Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Transformation von Daten des Amtlichen Liegenschaftskatasterinformationssystem in semantische Datenformate

Betreuender Hochschullehrer: Prof. Dr. Hans Gert Gräbe
Außeruniversitäre Betreuer: Dr. Christian Zinke-Wehlmann, Norman Radke vom InfAi Institut

Robin Seidel

6.05.2022

Inhaltsverzeichnis

1 Einleitung

1.1 Zusammenfassung

Diese Bachelorarbeit beschreibt wie eine automatisierte Transformation von Daten aus dem Amtlichen Liegenschaftskataster Informationssystems kurz ALKIS in einen RDF Wissensgraph umzusetzen ist. Im Vordergrund dabei 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 folgende Funktionen umfasst: Download der Daten, Transformation der Daten, Speicherung und Bereitstellung durch Sparql Querys.

1.2 Motivation

Mit der Einführung des World Wide Webs im Jahre 1989(Quelle 1) 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 Webseiten. 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 90% der Bevölkerung festgehalten wurde. (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 auch wichtig einen Weg zu finden die Daten im Internet so zur Verfügung zustellen, sodass sie effizient von Computern verarbeitet werden können um eine optimale Informationsverarbeitung und Verteilung gewährleisten zu können. Die Lösung hierzu sind RDF Graphen, oder auch das semantische Web genannt. Das semantische Web war bereits eine Idee des Entwicklers und W3C Gründers Tim Burner Lee.(Quelle) 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. Durch diese besondere Art der Daten kann eine 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.(Quelle) 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 daraus resultierende verbesserte Nutzbarkeit für viele weitere Bereiche der Informationsbeschaffung und Informationsverarbeitung. (Quelle) Die Daten des amtlichen Liegenschaftskatasterinformationssystems werden zwar in einem gewissen Standard und einer Einheitlichkeit durch die amtlichen Vermessungsinstitute erhoben, jedoch ist die Nutzung dieser Daten hauptsächlich für Nutzer eines Geoinformationssystems (GIS) vorgesehen.(Quelle) Eine Nutzung der Daten durch gewöhnliche Nutzer, die nicht mit GIS vertraut sind ist nur sehr erschwert möglich. Eine Überführung der ALKIS Daten in einen RDF Wissensgraphen ermöglicht es das Potenzial der Daten freizuschalten und die Nutzung für zahlreiche andere Bereiche zur Verfügung zustellen. In diesem Projekt soll die Erstellung eines solchen RDF Wissensgraphen exemplarisch erläutert werden.

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.(QUelle) 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. Die Verwaltung des AAA- Modells erfolgt durch die ADV (Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland). Das AAA Modell ist seit 2015 in allen Bundesländern aktiv. Ziel der Entwicklung eines solchen Modells war es, eine einheitlich 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.

2.2 ALKIS Daten

In den ALKIS Daten werden Objekte beschrieben die in Verbindung mit den Liegenschaften stehen. Diese Objekte sind unterteilt in Objektartengruppen. (QUelle Objektartenkatalog) Neben der Objektart Angaben zum Flurstück, existieren auch zum Beispiel die Bereiche Gebäude und Tatsächliche Nutzung. Die exakte Definition der ALKIS Objekte kann dem Objektartenkatalog Seite 22 entnommen werden. Im Rahmen des Limbo Projekteshttps://gitlab.com/limbo-project/alkis, ein Projekt was sich zum Ziel genommen hat Open Data zuverbessern und leichter zugänglich zumachen.https://www.limbo-project.org/, wurde bereits eine Ontologie erstellt die die ALKIS Domain beschreiben soll. Diese richtet sich nach den Definitionen im ALKIS Objektartenkatalog. (TODO eventuell hier Diagramm Ont) 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_webdienste20.3.2022 Die Hauptaufgabe eines WMS ist die Visualisierung der Geodaten, also 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 gewünschte 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=DescribeFeatureType wird 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&Count=10`. Diese Beispiel Anfrage würde 10 Objekte vom Typ Flurstück für das Bundesland NRW liefern. Die Anfrage selbst ist ein HTTP Get Call und beinhaltet neben der Basis URL verschiedene Parameter die erforderlich (z.B.: Version, Typenames) oder optional (z.B.:Count, Resulttype) sein können. Da der WFS ein Downloaddienst ist, erhält man als Response ein XML Dokument, das die abgefragten ALKIS Objekte enthält. Generell existieren drei Datenaustauschformate die vom WFS angefordert werden können. (Seite 4 Produktspezifikation) Es gibt das NAS-konforme Schema, dieses ist für fachlich sehr gut vertraute Nutzer, speziell für Facharbeiter im Vermessungsbereich gedacht. Da die Geometriestrukturen dort sehr komplex aufgebaut sind und nur über spezielle Programme verarbeitet werden können. Weiterhin gibt es das AAA-konforme Schema, dieses ist mit herkömmlichen GIS Systemem nutzbar und ist für fachlich gut vertraute Nutzer entworfen. Außerdem gibt es das vereinfachte Schema, welches für fachlich nicht tief vertraute Nutzer gedacht ist. Dieses Schema weicht leicht von dem im AAA-Modell definierten Objekten ab. Die Inhalte werden in einer leicht vereinfachten Form definiert. Die Bundesländer sind verpflichtet die ALKIS Daten zuveröffentlichen. Jedoch sind in den Produktspezifikationen des ADV keine Festlegungen von eventuellen Zugriffsrechten und Abrechnungsmodellen inbegriffen. Woraus gewisse Zugriffsbeschränkungen einiger Bundesländer resultieren.(siehe Tabelle) 10 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`

Bundesland	vereinfachtes Schema	AAA Schema	NAS Schema	Zugriffsbeschr.
Baden-Württemberg				Authtentifizierung, Quellennachweiß: LGL, www.lgl-bw.de
Bayern	X			Authentifizierung
Berlin		X		Quellennachweiß: Geoportal Berlin, dl-de/by-2-0, (Daten geändert)
Brandenburg	X	X	X	Quellennachweiß: GeoBasis-DE/LGB,dl-de/by-2-0,(Daten geändert)
Bremen			X	Authentifizierung
Hamburg	X			keine
Hessen	X		X	keine
Mecklenburg-Vorpommern	X			Kostenpflichtig, Quellennachweiß: GeoBasis-DE/M-V 2022 (Daten geändert)
Niedersachsen	X	X	X	Kostenpflichtig
NRW	X	X	X	Quellennachweiß: Geobasis NRW, dl-de/by-2-0,(Daten geändert)
Rheinland-Pfalz	X			Authtentifizierung, kostenpflichtig
Saarland	X			Authtentifizierung, kostenpflichtig
Sachsen	X			Quellennachweißes: GeoSN, dl-de/by-2-0,(Daten geändert)
Sachsen-Anhalt	X		X	kostenpflichtig
Schleswig-Holtstein				Authentifizierung, kostenpflichtig
Thüringen	X			Authentifizierung und Quellennachweiß: GDI-TH, dl-de/by-2-0, (Daten geändert)

*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 vorstellen. 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 dbpedia 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 es entstehen Netzwerke die auch Graphen genannt werden. 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.

```
<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 sind es jedoch zwei verschiedene XML Bäume. Dies zeigt das die Nutzung von Technologien des Semantischen Webs sehr wertvoll sein können, da die automatisierte Verarbeitung von Daten für viele Dinge die Grundlage bildet. Ein Beispiel wäre die Entwicklung von Algorithmen die auf künstlicher Intelligenz beruhen. Solche Algorithmen arbeiten oft erst dann effizient, wenn Sie auf Daten der gleichen Art aufbauen und eben nicht auf Massen von unstrukturierten Daten. (Quelle einfügen)

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.(Quelle einfügen) 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.

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 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.

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 - Showcase

- Der Nutzer kann die Anzahl der abgerufenen Objekte beschränken.

- Es sollen alle verfügbaren ALKIS Schnittstellen abgefragt, und die verfügbaren Datensätze heruntergeladen werden.
- Alle vorhandenen Datensätze sollen transformiert werden.
- Alle erfolgreich umgewandelten Datensätze werden abgespeichert.
- Dem Nutzer wird über eine Textausgabe vermittelt ob die Transformation erfolgreich war.

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 sollten 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

Das Programm soll dem Nutzer als 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. Wobei der erste Fall die automatisierte Transformation beinhaltet. Fälle 2 bis 5 dienen als Werkzeuge um die Teilschritte der Transformation auszuführen. 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 wie das Kernproblem gelöst wurde, wurde im Programm ein Showcase implementiert, dieser soll den Anwendungsfall 1 abbilden. 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, werden für diesen Showcase als Datengrundlage die ALKIS Daten im vereinfachten Schema genutzt. 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 zu vermeiden 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 50000 Flurstück Objekte begrenzt. Der Grund warum nicht alle Datensätze für den Showcase heruntergeladen werden, ist Zeit, die dies beanspruchen würde. Eine Anfrage an den WFS ist auf 20000 Flurstücke begrenzt. Mithilfe von einer Pagination(TODO Erklärung) ist es möglich an alle Objekte zukommen. In diesem Programm wird für jeden WFS Response eine neue XML Datei angelegt. Für das Bundesland Sachsen wäre eine XML Datei die alle Flurstück Objekte beinhaltet etwas 6 Gigabyte groß. Was für eine Downloadrate von 100000 kbit/s eine Downloadzeit von 8 Minuten ausmacht. Die Downloadzeit ist also nicht unbedingt das Problem. Aufgrund einer gewissen Komplexität der Mapping Regeln besitzt das Programm rmlmapper jedoch eine sehr hohe Laufzeit. Nach einigen Tests, mit einer Rechenleistung eines Intel(R) Core(TM) i5-4670K CPU @ 3.40GHz, ergibt sich eine Laufzeit von ungefähr 6 Stunden für eine XML Datei mit 6 Gigabyte ergeben. Für die 5 verfügbaren Bundesländer ergibt sich somit eine Gesamtzeit von mindestens 30 Stunden. Dies

ist für diesen Showcase nicht umsetzbar. Deshalb werden die Flurstück Objekte auf eine Anzahl von 50000 begrenzt. 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, um den Erfolg der Transformation zu zeigen.

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 5 Bundesländer heruntergeladen. 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 Request URL 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=10000&STARTINDEX=0
```

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 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 showCaseFile0.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 erzeugten Turtle Dateien in der 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:

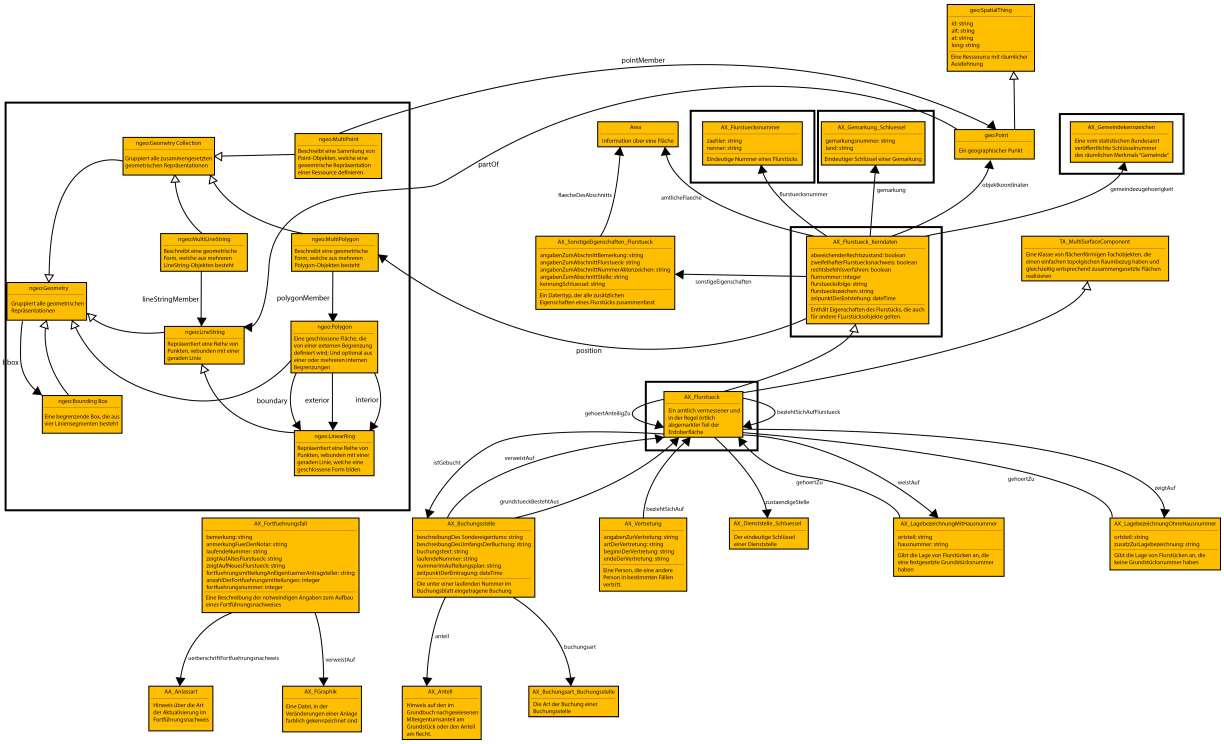
```
...
```

Somit ist die automatisierte Transformation der ALKIS Daten abgeschlossen.

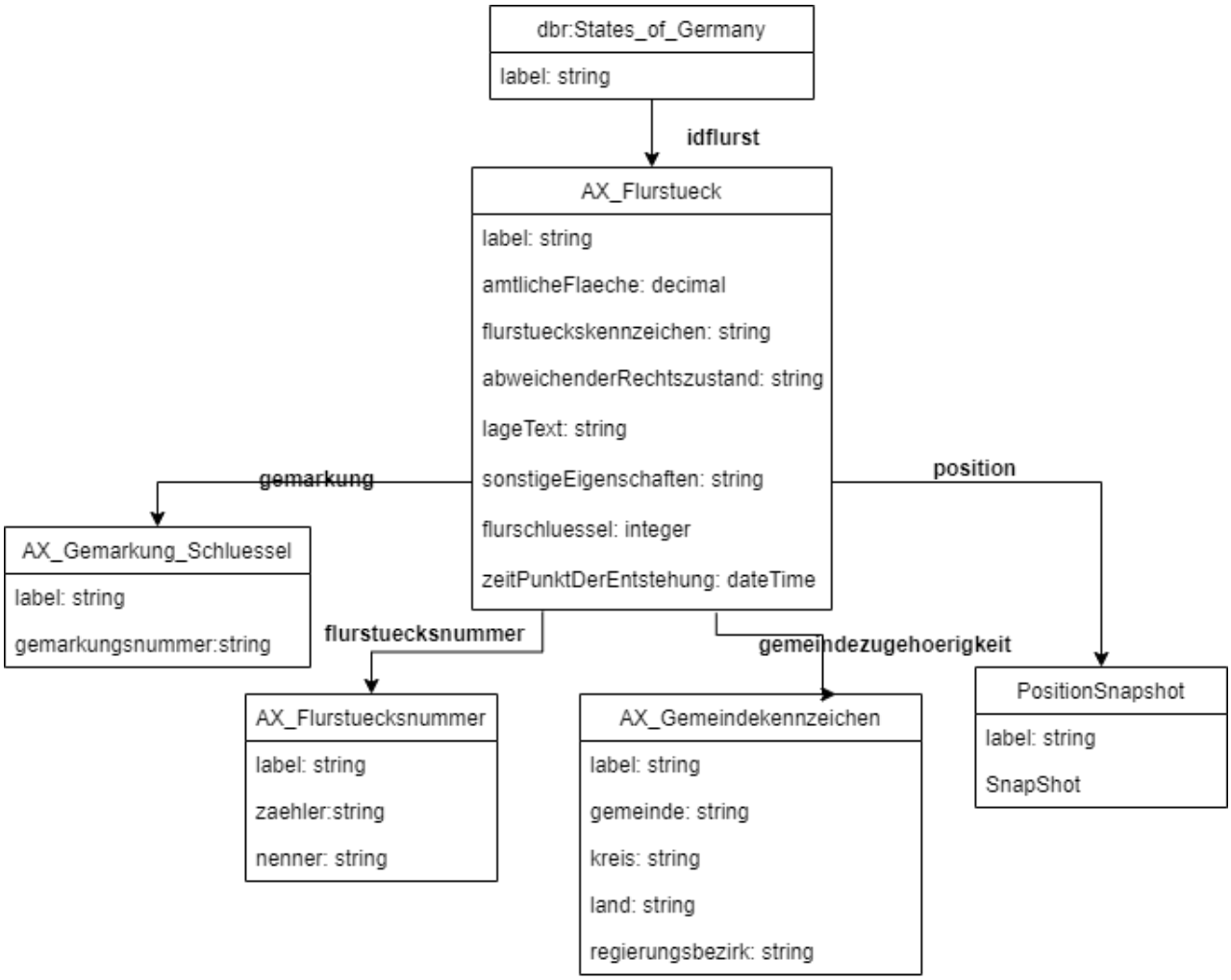
5.2 Beschreibung Mapping

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 des vereinfachten Schemas vorkommen können 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 17. Das Zielformat für die Transformation ist die im Limbo Projekt erstellte ALKIS Ontology (Todo

Link einfügen) Diese wiederum beruht auf der durch den ADV erstellten Objektartenkatalog. `https://www.adv-online.de/icc/extdeu/nav/a63/binarywriterservlet?imgUid=b001016e-7efa-8461-e336-b6951fa2e0c9&uBasVariant=11111111-1111` In jenem ist exakt beschreiben wie sich die Objekte des AAA- Modells zusammensetzen und welche Relationen sie besitzen. In dem für dieses Projekt verwendet vereinfachten Schema, sind nicht alle Objekte und Relationen enthalten. In der folgenden Grafik ist zuerkennen welche Objekte der ALKIS Ontology im vereinfachten Schema auftreten.



Das entgültige vereinfachte Zielformat für die Transformation dieses Projektes sieht so aus.



Nun folgt eine Beschreibung der Yarrrml Mapping Regeln, die angeben wie aus der XML Datei, die RDF Tripel entstehen. Für jedes der 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, es wird einfach übersprungen. 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 zu bekommen ist der Iterator für Flurstücke auf FeatureCollection/member/Flurstueckëingestellt.

```
sources:
  flurstueck-source:
      access: TestData/Sac/vereinfachtes-schema0.xml
      referenceFormulation: xpath
      iterator: FeatureCollection/member/Flurstueck
```

Hier sieht man einen Ausschnitt aus der XML Datei auf die zugegriffen wird. Indem ein Flurstück Objekt mit der ID DESNALK05e0000cC beschrieben wird. TODO Diese ID beginnt mit dem Kürzel DE, das für Deutschland steht. Danach folgt ein Kürzel für

das jeweilige Bundesland. Eine Zahlen und Buchstabenfolge identifiziert das Flurstück eindeutig(Todo).quelle geoinfodoc s92

```
<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"
<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 14.676
</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>
```

Als erstes Subjekt wird das jeweilige Bundesland vom Typ `https://dbpedia.org/resource/States_of_Germany` gemappt. Es besitzt die Relation AX_Flurstueck mit dem Objekt idflurst.

```
subjects: dbr:$(land)
    predicateobjects:
      - [ dbo:type, dbr:States_of_Germany ]
      - [ rdfs:label, "Bundesland der BRD"]
      - predicates: alkis:AX_Flurstueck
        objects:
          value: ex:$(idflurst)
          type: iri
```

Aus dieser Mapping Regeln entstehen später die Tripel:

```
hier tripel für Bundesland mit Flurstücken
```

Das Objekt http://example.com/$(idflurst) wird nun selbst zum Subjekt. Es wird mit dem Prefix ex:, der für http://example.com steht und der ID des Flurstück eindeutig identifiziert.

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

Weiterhin ist das erste Tripel mit der Flurstücks Id vom Type alkis:AX_Flurstueck. Woraus sich das hier angegebene Tripel ergibt.

```
<http://example.com/DESNALK05e0000cC>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://w3id.org/alkis/AX_Flurstueck>
```

Weiterhin hat ein Flurstück mehrere einfache Attribute die folgendermaßen festgelegt werden.

```
predicateobjects:
      - [ a, alkis:AX_Flurstueck ]
      - [ rdfs:label, "Ein Flurstueck ist ein Teil der Erdoberfläche, der von einer im Liegenschaftskata
      - [ alkis:amtlicheFlaeche, $(flaeche), xsd:decimal ]
      - [ alkis:flurstueckskennzeichen, $(flstkennz), xsd:string ]
      - [ alkis:abweichenderRechtszustand, $(abwrecht), xsd:string ]
      - [ alkis:lageText, $(lagebeztxt), xsd:string ]
      - [ alkis:sonstigeEigenschaften, $(tntxt), xsd:string ]
      - [ alkis:flurschluessel, $(flurschl), xsd:integer ]
      - [ alkis:zeitPunktDerEntstehung, $(aktualit), xsd:dateTime ]
```

Das Symbol '$' zeigt hier an wann es sich um eine Variable handelt, die aus einer XML Datei genommen wird. Danach werden die Objekte definiert die später selbst als Subjekte dienen, da diese selber weitere Eigenschaften besitzen. Wie Zum Beispiel die Gemarkung eines Flurstücks.

```
- predicates: alkis:gemarkung
        objects:
          value: ex:$(idflurst)_gemarkung
          type: iri
```

Diese wird hier mit einer eindeutigen IRI definiert. Diese setzt sich aus dem Prefix ex: und der ID des Flurstücks zusammen. Nach der Transformation bildet sie folgendes Tripel:

```
-
<http://example.com/DESNALK05e0000cC> <https://w3id.org/alkis/gemarkung>
<http://example.com/DESNALK05e0000cC_gemarkung>
```

Im Wortlaut bedeutet dies, die ID DESNALK05e0000cC hat als Gemarkung den Wert DESNALK05e0000cC_gemarkung. Da das Objekt in diesem Tripel als IRI dargestellt ist, bedeutet dies auch, dass es ebenfalls als ein neues Subjekt dient. Dieses wird folgendermaßen beschrieben:

```
alkis_gemarkung:
    sources: flurstueck-source
    subjects: ex:$(idflurst)_gemarkung
    predicateobjects:
      - [ a, alkis:AX_Gemarkung_Schluessel ]
      - [ alkis:gemarkungsnummer, $(gemaschl), xsd:integer ]
      - [ alkis:gemarkung, $(gemarkung), xsd:string]
```

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

```
:rules_000 rdf:type void:Dataset ;
void:exampleResource :map_bundesland_000, :map_alkis_flurstueck_000, :map_alkis_gemarkung_000, :map_alki

:source_000 rdf:type rml:LogicalSource ;
rdfs:label "flurstueck-source" ;
rml:source "TestData/Sac/vereinfachtes-schema0.xml" ;
rml:iterator "FeatureCollection/member/Flurstueck" ;
rml:referenceFormulation ql:XPath .

:map_bundesland_000 rml:logicalSource :source_000 ;
rdf:type rr:TriplesMap ;
rdfs:label "bundesland" ;
rr:subjectMap :s_000 ;
rr:predicateObjectMap :pom_000, :pom_001, :pom_002 .

:s_000 rdf:type rr:SubjectMap ;
rr:template "https://dbpedia.org/resource/{land}" .

:pom_000 rdf:type rr:PredicateObjectMap ;
rr:predicateMap :pm_000 ;
rr:objectMap :om_000 .

:pm_000 rdf:type rr:PredicateMap ;
rr:constant dbo:type .

:om_000 rdf:type rr:ObjectMap ;
rr:constant "https://dbpedia.org/resource/States_of_Germany" ;
rr:termType rr:Literal .
```

Diese Datei wird schlussendlich vom rmlmapper benutzt um die finale Turtle Datei zuerstellen.

5.3 genutzte Technologien

(todo umformulierung) 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 der URL: `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 wurde Apache Jena Fuseki verwendet. `https://jena.apache.org/documentation/fuseki2/`

6 Ergebnisse und Evaluation

ALKIS Daten haben mangelhafte Auffindbarkeit und eine daraus resultiernde schlechtere 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? Das Ergebnis der Transformation der ALKIS Daten ist ein RDF Wissensgraph, der in einer Apache Fuseki Graph Datenbank abgespeichert ist. Die Vorteile der nun vorliegenden Daten ist klar erkennbar. Dadurch das die Subjekte und Objekte des RDF Graphen als eindeutige IRI'S vorkommen, bekommen die Daten eine Bedeutung, da sie einer Ontology zu zuordnen sind. Aufgrund dessen, dass die Daten nun als 'linked Data' vorliegen ist eine Verknüpfung mit anderen Daten möglich. Dadurch dass die Daten in einer Graph Datenbank, so wie Apache Fuseki eine ist, abgespeichert sind, ist die Anwendung der standardisierten Abfragesprache SPARQL möglich. Die Daten können mit durch SELECT Querys abgefragt werden.

```
select query
```

Oder durch UPDATE beziehungsweise CONSTRUCT Querys verändert oder erweitert werden.

```
update query
```

Eine weitere These der Arbeit war ob eine vollständige Transformation der ALKIS Daten in RDF überhaupt möglich sei. Auch diese These wurde durch die Programmierung des Programmes ALKIS-transform bewiesen. Eine Anleitung zur Nutzung des Programmes findet man unter `https://github.com/banzaiiiiii/ALKIS-transform/blob/master/README.md`

7 Ausblick

Die Steuerung des Programms durch die Zahleneingabe Eins bis Fünf 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 werden. 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 stehen. (TODO) 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 in Betracht ziehen von weiteren Quellen. Neben den Datensätzen aus dem Vereinfachten Schema, die in dieser Arbeit transformiert wurden, enthält das AAA Modell viele weitere Datensätze über zum Beispiel Gebäude und die Nutzung von Flurstücken. Noch ein Punkt ist die Anreicherung der transformierten RDF Graphen durch weitere Daten.(TODO) Dies würde den erstellten RDF Wissensgraph noch nützlicher für etwaige Abfragen machen.

Weiterhin konnten aufgrund sehr hoher Transformationszeit und mangelnder Rechenleistung nicht alle ALKIS Flurstücke abgerufen und transformiert werden. Dieses Problem ist zu lösen indem man das entwickelte Programm ALKIS-transform auf zum Beispiel einem immer laufenden Server installiert und über eine längere Zeit laufen lässt. Um die transformierten Daten aktuell zuhalten wäre eine Aktualisierung immer dann notwendig wenn auch die ALKIS Daten aktualisiert werden, also etwa alle 3 Monate.(TODO Quelle)

7.1 Notizen

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

Koordinaten werden in UTM angegeben

8 Quellenverzeichnis

8.1 Quellen ALKIS Daten

Für Bundesländer wo die ALKIS Daten direkt abrufbar sind, wird die URL zum abrufen der Daten angegeben(mit getCapabilities Funktion). Für Bundesländer wo eine Anmeldung oder eine Bezahlung notwendig ist, wird die dahingehende Quelle angegeben. Die WFS Endpunkte, für die Bundesländer wo eine Authentifizierung notwendig ist, sind in der pyton Datei AlkisDataService.py in einem Dictionary gespeichert. Wenn mehrere abrufbare Datensätze verfügbar sind, ist die URL zu dem vereinfachten Schema angegeben.
- Baden-Württemberg:
`https://www.lgl-bw.de/export/sites/lgl/Produkte/Galerien/Dokumente/Flyer_WFS_ALKIS.pdf`
Abruf Datum: 3.5.2022
- Bayern:
`https://geodatenonline.bayern.de/geodatenonline/seiten/wfs_alkis`
Abruf Datum: 3.5.2022
- Berlin:
`https://fbinter.stadt-berlin.de/fb/wfs/data/senstadt/s_wfs_alkis?Service=WFS&REQUEST=Getcapabilities`
Abruf Datum: 3.5.2022
- Brandenburg:
`https://isk.geobasis-bb.de/ows/alkis_vereinf_wfs?Service=WFS&REQUEST=Getcapabilities`
Abruf Datum: 3.5.2022
- Bremen:
`https://metaver.de/trefferanzeige?cmd=doShowDocument&docuuid=FA4D0ED5-EE47-4959-A620-DA34AA76663E&plugid=/ingrid-group:ige-iplug-h`
Abruf Datum: 3.5.2022
- Hamburg:
`https://geodienste.hamburg.de/WFS_HH_ALKIS_vereinfacht?Service=WFS&REQUEST=Getcapabilities`
Abruf Datum: 3.5.2022
- Hessen:
`https://www.gds.hessen.de/wfs2/aaa-suite/cgi-bin/alkis/vereinf/wfs?Service=WFS&REQUEST=Getcapabilities`
Abruf Datum: 3.5.2022
- Mecklenburg-Vorpommern:
`https://www.geodaten-mv.de/dienste/alkis_wfs_einfach?Service=WFS&REQUEST=Getcapabilities`

Abruf Datum: 3.5.2022
- Niedersachsen:
`https://www.lgln.niedersachsen.de/download/126454/Informationsuebersi`
`Geodatendienste.pdf`
Abruf Datum: 3.5.2022
- NRW:
`https://www.wfs.nrw.de/geobasis/wfs_nw_alkis_vereinfacht?Service=`
`WFS&REQUEST=Getcapabilities`
Abruf Datum: 3.5.2022
- Rheinland-Pfalz:
`https://www.geoportal.rlp.de/spatial-objects/353`
Abruf Datum: 3.5.2022
- Saarland:
`https://geoportal.saarland.de/spatial-objects/325?Service=WFS&`
`Request=GetCapabilities`
Abruf Datum: 3.5.2022
- Sachsen:
`https://geodienste.sachsen.de/aaa/public_alkis/vereinf/wfs?Service=`
`WFS&Request=GetCapabilities`
Abruf Datum: 3.5.2022
- Sachsen-Anhalt:
`https://www.lvermgeo.sachsen-anhalt.de/de/alkis-geowebdienste.`
`html`
Abruf Datum: 3.5.2022
- Schleswig-Holtstein:
`https://www.schleswig-holstein.de/DE/Landesregierung/LVERMGEOSH/`
`Service/serviceGeobasisdaten/geodatenService_Geobasisdaten_digALKIS.`
`html`
Abruf Datum: 3.5.2022
- Thüringen:
`http://www.geoproxy.geoportal-th.de/geoproxy/services?SERVICE=`
`WFS&REQUEST=GetCapabilities`
Abruf Datum: 3.5.2022

8.2 Quellen