Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

# Transformation von Daten des Amtlichen Liegenschaftskatasterinformationssystem in semantische Datenformate

BACHELORARBEIT

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

Robin Seidel

9.5.2022

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Zusammenfassung

Diese Bachelorarbeit beschreibt wie eine automatisierte Transformation von Daten aus dem Amtlichen Liegenschaftskatasterinformationssystems kurz ALKIS in einen RDF Wissensgraph umzusetzen ist. Im Vordergrund 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 durch Tim Burner Lee ist das Internet heute viel mehr als dass 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. Destatis - 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, dass sie effizient von Computern verarbeitet werden können, um eine optimale Informationsverarbeitung und Verteilung gewährleisten zu können. Die Lösung hierzu sind RDF Graphen, die ein semantisches Web bilden sollen. Das semantische Web war bereits eine Idee des Entwicklers und W3C 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. Aus diesem Grund hat Burner Lee 4 wichtige Regeln erstellt, die angeben wie ein Web aus Linked Data oder gleichermaßen ein Web aus Webseiten wachsen und entstehen soll:
1. Use URIs as names for things

2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)
4. Include links to other URIs. so that they can discover more things.
(Berners-Lee, Tim 07/2006)
Durch diese besondere Art der Daten kann eine Software diese anschließend besser nutzen und verarbeiten. Das semantische Web ist damit sozusagen eine Alternative zu Webseiten die ausschließlich für die menschliche Nutzung geeignet sind. (vgl. Kück, G(2004) 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. Die ALKIS Daten 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. Eine Nutzung der Daten durch gewöhnliche Nutzer, die nicht mit GIS vertraut sind ist nur sehr schwer 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. In ATKIS (Amtliches Topographisch-Kartographisches Informationssystem) geht es um digitale Karten zu Landschaften und Topografie.(vgl. ALKIS GeoInfoDok S.15/16) ALKIS umfasst die für uns relevanten Informationen aus dem Liegenschaftsbuch und Liegenschaftskarte. 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.Bei der Erschaffung des Datenmodells wurden Standards und Normen

verwendet die vom W3C (World Wide Web Consortium), OGC (Open Geospatial Consortium Inc) und von der ISO (International Organization for Standardization) anerkannt sind. Durch die Einführung von ALKIS wurden alle Daten des Liegenschaftskatasters redundanzfrei zusammengeführt. (vgl. ADV 2022) Kataster- 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. Diese Nachteile konnten durch die Einführung von ALKIS gelöst werden.(vgl. ALKIS GeoInfoDoc ab S.15)

## 2.2 ALKIS Daten

In den ALKIS Daten werden Objekte beschrieben die in Verbindung mit den Liegenschaften stehen. Diese Objekte sind unterteilt in Objektartengruppen. Neben der Objektart Angaben zum Flurstück, existieren auch zum Beispiel die Bereiche Gebäude und Tatsächliche Nutzung. (vgl. ALKIS Objektartenkatalog ab S.22) Die exakte Definition der ALKIS Objekte kann dem Objektartenkatalog entnommen werden. Im Rahmen des Limbo Projektes[1] wurde bereits eine Ontologie[2] erstellt die die ALKIS Domain beschreiben soll. Diese richtet sich nach den Definitionen im ALKIS Objektartenkatalog. Eine Ontologie('Lehre vom Sein') ist eine geordnete Menge von Begriffen und Beziehungen, die ein Objekt eines Bereiches oder auch abstrakte Konzepte formal beschreibt. (vgl. Retresco) 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. (vgl. ALKIS GeoInfoDok ab S.92 4.1) 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 definierten Geodatendienste sind ein Teil der NAS Spezifikation. Geodatendienste sind standardisierte Services die über eine URL aufgerufen und zum Austausch von Daten genutzt werden. (vgl. Bayerische Vermessungsverwaltung) Die Hauptaufgabe eines WMS ist die Visualisierung der Geodaten, der WMS ist also ein Darstellungsdienst für geografische Daten. Die Antwort (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

[1] ein Projekt mit dem Ziel Open Data zu verbessern und leichter zugänglich zumachen (vgl. Limbo Projekt `https://www.limbo-project.org/`)

[2] ALKIS Ontologie: `https://gitlab.com/limbo-project/alkis/-/blob/master/alkis.owl`

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. Die Basis URL ist für das Beispiel NRW folgende:

```
https://www.wfs.nrw.de/geobasis/wfs_nw_alkis_vereinfacht?
```

Listing 1: WFS Basis URL

An diese URL können nun verschiedene Parameter angehangen werden, um die gewünschte Response des Servers zu erhalten. Der WFS hat verschiedene Abfrageoptionen. Mit

```
Request=GetCapabilities
```

Listing 2: WFS GetCapabilities Aufruf

werden alle Fähigkeiten und Metadaten des aufgerufenen Dienstes abgefragt.Durch

```
Request=DescribeFeatureType
```

Listing 3: WFS DescribeFeatureType Aufruf

wird die Struktur eines FeatureTypes abgefragt. Die wichtigste Abfrageoption ist allerdings folgende

```
Request=GetFeature
```

Listing 4: WFS GetFeature Aufruf

Diese liefert die gewünschten Objekte. Eine komplette 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
```

Listing 5: WFS Beispiel Request URL

Dieses Beispiel würde 10 Objekte vom Typ Flurstück für das Bundesland NRW liefern. Die Request selbst ist ein HTTP Get Call[3] 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 Download Dienst ist, erhält man als Response ein XML Dokument, das die abgefragten ALKIS Objekte enthält. Generell existieren drei Datenaustauschformate die vom WFS angefordert werden können. (vgl. AdV 2016: WFS Produktspezifikation S.4) 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 (vgl. WFS Produktspezifikation S.10), 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

[3]GET CALL: Ein HTTP GET Call dient dazu eine Ressource eines Web Servers anzufordern

vereinfachten Form definiert. Die Bundesländer sind verpflichtet die ALKIS Daten zu veröffentlichen. Jedoch sind in den Produktspezifikationen des ADV keine Festlegungen von eventuellen Zugriffsrechten und Abrechnungsmodellen inbegriffen. Woraus gewisse Zugriffsbeschränkungen einiger Bundesländer resultieren. (Abbildung 1) 10 Bundesländer haben zu mindestens einige Geodatensätze frei zugänglich gemacht. Sie unterliegen der Open Data Lizenz: "Datenlizenz Deutschland - Namensnennung- Version2.0"[4]

[4]Lizenz:`https://www.govdata.de/dl-de/by-2-0`

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

*Die Endpoints für die einzelnen Bundesländer sind im Anhang zufinden.
*Die benötigten Lizenzangaben der Daten wurden dem jeweiligen getCapabilities Aufruf entnommen. Beispiel: `https://geodienste.sachsen.de/aaa/public_alkis/vereinf/wfs?Service=WFS&Request=GetCapabilities` Abschnitt Fees. Aus diesem Grund konnte für zugriffsbeschränkte Bundesländer eine eventuell notwendige Lizenzangabe nicht ermittelt werden.

Tabelle 1: Tabelle ALKIS Daten

## 2.3 Das Resource Description Framework

Das Resource Description Framework kurz RDF, ist ein vom W3C konzipiertes Datenformat zum Beschreiben von Objekten im Internet. RDF ist ein universelles, maschinenlesbares Format, das für den Austausch und der Darstellung von jeglicher Art von Daten gedacht ist. (vgl. W3C 02/2004) Ursprünglich wurde es genutzt um Metadaten zu beschreiben. Später wurde es auch in vielen verschiedenen Gebieten und generellen Anwendungen benutzt. Heute gilt es als grundlegender Baustein des Semantischen Webs. (vgl. W3C (02/1999)) Eine Aussage in RDF nennt man Tripel. Ein Tripel besteht aus den Teilen Subjekt, Prädikat und Objekt. Ein Subjekt kann man sich als den Identifier einer Entity der zu beschreibenden Aussage vorstellen. Das Prädikat ist in diesem Kontext der Attributname und das Subjekt der Attributwert. Alle 3 Teile eines Tripel können als Uniform Resource Identifier (URI) dargestellt werden, wobei nur das Objekt ein Literal sein kann. Subjekt und das Objekt können auch eine blank node sein. Eine blank note, ist ein Knoten in einem RDF Graphen für den keine URL oder kein Literal angegeben ist.

```
@prefix dbo: https://dbpedia.org/ontology/
@prefix dbr: https://dbpedia.org/resource/

<dbr:Leipzig> <dbo:country> <dbr:Germany> .
```

Listing 6: RDF Example

Im oberen Beispiel[5] identifiziert die URI für 'country' das Prädikat eindeutig. Die URI zeigt das es sich nicht um irgendein beliebiges 'country' handelt sondern, dass es Bestandteil der von Dbpedia veröffentlichten Ontology Country ist. Da Objekte auch als URI's dargestellt werden können, kann die gleiche Ressource Objekt eines Tripel und gleichzeitig als Subjekt eines weiteren Tripel dienen. Im oben gezeigten Beispiel ist 'Germany' das Objekt und in dem Beispiel:

```
<dbr:Germany> <dbo:PopulatedPlace/area> 357022.0 .
```

Listing 7: RDF Example

ist Germany das Subjekt. Somit werden die beschriebenen Aussagen verbunden und es entstehen Netzwerke die auch Graphen genannt werden. Um die URI's kürzer und leichter lesbar zumachen, werden Prefixes angewendet, welche am Anfang eines Dokumentes definiert werden. Ein Prefix beschreibt die ganze URI bis auf den letzten Teil. Die verwendete Turtle Serialisierung zeichnet sich dadurch aus, das wenig Speicherplatz benötigt wird und sie für den Menschen intuitiv lesbar ist. URI's werden in eckigen Klammer geschrieben und Literale in Anführungszeichen. Die Tripel sind durch einen Punkt getrennt.

[5] Aus Gründen der besseren Lesbarkeit wird in allen RDF Beispielen die Turtle Schreibweise verwendet und Prefixes werden nur einmal definiert und dann verwendet.

## 2.4 RML & Yarrrml

Die RDF Mapping Language (RML) (vgl. Ben de Meester, Pieter Heyvaert, Thomas Delva 2020) 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[6] 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 abzubilden. RML ist kein offiziell bestätigter Standard, es erweitert nur R2RML. Da in dieser Arbeit Daten ausgehend vom XML Format umgewandelt werden, ist die Verwendung von RML Regeln zu bevorzugen. Yarrrml wiederum ist eine Sprache, die auf Yaml [7] 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, somit wird die Effektivität der RDF Graphen Erstellung erhöht. (vgl. Ben De Meester, Dylan Van Assche, Pieter Heyvaert 08/2021)

## 2.5 Tripelstores

Zur Speicherung von Graphen werden keine herkömmlichen relationalen Datenbanken genutzt, sondern speziell dafür konzipierte Graphen Datenbanken. Dadurch das 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. (vgl. Ontotext)

## 2.6 SPARQL

SPARQL steht für Sparql Protocol and RDF Query Language und ist ein vom W3C festgelegter Standard 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 Tripel vom angefragten Datensatz angesprochen werden. Die WHERE Klausel besteht aus einer oder mehreren Aussagen. Diese sind wie RDF Tripel, mit dem Unterschied, dass die drei Bestandteile durch Variablen ersetzt werden können. Variablen beginnen mit einem '$'. In der SELECT Klausel wird angegeben welche Variablen dem Nutzer angezeigt werden. (vgl. W3C: SPARQL 1.1 Overview) Im folgenden Sparql Beispiel wird die Fläche von Deutschland erfragt.

[6] R2RML: `https://www.w3.org/TR/r2rml/`
[7] YAML: `https://yaml.org/spec/1.2.2/`

```
SELECT $flaeche
WHERE
        {
        <dbr:Germany> <dbo:PopulatedPlace/area> $flaeche .
        }
```

Listing 8: SPARQL Example

Die Ausgabe der SPARQL Query, ist der in der 'flaeche' Variable zugewiesene Wert.

## 2.7 RDF vs XML

Bei der Recherche dieser Arbeit ist die zentrale Frage entstanden, ob und warum man überhaupt RDF nutzen sollte, wenn dies ohnehin auf XML beruht und die ALKIS Daten bereits in diesem universellem Format vorliegen? Und was ist der Vorteil von RDF gegenüber anderen Datenaustauschformaten? Eine Aussage in XML kann auf verschiedene Arten ausgedrückt werden. Für den Nutzer der diese Aussagen liest bedeuten Sie das Gleiche. Für eine Maschine beziehungsweise einen Computer sind es verschiedene XML Bäume. In RDF wird die Aussage der 'Autor der Bachelorarbeit ist Robin Seidel' als Tripel dargestellt:

```
tripel(Autor, Bachelorarbeit, RobinSeidel)
```

Listing 9: RDF vereinfachte Darstellung

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>
```

Listing 10: XML Beispiel

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, dass 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. (vgl. Berners-Lee, Tim (1998))

# 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. (vgl. Luber, Litzel (2018)) Der Extract Teil des Prozesses ist, die notwendigen ALKIS Daten zu sammeln. Dies gelingt durch einen sogenannten http Request 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 Bestandsdaten. 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 einen möglichst nützlichen Graphen ergeben. Im Load Teil des Prozesses werden die Daten dann in einen Tripel Store geladen. Dieser dient als Datenbank für Graphen. Die transformierten RDF Daten können nun über SPARQL Abfragen abgerufen werden.

## 3.2 Problemanalyse

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. (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, da er alle wichtigen Daten über die Flurstücke der jeweiligen Bundesländer enthält. Wie bereits oben erwähnt werden ALKIS Daten in drei verschiedenen Schema Varianten angeboten. Die Objekte, die diese besitzen unterscheiden sich zwar nur ein wenig, dennoch ist es für das Mapping entscheidend welche Schema Variante man benutzen möchte. Man benötigt für die Transformation zu RDF Daten, drei verschiedene Sets an Mapping Regeln damit diese die XML Dateien richtig ansprechen können. Ein weiteres Problem ist, dass die verschiedenen Bundesländer

nicht alle die gleiche Schema Variante anbieten. Um alle verschiedenen Schemata auf das gleiche Zielformat abzubilden, müsste auf einige Relationen der Objekte verzichtet werden. Zum Beispiel besitzt das vereinfachte Schema die Relation 'zeigt auf' nicht. Diese Relation stellt Verbindungen zwischen Flurstücken her, die in irgendeiner Weise zusammen gehören. Für die Transformation der Daten gibt es auch mehrere Möglichkeiten. Eine Herangehensweise wäre die Entwicklung eines Parsers, der es ermöglicht aus den vorhandenen ALKIS XML Dateien, RDF Dateien direkt 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 sehr komplex, womit ein erheblicher Mehraufwand für das Projekt einhergehen würde. Eine weitere Möglichkeit ist Verwendung von bereits existierenden Software Tools, die entwickelt wurden um Daten aller Art in RDF umzuwandeln. Eine solche Software ist zum Beispiel der RML-Mapper.[8] Auf der einen Seite bietet so eine Software den Vorteil, dass sie für die angegebenen Anwendungsfälle zuverlässig funktioniert, auf der anderen Seite muss man sich intensiv mit der verwendeten Software auseinandersetzen und mit den vorhandenen Gegebenheiten arbeiten. Weiterhin benötigt das Programm RMLMapper Regeln als Input, welche in einem speziellen Format, dem RML Format (siehe Punkt 2.4), vorliegen müssen. Daraus stellt sich wiederum die Frage, auf welche Art und Weise diese RML Regeln entstehen sollen? Für die Erstellung dieser Regeln gibt es folgende zwei Möglichkeiten. Entweder man erstellt auf direktem Wege RML Regeln oder man verwendet eine weitere Software, die den Prozess der Mapping Regel Erstellung erleichtert. Diese Software ist der YARRRML Parser.[9] Dieser wandelt zuvor erstellte Mapping Regeln im Yaml Format in RML Regeln um. Wie bereits in Punkt 2.4 beschrieben, ist es mit Yarrrml möglich auf einfachere Art und Weise deklarative Mapping Regeln zu erstellen. Vorteil an diesem Vorgehen ist, dass Yarrrml Regeln leichter zu erstellen sind als RML Regeln. Nachteil ist das Yarrrml nur in einer inoffiziellen Version vorliegt und die Dokumentation bezüglich der Anwendung daher eher schlecht ist. Für die Abspeicherung von RDF Daten ist wiederum eine spezielle Art von Datenbank nötig, eine sogenannte Graphdatenbank.(siehe 2.5 Tripelstores) Bei jener werden Daten anstatt auf die herkömmliche Art und Weise nicht in Tabellen, sondern in Graphen strukturiert abgespeichert. Auch bei der Abspeicherung der transformierten RDF Daten gab es anfangs mehrere Möglichkeiten. Da das Angebot von Graphdatenbanken relativ groß ist, musste ein für das Projekt passendes System ausgewählt werden. In Frage kam hierfür anfangs die Open Source Graphdatenbank Neo4j [10] mit der Erweiterung neosemantics.[11] Die Implementierung in das Projekt ist einfach und die Speicherung der ALKIS Daten in Neo4j funktioniert ohne Probleme schnell und

[8] RMLMapper:`https://github.com/RMLio/rmlmapper-java`
[9] Yarrrml Parser: `https://github.com/RMLio/yarrrml-parser`
[10] Neo4J: `https://neo4j.com/`
[11] neosemantics: `https://neo4j.com/labs/neosemantics/`

zuverlässig. Jedoch ist es in Neo4j vorgesehen als Abfragesprache Cypher zuverwenden, anstatt der für RDF Graphen üblicherweise verwendeten Abfragesprache SPARQL, da dies die weitere Verwendung mit anderen Programmen einschränken würde. (siehe Punkt 7) Eine weitere Möglichkeit war die Verwendung des Open-Source Framework Apache Jena Fuseki Apache Jena Fuseki ist speziell für den Austausch von semantischen Daten entwickelt.(vgl. Apache Jena Fuseki: Dokumentation) Ein weiteres Problem in den Alkis Daten ist die Darstellung der Geometrie der Flurstücke. Die Struktur der geografischen Objekte die das Flurstück beschreiben sollen sind in ALKIS für die spezielle Nutzung mit GIS Programmen vorgesehen und für eine anderweitige Nutzung unnötig komplex dargestellt.

```
<geometrie>
<gml:MultiSurface gml:id="flurstueck.id.235987316.geometrie.Geom_0" srsName="urn:ogc:def
    :crs:EPSG::4258" srsDimension="2">
<gml:surfaceMember>
<gml:Polygon gml:id="flurstueck.id.235987316.geometrie.Geom_1">
<gml:exterior>
<gml:LinearRing>
<gml:posList>51.4351696588377 13.2288952292288 51.4350508147148 13.2289365868429
    51.4347486187064 13.2290417520494 51.4347425261058 13.2298444685916 51.4352915438997
     13.2295902189621 51.4351987155717 13.2289101763231 51.4351696588377
    13.2288952292288</gml:posList>
</gml:LinearRing>
</gml:exterior>
</gml:Polygon>
</gml:surfaceMember>
</gml:MultiSurface>
</geometrie>
```

Listing 11: Geometrie Struktur aus vereinfachtes-schema.xml

Zum Mapping der Koordinaten, ergeben sich mehrere mögliche Herangehensweisen. Die erste ist das Mapping der Objekte so wie es die AdV Definition und das im Limbo Projekt erstellte Diagramm es vorgeben. (siehe Abbildung 1)

Die Zweite Möglichkeit ist ein Mapping von einem Fingerprint der Objekte. (vgl. Gräbe, Hans-Gert, ab S.6) Bei diesem Vorgehen wird die komplette XML Struktur der Geometrie so übernommen wie sie in der Originaldatei auftritt. Diese Umsetzungsmöglichkeit ist perspektivisch die beste, jedoch ist sie mit den bisher gegebenen Transformationswerkzeugen nicht perfekt umsetzbar, da die XML Struktur als leerer String gemappt wird. Aus diesem Grund wurde der String der die Koordinaten Liste enthält als ein Literal übernommen und in ein einziges Geometrie Objekt gemappt.


Abbildung 1: Auschnitt Geometrie aus ALKIS Ontology Limbo Projekt

# 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

- | NFA 1 | Alle Eingaben des Benutzers müssen validiert werden, bei fehlgeschlagener Validierung wird die Eingabe abgelehnt.
- | NFA 2 | Wenn es zu unerwarteten internen Fehlern im Programm kommt, ist ein Datenverlust zu vermeiden.

- | NFA 3 | Ein unerwartetes Abstürzen des Programms sollte vermieden werden.

### 4.1.2 User Experience

- | NFA 4 | Bei einer Eingabe durch den Nutzer, die das Programm nicht verwerten kann, wird dem Nutzer eine erneute Eingabe ermöglicht.
- | NFA 5 | Bei richtigen oder falschen User Input sollte das Programm dem Nutzer eine angemessene Antwort über Erfolg oder Misserfolg wiedergeben.
- | NFA 6 | 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.
- | NFA 7 | Das Programm sollte von so vielen Betriebssystemen wie möglich unterstützt werden.
- | NFA 8 | Das Programm sollte mit den aktuellen Versionen des RMLMappers und des yarrrml-parsers kompatibel sein.
- | NFA 9 | 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

- | FA 1 | Der Nutzer kann die Anzahl der abgerufenen Objekte beschränken.
- | FA 2 | Es sollen alle verfügbaren ALKIS Schnittstellen abgefragt, und die verfügbaren Datensätze heruntergeladen werden.
- | FA 3 | Alle vorhandenen Datensätze sollen transformiert werden.
- | FA 4 | Alle erfolgreich umgewandelten Datensätze werden abgespeichert.
- | FA 5 | Dem Nutzer wird über eine Textausgabe vermittelt ob die Transformation erfolgreich war.

### 4.2.2 Anforderungen an Download Funktion

- | FA 6 | Der Nutzer sollte die Möglichkeit haben aus einer vorgegebenen Liste, den gewünschten Datensatz auszuwählen.
- | FA 7 | Die Datensätze in der Liste sollten verfügbar sein.

- | FA 8 | Der Nutzer hat die Möglichkeit mit der Angabe der Gemarkung oder Gemarkungsnummer den Abruf der Daten weiter zu spezifizieren.

### 4.2.3 Anforderungen an die Transformation

- | FA 9 | Der Nutzer kann selbst angeben welcher Datensatz transformiert werden soll.

### 4.2.4 Anforderungen an die Speicherung der transformierten Daten

- | FA 9 | Das Programm sollte eine Überführung der Daten in eine Apache Jena Fuseki Graph Datenbank ermöglichen.
- | FA 10 | Der Nutzer kann selbst einen Fuseki Server Endpunkt angeben, an den die Daten übertragen werden sollen.
- | FA 11 | 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

- | FA 12 | Dem Anwender soll ermöglicht werden, die Daten in Form von Sparql Querys abzufragen.
- | FA 13 | Das Ergebnis der Anfrage wird dem Nutzer in Form von Tripels im Turtle Format angezeigt.

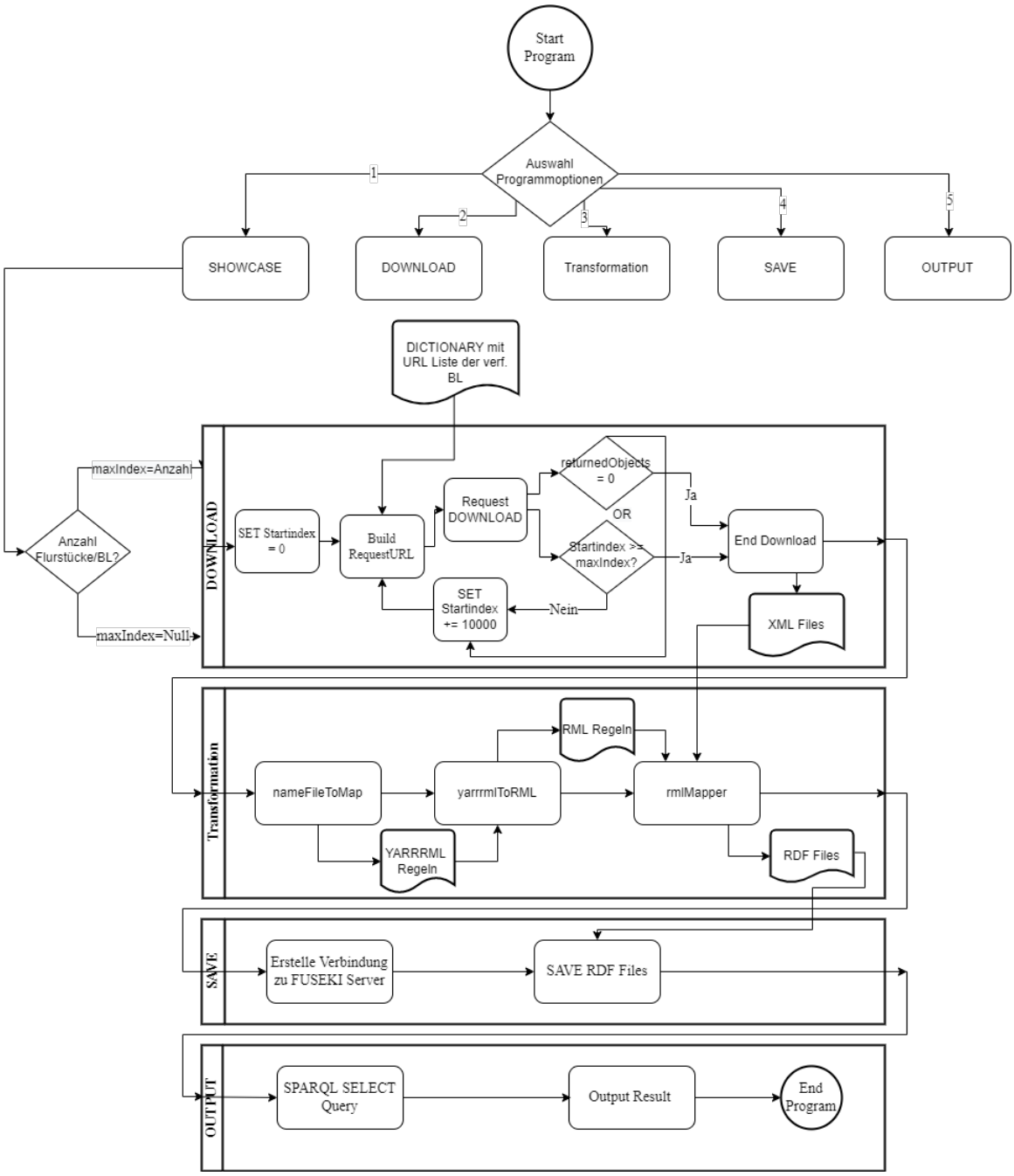
# 5 Implementierung

## 5.1 Beschreibung der Implementierung

Das entwickelte 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 Kernproblem zu lösen. Dies dient zur Bestimmung der Anforderungen und es wird absehbar ob die geplanten Funktionen der Software umsetzbar sind. Dabei werden jedoch gewisse Aspekte vernachlässigt. Zum Beispiel ist es

im Programm nicht vorgesehen für jeden Use Case ausführliche Unit Tests anzufertigen. Weiterhin ist die allgemeine Codequalität eines Prototyps oft nicht optimal. Es werden nicht alle SOLID[12] Prinzipien berücksichtigt. Eine Optimierung und Erweiterung einiger Funktionen im Nachhinein ist wahrscheinlich. (vgl. Businessinsider(01/2019): Prototyping) Das Kernproblem dieses Projektes ist es, einen ALKIS Datensatz zu einem sinnvollen RDF Graphen umzuwandeln. Zur Verdeutlichung der Lösung des Kernproblems, wurde im Programm ein Showcase implementiert, welcher im Programm ausgeführt wird, nachdem der User die Zahl '1' eingibt. Nach minimaler Nutzerinteraktion soll hier eine automatische Transformation der Daten angestoßen werden. Der Showcase für das in dieser Arbeit entwickelte Programm wird folgendermaßen 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 das vereinfachte Schema für alle 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. Zusammenfassend lässt sich sagen das ALKIS Daten für die Bundesländer Sachsen, Hessen, Brandenburg, Nordrhein-Westfalen und Hamburg im vereinfachten Schema heruntergeladen werden. Die heruntergeladen Datensätze werden auf je 5000 Flurstück Objekte pro Bundesland begrenzt. Der Grund warum nicht alle Datensätze für den Showcase heruntergeladen werden, ist die enorme Zeit die dies beanspruchen würde. Eine Request an den WFS ist auf 20000 Flurstücke begrenzt. Mithilfe von einer Pagination, also einer Aufteilung der Flurstück Objekte auf mehrere Dateien, ist es möglich an alle Objekte zukommen. In diesem Programm wird für jede WFS Response eine neue XML Datei angelegt. Für das Bundesland Sachsen wäre eine XML Datei die alle Flurstück Objekte beinhaltet etwa 6 Gigabyte groß. Was für eine Downloadrate von 100000 kbit/s eine Downloadzeit von 8 Minuten ergibt. 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. Für die Daten der 5 verfügbaren Bundesländer ergibt sich somit eine Gesamtzeit von mindestens 30 Stunden. Da diese hohe Berechnungszeit für den Zweck dieses Showcases sehr hoch ist, werden die Flurstück Objekte auf eine Anzahl von 5000 begrenzt.

[12]SOLID: SRP: Single-Responsibility-Prinzip, OCP: Open-Closed-Prinzip, LSP: Liskov'sche Substitutions-Prinzip, ISP: Interface-Segregation-Prinzip, DIP: Dependency-Inversion-Prinzip(vgl. Martin, Robert 2000 S.4-16)




Bei Programmstart wird der Nutzer aufgefordert auszuwählen welchen Anwendungsfall er nutzen möchte. Diese Auswahl wird im Programm durch ein 'if-Statement' ermöglicht. Bei Eingabe der Zahl Eins, wird der zuvor definierte Showcase ausgelöst. Der Nutzer hat entweder die Möglichkeit alle Flurstücke abzurufen oder alternativ eine Anzahl an heruntergeladenen Flurstücken festzulegen, um die Zeit der Transformation zu begrenzen. Der Download endet erst wenn die angegebene Anzahl von Flurstücken erreicht ist oder der Parameter returnedObjects = 0 ist. Zunächst sollen die Datensätze für die 5 Bundesländer heruntergeladen werden.

```
<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>
<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>
```

Listing 12: XML Struktur eines Flurstückes mit der ID: DESNALK05e0000cC aus vereinfachtes-schema.xml

Dies gelingt in dem für jedes Bundesland eine Request URL erstellt wird. Die Basis URL wird einem Dictionary entnommen, indem die ohne Zugriffsbeschränkung verfügbaren WFS Endpunkte der Bundesländer gespeichert sind.

```
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?"
}
```

Listing 13: WFS Dictionary

Ein Dictionary ist eine Ansammlung von Keys und passenden Values. Eine Request URL für das Bundesland 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=1000&STARTINDEX=0
```

Listing 14: Request URL für Sachsen

Diese Request URL enthält neben der Basis URL des Datensatzes, den Request Typ: GetFeature, die Version des WFS: 2.0.0, den Type des abgefragten Objektes: Flurstück, den Code eines Koordinatenreferenzsystems: EPSG:4258, den Namespace, die Anzahl der abgefragten Objekte für diesen Request: 1000 und den Startindex. Da die abgefragten Objekte in einem Aufruf bei einigen Schnittstellen limitiert sind, müssen mehrere Requests an den WFS erfolgen, um an mehr Objekte zu gelangen. Dies gelingt indem man den Request mit dem Parameter Count limitiert und einen Startindex festlegt. Wenn man nun 5000 Flurstück Objekte abfragen möchte ist es nötig 5 Requests zustellen und den Startindex jedesmal um 1000 zu erhöhen. Mithilfe dieser Request URL wird ein HTTP GET Call durchgeführt. 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 als XML Datei mit dem passenden Index im Namen gespeichert. Der Download Vorgang endet wenn der Startindex den festgelegen maximalen Index erreicht oder überschreitet.

```
def executeShowCaseDownload(maxIndex=None):
    listAvailableStates = [
                           "NRW-vereinfacht",
                           "Brandenburg-vereinfacht",
                           "Hamburg-vereinfacht",
                           "Hessen-vereinfacht",
                           "Sachsen-vereinfacht",
                          ]
    Typenames = "&TYPENAMES=ave:Flurstueck"
    KoordinatenType = "&srsName=EPSG:4258"
    Namespaces = "&NAMESPACES=xmlns(ave,http://repository.gdi-de.org/schemas/adv/produkt
        /alkis-vereinfacht/2.0)"
    AnzahlObjekte = "&Count=10000"

    for state in listAvailableStates:
        StartIndex = 0
```

```
        while(True):
            try:
                response = requests.get(WFS_dictionary[
                                        state] + "Service=WFS&REQUEST=GetFeature&
                                            Version=2.0.0" + Typenames +
                                            KoordinatenType + Namespaces +
                                            AnzahlObjekte +
                                    "&STARTINDEX=" + str(StartIndex),
                                        allow_redirects=True)
                # if maxindex is reached end download
                if (maxIndex is not None and StartIndex >= maxIndex) or 'numberReturned
                    ="0"' in response.text:
                    print("Download finished or maxIndex is reached!\n")
                    break
                with open("TestData/" + state[0:3] + "/vereinfachtes-schema" + str(
                    StartIndex) + ".xml", 'wb') as file:
                    file.write(response.content)
                print("Download for " + state + " was successful!" + " with index at: "
                    + str(StartIndex))
                StartIndex += 1000
            except Exception as e:
                print(e)
                break
```

Listing 15: Funktion executeShowCaseDownload

Nun wird für jede XML Datei die Transformation durchgeführt. Die Datensätze werden sequentiell 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 passenden Mapping File nach dem String '- access: ' gesucht 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='')
```

Listing 16: Funktion nameFileToMap

Nun ist zugesichert das die Funktion yarrrmlToRml() ausgeführt werden kann. Die Aufgabe dieser Funktion ist es den yarrrml-parser aufzurufen. Dieser wandelt die bereits definierten yarrrml Mapping Regeln in RML Regeln um. Eine genaue Beschreibung der Mapping Regeln erfolgt im Abschnitt 5.2. Dieser Schritt, also die Umwandlung der Mapping Regeln von yarrrml Regeln in RML 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 RML, 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. 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(maxIndex=None):
    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",
                 "Output/Ham/showcaseFile",
                 "Output/Hes/showcaseFile",
                 "Output/NRW/showcaseFile",
                 "Output/Sac/showcaseFile",
                 ]
    for file in filesList:
        StartIndex = 0
        while True:
            if maxIndex is not None and StartIndex >= maxIndex:
                print("Saving data finished!")
                break
            fileWithIndex = file + str(StartIndex) + ".ttl"
            store.load(fileWithIndex, format="ttl")
            print("the file " + fileWithIndex + " got saved to the fuseki server!")
            StartIndex += 1000
```

Listing 17: Funktion executeShowCaseSave

Nach erfolgreicher Überführung in die Graph Datenbank, stehen die RDF Triple zur weiteren Verarbeitung bereit.

| graph name | triples |
|---|---|
| default graph | 0 |
| http://example.org/alkis_graph | 600961 |

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

```
SELECT $s $p $o
FROM <http://example.org/alkis_graph>
WHERE {
  <http://example.com/DESNALK05e0000cC> $p $o
}
```

Listing 18: SPARQl Query

Der SPARQL Endpunkt antwortet mit dann mit dem erwartetem Ergebnis. Damit 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 Tripel 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 der WFS Produktspezifikation S.17. Das Zielformat für die Transformation ist die im Limbo Projekt erstellte ALKIS Ontologie.[13] Diese beruht wiederum auf den durch den ADV erstellten Objektartenkatalog.[14] In jenem ist exakt beschreiben wie sich die Objekte des AAA- Modells zusammensetzen und welche Relationen sie besitzen. In dem für dieses Projekt verwendeten vereinfachten Schema, sind nicht alle Objekte und Relationen enthalten. In der folgenden Grafik ist zu erkennen welche Objekte der ALKIS Ontologie im vereinfachten Schema auftreten. Das endgültige vereinfachte Zielformat für die Transformation dieses Projektes sieht so aus.

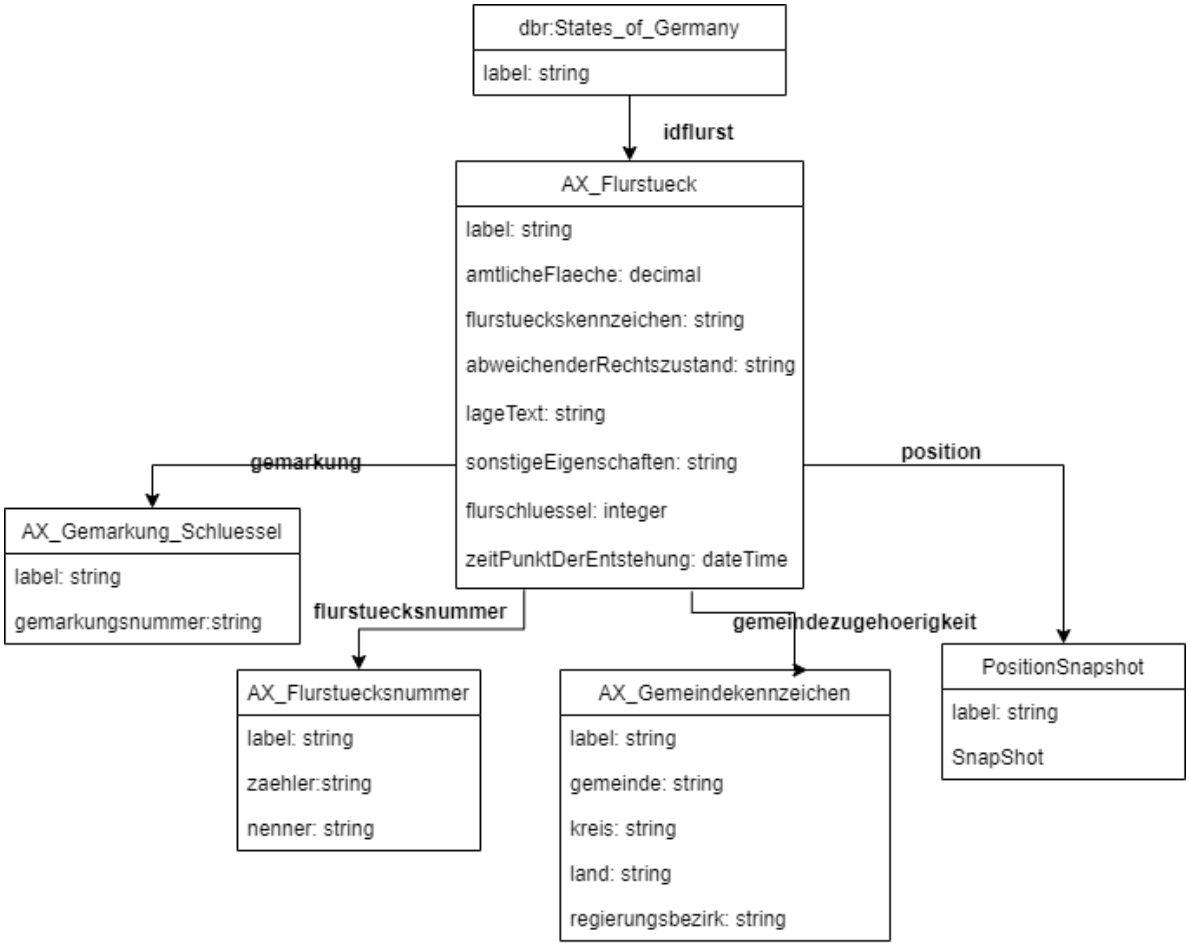



Abbildung 2: Zielformat der Transformation

[13] ALKIS Diagramm: https://gitlab.com/limbo-project/alkis/-/raw/master/diagrams/ALKIS.Diagram.png

[14] Objektartenkatalog: https://www.adv-online.de/icc/extdeu/nav/a63/binarywriterservlet?imgUid=b001016e-7efa-8461-e336-b6951fa2e0c9&uBasVariant=11111111-1111-1111-1111-111111111111

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 Tripel 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/Flurstueck' eingestellt.

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

Listing 19: Auschnitt aus yarrrml Mapping, Definition der Parameter access, referenceFormulation und Iterator

Hier sieht man einen Ausschnitt aus der XML Datei auf die zugegriffen wird. Indem ein Flurstück Objekt mit der ID DESNALK05e0000cC beschrieben wird. 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. (vgl. AdV Geoinfodoc S.92) 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
```

Listing 20: Beispiel aus yarrrml Mapping, Mapping Regeln für das bilden des States of Germany Subjektes

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. 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>
```

Listing 21: Beispiel Triple für Flurstück DESNALK05e0000cC

Außerdem 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
          Liegenschaftskataster festgelegten Grenzlinie umschlossen und mit einer Nummer
           bezeichnet ist. Es ist die Buchungseinheit des Liegenschaftskatasters." ]
      - [ 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 ]
```

Listing 22: yarrrml Mapping Regeln zur Bildung der einfachen Attribute

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 weitere Eigenschaften besitzen. Wie Zum Beispiel die Gemarkung eines Flurstücks.

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

Listing 23: yarrrml Mapping Regeln zur Bildung des Objektes Gemarkung

Diese wird hier mit einer eindeutigen URI definiert. Diese setzt sich aus dem Prefix ex: und der ID des Flurstücks zusammen. Im Wortlaut bedeutet dies, die ID DESNALK05e0000cC hat als Gemarkung den Wert DESNALK05e0000cC_gemarkung. Da das Objekt in diesem Tripel als URI 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]
```

Listing 24: yarrrml Mapping Regeln zur Bildung des Subjektes Gemarkung mit seinen Eigenschaften

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

```
:rules_000 rdf:type void:Dataset ;
        void:exampleResource :map_bundesland_000, :map_alkis_flurstueck_000, :
            map_alkis_gemarkung_000, :map_alkis_flurstuecksnummer_000, :
            map_alkis_gemeindezugehoerigkeit_000, :map_alkis_position_000 .
```

```
: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 .
```

Listing 25: Auschnitt RML-Mapping.ttl

Diese Datei wird schlussendlich vom RMLMapper benutzt um die finale Turtle Datei zu erstellen.

```
dbr:Saxony rdfs:label "Bundesland der BRD";
  dbo:type "https://dbpedia.org/resource/States_of_Germany";
  alkis:AX_Flurstueck ex:DESNALK05e000018, ex:DESNALK05e00001C, ex:DESNALK05e00001E,
    ex:DESNALK05e00001K, ex:DESNALK05e00001Q, ex:DESNALK05e00001l, ex:DESNALK05e00001n,
    ex:DESNALK05e00001r, ex:DESNALK05e00001t, ex:DESNALK05e00001z, ex:DESNALK05e000023,
        ex:DESNALK05e0000cC,
    ...

ex:DESNALK05e0000cC a alkis:AX_Flurstueck;
        rdfs:label 'Ein Flurstueck ist ein Teil der Erdoberfläche, der von einer im
            Liegenschaftskataster festgelegten Grenzlinie umschlossen und mit einer
            Nummer bezeichnet ist. Es ist die Buchungseinheit des Liegenschaftskatasters
            .';
  alkis:Flurstuecksnummer ex:DESNALK05e0000cC_flurstuecksnummer;
  alkis:abweichenderRechtszustand "Kein abweichender Rechtszustand";
  alkis:amtlicheFlaeche 6321.0;
  alkis:flurstueckskennzeichen "146325___00121000100";
  alkis:gemarkung ex:DESNALK05e0000cC_gemarkung;
  alkis:gemeindezugehoerigkeit ex:DESNALK05e0000cC_gemeindezugehoerigkeit;
  alkis:lageText "ohne Lage";
  alkis:position ex:DESNALK05e0000cC_position;
  alkis:sonstigeEigenschaften "Wald;6321";
  alkis:zeitPunktDerEntstehung "2013-11-08Z"^^<http://www.w3.org/2001/XMLSchema#dateTime
      > .

ex:DESNALK05e0000cC_flurstuecksnummer a alkis:AX_Flurstuecksnummer;
  rdfs:label 'Flustücksnummer ist die Nummer die dem Flurstück zugeordnet ist.';
  alkis:zaehler 121 .
```

```
ex:DESNALK05e0000cC_gemeindezugehoerigkeit a alkis:AX_Gemeindekennzeichen;
  rdfs:label ' Die Gemeindezugehörigkeit enthält das Gemeindekennzeichen zur Zuordnung
      der Flurstücksdaten zu einer Gemeinde.';
  alkis:gemeinde "Kreba-Neudorf";
  alkis:land "Freistaat Sachsen";
  alkis:regierungsbezirk "NUTS 2-Region Dresden" .

ex:DESNALK05e0000cC_position a ex:PositionSnapshot;
  ex:Snapshot "51.3278754729631 14.6745851070875 51.3275425205472
      14.6748165442732 51.3281204599786 14.676750975996 51.328539455138 14.6769051793117
       51.3278754729631 14.6745851070875";
  rdfs:label "Beschreibt die Geometrie eines Flurstückes" .
```

Listing 26: .tll für das Flurstück mit der ID: DESNALK05e0000cC , Die Liste der Bundesländer ist nur angedeutet

### 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. Zur Umwandlung der Mapping Regeln im .yml Format in RML wurde der yarrrml parser footnoteyarrrml-parser: `https://github.com/RMLio/yarrrml-parser` benutzt. Zur schlussendlichen Transformation der Daten in das Turtle Format wurde der rmlmapper footnoteRML-Mapper: `https://github.com/RMLio/rmlmapper-java` verwendet. Weiterhin wurde zur Verarbeitung von RDF im Code, die Python Bibliothek rdflib 6.1.1 footnoteRDFLib: `https://github.com/RDFLib/rdflib/#getting-started` genutzt. Als RDF Tripel Store wurde Apache Jena Fuseki verwendet. footnoteFuseki: `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 URI's vorkommen, bekommen die Daten eine Bedeutung, da sie einer Ontologie 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 im RDF Format

können durch SELECT Querys abgefragt oder durch UPDATE beziehungsweise CONSTRUCT Querys verändert oder erweitert werden. 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

Zum jetzigen Zeitpunkt können 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. 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 weiterer 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, zum Beispiel über Gebäude und die Nutzung von Flurstücken. 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. 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 einem performanteren Computer 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. Mit einer parallelen anstatt sequentiellen Ausführung würde man außerdem eine Beschleunigung der Transformation erreichen.

# 8 Quellcode Informationen

Der Quellcode der in diesem Projekt entwickelten Software ist unter der Github Seite `https://github.com/banzaiiiiii/ALKIS-transform` abrufbar. Die Software wurde als OpenSource Projekt erstellt und kann nach Absprache weiterentwickelt werden. Eine Anleitung zur Nutzung ist im readme Bereich der Github Seite hinterlegt.

# 9 Quellenverzeichnis

## 9.1 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-hb`
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

## 9.2 Literaturverzeichnis

ADV (2022): Die Liegenschaftskataster - was ist das? https://www.adv-online.de/AdV-Produkte/Liegenschaftskataster/ALKIS/broker.jsp?uMen=e5670f15-8e71-3c01-e1f3-351ec0023010 Abruf Datum: 5.5.2022

ADV (2022): Amtliches Liegenschaftskatasterinformationssystem https://www.adv-online.de/AdV-Produkte/Liegenschaftskataster/ALKIS/ Abruf Datum: 5.5.2022

AdV (2016): WFS Produktspezifikation Version 2.0 https://www.adv-online.de/AdV-Produkte/Standards-und-Produktblaetter/Standards-des-Liegenschaftskatasters/binarywriterservlet?imgUid=f49502a0-36fa-6b61-c2d2-1bf43b36c4c2&uBasVariant=11111111-1111-1111-1111-111111111111 Abruf Datum: 1.4.2022

ALKIS Objektartenkatalog https://www.adv-online.de/icc/extdeu/nav/a63/binarywriterservlet?imgUid=b001016e-7efa-8461-e336-b6951fa2e0c9&uBasVariant=11111111-1111-1111-1111-111111111111 Abruf Datum: 1.4.2022

ALKIS GeoInfoDok https://www.adv-online.de/GeoInfoDok/binarywriterservle
imgUid=e9e304b7-96d3-de71-d261-dde30c21d06f&uBasVariant=11111111-1111-1111-1111-111111111111 Abruf Datum: 1.4.2022

Apache Jena Fuseki (2022): Apache Jena Fuseki Dokumentation https://jena.apache.org/documentation/fuseki2/ Abruf Datum: 6.5.2022

Bayerische Vermessungsverwaltung: Was sind Geodatendienste https://geodatenonline.bayern.de/geodatenonline/seiten/wms_webdienste Abrufdatum: 1.5.2022

Ben De Meester, Dylan Van Assche, Pieter Heyvaert (08/2021): Yarrrml Specification https://rml.io/yarrrml/spec/ Abrufdatum: 5.5.2022

Ben de Meester, Pieter Heyvaert, Thomas Delva (2020): RDF Mapping Language(RML) https://rml.io/specs/rml/ Abrufdatum: 5.5.2022

Berners-Lee, Tim (1998): DesignIssues https://www.w3.org/DesignIssues/RDF-XML.html Abrufdatum: 7.5.2022

Berners-Lee, Tim (2000): The Semantic Web https://www.w3.org/2000/Talks/0906-xmlweb-tbl/text.htm Abrufdatum: 7.5.2022

Berners-Lee, Tim (07/2006): Linked Data https://www.w3.org/DesignIssues/LinkedData.html Abrufdatum: 7.5.2022

Bezirksregierung Köln (07/2020): Anleitung zur Nutzung eines Web Feature Service(WFS) https://www.bezreg-koeln.nrw.de/brk_internet/geobasis/webdienste/anleitung_wfs.pdf Abrufdatum: 2.5.2022

Businessinsider(01/2019): Prototyping https://www.businessinsider.de/gruenderszen
lexikon/begriffe/prototyping/ Abrufdatum: 8.5.2022

GeoSN (06/2021): Anwendungsbeschreibung des ALKIS-WFS(vereinfachtes Schema) https://geoportal.sachsen.de/portal/dokumente/WFS_ALKIS_Anwendungsbes
pdf Abrufdatum: 1.2.2022

Gräbe, Hans-Gert :Semantic-aware Fingerprints of Symbolic Research Data https://symbolicdata.github.io//Papers/icms-16.pdf Abrufdatum: 27.4.2022

Köllinger, Phillipp (2003): Internetnutzung in Deutschland: nach Boom nun langsamerer Anstieg erwartet, DIW Wochenbericht https://www.econstor.eu/bitstream/10419/151237/1/03-30-1.pdf Abrufdatum: 1.2.2022

Kück, G (2004): Tim Berners-Lee's Semantic Web https://www.researchgate.net/publication/307845029_Tim_Berners-Lee's_Semantic_Web Abrufdatum: 13.3.2022

Limbo Projekt: Das war LIMBO! https://www.limbo-project.org/ Abrufdatum: 2.3.2022

LNISA,volume 12072 (06/2020): Chapter 4 Creation of Knowledge Graphs https://link.springer.com/chapter/10.1007/978-3-030-53199-7_4 Abrufda-

tum: 29.3.2022

Luber, Stefan/Litzel, Nico (11/2018): Was ist ETL? https://www.bigdata-insider.de/was-ist-etl-extract-transform-load-a-776549/ Abrufdatum: 7.4.2022

Martin, Robert C. (2000): Design Principles and Design Patterns Seite 4- 16 http://staff.cs.utu.fi/staff/jouni.smed/doos_06/material/DesignPrinciplesAnd
pdf Abrufdatum: 5.5.2022

Meinel, Christoph (03/2021): Die Vision des intelligenten Webs https://www.spektrum.de/kolumne/semantic-web-wie-das-internet-inhalte-verstehen-koennte/1841389 Abrufdatum: 29.3.2022

Niedersächsische Vermessungs- und Katasterverwaltung (2010): Basiswissen Alkishttps://www.lgln.niedersachsen.de/download/126790/Basiswissen_ALKIS_ETRS89_Schulungsmaterial_Stand_12.04.2010.pdf Abrufdatum: 1.2.2022

Ontotext What is an RDF Triplestore https://www.ontotext.com/knowledgehub/fundamentals/what-is-rdf-triplestore/ Abrufdatum: 29.3.2022

Retresco: Ontologien in der Informatik https://www.retresco.de/lexikon/ontologie/ Abrufdatum: 8.5.2022

Statistisches Bundesamt (2018): 90% der Bevölkerung in Deutschland sind online https://www.destatis.de/DE/Presse/Pressemitteilungen/2018/09/PD18_330_634.html Abrufdatum: 1.2.2022

W3C (02/2004): Resource Description Framework(RDF):Concepts and Abstract Syntax https://www.w3.org/TR/rdf-concepts/ 29.11.2021

W3C (02/1999): Resource Description Framework(RDF) Model and Syntax Specification https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ 29.11.2021

W3C (03/2013: SPARQL 1.1 Overview https://www.w3.org/TR/sparql11-overview/ 7.1.2022

# 10 Erklärung zur Selbstständigkeit

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann. Ich versichere, dass das elektronische Exemplar mit den gedruckten Exemplaren übereinstimmt.

Ort: Datum: Unterschrift: