Towards Sustainable view-based Extract-Transform-Load (ETL) Fusion of Open Data

Kay Müller, Claus Stadler, Ritesh Kumar Singh, Sebastian Hellmann

AKSW/KILT, Leipzig University & DBpedia Association
{kay.mueller,cstadler,ritesh.kumar.singh,hellmann}@informatik.uni-leipzig.de
http://aksw.org/Groups/KILT.html

Abstract. Openly available datasets originate from different data providers which range from government agencies, over commercial enterprises to communities of data enthusiasts. Integrating different source datasets into a single RDF graph by using ETL (Extract-Transform-Load) systems which perform offline transformation, ontology matching and linking techniques usually takes many iterations of revisions until the target dataset is made free of the most obvious mapping, linking and consistency errors. Since ETL systems produce the RDF offline, any mapping or content change requires a re-ingest of the relevant source data. When dealing with heterogeneous source datasets, creating a unified target dataset can be a tedious undertaking. Therefore the paper proposes an RDF view based ingestion approach, which allows real-time "debugging" of the unified dataset where mappings and links can be changed with immediate effect. Once the unified graph passes all data quality tests, the RDF can be materialized. This process poses an alternative to existing ETL solutions.

Keywords: Linked Data, ETL, RDF View

1 Introduction and Related Work

Traditionally, ETL (Extract-Transform-Load) systems are used to convert different source datasets into a structured target dataset. These systems are commonly used in the area of data warehousing. As the name suggests these frameworks follow the workflow of: 1.) *Extract* required information from source datasets, 2.) *Transform* portions of the source data into a target model using schema mappings, normalizations and deduplications, 3.) *Load* the data into a store, possibly with versioning support.

In the Semantic Web community, tools such as LDIF[1], OpenRefine[2] and Unified Views[3] have been developed to allow Linked Data developers to create new

[1] http://ldif.wbsg.de/
[2] http://openrefine.org/
[3] https://www.semantic-web.at/unifiedviews

Linked Data content. Since many Linked Data data publishers do not always follow existing best practices and (quasi-)standards of the Linked Data community, it potentially complicates linking tasks and decreases the overall data quality of source datasets. The LOD Laundromat [1] was created due to this dilemma: This system crawls the Linked Data Cloud[4] and creates corresponding standard-compliant datasets which can be downloaded from the website. Since data quality can vary between datasets, using an ETL system to convert the source data into the target format might result in mapping and content errors, which might only be discovered once all the data is converted. Very often this might result in a re-ingest of the source data through the whole ETL pipeline. These trial and error attempts can be very time and resource consuming. When working on integrating different company datasets into a knowledge graph using the standard ETL approach, we realised that it was necessary to rerun the pipeline many times due to conversion, linking or mapping errors. Once the ETL pipeline was finished, we examined the ETL output for errors. We fixed the errors in the ETL pipeline and rerun the pipeline. This process was repeated until a satisfiable ETL output was generated. Hence the idea was born to create an ETL system, which can perform online dataset updates. As was shown in [2] RDF views can present a feasible alternative to standard ETL approaches in the context of data quality management. Based on the concept of RDF views, this paper proposes a novel architecture which gives the possibility to "debug" knowledge graphs. The term "debugging" is used in broad sense and illustrates the idea that the effects of changes at the linking, mapping or data normalization pipeline steps can immediately be examined without having to wait for a complete re-ingest, hence allowing an immediate "debugging/evaluation" of changes and their impact on the target dataset.

2 Design

In order to ease the quality assessment of heterogeneous datasets, we propose a view-based RDF transformation design. This design bears the advantage that many data and transformation related changes can be reviewed promptly instead of potentially having to re-ingest the data. The proposed design is shown in Figure 1. In order to support the handling of heterogeneous datasets we identified the following main features:

- *RDF views*: Since our proposed architecture uses RDF-based views, we suggest to convert the source data directly to RDF without cleaning the input data. As RDF-2-RDF mapping languages, SPARQL CONSTRUCT[5] queries with minor syntax extensions for naming views and quad support could be exploited. Another option is to adapt RML[6] for this purpose. Compared to SQL-based views, the RDF-based views approach has the advantage that SPARQL queries of one layer can be transformed into SPARQL queries of another layer without limitations of SPARQL-2-SQL conversions.

[4] `http://lod-cloud.net/`
[5] `https://www.w3.org/TR/rdf-sparql-query/#construct`
[6] `http://semweb.mmlab.be/rml/spec.html`

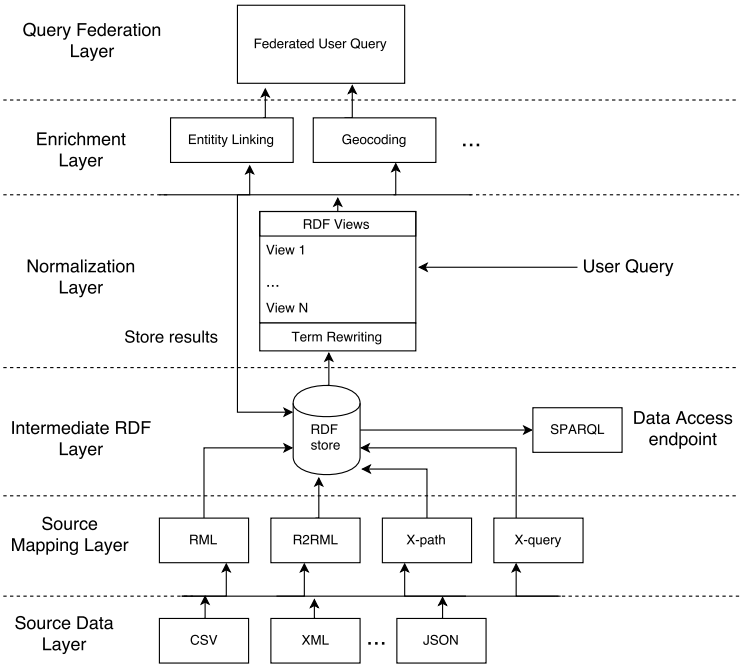

Fig. 1. RDF Views Design Diagram

- Functional Indexes for transformation functions: The input data is rarely expected to fit the target ontology and all defined requirements. In order to speed up access to data normalized via views, the proposed system supports functional indices similar to those supported by traditional RDBMS[7].
- *Real-time updates*: Due to the virtualization of the RDF and Normalization Layer, changes applied in these layers have immediate effect on the virtualized RDF views. This feature allows ontology engineers to follow an iterative process when creating a new dataset, rather than re-ingesting and checking parts of the dataset in order to save time.
- *Unified Querying and ETL*: All relevant source data is expected to have been mapped to appropriate target ontologies. Using SPARQL, it is possible to retrieve portions of the data on-the-fly as well as to export all data at once.

Our design consists of 6 layers: The *Source Data Layer* is used to ingest the original source data via the *Source Mapping Layer* into the RDF backend using mappings which alter the source data as little as possible. The *Intermediate RDF Layer* provides direct access to the non-normalized RDF data. The *Normalization Layer* expresses normalizations by means of RDF2RDF views, possibly supported by functional indices and caches. These indices and caches are supposed to improve the query performance against the normalized RDF

[7] For example, see Postgres: `http://www.postgresql.org/docs/9.5/static/indexes-expressional.html`

data. Thereby, there are two kinds of views: Term-based views for crafting RDF terms, especially IRIs, and triple-based views for relating the terms to each other. If required the *Enrichment Layer* can be used to trigger the additional generation of information about existing entities. Finally the *Query Federation Layer* can be used to integrate other (possibly virtual) SPARQL endpoints. Note, that changes made in one layer are automatically propagated to the upper layers.

3 Discussion and Future Work

The authors are aware that RDF views come with performance implications [2]. It can be compared to a debug executable which executes slower by an order of magnitude compared to a release version. But this executable offers a lot of features which support the debugging process. In the same way the proposed design allows unique knowledge graph "debugging/evaluation" capabilities. For some use cases the performance of this design might be sufficient and would not require a full RDF export. If a "release" version of the dataset is required, SPARQL queries can be used to retrieve portions of the data on-the-fly as well as export all data at once. This "release" data can then be loaded into a graph backend. This generic architecture can be used to add different data sources to the knowledge graph. These data sources could be databases, REST APIs, etc. Since a lot of source data is only accessible via REST APIs or via databases, this opens new integration opportunities for source data. These backends could be integrated via the *Query Federation Layer*. In addition, it would be possible to add a *Meta Data Layer* which could store provenance, confidence and other meta-information which is relevant for relations and entities. In addition it would be possible to add a *Fusion Layer* which would use all *owl:SameAs* relationships and entity fusion algorithms to show a fused view of all entities for the imported source datasets. Despite a possible performance loss by using RDF views, the advantages of such a design surpass the disadvantages, especially since it can be combined with other existing solutions at each layer. Hence we believe that this novel design will allow the creation and curation of knowledge graphs in a new, iterative way.

Acknowledgments. This work was supported by grants from the Federal Ministry for Economic Affairs and Energy of Germany (BMWi) for the SmartDataWeb Project (GA-01MD15010B)[8]

References

1. W. Beek, L. Rietveld, H. R. Bazoobandi, J. Wielemaker, and S. Schlobach. Lod laundromat: A uniform way of publishing other people's dirty data. In *ISWC*, 2014.
2. N. Konstantinou, D. E. Spanos, and N. Mitrou. Transient and Persistent RDF Views over Relational Databases in the Context of Digital Repositories. *Communications in Computer and Information Science*, 390 CCIS:342–354, 2013.

[8] `http://smartdataweb.de/`