BowlognaBench—Benchmarking RDF Analytics

Gianluca Demartini[1], Iliya Enchev[1], Marcin Wylot[1]
Joël Gapany[2], and Philippe Cudré-Mauroux[1]

[1] eXascale Infolab
[2] Faculty of Humanities
University of Fribourg, Switzerland
`firstname.lastname@unifr.ch`

Summary. The proliferation of semantic data on the Web requires RDF database systems to constantly improve their scalability and efficiency. At the same time, users are increasingly interested in investigating large collections of online data by performing complex analytic queries (e.g.,"how did university student performance evolve over the last 5 years?"). This paper introduces a novel benchmark for evaluating and comparing the efficiency of Semantic Web data management systems on analytic queries. Our benchmark models a real-world setting derived from the Bologna process and offers a broad set of queries reflecting a large panel of concrete, data-intensive user needs.

1 Introduction

The amount of semantic data available on the Web is rapidly increasing due to successful initiatives such as the Linked Open Data movement[1]. At the same time, semantic knowledge bases are being developed and improved in order to face the pressing demand of storing and querying large scale datasets. As examples, the semantic search engine Sindice[2] indexes up to 200M RDF documents, while very large datasets containing billions of RDF triples[3] are increasingly common. The bigger the datasets grow in size, the more interesting it becomes to perform complex analysis on RDF data, in order for example to identify patterns or discover data correlations. This is somehow similar to OLAP (On-Line Analytical Processing) transactions on large databases where complex read-mostly queries are run over big datasets.

Modern business processes generate much data which is often stored for further analysis or even for legal purposes. Such data can be seen an important asset for a company willing to improve their processes by means of data analysis. One of the current open challenges that limits such analysis is the poor performance of data management platforms. Specifically, for the case of graph data (of which business process data is a good example) much improvement can be obtained.

[1] `http://linkeddata.org/`
[2] `http://sindice.com/`
[3] `http://km.aifb.kit.edu/projects/btc-2010/`

In order to foster such efficiency and scalability advances that address the need for business process analysis, in this paper[4], we propose a benchmark for evaluating and comparing the efficiency of knowledge management systems focusing in particular on *analytic queries* over RDF data. Other benchmarks for RDF storage systems have been proposed already—the most popular one being LUBM [9], which provides an ontology describing universities together with an automatic instance generator and a set of 14 simple queries. Extensions of such benchmarks have also been proposed. For example, Minack *et al.* [11] created a benchmark for full-text search over semantic repositories. Further RDF benchmarks include a benchmark over full-text data (see, for instance, [3, 13]), a benchmark based on scientific publications (that is, DBLP) [14], and a benchmark based on a Web data exploration application [1]. Recently, the DBpedia SPARQL Benchmark [12] has also been proposed: it consists of a RDF store benchmark which focuses on evaluating system performance over real queries and real data. It contains queries mined from real logs of the DBPedia system[5] and resembles original DBPedia data.

Why is there a need for a new RDF benchmark? The simple answer is that none of the aforementioned benchmarks focuses on complex analytic queries. Analytic operations have become a major concern of any modern company, as they are used to measure and improve the performance of many complex operations (from data warehousing to reporting or data mining). Traditional relational systems were designed and optimized for transactional queries mostly, and cannot handle analytic queries efficiently. This recently led to many advances in data management technologies with the emergence of companies like Teradata[6] or, more recently, Vertica[7], and the definition of new benchmarks, such as TPC-H[8] (which is the standard benchmark for data analytics today) or the recent benchmark we suggested for mixed OLAP/OLTP settings [8]. In the university context we can expect relatively low volumes of *insert* and *update* queries (as compared, for example, to an e-commerce scenario). For this reason, we thought that it would be interesting to focus on analytic and trend queries rather than supporting high volume of transactions. To the best of our knowledge, BowlognaBench is the first RDF benchmark focusing on this topic.

Beyond its focus on data analytics, we feel that BowlognaBench is based on a more realistic ontology than previous attempts. Duan *et al.* [6] recently proposed a comparison between available RDF benchmarks and real datasets such as DBpedia [2] and YAGO [15]. Their study showed that existing benchmarks have little in common with real world datasets. This finding confirms our motivation for creating a novel benchmark for RDF that better reflects user information needs. LUBM, the most popular RDF storage benchmark, does not directly reflect a real university setting, while the ontology we propose models

[4] This paper is an extended version of [4].
[5] `http://dbpedia.org/`
[6] `http://www.teradata.com/`
[7] `http://www.vertica.com/`
[8] `http://www.tpc.org/tpch/`

an academic setting as prescribed by the Bologna process. The second drawback of LUBM is that its set of queries tries to cover possible failures in knowledge management systems and advanced (but rather rare) inference cases, rather than model real user information needs. We develop a set of queries of real interest to the academic domain (e.g., "what is the percentage of students who continue their university studies after the Bachelor?"). Third, the LUBM benchmark does not involve the temporal dimension, which is a very important aspect of analytic queries; in our benchmark, we propose to evaluate queries asking for specific time or time intervals over the data (e.g., "how did student performance evolve over the last 5 years"?), which is a popular type of query that need to be supported by current systems (see for example [10]).

In summary, BowlognaBench provides a framework for evaluating the performance of RDF systems on a real-world context derived from the Bologna process; it strains systems using both analytic and temporal queries; and it models real academic information needs.

The rest of the paper is structured as follows. In Section 2 we briefly describe the Bologna reform of the European university studies. Then, in Section 3 we present the lexicon of terms relevant to the Bologna process that has been defined in order to avoid ambiguities and misunderstandings. Next, in Section 4 we introduce the Bowlogna Ontology that has been developed to model the university setting after the Bologna reform. Together with the set of queries defined on top of the Bowlogna Ontology in Section 5, we propose the Bowlogna Benchmark to evaluate and compare efficiency of knowledge bases for analytic queries. In Section 6 we present an experimental comparison of different systems over the proposed benchmark. Finally, we conclude the paper in Section 7.

2 Information Systems in The Bologna Process

The Bologna Process is the set of reforms into which European higher education institutions entered from June 1999 onward, after the ministries of education of 29 European countries convened in Bologna and laid new foundations for the future European higher education. The Bologna Declaration defines a general framework in which signatory countries commit themselves to develop their higher education system. As such, it also leaves a certain margin for interpretation, which has allowed local appreciation and various implementations of the process. As a consequence, during the last decade, a specific form a relationship between universities has unquestionably developed, according to which both cooperation and competition in teaching and research are equally fostered.

In this context, the need to make available (for example, in RDF form) comprehensive and transferable datasets about current students and course offerings is increasing dramatically, as a key dimension of quality processes. In this competitive environment also, the capacity to attract new students, especially at the master level, is generally regarded as a factor of success. It is therefore not surprising that universities invest considerable efforts into the development of new information infrastructures, and into increasingly sophisticated analytic tool to help them monitor and assess their performance.

3 The Bologna Linguistic Lexicon

The Bologna Process initiated a radical change within higher education institutions. This change encompasses the creation of new administrative procedures, inclusive quality procedures, in the every day life of the universities. It also gave rise to the emergence of new concepts for the description of curricula. In this respect, we recently published [7] in the Bologna Handbook the basic set of entities that correspond to the new system of studies and that seem to be rather stable across time and institutions. In this paper, we described how the compilation of a lexicon of about 60 definitions contributed to the implementation of the Bologna Process in our institution. This booklet has been spread among academic and administrative staff, in order to create a common understanding of Bologna. It is worth noting that the defined entities are concepts, rather than single words, and that they were selected and defined by professional terminologist who adopted a rigorous, text-based approach to study the Bologna Process.

4 The Bowlogna Ontology

The formal lexicon described above paved the way for an ontology-based computerization of the Bologna process. Based on our understanding of the Bologna and on the Bologna lexicon, we developed an OWL ontology describing all the relevant concepts and their relations[9]. Figure 1 shows some of the classes in the Bowlogna ontology together with their relations. The ontology contains 29 top level classes (67 in total) describing concepts like students, professors, student evaluations (i.e., exams where students get a grade from a lecturer in the context of a teaching unit), teaching units (i.e., courses given by lecturers and attended by students), ECTS credits, as well as formal documents such as, for instance, study programs, certificates, or grade transcripts. Additionally, all definitions from the lexicon, both in German and French, are included in the ontology.

One of the key classes in this ontology is "Evaluation" in which a Student, a Professor and a Teaching Unit are involved. This class models the event when a student, after attending a course given by a professor (i.e., a teaching unit) is evaluated and given a grade. Properties of teaching units include the semester and the language in which the units are given, as well as the number of corresponding ECTS credits earned by students who successfully follow the course. The class Student also has a number of properties including the name of the student, his enrollment or graduation date for Bachelor and Maser degrees. The same student instance can be, over time, registered for different academic degrees (e.g., first a Bachelor and then a Master). The class "ECTS credit" is linked

[9] For a comprehensive description of the Bowlogna ontology see [5].

to all classes directly or indirectly related to the measurement of students advancement: for example, teaching units and study programs. The class Semester, with start and end dates, enables queries on the temporal dimensions (e.g., the number of students enrolled at the university during a certain semester).

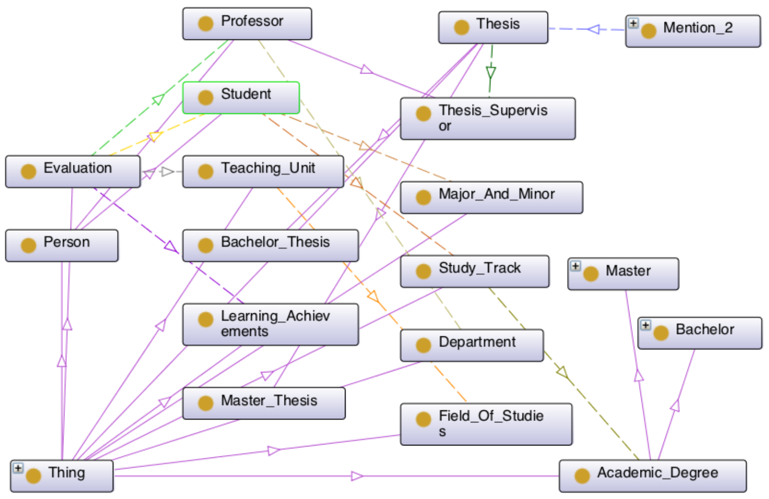


Fig. 1. The Bowlogna ontology: key classes and relations.

4.1 Public and Private Ontology Parts

The Bowlogna ontology can be divided in two important parts according to the type of information stored. Some information are public and can be shared with other universities as well as with the general public. Examples of such public information include the departments, the teaching units together with information about their ECTS credits and teaching language. The second part of the ontology consists of information which should not be publicly available, such as grades given to students. In real instantiations of the ontology, the private part will contain many more instances as compared to the public part. It is also clear that aggregations over private information might often be of general interest (e.g., the number of students currently enrolled at the university), and can be safely shared.

4.2 Additional Testing for Scalability: Multiple Universities

The ontology we described so far is suitable to describe a single university setting. As the benchmark we are proposing aims at testing system performance as the amount of stored data increases, we can imagine, for example, to test systems on different repositories containing an increasing number of students or departments. Additionally, we can imagine that different universities may adopt the same ontology for describing their data. In such settings, we can even imagine exchange students following teaching units in different universities. In this case, we can scale the size of the dataset and evaluate system performance on the private part of a single a university as well as on the public part of several universities.

5 Queries

In this section, we describe the various queries we defined in order to evaluate the performance of RDF data management systems systems in an analytic setting. The proposed set of queries aims at covering a broad range of requests, including Average, Max, Min, GroupBy, Top-K, Rank, and Path queries. In the following we describe the queries using natural language. The complete set of SPARQL queries is available in the Appendix as well as on the benchmark website.

5.1 Count Queries

This subset of queries requires systems to count the number of instances having specific characteristics. Some of them only need to access the public information, and thus can be scaled to multiple universities, while others access private information such as the result of student evaluations.

`Query 1. What is the percentage of master theses that received a mention?`
For this query the system has to select the successful master theses and to check how many of them are related to a Mention object. In general, such information is publicly available, thus systems only need to access the public part of the ontology.

`Query 2. What is the percentage of students who continue their university studies beyond the Bachelor?`
This query needs to check, for every student who obtained a Bachelor degree, whether he/she is registered to any Master program, count the number of such students, and finally divide it by the total number of students who registered for a Bachelor degree. In order to run this query the system only has to access the public part of the ontology (provided that registration information is increasingly shared across universities).

`Query 3. What is the current number of ECTS credits` *Student0* `has acquired?`
This is a more transactional query focusing on a specific instance (i.e., *Student0*), which requires to check for each evaluation taken by this student whether it was successful or not. For all such cases, the system must then sum

the number of ECTS credits assigned to the related teaching unit. For this query, systems have to access the private part of the ontology (i.e., grades of evaluations).

5.2 Selection Queries

This group aims at retrieving a large set of instances from the knowledge base. The queries may be simple in terms of operators but it requires knowledge bases to handle large amount of data.

`Query 4. Return all students whose surname starts with 'A'.`
This query returns a portion of all the student instances stored in the knowledge base. This type of queries may be useful when creating catalogs of students where, for each letter of the alphabet, there is a page containing all students whose surname starts with the same letter. As student names should not be released to the general public, in order to run this query systems needs to be able to access the private part of the dataset.

5.3 Molecule Queries

This type of queries aims at retrieving all the information surrounding a specific instance in the graph. To answer such queries, systems have to be efficient in retrieving all triples in the vicinity of a given node in the RDF graph, which represents a complex operation (with series of *joins*) for many transactional systems.

`Query 5. Return all information about` *`Student0`* `within a scope of two.`
This query returns all triples connected directly or indirectly to *Student0*, that is, a subgraph centered on *Student0* and containing all the triples directly attached to the student, plus the triples attached to those triples. As private student information are also requested, this query needs to access the private as well as the public part of the ontology.

5.4 Max and Min Queries

These queries require to find some specific instances which, among others of the same type, have the highest/lowest value for the requested characteristic.

`Query 6. Which teaching unit has the lowest success rate?`
This query requires to compute the success rate, that is, the number of successful student evaluations divided by the total number of student

evaluations done for that teaching unit, and find the minimum. It needs to access private information of the ontology.

`Query 7. Who is the professor who supervised most theses?`
This query requires, after counting the number of theses grouped by professors, to compute the maximum and return the professor. For this query, the functionality of subclass inference needs to be additionally supported by systems. This is the case as matching instances are of type Thesis_Supervisor which is a subclass of Professor. Thus, systems have to infer that such instances are of type Professor to correctly answer the query. The information required to answer this query is typically publicly available.

5.5 Ranking and TopK Queries

For such queries, the systems have to either rank all instances of a certain class or to find the top k elements of a ranked list.

`Query 8. Who are the top 5 performing students in` *`StudyTrack0`*
`and semester` *`Semester0`*`?`
This is a *top-k* query that asks the system to rank students for a specific Study Track and Semester after computing the average grade obtained in all the evaluations they have performed so far. This requires to access student grades, which is private information.

5.6 Temporal Queries

This set of queries aims at checking system support and efficiency of temporal queries, that is, requests that involve a filter of instances based on a specific date or time interval.

`Query 9. What is the average completion time of Bachelor studies`
`for each Study Track?`
This query requires the repository to compute the completion time for each student of the university, average them and group the results by study track. Information about registration and graduation date may be private or public depending on the university.

`Query 10. How did university student performance evolve over`
`the last 3 semesters?`
This query requires the system to compute the average grade of all student evaluations grouped by semester. Evaluation results required to answer this query are private information.

5.7 Path Queries

This type of query wants to test the ability of systems to exploit several nodes in the ontology joined through RDF predicates.

`Query 11. What are the names of students who took an exam with a professor of` *`Department0`*`?`
This is a path query in the sense that it requires to join triples over several different predicates (namely, Student:hasFamilyName, Student:hasFirstName, Evaluation:performedByStudent, Evaluation:evaluatedByProfessor, Professor:isAffiliatedWithDepartment, Department:hasName). As student names should not be released to the general public, in order to run this query systems needs to be able to access the private part of the dataset.

5.8 Multiple University Queries

As the goal of building an RDF benchmark is to test for scalability of systems as the size of the dataset grows, we can imagine to run certain queries over several university repositories and aggregate the results to provide a final answer to the user. In this setting, the amount of data that has to be managed can increase a lot as compared to a single university, but, on the other hand, initial queries may be run in parallel and results aggregated later on.

`Query 12. In which university can I attend` *`Teaching_Unit0`* `in English?`
In case different universities share their ontology, it is possible to run queries over different instances of the same schema. In this case, the system has to check the language of the instance *Teaching_Unit0* in all accessible universities. Information about teaching units is assumed to be public for all universities.

`Query 13. How did the number of newly registered students to each University evolve over the last 5 years?`
For this query, systems need to find the number of students enrolled at each university for each semester. This computation requires to check the registration and the possible graduation dates for each student and compare them with the semester start and end dates. We consider that information about registration and graduation date are shared across collaborating universities.

5.9 Query Classification

Table 1 presents a classification of the proposed queries based on several dimensions. First, we classify queries accordingly to the fact that they need to access the public or the private part of the ontology (see Section 4.1). Then, we use the dimensions that were introduced in [9], that is, Input Size, Selectivity, and Complexity. Input size measures the number of instances involved in the query: we classify a query as having a large input size if it involves more than 5% of instances, and small otherwise. Selectivity measures the amount of instances that match the query: we classify a query as having high selectivity if less than 10% of instances match the query, and low otherwise. Complexity measures the amount of classes and properties involved in the query: queries are classified as having high or low complexity accordingly to the RDF schema we have defined.

Table 1. Classification of queries according to their need to access private and public data, input size, selectivity, and complexity.

	Count			Selection	Molecule	MaxMin		TopK	Temp		Path	MultiUniv	
Query	1	2	3	4	5	6	7	8	9	10	11	12	13
Public	x	x					x	x	x		x	x	x
Private			x	x	x	x		x	x	x	x		x
Input Size	Small	Large	Small	Large	Small	Large	Small	Large	Large	Large	Large	Large	Large
Selectivity	High	Low	Low	High	Low	Low	Low	High	Low	Low	Low	Low	Low
Complexity	Low	Low	Low	Low	High	High	Low	High	Low	High	High	Low	High

As we can see the majority of queries have a low selectivity which reflects our intent of performing analytic queries, that is, queries for which a lot of data is retrieved and aggregated. For the same reason, most of the queries have a large input. Finally, queries are equally divided in high and low complexities.

6 Comparing RDF systems using BowlognaBench

In this section we report the results of an experiment which aims at comparing the performance of different knowledge base platforms on the analytic queries of BowlognaBench. First, we describe all the additional elements composing the proposed benchmark, namely the instance generator and the BowlognaBench website.

BowlognaBench Instance Generator. The artificial instance generator provided with BowlognaBench is a Java program that takes as input the number of departments, the number of fields per department, and the number of semesters to scale over the time dimension as well. It generates a populated Bowlogna ontology by producing RDF files (one per department) in NTriple format. In order to generate large datasets, it is recommended that at least 2G of memory are assigned to the Java process.

Artificial data may not well reflect real data distribution [6]. For this reason, we analyzed a sample of real academic data provided by the University of Fribourg administration. The observed properties are used to add additional parameters to the BowlognaBench artificial instance generator. In the following we report some observations made over the real data we had access to.

The data describes professors, departments, teaching units, and fields of study over 34 semesters at the University of Fribourg. Figure 2 shows the distribution of professors with respect to the number of teaching units related to them. As expected this follows a power law distribution where we can see that many professors teach only one teaching unit while few professors teach many units. We manually inspected special cases such as, for example, the professor with most teaching units and observed that she is an University administration employee assigned to 643 foreign language courses. Strictly speaking, this is not an error in the data but rather an imprecise link as the relation between that person and the language courses is not of the same type as for the others. Such cases are typical in real data but not so easy to find in artificially generated data.

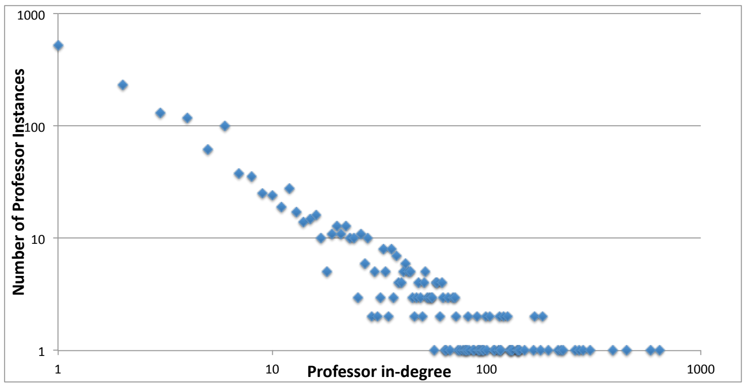


Fig. 2. In-degree of Professor instances (i.e., Teaching Units related to a Professor).

Another aspect of real data is *incompleteness*. While it is easy to generate complete artificial data, systems running over real data need to face the problem of missing values. In the available real data we observed that 0.8% of the teaching units have a 'null' value for the 'number of ECTS credits' field which is not supposed to be left empty. Another example of data incompleteness is shown in Figure 3 where the out-degree of the Teaching Unit class is displayed. We can see that 17 teaching units have out-degree of 1 which is wrong as at

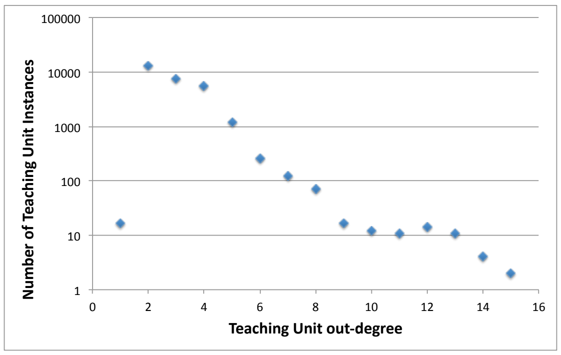


Fig. 3. Out-degree of Teaching Unit instances (i.e., relations with other entities).

least 'bb:belongsToFieldOfStudies' and 'bb:isForSemester' should be defined for a teaching unit. Otherwise, the data distribution looks as expected.

BowlognaBench Website. The benchmark we produced and used for the following experiments—that is, the Bowlogna ontology together with the set of queries and the Bowlogna artificial instance generator—is available for download at: `http://diuf.unifr.ch/xi/bowlognabench/`. The instance generator includes various options to scale and change the statistical distribution of the data. On the BowlognaBench website we also allow third-parties to publish their own test configurations and performance results.

6.1 Experimental Setting

We compared the following RDF systems: 4Store 4s-query revision v1.1.3, RDF-3X 0.3.5, Virtuoso Open-Source Edition 6.1.3, and our own dipLODocus[RDF] system [16]. In order to run such comparison we use two different datasets generated with the BowlognaBench Instance Generator for 1 and 10 departments, 4 fields per department and 15 semesters. The datasets contain 1.2 and 12 million RDF triples for 273MB and 2.7GB respectively. We run the 13 queries of BowlognaBench to compare the query execution time for four different RDF systems. Reported execution times are obtain by averaging over 10 executions of the same query over a warm-cache system. Aggregation (COUNT, SUM, AVG, etc.) and operations such as percentage or ratio have not been performed as not supported by all system: instead, all data needed to compute such aggregates is retrieved. We also compare the load time and the size of the created indexes. Experiments were run on a machine with Intel Core i7-2600 CPU @ 3.40GHz, 8GB RAM, Ubuntu 11.10.

RDF systems. In the following we provide some details about the different systems we compared over BowlognaBench:

RDF 3X[10] creates various heavily compressed indices starting from a huge triple-table. Compression is done by means of dictionary encoding and byte-wise compression. Indices are based on the six possible permutations of RDF triple elements, and aggregate indices storing only two out of the three elements. The RDF-3X query executor optimizes join orderings and determine the cheapest query plan by means of a dedicated cost-model.

4Store[11] from Garlik is a recent parallel RDF database distributing triples using a round-robin approach. It stores data in quadruple-tables.

Virtuoso[12] is an object-relational database system using bitmap indices to optimize the storage and processing of RDF data.

dipLODocus[RDF][13] is a hybrid RDF system combining subgraph storage (where joins are materialized) and column-store technologies (where values are compactly stored in vectors) to answer both triple pattern queries and analytic queries very efficiently.

6.2 Experimental Results

Table 2 shows the index size and the load time for the two datasets and four RDF systems. We can see that Virtuoso takes more time to load and index the dataset

Table 2. Load time and index size for four RDF systems and two datasets.

	RDF 3X	4Store	Virtuoso	dipLODocus
Load Time 1 dept. (s)	11.94	6.25	31.71	18.35
Load Time 10 dept. (s)	139.55	69.65	363.24	526.65
Indices size 1 dept. (MB)	60	192	108	92
Indices size 10 dept. (MB)	618	1752	616	920

but the size of the indices scales better than for other systems. The faster system is 4Store which also has the biggest indices. RDF 3X, Virtuoso, and dipLODocus achieve good compression with Virtuoso showing the best scaling.

Figure 4 and 5 report the experimental results for the datasets consisting of 1 and 10 departments respectively. The values indicate query execution times for each query of the BowlognaBench benchmark[14].

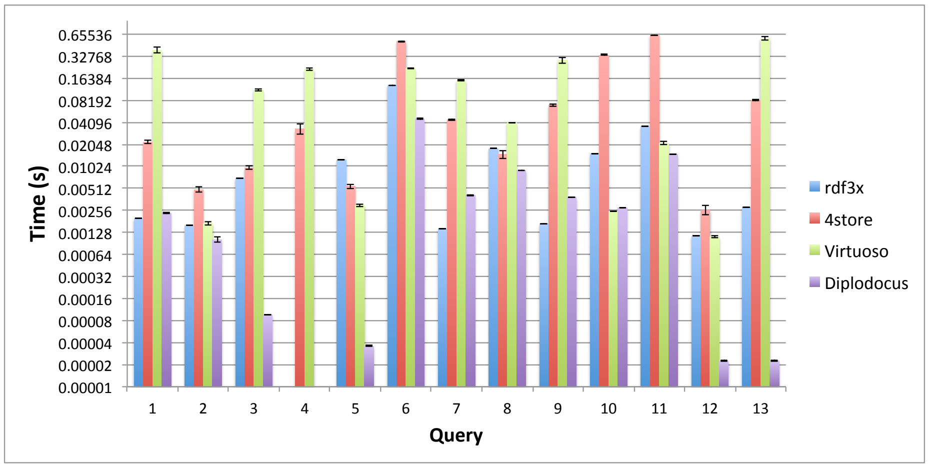


Fig. 4. Query execution time for BowlognaBench queries over the single department dataset.

As we can see, query execution time for the BowlognaBench analytic queries strongly vary for different systems (note that Figure 4 and 5 show query execution time on a logarithmic scale). For the smaller of the two datasets (i.e., data

[14] Query 4 could not be run on RDF 3X and dipLODocus as they do not provide full SPARQL support (i.e., for this query no support for pattern matching is provided).

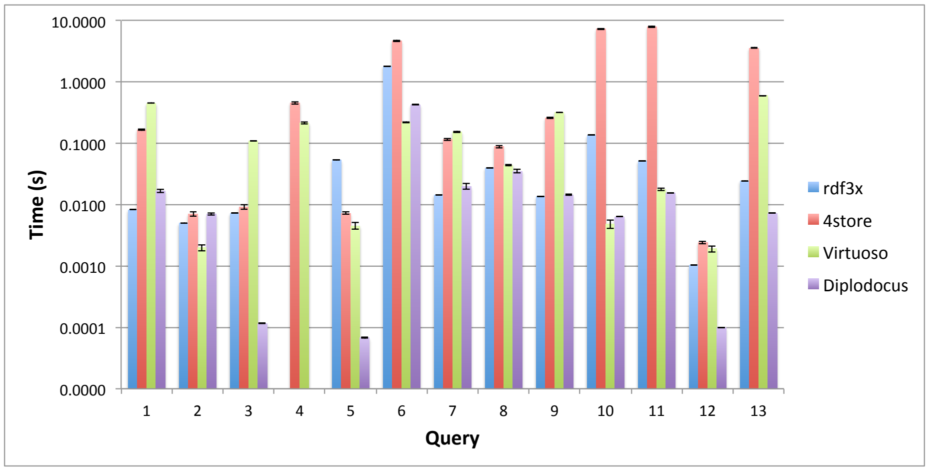


Fig. 5. Query execution time for BowlognaBench queries over the 10 departments dataset.

for a single department) the dipLODocus system is faster than others for 9 out of 13 queries. Specifically, shortest query executions can be observed for queries 12 and 13. The slowest is the path query which involves several joins. The RDF 3X system obtained the best performance on 3 queries (i.e., query 1, 7, and 9). We can see that query 8 (i.e., top K) is not easy to be efficiently answered for all systems. Query 3 and 11 are also challenging for most systems because of the several joins involved.

For the bigger dataset (i.e., data for 10 departments) we can observe the slower performances of 4Store for 7 out of 13 queries as compared with the other three systems: for some queries (e.g., 11) the execution times took up to 8 seconds. One difference that we can observe as compared with the smaller dataset is the good result of Virtuoso: it performed faster than other systems on 4 out of 13 queries. The dipLODocus system performed best on 6 queries with dramatic improvements for some queries (e.g., 5 the molecule query) showing execution times orders of magnitude smaller than other systems. In general, we can again observe that query 8 and 11 are difficult to answer efficiently because of several joins.

7 Conclusions

As RDF continues to grow in popularity, being able to efficiently execute analytic, data-intensive operations over large RDF graphs is becoming increasingly important in the business process analysis field as well as in others. In this paper, we proposed a benchmark for evaluating and comparing the performance and scalability of knowledge base systems on complex analytic queries. After describing the European academic landscape as outlined by the Bologna reform, we introduced BowlognaBench, a benchmark for RDF systems consisting of an ontology derived from a systematic study of the official Bologna documents, a set of complex queries derived from concrete user information needs, and an automatic instance generator that allows to populate the ontology with a given number of instances and relations. Finally, we compared the query execution time of different RDF systems over the proposed benchmark showing common limitations and bottlenecks of such systems as, for example, queries involving several joins. As for former benchmarks, we hope that this project will stimulate a healthy competition among RDF data management systems as well as foster the emergence of new RDF back-ends geared towards complex, data-intensive Semantic Web applications.

8 Acknowledgment

This work is supported (in part) by the Swiss National Science Foundation under grant number PP00P2_128459.

References

1. Daniel J. Abadi, Adam Marcus, Samuel R. Madden, and Kate Hollenbach. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 411–422. VLDB Endowment, 2007.
2. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.
3. C. Bizer and A. Schultz. Benchmarking the performance of storage systems that expose SPARQL endpoints. In *Proceedings of the ISWC Workshop on Scalable Semantic Web Knowledgebase systems*, 2008.
4. Gianluca Demartini, Iliya Enchev, Joël Gapany, and Philippe Cudré-Mauroux. BowlognaBench—Benchmarking RDF Analytics. In *First International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2011)*, 2011.
5. Gianluca Demartini, Iliya Enchev, Joël Gapany, and Philippe Cudré-Mauroux. The Bowlogna Ontology: Fostering Open Curricula and Agile Knowledge Bases for Europe's Higher Education Landscape. *Semantic Web - Interoperability, Usability, Applicability*, 2012.
6. Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. *SIGMOD*, 2011.
7. Joël Gapany and Guido Vergauwen. Curricular design and computerisation: are information systems useful in curricular reorganisation? *European University Association: the Bologna Handbook*, 10 (C 3.9-3), 2010.
8. Martin Grund, Jens Krüger, Hasso Plattner, Alexander Zeier, Philippe Cudre-Mauroux, and Samuel Madden. Hyrise: a main memory hybrid storage engine. *Proc. VLDB Endow.*, 4:105–116, November 2010.

9. Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158–182, 2005.
10. Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, and Gerhard Weikum. YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *WWW (Companion Volume)*, pages 229–232. ACM, 2011.
11. Enrico Minack, Wolf Siberski, and Wolfgang Nejdl. Benchmarking Fulltext Search Performance of RDF Stores. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 81–95. Springer, 2009.
12. Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data. In Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7031 of *Lecture Notes in Computer Science*, pages 454–469. Springer, 2011.
13. M. Schmidt, T. Hornung, N. Küchlin, G. Lausen, and C. Pinkel. An experimental comparison of RDF data management approaches in a SPARQL benchmark scenario. *The Semantic Web-ISWC 2008*, pages 82–97, 2008.
14. M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel. SPˆ 2Bench: A SPARQL Performance Benchmark. In *IEEE 25th International Conference on Data Engineering, ICDE'09*, pages 222–233. IEEE, 2009.
15. Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.
16. Marcin Wylot, Jigé Pont, Mariusz Wisniewski, and Philippe Cudré-Mauroux. dipLODocus[RDF] - Short and Long-Tail RDF Analytics for Massive Webs of Data. In *International Semantic Web Conference*, pages 778–793, 2011.

Appendix A: SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bb: <http://diuf.unifr.ch/main/xi/BowlognaBench/2011/05/bb#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Query 1. What is the percentage of master theses that received a mention?

```
SELECT COUNT(?X) as MasterTheses
WHERE {
?X rdf:type bb:Master_Thesis }

SELECT COUNT(?X) as MasterThesesWithMention
WHERE {
?X rdf:type bb:Master_Thesis .
?Y rdf:type bb:Mention .
?Y bb:mentionGivenTo ?X }
```

Query 2. What is the percentage of students who continue their university studies beyond the Bachelor?

```
SELECT COUNT(?X) as studentsWithBachelorDegree
WHERE{
?X rdf:type bb:Student .
?X bb:endsBachelorStudiesOn ?end^^xsd:date .
?end < "currentDate"^^xsd:date }

SELECT COUNT(?Y) as studentsEnrolledAtMaster
WHERE{
?Y rdf:type bb:Student .
?Y bb:enrolledForMasterStudiesOn ?start^^xsd:date .
?start < "currentDate"^^xsd:date }
```

Query 3. What is the current number of ECTS credits *Student0* has acquired?

```
SELECT SUM(?credits) as ECTSCredits
WHERE{
?X rdf:type bb:Student .
?Y rdf:type bb:Evaluation .
?Z rdf:type bb:Teaching_Unit .
?X bb:hasFamilyName "Student0" .
?X bb:hasFirstName "Student0" .
?Y bb:performedByStudent ?X .
?Y bb:evaluatesTeachingUnit ?Z .
?Y bb:hasMark ?M .
?M >= "C" .
?Z bb:hasNumberOfECTS ?credits }
```

Query 4. Return all students whose surname starts with ‘A’.

```
SELECT ?S
WHERE{
?S rdf:type bb:Student .
?S bb:hasFamilyName ?ln
FILTER (regex(?ln, "^A")) }
```

Query 5. Return all information about *Student0* within a scope of two.

```
SELECT ?P1, ?O1, ?P2, ?O2
WHERE{
?S rdf:type bb:Student .
?S bb:hasFamilyName "Student0" .
?S bb:hasFirstName "Student0" .
?S ?P1 ?O1 .
?O1 ?P2 ?O2 }

SELECT ?S1, ?P1, ?S2, ?P2
WHERE{
?O rdf:type bb:Student .
?O bb:hasFamilyName "Student0" .
?O bb:hasFirstName "Student0"
?S1 ?P1 ?O
?S2 ?P2 ?S1 }
```

Query 6. Which teaching unit has the lowest success rate?

```
SELECT ?X, count(?E) as FailuresForTeachingUnit
WHERE{
?X rdf:type bb:Teaching_Unit .
?E rdf:type bb:Evaluation .
?E bb:evaluatesTeachingUnit ?X .
?E bb:hasMark ?Y .
?Y < "C" }
GROUP BY ?X

SELECT ?X, count(?E) as EvaluationsForTeachingUnit
WHERE{
?X rdf:type bb:Teaching_Unit .
?E rdf:type bb:Evaluation .
?E bb:evaluatesTeachingUnit ?X }
GROUP BY ?X
```

Query 7. Who is the professor who supervised most theses?

```
SELECT ?X, COUNT(?Y) as numOfThesisSupervised
WHERE{
?X rdf:type bb:Professor .
?Y rdf:type bb:Thesis .
?Y bb:supervisedBy ?X }
ORDER BY numOfThesisSupervised
LIMIT 1
```

Query 8. Who are the top 5 performing students in *StudyTrack0* and semester *Semester0*?

```
SELECT ?X, AVG(?M) as ?avgMark
WHERE{
?X rdf:type bb:Student .
?Y rdf:type bb:Evaluation .
?T rdf:type bb:Study_Track .
?S rdf:type bb:Semester .
?T bb:hasName "StudyTrack0" .
?S bb:hasName "Semester0" .
?X bb:isInStudyTrack ?T .
?Y bb:isForSemester ?S .
?Y bb:performedByStudent ?X .
?Y hasMark ?M }
GROUP BY ?X
ORDER BY ?avgMark
LIMIT 5
```

Query 9. What is the average completion time of Bachelor studies for each Study Track?

```
SELECT ?T, AVG(?end-?start) as CompletionTime
WHERE{
?T rdf:type bb:Study_Track .
?X rdf:type bb:Student .
?X bb:enrolledForBachelorStudiesOn ?start^^xsd:date .
?X bb:endsBachelorStudiesOn ?end^^xsd:date .
?X bb:isInStudyTrack ?T }
GROUP BY ?T
```

Query 10. How did university student performance evolve over the last 3 semesters? *Repeat for all semesters*

```
SELECT AVG(?mark) as AverageSemester1
WHERE{
?X rdf:type bb:Evaluation .
?Y rdf:type bb:Teaching_Unit .
?Z rdf:type bb:Semester .
?X bb:evaluatesTeachingUnit ?Y .
?X bb:isForSemester ?Z .
?Z bb:hasName "Semester1" .
?X bb:hasMark ?mark }
```

Query 11. What are the names of students who took an exam with a professor of *Department0*?

```
SELECT ?FamName, ?FirstName
WHERE{
?X rdf:type bb:Student .
?Y rdf:type bb:Professor .
?Z rdf:type bb:Evaluation .
?D rdf:type bb:Department .
?Z bb:performedByStudent ?X .
?Z bb:evaluatedByProfessor ?Y .
?Y bb:isAffiliatedWithDepartment ?D .
?X bb:hasFamilyName ?FamName .
?X bb:hasFirstName ?FirstName .
?D bb:hasName "Department0" }
```

Query 12. In which university can I attend *Teaching_Unit0* in English?
Repeat for all universities.

```
SELECT ?X
WHERE{
?X rdf:type bb:Teaching_Unit .
?X bb:isTaughtInLanguage "EN" .
?X bb:hasName "TeachingUnit0" }
```

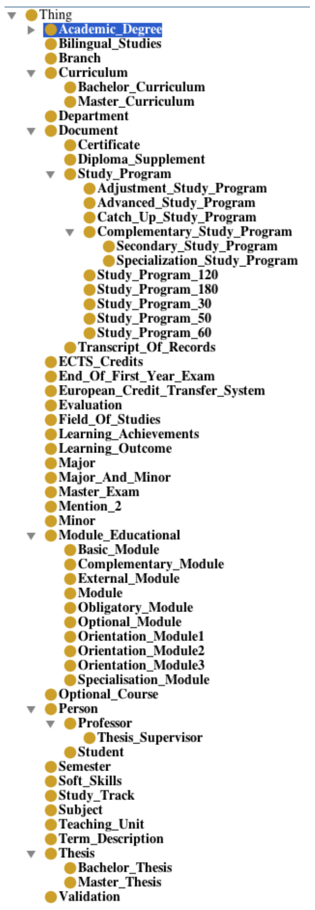
Query 13. How did the number of newly registered students to each University evolve over the last 5 years?
Repeat for all semesters and all universities.

```
SELECT COUNT(?X) as BachelorStudentsSemester0
WHERE{
?X rdf:type bb:Student .
?S rdf:type bb:Semester .
?S bb:hasName "Semester0" .
?S bb:beginsOnDate ?start^^xsd:date .
?S bb:endsOnDate ?end^^xsd:date .
?X bb:enrolledForBachelorStudiesOn ?Y^^xsd:date .
?Y <= ?end .
?Y >= ?start }

SELECT COUNT(?X) as MasterStudentsSemester0
WHERE{
?X rdf:type bb:Student .
?S rdf:type bb:Semester .
?S bb:hasName "Semester0" .
?S bb:beginsOnDate ?start^^xsd:date .
?S bb:endsOnDate ?end^^xsd:date .
?X bb:enrolledForMasterStudiesOn ?Y^^xsd:date .
?Y <= ?end .
?Y >= ?start }
```

Appendix B: The Bowlogna Ontology Classes

Figure 6 presents all the 67 classes defined in the Bowlogna ontology.



- Thing
  - Academic_Degree
  - Bilingual_Studies
  - Branch
  - Curriculum
    - Bachelor_Curriculum
    - Master_Curriculum
  - Department
  - Document
    - Certificate
    - Diploma_Supplement
    - Study_Program
      - Adjustment_Study_Program
      - Advanced_Study_Program
      - Catch_Up_Study_Program
      - Complementary_Study_Program
        - Secondary_Study_Program
        - Specialization_Study_Program
      - Study_Program_120
      - Study_Program_180
      - Study_Program_30
      - Study_Program_50
      - Study_Program_60
    - Transcript_Of_Records
  - ECTS_Credits
  - End_Of_First_Year_Exam
  - European_Credit_Transfer_System
  - Evaluation
  - Field_Of_Studies
  - Learning_Achievements
  - Learning_Outcome
  - Major
  - Major_And_Minor
  - Master_Exam
  - Mention_2
  - Minor
  - Module_Educational
    - Basic_Module
    - Complementary_Module
    - External_Module
    - Module
    - Obligatory_Module
    - Optional_Module
    - Orientation_Module1
    - Orientation_Module2
    - Orientation_Module3
    - Specialisation_Module
  - Optional_Course
  - Person
    - Professor
      - Thesis_Supervisor
    - Student
  - Semester
  - Soft_Skills
  - Study_Track
  - Subject
  - Teaching_Unit
  - Term_Description
  - Thesis
    - Bachelor_Thesis
    - Master_Thesis
  - Validation

Fig. 6. The class hierarchy of the Bowlogna ontology