# 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* | *p* | *o* | *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*** |

⇐

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

| | *s* | *p* | *o* | *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* | $\zeta_1$ |
| *Student* | *sc* | *Person* | $\zeta_2$ |

| *s* | *p* | *o* | *permission* |
|---|---|---|---|
| *&a* | *type* | *Person* | $\zeta_1 \odot \zeta_2$ |

- $\zeta_1$ $\zeta_2$: abstract tokens
- $\odot$ : 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

$A_1$ : (*construct* {?x *firstName* ?y}
*where* {?x type *Student* }, *l1*)

$A_2$ : (*construct* {?x sc ?y}, *l2*)

$A_3$ : (*construct* {?x type *Student* }, *l3*)

*Authorizations*
*(Query, abstract token)*

| | ***s*** | ***p*** | ***o*** | *l* |
|---|---|---|---|---|
| $q_1$: | *Student* | sc | *Person* | *l2* |
| $q_2$: | *Person* | sc | *Agent* | *l2* |
| $q_3$: | &a | type | *Student* | *l3* |
| $q_4$: | &a | *firstName* | *Alice* | *l1* |
| $q_5$: | *Agent* | type | class | *l4* |
| $q_6$: | *Student* | sc | *Person* | *l5* |

*RDF quadruples*

# Annotation: Applying RDFS Inference Rules

## RDFS Inference: quadruple generating rules

$(A_1, sc, A_2, l_1)$ $(A_2, sc, A_3, l_2)$ ⟹ $(A_1, sc, A_3, l_1 \odot l_2)$

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

| | *s* | *p* | *o* | *l* |
|---|---|---|---|---|
| $q_1$: | *Student* | sc | *Person* | *l2* |
| $q_2$: | *Person* | sc | *Agent* | *l2* |
| $q_3$: | *&a* | type | *Student* | *l3* |
| $q_6$: | *Student* | sc | *Person* | *l5* |

*RDF quadruples*

| | ***s*** | ***p*** | ***o*** | ***l*** |
|---|---|---|---|---|
| $q_8$: | *Student* | sc | *Agent* | *l2* ⊙ *l2* |
| $q_9$: | *Student* | sc | *Agent* | *l5* ⊙ *l2* |
| $q_{10}$: | *&a* | type | *Person* | *l3* ⊙ *l2* |
| $q_{11}$: | *&a* | type | *Agent* | (*l3* ⊙ *l2*) ⊙ *l2* |
| $q_{12}$: | *&a* | type | *Agent* | (*l5* ⊙ *l2*) ⊙ *l2* |

*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* | *permision* | |
|---|---|---|---|---|---|
| $q_{11}$: | *&a* | *type* | *Student* | *l3* | *true* |
| $q_{12}$: | *Student* | *sc* | *Person* | *l2* | *false* |

- *l3* maps to *true* and *l2* maps to *false*
- ⊙ maps to ***logical conjunction***

| *s* | *p* | *o* | *permission* |
|---|---|---|---|
| *&a* | *type* | *Person* | *false* |

*true* **and** *false*

# Abstract Access Control Models for RDF: Updates

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

| | *s* | *p* | *o* | *permision* | |
|---|---|---|---|---|---|
| $q_{11}$: | &a | type | Student | l3 | *false* |
| $q_{12}$: | Student | sc | Person | l2 | *true* |

- *l3* maps to *false* and *l2* maps to *true*
- ⊙ maps to ***logical disjunction***

| *s* | *p* | *o* | *permission* |
|---|---|---|---|
| &a | type | Person | *true* |

*false* **or** *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?*