Access control in RDF Databases

Petr Křemen
petr.kremen@gmail.com

Outline

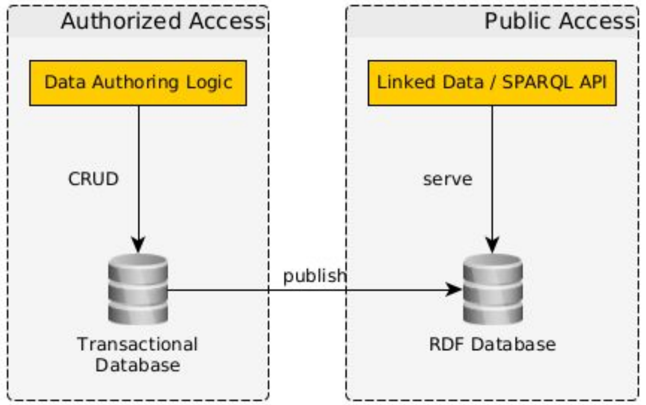
- Motivation

- GraphDB
- Stardog
- Fluree

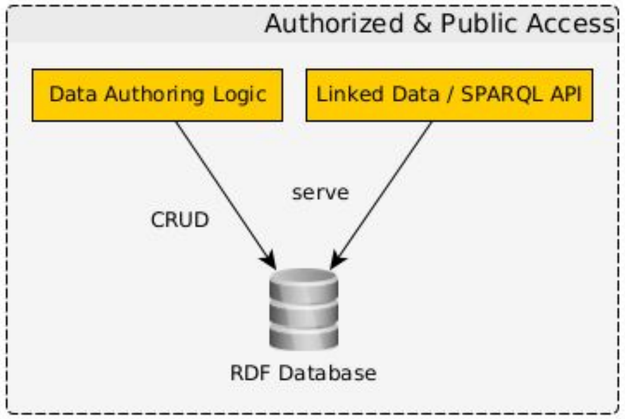
- Conclusions

Motivation

- RDF are traditionally used for hosting linked data, which are historically open
- for linked ***enterprise*** data security needs to be handled.



Current



Desired

Access Control Mechanisms

- RBAC - role-based access control
- ABAC - attribute-based access control
- FGAC - fine-grained access control

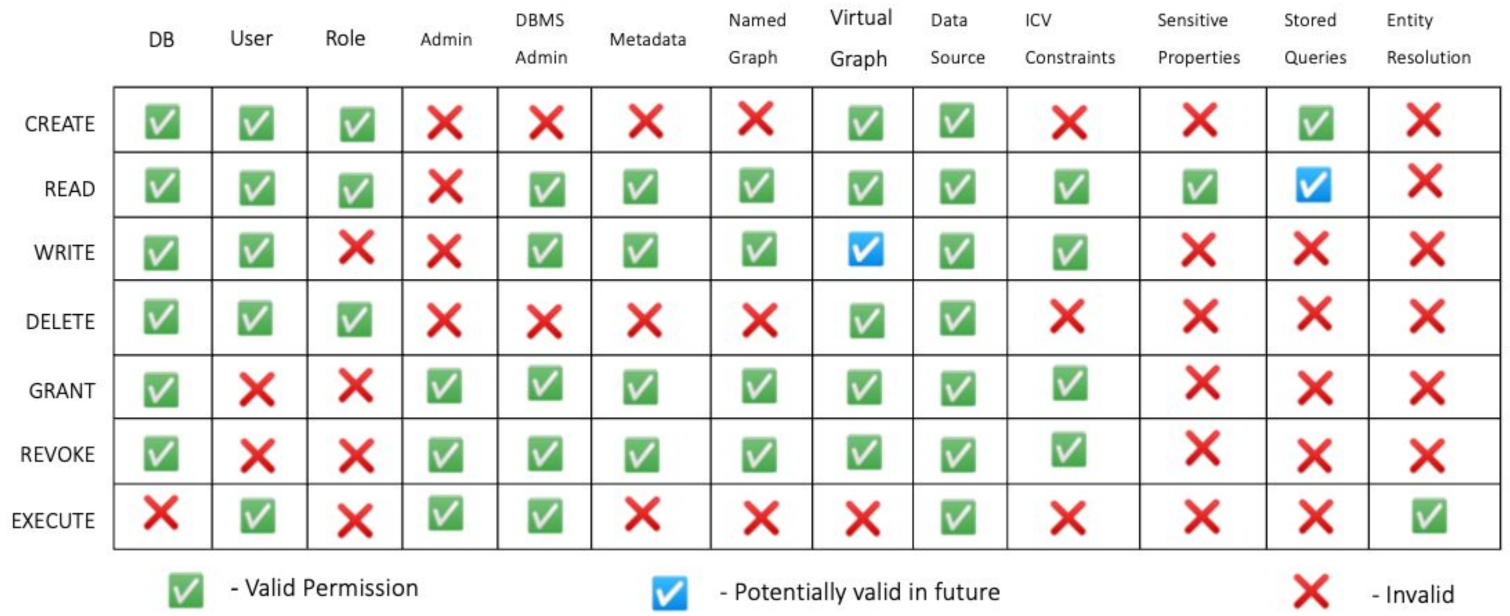
Note: ABAC = FGAC in traditional literature. In the context of RDF, let's use FGAC for the triple-level security.

Security in existing RDF databases

	RBAC	ABAC	FGAC
Allegrograph	yes	yes (FGAC)	yes (security filters)
Amazone Neptune	yes	yes (FGAC)	yes (not RDF-based)
Virtuoso Enterprise	yes	yes (VAL)	no
Stardog	**yes**	**yes**	**no**
GraphDB Enterprise	**yes**	**yes (FGAC)**	**yes**
Anzograph	yes	no	no
Fluree	yes	yes	yes (RelBac, SmartFunction)

Stardog RBAC

Apache Shiro (authentication, authorization, sessions), jBCrypt (hashing)
https://docs.stardog.com/operating-stardog/security/security-model



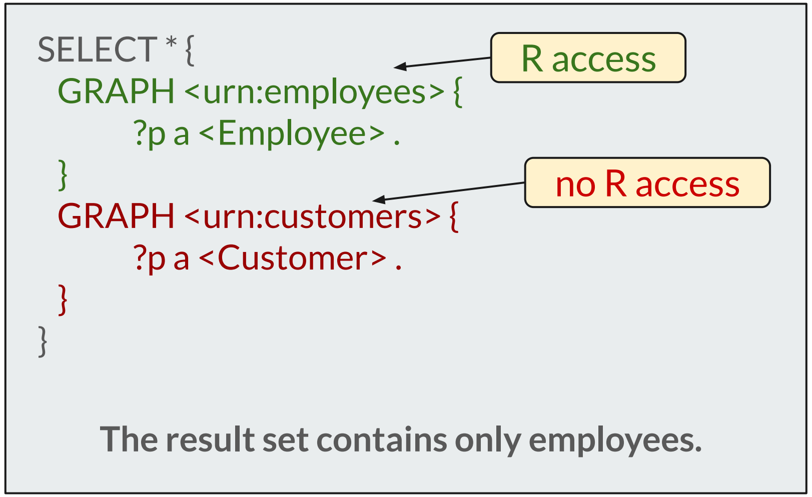
	DB	User	Role	Admin	DBMS Admin	Metadata	Named Graph	Virtual Graph	Data Source	ICV Constraints	Sensitive Properties	Stored Queries	Entity Resolution
CREATE	✅	✅	✅	❌	❌	❌	❌	✅	✅	❌	❌	✅	❌
READ	✅	✅	✅	❌	✅	✅	✅	✅	✅	✅	✅	☑️	❌
WRITE	✅	✅	❌	❌	✅	✅	✅	☑️	✅	✅	❌	❌	❌
DELETE	✅	✅	✅	❌	❌	❌	❌	✅	✅	❌	❌	❌	❌
GRANT	✅	❌	❌	✅	✅	✅	✅	✅	✅	✅	❌	❌	❌
REVOKE	✅	❌	❌	✅	✅	✅	✅	✅	✅	✅	❌	❌	❌
EXECUTE	❌	✅	❌	✅	✅	❌	❌	❌	✅	❌	❌	❌	✅

✅ - Valid Permission ☑️ - Potentially valid in future ❌ - Invalid

Stardog - Named-graph read permission

https://docs.stardog.com/operating-stardog/security/named-graph-security

- **silently ignoring unauthorized graphs**
- only applies to non-schema axioms. Schema axioms are always visible and used for inference.
- i.e. SELECT, ASK, CONSTRUCT, as well as WHERE section of SPARQL UPDATE



```
SELECT * {
  GRAPH <urn:employees> {
        ?p a <Employee> .
  }
  GRAPH <urn:customers> {
        ?p a <Customer> .
  }
}
```

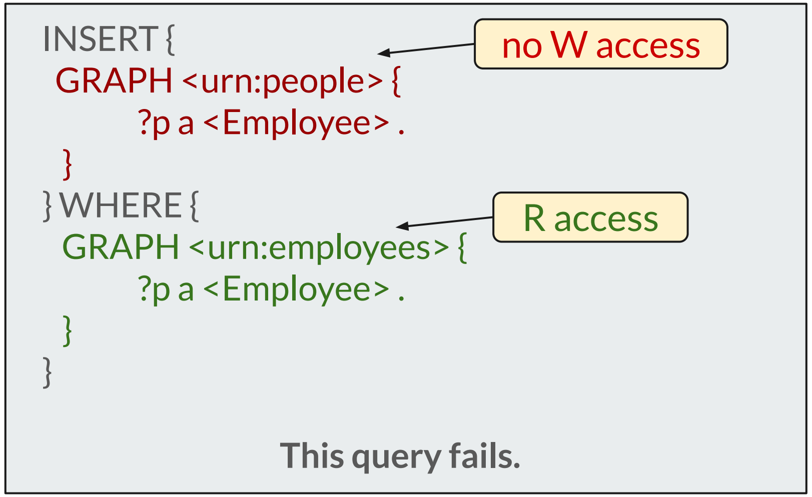
The result set contains only employees.

Stardog - Named-graph write permission

https://docs.stardog.com/operating-stardog/security/named-graph-security

- **writing into unauthorized graph fails**

- transactional, i.e. if an UPDATE tries to modify an unauthorized named graph, the whole UPDATE is rolled back



```
INSERT {
 GRAPH <urn:people> {
        ?p a <Employee> .
 }
} WHERE {
 GRAPH <urn:employees> {
        ?p a <Employee> .
 }
}
```

This query fails.

Stardog - ABAC

Property-based Data Protection
https://docs.stardog.com/operating-stardog/security/fine-grained-security

- restricted **read** access to the set S = { P1, P2, P3 } of ***sensitive* properties** by an equivalent of the following data transformation:

```
INSERT { ?subject ?property ?masked }
DELETE { ?subject ?property ?object }
WHERE {
    ?subject ?property ?object .
    FILTER (?property in S )
    BIND(mask(?object)  AS ?masked)
}
```

configurable,
e.g. constant, or SHA256

```
SELECT * {
  ?p a <Employee> ;
     <firstName> ?fn ;
     <surName> ?sn .
}
```

S={ <surName>}

This query returns surnames masked :

?p	?fn	?sn
<person1>	**"John"**	**"...eba36 ..."**

Stardog - ABAC - broken property chains

- restricted **read** access to the set S = { P1, P2, P3 } of ***sensitive* properties** by an equivalent of the following data transformation:

```
INSERT { ?subject ?property ?masked }
DELETE { ?subject ?property ?object }
WHERE {
    ?subject ?property ?object .
    FILTER (?property in S )
    BIND(mask(?object)  AS ?masked)
}
```

configurable,
e.g. constant, or SHA256

```
SELECT * {
  ?p <address>/<street> ?a .
}
```

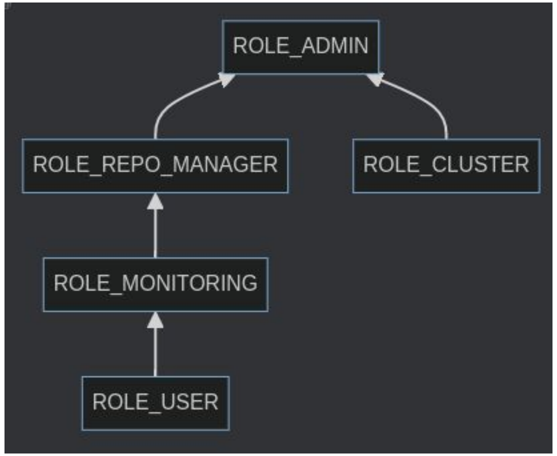
S={ <address>}

This query returns nothing.

Still experimental - no support for restricting W access, values still can be revealed through FTS.

GraphDB - RBAC

- predefined roles



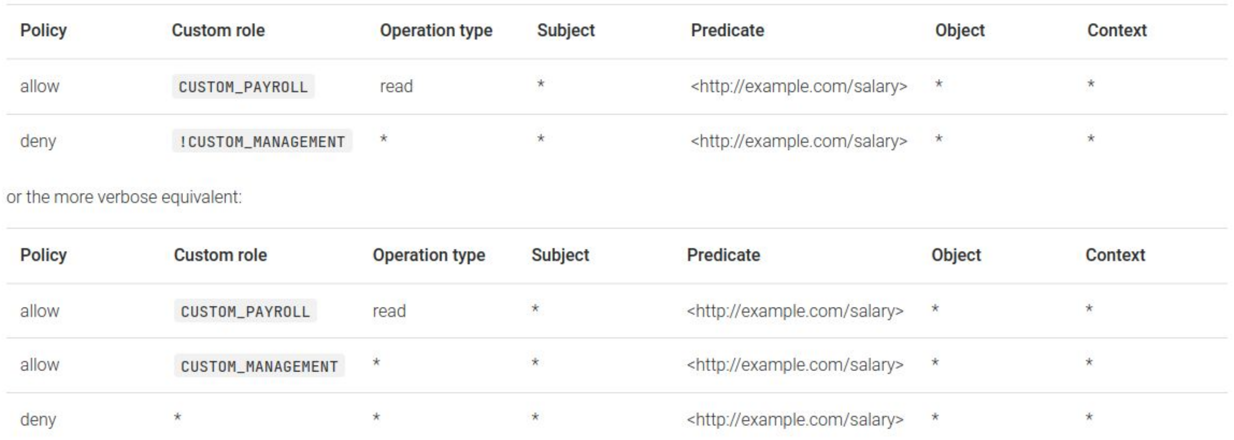
- custom roles - for FGAC

Inherent role and permissions	**Regular user**	**Repository manager**	**Administrator**
Core role	ROLE_USER	ROLE_REPO_MANAGER	ROLE_ADMIN
Bypass FGAC checks	no	yes	yes
Read access to a specific repository	optional	no	no
Read/write access to a specific repository	optional	no	no
Read/write access to all repositories	no	yes	yes
Create, edit, and delete repositories	no	yes	yes
Access monitoring	no	yes	yes
Manage Connectors	no	yes	yes
Manage Users and Access	no	no	yes
Manage the cluster	no	no	yes
Attach remote locations	no	no	yes
View system information	no	no	yes

GraphDB Enterprise - FGAC

https://graphdb.ontotext.com/documentation/10.6/fine-grained-access-control.html

- scopes
  - statement
  - clear graph
  - plugin
  - system
- **reading miss - silently ignoring**
- **writing miss - failure**
- evaluated top-down until match, or end of list (-> allow)



Policy	Custom role	Operation type	Subject	Predicate	Object	Context
allow	`CUSTOM_PAYROLL`	read	*	<http://example.com/salary>	*	*
deny	`!CUSTOM_MANAGEMENT`	*	*	<http://example.com/salary>	*	*

or the more verbose equivalent:

Policy	Custom role	Operation type	Subject	Predicate	Object	Context
allow	`CUSTOM_PAYROLL`	read	*	<http://example.com/salary>	*	*
allow	`CUSTOM_MANAGEMENT`	*	*	<http://example.com/salary>	*	*
deny	*	*	*	<http://example.com/salary>	*	*

GraphDB Enterprise - FGAC

https://graphdb.ontotext.com/documentation/10.6/fine-grained-access-control.html

- inferred statements - all or none
- no support for bnodes
- use-cases
  - ***entities***
  - sensitive predicates
  - named graphs

Security-related directions for RDF

- Web Access Control for SOLID (https://solidproject.org/TR/wac)

- SHACL ACL (https://github.com/SDM-TIB/SHACL-ACL)

- Dataspace Protocol (https://docs.internationaldataspaces.org/ids-knowledgebase/v/dataspace-protocol)

Thank You