The Linked Data Benchmark Council: a Graph and RDF industry benchmarking effort

Renzo Angles[1,2], Peter Boncz[3], Josep Larriba-Pey[4], Irini Fundulaki[5], Thomas Neumann[6], Orri Erling[7], Peter Neubauer[8], Norbert Martinez-Bazan[9], Venelin Kotsev[10], Ioan Toma[11]
[1]Universidad de Talca, Chile; [2]VU University Amsterdam, The Netherlands;
[3]CWI, Amsterdam, The Netherlands; [4]Universitat Politècnica de Catalunya, Spain;
[5]FORTH, Heraklion, Greece; [6]Technische Universität München, Munich, Germany;
[7]OpenLink Software, United Kingdom; [8]Neo Technology, Sweden;
[9]Sparsity Technologies, Spain; [10]Ontotext AD, Bulgaria; [11]Universität Innsbruck, Austria
rangles@utalca.cl, boncz@cwi.nl, larri@ac.upc.edu, fundul@ics.forth.gr,
neumann@in.tum.de, oerling@openlinksw.com, peter.neubauer@neotechnology.com,
norbert@sparsity-technologies.com, venelin.kotsev@ontotext.com, ioan.toma@sti2.at

ABSTRACT

The Linked Data Benchmark Council (LDBC) is an EU project that aims to develop industry-strength benchmarks for graph and RDF data management systems. It includes the creation of a non-profit LDBC organization, where industry players and academia come together for managing the development of benchmarks as well as auditing and publishing official results. We present an overview of the project including its goals and organization, and describe its process and design methodology for benchmark development. We introduce so-called "choke-point" based benchmark development through which experts identify key technical challenges, and introduce them in the benchmark workload. Finally, we present the status of two benchmarks currently in development, one targeting graph data management systems using a social network data case, and the other targeting RDF systems using a data publishing case.

1. INTRODUCTION

Graph and RDF databases have been created to support the storing and analysis of complex relationships in highly interconnected data occurring in application domains like e.g. social network analysis, linked open data, etc. Both data management technologies hold the notion of graph as their main abstraction mechanism for data modeling and querying. These technologies are relatively young, certainly in their tradition of industry-strength products supporting this graph data model, yet have aroused significant interest from users looking beyond the relational model. In order for practitioners to compare these new data management systems with each other, and with established relational technology, benchmarks can play a helpful role. Relevant benchmark challenges can further help spur technological progress, to more quickly mature these nascent industries.

Current graph and RDF benchmarks, however, do not fully attain all the desirable characteristics [5] (i.e., relevant, repeatable, fair, verifiable and economical), and sometimes neglect the particularities and requirements in RDF and graph data management [2, 3, 4, 7] (e.g. complex graph queries over irregularly shaped and correlated data).

The Linked Data Benchmark Council (LDBC)[1] is an EU project that brings together a community of academic researchers and industry, whose main objective is the development of open source, yet industrial grade, benchmarks for graph and RDF databases. The founding industry members of LDBC are the graph database companies Neo Technologies and Sparsity Technologies, and the RDF database companies Ontotext and OpenLink Systems. A result of the project will be the LDBC non-profit organization, open for worldwide industry participation, which during and after the end of the EU project will supervise the creation and maintenance of the benchmarks as well as the activities for obtaining, auditing and publishing the benchmarking results.

This paper describes the goals of the LDBC as well as its organizational structure. It also presents the status of two benchmarks in current development: a graph data management benchmark based on the social network use-case, and an industry-strength RDF benchmark for semantic data enrichment based on a real-life semantic publishing use case (the BBC semantic publishing platform).

[1]Linked Data Benchmark Council is EU project FP7-317548 (see `http://ldbc.eu`). Renzo Angles was funded by Fondecyt Chile grant 11100364.

In this paper we also describe a process for developing benchmarks based on technical challenges called "choke points", developed by LDBC. This methodology depends on a combination of workload input by end users, and access to true technical experts in the architecture of the systems being benchmarked. The overall goal of the choke-point based approach is to ensure that a benchmark workload covers a spectrum of technical challenges, forcing systems onto a path of technological innovation in order to score good results.

2. ORGANIZATIONAL STRUCTURE

The LDBC is organized in three platforms:

The board of LDBC members. This is formed by one representative (director) per each member organization of LDBC with voice and one single vote per organization in their assembly. Meetings of the directors make all policy decisions, though certain decisions, among which the adoption of new benchmarks, is handled through a written vote and requires an absolute majority of all members.

Technical User Community (TUC)[2]. It is an open organization that brings together users of RDF and graph technologies, researchers, industry participants, and delegates of the LDBC members in physical events (TUC meetings) to discuss about possible benchmark use cases and scenarios and to assess the quality of the benchmark proposals and the adequacy to their needs. LDBC organizes multiple TUC meetings per year, providing logistics and travel assistance to external invitees.

Task Forces. A Task Force is an internal LDBC structure that carries the development of a benchmark from beginning to end. It is formed by experts from member organizations of LDBC, and it works on the proposal, creation of the use case and scenario based on the TUC discussions, choke points suggested by technical experts and the implementation of the different parts of the benchmark (e.g. data and workload generation).

3. DEVELOPMENT METHODOLOGY

The development of a benchmark in LDBC is initiated by board decision, designed and constructed by a task force, and supported by TUC input and feedback. This process results in the creation of four main elements: (1) the data schema, which defines the structure of the data used by the benchmark; (2) the workload, which defines the set of operations that the system under benchmarking has to perform during the benchmark execution; (3) performance metrics, which are used to measure (quantitatively) the performance of the systems; and (4) execution rules, which are defined to assure that the results from different executions of the benchmark are valid and comparable.

The final result, a benchmark specification, consists of both textual documentation and - insofar possible - a standard implementation (i.e. data generator, workload generator and test driver). LDBC software is open source, and is disseminated through GitHub (see `https://github.com/ldbc`).

Choke Point based Design. Literature until now has described the technical work required when designing a good database system benchmark in relatively vague terms. LDBC intends to formalize some of the best practices and raise the state-of-the-art in this area, in its guidelines for benchmark development.

On the surface, a benchmark models a particular scenario, and this should be believable, in the sense that users of the benchmark must be able to understand the scenario and believe that this use-case matches a larger class of use cases appearing in practice. On a deeper – technical – level, however, a benchmark exposes technology to a workload. Here, a benchmark is valuable if its workload stresses important technical functionality of actual systems. This stress on elements of particular technical functionality we call "choke points". To understand benchmarks on this technical level, intimate knowledge of actual system architectures is needed. The LDBC consortium was set up to gain access to those architects of the initial LDBC industry members, as well as to the architects of database systems Dex[3], Neo4j[4], Virtuoso[5], RDF-3X[6], Hyper[7], MonetDB[8] and Vectorwise[9].

LDBC authors [1] analyzed the relational TPC-H benchmark in terms of 28 different choke points; providing both a good illustration of the choke point concept, and an interesting to-do list for those optimizing a system for TPC-H. Specific examples among those 28 are choke points like exploiting functional dependencies in group-by, foreign-key joins with a low match ratio (to be exploited by e.g. bloom filters), and discovering correlation among key attributes in a clustered index (e.g. using zone maps).

Choke points can be an important design ele-

[2] `http://www.ldbc.eu:8090/display/TUC/`

[3] `http://www.sparsity-technologies.com`

[4] `http://www.neo4j.org`

[5] `http://virtuoso.openlinksw.com`

[6] `https://www.mpi-inf.mpg.de/~neumann/rdf3x/`

[7] `http://hyper-db.de`

[8] `http://www.monetdb.org`

[9] http://www.actian.com/vectorwise

ment during benchmark definition. The technical experts in a task force identify choke points relevant for a scenario, and document these explicitly. Subsequently, as the benchmark workload evolves during the process of its definition, a close watch is kept on which queries in the workload test which choke point to which extent, aiming for complete coverage using a limited amount of queries. Choke points thus can ensure that existent techniques are present in a system, but can also be used to reward future systems that improve performance on still open technical challenges.

3.1 Phases of the design process

Analysis. This phase is oriented to determine the requirements of the benchmark based on the analysis of the application domain, workload characterization, and selection and definition of choke points. The workload characterization abstracts the selected real-life scenario into a basic data schema, identifying the typical data structures found and their relationships. Workload requirements are derived from captured static and dynamic behavior of real workloads (e.g. obtained from members or through the TUC). The identification of choke points comes from the architectural analysis in existing systems from expert knowledge.

Design. In this phase, a detailed schema is formulated, including design of attribute value distributions and correlations, and join connectivity between different schematic elements. Also, the workload is fleshed out into concrete sets of queries with example parameter bindings. A check is made on which of the queries in the workload hit which choke points, ensuring that all are covered. Finally, we define the metrics for measuring performance, and rules for benchmark execution and result reporting.

Implementation. In this phase, the needed software tools are designed and implemented, particularly the data generator, the workload generator, and drivers for one or more systems. Data generators must conform to certain minimum standards, for instance, in case of benchmarks at scale, these should be designed with parallelism in mind. Special properties and relationships in the data must be specified and implemented (e.g. data consistency, data distributions and correlations).

The workload generator needs to chose substitution parameters for the operations of the workload. Proper selection of substitution parameters is steered by the data distributions and their correlations as generated by the test driver, and by the choke points behind the individual queries in the workload, and may require post-generation dataset analysis. An important point is that different substitution parameters for one query should always lead to (roughly) the same data access and execution characteristics (in terms of e.g. cardinalities, locality and optimal query plans) such that behavior is stable and understandable.

Testing. With the ability to run the workload on one or more existing systems comes the task of testing it. A basic aim is *verification* to ensure that the implementation yields the correct and intended results. A second aim is *validation* by measuring not only the performance, but all relevant quantitative and qualitative features of the benchmark (cardinalities, query plans, operators used, detailed performance profiles). This experimentation thus provides insight in how far the benchmark indeed tests the choke points that were targeted.

Considering that the data generator, the workload generator and the methods for substitution parameter generation are mutually dependent, benchmark development is by necessity iterative, such that we may fall back to the Design stage to refine these aspects.

Distribution. This phase is oriented to prepare the benchmark for wider usage. It consists of cleaning up the design and implementation to include only the relevant pieces of software. It includes all the operations to package the test drivers, datasets and/or data generators as well as documentation (including execution, auditing and reporting rules).

4. ONGOING DEVELOPMENT

4.1 Social Network Benchmark

The Social Network Benchmark (SNB)[10] is designed for evaluating a broad range of technologies for tackling graph data management workloads. The systems targeted are quite broad: from graph, RDF, and relational database systems to Pregel-like graph programming frameworks.

The scenario of the benchmark, a social network, is chosen with the following goals in mind: it should be understandable to a large audience, and this audience should also understand the relevance of managing such data; the scenario in the benchmark should cover the complete range of interesting challenges, according to the benchmark scope; and the query challenges in it should be realistic in the sense that, though synthetic, similar data and workloads are encountered in practice.

[10] `http://www.ldbc.eu:8090/display/TUC/Social+network+benchmark+task+force`

SNB includes a data generator[11] that enables the creation of synthetic social network data with the following characteristics: the data schema is representative of a real social network (see Figure 1); the data generated includes properties occurring in real data, e.g. irregular structure, structure/value correlations and power-law distributions; and the software generator is easy-to-use, configurable and scalable.

The requirement to generate at scale a complex social graph with special data distributions that at the same time exhibits certain interesting value correlations (e.g. German people having predominantly German names) and structural correlations (e.g. friends being mostly people living near), poses an interesting challenge. The SNB data generator builds on the work on correlated social network generation in S3G2 [6], whose source code has been adapted to the SNB data schema. S3G2 comes with the ability to leverage parallelism through Hadoop, ensuring fast and scalable generation of huge datasets.

SNB is intended to cover all main aspects of social network data management, and therefore splits into three separate workloads:

– **Interactive workload.** This workload tests system throughput with relatively simple queries and concurrent updates. The workloads test ACID features and scalability in an online operational setting. Given the high write intensity, this workload may also be used to let the dataset grow, which will be implemented by pre-generating data in the generator but only importing the data corresponding to one time point in the bulk load, and playing out the rest of the modifications in the update workload.

– **Business intelligence workload.** This workload consists of complex structured queries for analyzing online behavior of users for marketing purposes. The workload stresses query execution and optimization. The targeted systems are expected to be those that offer an abstract query language. Queries typically touch a large fraction of the data and do not require repeatable read.

– **Graph Analytics Workload.** This workload tests the functionality and scalability of the systems for graph analytics that typically cannot be expressed in a query language. The analytics is done on most of the data in the graph as a single operation and produces large intermediate results. The analysis is not expected to be transactional or need isolation. This workload targets graph programming frameworks, though systems with a query-language might compete using iterative implementations that repeatedly fire queries and keep intermediate results in temporary data structures.

A benchmarked system does not need to run all workloads. Each workload in SNB produces a single metric for performance at the given scale and a price/performance metric at the scale.

4.2 Semantic Publishing Benchmark

The Semantic Publishing Benchmark (SPB)[12] simulates the management and consumption of RDF metadata that describes media assets, or creative works. The scenario is a media organization that maintains RDF descriptions of its catalogue of creative works (e.g. from the BBC). The benchmark is designed to reflect a scenario where a large number of aggregation agents provide the heavy query workload, while at the same time a steady stream of creative work description management operations are in progress. This benchmark plainly targets RDF database systems, which support at least basic forms of semantic inference.

The RDF descriptions of this benchmark use an ontology that defines numerous properties for content, for example: date of creation, short/long descriptions, etc. Furthermore, a tagging ontology is used to connect individual creative work descriptions to instances from reference datasets, including sports, geographical, or political information. The data used will fall under the following categories: reference data, which is a combination of several Linked Open Data datasets, e.g. GeoNames and DBpedia; domain ontologies, that are specialist ontologies used to describe certain areas of expertise of the publishing, e.g., sport and education; publication asset ontologies, that describe the structure and form of the assets that are published, e.g., news stories, photos, video, audio, etc.; and tagging ontologies and the metadata, that links assets with reference/domain ontologies.

The data generator is initialized by using several ontologies and datasets. The instance data collected from these datasets are then used at several points during the execution of the benchmark. Data generation is performed by generating SPARQL fragments for create operations on creative works and executing them against the RDF database system.

Two separate workloads are modeled in SPB:

– **Editorial workload.** It simulates creating, updating and deleting creative work metadata descriptions. Media companies use both manual and semi-automated processes for efficiently and correctly managing asset descriptions, as well as annotating them

[11] https://github.com/ldbc/ldbc_socialnet_bm/tree/master/ldbc_socialnet_dbgen

[12] http://www.ldbc.eu:8090/display/TUC/Semantic+Publishing+Task+Force

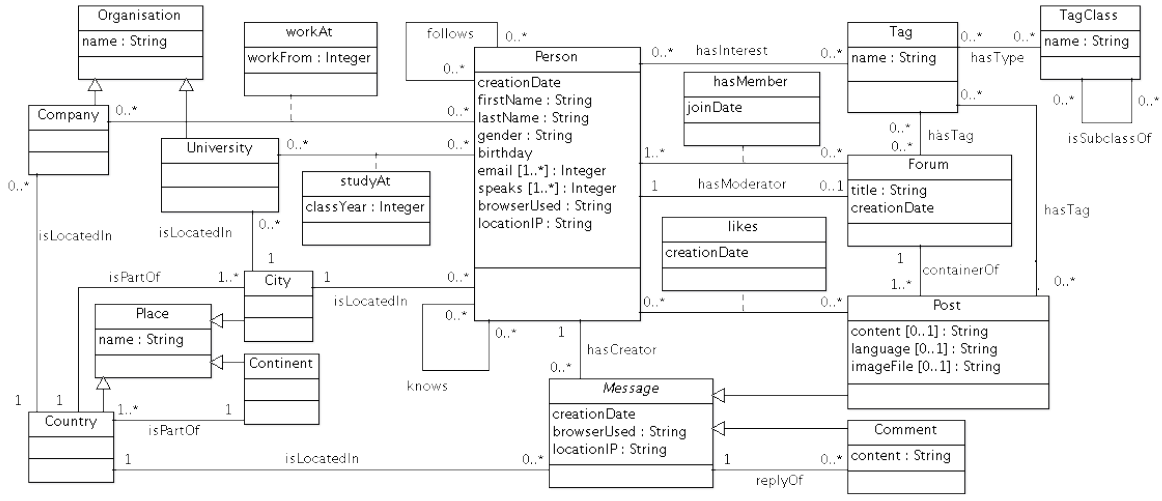

Figure 1: The data schema of the Social Network Benchmark represented in UML.

with relevant instances from reference ontologies.

– **Aggregation workload.** It simulates the dynamic aggregation of content for consumption by the distribution pipelines (e.g. a web-site). The publishing activity is described as "dynamic", because the content is not manually selected and arranged on, say, a web page. Instead, templates for pages are defined and the content is selected when a consumer accesses the page. In this workload, SPARQL queries are used to find relevant content.

Measurement in SPB is performed on the update/retrieve rate of queries executed by editorial and aggregation agents for a fixed amount of time. Metrics describe the queries per second rate that each RDF database system is capable to sustain during the benchmarking period.

5. CONCLUSIONS

In this paper we presented the Linked Data Benchmark Council (LDBC), a new initiative towards for benchmarking Graph and RDF data management systems. LDBC aims to bring some of the best practices of the TPC to the small but growing graph and RDF database industry. A main technical advance is its "choke point" driven benchmark design, which ensures that interesting and well-chosen technical challenges will emerge from implementing the benchmarks. The LDBC currently has two benchmarks under development by its "task forces": the Social Network Benchmark (SNB) and the Semantic Publishing Benchmark (SPB). The latter focuses on testing RDF database systems, whereas the former actually splits into three different sub-benchmarks (the interactive workload, the BI workload, and the graph analytics workload) that all work on a shared dataset. This benchmark thus targets graph, RDF and relational database systems, as well as graph programming frameworks.

We hereby invite the reader to join the Technical User Community (TUC) to influence the LDBC.

6. REFERENCES

[1] P. Boncz, T. Neumann, and O. Erling. TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark. In *TPCTC*, 2013.

[2] D. Dominguez-Sal, N. Martinez-Bazan, V. Muntes-Mulero, P. Baleta, and J. L. Larriba-Pey. A Discussion on the Design of Graph Database Benchmarks. In *TPCTC*, 2010.

[3] S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In *SIGMOD*. ACM Press, 2011.

[4] Y. Guo, A. Qasem, Z. Pan, and J. Heflin. A Requirements Driven Framework for Benchmarking Semantic Web Knowledge Base Systems. *TKDE*, 19(2):297–309, 2007.

[5] K. Huppler. The Art of Building a Good Benchmark. In *TPCTC*, August 2009.

[6] M.-D. Pham, P. A. Boncz, and O. Erling. S3G2: A Scalable Structure-Correlated Social Graph Generator. In *TPCTC*, 2012.

[7] T. Weithöner, T. Liebig, M. Luther, and S. Böhm. What's Wrong with OWL Benchmarks. In *SSWS*, 2006.