A Scalable Benchmark for OBDA Systems: Preliminary Report*

Diego Calvanese, Davide Lanti, Martin Rezk, Mindaugas Slusnys, and Guohui Xiao

Faculty of Computer Science, Free University of Bozen-Bolzano, Italy

Abstract. In ontology-based data access (OBDA), the aim is to provide a high-level conceptual view over potentially very large (relational) data sources by means of a mediating ontology. The ontology is connected to the data sources through a declarative specification given in terms of mappings that relate each (class and property) symbol in the ontology to an (SQL) view over the data. Although prototype OBDA systems providing the ability to answer SPARQL queries over the ontology are available, a significant challenge remains: performance. To properly evaluate OBDA systems, benchmarks tailored towards the requirements in this setting are needed. OWL benchmarks, which have been developed to test the performance of generic SPARQL query engines, however, fail at 1) exhibiting a complex real-world ontology, 2) providing challenging real world queries, 3) providing large amounts of real-world data, and the possibility to test a system over data of increasing size, and 4) capturing important OBDA-specific measures related to the rewriting-based query answering approach in OBDA. In this work, we propose a novel benchmark for OBDA systems based on a real world use-case adopted in the EU project Optique. We validate our benchmark on the system Ontop, showing that it is more adequate than previous benchmarks not tailored for OBDA.

1 Introduction

In ontology-based data access (OBDA), the aim is to provide a high-level conceptual view over potentially very large (usually relational) data sources by means of a mediating ontology. Queries are posed over such conceptual layer and then translated into queries over the data layer. The ontology is connected to the data sources through a declarative specification given in terms of mappings that relate each (class and property) symbol in the ontology to an (SQL) view over the data. The W3C standard R2RML [13], was created with the goal of providing a standardized language for the specification of mappings in the OBDA setting.

To properly evaluate the performance of OBDA systems, benchmarks tailored towards the requirements in this setting are needed. OWL benchmarks, which have been developed to test the performance of generic SPARQL query engines, however, fail at 1) exhibiting a complex real-world ontology, 2) providing challenging real world queries, 3) providing large amounts of real-world data, and the possibility to test a system over data of increasing size, and 4) capturing important OBDA-specific measures related to the rewriting-based query answering approach in OBDA. For instance, in the Berlin SPARQL

* This paper is supported by the EU under the large-scale integrating project (IP) Optique (Scalable End-user Access to Big Data), grant agreement n. FP7-318338.

Benchmark [7], although it is possible to configure the data size (in triples), there is no ontology to measure reasoning tasks and the queries are rather simple. Therefore it is hard to evaluate some of the key features of OBDA system, such as rewritings with respect to an ontology and/or a set of mappings. Moreover, the data is fully artificial, hence it is difficult to assess the significance of the obtained performance results with respect to real world settings. Another popular benchmark is FishMark [5]. In this benchmark the queries are more challenging, and "real world " data is used. However, the benchmark does not come with an ontology and the data size is rather small (≈20M triples). A popular benchmark that does come with an ontology and with the possibility of generating data (triples) of arbitrary size is LUBM [15]. However, the ontology is rather small, and the benchmark is not tailored towards OBDA, since no mappings to data sources are provided. Moreover, the queries in combination with the ontology appear to provide unnatural results, e.g., they either return no results or a very large portion of the data.

In this work, we propose a novel benchmark for OBDA systems based on the Norwegian Petroleum Directorate (NPD), which is a real world use-case adopted in the EU project Optique [4].Specifically, we adopt the NPD Fact Pages as dataset, the NPD Ontology, which has been mapped to the NPD Fact Pages stored in a relational database, and queries over such an ontology developed by domain experts. The main challenge we address here has been to develop a data generator for generating datasets of increasing size, starting from the available data. This problem has been studied before in the context of databases [6], where increasing the data size is achieved by encoding domain-specific information into the data generator [12, 8, 9]. One drawback of this approach is that each benchmark requires its ad-hoc generator, and also that it disregards OBDA specific aspects. In the context of triple stores, [24, 16] present an interesting approach based on machine learning. Unfortunately, the approach proposed in these papers is specifically tailored for triple stores, and thus it is not directly applicable to the OBDA settings. Applying these approaches to OBDA, in fact, is far from trivial and closely related to the *view update problem* [11].

We present the NPD benchmark[1] in Section 2, discuss our data generator in Section 3, validate our benchmark in Section 4, and conclude in Section 5.

2 NPD Benchmark

The Norwegian Petroleum Directorate [1] (NPD) is a governmental organisation whose main objective is to contribute to maximize the value that society can obtain from the oil and gas activities. The initial dataset that we use are the *NPD Fact Pages* [2], which contains information regarding the petroleum activities on the Norwegian continental shelf. The ontology, the query set, and the mappings to the dataset have all been developed at the University of Oslo [22], and are freely available online [3]. Next we provide more details on each of these items.

The Ontology. The ontology contains OWL axioms specifying comprehensive information about the underlying concepts in the dataset. Since we are interested in benchmarking OBDA systems that are able to rewrite SPARQL queries over the ontology into FOL queries (therefore, SQL) queries that can be evaluated by a relational DBMS, we concentrate here on the OWL 2 QL profile [20] of OWL, which guarantees FO-rewritability of

[1] `https://github.com/ontop/npd-benchmark`

Table 1. Ontology Statistics

#classes	#obj_prop	#data_prop	#intensional axioms	max_depth	avg_siblings
343	142	238	1451	10	4.83

unions of conjunctive queries (see, e.g., [10]). Table 1 shows the statistics for the maximal OWL 2 QL subset of the NPD ontology. This ontology is suitable for benchmarking reasoning tasks, given that *(i)* it is a complex ontology in terms of number of classes, maximum depth of the class hierarchy, and average number of sibling classes for each class [18], making it suitable for reasoning w.r.t. hierarchies; and *(ii)* it contains existentials in the right-hand side of ontology axioms. These axioms infer unnamed individuals in the virtual instance that cannot be retrieved as part of the answer, but can affect the evaluation of the query as they strongly affect the size of the rewritten query.

The Query Set. The original NPD query set contains 25 queries obtained by interviewing users of the NPD dataset. Starting from the original NPD query set, we devised 12 queries having different degrees of complexity (see Table 2). In the OBDA context, it is recognized that queries involving classes (or object/data properties) with a rich hierarchy, or making use of existentially quantified variables in a rather sophisticated way (i.e., giving rise to *tree witnesses* [17, 21]) are harder to answer than other queries, because they produce more complex rewritings. We also fixed some minor issues, e.g., the absence in the ontology of certain concepts present in the queries, removing aggregates not supported by *Ontop*, and flattening of nested sub-queries.

The Mappings. The mapping consists of 1190 assertions mapping a total of 464 among classes, objects properties, and data properties. The SQL queries in the mappings, in terms of their direct DATALOG translation, count an average of 2.6 rules, with 1.7 joins per rule. We observe that the mappings have not been optimised to take full advantage of an OBDA framework, e.g., by trying to minimize the number of mappings that refer to the same ontology class or property, so as to reduce the size of the SQL query generated by unfolding the mapping.

3 The Data Generator

The generator produces data according to certain features of an input training database. The algorithm is not specific for NPD, and it can in principle be applied to every database.

Table 2. Statistics for the queries considered in the benchmark

query	#tot_op	#join	#depth	#tw	max(#subclasses)
Q1	12	4	8	0	0
Q2	14	5	9	0	0
Q3	10	3	7	0	0
Q4	14	5	9	0	0
Q5	14	5	8	0	0
Q6	18	6	12	2	23
Q7	17	7	10	0	0
Q8	10	3	7	0	0
Q9	10	3	7	0	38
Q10	9	2	7	0	0
Q11	20	7	12	2	23
Q12	26	8	12	4	23

We put special care in making the generation process fast, so as to be able to generate very large datasets. The algorithm starts from a non-empty database D. Given a *desired increment* $f > 0$, it generates a new database such that $|T'| = |T| \cdot (1 + f)$, for each table T of D that has to be incremented (where $|T|$ denotes the number of tuples of T).

Duplicate Values Generation. Values in each column $T.C$ of a table $T \in D$ are generated with a *duplicate ratio* $(\|T.C\| - |T.C|)/\|T.C\|$, where $\|T.C\|$ (resp., $|T.C|$) denotes the cardinality of the column $T.C$ under the multi-set (resp., set) semantics. A duplicate ratio "close to 1" indicates that the content of the column is essentially *independent* from the size of the database, and it should not be increased by the data generator.

Fresh Values Generation. For each non-string column $T.C$ over a totally ordered domain, the generator chooses values from the interval $\mathcal{I} := [\min(T.C), \max(T.C)]$. If the number of values to insert exceeds the number of different fresh values that can be chosen from the interval $\mathcal{I}$, then values greater than $\max(T.C)$ are allowed.

Chase Cycles. The data generation is done respecting foreign keys. Let $T_1 \to T_2$ denote the presence of a foreign key from table T_1 to table T_2. In case of a cycle $T_1 \to T_2 \to \cdots \to T_k \to T_1$, inserting a tuple in T_1 could potentially trigger an infinite number of insertions. By analyzing the input database, the generator prevents an infinite chain of insertions while ensuring that no foreign key constraint is violated.

Geometric Types. The NPD database makes use of geometric datatypes available in MySQL. Some of them come with constraints, e.g., a polygon is a closed non-intersecting line composed of a finite number of straight lines. For each geometric column in the database, the generator first identifies the minimal rectangular region of space enclosing all the values in the column, and then it generates values in that region.

Generator Validation. We aim at producing synthetic data that approximates the real-world data well. By considering the mapping assertions, it is possible to establish the *expected growth* of the ontology elements w.r.t. the growth of the database. For our analysis we focus here on those ontology elements for which the expected growth has to be either linear (w.r.t. the growth of the database) or absent. We tested this way 138 concept names, 28 object properties and 226 data properties.

Table 3 reports the results of the aforementioned analysis. The first column indicates the type of ontology elements being analyzed, and the specified increment f (e.g., "class_npd2" refers to the population of classes for the database incremented with factor $f = 2$). The columns "avg err" show the averages of the errors between the expected growth and the observed growth, in terms of percentage of the expected growth. The remaining columns report the number and percentage of ontology elements for which the error was greater than 50%.

Concerning concepts, our generator seems to behave well. This is because concepts are usually populated starting from primary keys in the database, or by simple queries that do not involve the Cartesian product of two or more tables (in this last case the growth would be quadratic). More attention, instead, has to be paid when populating object or data properties. Our data generator, indeed, considers the distribution of the elements in a column-wise manner, and ignores the distribution of the actual relations in the ontology. Observe that, in order to solve this problem, analyzing the distribution at the level of tuples in relations might in general be insufficient, since object/data properties

Table 3. Comparison between heuristic and random data generator

	avg err		err $\geq$50% (absolute)		err $\geq$50% (relative)	
type_db	heuristic	random	heuristic	random	heuristic	random
class_npd2	3.24%	370.08%	2	67	1.45%	48.55%
class_npd10	6.19%	505.02%	3	67	2.17%	48.55%
obj_npd2	87.48%	648.22%	8	12	28.57%	42.86%
obj_npd10	90.19%	883.92%	8	12	28.57%	42.86%
data_npd2	39.38%	96.30%	20	46	8.85%	20.35%
data_npd10	53.49%	131.17%	28	50	12.39%	22.12%

can be populated with columns from different tables. On the other hand, a more accurate approximation working at the level of the triples in the ontology [24] might be unfeasible, because of the well-known view update problem [11] and of the quantity of triples that need to be generated in order to create big datasets. However, the problem is not the same as the view update, since we are not interested in reflecting a particular "virtual instance" into a "physical (relational) source", but instead we aim at creating a physical source that produces a virtual instance satisfying certain statistics on some key features. Further investigation is left to future work.

Finally, the performed tests and the data reported in Table 3 indicate that our heuristic generator substantially outperforms a pure random generator.

4 Benchmark Results

We ran the benchmark on the *Ontop* system[2] [21], which, to the best of our knowledge, is the only one fully implemented OBDA system that is freely available. MySQL was used as underlying relational database system. The hardware consisted of an HP Proliant server with 24 Intel Xeon X5690 CPUs (144 cores @3.47GHz), 106GB of RAM and a 1TB 15K RPM HD. The OS is Ubuntu 12.04 LTS. Due to space constraints, we present the results for only one running client.

Results were obtained with a configuration that does not consider anonymous individuals. This is motivated by the fact that none of the queries for which this kind of reasoning would give additional answers (i.e., queries with tree witnesses) could be rewritten, and executed, within the given timeout (1 min.).

[2] `http://ontop.inf.unibz.it/`

Table 4. Tractable queries

db	avg(ex_time)	avg(out_time)	avg(res_size)	qmpH	#(triples)
NPD	44	102	15960	2167.37	≈2M
NPD2	70	182	30701	1528.01	≈6M
NPD10	148	463	81770	803.86	≈25M
NPD50	338	1001	186047	346.87	≈116M
NPD100	547	1361	249902	217.36	≈220M
NPD500	2415	5746	943676	57.80	≈1.3B
NPD1500	6740	18582	2575679	17.66	≈4B

Table 5. Hard queries

query	#unfolding	#tw	NPD	NPD2	NPD5	NPD10	NPD10 RAND
q6	48	2	513	5595	19558	—	6948
q9	570	0	1180	1949	3418	49411	1519
q10	24	0	52	125	211	243	393
q11	24	2	385	9422	55436	—	3255
q12	48	4	632	46435	—	—	5180

In order to test the scalability of the system w.r.t. the growth of the database, we used the data generator described in Section 3 and produced several databases, the largest being approximately 1500 times bigger than the original one ("NPD1500" in Table 4, $\approx$ 117 GB of size on disk).

Table 4 refers to a mix of the 7 easiest queries from the initial query set, for which the unfolding of the mapping produces exactly one non-union SQL query. The mix is executed 10 times, each time with different parameters at the filter conditions, so that the effect of caching is minimized, and statistics were collected for each execution. These are the only queries which display a *response time*—that is, the sum of the *query execution time* (avg(ex_time)) and the time spent by the system to display the results to the user (avg(out_time))—that is less than one minute for every database in the benchmark. Results are in milliseconds, and columns qmpH and avg(res_size) refer to *query mixes per hour* and the *average number of results* for queries in the mix, respectively.

Table 5 contains results for the 5 hardest queries. Each query was run once with a timeout of 1 minute on the response time. Observe that the response time tends to grow faster than the growth of the underlying database. This follows from the complexity of the queries produced by the unfolding step (column #unfolding indicates the number of union operators in the produced sql query), which usually contain several joins (remember that the worst case cardinality of a result set produced by a join is quadratic in the size of the original tables). Column NPD10 RAND witnesses how using a purely random data generator gives rise to datasets for which the queries are much simpler to evaluate. This is mainly due to the fact that a random generation of values tends to decrease the ratio of duplicates inside columns, resulting in smaller join results over the tables [23]. Hence, purely randomly generated datasets are not appropriate for benchmarking.

5 Conclusions and Future Work

The benchmark proposed in this work is the first one that thoroughly analyzes a complete OBDA system in all significant components. So far, little or no work has been done in this direction, as pointed out in [19], since the research community has mostly focused on *rewriting* engines. Thanks to our work, we have gained a better understanding of the current state of the art for OBDA systems: first, we confirm that the unfolding phase is the real bottleneck of modern OBDA systems [14]; second, more research work is needed in order to understand how to improve the design of mappings, avoiding the use of mappings that give rise to huge queries after unfolding.

We conclude by observing that for a better analysis it is crucial to refine the generator in such a way that domain-specific information is taken into account, and a better approximation of real-world data is produced.

References

1. Norwegian Petroleum Directorate, http://www.npd.no/en/, accessed: 2014-05-20
2. NPD FactPages, http://factpages.npd.no/factpages/, accessed: 2014-05-20
3. NPD's FactPages as semantic web data, http://sws.ifi.uio.no/project/npd-v2/, accessed: 2014-05-01
4. Optique: Scalable end-user access to Big Data, http://www.optique-project.eu/, accessed: 2014-05-01
5. Bail, S., Alkiviadous, S., Parsia, B., Workman, D., van Harmelen, M., Goncalves, R.S., Garilao, C.: FishMark: A linked data application benchmark. In: Proc. of the Joint Workshop on Scalable and High-Performance Semantic Web Systems (SSWS+HPCSW 2012), vol. 943, pp. 1–15. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/ (2012)
6. Bitton, D., Dewitt, D.J., Turbyfill, C.: Benchmarking database systems: A systematic approach. In: Proc. of the 9th Int. Conf. on Very Large Data Bases (VLDB 1983). pp. 8–19 (1983)
7. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. Int. J. on Semantic Web and Information Systems 5(2), 1–24 (2009)
8. Bruno, N., Chaudhuri, S.: Flexible database generators. In: Proc. of the 35th Int. Conf. on Very Large Data Bases (VLDB 2009). pp. 1097–1107 (2005)
9. Bsche, K., Sellam, T., Pirk, H., Beier, R., Mieth, P., Manegold, S.: Scalable generation of synthetic GPS traces with real-life data characteristics. In: Selected Topics in Performance Evaluation and Benchmarking, Lecture Notes in Computer Science, vol. 7755, pp. 140–155. Springer (2013)
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodríguez-Muro, M., Rosati, R.: Ontologies and databases: The *DL-Lite* approach. In: Tessaris, S., Franconi, E. (eds.) Reasoning Web. Semantic Technologies for Informations Systems – 5th Int. Summer School Tutorial Lectures (RW 2009), Lecture Notes in Computer Science, vol. 5689, pp. 255–356. Springer (2009)
11. Cosmadakis, S.S., Papadimitriou, C.H.: Updates of relational views. J. of the ACM 31(4), 742–760 (1984)
12. Crolotte, A., Ghazal, A.: Introducing skew into the TPC-H Benchmark. In: Topics in Performance Evaluation, Measurement and Characterization, Lecture Notes in Computer Science, vol. 7144, pp. 137–145. Springer (2012)
13. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C Recommendation, World Wide Web Consortium (Sep 2012), available at http://www.w3.org/TR/r2rml/
14. Di Pinto, F., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: Proc. of the 16th Int. Conf. on Extending Database Technology (EDBT 2013). pp. 561–572 (2013)
15. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. of Web Semantics 3(2–3), 158–182 (2005)
16. Guo, Y., Qasem, A., Pan, Z., Heflin, J.: A requirements driven framework for benchmarking semantic web knowledge base systems. IEEE Trans. on Knowledge and Data Engineering 19(2), 297–309 (Feb 2007)
17. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to query answering in *DL-Lite*. In: Proc. of the 12th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2010). pp. 247–257 (2010)
18. LePendu, P., Noy, N.F., Jonquet, C., Alexander, P.R., Shah, N.H., Musen, M.A.: Optimize first, buy later: Analyzing metrics to ramp-up very large knowledge bases. In: Proc. of the 9th Int. Semantic Web Conf. (ISWC 2010). Lecture Notes in Computer Science, vol. 6496, pp. 486–501. Springer (2010)

19. Mora, J., Corcho, O.: Towards a systematic benchmarking of ontology-based query rewriting systems. In: Proc. of the 12th Int. Semantic Web Conf. (ISWC 2013). Lecture Notes in Computer Science, vol. 8218, pp. 369–384. Springer (2013)
20. Motik, B., Fokoue, A., Horrocks, I., Wu, Z., Lutz, C., Cuenca Grau, B.: OWL 2 web ontology language profiles. W3C Recommendation, World Wide Web Consortium (Oct 2009), available at `http://www.w3.org/TR/owl-profiles/`
21. Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: Ontop of databases. In: Proc. of the 12th Int. Semantic Web Conf. (ISWC 2013). Lecture Notes in Computer Science, vol. 8218, pp. 558–573. Springer (2013)
22. Skjæveland, M.G., Lian, E.H.: Benefits of publishing the Norwegian Petroleum Directorate's FactPages as Linked Open Data. In: Proc. of Norsk informatikkonferanse (NIK 2013). Tapir (2013)
23. Swami, A., Schiefer, K.: On the estimation of join result sizes. In: Proc. of the 4th Int. Conf. on Extending Database Technology (EDBT 1994). Lecture Notes in Computer Science, vol. 779, pp. 287–300. Springer (1994)
24. Wang, S.Y., Guo, Y., Qasem, A., Heflin, J.: Rapid benchmarking for semantic web knowledge base systems. In: Proc. of the 4th Int. Semantic Web Conf. (ISWC 2005). Lecture Notes in Computer Science, vol. 3729, pp. 758–772. Springer (2005)