



Abstract Access Control Models for Dynamic RDF Datasets

Irini Fundulaki
CWI & FORTH-ICS
Giorgos Flouris
FORTH-ICS
Vassilis Papakonstantinou
FORTH-ICS & University of Crete

Controlling Access to RDF Data

Why RDF Data?

- RDF is the de-facto standard for publishing data in the Linked Open Data Cloud
 - Public Government Data (US, UK, France, Austria, The Netherlands, ...)
 - E-Science (astronomy, life sciences, earth sciences)
 - Social Networks
 - DBPedia, Wikipedia, CIA World FactBook, ...
- Why Access Control?
 - Crucial for *sensitive* content since it ensures the *selective exposure* of information to different classes of users

Controlling Access to RDF Data

- Fine-grained Access Control Model for RDF
 - focus at the RDF triple level
 - focus on *read-only permissions*
 - with support for RDFS inference to infer new knowledge
 - encodes how an access label has been computed
 - contributing triples
- Implementation of a *fine-grained,access control framework* on top of *the* MonetDB column store engine

Access Control Annotations

 Standard access control models associate a concrete access label to a triple

S	p	o	permission
& a	type	Student	allowed
Student	SC	Person	denied

 An implied RDF triple can be accessed if and only if all its implying triples can be accessed

s	\boldsymbol{p}	0	permission		
& a	type	Person	denied		

Access Control Annotations

• In the case of *any kind of update*, the implied triples & their labels must be re-computed

S	p	O	permission
 & a	type	Student	allowed
Student	SC	Person	allowed \Leftarrow

• An implied RDF triple can be accessed if and only if all its implying triples can be accessed

	S	p	0	permission	
	& a	type	Person	allowed	

the overhead can be substantial when updates occur frequently

Access Control Annotations

- Annotation models are easy to handle but are not amenable to changes since there is no knowledge of the affected triples
- Any change leads to the re-computation of inferred triples and their labels
 - if the access label of one triple changes
 - if a triple is deleted, modified or added
 - if the semantics according to which the labels of inferred triples are computed change
 - if the policy changes (a liberal policy becomes conservative)

Abstract Access Control Models for RDF

- Encode how the label of an implied triple was computed
- Triples are assigned abstract tokens and not concrete values

S	p	o	permission
& a	type	Student	L
Student	SC	Person	$oxed{\mathcal{L}_{2}}$

S	p	<i>o</i>	permission
& a	type	Person	$l_1 \odot l_2$

- $l_1 l_2$: abstract tokens
- ○ : operator that encodes that inference was used to produce the inferred triple

Annotation: Computing the Access Labels

- Triples are assigned labels through *authorization queries*
- RDFS *inference rules* are applied to infer new knowledge

1 . (construct 12x first Name 211)

A_1 : (construct {:x firstName :y} where {?x type Student }, 11)			1		
——————————————————————————————————————		S	p	0	1
A_2 : (construct {? x sc ? y }, 12)	<i>q</i> ₁ :	Student	SC	o Person Agent Student Alice class Person	12
A_3 : (construct {?x type Student }, 13)	q_2 :	Person	SC	Agent	12
Authorizations	<i>q</i> ₃ :	&a	type	Student	13
(Query, abstract token)	q_4 :	& a	firstName	Alice	11
	<i>q</i> ₅ :	Agent	type	class	14
	<i>q</i> ₆ :	Student	SC	Person	15

Annotation: Applying RDFS Inference Rules

RDFS Inference: quadruple generating rules

$$(A_1, sc, A_2, l_1)$$
 (A_2, sc, A_3, l_2) \longrightarrow $(A_1, sc, A_3, l_1 \odot l_2)$ $(\&r_1, type, A_1, l_1)$ (A_1, sc, A_2, l_2) \longrightarrow $(\&r_1, type, A_2, l_1 \odot l_2)$

	S	p	0	1	
<i>q</i> ₁ :	Student	sc	Person	12	
<i>q</i> ₂ :	Person	sc	Agent	12	
<i>q</i> ₃ :	&a	type	Student	13	
<i>q</i> ₆ :	Student	SC	Person	<i>1</i> 5	

RDF quadruples

	s	p	o	1
q_8 :	Student	SC	Agent	12 🔾 12
q_9 :	Student	SC	Agent	<i>15</i> ⊙ <i>12</i>
<i>q</i> ₁₀ :	&a	type	Person	<i>13</i> ⊙ <i>12</i>
<i>q</i> ₁₁ :	&a	type	Agent	(13 ⊙ 12) ⊙ 12
<i>q</i> ₁₂ :	&a	type	Agent	(15 ⊙ 12) ⊙ 12

Inferred RDF quadruples

Evaluation: Assign Concrete Values to Abstract Expressions

- Set of Concrete Tokens and a Mapping from abstract to concrete tokens
- Set of Concrete operators that implement the abstract ones
- Conflict resolution operator to resolve ambiguous labels
- *Access Function* to decide when a triple is accessible

Abstract Access Control Models for RDF

• Use of *concrete policies* to assign concrete values to the *abstract tokens* and *operators*

	S	p	o	pern	nision
 <i>q</i> ₁₁ :	& a	type	Student	13	true
<i>q</i> ₁₂ :	Student	SC	Person	12	false

- 13 maps to true and 12 maps to false
- ① maps to *logical conjunction*

S	p	0	permission				
& a	type	Person	false	true	and	false	l
	I				Eu	ıropean Data Foru	ım 2012

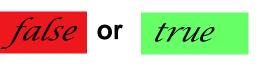
Abstract Access Control Models for RDF: Updates

• If a *concrete policy* changes, we need to re-compute the expressions

	S	p	o	perm	ision	
<i>q</i> ₁₁ :	& a	type	Student	13	false	
<i>q</i> ₁₂ :	Student	SC	Person	12	true	

- 13 maps to false and 12 maps to true
- ⊙ maps to *logical disjunction*

$oldsymbol{S}$	p	0	permission
& a	type	Person	true



Pros & Cons of Abstract Access Control Models

• Pros:

- The same application can experiment with different concrete policies over the same dataset
 - liberal vs conservative policies for different classes of users
- Different applications can experiment with different concrete policies for the same data
- In the case of updates there is no need re-compute the inferred triples

• Cons:

- overhead in the required storage space
 - algebraic expressions can become complex depending on the structure of the dataset





Questions?