\cdots\cdots\cdots \quad (\quad A \quad \text{OPT} \quad B \quad) \quad \cdots\cdots\cdots\cdots)$$
$$\qquad\uparrow \qquad\qquad\quad \uparrow \qquad\qquad\quad \uparrow \qquad\qquad\quad \uparrow$$

if a variable occurs inside B and anywhere outside the OPT, then the variable must also occur inside A.

Example

$$(\; (\; (?Y, \text{name}, \text{paul}) \ \text{OPT} \ (\underset{\uparrow}{?X}, \text{email}, ?Z) \;) \quad \text{AND} \quad (?X, \text{name}, \text{john}) \;)$$

Well–designed patterns

Definition

A graph pattern is well–designed iff for every OPT in the pattern

$$(\cdots\cdots\cdots\cdots \quad (\quad \underset{\uparrow}{A} \quad \text{OPT} \quad \underset{\uparrow}{B} \quad) \quad \cdots\cdots\cdots\cdots)$$

if a variable occurs inside B and anywhere outside the OPT, then the variable must also occur inside A.

Example

$$\big(\, (\, (?Y, \text{name}, \text{paul}) \text{ OPT } (\underset{\uparrow}{?X}, \text{email}, ?Z) \,) \quad \text{AND} \quad (\underset{\uparrow}{?X}, \text{name}, \text{john}) \,\big)$$

Well–designed patterns

Definition

A graph pattern is well–designed iff for every OPT in the pattern

$$(\cdots\cdots\cdots\cdots \quad (\quad A \quad \text{OPT} \quad B \quad) \quad \cdots\cdots\cdots\cdots)$$

$$\qquad \uparrow \qquad\qquad\qquad \uparrow \qquad\qquad\qquad \uparrow \qquad\qquad\qquad \uparrow$$

if a variable occurs inside B and anywhere outside the OPT, then the variable must also occur inside A.

Example

$$(\ (\ (?Y,\ \text{name},\ \text{paul})\ \text{OPT}\ (?X,\ \text{email},\ ?Z)\ )\quad \text{AND}\quad (?X,\ \text{name},\ \text{john})\ )$$

Well–designed patterns and PSPACE-hardness

In the PSPACE-hardness reduction we use this formula:

$$\begin{array}{lcl} P_\varphi & : & (a, \texttt{true}, ?B_0) \ \text{OPT} \ (P_1 \ \text{OPT} \ (Q_1 \ \text{AND} \ P_\psi)) \\ P_1 & : & (a, \texttt{tv}, ?X_1) \\ Q_1 & : & (a, \texttt{tv}, ?X_1) \ \text{AND} \ (a, \texttt{tv}, ?Y_1) \ \text{AND} \ (a, \texttt{false}, ?B_0) \end{array}$$

Well–designed patterns and PSPACE-hardness

In the PSPACE-hardness reduction we use this formula:

$$
\begin{array}{lcl}
P_\varphi & : & (a, \texttt{true}, ?B_0) \ \text{OPT} \ (P_1 \ \ \text{OPT} \ (Q_1 \ \ \text{AND} \ \ P_\psi)) \\
P_1 & : & (a, \texttt{tv}, ?X_1) \\
Q_1 & : & (a, \texttt{tv}, ?X_1) \ \text{AND} \ (a, \texttt{tv}, ?Y_1) \ \text{AND} \ (a, \texttt{false}, ?B_0)
\end{array}
$$

It is not well-designed: B_0

Well–designed patterns: reordering/optimization

For well-designed patterns

- P_1 AND $(P_2$ OPT $P_3) \equiv (P_1$ AND $P_2)$ OPT P_3
- $(P_1$ OPT $P_2)$ OPT $P_3 \equiv (P_1$ OPT $P_3)$ OPT P_2

Theorem (OPT Normal Form)

Every well–designed pattern is equivalent to one of the form

$$(\cdots(\,t_1 \text{ AND } \cdots \text{ AND } t_k\,) \text{ OPT } O_1\,)\cdots) \text{ OPT } O_n\,)$$

where each t_i is a triple pattern, and each O_j is a pattern of the same form.

Well–designed patterns: reordering/optimization

For well-designed patterns

- P_1 AND $(P_2$ OPT $P_3) \equiv (P_1$ AND $P_2)$ OPT P_3
- $(P_1$ OPT $P_2)$ OPT $P_3 \equiv (P_1$ OPT $P_3)$ OPT P_2

Theorem (OPT Normal Form)

Every well–designed pattern is equivalent to one of the form

$$(\cdots(\,t_1 \text{ AND } \cdots \text{ AND } t_k\,) \text{ OPT } O_1\,)\cdots) \text{ OPT } O_n\,)$$

where each t_i is a triple pattern, and each O_j is a pattern of the same form.

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS:

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics,

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics, entailment of RDFS graphs,

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics, entailment of RDFS graphs, normal forms for RDFS graphs (closure and core).

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.

- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics, entailment of RDFS graphs, normal forms for RDFS graphs (closure and core).
  - SPARQL:

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics, entailment of RDFS graphs, normal forms for RDFS graphs (closure and core).
  - SPARQL: Formal semantics,

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics, entailment of RDFS graphs, normal forms for RDFS graphs (closure and core).
  - SPARQL: Formal semantics, complexity of query evaluation,

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics, entailment of RDFS graphs, normal forms for RDFS graphs (closure and core).
  - SPARQL: Formal semantics, complexity of query evaluation, query optimization.

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics, entailment of RDFS graphs, normal forms for RDFS graphs (closure and core).
  - SPARQL: Formal semantics, complexity of query evaluation, query optimization.
  - Updating

Final remarks

- RDFS can be considered a new data model.
  - It is the W3C's recommendation for describing Web metadata.
- RDFS can definitely benefit from database technology.
  - RDFS: Formal semantics, entailment of RDFS graphs, normal forms for RDFS graphs (closure and core).
  - SPARQL: Formal semantics, complexity of query evaluation, query optimization.
  - Updating
  - . . .

References

- A. Chandra, P. Merlin, *Optimal Implementation of Conjunctive Queries in Relational Databases*. In *STOC* 1977.
- R. Fagin, P. Kolaitis, L. Popa, *Data Exchange: Getting to the Core*. In *PODS* 2003.
- C. Gutierrez, C. Hurtado, A. O. Mendelzon, *Foundations of Semantic Web Databases*. In *PODS* 2004.
- P. Hayes, *RDF Semantics*. W3C Recommendation 2004.
- P. Hell, J. Nesetril, *The Core of a Graph*. *Discrete Mathematics* 1992.
- S. Muñoz, J. Pérez, C. Gutierrez, *Minimal Deductive Systems for RDF*. In *ESWC* 2007.
- J. Pérez, M. Arenas, C. Gutierrez, *Semantics and Complexity of SPARQL*. In *ISWC* 2006.