Universität Leipzig
Fakultät für Mathematik und Informatik
Wintersemester 2021/22
Bachelorarbeit

Transformation von Daten des Amtlichen Liegenschaftskatasterinformationssystem in semantische Datenformate

---

Betreuer: Prof. Dr. Hans Gert Gräbe, Dr. Christian Zinke-Wehlmann, Norman Radke

Abgabedatum: .03.2022

Robin Seidel

Inhaltsverzeichnis

1 Einleitung

1.1 Zusammenfassung

Diese Bachelorarbeit befasst sich mit der Transformation von Daten aus dem Amtlichen Liegenschaftskataster Informationssystems kurz ALKIS in ein semantisches Format. Im Vordergrund dieser Arbeit stehen dabei die Fragen inwiefern ALKIS Daten in semantischen Formaten nützlicher als die vorhandenen Daten im XML Format sind und ob eine vollständige Automatisierung der Transformation der Daten überhaupt möglich ist. Zur Umsetzung dieser Fragen wurde eine Software implementiert, die die Funktionen Download der Daten, Transformation der Daten, Speicherung und Bereitstellung umfasst.

1.2 Motivation

Mit der Einführung des World Wide Webs im Jahre 1990 durch Tim Burner Lee ist das Internet heute viel mehr als das, was es früher einmal war. Der Mensch nutzt das Internet immer häufiger als Hauptinformations- und Interaktionsmöglichkeit über diverse Unterhaltungskanäle wie Facebook, Youtube oder andere Websites. (QUELLE) Während am Anfang lediglich ein paar Dutzend Webseiten existierten sind es mittlerweile über zwei Milliarden mit 100 Milliarden Webdokumenten, die sich wiederum alle 6 Monate verdoppeln. (vgl. Meinel 2021) Aber es entstehen nicht immer nur mehr Webseiten auch die Anzahl der Internetnutzer hat sich in den letzten Jahren drastisch gesteigert. 1995 hat der Spiegel 250.000 Nutzer in Deutschland geschätzt, während 2018 durch das statistische Bundesamt ein Anstieg auf 90wurde. (vgl. Köllinger 2003 zitiert nach: Der Tagesspiegel vom 19. August 1995, S. 23) Demnach nutzen mittlerweile 66,5 Millionen Menschen ab 10 Jahren das Internet als Informations- und Interaktionsquelle. (vgl. Statistisches Bundesamt 2018) Bei einer solchen Masse an Informationen und Daten wird es immer schwerer relevante Informationen herauszufiltern, obwohl das Hauptziel des Internets einst nicht die Unterhaltung, sondern die Informationsbeschaffung war. (QUELLE) Deshalb ist es wichtig die vorhandenen Daten mit Hilfe von RDF Graphen in einer geordneten Art und Weise zur Verfügung zu stellen. Das semantische Web war bereits eine Idee des Entwicklers/Gründers Tim Burner Lee. Er hatte die Vision, dass das Netz von Informationen nicht nur Dokumente und Daten miteinander verbindet, sondern das diesen Daten auch eine Bedeutung gegeben wird. Damit ermöglicht man einer Maschine die Anhäufung von Daten automatisiert nutzen zu können. - beschreibung semantik web und wichtigste Prinzipien - 4 prinzipien: tim berner lee - nutzen von URI's als namen für objekte - zur eindeutigen identifikation für dinge - nutze HTTP URIs - so können Menschen die Objekte nachschauen - nutze anerkannte standarts(RDF, Sparql) - linke URIS weiter Das semantische Web ist eine Art rohe(also unebarbeitete) Daten zuübertragen. (Interoperabilität) Um die Informationen die das Web liefert in einen strukturierten

Weg wiederzugeben. Mit Hilfe der daraus entstehenden strukturierten Daten kann die Software diese Daten anschließend besser nutzen und verarbeiten. Das semantische Web ist damit sozusagen eine Alternative zu Webseiten die ausschließlich für die menschliche Nutzung geeignet sind. Die Vorteile die eine Semantifizierung der Daten mit sich bringt sind unter anderem das Abfragen durch Sparql Querys, eine Verbesserung der Bedeutung der Daten, das Verbinden von Daten und die Nutzbarkeit für andere Bereiche. - überleitung auf Alkis Thematik Alkis Daten(siehe Begriffsklärungen) werden zwar in gewissen Standart und Einheitlichkeit erhoben, allerdings ist die Verarbeitung dieser Daten hauptsächlich für GeoProgramme(näher erklären) vorgesehen. Die Art der Daten ist nicht für die optimale Verarbeitung von ComputerProgrammen und zur Nutzung für den gewöhnliochen Nutzer vorgesehen. Eine Überführung in Rdf Graphen ermöglicht es das Potenzial der Daten freizuschalten und die Nutzung für zahlreiche andere Bereiche zur Verfügung zustellen. Quellen: (https://www.spektrum.de/kolumne/semantic-web-wie-das-internet-inhalte-verstehen-koennte/ 1841389) paper über tim terner lees arbeit https://www.researchgate.net/publication/307845029_Tim_Berners-Lees_Semantic_Web paper von burner über das semantic web 10.3.2022 https://www.w3.org/2000/Talks/0906-xmlweb-tbl/text.htm statistik bundeasamt https://www.destatis.de/DE/Presse/Pressemitteilungen/2018/09/PD18_330_634.html Köllinger, Philipphttps://www.econstor.eu/handle/10419/151237 (Verlag und Jahr noch angeben

2 Grundlagen

Hier werden die wichtigsten Konzepte erklärt, die in dieser Arbeit Anwendung gefunden haben.

2.1 ALKIS

Das Amtliche Liegenschaftskatasterinformationssystem, kurz ALKIS, ist ein Teil des AAA-Modells (AFIS-ALKIS-ATKIS Modell). AFIS (Amtliches Festpunktinformationssystem) ist dabei ein Raumbezugssystem, indem Informationen zu Festpunkten modelliert werden. In ATKIS (Amtliche Topographisch-Kartographische Informationssystem) geht es um digitale Karten zu Landschaften und Topografie. (seite 8 geoinfodoc). ALKIS umfasst die für uns relevanten Informationen aus dem Liegenschaftsbuch und -karte sowie Punktnachweisen. Die Verwaltung des AAA- Modells erfolgt durch die ADV (Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland). Das Modell ist seit 2015 in allen Bundesländern aktiv. Ziel der Entwicklung eines solchen Modells war eine einheitliche und fachübergreifende Nutzung von Geodaten zu ermöglichen.(seite 2 geoinfodoc). Bei der Erschaffung des Datenmodells wurden Standards und Normen verwendet die vom W3C(World Wide Web Consortium), OGC(opengeospatial consortium) und von der ISO(internationen Organization for Standardization) anerkannt sind. WELCHE STANDARDS UND NORMEN Durch die Einführung von ALKIS wurden alle Daten des Liegenschaftskatasters redundanzfrei zusammengeführt. (quelle s 10) Katatster- und Liegenschaftsinformationen sind Geobasisdaten über sogenannte Liegenschaften, also zum Beispiel Flurstücke oder Gebäude. Diese Geobasisdaten werden durch die Vermessungs- und Katasterverwaltungen der Länder erhoben. Die Nachteile, die die Führung eines Liegenschaftsbuches mit sich bringen, sind deutlich: teilweise technisch veraltete Konzepte, verschiedene Datenformate, getrennte und teilweise redundante Datenhaltung und länderspezifische Unterschiede treffen hier aufeinander. (Quelle s8 13.3) Diese Nachteile konnten durch die Einführung von ALKIS gelöst werden.

https://www.adv-online.de/AdV-Produkte/Liegenschaftskataster/ALKIS/broker.jsp?uMen=e5670f15-8e71-3c01-e1f3-351ec0023010 https://www.adv-online.de/AdV-Produkte/Liegenschaftskataster/ALKIS/
11.3.22 ADV Online - ALkis Beschreibung

2.2 ALKIS Daten

Die Veröffentlichung der ALKIS Daten erfolgt über Schnittstelle,n die von den jeweiligen Bundesländern angeboten werden. Einige Bundesländer haben jedoch gewisse Zugriffsbeschränkungen. 7 Bundesländer haben zumindestens einige Geodatensätze frei zugänglich gemacht. Sie unterliegen der Open Data Lizenz: "Datenlizenz Deutschland - Namensnennung- Version2.0"https://www.govdata.de/dl-de/by-2-0 Andere WFS Schnittstellen sind nur durch Anmeldung und oder mit Kosten verbunden. (siehe Tabelle)(https://learn.opengeoedu.de/opendata/vorlesung/offene-geodaten/amtliche-geodaten

Die Bestandsdaten aus ALKIS können über eine sogenannte normbasierte Austauschschnittstelle(NAS) abgerufen werden. Diese wurde speziell für den Austausch von Daten des AAA - Modells vom ADV entwickelt. Als Standards werden unter anderem XML, WFS (Web Feature Server) oder auch WMS (Web Map Server) genutzt. XML wird dabei als Datenaustauschformat verwendet, während WFS und WMS zum Abrufen der ALKIS-Objekte dienen. Diese vom OGC (Open Geospatial Consortium Inc) definierten Geodatendienste sind ein Teil der NAS Spezifikation. Geodatendienste sind standardisierte Services die über eine URL aufgerufen und zum Austausch von Daten genutzt werden. https://geodatenonline.bayern.de/geodatenonline/seiten/wms_webdienste3.2022 Die Hauptaufgabe eines WMS ist die Visualisierung der Geodaten, er ist ein Darstellungsdienst. Das Ergebnis (Response) einer Anfrage (Request) auf einen WMS ist ein Kartenausschnitt. Der WFS ist ein Download Service, dem Rohdaten in Form der vom ADV definierten Objekte zugrunde liegen. Beim WFS bleibt die volle Struktur der Daten erhalten und sie werden im vollem Ausmaß bereit gestellt und sind somit optimal zur Weiterverarbeitung geeignet. Um an die Daten zukommen die für das Projekt benötigt werden, wird der WFS genutzt. Diese Schnittstelle besitzt drei verschiedene Abfrageoptionen. Die Basis URL ist für das Beispiel NRW: https://www.wfs.nrw.de/geobasis/wfs_nw_alkis_vereinfacht?. An diese URL können nun verschiedene Parameter angehangen werden, um die richtige Antwort (Response) des Servers zu erhalten. Der WFS hat verschiedene Abfrageoptionen. Mit Request=GetCapabilities"werden alle Fähigkeiten und Metadaten des aufgerufenen Dienstes abgerufen. Durch Request=DescribeFeatureTy die Struktur eines FeatureTypes abgefragt. Die wichtigste Abfrageoption ist Request=GetFeature". Eine komplette Anfrage (Request) an den WFS könnte so aussehen: https://www.wfs.nrw.de/geobasis/wfs_nw_alkis_vereinfachtService=WFS&REQUEST=GetFeature&VERSION=2.0.0&Typenames=Flurstueck&Countasdfasdfasfdasdf=10. Die Anfrage ist ein HTTP Get Call und beinhaltet neben der Basis URl verschiedene Parameter die erforderlich (Beispiel: Version, Typenames) oder optional (Beispiel:Count, Resulttype) sein können. Da der WFS ein Downloaddienst ist, erhält man als Response ein XML Dokument, das die abgefragten ALKIS Objekte enthält. Eine Übersicht über

alle möglichen Objekte findet man im Objektartenkatalog Seite 22. Das für uns wichtigste Objekt ist das AX_Flurstueck. Laut Definition(Objektartenkatalog seite33) ist ein Flurstück ein Teil der Erdoberfläche, das mit einer Grenzlinie umschlossen ist. Das Objekt AX_Flurstueck besitzt die Attribute zustaendigeStelle, gemarkung, flurstuecksnummer, flurstueckskennzeichen, amtlicheFlaeche, flurnummer, und die Relationen istGebucht, zeigtAuf, weistAuf, gehoertAnteiligZu, beziegtSichAufFlurstueck etc.. Die Veröffentlichung der ALKIS Daten erfolgt über Schnittstellen die von den jeweiligen Bundesländern angeboten werden. Einige Bundesländer haben jedoch gewisse Zugriffsbeschränkungen. 7 Bundesländer haben zu mindestens einige Geodatensätze frei zugänglich gemacht. Sie unterliegen der Open Data Lizenz: "Datenlizenz Deutschland - Namensnennung-Version2.0"https://www.govdata.de/dl-de/by-2-0 Andere WFS Schnittstellen sind nur durch Anmeldung und oder mit Kosten verbunden. (siehe Tabelle)(https://learn.opengeoedu.de/opendata/vorles
geodaten/

https://www.adv-online.de/icc/extdeu/binarywriterservlet?imgUid=8f830072-8de8-9221-d5ad-8f138a438ad1&uBasVariant=11111111-1111
geoInfoDoc

https://www.lgln.niedersachsen.de/download/126790/Basiswissen_ALKIS_ETRS89_Schulungsmaterial_Stand_12.04.2010.pdf bAsiswissen
alkis 10.3

https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/webdienste/anleitung_wfs.pdf

Bundesland	verfügbare Datensätze	Zugriffsbeschr.
NRW	AAA, NAS, vereinfacht	Quellennachweiß: Geobasis NRW, dl-de/by-2-0, (Daten geändert)
Berlin	AAA(funktioniert nicht, featuretype nicht gefunden)	Quellennachweiß: Geoportal Berlin, dl-de/by-2-0, (Daten geändert)
Thüringen	keine Angabe	Authentifizierung und Quellennachweiß: GDI-TH, dl-de/by-2-0, (Daten geändert)
Brandenburg	AAA, NAS, vereinfacht	Quellennachweiß: GeoBasis-DE/LGB, dl-de/by-2-0, (Daten geändert)
Mecklenburg-Vorpommern	vereinfacht	Kostenpflichtig, Quellennachweiß: GeoBasis-DE/M-V 2022 (Daten geändert)
Hamburg	vereinfacht	keine
Sachsen	vereinfacht	Quellennachweißes: GeoSN, dl-de/by-2-0, (Daten geändert)
Sachsen-Anhalt	NAS-konform, vereinfacht	kostenpflichtig
Bayern	vereinfacht	Authentifizierung /Freischaltung notwendig
Niedersachsen	NAS-konform, AAA-basiert, vereinfacht	kostenpflichtig
Hessen	vereinfacht, NAS-konform	keine
Rheinland-Pfalz	vereinfacht	Freischaltung notwendig, kostenpflichtig
Schleswig-Holtstein	keine Angabe	Freischaltung notwendig, kostenpflichtig
Baden-Württemberg	keine Angabe	auth notwendig, Quellennachweiß: LGL, www.lgl-bw.de
Bremen	keine Angabe	Authentifizierung /Freischaltung notwendig
Saarland	keine Angabe	Freischaltung notwendig, kostenpflichtig

*Die Endpoints für die einzelnen Bundesländer sind im Anhang zufinden.

2.3 RDF

Das Resource Description Framework kurz RDF, ist ein vom W3C konzipiertes Datenformat zum Beschreiben von Objekten im Internet. https://www.w3.org/TR/rdf-concepts
RDF ist ein universelles, maschinenlesbares Format, das für den Austausch und der Darstellung von jeglicher Art von Daten gedacht ist. Ursprünglich wurde es genutzt um Metadaten zu beschreiben.(Beleg) Später wurde es auch in vielen verschiedenen Gebieten (WELCHE?) und generellen Anwendungen benutzt. Heute gilt es als grundlegender Baustein des Semantischen Webs. Eine Aussage in RDF nennt man Triple. Ein Tripel besteht aus den Teilen Subjekt, Prädikat und Objekt. Ein Subjekt kann man sich als den Entity Identifier der zubeschreibenden Aussage vorstelllen. Das Prädikat ist in diesem Kontext der Attribut Name und das Subjekt der Attribut Wert. Das Subjekt und das Prädikat werden als IRI dargestellt, damit die beschriebene Sache eindeutig identifiziert wird.

```
<https://dbpedia.org/page/Leipzig> <https://dbpedia.org/ontology/country> <http://dbpedia.org/resource/
```

In diesem Beispiel identifiziert die IRI für country"das Prädikat eindeutig. Die IRI zeigt das es sich nicht um irgendein beliebiges 'country' handelt sondern, das es Bestandteil der von dppedia veröffentlichten Ontology Country ist. IRI's sehen und haben ähnlichen Nutzen wie die allgemein bekannten URI'S, jedoch sind sie keine Adressen oder Locators sondern jediglich Identifiers. Objekte können auch IRI's seien. Somit kann die gleiche Ressource Objekt eines Triples und gleichzeitg als Subjekt eines weiteren Triples dienen. Im oben gezeigten Beispiel ist "Germany"das Objekt und in dem Beispiel:

```
<https://dbpedia.org/page/Germany> <https://dbpedia.org/ontology/dbo:PopulatedPlace/area> 357022.0 .
```

ist Germany das Subjekt. Somit werden die beschriebenen Aussagen verbunden und zu Netzwerken oder auch Graphen genannt verbunden. Um die IRI's kürzer und leichter lesbar zumachen, werden Prefixes angewendet, diese werden am Anfang eines Dokumentes definiert. Ein Prefix beschreibt die ganze IRI, bis auf den letzten Teil. Somit ergibt sich das RDF Dokument in der Turtle Serialisierung:

```
@prefix dbo: https://dbpedia.org/ontology/
@prefix dbr: https://dbpedia.org/page/
 <dbr:Leipzig> <dbo:country> <dbr:Germany> .
 <dbr:Germany> <dbo:PopulatedPlace/area> "357022.0" .
```

Die Turtle Serialisierung zeichnet sich dadurch aus, dass wenig Speicherplatz benötigt wird und sie für den Menschen intuitiv lesbar ist. URIS werden in eckigen Klammer geschrieben und Literale in Anführungszeichen. Die Triples sind durch einen Punkt getrennt.

2.4 RML & Yarrrml

Die RDF Mapping Language (RML) https://rml.io/specs/rml/ ist eine Sprache mit der man Regeln aufstellt, um Datenstrukturen die in verschiedenen Formaten vorliegen in RDF umzuwandeln. RML beruht und erweitert die vom W3C definierte Sprache R2RML https://www.w3.org/TR/r2rml/ die dafür genutzt wird ein spezielles Mapping anzufertigen, um Daten aus einer relationalen Datenbank in ein RDF Datenset umzuwandeln. Sowohl R2RML Mappings als auch RML Mappings sind RDF Graphen und werden in Turtle geschrieben. Mit RML ist es möglich Daten aus heterogenen Datenquellen auf das RDF Format zu mappen. RML ist kein offiziell bestätigter Standard, es erweitert nur R2RML. Das bedeutet, dass RML nicht nur auf relationale Datenbanken, sondern auch auf weitere Datenstrukturen angewendet werden kann. Da in dieser Arbeit Daten in das XML Format umgewandelt werden, ist die Verwendung von RML Regeln zu bevorzugen. (REGELN?) Yarrrml https://rml.io/yarrrml/spec/ ist wiederum eine Sprache, die auf Yaml https://yaml.org/spec/1.2.2/ beruht. Sie ermöglicht es auf einfache Art und Weise deklarative Regeln für das Mapping von verschiedenen Datenquellen auf RDF anzuwenden. Yarrrml hat den gleichen Zweck wie RML, mit dem Vorteil, dass es für den Nutzer einfacher ist die Mapping Regeln zu erstellen, wird somit die Effektivität der RDF Graphen Erstellung erhöht. (DEN ABSATZ NOCH MAL)

2.5 Tripelstores

Erklärung von Triplestores eventuell noch hier

2.6 SPARQL

SPARQL steht für Sparql Protocol and RDF Query Language und ist ein vom W3C festgelegter Standart zum Abfragen von RDF Daten. Wie in RDF Turtle können auch in SPARQL Prefixes definiert werden sodass, man nicht die ganze URI ausschreiben muss. Eine SPARQL Query besteht aus verschiedenen Klauseln. In der WHERE Klausel wird beschrieben, welche Triples vom angefragten Datensatz angesprochen werden. Die Where Klausel besteht aus einer oder mehreren Aussagen. Diese sind wie RDF Triple, nur das die drei Bestandteile durch Variablen ersetzt werden können. Variablen beginnen mit einem '?'. In der SELECT Klausel wird angegeben welche Variablen dem Nutzer angezeigt werden. Sparql Beispiel um die Fläche von Deutschland zu erfragen.

```
PREFIX dbr: <https://dbpedia.org/page/>
PREFIX dbo: <https://dbpedia.org/ontology/>

SELECT ?flaeche
WHERE
{
```

```
<dbr:Germany> <dbo:PopulatedPlace/area> ?flaeche .
}
```

Die Ausgabe der Anfrage ist der der ?flache Variable zugewiesener Wert.

2.7 RDF vs XML

Bei der Recherche dieser Arbeit ist dabei eine zentrale Frage entstanden. Warum sollte man überhaupt RDF nutzen, wenn dies ohnehin auf XML beruht und die ALKIS Daten bereits in diesem universellem Format vorliegen? Was ist der Vorteil von RDF gegenüber anderen Datenaustauschformaten? Für die Beantwortung der Frage wird auf das oben genannte Beispiel (SEITE) zurückgegriffen.
Eine Aussage kann in XML durch verschiedene Arten ausgedrückt werden. Für den Nutzer der diese Aussagen liest bedeuten Sie das Gleiche. Für eine Maschine bzw. einen Computer sind es verschiedene XML Bäume. In RDF wird die Aussage der Äutor der Bachelorarbeit ist RobinSSeidel als Triple dargestellt. triple(Autor, Bachelorarbeit, RobinSeidel) in XML kann diese Aussage auf verschiedene Art und Weise dargestellt werden (SATZBAU?)

```
<autor>
<von>bachelorarbeit</von>
<name>Robin Seidel </name>
</autor>
oder auch:
<bachelorarbeit>
<autor>Robin Seidel </autor>
</bachelorarbeit>
```

Für den Nutzer der diese XML Dokumente liest, haben diese die gleiche Bedeutung. Für eine Maschine bzw. einen Computer sind es jedoch zwei verschiedene XML Bäume. Dies zeigt das die Nutzung von Technologien des Semantischen Webs sehr wertvoll sein können. https://www.w3.org/DesignIssues/RDF-XML.html https://dbs.uni-leipzig.de/html/seminararbeiten/semSS99/arbeit5/Rdf.html

3 State of the Art

3.1 Problembeschreibung

Das Ziel dieser Arbeit ist die Umwandlung von ALKIS Daten in ein semantisches Format. Um dieses Ziel zu erreichen findet der ETL-Prozess Anwendung. Der ETL-Prozess(Extract, Transform, Load) ist ein typisches Vorgehen in der Welt von Big-Data. Das Ziel des Prozesses sind Daten aus unterschiedlichen Datenquellen für die weitere Verarbeitung vorzubereiten und anschließend wieder bereitzustellen. Der Extract(auf deutsch extrahieren) Teil des Prozesses ist es die notwendigen ALKIS Daten zu sammeln. Dies gelingt durch HTTP-Requests an die Schnittstellen die von den Behörden der Bundesländer bereit gestellt werden. (siehe Tabelle 2.2) Durch die gezielte Angabe der richtigen Parameter erhält man so die notwendigen ALKIS Rohdaten. Der Teilprozess der Transformation enthält wiederum mehrere Einzelschritte. Ziel der Transformation ist es die Daten in ein semantisches Format so umzuwandeln, dass sie verknüpft sind und durch eine SPARQL Schnittstelle abgerufen werden können. Die ALKIS Daten müssen nun mithilfe von Mapping Regeln so konstruiert werden, dass sie am Ende ein möglichst nützlichen und strukturierten Graphen ergeben. Im Load Teil des Prozesses werden die Daten dann in einen Triple Store geladen. Dieser dient als Datenbank für Graphen. Die transformierten RDF Daten können nun über SPARQL Abfragen abgerufen werden.

https://www.bigdata-insider.de/was-ist-etl-extract-transform-load-a
21.3

3.2 Möglichkeiten der Transformation

Im ersten Schritt der Arbeit war es nötig, die möglichen Herangehensweisen an das Projekt und die damit einhergehenden Probleme zu identifizieren. ALKIS Daten werden in vielen verschiedenen Formaten angeboten. Es gibt zum Beispiel Datensätze über Gebäude, Bodenschätzungen und über Flurstücke. Die Art und Weise wie die Daten aufgebaut sind und abgerufen werden ist zwar durch den ADV geregelt, jedoch kommt es in den verschiedenen Bundesländern hinsichtlich der Erreichbarkeit dieser Datensätze noch zu großen Unterschieden. So gibt es beispielsweise Datensätze ohne Zugriffsbeschränkung und Datensätze auf die nur durch Bezahlung zugegriffen werden kann. (näheres siehe 2.2 ALKIS DATEN) Da es das Ziel dieser Arbeit ist, einen Wissensgraphen über ALKIS Daten zu erstellen, wurde der Fokus auf den Datensatz Flurstuecke"gesetzt. Dieser ist der Datensatz mit den wichtigsten Informationen, er enthält alle wichtigen Daten über die Liegenschaften und Flurstücke der jeweiligen Bundesländer. Eine Möglichkeit um dieses Problem zu bewältigen, wäre die manuelle Erstellung der RDF Dateien. Aufgrund des Ausmaßes der Daten ist diese Möglichkeit allerdings als unrealistisch zu betrachteten. Eine weitere Herangehensweise wäre die Entwicklung eines SELBSTENTWICKELTEN

Parser, [es einen selbstentwickleten Parser zu entwickeln,] der es ermöglicht aus den vorhandenen ALKIS XML Dateien, RDF Dateien zu erzeugen. Vorteile dieser Methode sind, dass der entwickelte Parser sehr gut an die Gegebenheiten der ALKIS Daten angepasst werden könnte und man damit die Transformation nach eigenen Wünschen steuern könnte. Allerdings wäre die Entwicklung eines solchen Programmes zu komplex und kompliziert, dass damit ein erheblicher Mehraufwand für das Projekt einhergehen würde. Eine weitere Herangehensweise, die letztlich auch in dieser Arbeit Anwendung gefunden hat, ist die Verwendung einer bereits existierenden Software, den rmlmapper(link hier). Auf der einen Seite bietet es den Vorteil, dass man davon ausgehen kann, dass es für die erwartenden Anwendungsfälle zuverlässig funktioniert. Auf der anderen Seite muss man aber mit den vorhandenen Gegebenheiten arbeiten und sich intensiv damit auseinandersetzen, wie die eventuell verwendete Software funktioniert. Wenn man nun davon ausgeht das zur Erstellung des RDF Graphens der rmlmapper verwendet wird, muss man sich die Frage stellen auf welche Art und Weise die RML Regeln entstehen, da der rmlmapper diese RML Regeln benötigt um aus verschiedenen Daten verknüpfte Daten zu erstellen. Dafür gibt es zwei Möglichkeiten. Entweder die Erstellung von RML Regeln mithilfe der der vom w3c spezifizierten Sprache R2RML https://www.w3.org/TR/r2rml/ oder mit Yarrrml. Wie in Punkt 2.4 beschrieben, ist es mit Yarrrml möglich auf einfachere Art und Weise deklarative Mapping Regeln zu erstellen. Diese müssen dann nur noch in R2RML Regeln umgewandelt werden. Vorteil daran ist das Yarrrml Regeln leichter zu erstellen sind als R2RML Regeln. Nachteil ist das Yarrrml nur in einer inoffiziellen Version vorliegt und die Dokumentation bezüglich der Anwendung eher mangelhaft ist. Bei der Abspeicherung der fertig umgewandelten RDF Graphen gibt es ebenfalls mehrere Möglichkeiten. Zur Speicherung von Graphen werden keine herkömmlichen relationalen Datenbanken genutzt, sondern speziell dafür konzipierte Graphen Datenbanken. Dadurch das Beziehungen/Relationen der Graphen in so einer Datenbank abgespeichert sind müssen diese bei einer Abfrage nicht erst berechnet werden sondern stehen bereits zur Verfügung. Dadurch ergibt sich eine bessere Performance gegenüber relationalen Datenbanken. Da das Angebot von Graphdatenbanken relativ groß ist, muss ein für das Projekt passendes System ausgewählt werden. In Frage kommen hierfür die Open Source Graphdatenbank Neo4j oder das Open-Source-Framework Apache Jena Fuseki. Bei ersterem werden Daten anstatt auf die herkömmliche Art und Weise nicht in Tabellen, sondern in Graphen strukturiert abgespeichert. Die Implementierung in das Projekt ist einfach und die Speicherung der ALKIS Daten in Neo4j funktioniert ohne Probleme schnell und zuverlässig. Jedoch ist es in Neo4j vorgesehen als Abfragesprache Cypher zuverwenden, anstatt der für Graphen in der Branche meist angewandten Sprache SPARQL. Apache Jena Fuseki ist speziell für den Austausch von semantischen Daten entwickelt. Hiermit ist auf simple Art und Weise möglich einen Fuseki SPARQL Server https://jena.apache.org/documentation/fuseki2/ aufzusetzen

und diesen über eine Programmierschnittstelle abzufragen.

4 Anforderungsanalyse

Zur Umsetzung des Projektes wird ein Programm zur automatisierten Transformation von ALKIS Daten entworfen und implementiert. Die Anforderungen die so eine Software haben sollte werden hier zur besseren Übersicht festgehalten. Das Programm soll dem Nutzer als Tool/Werkzeug dienen, Daten aus ALKIS, speziell den Datensätzen in dem Flurstücke beschrieben werden, in RDF umzuwandeln. Deshalb ist es hilfreich wenn der Nutzer des Programms mithilfe von User Input bestimmen kann, was er tun möchte. Das Programm ist für fünf Anwendungsfälle konfiguriert.

4.1 Nicht-funktionale Anforderungen

4.1.1 Fehlerunanfälligkeit

- Alle Eingaben des Benutzers müssen validiert werden, bei fehlgeschlagener Validierung wird die Eingabe abgelehnt.
- Wenn es zu unerwarteten internen Fehlern im Programm kommt, ist ein Datenverlust zuvermeiden.
- Ein unerwartetes Abstürzen des Programms sollte vermieden werden.

4.1.2 User Experience

- Bei einer Eingabe durch den Nutzer, die das Programm nicht verwerten kann, wird dem Nutzer eine erneute Eingabe ermöglicht
- Bei richtigen oder falschen User Input sollte das Programm dem Nutzer eine angemessene Antwort über Erfolg oder Misserfolg wiedergeben.
- Der Ablauf des Programmes und dessen Anweisungen an den Nutzer sollten so einfach wie möglich gehalten werden, damit die Steuerung des Programmverlaufs durch den Nutzer möglichst intuitiv erfolgen kann.
- Das Programm sollte von so vielen Betriebssystemen wie möglich unterstützt werden.
- Das Programm sollte mit den aktuellen Versionen des rmlmappers und des yarrrml-parsers kompatibel sein.
- Der Nutzer hat die Möglichkeit speziell für das Programm benötigte Daten in einer Konfigurationsdatei zu hinterlegen.

4.2 Funktionale Anforderungen

4.2.1 Anforderungen an die automatisierte Transformation

- Nach Auswahl der Funktion muss der Nutzer keine weitere Eingabe durchführen.
- Der Nutzer hat die Option ein Bundesland auszuwählen. Bei Nichtangabe werden alle Bundsländer abgefragt.
- Es sollen alle verfügbaren ALKIS Schnittstellen abgefragt, und die verfügbaren Datensätze werden heruntergeladen.
- Alle vorhandenen Datensätze sollen transformiert werden.
- Alle erfolgreich umgewandelten Datensätze werden abgespeichert.
- Dem Nutzer werden alle erstellten Triples angezeigt.

4.2.2 Anforderungen an Download Funktion

- Der Nutzer sollte die Möglichkeit haben aus einer vorgegebenen Liste, den gewünschten Datensatz auszuwählen.
- Die Datensätze in der Liste sollen verfügbar sein
- Der Nutzer hat die Möglichkeit mit der Angabe der Gemarkung oder Gemarkungsnummer den Abruf der Daten weiter zuspezifizieren.

4.2.3 Anforderungen an die Transformation

- Der Nutzer kann selbst angeben welcher Datensatz transformiert werden soll.

4.2.4 Anforderungen an die Speicherung der transformierten Daten

- Das Programm sollte eine Überführung der Daten in eine Apache Jena Fuseki Graph Datenbank ermöglichen.
- Der Nutzer kann selbst einen Fuseki Server Endpunkt angaben, an den die Daten übertragen werden sollen.
- Der Nutzer sollte einen Pfad angeben können, um die Datei zu identifizieren, die abgespeichert werden soll.

4.2.5 Anforderungen an das Abrufen der Daten

- Der Anwender kann aus einer Reihe von default Querys auswählen.
- Dem Anwender soll ermöglicht werden, die Daten in Form von Sparql Querys abzufragen.
- Das Ergebnis der Anfrage wird dem Nutzer in Form von Triples im Turtle Format angezeigt.

5 Implementierung

5.1 Beschreibung der Implementierung

Zur Umsetzung der beschriebenen Software Anforderungen, wurde eine Konsolenanwendung prototypisch programmiert. Prototyping ist eine Methodik in der Softwareentwicklung, die zum Ziel hat schnell ein gewisses Ziel zu erreichen. Dies dient zur Bestimmung der Anforderungen und es wird dadurch absehbar ob alle geplanten Funktionen der Software umsetzbar sind.(Auf was wurde verzichtet/vernachlässigt?) Das Kernproblem dieses Projektes ist es einen ALKIS Datensatz zu einen sinnvollen RDF Graphen umzuwandeln. https://de.ryte.com/wiki/Prototyping Zur Verdeutlichung das das Kernproblem gelöst wurde, wurde im Programm ein Showcase implementiert. Nach minimaler Nutzerinteraktion soll hier eine automatische Transformation der Daten angestoßen werden. Der Showcase ist wie folgt definiert: Da möglichst viele Daten für die Transformation verwendet werden sollen, wird für diesen Showcase als Datengrundlage, die ALKIS DAten im vereinfachten Schema benutzt. Für die Bundesländer in denen die Daten ohne Zugriffsbeschränkungen verfügbar sind, ist dieser Datensatz für alle Bundesländer abrufbar.(siehe Tabelle 2.2) Zum Zweck der Einfachheit und um Heterogenität der transformierten Daten zuvermeiden wird die Verwendung der anderen zwei Schemata vermieden. Es werden also ALKIS Daten für die Bundesländer Sachsen, Hessen, Brandenburg, Nordrhein-Westfalen und Hamburg im vereinfachten Schema heruntergeladen . Die heruntergeladen Datensätze sind auf je 20000 Flurstücke begrenzt, da diese das maximal Limit einer Anfrage an den WFS ist. Um alle Flurstücksdaten herunterzuladen müsste man mehrere Aufrufe hintereinander ausführen, was aber nicht Sinn und Zweck dieses Showcases sein soll. Anschließend werden diese Daten in RDF anhand der MApping Regeln transformiert und in einer Fuseki Graph Datenbank gespeichert. Am Ende des Prozesses werden bestimmte Tripel mithilfe einer vorgefertigten Sparql Query ausgegeben.

- Download der Datensätze vereinfachtes Schema
- Brandenburg, Hamburg, Hessen, NRW, Sachsen

- 20000 Flurstueck Objekten pro Datensatz
- Transformation der Datensätze von .xml in .ttl
- Speicherung der Transformierten Daten in Graph DB
- Vorgefertigte Sparql Query um Verknüpfung der Datensätze zuzeigen

Bei Programmstart wird der Nutzer aufgefordert auszuwählen welchen Anwendungsfall er nutzen möchte. Diese Auswahl wird im Programm durch ein ïf-Statementërmöglicht. Bei Eingabe der Zahl Eins, wird der zuvor definierte SShowcaseäusgelöst. Zunächst werden die Datensätze für die fünf Bundesländer gedownloaded. Die URL's für die Endpunkte sind in einem Dictionary gespeichert. Hier ein Aussschnitt aus der Datei AlkisDataService.py, wo sich das Dictionary für die WFS Endpunkte befindet. Das Dictionary enthält alle Endpunkte für die ALKIS Datensätze die zum Zeitpunkt der Bearbeitung der Bachelorabeit, ohne Zugriffsbeschränkung erreichbar waren. Um auf weitere Datensätze zuzugreifen müssen diese an dieser Stelle eingetragen werden.

```
WFS_dictionary = {
 "NRW-vereinfacht": "https://www.wfs.nrw.de/geobasis/wfs_nw_alkis_vereinfacht?",
    "NRW-AAA-basiert": "https://www.wfs.nrw.de/geobasis/wfs_nw_alkis_aaa-modell-basiert?",
    "NRW-NAS-konform": "https://www.wfs.nrw.de/geobasis/wfs_nw_alkis_nas-konform?",
    "Brandenburg-vereinfacht": "https://isk.geobasis-bb.de/ows/alkis_vereinf_wfs?",
    "Brandenburg-AAA-basiert": "https://isk.geobasis-bb.de/ows/alkis_sf_wfs?",
    "Brandenburg-NAS-konform": "https://isk.geobasis-bb.de/ows/alkis_nas_wfs?SERVICE=WFS&",
    "Hamburg-vereinfacht": "https://geodienste.hamburg.de/WFS_HH_ALKIS_vereinfacht?",
    "Sachsen-vereinfacht": "https://geodienste.sachsen.de/aaa/public_alkis/vereinf/wfs?",
    "Hessen-vereinfacht": "https://www.gds.hessen.de/wfs2/aaa-suite/cgi-bin/alkis/vereinf/wfs?",
    "Hessen-NAS-konform": "https://www.gds.hessen.de/wfs2/aaa-suite/cgi-bin/alkis/nas/wfs?"
}
```

Ein Dictionary ist eine Ansammlung von Keys und passenden Values. Nun wird für jeden der fünf Bundesländer eine RequestUrl konstruiert. Die Request URL für Sachsen lautet beispielsweise:

```
https://geodienste.sachsen.de/aaa/public_alkis/vereinf/wfs?
Service=WFS&REQUEST=GetFeature&Version=2.0.0
&TYPENAMES=ave:Flurstueck&srsName=EPSG:4258
&NAMESPACES=xmlns(ave,http://repository.gdi-de.org/schemas/adv/produkt/
alkis-vereinfacht/2.0)&Count=20000
```

Mithilfe derer wird ein HTTP GET Call an die ausgewählte URL ausgeführt. Die Ausführung der verschieden GET Calls wird mithilfe der Python Library Requests ermöglicht. Durch die Anwendung von einem Try Catch Block werden unerwartete Fehler abgefangen und das Programm nicht in seinem Ablauf gestört. Nach der erwarteten Response vom angefragtem WFS, werden die Datensätze im Ordner Testdata gespeichert. Nun wird für jede der fünf xml Dateien die Transformation durchgeführt. Die Datensätze werden Schritt für Schritt in einer FOR-Schleife abgehandelt. Zunächst wird der Name

der Datei die transformiert werden soll in die Yarrml Mapping File eingetragen. Dies gelingt indem, in der zum ALKIS Schema(da es drei aufkommende Schemavarianten gibt) passenden Mapping File, nach dem String access: "gesucht wird und an dieser Stelle der passende Pfad der xml Datei eingetragen wird.

```
def nameFileToMap(FileName):
    mappingFilePath = Helper.getProjektPath() + "MappingFiles/" + pickMappingFile(FileName)
    with fileinput.FileInput(mappingFilePath,
                             inplace=True) as f:
        for line in f:
            if (line.startswith("      - access:") == True):
                print("      - access: " + FileName, end='\n')
            elif(line.startswith("       access:") == True):
                print("      access: " + FileName, end='\n')
            else:
                print(line, end='')
```

Nun ist es zugesichert das die Funktion yarrrmlToRml() ausgeführt werden kann. Die Aufgabe dieser Funktion ist es den yarrrml-parser aufzurufen(siehe 5.2 genutzte Technologien). Dieser wandelt die bereits definierten yarrrml Mapping Regeln in R2RML Regeln um. Eine Beschreibung der Mapping Regeln erfolgt im Abschnitt 5.2. Dieser Schritt, also die Umwandlung der Mapping Regeln von yarrrml Regeln in R2RML Regeln, ist notwendig da der rmlMapper der im nächsten Schritt verwendet wird nur mit diesem Format umgehen kann. Nach erfolgreicher Umwandlung der Mapping Regeln in R2RML, wird mithilfe der Python Bibliothek subprocess das Programm rmlmapper aufgerufen. Die nötigen Parameter die dieses Programm benötigt sind vordefiniert im Code vorhanden.

Der rmlmapper erzeugt nun anhand der vorliegenden Mapping Regeln Dateien im Turtle Format. Diese enthält alle erzeugten Triple. Hier ein Ausschnitt aus der Datei showCaseFile.ttl für Sachsen.

```
ex:DESNALK05e0000Op a alkis:AX_Flurstueck;
  ngeo:geometrie ex:DESNALK05e0000Op_geometrie;
  alkis:Flurstuecksnummer ex:DESNALK05e0000Op_flurstuecksnummer;
  alkis:abweichenderRechtszustand "Kein abweichender Rechtszustand";
  alkis:amtlicheFlaeche 1529.0;
  alkis:flurstueckskennzeichen "146309___00476000100";
  alkis:gemarkung ex:DESNALK05e0000Op_gemarkung;
  alkis:gemeindezugehoerigkeit ex:DESNALK05e0000Op_gemeindezugehoerigkeit;
  alkis:lageText "Bautzener Straße 40";
  alkis:lebenszeitintervall ex:DESNALK05e0000Op_lebenszeitintervall;
  alkis:sonstigeEigenschaften "Wohnbaufläche;1529" .

ex:DESNALK05e0000Op_flurstuecksnummer a alkis:AX_Flurstuecksnummer;
  alkis:zaehler 476 .

ex:DESNALK05e0000Op_gemarkung a alkis:AX_Gemarkung_Schluessel;
  alkis:gemarkung "Kreba-Neudorf Flur 4";
  alkis:gemarkungsnummer 146309 .

ex:DESNALK05e0000Op_gemeindezugehoerigkeit a alkis:AX_Gemeindekennzeichen;
  alkis:gemeinde "Kreba-Neudorf";
```

```
  alkis:land "Freistaat Sachsen";
  alkis:regierungsbezirk "NUTS 2-Region Dresden" .

ex:DESNALK05e0000Op_lebenszeitintervall a alkis:AA_lebenszeitintervall;
  alkis:beginnt "2013-11-08Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
```

Anschließend an die automatische Transformation werden die fünf erzeugten Turtle Dateien in einer Graph Datenbank Apache Fuseki gespeichert. Es wird eine Update Query an den Fuseki Endpoint gesendet und der Graph wird aktualisiert. Das Ansprechen des Fuseki Endpoint über den Python Programmcode wird mit der Bibliothek rdflib ermöglicht.

```
def executeShowCaseSave():
    store = sparqlstore.SPARQLUpdateStore()
    store.open((Helper.getQueryEndpoint(), Helper.getUpdateEndpoint()))

    alkis_graph = URIRef('http://example.org/alkis_graph')
    store = Graph(store, identifier=alkis_graph)

    filesList = ["Output/Bra/showCaseFile.ttl",
                 "Output/Ham/showcaseFile.ttl",
                 "Output/Hes/showcaseFile.ttl",
                 "Output/NRW/showcaseFile.ttl",
                 "Output/Sac/showcaseFile.ttl",
                 ]
    for file in filesList:
        store.load(file, format="ttl")
```

Im letzten Schritt wird der Fuseki Endpoint über eine vordefinierte Sparql Query abgefragt.

```
hier Sparql Query
```

Die Response des Endpoint ist:

```
...
```

Somit ist die automatisierte Transformation der ALKIS Daten abgeschlossen.

5.2 Beschreibung Mapping Regeln

In diesem Teil der Arbeit, wird beschrieben wie mithilfe von yarrrml Mapping Regeln die Erzeugung der Triples im RDF Format erfolgt. Generell werden alle möglichen ALKIS Objekte gemappt. Die Beschreibung welche Objekte in den .xml Dateien vorkommen können oder vorkommen müssen findet man in https://www.adv-online.de/AdV-Produkte/Standards-und-Produktblaetter/Standards-des-Liegenschaft binarywriterservlet?imgUid=2d4606af-6d37-4551-97b5-9b77072e13d6&uBasVariant=11111111-1111-1111-1111-111111111111 Seite 16. Für jeder dieser Objekte wird eine Mapping Regel erstellt, wenn ein Objekt im abgefragten

Datensatz nicht vorkommt oder es für ein bestimmtes Flurstück einfach nicht existiert, wird dieses auch nicht in den Transformierten RDF Triples vorkommen. Am Anfang des yarrrml Dokumentes, wo die Mapping Regeln erstellt werden, wird definiert auf welche Quelle die Regeln anzuwenden sind. In diesem Beispiel handelt es sich um die Datei vereinfachtes-schema.xml für das Bundesland Sachsen. Der Iterator gibt an auf welche Tiefe im XML Dokument zugegriffen werden soll. Um Zugriff auf die Eigenschaften eines Flurstücks zubekommen ist, der Iterator für Flurstücke auf FeatureCollection/member/Flurstueckëingestellt.

```
sources:
  flurstueck-source:
      access: TestData/Sac/vereinfachtes-schema.xml
      referenceFormulation: xpath
      iterator: FeatureCollection/member/Flurstueck
```

Hier sieht man einen Ausschnitt aus der XML Datei auf die zugegriffen wird.

```
<Flurstueck gml:id="DESNALK05e0000cCFL">
<gml:identifier codeSpace="urn:adv:oid:">urn:adv:oid:DESNALK05e0000cCFL</gml:identifier>
<idflurst>DESNALK05e0000cC</idflurst>
<flstkennz>146325___00121000100</flstkennz>
<land>Freistaat Sachsen</land>
<landschl>14</landschl>
<gemarkung>Kreba-Neudorf Flur 25</gemarkung>
<gemaschl>146325</gemaschl>
<flstnrzae>121</flstnrzae>
<flstnrnen>1</flstnrnen>
<regbezirk>NUTS 2-Region Dresden</regbezirk>
<regbezschl>146</regbezschl>
<kreis>Landkreis Görlitz</kreis>
<kreisschl>14626</kreisschl>
<gemeinde>Kreba-Neudorf</gemeinde>
<gmdschl>14626260</gmdschl>
<oid>DESNALK05e0000cCFL</oid>
<aktualit>2013-11-08Z</aktualit>
<geometrie>
<gml:MultiSurface gml:id="flurstueck.id.86464317.geometrie.Geom_0"
 srsName="urn:ogc:def:crs:EPSG::4258" srsDimension="2">
<gml:surfaceMember>
<gml:Polygon gml:id="flurstueck.id.86464317.geometrie.Geom_1">
<gml:exterior>
<gml:LinearRing>
<gml:posList>51.3278754729631 14.6745851070875 51.3275425205472 14.6748165442732
 51.3281204599786 14.676750975996 51.328539455138 14.6769051793117
  51.3278754729631 14.6745851070875</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Polygon>
</gml:surfaceMember>
</gml:MultiSurface>
</geometrie>
<flaeche>6321</flaeche>
<abwrecht>Kein abweichender Rechtszustand</abwrecht>
<lagebeztxt>ohne Lage</lagebeztxt>
```

```
<tntxt>Wald;6321</tntxt>
</Flurstueck>
```

Um zu entscheiden ob ein ALKIS Objekt schluessendlich zum Subjekt Prädikat oder Objekt wird, wird die im Limbo Projekt erstellte ALKIS Ontologie herangezogen. https://gitlab.com/limbo-project/alkis/-/blob/master/alkis.owl Dort ist angegeben ob ein ALKIS Objekt vom Typ eine Objekt Property oder eine Datatype Property ist. Und man kann die Beziehungen ableiten, die die Objekte untereinander haben. Zuerst wird als Subjekt die Flurstücks ID definiert. Diese ist laut(QUELLE) eindeutig und wird mit der URI:

```
http://example.com/DESNALK05e0000cC
```

eindeutig als Teil des ALKIS Transform Projekts identifiziert.

Der yarrrml-Parser wandelt dann die erstellten yarrrml Regeln in valide R2RML Regeln um, sodass diese von rmlmapper genutztt werden können. Hier ein Ausschnitt aus der Datei R2RML-Mapping.ttl

```
:rules_000 rdf:type void:Dataset ;
void:exampleResource :map_alkis_flurstueck_000, :map_alkis_flurstuecksnummer_000 .

:map_alkis_flurstueck_000 rml:logicalSource :source_000 ;
rdf:type rr:TriplesMap ;
rdfs:label "alkis_flurstueck" ;
rr:subjectMap :s_000 ;
rr:predicateObjectMap :pom_000, :pom_001 .

:source_000 rdf:type rml:LogicalSource ;
rml:source "/Testdata/NRW/AAA-basiert.xml" ;
rml:iterator "FeatureCollection/member/AX_Flurstueck" ;
rml:referenceFormulation ql:XPath .
```

5.3 genutzte Technologien

Um die Anwendung zu entwickeln wurde die Programmiersprache Python gewählt. Als IDE(Integrated Development Environment) wurde PyCharm benutzt. Zur Versionsverwaltung wurde die Software git, mit dem Repository github genutzt. Der Zugang zum Programm Code und einer Anleitung zur Installation des Programmes wird bereit gestellt unter dem Link: https://github.com/banzaiiiiii/ALKIS-transform Zur Umwandlung der Mapping Regeln im .yml Format in R2RML wurde der Yarrrml Parser https://github.com/RMLio/yarrrml-parser benutzt. Zur schlussendlichen Transformation der Daten in das Turtle Format wurde der rmlmapper https://github.com/RMLio/rmlmapper-java verwendet. Weiterhin wurde zur Verarbeitung von RDF im Code, die Python Bibliothek rdflib 6.1.1 https://github.com/RDFLib/

rdflib/#getting-started genutzt. Als RDF Triple Store oder auch Graph Datenbank wurde Apache Jena Fuseki verwendet. https://jena.apache.org/documentation/fuseki2/

https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/
https://www.youtube.com/watch?v=eDRfKAKZMQE
https://db-engines.com/en/system/GraphDB

6 Ergebnisse und Evaluation

Wurde die These/Forschungsfrage gelöst.

ALKIS Daten haben mangelhafte Auffindbarkeit und daraus resultiernde Nutzbarkeit der Daten für Zwecke abseits der professionellen Nutzung durch GIS Programme. Wie hat sich das Nutzungspotenzial der Daten durch die Transformation der Daten geändert?

Vorteile von linked Data: Because of the graph nature, applications can automatically deduct new information (reasoning); Using RDF ontologies, you can give semantic meaning to your data; Because of the linked nature, interlinking with other datasets at other places becomes possible; RDF provides a standard way of querying using SPARQL. The two previous properties allow different datasets to act as a federated system.

7 Ausblick

Die Steuerung des Programms durch die Zahleneingabe Eins bis Vier vereinfacht das Programm und lässt sich auf die Methodik des Prototypings zurückführen. Um die Nutzerfreundlichkeit in Zukunft weiter zu erhöhen sollte ein solches Programm perspektivisch eine Web Anwendung sein. Außerdem können zum jetzigen Zeitpunkt nicht alle ALKIS Schnittstellen der Bundesländer angesprochen werden, da wie bereits erwähnt, einige Bundesländer eine vorherige Anmeldung benötigen oder Inhalte nur kostenpflichtig zur Verfügung stellen. (siehe Tabelle Abschnitt so und so) Um dieses Problem zu lösen, müsste man das Programm , um alle Schnittstellen abrufen zu können um ein Benutzermanagement erweitern. Dies hängt allerdings auch von den einzelnen Bundesländern selbst ab, ob sie sich in Zukunft dazu entscheiden ALKIS Daten unter einer Open Data Lizenz zu veröffentlichen.

Ein weiter Punkt der nützlich wäre ist die Transformation von weiteren ALKIS Daten und das inbetracht ziehen von weiteren Quellen. Neben den Datensätzen über Flurstücke, die in dieser Arbeit transformiert werden, enthält das AAA MOdell viele weitere Datensätze über zum Beispiel Gebäude, tatsächliche Nutzung etc..(siehe geoinfodoc)

Noch ein Punkt ist die Anreicherung der transformierten RDF Graphen durch weitere Daten.... Dies würde den erstellten RDF Wissensgraph noch nützlicher für etwaige

Abfragen machen.

- Abrufen aller Alkis Flurstücke und Transformation. (durch mehrere WFS Aufrufe)

7.1 Notizen

- Zugriffsbeschränkungen sowie Lizenzbestimmungen sind aus getcapabilities Aufruf entnommen - gml:id ist dokumentenweit eindeutig(durch entstehungsdatum-zeit)(quelle geoinfodoc s92)

- - 4 prinzipien: tim berner lee - nutzen von URI's als namen für objekte - zur eindeutigen identifikation für dinge - nutze HTTP URIs - so können Menschen die Objekte nachschauen - nutze anerkannte standarts(RDF, Sparql) - linke URIS weiter - Koordinaten werden in UTM angegeben

8 Quellenverzeichnis

- 4.2.22
zu einleitungserklägungen
- neo4j vs graph db:

Objektenartenübersicht ALKIS: https://www.adv-online.de/icc/extdeu/nav/a63/binarywriterservlet?imgUid=b001016e-7efa-8461-e336-b6951fa2e0&uBasVariant=11111111-1111-1111-1111-111111111111 Objektartenkatalog Seite: 22

GeoinfoDoc: https://www.adv-online.de/GeoInfoDok/binarywriterservlet?imgUid=e9e304b7-96d3-de71-d261-dde30c21d06f&uBasVariant=11111111-1111&isDownload=true

WFS Spezifikation: file:///C:/Users/robin/Downloads/05-008c1_OGC_Web_Services_Common_Specification_Corrigendum.pdf