Berner
Fachhochschule

Zugriffskontrolle Linked Master Data

Pascal Mainini and Alain Rohrbach

2024

Inhaltsverzeichnis

Einleitung

Mit dem Projekt **Linked Master Data** der Bundeskanzlei soll die Eignung von Linked Data Technologien (LD) für die Bereitstellung von dezentral geführten Stammdatenattributen zwischen Bund, Kantonen und Gemeinden untersucht werden. Dies beinhaltet einerseits die Modellierung der notwendigen Datenstrukturen / Semantik um diese abzubilden und nutzbar zu machen, andererseits auch die Evaluation von geeigneten Authentifizierungs- und Autorisierungsmechanismen oder genereller der *Zugriffskontrolle* auf diese Daten.

Das vorliegende Konzept behandelt nur den Aspekt der Zugriffskontrolle. Es besteht aus verschiedenen Teilen:

1. Einer **Anforderungsanalyse**, welche basierend auf dem heutigen Ist-Zustand, die Anforderungen an ein zukünftiges LD-System ermittelt.
2. Der **Technologieevaluation**: Untersuchung vorhandener Technologien und Konzepte, welche die Umsetzung der Anforderungen ermöglichen / unterstützen.
3. Einer Vorstellung von verschiedenen **Architekturvarianten**, basierend auf der Technologieevaluation.
4. Dem **Konzept** mit der Beschreibung einer möglichen Umsetzung.

Basierend auf dem Konzept wird ebenfalls ein Prototyp entwickelt, welcher die Umsetzung von diesem exemplarisch aufzeigt.

Abgrenzung: LINDAS

Bereits heute können öffentliche Verwaltungen ihre Daten für Bezüger:innen durch die Linked Data Services (LINDAS) des Bundesarchivs bereitstellen. LINDAS ist eine erfolgreiche Anwendung von Linked Data in der Schweiz und bietet Zugriff auf verschiedenste Datensätze (vgl. z.B. [1], [2], [3]).

Im Gegensatz zu den in diesem Projekt erforschten Ansätze, ist LINDAS auf die Verwaltung öffentlicher Daten ausgelegt und bedarf daher auch keiner Zugriffskontrolle.

Anforderungsanalyse

Dieses Kapitel beinhaltet die Anforderungsanalyse. Für diese wird methodisch zuerst der Systemkontext abgegrenzt, sowie die Beteiligten erfasst. Anschliessend wird die Interaktion zwischen den gefundenen Beteiligten und dem System mittels Use Cases beschrieben; ein Use Case dokumentiert hierbei eine spezifische Interaktion mit dem System, z.B. das Abfragen eines Datensatzes. Aus den Use Cases werden im letzten Schritt die konkreten Anforderungen an das System abgeleitet: was für Bedingungen sind für jeden einzelnen Use Case notwendig und hinreichend, damit dieser durch das System ermöglicht wird?

Systemkontext und -Abgrenzung

Der Systemkontext zeigt auf, wo/wie das beschriebene System mit anderen Systemen interagiert, resp. sich in eine bestehende oder angedachte, zukünftige Systemlandschaft eingliedert. Primäres Ziel hierbei ist die Abgrenzung der Systeme. Diese dient dazu, die Reichweite des vorliegenden Konzepts zu definieren und festzuhalten: was ist Teil des Konzepts, und was wird nicht betrachtet?

Dieses Konzept befasst sich primär mit der Zugriffskontrolle auf Personenstammdaten, es sollte sich allerdings ohne grössere Anpassungen auch auf andere Arten von Stammdaten, oder prinzipiell auch auf LD im Allgemeinen, anwenden lassen.

Bestehende Systemlandschaft für Stammdaten

In der Schweiz gibt es Stammdaten für unterschiedliche Entitäten, so z.B. für Personen, Unternehmen und Gebäude. Im föderalen System der Schweiz werden diese in verschiedenen Registern bereits heute dezentral geführt und unterliegen somit auch verschiedenen Zuständigkeiten. So werden bspw. die Register für Unternehmen und Gebäude auf Bundesebene geführt. Auch für Personen gibt es auf Bundesebene verschiedene Register, es werden jedoch zusätzlich auch im Einwohnerregister der entsprechenden Wohngemeinde (oder -Kantons) Attribute, insbesondere das Meldeverhältnis (Wohnsitz, vgl. [4]) geführt. Eine gute Übersicht hierzu bietet [5].

Relevante Daten und Systeme für den Prototypen

Für das Konzept sowie auch für den Prototypen werden fiktive Personenstammdaten angenommen, welche den realen Stammdaten nachempfunden sind. Die Definition der Datenstrukturen, sowie die Erzeugung von Beispieldaten ist nicht Teil dieses Konzepts.

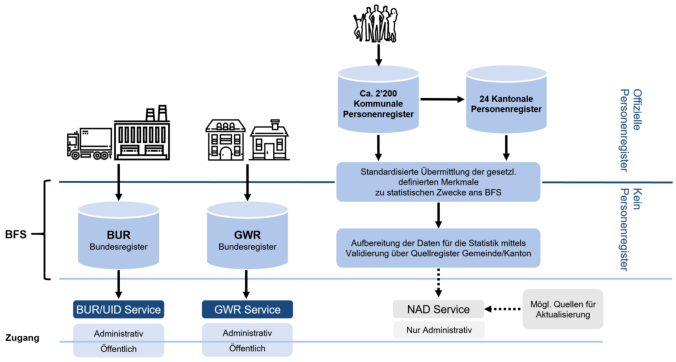

Abbildung 1: Arten von Registern und Positionierung ([5])

Soweit für die einzelnen Use Cases notwendig, werden verschiedene reale Register simuliert. Die Simulation der Register beschränkt sich nur darauf, unterschiedliche Verantwortungsbereiche beim Führen von Attributen abzubilden. Insbesondere werden heutige Verteilmechanismen nicht nachgebildet und die Simulation ist nur so umfassend, wie für den Prototypen notwendig.

Folgende Register sind Teil der Use Cases, welche im Prototypen abgebildet werden:

- Das **UPI-Register** wird vom Bund, von der zentralen Ausgleichsstelle (ZAS), betrieben und verwaltet die *Unique Person Identification*, also die AHV-Nummer, einer natürlichen Person. Es beinhaltet auch noch weitere personenspezifische Attribute.
- Die **Einwohnerregister (EWR)**, welche von den Gemeinden betrieben werden (ca. 2200 Register). EWR führen insbesondere das *Meldeverhältnis*, sowie andere, spezifische Attribute. Hierbei handelt es sich um ein oder mehrere exemplarische Einwohnerregister, welche(s) die grundlegenden Personenstammdaten verwaltet/verwalten. In der Realität gibt es aufgrund der Anzahl Gemeinden in der Schweiz ca. 2200 Einwohnerregister. Diese Zahl ist für das Konzept jedoch nicht von Bedeutung, und es wird nur von einem EWR, oder allenfalls einer geringen Teilmenge, soweit für die Entwicklung des Konzeptes relevant, ausgegangen.

Abgrenzung

Das vorliegende Konzept beschränkt sich auf die o.g. Register. Insbesondere sind die folgend genannten Systeme *nicht* Teil des Konzepts und werden im Prototypen auch nicht umgesetzt:

- Das **Gebäude- und Wohnungsregister (GWR)**: Zusätzlich zu den o.g. Registern sind ebenfalls Daten aus diesem Register notwendig. Es wird jedoch an anderer

Stelle modelliert und ist explizit ausgeschlossen. Wo nötig, werden fiktive Referenzen/Identifikatoren verwendet.[1]

- **Thematische Register**: Weitere, spezifische Register wie z.B. Infostar, ZEMIS, Ordipro, eVera (nicht abschliessend).
- Reale Systeme, welche **Identitäten** bereitstellen, die für die Zugriffskontrolle notwendig sind (IAM-Systeme). Für den Prototypen wird ein solches System als gegeben betrachtet und die benötigten Rollen werden nur simuliert.

Beteiligte (Akteur:innen)

Beteiligte beinhalten relevante Entitäten, welche mit dem angedachten System interagieren. Es kann sich dabei um einzelne Personen, aber auch um Institutionen, Rollen etc. handeln, die direkt mit dem System interagieren. Ebenso werden auch technologische Komponenten anderer Systeme, mit welchen ein Datenaustausch stattfindet oder eine andere Verbindung besteht, als Beteiligte bezeichnet.

Die folgenden Abschnitte beschreiben die für das Konzept relevanten und identifizierten Beteiligten.

B.1: Betreiber:in Einwohnerregister

Die Betreiber:in Einwohnerregister führt das Einwohnerregister einer Gemeinde oder eines Kantons. Sie ist für die Pflege der Daten im Register zuständig und erteilt Zugriffsberechtigungen auf diese. Für jede Zugriffsberechtigung ist eine entsprechende gesetzliche Grundlage notwendig.

Wie beschrieben, wird für dieses Konzept ein exemplarisches EWR modelliert. Dieses ist der Kantonalen Einwohnerplattform (KEP) des Kantons Zürich nachempfunden. Es handelt sich bei der KEP um einen Zusammenzug der einzelnen EWR des Kantons Zürich und nicht um ein eigenständiges EWR. Zugriffsberechtigungen werden hier zentral vergeben, die Webseite der KEP ([6]) bietet eine Übersicht und listet aktuell 58 Datenbezüger:innen. [2] Eine ähnliche Plattform wird z.B. auch im Kanton Bern betrieben (GERES-Plattform, [7]).

[1] Entschieden Workshop 21. März 2024.
[2] Stand März 2024

B.2: Betreiber:in UPI-Register

Das UPI-Register wird von der Zentralen Ausgleichsstelle (ZAS) betrieben, welche dem Eidgenössischen Finanzdepartement (EFD) unterstellt ist. Eine Abfrage von Datensätzen ist entweder interaktiv mittels UPIViewer, oder aber über die UPIServices (Webservice) möglich. Zugriff auf UPIViewer und UPIServices ist nur für sog. systematische Benutzer:innen möglich, welche entsprechend registriert sind. [8]


Letztes Update: 03.09.2018 17:51:23

Identifikationsnummer:	756.2951.5346.43
Name:	Muster
Vorname:	Heidi
Geburtsdatum:	01.04.1900
Geschlecht:	♀ Frau
Nationalität:	SCHWEIZ
Ledigenname:	
Name gemäss Reisepass:	
Geburtsort:	Maienfeld, SCHWEIZ
Name der Mutter:	Muster, Maria
Name des Vaters:	Muster, Tobias
⚠ Todesdatum:	01.04.2000

Sofern die oben angezeigten Daten Fehler enthalten und Sie diese berichtigen lassen wollen, bitten wir Sie sich an die ZAS zu wenden.
upi@zas.admin.ch

Abbildung 2: Beispiel Resultat UPIViewer ([9])

B.3: Erhebungsstelle Radio und Fernsehen (ES)

Die Erhebungsstelle Radio und Fernsehen (ES) ist eine *Datenbezügerin*. Sie ist zuständig für die Erhebung der Haushaltsabgabe (Radio- und Fernsehgebühren) und benötigt hierzu sowohl Daten aus dem UPI-Register wie auch von den EWRs. Die ES ist eine Stelle ausserhalb der Bundesverwaltung, welche vom BAKOM ernannt wird, die gesetzliche Grundlage hierzu findet sich im Bundesgesetz über Radio und Fernsehen ([10], Art. 69). Stand 2024 wird diese Aufgabe von der Serafe AG übernommen.

Use Cases

In diesem Abschnitt werden die für den Prototypen relevanten Use Cases festgehalten. Ein Use Case beschreibt die Interaktion von Beteiligten mit dem geplanten System.

Naturgemäss gibt es eine Vielzahl *möglicher* Use Cases für Personenstammdaten, so wären bspw. alleine für die KEP Zürich aktuell mindestens 58 Use Cases (bezogen auf die aktuelle Anzahl Datenbezüger:innen) möglich. Exemplarisch wird für den Prototypen nur ein Use Case (UC.1) umgesetzt. Zwei zusätzliche Use Cases, welche während der Analyse identifiziert wurden, werden hier zu Dokumentationszwecken festgehalten (UC.2 und UC.3).

UC.1: Erhebung Radio- und Fernsehgebühren

Um die Radio- und Fernsehgebühren *pro Haushalt* zu erheben, ist eine Erhebungsstelle gemäss [10], Art. 69g berechtigt, Daten zu den Haushalten aus den Einwohnerregistern sowie auch aus Ordipro zu erheben.[3] Ebenso ist sie berechtigt, die AHV-Nummer systematisch zu verwenden und erhält somit Zugriff aufs UPI-Register.

Es handelt sich um eine **Bulk-Abfrage**: alle Datensätze bezüglich aktueller Meldeverhältnisse werden übermittelt.

UC.2: Steuerveranlagung von Einzelunternehmen

Die eidgenössische Steuerverwaltung muss zur Steuerveranlagung von Einzelunternehmen den aktuellen Wohnsitz einer bestimmten natürlichen Person ermitteln. Hierfür verwendet die Steuerverwaltung die AHV-Nummer der Person und ermittelt beim UPI-Register die aktuelle Wohngemeinde für das primäre Meldeverhältnis. Aus der Antwort des UPI-Registers fragt die Steuerverwaltung bei der entsprechenden Gemeinde/Kanton das primäre Meldeverhältnis ab.

Es handelt sich um eine **Einzelabfrage**: Datensätze zu individuellen, natürlichen Personen werden übermittelt.

UC.3: Personensuche Feuerwehr

Für eine Überprüfung von vermissten Personen benötigt die Feuerwehr / der Katastrophenschutz alle gemeldeten Personen (Erst- und Zweitwohnsitz) eines bestimmten Gebäudes, allenfalls auch aller Gebäude eines Perimeters.[4] Die Berechtigung wird nur im Fall eines aktuellen Vorfalls gewährt. Für alle Personen mit mehr als einem Meldeverhältnis werden auch alle weiteren Meldeverhältnisse in der Schweiz benötigt (zwecks Kontaktaufnahme).

Es erfolgt eine Anfrage bei der Gemeinde für die natürlichen Personen (AHV-Nummern, Namen und Kontaktdaten), welche einem bestimmten Gebäude / einer Wohnung zugeordnet sind. Anhand der AHV-Nummer werden beim UPI-Register weitere aktuelle Meldeverhältnisse abgefragt. Sollte es solche geben, werden bei den weiteren Gemeinden die Namen und Kontaktdaten abgefragt.

Es handelt sich um eine **Bulk-Abfrage**: alle *betroffenen* Datensätze des Einwohnerregisters werden übermittelt. Ebenso werden mehrere Datensätze beim UPI-Register und allenfalls betroffenen, weiteren Einwohnerregistern abgefragt.

[3] Ordipro ist nicht Teil des Prototypen, siehe Abgrenzung.
[4] Es gibt bereits spezifische, Linked Data-basierte Systeme im Feuerwehreinsatz, vgl. [11].

Offene Frage: Der Zugriff beim UPI-Register soll dabei nur für den aktuellen Vorfall und die betroffenen AHV-Nummern gewährt werden. Hierzu müsste das UPI aber wissen (in den Daten haben), welche AHV-Nummern effektiv betroffen sind. Andernfalls wäre es (für die Feuerwehr) möglich, UPI-Abfragen mit beliebigen, gewählten AHV-Nummern zu tätigen und so Zugriff auf mehr Daten als vorgesehen zu erhalten.

Anforderungen

- **Berechtigung aufgrund einer Rechtsgrundlage**
- Berechtigung aufgrund eines Falles
  - *Möglicherweise fallspezifisch unterschiedliche Attribute*
- Berechtigung aufgrund eines Vertrags mit der öffentlichen Verwaltung z.B. Erhebungsstelle Radio-/Fernsehgebühren)
- Berechtigung wird durch Auftraggeber:in gesteuert

Hierbei muss zwischen abfragenden und aktiv verteilenden Verfahren unterschieden werden. Im Prototyp werden nur abfragende Verfahren betrachtet.

Technologieevaluation

Dieses Kapitel analysiert, wie die Zugriffskontrolle in verschiedenen, gängigen Triple Stores ausgestaltet ist. Diese verwenden hierzu unterschiedliche Paradigmen:

- **ACLs (Access Control Lists)**

  Es werden Listen verwendet, um die Zugriffskontrolle auf eine Ressource zu beschreiben.

- **RBAC (Role-Based Access Control)**

  Art der Zugriffskontrolle, bei welcher der Zugang zu Ressourcen anhand von *Rollen* erfolgt. Nutzer:innen werden einer bestimmten Rolle zugeordnet, welche deren Zugriffsrechte steuert.

- **ABAC (Attribute-Based Access Control)**

  Vergleichbar mit RBAC, wobei der Zugang anhand von Attributen statt Rollen erfolgt. Nutzer:innen können dabei mehrere Attribute zugewiesen werden.

- **FGAC (Fine Grained Access Control)**

  Art der Zugriffskontrolle, bei welcher die Berechtigung auf spezifische Datensätze bezogen ist. Bei Linked Data kann hiermit der Zugriff basierend auf Attributen einzelner Triples gesteuert werden.

Bezogen auf die Anforderungen kann bereits einleitend festgehalten werden, dass sich eigentlich nur FGAC für den Prototypen eignet.

Übersicht Triplestores

Untenstehende Tabelle listet die untersuchten Triplestores auf, (technische) Details sind den nachfolgenden Abschnitten zu entnehmen.

Zusammengefasst lässt sich feststellen, dass die Umsetzung der Zugriffskontrolle stark vom jeweiligen Triplestore abhängt. Es gibt grosse Unterschiede in deren Umsetzung, sowie auch in der Konfigurierbarkeit.

Umfassende Möglichkeiten zur Zugriffskontrolle bieten Fuseki2 (ABAC) sowie GraphDB (FGAC). Eine besondere Stellung nimmt das Jena Permissions Framework ein, mit welchem programmatisch präzise Zugriffskontrollen für Fuseki2 realisiert werden können.

Zur *Authentifizierung* bieten die meisten der Untersuchten Triple Stores verschiedene Mechanismen, bspw. die Anbindung mittels LDAP und OAuth.

Produkt	Open Source	Lizenz	Access Control Features
Fuseki2	ja	Apache-2.0	ACL, Graph-ACL, ABAC
Telicent	ja (?)	(?)	(?)
Oxigraph	ja	Apache-2.0, MIT	keine
Stardog	nein	Enterprise, Academic	RBAC, FGAC
Oracle	nein	(?)	(?)
GraphDB	nein	Enterprise, Standard, Free	RBAC, FGAC
Parliament	ja	BSD-3-Clause	(?)
Virtuoso	nein	Enterprise	ACL, RBAC, FGAC
OpenLink Virtuoso	ja	GPL-2.0	ACL, RBAC, FGAC

Zu Feldern welche mit (?) markiert sind, wurden entweder keine oder nur unzureichende Informationen gefunden.

Fuseki2

Access Control Eine detaillierte Beschreibung der Zugriffskontrolle kann hier gefunden werden.

Fuseki Main unterstützt grundsätzlich `basic` und `digest` Authentifizierung. Es ist allerdings möglich andere Authentifizierungsarten durch Eclipse Jetty vorzunehmen.

Die Zugriffskontrolle erfolgt durch ein Role-Based Access Model und kann auf den folgenden Ebenen konfiguriert werden:

Server, verwaltet Zugriff über alle Anfragen:

```
<#server> fuseki:allowedUsers "user1", "user2".
```

Datasets, verwaltet Zugriff auf alle Endpunkte eines Datasets:

```
<#service> fuseki:allowedUsers "user1", "user2".
```

Endpoints, verwaltet Zugriff auf spezifischen Endpunkten eines Datasets:

```
<#service> fuseki:endpoint [
  fuseki:allowedUsers "user1", "user2"
].
```

Graphs, verwaltet Zugriff auf bestimmten Graphen.

Jena Permissions - Security Evaluator Jena Permissions ist ein Framework zur Implementierung von Sicherheitsstrategien auf Graphen. Es fängt Aufrufe an die Graph- oder Modellschnittstelle transparent ab, wertet Zugriffsbeschränkungen aus und erlaubt oder verweigert den Zugriff.

Unter anderem kann durch die Implementierung der folgenden Methode der Zugriff auf einzelne Triples präzise gesteuert werden.

```
public boolean evaluate(Object principal,
                        Action action,
                        Node graphIRI,
                        Triple triple)
                        throws AuthenticationRequiredException;
```

Attribute-Based Access Control Attribute-Based Access Control ist ein Erweiterungsmodul für Apache Jena Fuseki, welches Security Labeling für RDF Triple bereitstellt.

Die Zugriffskontrolle erfolgt über ein ABAC Model, wobei Usern verschiedene Attribute zugewiesen werden können:

```
PREFIX authz: <http://telicent.io/security#>

[] authz:user "user1" ;
    authz:userAttribute "manager";
    authz:userAttribute "project-Y";
    .

[] authz:user "user2" ;
    authz:userAttribute "engineer";
    .

[] authz:user "user3" ;
    authz:userAttribute "unused";
    .
```

Es ist zudem möglich hierarchische Attributwerte zu definieren, wobei User mit einer höheren Berechtigung auf weniger geschützte Inhalte zugreifen können:

```
[] authz:hierarchy [
  authz:attribute "status" ;
  authz:attributeValues "public, confidential, sensitive, private"
];
```

Die Zugriffsrechte werden durch Labels und Patterns definiert. Ein Label beschreibt jeweils die Zugriffsrechte, welche für die vom Pattern erkannten Triples gelten.

Das folgende Beispiel zeigt, wie der Zugriff auf Triple-Ebene gesteuert werden kann.

- Das Triple `:person4321 :phone "0400 111 333"` ist für alle User zugreifbar.
- Das Triple `:person4321 :phone "0400 111 222"` ist zugreifbar für alle User mit dem Attribut `employee`.
- Alle anderen Triples mit dem Prädikat `:phone` sind für niemanden zugreifbar.

```
PREFIX rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
PREFIX authz:  <http://telicent.io/security#>
PREFIX ANY     <jena:ANY>
PREFIX :        <http://example/>

:person4321 rdf:type foaf:Person;
    :phone "0400 111 222" ;
    :phone "0400 111 333" ;
    rdfs:label "Jones" ;
    .

GRAPH authz:labels {
    [ authz:pattern ':person4321 :phone "0400 111 333"' ;
      authz:label "*" ] .

    [ authz:pattern ':person4321 :phone "0400 111 222"' ;
      authz:label "employee" ] .

    [ authz:pattern 'ANY :phone ANY' ;
      authz:label "!" ] .
}
```

Stardog

Stardog unterstützt Authentifizierung durch OAuth 2.0 und JWT Tokens. LDAP kann ebenfalls zur Authentifizierung genutzt werden.

Fine Grained Access Control Stardog bietet Fine Grained Security für sensible Eigenschaften. Unter Sensiblen Eigenschaften sind Prädikate von Triples zu verstehen, welche nur für bestimmte Gruppen zugänglich sind. Der Zugriff wird in einer Sensitive Property List verwaltet.

Angenommen die Eigenschaft `property` eines Triples `{ subject, property, object }` ist als sensibel aufgelistet. Ein autorisierter User kann direkt auf dieses Triple zugreifen,

während für nicht autorisierte User die sensible Eigenschaft zuerst maskiert und ein neues Triple `{ subject, property, masked }` zurückgegeben wird.

Die Dokumentation beschreibt die folgenden Limitationen:

> Values of sensitive nodes can be revealed through zero-length paths (e.g., queries like `?s :p? ?o`) and full-text search. Technically, these are violations of the definition based on the Update query. However, it should not be possible to see connections of these values to other nodes via sensitive properties.
>
> Values of sensitive nodes can be revealed through edge properties. The sensitive properties feature should not be used when the edge properties feature is enabled.
>
> Sensitive property permissions only limit read access. Users without read permission to sensitive properties can still write triples with those properties, even though they would not be able to read the values they have written.

Oxigraph

Oxigraph bietet keine native Unterstützung für Authentifizierung oder Autorisierung.

In einem Issue wird über eine mögliche Umsetzung eines Authentifizierungsprozesses diskutiert.

GraphDB

GraphDB nutzt ein Hierarchical Role-Based Access Control Model zur Autorisierung.

GraphDB unterstütz sowohl die Integration von LDAP als auch OAuth zur Authentifizierung und Autorisierung.

Fine Grained Access Control GraphDB unterstützt eine Vielzahl von Fine Grained Access Control Funktionen, einschliesslich Zugriffskontrolle auf Triple-Ebene.

Dadurch ist es möglich, den Lese-/Schreibzugriff auf der Grundlage eines beliebigen Teils eines RDF-Statements zuzulassen oder zu verweigern. Ein Statement umschreibt hierbei ein Quad, welcher aus dem Subjekt, dem Prädikat, dem Objekt und dem Named Graph besteht.

Das folgende Beispiel zeigt, wie der Zugriff auf einen einzelnen Graphen beschränkt werden kann. User mit der Rolle `GROUP1` können auf sämtliche Triples im Graphen `<http://example.com/project1>` zugreifen.

```
{
  "policy": "deny",
  "role": "*",
  "scope": "statement",
  "operation": "*",
  "subject": "*",
  "predicate": "*",
  "object": "*",
  "context": "*"
},
{
  "policy": "allow",
  "role": "GROUP1",
  "scope": "statement",
  "operation": "*",
  "subject": "*",
  "predicate": "*",
  "object": "*",
  "context": "<http://example.com/project1>"
}
```

Der Zugriff kann weiter auf ein Triple mit einem bestimmten Prädikat eingeschränkt werden. Im folgenden Beispiel können User mit der Rolle `GROUP1` nur noch auf Triples mit dem Prädikat `<http://example.org/name>` zugreifen.

```
{
  "policy": "deny",
  "role": "*",
  "scope": "statement",
  "operation": "*",
  "subject": "*",
  "predicate": "*",
  "object": "*",
  "context": "*"
},
{
  "policy": "allow",
  "role": "GROUP1",
  "scope": "statement",
  "operation": "read",
  "subject": "*",
  "predicate": "<http://example.com/name>",
  "object": "*",
  "context": "<http://example.com/project1>"
```

```
}
```

Weitere Beispiele können hier gefunden werden.

Virtuoso

Virtuoso unterstützt OAuth und LDAP Integration.

Die vollständige Dokumentation kann hier eingesehen werden.

Solid Project

Neben eigentlichen Triplestores gibt es noch andere interessante Ansätze. Hervorzuheben ist hierbei insbesondere das Solid Project.

Solid ist ein offener Standard, um persönliche Daten im Web zu strukturieren und zu verteilen. Er soll Nutzer:innen ermöglichen Daten über selbstgehostete Server (sogenannte Pods) für Datenbezüger:innen wie Apps oder Organisationen freizugeben. Solid stützt sich dabei auf Linked Data, um eine hohe Interoperabilität zwischen Pods zu gewährleisten. Ein zentraler Punkt von Solid liegt darin, die Zugriffsverwaltung auf die eigenen Daten individuell gestalten zu können und den Zugriff für Drittparteien mit beliebiger Granularität einzuschränken. Das Solid Projekt ist nicht auf einen bestimmten Triple Store ausgerichtet.

Solid deckt sich in gewissen Bereichen mit den Anforderungen dieses Projekts, und könnte als möglicher Ansatz zur Stammdatenverwaltung dienen. Es sind jedoch weitere Abklärungen, z.B. bezüglich der Granularität der Zugriffsbeschränkungen notwendig. Details zum Zugriffsschutz finden sich im Solid Protocol Report. Dieser bleibt bezüglich der Granularität vage, gemäss einer Forumsdiskussion sollten Beschränkungen allerdings auch auf Triple-Stufe möglich sein.

Berechtigungen werden mithilfe von ACL-Ontologien beschrieben. Diese werden entweder durch Web Access Control (WAC) und/oder Access Control Policy (ACP) umgesetzt. Die Dokumentation verlinkt auf die folgenden Ontologien:

- WAC: http://www.w3.org/ns/auth/acl
- ACP: http://www.w3.org/ns/solid/acp

Architekturvarianten

Aufgrund der verteilten Architektur sowie der unterschiedlichen Eigenschaften der Triplestores sind für die Umsetzung einer Linked Data Infrastruktur für Personenstammdaten verschiedene Architekturen möglich. Eine Auswahl von Varianten wird im Folgenden beschrieben und miteinander verglichen.

Für den Prototypen wird die Variante SPARQL-Proxy umgesetzt.[5] Bei einer realen Umsetzung ist es jedoch auch denkbar, dass verschiedene Betreiber:innen unterschiedliche Lösungen umsetzen. Allen Varianten gemeinsam ist jedoch, dass der Zugriff auf die individuellen Triplestores mittels SPARQL 1.1 ([12]) erfolgt. Ebenso werden alle für den Zugriff erforderlichen Identitäten als gegeben angenommen (vgl. Abgrenzung) und die Unterstützung entsprechender IAM-Schnittstellen bei den verwendeten Triplestores als erforderlich angesehen.

Standardisierung einheitlicher Triplestore

In dieser Variante wird für das gesamte Ökosystem ein einheitlicher Triplestore evaluiert und als Standard definiert. Für diesen werden dann entsprechende Konfigurationen sowie Dokumentation (bspw. Schulungsunterlagen für Betreiber:innen) erarbeitet. Sollte der gewählte Triplestore die Möglichkeiten zum Zugriffsschutz nicht, oder nur unzureichend, bieten könnte z.B. per zentraler Beschaffung eine Anpassung / Ergänzung erwirkt werden.

Vorteile

- Potentielle Umsetzung mit wenig Entwicklungsaufwand.
- Aufbau von Wissen für das gesamte Ökosystem, z.B. Best Practices etc.

Nachteile

- Standardisierter Triplestore eventuell nicht für alle Betreiber:innen sinnvoll (unterschiedliche Anforderungen z.B. interne Schnittstellen, Skalierbarkeit, Preis).
- Bedingt vermutlich Ausschreibung / Beschaffung von Anpassungen bestehender Triplestores.

[5] Entscheid Workshop 30. April 2024.

Config-Engine

In dieser Variante können verschiedene Triplestores eingesetzt werden, eine *Config-Engine* generiert für die verschiedenen Triplestores entsprechende Konfigurationen für die Zugriffskontrolle. Es wird ein unabhängiges Format zur Spezifikation der Zugriffsrechte genutzt (evtl. bestehende Ontologien / sonst. Formate oder Eigenentwicklung). Für jeden zu unterstützenden Triplestore wird ein *Konfigurations-Adapter* entwickelt, welcher eine Übersetzung des unabhängigen Formats in die Triplestore-spezifische Konfiguration vornimmt.

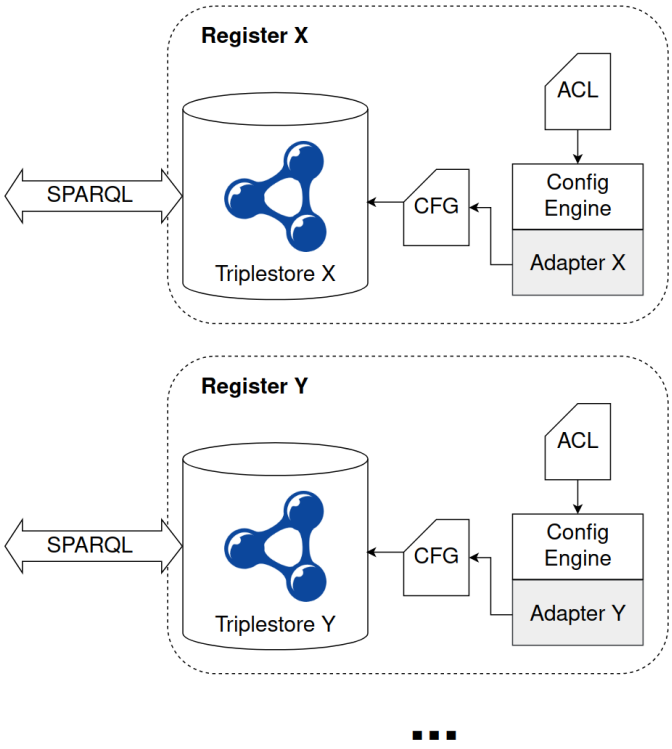

Abbildung 3: Config-Engine

Vorteile

- Einsatz verschiedener Triplestores möglich, sofern Konfigurations-Adapter implementiert.
- Von konkreten Triplestores unabhängige Konfiguration der Zugriffskontrolle.

Nachteile

- Initialer Entwicklungsaufwand pro zu unterstützenden Triplestore.

- Fortlaufender Entwicklungsaufwand für die Anpassung an neue Versionen von bereits angebundenen Triplestores.
- Komplexität.

SPARQL-Proxy

Ein SPARQL-Proxy ist ein SPARQL-Endpunkt, welcher dem eigentlichen Endpunkt eines Registers vorgeschaltet wird (analog z.B. HTTP-Proxy vor Webserver). Die Abfrage von aussen erfolgt ausschliesslich über den Proxy, dieser kann die Anfrage (die SPARQL-Query) verändern bevor sie an den eigentlichen Endpunkt weitergeleitet wird, oder auch die Beantwortung verweigern. Ein Konzept für einen solchen SPARQL-Proxy zum Zugriffsschutz wird auch in [13] beschrieben und in [14] weiterentwickelt.

Für das Ökosystem würde der Proxy den Zugriffschutz übernehmen, d.h. Authentifizierung (Identitätsprüfung der Nutzenden) sowie Authorisierung (prüfen der Zugriffsrechte mit FGAC). Wie bei Config-Engine wird ein unabhängiges Format zur Spezifizierung der Zugriffsrechte genutzt.

Vorteile

- Universelle, von den eingesetzten Triplestores unabhängige Lösung.
- Mögliche Anbindung von anderen Systemen anstelle Triplestores, z.B. relationale Datenbanken ([15],[16]) und/oder SPHN Connector ([17]) etc.

Nachteile

- Komplexität Umsetzung (z.B. Implementation SPARQL-Endpunkt).
- Grössere und komplexere Systemlandschaft (bspw. bezüglich Verfügbarkeit und Durchsatz).
- Weiterentwicklung: notwendige Umsetzung von neuen SPARQL-Standards und anderen, benötigten Technologien.

Datenmodellierung / Separation

In dieser Variante wird der Zugriffschutz auf die Daten durch entsprechende Modellierung und/oder Separation der Daten erreicht. Hierzu sind verschiedene Lösungen denkbar, bspw.:

- Duplikation/Separation in verschiedene Named Graphs, z.B. pro Datenbezüger:in oder sogar pro Triple.
- Zugriffsschutz durch individuelle (SPARQL-)Endpunkte pro Datenbezüger:in oder entsprechende ACLs auf (Named-)Graph-Ebene.

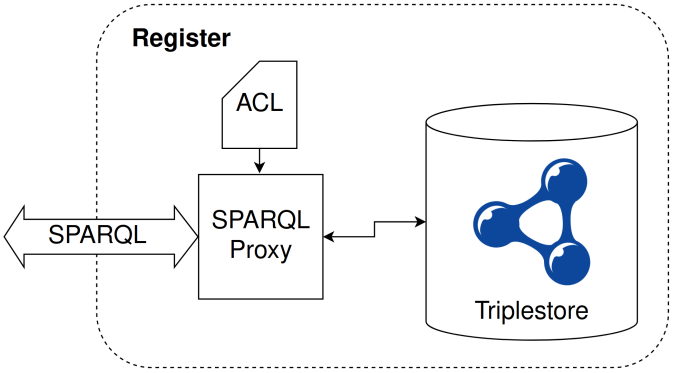

Abbildung 4: SPARQL-Proxy

- Modellierung der Daten: Durch entsprechende Verlinkung oder das Weglassen einer solchen (z.B. nur unidirektionale Verlinkung) wird der Zugriff eingeschränkt.

Solche Konzepte werden teilweise auch bei Fedlex ([18]) verwendet.

Vorteile

- Mit verschiedenen Triplestores ohne Anpassung umsetzbar.

Nachteile

- Erhöhte Komplexität bei der Datenbewirtschaftung.
- Fehleranfälligkeit.

Verschlüsselung der Daten

Eine weitere Möglichkeit, Zugriffsschutz auf Linked Data umzusetzen, besteht darin, die effektiven Daten pro Zugriffsberechtigung, also z.B. pro Datenbezüger:in verschlüsselt abzulegen.

Dieser Ansatz wurde nicht weiterverfolgt, wird hier jedoch der Vollständigkeit halber aufgelistet. Siehe z.B. [19].

Konzept SPARQL-Proxy

Gemäss Entscheid (Architekturvarianten) wird für das Projekt ein prototypischer SPARQL-Proxy entwickelt. Die Funktion des Proxies kann grob in die folgenden Schritte / Teilaspekte aufgeteilt werden:

- Der Proxy stellt einen SPARQL-Endpunkt zur Verfügung, welcher SPARQL-Abfragen über HTTP verarbeitet.
- Empfangene Abfragen werden mit zusätzlichen Ausdrücken erweitert, welche eine Einschränkung der Abfrage gemäss den Berechtigungen der Datennutzer:in bewirken.
- Die so erweiterte Abfrage wird als neue SPARQL-Abfrage an den *Backend-Triplestore* gesendet (an das eigentliche Register).
- Die vom Backend-Triplestore erhaltenen Daten werden temporär in einen zum Proxy gehörenden, *internen Triplestore* (in-memory) abgefüllt.
- Die ursprüngliche Abfrage der Datennutzer:in wird erneut auf den nun im internen Triplestore vorhandenen Daten ausgeführt.
- Die Antwort vom internen Triplestore wird als Antwort auf die ursprüngliche Abfrage zurückgesendet.

Diese Form der Umsetzung macht Abstriche bei der Effizienz, da z.B. die Daten zweimal abgefragt werden. Sie ist jedoch bedeutend einfacher umzusetzen im Vergleich zu einer Variante, welche die originale Abfrage direkt modifiziert, resp. die zurückgesandten Daten filtert. Da es sich um eine prototypische Implementation handelt, wurde der Einfachheit der Implementation der Vorzug gegeben. Funktional ergeben sich dadurch keine Unterschiede.

Architektur und Komponenten

Der Proxy besteht im Wesentlichen aus vier Komponenten, welche im Folgenden beschrieben werden. Die untenstehende Grafik zeigt die Zusammenhänge der jeweiligen Komponenten auf.

Intent Provider

Die Aufgabe des Intent Providers ist es, die wesentlichen Informationen aus der ursprünglichen SPARQL-Abfrage zu extrahieren, aufzuarbeiten und daraus einen sog. *Intent* zu formulieren. Der Begriff wird auch von [13] verwendet und fasst die Information über das Abgefragte (Intent-Query) und die Berechtigung der abfragenden Partei (Intent-Role) zusammen. Der Intent Provider ist für die Authentifizierung zuständig und ordnet der Anfrage bei Erfolg eine entsprechende Rolle zu.

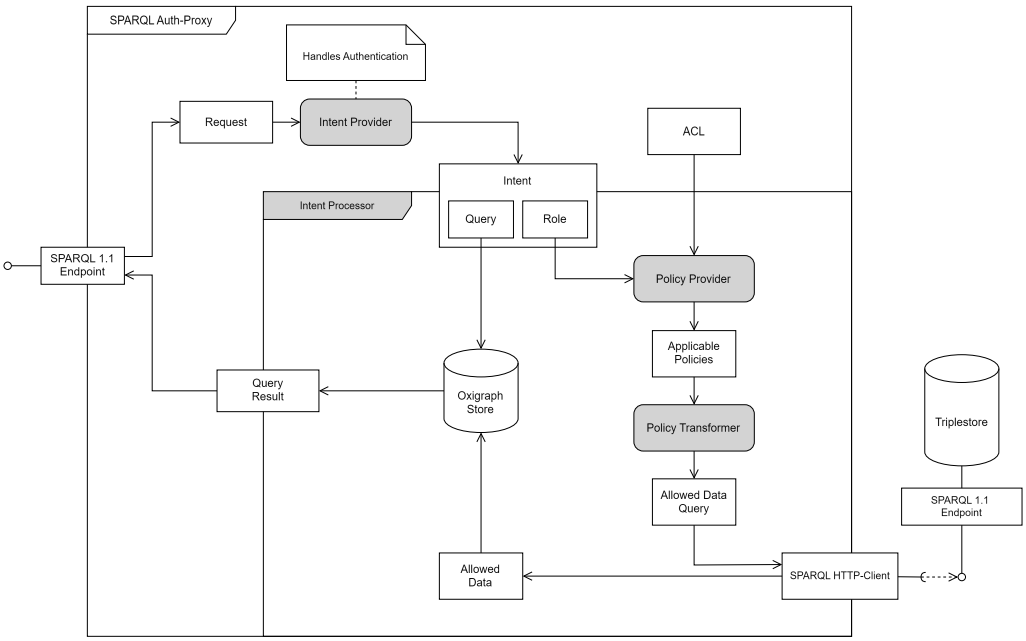

Abbildung 5: Architektur Diagram Proxy

Im Prototypen wird nur ein sehr minimaler Intent Provider umgesetzt, welcher auf einfache Rollen konfiguriert werden kann und auf die Authentifizierung, resp. die Anbindung an IAM-Systeme verzichtet (vgl. Abgrenzung).

Intent Processor

Der Intent Processor ist für die Ausführung des Intents zuständig. Tritt ein neuer Intent auf, wird durch den Policy Provider eine Menge an Policies ausgewählt, welche sich auf die Intent-Role anwenden lassen. Diese Policies entscheiden, auf welche Triples die Intent-Query schlussendlich zugreifen kann. Dazu werden die Policies durch den Intent Transformer in die Allowed-Data-Query umgesetzt, welche anschliessend auf den Backend-Triplestore ausgeführt wird. Die vom Backend-Triplestore erhaltene Allowed-Data wird vorübergehend im lokalen Triplestore zwischengespeichert. Letztlich wird die Intent-Query auf diesem temporären Datensatz ausgeführt und das Resultat durch den Proxy-Endpunkt zurückgegeben.

Policy Provider

Der Policy Provider bildet die Schnittstelle zwischen der Konfiguration der Zugriffsberechtigungen (s. Konfiguration Zugriffsberechtigungen (ACL), unten) und dem Intent. Er bildet die Menge der gemäss identifizierter Rolle anwendbarer Berechtigungen (Applicable Policies). Diese werden benötigt um die SPARQL-Anfrage an den Backend-Triplestore entsprechend zu erweitern.

Policy Transformer

Der Policy Transformer nutzt die vom Policy Provider identifizierte Menge an Berechtigungen, um die ursprüngliche SPARQL-Abfrage zu erweitern. Die Erweiterung der Abfrage dient dabei dem Filtern der Daten bereits auf dem Backend-Triplestore. Auch wird die Abfrage dabei in eine *SPARQL-Construct Abfrage* umgewandelt, welche es ermöglicht, die empfangenen Daten ohne weitere Transformation in den internen, temporären Triplestore abzufüllen. Somit wird sichergestellt, dass ein gemäss der Berechtigungen geschützter Datensatz erstellt wird.

Es ist zwingend, dass diese Filterung vor dem Ausführen der ursprünglichen Anfrage (Intent-Query) geschieht. Würde die Intent-Query zuerst auf den Backend-Triplestore angewendet werden, könnten die resultierenden Daten nicht mehr auf die ursprünglichen Triples zurückgeführt werden und die Policies liessen sich nicht mehr anwenden.

Konfiguration Zugriffsberechtigungen (ACL)

Der Proxy nutzt ein rollenbasiertes Zugriffskonzept (RBAC), wobei der Zugriff auf Triples durch Rollen verschieden gesteuert werden kann. Die ACL wird mittels einer Menge von Policies (Berechtigungen) beschrieben. Mit einer Policy lassen sich Triple-Muster beschreiben, welche den Zugriff für bestimmte Rollen erlauben oder verweigern.

Das folgende Beispiel zeigt eine Policy, welche der Rolle `user` den Zugriff auf alle Triples mit dem Prädikat `foaf:givenName` verbietet.

```
{
  "role": "user",
  "permission": "deny",
  "subject": "*",
  "predicate": "foaf:givenName",
  "object": "*"
}
```

Einschränkungen

Federated Queries

Der SPARQL-Endpunkt des Proxys ist nicht in der Lage, SPARQL Federated Queries zu verarbeiten. Dies betrifft sowohl den Einsatz als weiterleitenden SPARQL-Endpunkt wie auch als Ziel einer Federated Query:

- *Nutzung als weiterleitender SPARQL-Endpunkt in einer Federated Query:* der für die Implementation gewählte in-memory Triplestore (Oxigraph) unterstützt keine Federated Queries. Somit kann der Proxy keine von der Datennutzer:in empfangenen Federated Queries verarbeiten.
- *Als Ziel (SPARQL "Service") einer Federated Query hinter einem Triplestore mit Unterstützung für Federated Queries:* Soll der Proxy in einer Federated Query als Service verwendet werden, muss die Authentisierung an den Proxy durchgereicht werden können. Hierfür existieren noch keine Standards. [20]

Performance-Verbesserungen Policy Transformer

Es ist anzumerken, dass die Allowed-Data-Query eine möglichst kleine Teilmenge des geschützten Datensatz bilden sollte, um das Übertragen irrelevanter Daten zu verringern und somit die Effizienz des Proxys zu steigern. Dies wurde in der Umsetzung des Prototyps nicht weiter berücksichtigt, da die Testdatensätze eine überschaubare Datenmenge aufweisen. Die Allowed-Data-Query könnte noch um Informationen der Intent-Query angereichert werden, was die Allowed-Data weiter eingrenzen würde, in dem für die Intent-Query irrelevante Triples ebenfalls ausgeschlossen werden.

Ausblick

Im vorliegenden Dokument wurden Möglichkeiten zur Zugriffskontrolle bei der Nutzung von Linked Data evaluiert. Im Rahmen einer Anforderungsanalyse wurde insbesondere die Nutzung von Linked Data als mögliche Technologie zur Umsetzung von dezentralen Personenstammdaten anhand von exemplarischen Use Cases untersucht. Die anschliessende Technologieevaluation hat verschiedene, gängige Triplestores in Bezug auf Möglichkeiten zur Zugriffskontrolle evaluiert und die Unterschiede festgehalten. Basierend auf diesen Erkenntnissen wurden verschiedene Architekturvarianten vorgeschlagen, eine davon wurde im Konzept SPARQL-Proxy weiter ausgearbeitet.

Prototyp SPARQL-Proxy

Um das Konzept praxisnah zu überprüfen, wurde parallel ein Prototyp implementiert. Dieser lässt sich vor einen beliebigen, SPAQRL 1.1-konformen Triplestore vorschalten und ermöglicht es, Zugriffskontrollkonzepte auf Triple-Ebene zu testen. Die Umsetzung wurde hier bewusst minimal gehalten und verzichtet beispielsweise wie beschrieben auf die Anbindung realer IAM-Systeme.

Der Proxy ist als Open-Source Software unter der MIT-Lizenz veröffentlicht und kann als Git-Repository auf GitHub heruntergeladen werden:

https://github.com/swiss/ld-prototype-proxy

Offene Punkte / Ausblick

Das vorliegende Konzept, zusammen mit dem prototypisch implementierten Proxy zeigt einen möglichen Lösungsweg zur Zugriffskontrolle in Linked Data auf. Im Laufe der Analyse und Implementation sind weitergehende Forschungsfragen aufgetaucht, welche im Rahmen des Projekts nicht beantwortet werden konnten und weiterer Abklärungen bedürfen:

- Die explizit abgegrenzte Anbindung an IAM-Systeme bringt insbesondere auch beim Einsatz eines Proxies weitere Fragen mit sich: wie wird authentifiziert? Inwiefern werden verschiedene Verfahren von aktuellen SPARQL-Clients unterstützt? Etc.
- Federated Queries sind aktuell mit dem Proxy-Ansatz nicht möglich. Sie bedürfen je nach Variante einer angepassten Implementation oder aber weiterer Standardisierung.
- Die Einschränkung des Zugriffs auf Linked Data und der Einsatz eines Proxies bringen weitere Fragen der Fehlerbehandlung mit sich: Wie wird anfragenden Clients eine fehlende Zugriffsberechtigung mitgeteilt? Hier gibt es verschiedene Möglichkeiten, z.B. mittels fehlender/maskierter Daten in der Antwort, Fehlermeldungen, etc. Aber auch: wie verhält es sich bei Fehlern im Zugriff auf den Backend-Triplestore?

- Auswirkungen auf die Performance, sowie Skalierbarkeit wurden explizit nicht untersucht.

Diese offenen Punkte müssen zur weitergehenden Erforschung der Zugriffskontrolle in Linked Data geklärt werden.

Literaturverzeichnis

[1] LINDAS, «Waldbrandgefahr». Verfügbar unter: https://environment.ld.admin.ch/foen/gefahren-waldbrand-warnung/1)

[2] LINDAS, «Strompreise». Verfügbar unter: https://energy.ld.admin.ch/elcom/electricityprice-canton

[3] LINDAS, «Marktzahlen». Verfügbar unter: https://agriculture.ld.admin.ch/foag/cube/MilkDairyProducts/Consumption_Price_Month

[4] Bundesamt für Statistik, «Meldeverhaeltnis». Verfügbar unter: https://www.bfs.admin.ch/bfs/de/home/register/personenregister/registerharmonisierung/meldeverhaeltnis.html

[5] Bundesamt für Statistik (BFS), «Personenstammdaten. Personendefinition, Datenverwaltung, Prozesse, Registerinhalte und Definition der Referenzdaten». Verfügbar unter: https://www.bfs.admin.ch/asset/de/22686132

[6] Kanton Zürich, «KEP | Einwohnerwesen | Kanton Zürich». Verfügbar unter: https://www.zh.ch/de/politik-staat/gemeinden/einwohnerwesen.html#-538763814

[7] Kanton Bern, «Gemeinderegister | Gemeinderegistersysteme-Plattform (GERES-Plattform)». Verfügbar unter: https://www.kaio.fin.be.ch/de/start/themen/rechtliche-grundlagen/gemeinderegister--geres-.html

[8] Zentrale Ausgleichsstelle ZAS, «Zentrale Ausgleichsstelle ZAS | Systematische Verwendung der AHV-Nummer (SBN)». Verfügbar unter: https://www.zas.admin.ch/zas/de/home/partenaires-et-institutions-/utilisation-systematique-du-navs13.html

[9] Zentrale Ausgleichsstelle ZAS, «Zentrale Ausgleichsstelle ZAS | Schnittstelle UPI-Viewer». Verfügbar unter: https://www.zas.admin.ch/zas/de/home/partenaires-et-institutions-/unique-person-identification--upi-/upiviewer.html

[10] Schweizerische Eidgenossenschaft, «Bundesgesetz über Radio und Fernsehen(RTVG) (SR784.40)». Verfügbar unter: https://www.fedlex.admin.ch/eli/cc/2007/150/de

[11] «FireGraph – powering Smart Data for Smarter Firefighters». Verfügbar unter: https://www.firegraph.info/

[12] W3C, «SPARQL 1.1 Query Language». Verfügbar unter: https://www.w3.org/TR/sparql11-query/

[13] R. Stojanov und M. Jovanovik, «Authorization Proxy for SPARQL Endpoints», in *ICT Innovations 2017*, D. Trajanov und V. Bakeva, Hrsg., Cham: Springer International Publishing, 2017, S. 205–218.

[14] R. Stojanov, S. Gramatikov, I. Mishkovski, und D. Trajanov, «Linked Data Authorization Platform», *IEEE Access*, Bd. 6, S. 1189–1213, 2018, doi: 10.1109/ACCESS.2017.2778029.

[15] W3C, «R2RML: RDB to RDF Mapping Language». Verfügbar unter: https://www.w3.org/TR/r2rml/

[16] Free University of Bozen-Bolzano, «Ontop - A Virtual Knowledge Graph System». Verfügbar unter: https://ontop-vkg.org/

[17] SPHN, «SPHN Connector». Verfügbar unter: https://git.dcc.sib.swiss/sphn-semantic-framework/sphn-connector

[18] Schweizerische Eidgenossenschaft, «data.fedlex.ch». Verfügbar unter: https://fedlex.data.admin.ch/

[19] J. D. Fernández, S. Kirrane, A. Polleres, und S. Steyskal, «Self-Enforcing Access Control for Encrypted RDF», in *The Semantic Web*, E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler, und O. Hartig, Hrsg., Cham: Springer International Publishing, 2017, S. 607–622.

[20] «GitHub.com, w3c/sparql, issue 117: add Authentication to Federation». Verfügbar unter: https://github.com/w3c/sparql-dev/issues/117