Noname manuscript No.
(will be inserted by the editor)

The Linked Data Benchmark Council Project

Peter Boncz · Irini Fundulaki · Andrey Gubichev · Josep Larriba-Pey · Thomas Neumann

Received: date / Accepted: date

Abstract Despite the fast growth and increasing popularity, the broad field of RDF and Graph database systems lacks an independent authority for developing benchmarks, and for neutrally assessing benchmark results through industry-strength auditing which would allow to quantify and compare the performance of existing and emerging systems.

Inspired by the impact of the Transaction Processing Performance Council (TPC) Benchmarks on relational databases, the LDBC consortium formed by University and Industry researchers and practitioners has recently launched a European Commision sponsored project that will offer the first comprehensive set of open and vendor-independent benchmarks for RDF and Graph technologies. The consortium will incorporate the Linked Data Benchmark Council (LDBC) which will survive the project and will supervise the process of obtaining and reporting results as well as fostering the creation and maintenance of new and existing benchmarks. This paper describes the state-of-the-art benchmarks in RDF and Graph databases and overviews the technical challenges that should be addressed in the development of such benchmarks. With this paper we would like to invite the readers to participate in the LDBC effort towards the development of Linked Data Benchmarks, both from the user prospective (by sharing available usage scenarios, datasets, query workloads) and the vendor perspective (by reporting the results of systems and research prototypes).

Keywords Linked Data · Benchmarking

Supported by LDBC EU FP7 project

A.Gubichev
TU Munich
Germany
Tel.:+49 89 28917259
E-mail: gubichev@in.tum.de

P.Boncz
CWI, Netherlands
E-mail: boncz@cwi.nl

I.Fundulaki,
FORTH, Greece
E-mail: fundul@ics.forth.gr

J.Larriba-Pey,
UPC, Spain
E-mail: larri@ac.upc.edu

T.Neumann
TUM, Germany
E-mail: neumann@in.tum.de

1 Introduction

Non-relational data management is emerging as a broad field that deals with complexly structured and heterogeneous data. The main examples of the fast-growing non-relational research domains are Semantic Web, noSQL and graph databases. Both semantic repositories (RDF databases with reasoners) and graph databases share a graph data model and pattern- and path-oriented query languages. Despite their increasing popularity in the research community and the growing interest of the industry, at present there is no comprehensive suite of benchmarks and no independent authority for verifying the results in the RDF/Graph world. Without commonly accepted, technically challenging benchmarks, the future development and adoption of these technologies is endangered by not giving to the industry the clear user-driven targets for performance and functionality. An example of such a benchmarking suite in the relational database world is the TPC benchmark family which boosted the progress of relational database management systems.

The aims of good benchmarks are (i) to help users choose the right system by demonstrating the systems' behavior in different tasks, and (ii) to motivate industry to improve existing systems by posing a set of difficult (but solvable) technical challenges. Designing a benchmark therefore includes the following activities:

– *analysis of the technical challenges* that can only be overcome by innovation (we call such challenges *choke points*). The Linked Data domain encompasses challenges in different technology areas like *core data management* (query processing and optimization, transactions), *graph analysis*, *data integration* and *reasoning*;
– *design of domain-specific use cases* and *engagement with the user community* to obtain real-life scenarios, examples, datasets and query workloads.

In this paper, we present the recently started Linked Data Benchmark Council Project and invite the readers to participate in this community effort towards benchmarking in the RDF/Graph data management fields, both by sharing their experiences (datasets, typical workloads) and by testing their prototypes and systems on suggested benchmarks.

The rest of the paper is organized as follows. First we underline the goals of the benchmarking initiative in Section 2. Section 3 briefly surveys existing benchmarks in relational and RDF/Graph domains. Then in Section 4 we go through the technical issues which will be addressed by the future set of benchmarks, including dataset generation, query optimization, reasoning and data integration tasks. In Section 5 we consider two vertical domains (publishing and social networks) and describe workloads in these domains that are meaningful to users. Section 6 gives an overview of the Council that will supervise benchmarking for RDF and Graph data management systems.

2 Goals of LDBC

The goal of the Linked Data Benchmark Council (LDBC) project is to create the first comprehensive suite of open, fair and vendor-neutral benchmarks for RDF/graph databases together with the LDBC Foundation which will define processes for obtaining, auditing and publishing results.

From the technical perspective, the project has the following important issues that will configure the process of creating benchmarks both for RDF and Graph databases, and that will survive it through the creation of the Linked Data Benchmark Council:

– **Methodology**. The project will create a set of guidelines on how to define, extract, support and analyse benchmarks coming from various use case scenarios and focusing on different features of the databases.
– **Use cases/benchmarks**. The use case scenarios will evolve towards a benchmark and will be decided thanks to the collaboration of the Technical User Community (TUC), that will put their needs through in regular meetings organized by the consortium. TUC members will be informed about the progress of design and implementation of the benchmarks, and asked for feedback and ideas regarding their needs and opinions of the benchmarks under development.
– **Metrics**. LDBC will define fair metrics to measure a binomial hardware/software on a benchmark, assuring quantitative values of performance and performance-cost.
– **Task Forces**. The creation of each benchmark will be pushed forward through a Task Force. This will create the synergies between the different members of the Consortium/Council to decide on specific use cases based on the needs of users and the input of software and hardware vendors. Based on the use case, the Task Force will create the necessary datasets, queries and specific technology difficulties to obtain applicable and scalable benchmarks.
– **Choke Points**. These are related to the difficulties of the different technologies for a given benchmark. Within each Task Force, it will be necessary to understand how to push technology by including queries that are difficult to solve, but not impossible, pushing the technologies to improve over the years, and those difficulties will be decided upon the choke point analysis of the use cases.
– **Data sets**. Open Data sets and/or open source data set generators will be created for each benchmark. These will be provided to the RDF and graph communities, technologists and users, to allow them to use the data for their own tests.
– **Auditing**. When a benchmark is eventually launched, companies will be able to run it on their binomial computer/software. In order to make the benchmark executions fair and vendor-neutral, it will be necessary to train people who will audit them.

3 Related work

3.1 Relational database benchmarking

For relational databases benchmarks are well established, in particular the well-known TPC[1] benchmarks. Although relational databases are different from the RDF/Graph usage scenario, these relational benchmarks introduced several best practice techniques that should be kept for benchmarking core database functionalities in LDBC.

The TPC benchmarks are extremely influential because they have introduced what now is considered important principles of benchmark design. One of their original key ideas was to offer an objective benchmark specification, and to ask vendors for full disclosure reports, including all hardware and software configuration details. This allows for repeatable experiments, and offers much more meaningful comparisons of benchmark results. These strictness and disclosure requirements were tightened even more over time by introducing external auditors that verified the way experiments were conducted and their results.

Besides these formal requirements, the TPC benchmarks take measures to produce meaningful and comparable benchmark numbers. The analytical benchmarks (TPC-H/R/D) for example use both queries-per-hour-at-size ($QphH@Size$), and costs-per-performance ($\$/QphH@Size$) metrics, which are both important concepts. The first, $QphH@Size$, allows for comparing the performance at different scale factors, as some architectures might be superior for very large data sets and others might aim at smaller data sizes. The second metric, $\$/QphH@Size$, normalizes the performance by costs. These costs are specified in the respective benchmarks, and they include, for example, hardware, software licenses, and maintenance for a certain period of time.

Despite being highly influential, the TPC benchmarks have their limitations. One particular weakness of the most commonly used TPC benchmarks (TPC-H and TPC-C) is that the generated data is uniformly distributed and uncorrelated. While this simplifies the setup, it also ignores very challenging real-world problems. The TPC-DS benchmark was designed to explicitly test for data skew, and also to test for more complex queries.

3.2 RDF database benchmarking

A number of RDF benchmarks that use real datasets have been proposed over the last years. The University of Leipzig developed a DBPedia benchmark [17] derived from the actual DBPedia query logs. Unfortunately due to its simple nature, the benchmark's query workload consists of mostly simple lookups that are not representative of either *inference* or *decision support queries*. The YAGO dataset [24] with its accompanying queries is another example of an RDF benchmark. YAGO is a vast knowledge base that integrates statements from Wikipedia, Wordnet, WordNet Domains, Universal WordNet and GeoNames and the queries were initially defined to benchmark the RDF-3X engine [18]. The Barton Library dataset[2] is obtained by converting the Machine-Readable Catalogue (MARC) of the MIT Libraries to RDF and the set of queries defined in Abadi et. al. [2]. The UniProt Knowledge Base (UniProtKB) [22] is a high-quality resource of protein sequence and one of the central datasets in the bio-medical subset of the Linked Open Data initiative, expressed in RDF and was used to test the scalability of query processing using mainly simple lookup queries. The Linked Sensor Dataset [19] contains expressive descriptions of approximately twenty thousand weather stations in the United States in RDF but does not come with a representative set of queries.

Another line of work includes synthetic benchmarks for RDF databases, typically consisting of a *data generator* and a *set of queries*. The Lehigh University Benchmark (LUBM)[3] is intended to evaluate the performance of Semantic Web repositories over a large data set that adheres to a university domain ontology expressed in OWL [16], customizable and repeatable synthetic data, a set of *test queries*, and several *performance metrics*. As the generated LUBM data is regular, and in fact can be represented using the relational model, the benchmark does not explore any of RDF's distinguishing properties and thus the data can be represented easily using relational tables. In addition, the nature of the generated graph makes it harder for testing the performance of join algorithms in the query engine. The LUBM query workload consists of fourteen mainly extensional lookup and join queries, that do not consider complex SPARQL operators or complex reasoning. UOBM [15] extends the LUBM in order to tackle (a) complex inference, and (b) scalability issues. In contrast with LUBM, UOBM uses both OWL Lite and OWL DL ontologies covering most of the constructs of these two sublanguages

[1] Transcation Processing Performance Council:`http://www.tpc.org/`

[2] `http://simile.mit.edu/wiki/Dataset:_Barton`

[3] `http://swat.cse.lehigh.edu/projects/lubm`

of OWL. UOBM queries are designed based on two principles (a) queries should consider multiple lookups and complex joins and (b) each query should support at least a different type of OWL inference covering a larger spectrum than its predecessor. SP2Bench [23] is one of the most commonly used benchmarks for evaluating the performance of RDF engines. The benchmark comprises of a data-generator that produces arbitrarily large documents, which builds upon DBLP [1] bibliographic schema. The benchmark consists of fourteen queries that are designed to test different aspects of SPARQL query optimization.

The Berlin SPARQL benchmark (BSBM) [5] is built around an e-commerce use case where the schema models the relationships between products and product reviews. It was was designed to test the mapping of relational databases to RDF requiring only relational database storage. BSBM comes with a set of fourteen queries (mainly simple lookups). The BSBM BI Benchmark[4] is the business intelligence workload of BSBM that although it addresses basic SPARQL 1.1. [11] features in the queries (e.g., aggregation and subqueries), is still primitive in comparison with comparable relational benchmarks like TPC-H and TPC-DS.

The Social Intelligence Benchmark (SIB) [20] simulates an RDF backend of a social network site (such as Facebook), in which users and their interactions form a graph of social activities such as writing posts, posting comments, creating/managing groups, etc.. SIB comes with a data generator, a set of analytical and update queries and a set of metrics. The Social Intelligence Benchmark is the first attempt to create a synthetic social graph with realistic data correlation.

Clearly, the existing RDF database benchmarks do not fully cover important RDF and SPARQL capabilities. Data remains relational at heart, it is sparsely interrelated, specified using inexpressive ontology languages and does not exhibit traits specific to RDF or graph workloads. In addition, the majority of the queries are either simple lookups (as dictated by the data) or simple joins, and do not include any advanced features such as optional, aggregation, negation and inference. Moreover, the effects of updates at the schema and at the instance level, are completely ignored.

3.3 Graph database benchmarking

As we have seen, the mainstream benchmarks cover the relational model workloads that are typical for an enterprise use case. However, as graph databases are designed with different types of queries in mind, these relational benchmarks are largely inadequate here. There has been a study [9] about the characteristics that a Graph database benchmark should include. Among those, the use cases to be considered, the operations that they give rise to, and the experimental settings are the most important for designing such benchmarks. Thus, Social Network management can be considered a representative use case, and graph queries like neighborhood extraction (within n steps or even unbounded neighborhoods), and structural similarity (like similarity between subgraphs or different small graphs in the data set) are good examples for such use case. All these operations do not exist in the RDBMS, and as a consequence not benchmarked there. On the other hand, the object oriented databases are quite similar to the graph databases (the objects and relations can be modeled with nodes and edges), and there are early proposals to benchmark them. One of them, the OO1 benchmark [8], generates the data representing one type of objects (nodes) with a fixed number of outgoing edges. The resulting graph is therefore very regular. The benchmark also includes three types of queries: (i) lookup of an object with a given identifier, (ii) traversal operation from a random node within a fixed number of hops, (iii) insertion of nodes and edges into the database. Another benchmark, coined the OO7 [7], describes three types of objects, organizing them as a tree of depth 7. Since object-nodes have a fixed number of edge-relations, the generated graph is also very regular as with the OO1 benchmark. However, the set of queries of OO7 is richer: they include the traversal operation and the selection queries that extract objects with specified attributes

The recent HPC Scalable Graph Analysis Benchmark [10], introduced by the graphanalysis.org, generates a graph according to the power-law distribution. It is equipped with four different queries, including bulk insert of nodes and edges, retrieval of the heaviest edge, k-hops operation, and calculating betweenness centrality. The performance of the last query is measured with edges traversed per second.

There are also some individual proposals for graph database benchmarking, but they lack universal acceptance or generality.

3.4 Instance matching and ETL benchmarking

Finding the matching instances (also referred to as record linkage, duplicate detection, entity resolution [4]) is a crucial and computationally expensive task in the Semantic Web domain. The existing benchmarks

[4] `http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/BusinessIntelligenceUseCase/`

mainly address evaluation of these techniques for XML and relational databases [14, 13]. As in the case of RDF benchmarks, the core features of RDF and ontologies (such as schema-free data, reasoning capabilities) are not covered. The Ontology Alignment Initiative (OAEI)[5] has proposed the most popular framework for ontology matching testing.

The Instance Matching Track of OAEI focuses on the evaluation of different instance matching techniques and tools for RDF and OWL datasets using a set of different benchmarks. The ARS, TDS and IIMB benchmarks were used for the OAEI IM track in 2009[6]. ARS is a benchmark that draws its data from the domain of scientific publications, TDS includes three datasets covering several topics and structured according to different ontologies. Finally, the IIMB[7] benchmark is generated using the ISLab Instance Matching Benchmark whose reference ontology is modified by applying a number of *value*, *structural* and *logical* transformations. The purpose of the benchmark focused on two main goals: i) to provide an evaluation dataset for various kinds of data trasformations and ii) to cover a wide spectrum of possible techniques and tools. The OAEI Instance Matching Track in 2010 proposed the Data Interlinking Task that was introduce to test the ability of systems to produce links in the Linked Open Data cloud and address the scalability dimension of instance matching systems. ONTOBI [25] is one of the most recent instance matching benchmarks and that take into account simple and complex transformations extending the ones proposed in IIMB for both schema and instance data. However, these modifications are done manually, and are not automated, as a large-scale benchmark would require.

Last, the STBenchmark [3] is a benchmark that takes as input a reference ontology, and applies several transformations to get the modified referemce ontology. It consists of two components: (a) a basic set of mapping scenarios and (b) a generator for mapping scenarios and source instances. The mapping scenarios refer to the different types of transformations and the generator for the mapping scenarios takes as input parameters related to the characteristics of the reference ontology, and produces the set of transformations that will be applied to the reference ontology. The main benchmark for relational ETL processes is TPC-ETL[8]. It has been proposed for comparing the performance of ETL systems, and provides a scalable workload that considers a wide range of dataset sizes, methodologies and metrics to compare different aspects of ETL systems. To the best of our knowledge, ETL benchmarks for RDF systems do not exist.

4 Main technical challenges

As benchmarks consist of datasets and tasks on the data, in this section we consider the technical challenges in these two big areas that LDBC benchmarks will encompass. The goal of LDBC here is to challenge the existing systems with problems which they cannot (yet) handle efficiently, and whose solution is possible but would require technical innovation. We will not present the concrete contests, but rather concentrate on the insights behind the design of good benchmarks.

4.1 Datasets

Most of the state-of-the-art benchmarks in RDF and Graph processing often require the data to be small enough to fit into the main memory. Moreover, current solutions typically only work fast on simple queries and without efficient reasoning or support for data integration tasks. To raise the bar in the technology and to push systems to handle very large datasets, we will present benchmarks that consider large-scale datasets and workloads that include complex queries.

The datasets typically used for benchmarking are either synthetic or coming from real-world data. For the query benchmarking purposes (as well as for transactional benchmarking) using generated datasets (with properties modeled after existing datasets) is beneficial since data generators allow to control the size and statistical properties of the data, thus making clear the technical challenges for systems.

The tasks of data integration, on the contrary, require using real world data, since such scenarios typically are focused on ”dirty”, highly irregular data, and cleaning up the data (i.e., removing duplicates, using a standard format) is part of the benchmarking task. Moreover the datasets should also consider dissimilarities not only at the instance but also at the schema level. The real datasets will be obtained in three ways: *(i)* collecting the publicly available ones, *(ii)* crawling the Web and *(iii)* as a result of the user provision.

In the initial stage of the project, LDBC plans to work with the publicly available datasets like the BBC publishing data, the Sindice Web Crawl, Billion Triple Challenge datasets, and with the synthetic dataset of the Social Intelligence Benchmark. The scale of the datasets will follow the one of the TPC datasets with the major difference that RDF is typically more verbose

[5] http://oaei.ontologymatching.org/
[6] http://oaei.ontologymatching.org/2009/
[7] http://islab.dico.unimi.it/iimb/
[8] `http://www.tpc.org/reports/status/default.asp`

than similar relational data, so a small RDF dataset will be for instance 12 billion triples (100 GB TPC-H data), and the large one will be 1.2 trillion triples (10 TB TPC-H data). Although the real data harvested from the Web will be much smaller than large-scale synthetic LDBC data (according to the latest diagram of the Linked Open Data cloud, there is more than 50 billion triples currently published online), the corresponding problems are already quite computationally expensive, such as schema alignment transformations or instance matching of a large number of objects.

4.2 Query Optimization Benchmarking

We are going to benchmark the RDF querying capabilities of systems by challenging them to do non-trivial algorithms for SPARQL query optimization. In this, the characteristic properties of the RDF data model which set it apart from the relational model (and thus require novel optimization techniques) will be used:

The RDF model is verbose. Unlike the relational model, where one record typically describes one entity, RDF data is more voluminous: one entity is described by several triples. Consider a simple SQL query over the table Person with all necessary attributes:

```
SELECT Name, Age, Gender
FROM Person
WHERE ID = 42
```

An equivalent SPARQL query will have the following form:

```
SELECT ?name ?age ?gender
WHERE {
   ?person <hasName> ?name.
   ?person <hasAge> ?age.
   ?person <hasGender> ?gender.
   ?person <hasID> "42"}
```

Since most of RDF systems store data in triple stores, executing this query will require performing three self-joins on variable `?person`, as opposed to a simple scan needed for a relational database on table Person to execute its SQL equivalent. For the query optimizer this means a rapid explosion of the plan search space. For example, a join of six tables becomes a join of eighteen tables (Q2 of TPC-H in SQL and SPARQL), and the number of possible plans grows from 6!=720 to 18!=6.4e15. This clearly calls for different efficient search space pruning techniques, and a good benchmark should stress it.

RDF data is highly correlated. Correlated data is a challenge already for the relational query optimizer. Usually, for simplicity the optimizer assumes that values of attributes are uniformly distributed, and values of two different attributes are independent, so that $Prob(A = a_1 \&\& B = b_1) = Prob(A = a_1) \cdot Prob(B = b_1)$. In reality, however, the values of attributes frequently depend on each other (the so-called *value correlations*): for example, the name and the country of origin of a person are strongly correlated. It has been shown that uniformity and independence assumptions lead to exponential growth of selectivity estimation errors when the number of joins in the query grows [12]. This effect only becomes more severe in RDF systems since the number of joins in the SPARQL query is larger than in the corresponding SQL query.

Another, RDF-specific, type of correlation is *structural correlation.* As an example, consider triple patterns (`?person`, <`hasName`>, `?name`) and (`?person`, <`hasAge`>, `?age`). Clearly, the two predicates <`hasName`> and <`hasAge`> almost always occur together (i.e., in triples with same subjects), so the selectivity of the join of these two patterns on variable `?person` is just the selectivity of any of the two patterns, say, 1e-4. The independence assumption would force us to estimate the selectivity of the join as $sel(\sigma_{P=hasName}) \cdot sel(\sigma_{P=hasAge})$=1e-8, i.e. to underestimate the size of the result by 4 orders of magnitude!

The combination of the two types of correlations is also quite frequent: Consider an example of triple pattern (`?person`, <`isCitizenOf`>, <`United_States`>) over the Yago dataset [24] . Now, the individual selectivities are as follows:

$$sel(\sigma_{P=isCitizenOf}) = 1.06 * 10^{-4}$$

$$sel(\sigma_{O=United\_States}) = 6.4 * 10^{-4}$$

while combined selectivity is

$$sel(\sigma_{P=isCitizenOf \wedge O=United\_States}) = 4.8 * 10^{-5}$$

We see that both P =<`isCitizenOf`> and O =<`United_States`> are quite selective, but their conjunction is in three orders of magnitude less selective than the mere multiplication of two selectivities according to the independence assumption. The value of the predicate (which corresponds to the "structure" of the graph) and the value of the object here are highly correlated for two reasons. First, the data in the English Wikipedia is somewhat US-centric, and therefore almost half of the triples with P=<`isCitizenOf`> are describing US citizens. Second, the <`isCitizenOf`> nearly requires the object to be a country, demonstrating a structural correlation between fields P and O.

Capturing the correlations for RDF databases is both important and more challenging than for relational systems: all explicit structure (like schema, metadata) present in the relational DBMS, are absent here, and the system has to infer such information from implicit correlations. The optimizer has to, consequently, take into account the correlations when estimating the cardinality of the (partial) plans. Therefore, the benchmark queries that address the issue of correlations will lead to advances in query optimization techniques.

Path traversals in SPARQL. Path queries specified with regular expressions on traversed predicates are part of SPARQL 1.1 standard. Note that, in principle, we would be able to translate this query into the standard SPARQL 1.0 [21] (and therefore into SQL) disjunctive query with union and several potentially long chain joins, if we knew the length of the paths and the exact schema in advance. However, this assumption contradicts the schema-free nature of RDF data, since it requires a priori knowledge about the path structure. The path queries are thus an intrinsic property of the RDF data model. As an example consider the following set of triples:

(Albert Einstein, bornIn, Ulm)
(Ulm, locatedIn, Baden-Württemberg)
(Berlin, locatedIn, Germany)

and the SPARQL query:

```
SELECT ?person
WHERE {
   ?person <bornIn> ?place.
   ?place <locatedIn>* <Germany> }
```

This query finds all the people born in Germany by matching the entities-places that can reach Germany via several instances of the <`locatedIn`> predicate (using the transitive closure on the predicate), and then finding the entities-persons that are related with the <`bornIn`> predicate to that entities-places.

Dynamic RDF databases. The issue of updates is entirely overlooked by existing RDF benchmarks. First, the bulk load time is of great interest, since efficient query processing usually requires heavy indexing, and time for creating and updating these indexes needs to be quantified. Second, there is a clear need for transaction support with full ACID guarantees, which have not been fully investigated by the RDF research community. Moreover, concurrent updates will greatly complicate query processing.

4.3 Graph Query Benchmarking

The following technical issues are of interest for the benchmark design in graph databases:

Query Language Currently there is no query language universally accepted by all graph databases. LDBC plans to review which languages have been proposed for graph database and their main features, to survey graph data access methods, and suggest the proper syntaxes for defining data and query workloads

Basic Operations There are operations like finding the diameter of the graph, betweenness, clustering coefficient, which currently are inexpressible in the graph query languages, but required by applications. The impedance mismatch that occurs between the programming language and the graph database API when implementing such operations, is a potential challenge for systems and should be exploited by the benchmarks.

Algorithms for disk-based graphs When dataset sizes grow and the workload touches the large share of data in the single query, then data is frequently read from the secondary storage. The benchmark should address this in order to encourage systems to make good use of memory hierarchy (e.g., main memory – SSD – disk). In many cases, an in-memory solution would be prohibitively expensive, although would deliver a good throughput. Using the cost/throughput metric will rule out such solutions.

4.4 Reasoning and integration benchmarking

Current RDF systems support some level of reasoning, typically concentrating on extended RDFS [6] and some OWL [16] primitives (e.g. subClassOf, subPropertyOf, sameAs, sameIndividualAs, equivalentProperty. equivalentClass, partOf), but at the same time this issue is largely overlooked by existing benchmarks. The experiments suggest that for such limited reasoning *forward chaining*, that is the materialization of all inference results is practical in that it has a reasonable effect on database size.

An open question, however, is what happens when the user switches to scenario-dependent reasoning rules that are not necessarily expressed as RDFS/Horst rules. These rule-sets may be formulated in terms of OWL, but may just as well exceed it: for example, how does the level of social connectedness correlate with the frequency of product returns or negative product reviews. Here rules are expressed sometimes as recursive counts and sums of events and the queries touch a large fraction of the data. In this situation it may no longer be affordable to pre-compute the results of reasoning, and some combination of backward and forward reasoning will perform better. Benchmarks should therefore use *extended reasoning* rules to demonstrate trade-offs of backward and forward chaining.

RDF use cases often involve enriching existing data with public data from Linked Open Data cloud. This process of creating and curating links is highly labor intensive and there are no metrics or benchmarks that quantify the progress. Two types of metrics to measure in this scenario are qualitative (based on *precision* and *recall*) and quantitative (*speed*). As part of integration problems, the support for RDF ETL tool-chain will be also investigated, including the integration of existing datasets with public LOD sources.

5 Use-case scenarios

To get a plausible set of queries from the users prospective, we turn our attention to two scenarios: semantic publishing and social network analysis.

5.1 Semantic Publishing

The basic idea of semantic publishing to help journalists/editors/staff spend more of their time concentrating on content and less time editing/designing web pages. Here, the content produced and tagged by journalists and editors will be rendered on a web page automatically and presented in the proper context, using semantic annotations among media items coupled with the reasoning capabilities of RDF stores.

The characteristics often found in semantic publishing scenarios are:

- Large datasets
- Moderately stable and well-structured publishing-specific ontologies
- Stream of updates with dense spikes at certain times (both inserts and deletes), with updates running concurrently
- Lightweight inference (expressivity equivalent to RDFS with some OWL primitives)

A typical workload will include CRUD operations on objects (articles and other journalistic assets), where objects are read much more frequently than updated. The object metadata connects different articles via a tagging ontology and domain specific ontologies.

The LDBC Technical User Community (TUC) hosts among others the BBC who has agreed to contribute its datasets and workloads in order to fully define this scenario.

5.2 Social Network Analysis

The analysis of online data based on activities of users in social networks plays an important role to detect trends in the use of products or opinions about the quality of a certain brand. Marketing companies now analyze for their customers how information propagates in different integrated social networks like Twitter, blogs and on-line media. The objective is to understand the roles of people in those networks like who are initiators of information, who are followers, etc. This information is precious for the purpose of knowing those who have a strong influence with their messages, either for positive or negative reasons. The actions to be taken in those cases range from the pure in kind incentive to an initiator of positive information, to the removal of an advertising campaign in case of the detection of negative information by an influential person in the network, or a rapid propagation of a negative comment.

In all those cases, the analysis of the individuals, their interactions and the propagation of the information they inject in the network calls for the use of graph technology. In some cases, there is a need to evaluate the information network in a very fast way, like in the detection of positive/negative information. The marketing companies are interested in a benchmarking effort of the type proposed in this project for the purpose of finding the fastest possible graph database management for this type of on-line social network analysis.

We are planning to construct the analytical queries on top of the Social Intelligence Benchmark [20].

6 Linked Data Benchmark Council

The goal of the LDBC effort is to establish the Linked Data Benchmark Council as a non-profit organization and make it a successful, lasting organization, supported by the RDF and graph database industry. Broad industry participation is an important goal of the project, and the letters of support from industry partners give an indication that it is achievable.

Once LDBC is successful, commercial and marketing concerns will start to play an important role in how the various member organizations interact with it. For this purpose, it is imperative that LDBC is designed such that it is capable of handling conflict situations in an orderly and fair way. For this we look at the example of the Transaction Processing Council (TPC), which has a firm statutory basis and democratic processes and committee structures in place to handle such situations.

Auditing Process It is anticipated that vendors will benchmark their products on their own premises using the optimum hardware and software deployment for their products and this is in keeping with established best practice. The typical process for publishing a benchmark result would be as follows:

– A vendor decides to run the benchmark.
– After the vendor has internally experimented with provisioning the system under test with appropriate hardware/software and tuning it accordingly and is satisfied with the results, the vendor may decide to publish.
– For the benchmark to be published, the run should be audited by an auditor that was previously certified by LDBC. In practice, this need would be met by the auditor having remote login to the system under test for the duration of the test.
– The result would be published with all relevant documentation by LDBC on its portal.
– In the event of results being challenged, the LDBC foundation can arbitrate in the matter, with all proceedings being on public record.

Many TPC benchmark results are obtained on equipment that would be unlikely in practical deployments, i.e. unusually large configurations. Also with LDBC, our approach is to allow vendors to benchmark on the equipment of their choice, and similarly to the TPC, the benchmarks will allow for this with an appropriate metric, i.e. something similar to a cost-per-transaction.

7 Conclusions

In this paper we presented the Linked Data Benchmarking Council, the initiative towards RDF/Graph benchmarking. We would like to invite the readers to join this community initiative by sharing their user experience, testing their systems and participating in the LDBC-related events[9]. We hope that, similarly to relational databases, adequate benchmarking will advance research in many aspects of RDF and Graph data management.

References

1. The DBLP Computer Science Bibliography. `http://www.informatik.uni-trier.de/~ley/db/`.
2. D. J. Abadi, A. Marcus, and B. Data. Scalable semantic web data management using vertical partitioning. In *VLDB*, 2007.
3. B. Alexe, W.-C. Tan, and Y. Velegrakis. STBenchmark: Towards a benchmark for mapping systems. In *PVLDB*, 2008.
4. I. Bhattacharya and L. Getoor. *Entity resolution in graphs. Mining Graph Data.* Wiley and Sons, 2006.
5. C. Bizer and A. Schultz. The Berlin Sparql Benchmark. *International Journal On Semantic Web and Information Systems*, 2009.
6. D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. `www.w3.org/TR/2004/REC-rdf-schema-20040210`, 2004.
7. M. J. Carey, D. J. DeWitt, and J. F. Naughton. The OO7 Benchmark. In *SIGMOD*, 1993.
8. R. G. G. Cattell and J. Skeen. Object operations benchmark. *ACM TODS*, 1992.
9. D. Dominguez-Sal, N. Martínez-Bazan, V. Muntés-Mulero, P. Baleta, and J.-L. Larriba-Pey. A Discussion on the Design of Graph Database Benchmarks. In *TPCTC*, 2010.
10. D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey. Survey of graph database performance on the HPC scalable graph analysis benchmark. In *WAIM*, 2010.
11. S. Harris and A. Seaborne. SPARQL 1.1 Query Language. `http://www.w3.org/TR/sparql11-query/`, November 2012. W3C Proposed Recommendation.
12. Y. E. Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. In *SIGMOD*, 1991.
13. A. Isaac, L. Van Der Meij, S. Schlobach, and S. Wang. An empirical study of instance-based ontology matching. In *ISWC/ASWC*, 2007.
14. H. Köpcke, A. Thor, and E. Rahm. Comparative evaluation of entity resolution approaches with FEVER. *VLDB Endowment*, 2(2), 2009.
15. L. Ma, Y. Yang, Z. Qiu, G, Xie, Y. Pan, and S. Liu. Towards a Complete OWL Ontology Benchmark. In *ESWC*, 2006.
16. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language. `http://www.w3.org/TR/owl-features/`, 2004.
17. M. Morsey, J. Lehmann, S. Auer, and A.-C. Ngonga Ngomo. DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In *ISWC*, 2011.
18. T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, 19(1), 2010.

[9] `http://ldbc.eu/events`

19. H. Patni, C. Henson, and A. Sheth. Linked sensor data. In *CTS*, 2010.
20. M.-D. Pham, P. Boncz, and O. Erling. S3G2: a Scalable Structure-correlated Social Graph Generator. In *TPCTC*, 2012.
21. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. `www.w3.org/TR/rdf-sparql-query`, January 2008.
22. N. Redaschi and U. Consortium. UniProt in RDF: Tackling Data Integration and Distributed Annotation with the Semantic Web. In *Biocuration Conference*, 2009.
23. M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel. Sp2bench: A SPARQL performance benchmark. In *ICDE*, 2009.
24. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.
25. K. Zeiss, S. Vater, and S. Conrad. A benchmark for testing instance-based ontology matching methods. In *IEKAW*, 2010.