An Evaluation of VIG with the BSBM Benchmark

Davide Lanti, Guohui Xiao, and Diego Calvanese

Free University of Bozen-Bolzano, Italy

Abstract. We present an experimental evaluation of VIG, a data scaler for OBDA benchmarks. Data scaling is a relatively recent approach, proposed in the database community, that allows for scaling an input data instance to s times its size, while preserving certain application-specific characteristics. A data scaler is a "general" generator, in the sense that it can be re-used on different database schemas, and that users are not required to manually input the data characteristics. VIG lifts the scaling approach from the database level to the OBDA level, where the domain information of ontologies and mappings has to be taken into account as well. To evaluate VIG, in this paper we use it to generate data for the Berlin SPARQL Benchmark (BSBM), and compare it with the official BSBM data generator.

1 Introduction

An important research problem in Big Data is how to provide end-users with transparent access to the data, abstracting from storage details. In Ontology-based Data Access (OBDA) [6], a solution is realized by presenting the data stored in a relational database to the end-users as a *virtual* RDF graph over which SPARQL queries can be posed. This solution is realized through *mappings* that link classes and properties in the ontology to queries over the database.

Benchmarking of OBDA systems requires scalability analyses taking into account data instances of increasing volumes. Such instances are often provided by generators of synthetic data which are either complex schema-specific implementations or require considerable manual input by the end-user. Applying either of the two approaches might be challenging in a setting like OBDA where database schemas tend to be particularly big and complex (e.g., 70 tables, some with more of 80 columns in [3]).

Data scaling [7] is a recent approach that tries to overcome this problem by automatically tuning the generation parameters through statistics collected over an initial data instance. Hence, the same generator can be reused in different contexts, as long as an initial data instance is available. A measure of quality for the produced data is defined in terms of results for the available queries, that should be *similar* to the one observed for real data of comparable volume. In the context of OBDA, taking as the only parameter for generation an initial data instance does not produce data of acceptable quality, since it has to comply with constraints deriving from the structure of the mappings and the ontology, that in turn derive from the application domain.

VIG is a data scaler for OBDA benchmarks designed to address these limitations. In the VIG system, the scaling approach is lifted from the instance level to the OBDA level by analyzing the structure of the mappings component. VIG is efficient and suitable to generate huge amounts of data, as tuples are generated in constant time without need to retrieve previously generated values. Furthermore, the generation can be parallelized up

to the number of columns in the schema, without communication overhead. The system is maintained by the Ontop team [2], is released [5] under an Apache 2.0 license, and comes with documentation in the form of Wiki pages.

The rest of the paper is structured as follows: in Section 2, we describe the similarity measures according to which VIG scales data. In Section 3, we evaluate VIG over the BSBM benchmark, and compare it against the official BSBM instances generator. Section 4 concludes the paper.

2 Data Scaling for OBDA Benchmarks: VIG Approach

The *data scaling problem* [7] is the problem of producing, starting from an initial dataset $\mathcal{D}$ over a schema Σ, a dataset $\mathcal{D}'$ which is also over Σ and *similar* to $\mathcal{D}$ but s times its size. Concerning the size, similarly to other approaches, VIG scales each table in $\mathcal{D}$ by a factor of s. The notion of *similarity*, instead, is application-based. Being our goal benchmarking, we define the similarity in terms of execution time for the queries in the benchmark. In VIG we slightly change the scaling problem to make it better suited for the OBDA context. In addition to $\mathcal{D}$, in fact, VIG takes as input also the mappings and uses them to estimate the workload for the OBDA system. This allows for a more realistic and OBDA-tuned generation. In order to generate "similar" instances, VIG adopts the following three similarity measures. More details can be found in [4].

Primary and Foreign Key Constraints. VIG is, to the best of our knowledge, the only data scaler able to generate in constant time tuples that satisfy multi-attribute primary keys for *weakly-identified entities* (i.e., entities that cannot be uniquely identified by their attributes alone.). The current implementation of VIG does not support multi-attribute foreign keys.

Column-based Duplicates and NULL Ratios. They respectively measure the ratio of duplicates and of nulls in a given column, and we consider them to be meaningful similarity measures as they are common parameters for the cost estimation performed by query planners in databases. These measures are maintained in $\mathcal{D}'$ for all *non-fixed* domain columns (i.e., columns in which the number of distinct values depends on s). *Fixed-domain* columns will only contain values observed in $\mathcal{D}$. Common cases of fixed-domain columns can be automatically detected by VIG through mappings analysis, and additional ones can be manually specified.

Cost of Joins between Semantically Related Columns. Through mapping analysis VIG identifies those columns for which a join operation is semantically meaningful (i.e., columns for which a join could occur during the evaluation of a user query). Generating data that guarantees the correct selectivity for these joins is crucial in order to deliver a realistic evaluation. In the NPD Benchmark, for instance, the SQL join between the pair of key columns realizing individuals for classes `ShallowWellbore` and `Exploration` returns an empty result on $\mathcal{D}$. In fact, these two classes are declared to be disjoint in the ontology. VIG can generate data that preserves the selectivity for this join, and therefore that satisfies the disjointness constraint specified over the ontology.

3 Evaluation with The BSBM Benchmark

The well-known Berlin SPARQL Benchmark (BSBM) [1] comes with a set of parametric queries, an ontology, mappings, an automatized test platform, and an ad-hoc data generator (GEN) that can generate data according a scale parameter given in terms of

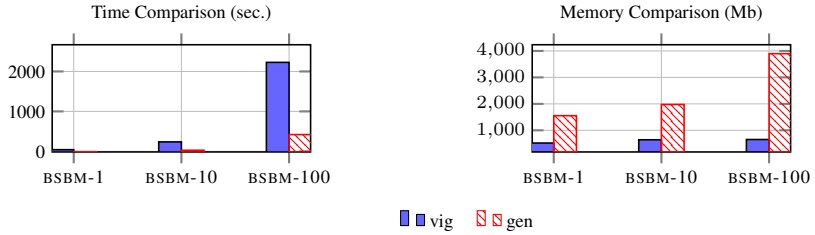

Fig. 1: Generation Time and Memory Comparison

number of products. The queries contain placeholders that the testing platform instantiates with values chosen from the generated ones.

We used the two generators to create six data instances, denoted as BSBM-s-g, where $s = 1, 10, 100$ indicates the scale factor with respect to an initial data instance of 10000 products (produced by GEN), and $g \in \{\text{VIG,GEN}\}$ indicates the generator used to produce the instance. The details of the experiment as well as the material needed for reproducing it can be found online [5]. The experiments have been ran on a HP Proliant server with 2 Intel Xeon X5690 CPUs (24 cores @3.47GHz), 106 GB of RAM and a 1 TB 15K RPM HD. The OS is Ubuntu 12.04 LTS.

Resources Consumption Comparison. Figure 1 shows the resources (time and memory) used by the two generators for creating the instances. For both generators the execution time grows approximately as the scale factor, which suggests that the generation of a single column value is in both cases independent from the size of the data instance to be generated. Observe that GEN is on average 5 times faster than VIG, but it also requires increasingly more memory as the amount of the data to generate increase, contrary to VIG that always requires the same amount of memory. We point out that VIG supports parallel generation up to the number of columns in the schema for $\mathcal{D}$, so we expect the execution time to be substantially lower on a multi-node machine.

Benchmark Queries Comparison. The top portion of Table 1 compares the execution times for the queries in the BSBM benchmark evaluated over the instances produced by VIG/GEN.Queries were ran on the *Ontop* OBDA system [2], over a MySQL back-end.

The testing platform of the BSBM benchmark instantiates the queries with concrete values coming from configuration files produced by GEN. This does not allow a fair comparison between the two generators, because it is biased towards the specific values produced by GEN. To run a fair comparison, we use the testing platform of the NPD benchmark, which is independent from the specific generator used and instantiates the queries only with values found in the provided database instance.

Finally, we slightly cleaned the mappings by removing redundancies and modified the queries by removing the `LIMIT` modifiers or relaxing excessively restricting filter conditions. Since these modifications do not affect the size of the produced SQL translation, the tested queries are at least as hard as the original ones.

Deviation for Predicates Growth. The bottom part of Table 1 shows the deviation, in terms of number of elements for each predicate (class, object or data property) in the ontology, between the instances generated by VIG and those generated by GEN. The first column reports the average deviation, and last two columns report the absolute number and relative percentage of predicates for which the deviation was greater than 5%. The

Table 1: Query Evaluation Results. BSBM Benchmark.

Comparison over BSBM Benchmark Queries (10 runs, 1 warm-up run)					
db	avg(ex_time) msec.	avg(out_time) msec.	avg(res_size) msec.	qmpH	avg(mix_time) [σ^2] msec.
BSBM-1-GEN	87	6	1425	4285	840 [101.747]
BSBM-1-VIG	77	3	841	4972	724 [85.387]
BSBM-10-GEN	628	29	11175	608	5916 [444.489]
BSBM-10-VIG	681	29	13429	563	6388 [606.057]
BSBM-100-GEN	6020	271	122169	63	56620 [5946.65]
BSBM-100-VIG	6022	212	83875	64	56117 [4508.37]

Predicates Growth Results			
type_db-scale	avg(dev)	dev $> 5\%$ (absolute)	dev $> 5\%$ (relative)
CLASS-BSBM-1	0%	0	0%
CLASS-BSBM-10	23.72%	2	25%
CLASS-BSBM-100	250.74%	2	25%
OBJ-BSBM-1	0%	0	0%
OBJ-BSBM-10	7.46%	2	20%
OBJ-BSBM-100	82.35%	2	20%
DATA-BSBM-1	$< 0.01\%$	0	0%
DATA-BSBM-10	2.84%	2	6.67%
DATA-BSBM-100	5.74%	2	6.67%

high deviation in the CLASS and OBJ rows in instances of scale factor 10 and 100 is due to a small number of outliers whose elements are built from tables that GEN, contrary to VIG, does not scale according to the scale factor.

4 Conclusion and Development Plan

In this work we evaluated VIG in the task of generating data for the BSBM benchmark. More precisely, we measured how *similar* is the data produced by VIG to the one produced by the native BSBM generator, obtaining encouraging results.

The current work plan is to enrich the quality of the data produced by adding more similarity measures to the generation process, such as multi-attribute foreign keys, non-uniform distributions, or joint-degree distributions [7]. Unfortunately, we expect that some of these measures will conflict with the feature of constant time for generation of tuples (e.g., joint-degree distributions require access to previously generated tuples).

Acknowledgment This paper is supported by the EU project Optique FP7-318338.

References

1. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. Int. J. on Semantic Web and Information Systems 5(2), 1–24 (2009)
2. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semantic Web J. (2016), to appear
3. Lanti, D., Rezk, M., Xiao, G., Calvanese, D.: The NPD benchmark: Reality check for OBDA systems. In: Proc. of EDBT. pp. 617–628. OpenProceedings.org (2015)
4. Lanti, D., Xiao, G., Calvanese, D.: Fast and simple data scaling for OBDA benchmarks. In: Proc. of BLINK (2016), to appear
5. Lanti, D., Xiao, G., Calvanese, D.: VIG. `https://github.com/ontop/vig` (2016)
6. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics X, 133–173 (2008)
7. Tay, Y., Dai, B.T., Wang, D.T., Sun, E.Y., Lin, Y., Lin, Y.: UpSizeR: Synthetically scaling an empirical relational database. Information Systems 38(8), 1168 – 1183 (2013)