Linked Data Querying through FCA-based Schema Indexing

Dominik Brosius and Steffen Staab

Institute for Web Science and Technologies, University of Koblenz-Landau,
{dbrosius, staab}@uni-koblenz.de

Abstract. *The efficiency of SPARQL query evaluation against Linked Open Data may benefit from schema-based indexing. However, many data items come with incomplete schema information or lack schema descriptions entirely. In this position paper, we outline an approach to an indexing of linked data graphs based on schemata induced through Formal Concept Analysis. We show how to map queries onto RDF graphs based on such derived schema information. We sketch next steps for realizing and optimizing the suggested approach.*

Keywords: Linked Data, Formal Concept Analysis, Schema Indexing

1 Introduction

The ease of consuming Linked Open Data (LOD) depends on the availability of schema information. This holds for tasks such as browsing, where a user may not know the structure of the data found in a dataset, or querying, where schema information could be used for query optimization. In the LOD setting a query can be evaluated against the original LOD datasets through query federation or by evaluating it centrally against a LOD crawl. In both scenarios, the efficiency of query evaluation may be improved by identifying those datasets schematically matching the query – maybe partially – while leaving out others.

In the LOD cloud schema information is sparse, as many data items come with incomplete schema information or lack schema descriptions entirely. Methods for schema induction and ontology learning have been studied to remedy this problem ([4], cf. section 2). A method, that has successfully been used in this context, is Formal Concept Analysis (FCA) (cf. [2], [3], [12]).

To our knowledge, FCA has not been used for constructing indices, that allow a schema-oriented query-to-graph mapping. Particularly, we are not aware of previous work on mapping queries to formal contexts. We expect that FCA will benefit this kind of indexing through the formally sound construction of schemata that are free of external heuristics and concise at the same time. In this paper, we outline an approach for such a schema index and point to future work on an implementation.

2 Related Work

For our approach to schema indexing, we propose a property-based schema induction using FCA. Property-based type clustering is discussed in literature as a remedy for the schema sparsity in LOD datasets. The feasibility of such approaches has been analyzed in [5]. The authors show that the properties of resources within LOD datasets can be used for deducing resource types. [10] argue for a property-based data access for programming with Linked Data in order to deal with a lack of type descriptions. [6] further corroborate this position, as they argue for property-based concept definitions. In order to ensure quality of such definitions, they propose a system for incorporating interactive user feedback.

Schema induction, the learning of schema information from data, is a way to handle schema sparsity. [11] describes a statistical approach to schema induction through association rule mining on transaction tables. A recent approach is presented in [7]. It combines the mining of property-based entity descriptions and type clustering over these using DBSCAN. An overview over the field of schema/ontology learning is given in [4].

FCA has been applied to problems related to ontology learning. In [3] it has been used for learning taxonomies from natural language text. The authors of [2] and [12] use attribute exploration from FCA for semi-automatic approaches to ontology completion. Lately, [1] used FCA-based association rule mining for completing type definitions given in DBPedia data through further implied information.

We combine the induction of schema with building and providing an index for query-to-graph mapping. An index for subgraph querying is presented in [14]. There the authors make use of a precomputed lattice of subgraphs in order to narrow down the candidates for subgraph queries. An approach similar to ours is demonstrated in [8]. The authors present a schema-based index that is consisting of three layers, each supporting different types of queries. The index is constructed through clustering of RDF type information in a stream-based fashion. The approaches presented in [7], [8] and [12] either introduce heuristics or human oversight or require case specific parameter tuning. We sketch a FCA-based schema index that is free of such external factors and general enough to map the diverse data found in the LOD cloud.

3 Preliminaries

In the following we give a short introduction to Formal Concept Analysis (FCA), a method for conceptual knowledge discovery and representation, as well as the basics of RDF and SPARQL.

We lend the following definitions 1, 2, 3 from [13]:

Definition 1 (Formal Context). *A formal context* $\mathbb{K} \coloneqq (G, M, I)$ *is a triple comprised of a set of objects* G*, a set of attributes* M *and an indicidence relation* $I \subseteq G \times M$ *encoding that "*g *has attribute* m*" iff* gIm*.*

Definition 2 (Formal Concept). *For $A \subseteq G$ and $B \subseteq M$ [13] define $A' := \{m \in M | \forall g \in A : gIm\}$, $B' := \{g \in G | \forall m \in B : gIm\}$.*

A formal concept is a pair (A, B) with $A' = B$ and $B' = A$. A is the extent, B is the intent of the concept. As in [13], we abbreviate the set of all formal concepts for a formal context as $\mathcal{L}(G, M, I)$.

Formal concepts fall naturally into a hierarchy, called a *concept lattice*, by a subconcept-superconcept-relation defined via the subset relations over G and M, respectively.

Definition 3 ($\leq$). *For two concepts (A_1, B_1), (A_2, B_2) [13] define $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2$ (equivalently, $\Leftrightarrow B_1 \supseteq B_2$). The pair $(\mathcal{L}(G, M, I), \leq)$ is called concept lattice.*

RDF Datasources of the LOD cloud contain RDF[1] data. For now, we abstract from the possibility of blank nodes in RDF data.

Definition 4 (RDF Graph). *Given the set of RDF resource identifiers U and the set of literals L and having the set of RDF terms $T := U \cup L$: An RDF graph is a set of RDF triples $D \subseteq \{<s, p, o> | s \in U, p \in U, o \in T\}$. We further define the set of all known RDF graphs $\mathcal{D} := \{D | D \text{ is a known graph}\}$.*

SPARQL A language for formulating graph oriented queries against RDF datasets is SPARQL[2]. For this paper, we will focus on Basic Graph Patterns that are used for formalizing graph pattern matching and, thus, are fundamental to SPARQL(cf. [9]). Here, we further assume that queries are only evaluated against the default graph of a datasource.

Definition 5 (Basic Graph Pattern). *Given a set V of variable names, a Triple Pattern tp is a triple $(s, p, o) \in (U \cup V) \times (U \cup V) \times (T \cup V)$. A Basic Graph Pattern (BGP) of a query q is a set containing a number of Triple Patterns. We abbreviate the set of all queries only containing a single BGP as BGP.*

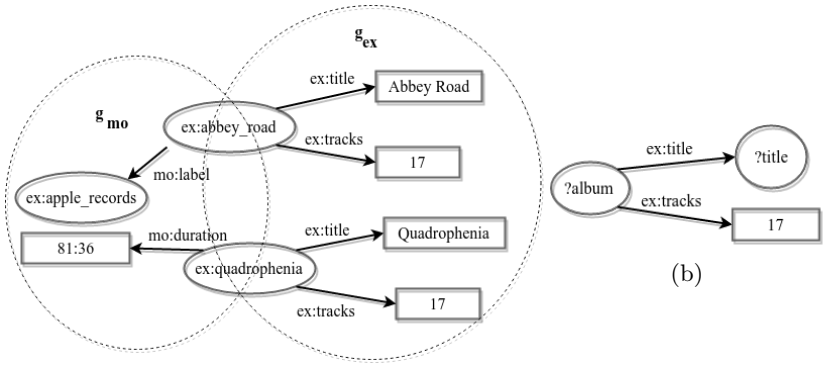
With this restriction to single-BGP queries, we define the solution to a query q through matching its BGP against the RDF graph D as $q(D)$ (cf. [9]). We further define a solution to q against a set of graphs $\mathcal{D}$ as $q(\mathcal{D}) := q(\bigcup_{D \in \mathcal{D}} D)$.

An example for a BGP and a possible solution is given in fig. 1 b), c).

4 Schema Index

In a query federation system a central query processor accepts and processes queries by dispatching (parts of) the queries to datasets (i.e., computing nodes serving them). In the case of LOD these datasets are graphs. The schema index maps queries onto minimal sets of graphs required for evaluating the given

[1] Resource Description Framework; `http://www.w3.org/TR/rdf-concepts/`
[2] `http://www.w3.org/TR/sparql11-overview/`



(a) (b)

album	title
ex:abbey:road	"Abbey Road"
ex:quadrophenia	"Quadrophenia"
...	...

(c)

Fig. 1: Examples of (a) RDF graphs, (b) a BGP and (c) a solution to the BGP

queries. The optimal schema index minimizes the subset of all known RDF graphs to be used for returning the complete solution of a query q:

$$\mathcal{D}_i(q) = \text{argmin}_{\mathcal{D}_j \subseteq \mathcal{D} \wedge q(\mathcal{D}_j) = q(\mathcal{D})} |\mathcal{D}_j|. \tag{1}$$

The improvement of efficiency of query evaluation then depends on the number of graphs left out of $\mathcal{D}_i(q)$. Judging a graph to be relevant for query evaluation may be based on a schematic match of a query and the graph.

In our approach the matching will be informed by an underlying lattice of property type clusters induced from the graph data via FCA. With our approach, the schema index constructs and maintains a schema lattice covering every graph encountered. We build this lattice by applying FCA on a formal context extracted from an RDF graph.

Definition 6 (Resource-Feature Map). *We define a feature domain F as the set of all known predicates p encountered in $\mathcal{D}$. A resource-feature map $f : \mathcal{D} \times U \to 2^F$ assigns an RDF resource r the set of describing features found in a graph: $f(D, r) = \{p \in F | \exists o \in T :< r, p, o >\in D\}$.*

Example 1. $f(\mathtt{g_{ex}}, \mathtt{ex:quadrophenia}) = \{\mathtt{ex:title}, \mathtt{ex:tracks}\}$

There are other possible manifestations of such a resource-feature map. For example, one could (also) map the types assigned to resources through `rdf:type`.

Definition 7 (Schema Lattice). *We define a graph-covering formal context $\mathbb{K} := (U, F, I_f)$ with the set of resources U, the feature domain F and $I_f :=$*

$\{(r,p)|\exists D \in \mathcal{D}, \exists r \in U : p \in f(D,r)\}$. *The schema lattice* $\mathcal{L}_S = (\mathcal{L}(U,F,I_f),\leq)$ *is the lattice constructed from* $\mathbb{K}$ *via FCA.*

While extracting features given in the graphs, the schema index will also store the set of graphs contributing individual features to the subsequent lattice construction.

Definition 8 (Feature-contributing Graph). *For an RDF resource* r *and a feature* $p \in F$*, we define a set of feature-contributing graphs as* $\gamma(r,p) = \{D|p \in f(D,r)\}$*. For the features of an intent* B*, we define the set of contributing graphs as* $\gamma'(B) = \{D|\exists p \in B, \exists r \in U : D \in \gamma(r,p)\}$.

We conceive the schema index as a pair $(\mathcal{L}_S, \Gamma)$ of the schema lattice $\mathcal{L}_S$ and a function Γ mapping queries to indexed graphs. In order to look up graphs that a query should be evaluated against, the schema index then derives the queries type information through the *Query-Type Map* function:

Definition 9 (Query-Type Map). *The Query-Type Map* Θ *is a function* Θ: $BGP \rightarrow 2^F$ *that maps a query* q *to a set of intents* $\Theta(q)$.

For a query q the schema index returns the set of schematically appropriate graphs as follows:

$$\Gamma(q) = \{D|D \in \gamma'(B) \text{ for some } B \in \Theta(q)\}.$$

For our running example (cf. fig. 1), the graph g_{ex} would be returned for the BGP, as here {`ex:title`, `ex:tracks`} constitutes an intent reflected in the query.

5 Outlook and Conclusion

In this position paper, we sketch the idea of a schema index for a query-to-graph mapping. For constructing the index we aim at an automatic schema induction process through FCA. We hypothesize that FCA is a good method for this use. One, it is independent of external factors, such as expert involvement as in [12] or a task specific parameter tuning as done in [7]. Two, by design it is a method targeting property-based entity descriptions as argued for in [6], [5].

Our proposition is that the schema index returns the graphs necessary for a complete query evaluation:

$$q(\Gamma(q)) = q(\mathcal{D}) \quad \text{(for every query } q\text{)} \tag{2}$$

However, $\Gamma(q)$ may return graphs that might be disregarded for evaluating BGP. Hence, the proposed schema index is not necessarily optimal. For a future implementation we plan to approximate the optimal case (cf. (1)). This must only be done to a degree that is sensible, since potential savings in execution time are being countered by costs for building and looking up the index.

Towards a realization of the proposed schema index, we expect to face questions such as for the size of formal contexts as given through the feature domains of a graph. Next steps for our research will also involve empirical analysis of runtime behaviour and scalability of FCA in the LOD use scenario. We have indications that in spite of exponential worst-case complexity, the property-based clustering of data items may lead to empirically acceptable runtime behaviour. A further question in this context is how to efficiently update existing lattices, when, e.g., encountering further features of an object while still ingesting a graph in a stream-based fashion. For such a scenario, we will also look into what lattice construction algorithm to choose. Finally, we plan to address the problems of reducing the size of formal contexts and lattices through appropriate data preparation and cleansing lattices of "noisy" concepts.

References

1. M. Alam, A. Buzmakov, V. Codocedo, and A. Napoli. Mining definitions from RDF annotations using formal concept analysis. *Proc. of IJCAI*, pages 823–829, 2015.
2. F. Baader, B. Ganter, and U. Sattler. Completing description logic knowledge bases using formal concept analysis. *Proc. of IJCAI*, pages 230–235, 2007.
3. P. Cimiano, A. Hotho, and S. Staab. Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Journal of Artificial Intelligence Research*, 24:305–339, 2011.
4. C. D'Amato, N. Fanizzi, and F. Esposito. Inductive learning for the Semantic Web: What does it buy? *Semantic Web*, 1(1-2):53–59, 2010.
5. T. Gottron, M. Knauf, S. Scheglmann, and A. Scherp. A Systematic Investigation of Explicit and Implicit Schema Information on the Linked Open Data Cloud. *Proc. of ESWC*, 7882:228–242, 2013.
6. S. Homoceanu, P. Wille, and W. Balke. ProSWIP: Property-based data access for semantic web interactive programming. In *Proc. of ISWC*, pages 184–199, 2013.
7. K. Kellou-Menouer and Z. Kedad. Schema discovery in RDF data sources. In *Proc. of ER*, pages 481–495, 2015.
8. M. Konrath, T. Gottron, S. Staab, and A. Scherp. Schemex - efficient construction of a data catalogue by stream-based indexing of linked data. *Web Semantics*, 16:52–58, 2012.
9. E. Prudhommeaux, A. Seaborne, et al. Sparql query language for rdf. *W3C recommendation, https://www.w3.org/TR/rdf-sparql-query/*, 15, 2008.
10. S. Scheglmann, G. Groener, S. Staab, and R. Lämmel. Incompleteness-aware programming with RDF data. *Proc. of the 2013 workshop on Data driven functional programming, DDFP 2013*, pages 11–14, 2013.
11. J. Völker and M. Niepert. Statistical schema induction. In *Proc. of ESWC*, pages 124–138, 2011.
12. J. Völker and S. Rudolph. Fostering web intelligence by semi-automatic OWL ontology refinement. *Proc. of Web Intelligence, WI 2008*, pages 454–460, 2008.
13. R. Wille. *Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts*, pages 445–470. 1982.
14. D. Yuan and P. Mitra. Lindex: A lattice-based index for graph databases. *VLDB Journal*, 22(2):229–252, 2013.