Bericht zum Projekt

Gentrifizierungsprozesse in Leipzig

Uni Leipzig, Sommersemester 2014

Marius Brunnert

22. September 2014

Inhaltsverzeichnis

1 Einleitung

Im Projekt „Gentrifizierungsprozesse in Leipzig“, das im Rahmen des Moduls „Kreativität und Technik“ unter Leitung von Prof. Dr. Gräbe und in Zusammenarbeit mit den Verantworlichen von `einundleipzig.de` (eine Webseite, die sich mit diesem Thema auseinander setzt) statt gefunden hat, haben wir uns in einer Gruppe von sechs Kursteilnehmern mit den Gentrifizierungsprozessen in Leipzig befasst und uns zum Ziel gesetzt, eine interaktive Karte zu erstellen, auf der diese visualisiert werden. Beim ersten Treffen haben wir über unsere Vorkenntnisse und Erwartungen gesprochen, Ideen gesammelt und schließlich das grundlegende Vorgehen geklärt. Um das Ziel zu erreichen, haben wir uns an der Scrum-Methodik aus der Informatik orientiert, allerdings „weekly“ statt „daily scrums“ geplant, und das Projekt in drei Sprints gegliedert.

Ziel des ersten Sprints war zum einen, zu analysieren, welche Informationen für unser Thema relevant sein könnten und mögliche Quellen zusammenzutragen, die entsprechende Daten liefern können. Zum anderen sollten die Teammitglieder sich kennen lernen, so dass im zweiten Sprint jeder eine konkrete Aufgabe, die zu seinen Qualifikationen passt, übernehmen und ein funktionierender Workflow etabliert werden konnte.

Im zweiten Sprint sollten dann die relevanten Daten aus den entsprechenden Quellen gesammelt, in ein einheitliches Format gebracht und ins OntoWiki transferiert werden. Außerdem sollte ein erster Prototyp der Karte erstellt werden.

Der dritte und letzte Sprint hatte schließlich das Ziel, das zuvor Erarbeitete zu einem „runden“ Gesamtergebnis zusammenzufassen, also die Daten, die sich besonders für eine grafische Darstellung in einer Karte eignen, auf dieser darzustellen.

Nach Abschluss des Projekts werden die Ergebnisse im Rahmen von `einundleipzig.de` weiter genutzt. Außerdem habe ich mit den Daten noch weitergearbeitet – siehe Abschnitt 5.

2 Meine Aufgaben

2.1 Erster Sprint

Im ersten Sprint habe ich Kontakt mit Roman Grabolle und dem HausHalten e.V. aufgenommen, um Informationen über Hausprojekte und alternative Strukturen in Leipzig zu sammeln. Nach einigen kleineren Hürden habe ich auch von beiden Rückmeldung bekommen und eine erste Übersicht über die verfügbaren Informationen in der Gruppe vorgestellt. Da bei diesen Daten allerdings nie ein Anspruch auf Vollständigkeit erhoben werden kann und die verschiedenen Projekte in ihrer Bedeutung für die Gentrifizierung in Leipzig recht divergent sind, haben wir uns letztlich entschlossen, diese Daten im Rahmen des Projekts nicht zu verwenden.

2.2 Zweiter und dritter Sprint

Im zweiten und dritten Sprint war ich zum einen die „Schnittstelle“ zwischen den Rohdaten und der Karte, das heißt, ich habe die Daten gesichtet, geordnet, miteinander verknüpft (z.B. aus den Wohnungen pro Stadtteil und der Größe des Stadtteils die Wohnungsdichte berechnet), sie ins .csv-Format gebracht, falls sie nicht bereits in diesem vorlagen, und die

.csv-Dateien schließlich über das Repository zur Verfügung gestellt. Zum anderen habe ich im OntoWiki entsprechende Ontologien erstellt und die Daten mit Hilfe von .ttl-Dateien in das OntoWiki transferiert.

2.3 Nach dem Projekt

Nach Abschluss des eigentlichen Projekts habe ich mich noch mit der Frage beschäftigt, wie man die Daten des Leipzig-Informationssystems (LIS), die wir zum Teil auch für unser Projekt genutzt haben, als *RDF Data Cube* darstellen kann, und schließlich einen solchen Data Cube mit Beispieldaten erstellt.

3 Darstellung der Daten im RDF

3.1 Allgemeine Form

Die grundlegende Form unserer .ttl-Dateien sieht wie folgt aus:

```
<prefixes>
<ldot:<ortsteil> gm:<attribut><yy> "<wert>"^^xsd:<datatype> .>*
```

Die `<prefixes>` sind in jeder .ttl-Datei identisch und definieren Namensraum-Präfixe:

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix gm: <http://leipzig-data.de/Data/gentri-14/model/> .
@prefix ldot: <http://leipzig-data.de/Data/Ortsteil/> .
```

Für `<ortsteil>` wird der entsprechende Leipziger Ortsteil – ohne Umlaute – eingetragen. Zum Beispiel „Zentrum-Ost“, „Schleussig“, „Gruenau-Nord“ und so weiter.

Für `<attribut>` wird die gemessene Beobachtung benannt. Zum Beispiel „Alter“, „Miete“, „IsAbwanderung“ und so weiter.

Für `<yy>` wird die auf zwei Ziffern abgekürzte Jahreszahl eingetragen. Zum Beispiel „00“, „01“ und so weiter.

Für `<wert>` wird der beobachtete Wert angegeben, wobei Nachkommastellen durch einen Punkt abgetrennt werden. Zum Beispiel „40“, „6.23“, und so weiter.

Für `<datatype>` wird, je nach Beobachtung, entweder `decimal` (mit Nachkommastellen) oder `integer` (ohne Nachkommastellen) gewählt.

Dieser Teil kann theoretisch beliebig oft mit unterschiedlich gewählten Attributen vorkommen. In der Praxis sind es allerding $63 \cdot y$ Zeilen, wobei 63 die Anzahl der Leipziger Ortsteile ist und y für die Anzahl der Jahre steht, für die Daten vorliegen.

3.2 Ein konkretes Beispiel

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix gm: <http://leipzig-data.de/Data/gentri-14/model/> .
@prefix ldot: <http://leipzig-data.de/Data/Ortsteil/> .

ldot:Zentrum gm:alter00 "49.1"^^xsd:decimal .
ldot:Zentrum-Ost gm:alter00 "45.2"^^xsd:decimal .
ldot:Zentrum-Suedost gm:alter00 "45.1"^^xsd:decimal .
ldot:Zentrum-Sued gm:alter00 "45.5"^^xsd:decimal .
usw.
```

4 Die Matrix

4.1 Allgemeines

Da wir unsere Karte nach Ortsteilen gegliedert haben, einige uns vorliegende Daten (Informationen zu Wohnorten Leipziger Studenten) allerdings nach Postleitzahlen gruppiert waren, war eine Umrechnung nötig, für die ich eine Matrix erstellt habe. Leider kann mit dieser Matrix nur eine näherungsweise Umrechnung erfolgen, da es mehrere Ungenauigkeiten gibt:

1. Wenn ein Postleitzahlbereich mehrere Ortsteile (ganz oder zum Teil) abdeckt, was relativ häufig vorkommt, gibt es keine konkreten Informationen darüber, wie viele Einwohner, die unter dieser Postleitzahl gemeldet sind, in Ortsteil A und wie viele in Ortsteil B (und ggf. in Ortsteil C) leben. Hier kann die Umrechnung also über einen zu bestimmenden *Anteilsfaktor* nur näherungsweise erfolgen.

2. Zur Bestimmung eines solchen Anteilsfaktors bin ich wie folgt vorgegangen: Grundlage waren zwei Karten[1] (Gliederung der Stadt nach PLZ bzw. nach Ortsteilen), aus denen die Zuordnung von Ortsteilen zu Postleitzahlbereichen abgelesen bzw. ein flächenmäßiger Anteil geschätzt wurde. Diese Zahlen wurden mit den Bevölkerungszahlen der Stadtteile (Stand 2013) multipliziert und daraus die Anteilsfaktoren bestimmt.

Das Ergebnis ist eine Matrix mit 34 Zeilen (entspricht den Postleitzahlbereichen) und 63 Spalten (entspricht den Ortsteilen). In jeder Position dieser Matrix ist angegeben, mit welchem Anteilsfaktor der Wert aus einem Postleitzahlbereich Y zu multiplizieren ist, um den Anteil dieses Werts zu bestimmen, der auf den Stadtteil X entfällt. Da sich ein Postleitzahlbereich über maximal drei Ortsteile erstreckt, enthalten die meisten Positionen eine null. Nach Konstruktion summieren die Anteilsfaktoren jeder Zeile zu eins.

4.2 Beispielrechnungen

1. Die Postleitzahl 04103 deckt die Ortsteile 01 und 02 zu jeweils 100% ab. Ortsteil 01 hatte (im Jahr 2013) 3 980 Einwohner, Ortsteil 02 hatte 11 515 Einwohner. Dies ergibt eine Gesamtzahl von 15 495 Einwohnern, die unter der Postleitzahl 04103 zusammengefasst

[1] `http://www.infos-sachsen.de/Orte/Stadt-L.php`

werden. Daran hat Ortsteil 01 einen Anteil von $\frac{3980}{15495} \approx 25.7\%$ und Ortsteil 02 einen Anteil von $\frac{11515}{15495} \approx 74,3\%$. Die Anteilsfaktoren sind also 0.257 bzw. 0.743.

2. Die Postleitzahl 04159 deckt die Ortsteile 80 und 81 zu jeweils 100%, sowie den Ortsteil 82 zu etwa 50% ab. Ortsteil 80 hatte 13 172 Einwohner, Ortsteil 81 hatte 6 536 Einwohner, Ortsteil 82 hatte 3 945 Einwohner – da dieser allerdings nur zur Hälfte in der Postleitzahl 04159 enthalten ist, rechnen wir auch nur mit der halben Einwohnerzahl: 1 972.5. Insgesamt deckt die Postleitzahl somit 21 680.5 Einwohner ab. Daran hat Ortsteil 80 einen Anteil von $\frac{13172}{21680.5} \approx 60.8\%$, Ortsteil 81 einen Anteil von $\frac{6536}{21680.5} \approx 30.1\%$ und Ortsteil 82 einen Anteil von $\frac{6536}{21680.5} \approx 9.1\%$. Die Anteilsfaktoren sind also 0.608, 0.301 und 0.091.

5 Darstellung der Daten als RDF Data Cube

5.1 Die Prefixes

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix sdmx: <http://purl.org/linked-data/sdmx#> .
@prefix sdmx-concept: <http://purl.org/linked-data/sdmx/2009/concept#> .
@prefix sdmx-dimension: <http://purl.org/linked-data/sdmx/2009/dimension#> .
@prefix sdmx-attribute: <http://purl.org/linked-data/sdmx/2009/attribute#> .
@prefix sdmx-measure: <http://purl.org/linked-data/sdmx/2009/measure#> .
@prefix gcube: <http://leipzig-data.de/Data/gentri-14/cube/> .
@prefix ldot: <http://leipzig-data.de/Data/Ortsteil/> .
```

5.2 Das DataSet

```
gcube:GentriLeipzig a qb:DataSet ;
  rdfs:label "Gentrifizierung in Leipzig" ;
  dct:source <http://statistik.leipzig.de/statdist/index.aspx> ;
  qb:structure gcube:GentriDSD .
```

5.3 Die DataStructureDefinition

Allgemein

```
gcube:GentriDSD a qb:DataStructureDefinition ;
  qb:component
     [ qb:dimension gcube:district; ],
     [ qb:dimension gcube:year; ],
     [ qb:dimension qb:measureType; ],
```

```
    [ qb:attribute sdmx-attribute:unitMeasure; ],
    [ qb:measure gcube:<measure>; ]* .
```

Für `[qb:measure gcube:<measure>; ]*` müssen die verwendeten `Measure`s eingesetzt werden. Es muss mindestens eine `Measure` geben.

Beispiel

```
    [ qb:measure gcube:extImmigrants; ],
    [ qb:measure gcube:meanRent; ],
 usw.
```

5.4 Die Dimension(Propertie)s

```
gcube:district a rdf:Property, qb:DimensionProperty ;
  rdfs:label "Ortsteil"@de ;
  rdfs:subPropertyOf sdmx-dimension:refArea ;
  qb:concept sdmx-concept:refArea .

gcube:year a rdf:Property, qb:DimensionProperty ;
  rdfs:label "Jahr"@de ;
  rdfs:subPropertyOf sdmx-dimension:refPeriod ;
  qb:concept sdmx-concept:refPeriod .
```

5.5 Die Measure(Propertie)s

Allgemein

```
gcube:<measure> a rdf:Property, qb:MeasureProperty ;
  rdfs:label "<measureLabel>"@de ;
  rdfs:subPropertyOf sdmx-measure:obsValue ;
  dct:source <source> ;
  rdfs:range xsd:<range> .
```

Für `<measure>` wird der Name der `MeasureProperty` angegeben, für `<measureLabel>` eine genauere Bezeichnung (auf Deutsch), für `<source>` ein Verweis auf die Quelle (in den folgenden Beispielen weggelassen), aus der die Daten stammen, und für `<range>` der erlaubte Wertebereich (hier: `integer` oder `decimal`).

Beispiel

```
gcube:extEmigrants a rdf:Property, qb:MeasureProperty ;
  rdfs:label "Wegzüge nach Außerhalb"@de ;
  rdfs:subPropertyOf sdmx-measure:obsValue ;
  rdfs:range xsd:integer .
```

```
gcube:meanAge a rdf:Property, qb:MeasureProperty ;
  rdfs:label "Durchschnittsalter"@de ;
  rdfs:subPropertyOf sdmx-measure:obsValue ;
  rdfs:range xsd:decimal .
```

5.6 Die Attribute(Propertie)s

Allgemein

```
gcube:<attribute> a rdf:Property, qb:AttributeProperty ;
  rdfs:label "<attributeLabel>"@de .
```

Für `<attribute>` wird der Name der `AttributeProperty` angegeben, für `<attributeLabel>` eine genauere Bezeichnung (auf Deutsch).

Beispiel

```
gcube:people a rdf:Property, qb:AttributeProperty ;
  rdfs:label "Menschen"@de .

gcube:euro a rdf:Property, qb:AttributeProperty ;
  rdfs:label "Euro"@de .
```

5.7 Die Observations

Allgemein

```
gcube:O<year>-<dnum>-GR<ln> gcube:<measure> "<amount>"^^xsd:<datatype> ;
  qb:measureType gcube:<measure> ;
  gcube:district ldot:<dname> ;
  gcube:year "<year>"^^xsd:gYear ;
  sdmx-attribute:unitMeasure gcube:<attribute> ;
  a qb:Observation ;
  qb:dataset gcube:GentriLeipzig .
```

Für `<year>` wird das Jahr, auf das sich die Daten der Observation beziehen, angegeben, für `<dname>` der Name (ohne Umlaute, „ss“ statt „ß“, „ae“ statt „ä“ usw.) des entsprechenden Ortsteils, für `<ln>` eine laufende Nummer über den unterschiedlichen Arten von `Observations` bzw. `MeasureProperties` (`extimmigrants` hat Nummer 0, `extemigrants` die Nummer 1, `locimmigrants` die Nummer 2, usw. – eine genaue Auflistung ist im Abschnitt 5.8) zu finden, für `<measure>` der Name der entsprechenden `MeasureProperty` (also die Art der `Observation`), für `<amount>` der beobachtete Wert, für `<datatype>` der erlaubte Datentyp (hier entweder `integer` oder `decimal`) und für `<attribute>` die entsprechende `AttributeProperty` (also die Angabe, in welcher Einheit die Observation „gezählt“ wird, z.B. „people“, „years“, oder „euro“).

Beispiel

```
gcube:O2000-00-GR0 gcube:extImmigrants "208"^^xsd:integer ;
  qb:measureType gcube:extImmigrants ;
  gcube:district gcube:district00 ;
  gcube:year "2000"^^xsd:gYear ;
  sdmx-attribute:unitMeasure gcube:people ;
  a qb:Observation ;
  qb:dataset gcube:GentriLeipzig .

gcube:O2000-01-GR0 gcube:extImmigrants "274"^^xsd:integer ;
  qb:measureType gcube:extImmigrants ;
  gcube:district gcube:district01 ;
  gcube:year "2000"^^xsd:gYear ;
  sdmx-attribute:unitMeasure gcube:people ;
  a qb:Observation ;
  qb:dataset gcube:GentriLeipzig .
```

5.8 Bisher vorhandene Daten

GR0: Zuzüge in Ortsteil x von außerhalb Leipzigs
(`qb:measureType gcube:extImmigrants`)

GR1: Wegzüge aus Ortsteil x nach außerhalb Leipzigs
(`qb:measureType gcube:extEmigrants`)

GR2: Zuzüge in Ortsteil x aus einem anderen Ortsteil
(`qb:measureType gcube:locImmigrants`)

GR3: Wegzüge aus Ortsteil x in einen anderen Ortsteil
(`qb:measureType gcube:locEmigrants`)

GR4: Durchschnittsalter in Ortsteil x (`qb:measureType gcube:meanAge`)

GR5: Studentenanteil in Ortsteil x (`qb:measureType gcube:studPercent`)

GR6: Durchschnittsmiete in Ortsteil x (`qb:measureType gcube:meanRent`)

GR7: Wohnungsdichte in Ortsteil x (`qb:measureType gcube:aptDensity`)

GR8: Arbeitslose in Ortsteil x (`qb:measureType gcube:unemployeds`)

5.9 Probleme

Language-Tags: Wenn die .ttl-Dateien Language-Tags (`@de`) enthalten, kommt es im OntoWiki aus bisher nicht nachvollziehbaren Gründen zu Fehlermeldungen. Ein Beispiel hierfür ist in der Datei `SPARQL_ERROR.txt` zusammengestellt.

qb-components: Wenn man die `qb-components` als Blank Nodes definiert, gehen beim Transfer ins OntoWiki die Meta- und Kontextinformationen der Komponenten verloren. Um das zu vermeiden, wurde auch eine Variante ohne Blank Nodes erstellt. (Siehe Abschnitt 7.1)

6 Weitere Schritte

6.1 Automatisierung der Umwandlung von .csv zu .ttl

Allgemeines

Da die Daten im LIS recht einheitlich im .csv-Format vorliegen, erscheint es sinnvoll, die Übertragung der weiteren LIS-Daten in den Data-Cube so weit wie möglich zu automatisieren. Ausgehend von einer entsprechenden .csv-Datei lässt sich eine weitgehend automatische Umwandlung in eine „Ergänzungs-.ttl-Datei“ in folgenden 8 Teilschritten umsetzen:

1. Erstellen der Prefixes. Die Prefixes sind (in diesem Cube) immer identisch, müssen aber in jeder RDF-Datei, wie unter 5.1 beschrieben, angegeben werden.
2. Am `DataSet` ändert sich nichts, dieses muss also bei Ergänzungen nicht noch einmal aufgeführt werden.
3. In der `DataStructureDefinition muss` die neue `Measure` mit Namen als solche definiert und als `qb:component` angeben werden.
4. An den `DimensionProperties` ändert sich nichts, diese müssen also bei Ergänzungen nicht noch einmal aufgeführt werden.
5. Die neue `Measure` muss, wie unter 5.5 angegeben, als `MeasureProperty` definiert werden. Die `<source>`-Angabe kann dabei nicht direkt aus der .csv-Datei bestimmt werden.
6. Entweder muss der neuen `Measure` eine passende `AttributeProperty` zugeordnet werden (in Schritt 8) oder, wie unter 5.6 angegeben, eine neue erstellt werden. Dieser Schritt kann nicht voll automatisch erfolgen.
7. An den `Districts` ändert sich nichts, diese müssen also bei Ergänzungen nicht noch einmal aufgeführt werden.
8. Für jede Zelle in der .csv-Datei muss eine `Observation`, wie unter 5.7 angegeben, erstellt werden.

Daten aktualisieren

Um aktuellere Versionen der Daten einzupflegen, zu denen es bereits `Measure`s gibt, muss man lediglich neue `Observation`s mit den passenden Parametern erstellen.

Besonderheiten

In einigen .csv-Dateien des LIS fehlen Angaben für einige Ortsteile, bzw. sind z. T. die Angaben für mehrere Ortsteile in einem Ortsteil zusammengefasst. In der .csv-Datei tauchen in den fehlenden oder zusammengefassten Zellen nur ”.”-Zeichen auf. (Zum Beispiel werden beim Wohnungsbestand vor 2010 die Ortsteile 27, 28 und 29, sowie die Ortsteile 54 und 55 zusammengefasst.) Sollten solche Zeichen in der .csv-Datei vorkommen, ist es also wieder nötig, manuell einzugreifen und direkt ins LIS zu schauen, wo durch Annotationen darauf

hingewiesen wird, welcher Zusammenhang konkret besteht. Fehlende Informationen für einzelne Zellen habe ich bisher mit dem Wert „-1“ gekennzeichnet. Sollte in anderen Statistiken auch mit negativen Werten gearbeitet werden, müsste man da nach Alternativen suchen.

6.2 Beispiel-Turtle-Datei zur Erweiterung des bestehenden Cubes

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix sdmx: <http://purl.org/linked-data/sdmx#> .
@prefix sdmx-concept: <http://purl.org/linked-data/sdmx/2009/concept#> .
@prefix sdmx-dimension: <http://purl.org/linked-data/sdmx/2009/dimension#> .
@prefix sdmx-attribute: <http://purl.org/linked-data/sdmx/2009/attribute#> .
@prefix sdmx-measure: <http://purl.org/linked-data/sdmx/2009/measure#> .
@prefix gcube: <http://leipzig-data.de/Data/gentri-14/cube/> .

gcube:DataSetDef qb:component [ qb:measure gcube:aptDensity ] .

gcube:aptDensity a rdf:Property, qb:MeasureProperty ;
  rdfs:label "Wohnungsdichte"@de ;
  rdfs:subPropertyOf sdmx-measure:obsValue ;
  rdfs:range xsd:integer .

gcube:aptsPerKm a rdf:Property, qb:AttributeProperty ;
  rdfs:label "Wohnungen pro Quadratkilometer"@de .

gcube:O2000-00-GR2 gcube:aptDensity "2209"^^xsd:integer ;
  qb:measureType gcube:aptDensity ;
  gcube:district gcube:district00 ;
  gcube:year "2000"^^xsd:gYear ;
  sdmx-attribute:unitMeasure gcube:aptsPerKm ;
  a qb:Observation ;
  qb:dataset gcube:GentriLeipzig .

gcube:O2000-01-GR2 gcube:aptDensity "1462"^^xsd:integer ;
  qb:measureType gcube:aptDensity ;
  gcube:district gcube:district01 ;
  gcube:year "2000"^^xsd:gYear ;
  sdmx-attribute:unitMeasure gcube:aptsPerKm ;
  a qb:Observation ;
  qb:dataset gcube:GentriLeipzig .

usw.
```

7 Anhang

7.1 Cube-Variante ohne Blank Nodes

Der Cube

Am `Cube` selbst muss lediglich in der `DataStructureDefinition` geändert werden, wie die Komponenten angegeben werden. Statt die Komponenten wie unter 5.3 angegeben als Blank Nodes zu definieren, wird jeder benötigten Komponente eine konkrete URI zugewiesen, die dann separat definiert werden muss.

```
gcube:datasetdef a qb:DataStructureDefinition ;
  qb:component
    gcube:districtDim, gcube:yearDim, gcube:measureDim,
    gcube:attributeComp, gcube:measureExtImm, gcube:measureExtEm, <...> .

gcube:districtDim qb:dimension gcube:district .
gcube:yearDim qb:dimension gcube:year .
gcube:measureDim qb:dimension qb:measureType .
gcube:attributeComp qb:attribute sdmx-attribute:unitMeasure .
gcube:measureExtImm qb:measure gcube:extImmigrants .
gcube:measureExtEm qb:measure gcube:extEmigrants . <...> .
```

Für `<...>` würden weitere Measures benannt und definiert werden.

Ansonsten ändert sich am Cube nichts.

Die Erweiterung

Auch bei der Erweiterung des Cubes muss das selbstverständlich entsprechend beachtet werden. In Schritt 3 der Umwandlung (6.1) muss die neue `Measure` zunächst in der `DataStructureDefinition` mit einer URI benannt und dann separat definiert werden.

```
gcube:datasetdef qb:component gcube:measure_aptdens .
gcube:measure_aptdens qb:measure gcube:aptdensity .
```

7.2 Dateien

RDF-Daten für die GentriMap

Verzeichnis `SimpleRDF`:

- `alter00_13.ttl` – Durchschnittsalter in den Ortsteilen von 2000 bis 2013.
- `anteilarbeitslose00_13.ttl` – Arbeitslosenanteil in den Ortsteilen von 2000 bis 2013
- `aszuwanderung00_13.ttl` – Zuwanderung von außerhalb Leipzig in die Ortsteile von 2000 bis 2013
- `isabwanderung00_13.ttl` – Abwanderung von den Ortsteilen in einen anderen Ortsteil von 2000 bis 2013

- `iszuwanderung00_13.ttl` – Zuwanderung in die Ortsteile aus einem anderen Ortsteil von 2000 bis 2013
- `kinderarmut07+09+11.ttl` – Prozentuale Kinderarmut in den Ortsteilen in 2007, 2009 und 2011 (Nicht exakt!)
- `studentenanteilot13.ttl` – Studentenanteil in den Ortsteilen in 2013[2]
- `studentenanteilst13.ttl` – prozentualer Anteil an Leipziger Studenten in den Ortsteilen in 2013[3]
- `wohnungsdichte00_11.ttl` – Wohnungen pro km^2 in den Ortsteilen von 2000 bis 2011

Der RDF Data Cube

Diese Daten sind im Verzeichnis `CUBE` zu finden.
DataSet, DataStructureDefinition, Dimension-, Measure- und AttributeProperties:

- `cubeBaseBlank.ttl` – mit BlankNodes, ohne LanguageTags
- `cubeBaseBlankLT.ttl` – mit BlankNodes, mit LanguageTags
- `cubeBaseNoBlank.ttl` – ohne BlankNodes, ohne LanguageTags
- `cubeBaseNoBlankLT.ttl` – ohne BlankNodes, mit LanguageTags

Die 63 Leipziger Ortsteile als `gcube:district`:

- `districts.ttl` – ohne LanguageTags
- `districtsLT.ttl` – mit LanguageTags

Die Observations:

- `observations/obsaszz.ttl` – GR0
- `observations/obsaswz.ttl` – GR1
- `observations/obsiszz.ttl` – GR2
- `observations/obsiswz.ttl` – GR3
- `observations/obsage.ttl` – GR4
- `observations/obsstuda.ttl` – GR5
- `observations/obsrent.ttl` – GR6
- `observations/obsapt.ttl` – GR7
- `observations/obsunemp.ttl` – GR8

Sonstiges

- `/matrixPLZtoORTSTEIL.csv` – Die unter 4 beschriebene Matrix mit einer zusätzlichen Spalte (Angabe der PLZ) und einer zusätzlichen Zeile (Angabe des Ortsteils).
- `/SPARQL_ Error.txt` – Ein Beispiel für die im Abschnitt 5.9 genannten SPARQL-Fehlermeldungen.
- `/Projektbericht.tex` – Dieser Projektbericht als LaTeX-Datei.

[2] Wie viel Prozent der Einwohner des Ortsteils sind Studenten?
[3] Wie viel Prozent der Studenten Leipzigs leben im Ortsteil?