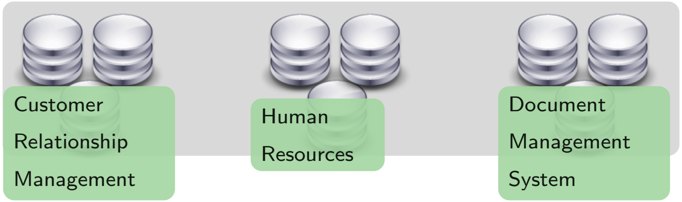
Protecting your RDF data

Nuno Lopes, Sabrina Kirrane, Axel Polleres, Alessandra Mileo

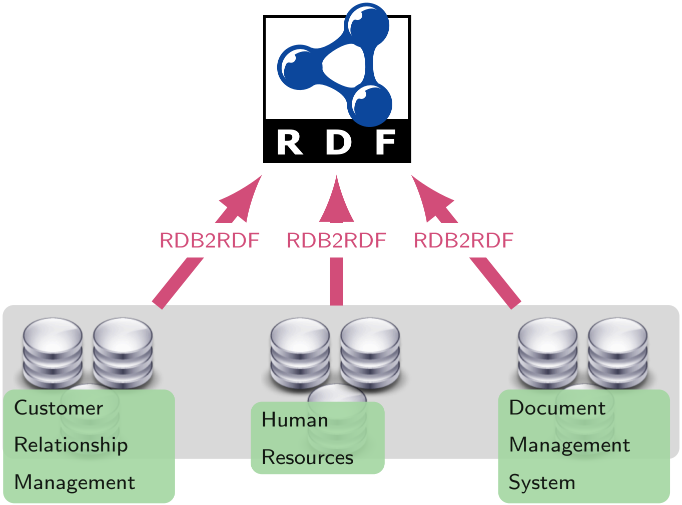
Data Integration in an Enterprise



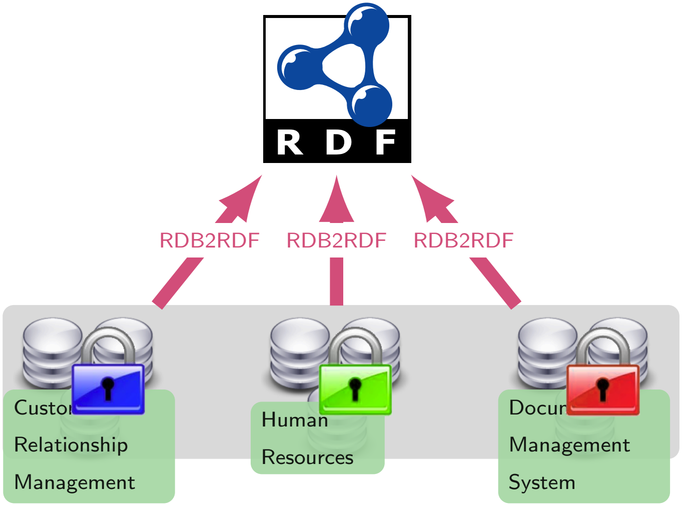
Data Integration in an Enterprise

Customer Relationship Management
Human Resources
Document Management System

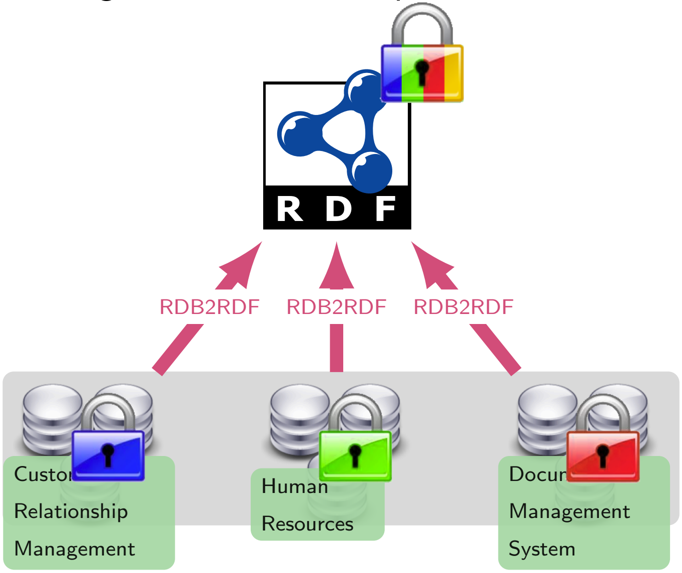
Data Integration in an Enterprise



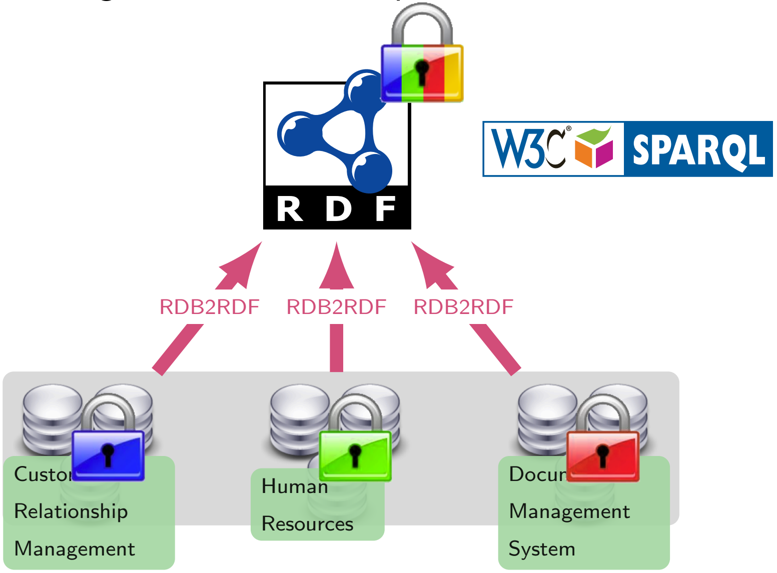
Data Integration in an Enterprise



Data Integration in an Enterprise



Data Integration in an Enterprise



Data Integration in an Enterprise

Data Integration in an Enterprise

Authorisation

Data Integration in an Enterprise

Authentication

Authorisation

Existing Annotated RDF(S) domains

Annotations refer to a specific **domain**

Existing Annotated RDF(S) domains

Annotations refer to a specific **domain**

Temporal:

```
:joeBloggs :worksFor :westportCars :[2010,2012]
```

Existing Annotated RDF(S) domains

Annotations refer to a specific **domain**

Temporal:

```
:joeBloggs :worksFor :westportCars :[2010,2012]
```

Fuzzy:

```
:joeBloggs :worksFor :westportCars :0.5
```

Existing Annotated RDF(S) domains

Annotations refer to a specific **domain**

Temporal:

```
:joeBloggs :worksFor :westportCars :[2010,2012]
```

Fuzzy:

```
:joeBloggs :worksFor :westportCars :0.5
```

Further extensions

RDFS inferences

SPARQL queries

Enterprise Data Example

Triples from: HR system and DMS system

Example RDF data

```
:joeBloggs :worksFor :westportCars .
:joeBloggs :salary 80000 .
:johnSmith :worksFor :westportCars .
:johnSmith :salary 40000 .
:joeBloggs :warnedFor :NonAttendance .
:joeBloggs :username "jb" .
:johnSmith :username "js" .
:warnedFor dom :OneWeekSuspended .
```

Enterprise Data Example

Triples from: HR system and DMS system

Example RDF data

```
:joeBloggs :worksFor :westportCars .
:joeBloggs :salary 80000 .
:johnSmith :worksFor :westportCars .
:johnSmith :salary 40000 .
:joeBloggs :warnedFor :NonAttendance .
:joeBloggs :username "jb" .
:johnSmith :username "js" .
:warnedFor dom :OneWeekSuspended .
```

How to add access control to this RDF data?

AC: Entities

Annotation Entities

- Entities present in the original systems:

$$jb, js, hr, emp$$

Represent users, groups, or roles

AC: Entities

Annotation Entities

- Entities present in the original systems:

$$jb, js, hr, emp$$

Represent users, groups, or roles

AC: Entities

Annotation Entities

- Entities present in the original systems:

$$jb, js, hr, emp$$

Represent users, groups, or roles

AC: Entities

Annotation Entities

- Entities present in the original systems:

$$jb, js, hr, emp$$

Represent users, groups, or roles

AC: Entities

Annotation Entities

- Entities present in the original systems:

  $$jb, js, hr, emp$$

  Represent users, groups, or roles
- Attributes:

  $$age = 30$$

AC: Entities

Annotation Entities

- Entities present in the original systems:

$$jb, js, hr, emp$$

  Represent users, groups, or roles
- Attributes:

$$age = 30$$

- Negated literals:

$$\neg js$$

AC: Annotation Values

```
:joeBloggs :salary 80000 :???
```

Annotation values specify entities which:

AC: Annotation Values

```
:joeBloggs :salary 80000 :hr
```

Annotation values specify entities which:

- allowed access: `hr`

AC: Annotation Values

```
:joeBloggs :salary 80000 :-js
```

Annotation values specify entities which:

- allowed access: `hr`
- denied access: `-js`

AC: Annotation Values

```
:joeBloggs :salary 80000 :[hr,-js]
```

Annotation values specify entities which:

- allowed access: `hr`
- denied access: `-js`
- must satisfy each credential: `[hr,-js]`

AC: Annotation Values

```
:joeBloggs :salary 80000 :[[hr,-js],[jb]]
```

Annotation values specify entities which:

- allowed access: `hr`
- denied access: `-js`
- must satisfy each credential: `[hr,-js]`
- must satisfy any set of credentials : `[[hr,-js],[jb]]`

AC: Annotation Evaluation

```
:joeBloggs :salary 80000 :[[hr,-js],[jb]]
```

AC: Annotation Evaluation

```
:joeBloggs :salary 80000 :[[hr,-js],[jb]]
```

non-recursive datalog with negation (nr-datalog$^{\neg}$)

AC: Annotation Evaluation

```
:joeBloggs :salary 80000 :[[hr,-js],[jb]]
```

non-recursive datalog with negation (nr-datalog$^\neg$)

Annotation values are the body of the rules in a Datalog program:

$$\begin{aligned} &\leftarrow \quad hr, \neg js. \\ &\leftarrow \quad jb. \end{aligned}$$

AC: Annotation Evaluation

```
:joeBloggs :salary 80000 :[[hr,-js],[jb]]
```

non-recursive datalog with negation (nr-datalog$^\neg$)

Annotation values are the body of the rules in a Datalog program:

$$\begin{aligned} access &\leftarrow hr, \neg js. \\ access &\leftarrow jb. \end{aligned}$$

rule head: *access*

AC: Annotation Evaluation

```
:joeBloggs :salary 80000 :[[hr,-js],[jb]]
```

non-recursive datalog with negation (nr-datalog$^\neg$)

Annotation values are the body of the rules in a Datalog program:

$$\begin{aligned} access &\leftarrow \quad hr, \neg js. \\ access &\leftarrow \quad jb. \end{aligned}$$

rule head: *access*

given the facts from the authentication service, if *access* holds the user is allowed to access the triple.

Annotated RDF(S) Inferences

Inference rules are **independent** of the annotation domain

Annotated RDF(S) Inferences

Inference rules are **independent** of the annotation domain

RDFS "domain" (dom) rule:

```
   ?prop dom ?Class .
   ?x ?prop ?y .
⇒  ?x type ?Class .
```

```
   :warnedFor dom :OneWeekSuspended .
   :joeBloggs :warnedFor :NonAttendance .
⇒  :joeBloggs type :OneWeekSuspended .
```

Annotated RDF(S) Inferences

Inference rules are **independent** of the annotation domain

Annotated RDFS "domain" (dom) rule:

```
   ?prop dom ?Class .   ?v1
   ?x ?prop ?y .        ?v2
⇒ ?x type ?Class .
```

```
   :warnedFor dom :OneWeekSuspended .      [[emp]]
   :joeBloggs :warnedFor :NonAttendance . [[hr], [jb]]
⇒ :joeBloggs type :OneWeekSuspended .
```

Annotated RDF(S) Inferences

Inference rules are **independent** of the annotation domain

Annotated RDFS "domain" (dom) rule:

```
   ?prop dom ?Class .  ?v1
   ?x ?prop ?y .       ?v2
⇒  ?x type ?Class .    [???]  Annotation of inferred triple?
```

```
   :warnedFor dom :OneWeekSuspended .     [[emp]]
   :joeBloggs :warnedFor :NonAttendance . [[hr], [jb]]
⇒  :joeBloggs type :OneWeekSuspended .    [???]
```

Inferring Annotation Values

```
   :warnedFor dom :OneWeekSuspended .     [[emp]]
   :joeBloggs :warnedFor :NonAttendance . [[hr],[jb]]
⇒ :joeBloggs type :OneWeekSuspended .     [???]
```

Inferring Annotation Values

```
   :warnedFor dom :OneWeekSuspended .     [[emp]]
   :joeBloggs :warnedFor :NonAttendance . [[hr],[jb]]
⇒ :joeBloggs type :OneWeekSuspended .
```

$$access \leftarrow emp.$$

Inferring Annotation Values

```
   :warnedFor dom :OneWeekSuspended .      [[emp]]
   :joeBloggs :warnedFor :NonAttendance . [[hr],[jb]]
⇒ :joeBloggs type :OneWeekSuspended .
```

$$access \leftarrow hr.$$

$$access \leftarrow jb.$$

Inferring Annotation Values

```
   :warnedFor dom :OneWeekSuspended .     [[emp]]
   :joeBloggs :warnedFor :NonAttendance . [[hr],[jb]]
⇒ :joeBloggs type :OneWeekSuspended .    [[emp,hr],[emp,jb]]
```

$$access \leftarrow emp, hr.$$

$$access \leftarrow emp, jb.$$

Inferring Annotation Values

```
   :warnedFor dom :OneWeekSuspended .    [[emp]]
   :joeBloggs :warnedFor :NonAttendance . [[hr],[jb]]
⇒ :joeBloggs type :OneWeekSuspended .    [[emp,hr],[emp,jb]]
```

$$access \leftarrow emp, hr.$$
$$access \leftarrow emp, jb.$$

Conjunction $(+)$ further restricts access to the triple.

Merging Triples

Additional rule to merge annotations of the same triple:

```
   :joeBloggs :salary 80000 .        [[hr,-js]]
   :joeBloggs :salary 80000 .        [[jb]]
⇒  :joeBloggs :salary 80000 .        [???]
```

$$access \leftarrow hr, \neg js.$$

$$access \leftarrow jb.$$

Merging Triples

Additional rule to merge annotations of the same triple:

```
   :joeBloggs :salary 80000 .        [[hr,-js]]
   :joeBloggs :salary 80000 .        [[jb]]
⇒  :joeBloggs :salary 80000 .
```

$$access \leftarrow hr, \neg js.$$

$$access \leftarrow jb.$$

Merging Triples

Additional rule to merge annotations of the same triple:

```
   :joeBloggs :salary 80000 .        [[hr,-js]]
   :joeBloggs :salary 80000 .        [[jb]]
⇒ :joeBloggs :salary 80000 .
```

$$access \leftarrow hr, \neg js.$$

$$access \leftarrow jb.$$

Merging Triples

Additional rule to merge annotations of the same triple:

```
   :joeBloggs :salary 80000 .        [[hr,-js]]
   :joeBloggs :salary 80000 .        [[jb]]
⇒ :joeBloggs :salary 80000 .        [[hr,-js], [jb]]
```

$$access \leftarrow hr, \neg js.$$

$$access \leftarrow jb.$$

Merging Triples

Additional rule to merge annotations of the same triple:

```
   :joeBloggs :salary 80000 .        [[hr,-js]]
   :joeBloggs :salary 80000 .        [[jb]]
⇒ :joeBloggs :salary 80000 .        [[hr,-js], [jb]]
```

$$access \leftarrow hr, \neg js.$$
$$access \leftarrow jb.$$

Disjunction ($\cup$) maintains the access restrictions.

The Access Control Annotation Domain

- the *representation* of the annotations: `[[hr,-js],[jb]]`

The Access Control Annotation Domain

- the *representation* of the annotations: `[[hr,-js],[jb]]`
- operator for inferring of annotations: $+$

The Access Control Annotation Domain

- the *representation* of the annotations: `[[hr,-js],[jb]]`
- operator for inferring of annotations: $+$
- operator for merging annotations: $\cup$

The Access Control Annotation Domain

- the *representation* of the annotations: `[[hr,-js],[jb]]`
- operator for inferring of annotations: $+$
- operator for merging annotations: $\cup$
- *empty* ($\perp$) annotation: `[]`

The Access Control Annotation Domain

- the *representation* of the annotations: `[[hr,-js],[jb]]`
- operator for inferring of annotations: $+$
- operator for merging annotations: $\cup$
- *empty* ($\perp$) annotation: $[]$
- *universal* ($\top$) annotation: $[[jb],[js],\ldots]$ (union of all entities)

Permission Management

```
:johnSmith type :Employee                :[[js]] .
:johnSmith :username "js"               :[[js]] .
:johnSmith :worksFor :westportCars      :[[hr]] .
:johnSmith :salary 40000 .              :[[hr]] .
```

Permission Management

```
:johnSmith type :Employee                 :[[js]] .
:johnSmith :username "js"                 :[[js]] .
:johnSmith :worksFor :westportCars        :[[hr]] .
:johnSmith :salary 40000 .                :[[hr]] .
```

Every :Employee can access their own information:

```
   ?X type :Employee .
   ?X :username ?U .
   ?X ?P ?O               :?λ1 .
⇒  ?X ?P ?O               :?λ1∪ [[?U]] .
```

Permission Management

```
:johnSmith type :Employee                :[[js]] .
:johnSmith :username "js"               :[[js]] .
:johnSmith :worksFor :westportCars      :[[hr],[js]] .
:johnSmith :salary 40000 .              :[[hr],[js]] .
```

Every :Employee can access their own information:

   ?X type :Employee .
   ?X :username ?U .
   ?X ?P ?O   :? λ_1 .
$\Rightarrow$?X ?P ?O   :? $\lambda_1 \cup$ [[?U]] .

AnQL AC Query

AC Annotated RDF(S)

```
:joeBloggs :worksFor :westportCars .
:joeBloggs :salary 80000   .
:johnSmith :worksFor :westportCars .
:johnSmith :salary 40000   .
```

AnQL AC Query

```
SELECT * WHERE { ?person :salary ?salary .    }
```

Query results

?person	?salary

AnQL AC Query

AC Annotated RDF(S)

```
:joeBloggs :worksFor :westportCars .
:joeBloggs :salary 80000   .
:johnSmith :worksFor :westportCars .
:johnSmith :salary 40000   .
```

AnQL AC Query

```
SELECT * WHERE { ?person :salary ?salary .    }
```

Query results

?person	?salary
:joeBloggs	80000
:johnSmith	40000

AnQL AC Query

AC Annotated RDF(S)

```
:joeBloggs :worksFor :westportCars .
:joeBloggs :salary 80000                :[[jb]] .
:johnSmith :worksFor :westportCars .
:johnSmith :salary 40000               :[[js]] .
```

AnQL AC Query

```
SELECT * WHERE { ?person :salary ?salary :[[js]] . }
```

Query results

?person	?salary

AnQL AC Query

AC Annotated RDF(S)

```
:joeBloggs :worksFor :westportCars .
:joeBloggs :salary 80000              :[[jb]] .
:johnSmith :worksFor :westportCars .
:johnSmith :salary 40000             :[[js]] .
```

AnQL AC Query

```
SELECT * WHERE { ?person :salary ?salary :[[js]] . }
```

Facts in the datalog program are: *js*.

Query results

?person	?salary

AnQL AC Query

AC Annotated RDF(S)

```
:joeBloggs :worksFor :westportCars .
:joeBloggs :salary 80000              :[[jb]] .
:johnSmith :worksFor :westportCars .
:johnSmith :salary 40000             :[[js]] .
```

AnQL AC Query

```
SELECT * WHERE { ?person :salary ?salary :[[js]] . }
```

Facts in the datalog program are: *js*.

Query results

?person	?salary
:joeBloggs	80000
:johnSmith	40000

AnQL AC Query

AC Annotated RDF(S)

```
:joeBloggs :worksFor :westportCars .
:joeBloggs :salary 80000             :[[jb]] .
:johnSmith :worksFor :westportCars .
:johnSmith :salary 40000            :[[js]] .
```

AnQL AC Query

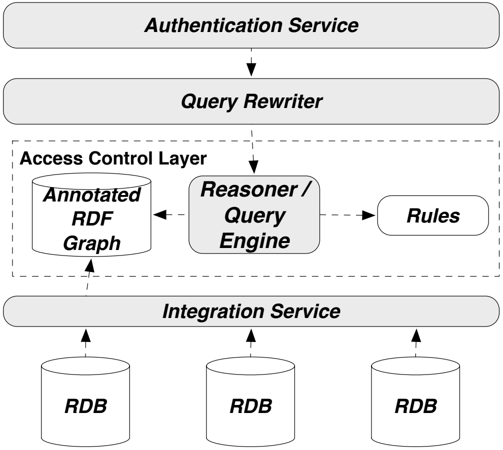
```
SELECT * WHERE { ?person :salary ?salary :[[js]] . }
```

Facts in the datalog program are: *js*.

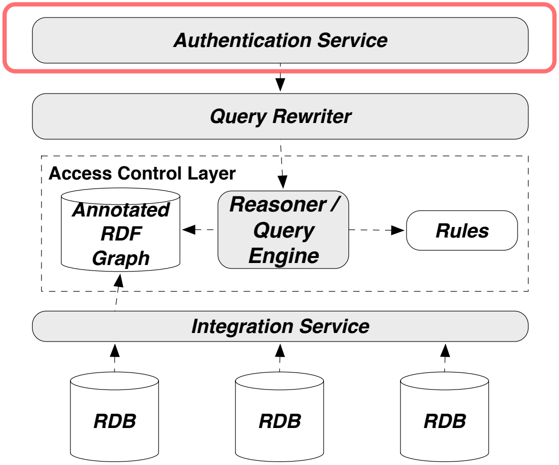
Query results

?person	?salary
:johnSmith	40000

Architecture

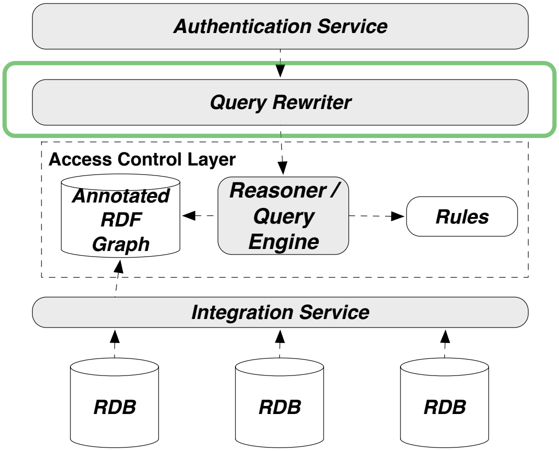


Architecture



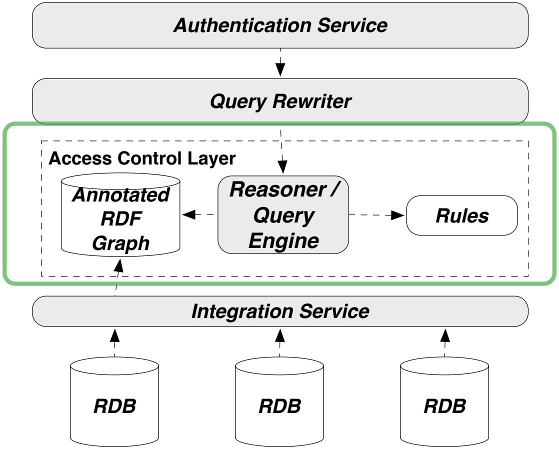
Not covered: Provides the users credentials: [[*js*]]

Architecture

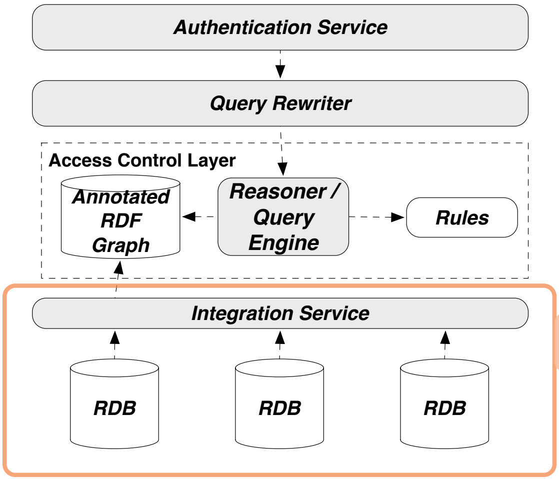


Input: SPARQL queries
Output: AnQL queries

Architecture



Architecture



Not covered: Data converted to RDF

Conclusions

- Represent access control permissions in RDF
- Extension of SPARQL for “secure” querying
- Semantics backwards compatible with RDF and SPARQL

Conclusions

- Represent access control permissions in RDF
- Extension of SPARQL for “secure” querying
- Semantics backwards compatible with RDF and SPARQL

Next steps:

- Integration with triple stores
- Optimisations (storage and querying)
- Permission Modelling and Management
- Combining provenance, trust and access control information

Conclusions

- Represent access control permissions in RDF
- Extension of SPARQL for "secure" querying
- Semantics backwards compatible with RDF and SPARQL

Next steps:

- Integration with triple stores
- Optimisations (storage and querying)
- Permission Modelling and Management
- Combining provenance, trust and access control information

Thank you! Questions?