ABSTRACT
The contemporary Web is formed mainly by documents, from which it is rather complicated to automatically retrieve hidden structured and interlinked information. The idea of Linked Data based primarily on RDF data triples seems to successfully follow this drawback. In recent years, a significant effort was made not only in a theoretical research, but also in the amount of Linked Data globally available. Since RDF triples are modelled as graph data, we cannot directly adopt the existing solutions from relational databases or XML technologies. Thus, several research questions remain opened.

The purpose of our ongoing research effort is to propose an efficient framework for querying Linked Data. This requires finding the compromise between storing data in local storages and accessing them directly on-demand in distributed data sources. Next, we need to choose a suitable querying language, propose auxiliary indexing structures and, finally, to define an ordering model for query results. The theoretical research will be supplemented by a prototype implementation and experiments over real-world data.

Categories and Subject Descriptors
H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms
Algorithms, Experimentation, Performance

Keywords
Linked Data, RDF, Distributed Data, SPARQL, Querying, Index Structures

1. INTRODUCTION
The contemporary Web is composed primarily from HTML documents, which are not suitable for automated retrieval of hidden, structured and interlinked information. Therefore, an idea of Linked Data based primarily on RDF appeared, attempting to supplement the Web of Documents towards the Web of Data.

Linked Data do not represent any particular standard, we only talk about a set of recommended techniques and rules, which lead to the publication of data in a way more feasible to the mentioned automated processing. Each real-world entity should by uniquely described by a URL identifier, these identifiers can be dereferenced by HTTP to obtain information about the given entities and, finally, that these entity representations should be interlinked together to form a global open data cloud.

One particular way to achieve the described aims is the usage of RDF, i.e. data modelled as triples conforming to a concept of subject-predicate-object. An alternative way to view these triples are graphs, where vertices stand for subjects and objects, edges represent the triples themselves and are labelled by predicates. At the implementation level we can publish RDF triples in a form of RDF/XML syntax and along the data we can also publish RDFS schemata and OWL ontologies restraining the allowed content of such RDF data.

In recent years, a significant effort was made not only in a theoretical research, but also in the amount of Linked Data globally available. If we could find about 10 globally important data sources with 920 millions of triples and 150 thousands of links in 2007 [21], following the CKAN directory these numbers increased to about 200 important sources, 25 billions of triples with 395 millions of links in 2010.

Clearly, the size of the Linked Data cloud significantly increases. However, since RDF triples are modelled as graphs, we cannot directly adopt existing solutions from relational databases and XML technologies. Thus, the area of Linked Data includes many open problems, starting from application architecture questions and ending, e.g., with user interaction paradigms.

The purpose of this paper is to describe our ongoing research effort, which should result in a proposal of an efficient framework for querying of Linked Data. The first essential ques-
tion is the architecture of such system. In particular, the compromise between storing local copies of RDF triples and between online accessing of distributed data sources needs to be found. Local data processing algorithms obviously may bring better performance, but since Linked Data change a lot, any local data may become obsolete unexpectedly fast.

It seems that the maintenance of local auxiliary index structures may solve this problem instead of storing data triples entirely locally. Using indexes we can retrieve distributed data sources participating on a query result, rapidly reducing the amount of data that are really needed to be accessed on-demand. In other words, our attention will be put not only in a proposal of such index data structures, we also need to harness characteristics of such structures to propose an adjusted query evaluation processor.

After a thorough analysis of advantages and disadvantages of the state-of-the-art solutions from the area of Linked Data storing, indexing and querying, our research effort will aim at a proposal of a novel querying framework over Linked Data, dealing with data volatility over time and the aspect of data distribution. The theoretical research will be supplemented by a prototype implementation and experiments over real-world data.

Outline. In Section 2 we provide an overview of open questions in the area of Linked Data research and a description of significant problems related to our intent. Section 3 introduces several existing solutions and known approaches, while in Section 4 we sketch our preliminary ideas how to achieve our goals. Finally, Section 5 concludes this paper.

2. RESEARCH PROBLEMS

Recent years show that the Linked Data concept already became widely accepted. The number of research papers indicates the interest from theoretical areas, moreover, also activities of ordinary web users contribute to the massive growth of publicly available Linked Data in the Web of Data.

Open Problems

As already outlined, we cannot directly adopt results of existing intensive research around relational databases and XML technologies and apply them in the context of Linked Data. As a consequence, we need to discuss ways how Linked Data should be stored, queried, indexed or presented to users in a human-oriented way. The following paragraphs inspired by [6] attempt to bring at least a brief overview of opened questions related to our formulated research intent.

Application Architectures. We can identify two basic variants how Linked Data can be stored, managed and further processed. First, we can maintain independent data copies in local storages, benefiting from convenient conditions for efficient query evaluation. On the other hand, we need to deal with data ageing. The second approach is based on accessing distributed data on-the-fly using link traversal. The choice of an appropriate architecture for a particular application seems not to be an easy task.

Links Maintenance. Experiments from [20] performed in the standard Web of Documents show that approximately every 50 days the web changes in 50% of its content. We can expect the similar behaviour even in the Web of Data. Moreover, we especially need not only to handle simple data modifications, we also need to deal with broken links between Linked Data and attempt to prevent them or correct them in order to achieve consistently connected data [24].

Quality and Relevance. Another question of retrieved data relevance, qualities or provenance arises not only because the amount of globally available Linked Data still grows. When the user wants only a limited number of hits returned as an answer to a submitted query, we need to sort all hits and create a required limited view. And a similar situation takes place in imprecise forms of querying.

Research Intent

The aim of our work is to propose a complete framework for efficient querying over Linked Data from distributed sets of data. The state-of-the-art approaches already focus this area, however, we can identify several nontrivial questions that still need to be sufficiently discussed.

As already outlined, we first need to find an architecture compromise between processing local or distributed data. This will lead to auxiliary index structures that can be used to increase query evaluation efficiency and to reduce required local storage demands. A query evaluation algorithm itself thus needs to be discussed as well. Although existing solutions are capable to adequately deal with named topics separately, especially the combination of large, distributed and changing sets of data remains open.

System Scalability. Even though existing approaches work with large sets of data, it seems that only authors of [2] performed experiments over a set with roughly 1.3 billions of RDF triples, i.e. a set that at least attempts to correspond to the current state of the Web of Data size explosion.

Distributed Datasets. Another significant problem is the data distribution in the entire cloud of the Web of Data. We have already sketched that frameworks based on local data copies would lead to probably unacceptable space demands. Accessing distributed data directly at runtime, on the other hand, would not provide sufficient performance for complex queries.

Query Evaluation. The suitable compromise can be found using auxiliary index structures. Therefore, we will not store original data locally, but only auxiliary statistics, graph pattern summaries or other derived data to support efficient query evaluation algorithms adjusted right for these structures.

Data Changeability. Not only local data copies, but also index structures themselves are apparently the subject of ageing. Since Linked Data and especially links between them change with an unexpected frequency, the proposed index structures need to be easily and frequently updated with low running costs.

3. EXISTING APPROACHES

The purpose of this section is a basic overview of the current work in our research area. We will briefly discuss ways of Linked Data browsing, searching or storing and in a more
detail we will introduce a few existing solutions from indexing and querying areas.

**Linked Data Browsers.** Similarly to traditional Web browsers that allow presentation and navigation between particular HTML pages by following hypertext links, the goal of Linked Data browsers is to present data in a human-friendly way and to enable navigation using typed links. In this category we can name, e.g., Tabulator browser [5].

**Human-oriented Search Engines.** While browsers can be treated as applications where we can navigate through the information space, search engines provide starting points for such browsing. Approaches called Falcons [7] and SWSE [18] enable searching over the global data space using keyword-based mechanisms.

**Application-oriented Indexes.** Contrary to the previously discussed search engines, application-oriented indexes serve the needs of particular applications built on the top of Linked Data. These applications are able to use index services through public interfaces. Two main representatives are Sindice [25] and Swoogle [12].

**Storages and Querying.** The basic way how to perform structural queries over RDF data is offered by SPARQL [13]. Existing solutions indicate that the way of Linked Data storing, indexing and processing is tightly connected together. From the point of data storing, the state-of-the-art solutions are covered especially by Hexastore [28], RDF-3X [23] or SW-Store [1] approaches. The three following subsections will provide a more detailed view on selected particular indexing techniques related to our research intent.

**Parameterised Structure Index**

A more detailed overview of related approaches can be started with work [27] attempting to propose a parameterised structure index, derived data partitioning and a query processing algorithm causing the reduction of joining costs by pruning queried data already at the level of an index.

The authors of this paper introduced an index structure, which forms a compact representation of an original RDF data graph attempting to underpin the structural similarity or even equality of the neighbourhood of particular vertices. This concept is based on bisimulation and bisimilarity relations, giving in a relation such two vertices of the original data graph that share the same outgoing and ingoing edges, reflecting only labels of these edges, i.e. names of predicates of represented subject-predicate-object triples. Vertices considered to be equivalent belong to the same equivalence class and, therefore, share the same neighbourhood characteristics. The mentioned parameterisation is based on a specification of two sets of predicate names, which are only considered for outgoing and ingoing edges respectively.

Being given a SPARQL query with variables and constants, our goal is to find a set of all possible functions assigning to variables corresponding values from the queried RDF data graph. The query evaluation procedure can be decomposed into two main steps. First, we attempt to find matches of variables against the index graph, next, we combine data elements from retrieved equivalence classes grouping them in order to acquire a final query result, i.e. a set of introduced mappings from variables to data values.

When constructing a graph index, we need to decide, which sets of predicate names we want to consider. Clearly, more labels included in these sets implies larger index graphs. On the other hand, larger graphs mean more fine-grained indexation and bring higher probabilities of successful pruning during the query evaluation, i.e. the possibility of throwing away classes that cannot contribute to the final query result.

**Data Summaries Index**

The purpose of work [16] is to provide an efficient system for querying over distributed RDF data using centralised auxiliary index. The query evaluation algorithm for conjunctive SPARQL queries first locally analyses a provided query and using the constructed index detects distributed data sets, only which can potentially contain triples contributing to an overall query result. As a consequence, the number of data triples needed to be accessed online for current data is significantly restricted and these on-demand requests can even be managed in parallel.

Assuming a rather high number of distributed data sets only with pure RDF data, particular triples are modelled as points in a 3-dimensional space. Subject, predicate and object coordinates are derived using hash functions transforming names into fixed numeric intervals in each corresponding dimension.

The main idea of the proposed index structure is to group summaries about close triples together, exploiting accepted techniques for indexing spatial data. For this purpose authors introduced QTrees, which are derived from standard R-Trees [15]. Internal nodes of such trees contain definitions of regions acting as minimal bounding boxes for nested nodes, however, leaf nodes are not intended to store RDF triples, but only summaries about them in a form of special structures called buckets. Each bucket is described as a spatial region and, in addition, it also contains basic statistical data about all distributed data sources that contain triples locally indexed right in a given bucket. More precisely, for each such data source we store a number of corresponding triples in it. It is worth pointing out that the size of the entire QTree does not depend on the size of a set of indexed triples, but only on fixed parameters.

The query evaluation algorithm involves three main phases. Before it can start, a submitted query first needs to be preprocessed. This means that each its pattern is converted into a 3-dimensional region, specified constants to hashed values, variables to maximal allowed intervals. During the first phase, for each pattern we traverse the QTree index from its root node towards leaf nodes and create a set of all buckets having a nonempty overlapping with a given pattern region. Therefore, only triples summarized by such buckets may contribute to a query pattern result. The second phase represents the joining of all individual query patterns. Note that we still do not work with RDF triples, we consider only buckets. In order to join two particular patterns, we just need to create the Cartesian product and then consider only those bucket combinations that have a nonempty overlapping in shared dimensions, resulting into a new set
of buckets, now with the number of dimensions increased by three. The shared dimensions are those dimensions that correspond to joining variables. Having processed the entire query, during the third phase we on-the-fly access suitable distributed data sets and retrieve current versions of RDF data triples. Having these triples, using an analogous algorithm for joining we are able to produce the final query result.

In other words, the purpose of the presented algorithm is not to locally process the standard query joining, but to locally decide, which remote data sources may contribute to the join query result and, thus, data sources which only need to be inspected and accessed. The approach evaluation shows that although the proposed idea seems to be promising, the QTree precision is below authors' expectations.

**Local Join Processing**

Authors of [2] introduced an efficient query processor for evaluating complex and less selective conjunctive SPARQL queries over large local databases of RDF triples. The mentioned less selectivity means queries, which are composed from individual patterns having potentially high number of RDF candidates conforming to them, and even joining variables between these patterns do not reduce the candidate sets at much.

The core of the work is an index structure based on a matrix modelling the entire set of all stored data triples. This matrix has three dimensions, one for all present values from the domain of subject names S, one dimension for predicates P and one for objects O, supposing that all these names are ordered and numbered using a sequence of non-negative integers. Each cell contains a bit value equal to 1 if and only if the triple determined by a given position is stored in the database. Along other auxiliary values, the index is physically organized as an ordinary file, where we successively store S-O and O-S matrix slices for all predicates, analogously P-O slices for all individual subjects, and P-S slices for all objects. The problem of the representation at the implementation level is thus reduced only to matrices with two dimensions, and since the original matrix is obviously sparse, authors use bit runs compression per individual matrix rows to drastically reduce the required space.

Once we have built the described index structure for a considered set of data triples, we can use it for efficient evaluation of queries. For this purpose the authors assume only queries without isolated patterns, i.e. all query patterns need to be connected together via joining variables.

The first step is a constraint graph construction. For each query pattern we add a vertex into this graph, similarly for each joining variable a vertex too. We add an edge between a pattern and a joining variable, if such variable has an occurrence in a given pattern. Next, we add edges between patterns sharing at least one joining variable and, finally, we add edges between joining variables, if they occur in the same pattern.

Using this graph, we are able to create an execution plan for a given query evaluation. We start with the decision of joining order, preferring patterns and their joining variables that seem to produce less triples participating on a final query result. This concept is a straightforward parallel to a standard join query evaluation in the theory of relational databases using algorithms of nested loops. After the execution plan is prepared, authors perform also the phase of pruning, throwing away triples that evidently cannot contribute to the final query result. In order the entire algorithm would be efficient and with minimal system memory requirements, all basic operations used during the mentioned pruning and subsequent joining need to be implemented right over compressed bit runs, from which the index is composed.

Performed experiments show high efficiency and scalability of this approach, emphasizing especially the size of considered data set containing 1.33 billions of triples. As authors admit, processing of queries with highly selective patterns, however, seems to be faster using conventional join processing algorithms. In the context of our research intent, we would need to investigate the behaviour of this index approach in a changeable environment, thus, if this index can support massive dynamic updates at all.

**Handling Broken Links**

One of the focused problems in our research effort is the consideration of the Linked Data volatility. This means that not only RDF triples inside one data source may change significantly over time, but also links between different data sets are liable to uncontrolled modifications.

Broken links can be caused by semantic and structural reasons. The first named category represents situations, when an originally intentioned link meaning changes. While this category can hardly be solved, the latter one provides at least a set of possible approaches based on preventive, adaptive or correction methods.

Authors of [24] introduced a framework enabling online detection of broken links in locally stored RDF triples. The proposed system automatically and periodically attempts to access remote data sources and when a particular link target resource seems not to be accessible, the system announces an existing resource removal or a move event, depending on a detected probability and a state of a predefined threshold value exceeding.

Other approaches can be based on links recomputation as in [17] or changes publication method in [3].

### 4. PRELIMINARY IDEAS

This section brings a more summarized overview of our research objectives, assumptions and planned system evaluation steps.

**Objectives.** The areas involving our attention are especially Linked Data storing, indexation and querying. Our work should result in a proposal of a complete framework that would be able to efficiently evaluate queries over distributed RDF data using index structures. Moreover, we would like to discuss also ways of query results ordering, aiming at possibilities of potentially limited size of returned hits and forms of fuzzy querying. The proposed system should have acceptable memory requirements, query evaluation efficiency and low index maintenance costs. It is clear that the
system will need to solve also questions of queries submitting and hits presentation in a human-friendly way, however, these areas will not be investigated and basic existing solutions will only be adopted, despite these areas can still be considered as open.

Assumptions. We would like to consider especially three aspects, which are not yet sufficiently surveyed. The nature of Linked Data assumes that data are distributed within the entire global cloud of the Web of Data. Since completely centralised solutions seem not to precisely follow this idea, we want to find a suitable compromise between completely centralised and totally distributed approaches. Next, the amount of globally available data in this cloud significantly increases each year. Only a few existing approaches are shown to be able to cope with such data amounts. Finally, Linked Data are the subject of nontrivial changes over time and, thus, the aspect of the data volatility cannot be ignored in the framework architecture or index structures proposal. In summary, we want to target at large amounts of distributed and volatile data concurrently.

Architecture. The proposed framework will be based on a locally managed index structures, which will be periodically updated to deal with detected changes in the global Web of Data. The querying will be accessible in a form of a human-friendly web page and also in a form of an endpoint serving for needs of automated web applications and services.

Methodology. Although we have outlined several existing approaches from the area of Linked Data indexing, a more detailed survey of the state-of-the-art solutions also from related areas is needed. The theoretical analysis and research should lead to the innovative proposal of an indexing and a query processing approach with a theoretical background model, data structures and algorithms. We will not only take into account main discovered advantages and disadvantages, we also want to perform an analysis of the real-world RDF data and harness acquired results.

Analysis. The knowledge of real-world data characteristics may generally lead to more efficient approaches, since we know, which allowed constructs are really used and how. Then we can determine which parts of our solution may hide bottlenecks and, therefore, which parts need more attention. Our data analysis should focus on two main aspects. First, we need to describe structural characteristics of real-world RDF data, especially structural features of RDF data graphs or usages of RDFS schemata and OWL ontologies including their constructs. The second aspect will investigate the real characteristics of the data volatility. While some modifications may be ignored, links between different data sources deserve an attention. In other words, we want to take into account real-world data features during the proposal of index structures, but the acquired results may also be used in the definition of data quality or provenance too.

Experiments. Besides the theoretical research, a prototype implementation is planned to be created as well. The evaluation will be based on experiments over synthetically generated data and also on real-world data from the contemporary Web of Data. For this purpose we will use our previously developed project called Analyzer [22], which provides a robust system for managing configurable analyses over documents of whatever types through user-implemented plugins. The proposed querying framework will also be compared to other existing approaches for storing RDF data and SPARQL querying endpoints, e.g. using a comparison based on Berlin SPARQL Benchmark Results [4].

Contribution. Although existing solutions already focus on the same research area too, these approaches do not target at all three main described objectives concurrently. Therefore, the contribution of our framework will be based on querying large amounts of Linked Data that are distributed and volatile too.

5. CONCLUSION

Recent years show that Linked Data became a suitable way of building the Web of Data, i.e. the web of data that can be automatically processed by programs without any special effort, i.e. without complicated information retrieval from standard HTML pages composing the contemporary Web of Documents.

Since Linked Data in a form of RDF standard represent graph data, we cannot directly use existing research results from areas of relational databases and XML technologies. And since the amount of Linked Data globally available still grows, several questions need to be solved out to offer evaluated and efficient systems for processing such data.

Our effort aims at the proposal of an efficient framework for processing queries over distributed RDF data that are the subject of ageing. Existing approaches probably show promising ways of solving our problem, however, the combination of all named assumptions remains unsolved.

6. REFERENCES


