Optique – Zooming In on Big Data Access

Martin Giese, University of Oslo, Norway, martingi@ifi.uio.no
Peter Haase, fluid Operations AG, Walldorf, Germany, phaase@gmail.com
Ernesto Jiménez-Ruiz, University of Oxford, UK, ernesto@cs.ox.ac.uk
Davide Lanti, Free University of Bozen-Bolzano, Italy, dlanti@inf.unibz.it
Özgür Özçep, University of Lübeck, Germany, oezcep@ifis.uni-luebeck.de
Martin Rezk, Free University of Bozen-Bolzano, Italy, mrezk@inf.unibz.it
Riccardo Rosati, Sapienza Università di Roma, Italy, roasti@dis.uniroma1.it
Ahmet Soylu, University of Oslo and Gjøvik University College, Norway, ahmets@ifi.uio.no
Guillermo Vega-Gorgojo, University of Oslo, Norway, guiveg@ifi.uio.no
Arild Waaler, University of Oslo, Norway, arild@ifi.uio.no
Guohui Xiao, Free University of Bozen-Bolzano, Italy, xiao@inf.unibz.it

Abstract

Despite the dramatic growth of data accumulated by enterprises, obtaining value out of it is extremely challenging. In particular, the data access bottleneck prevents domain experts from getting the right piece of data within a constrained time frame. The Optique Platform unlocks the access to Big Data by providing end users support for directly formulating their information needs through an intuitive visual query interface. The submitted query is then transformed into highly optimized queries over the data sources, which may include streaming data, and exploiting massive parallelism in the backend whenever possible. The Optique Platform thus responds to one major challenge posed by Big Data in data-intensive industrial settings.

Keywords

8.II.VIII.VI Knowledge management applications, 8.II.IV.IV Distributed databases, 8.V.II.VI Graphical user interfaces, 8.II.IV.VIII Query processing, 8.II.IV.XIII Temporal databases

I. INTRODUCTION

In order for end users to create value out of the vast resources of data that are now to a rapidly increasing degree available, they need to be able to explore the data in ways not foreseen when the data was stored. A key aspect of this is the ability to flexibly access the data, and in particular the ability to pose *ad hoc* queries that join information from different sources. Any system that, in the context of Big Data, pretends to put such capabilities in the hands of the end users must provide a wide range of advanced functionalities. The designers of such solutions must at least seek to touch a balance on the following, partially contradictory, requirements:

- *Prerequisites:* The system must have an acceptable installation overhead. In the context of Big Data this will in most cases rule out solutions that require, e.g., a massive reorganization of data sources or a massive manual re-engineering of meta-data, like tagging.
- *Usability:* End-users must be able to access the data on their own, without the need for a specialized IT support staff. In the context of Big Data, this requires that most of the complexity of the various data sources must by default be hidden from the end users. End users must then be exposed to details on demand, in a gradual and controlled way.
- *Scalability:* The solution must scale, not only with respect to the volumes of Big Data, but also with respect to schema complexity and data velocity.
- *Scope:* Big Data scenarios typically involve a wide variety of data types and sources, and a combination of static and streaming data arriving at a high velocity. The system must be able to accommodate the variety and velocity of the underlying data sources.

In this paper, we present the Optique Platform, a new and novel solution to one of the great challenges posed by Big Data: End users' difficulty of getting hold of the data needed for a particular task within the time frame they have for accessing it. We thus concentrate on the access to large, *complex* and *structured* corporate data sources, i.e. not only large volumes of data, but also complex schemata, with varied types, and, often, streaming data. The end users we target are domain experts from engineering disciplines, e.g. hydrocarbon exploration, who are not skilled in database access or data manipulation. Nevertheless, they require the ability to pose *ad hoc* complex, structured queries, typically combining information from a variety of tables.

If we disregard for a moment the extreme size and complexity and the streaming aspect, the shape of the data and the targeted queries is therefore similar to the situation in conventional relational data stores. In order to provide an automated, end-to-end connection between complex information needs and relational data stores, a technology known as *Ontology Based Data Access (OBDA)* [1, 2] has recently emerged. The central idea of OBDA is to allow end users to express their information needs using familiar and comprehensible terms, and to translate these information needs into queries over the data sources.

More specifically, OBDA uses an *ontology* to describe the end users' domain vocabulary in a way that allows a rigorous mathematical interpretation. End user queries are formulated using that ontology. On the other hand a set of *mappings* is provided that describes for each concept, relationship, and attribute in the ontology, how it is represented in the data sources. The ontology and mappings together form a declarative description of the OBDA setup that can be used to drive the query translation process.

A number of prototypes implementing the OBDA idea have been developed, but they concentrate mostly on the aspect of rewriting formal relational queries from an end user schema to a source schema. This is important and challenging, but we argue that it is only one aspect of what is required to provide an end-to-end solution for accessing Big Data sources. In the remainder of this article we will point out some ways in which the conventional OBDA approach breaks down under the stress of Big Data, and present how the Optique Platform remedies the situation.

II. The Optique Platform

The Optique Platform [1, 2] is based on OBDA, but goes beyond the OBDA paradigm by addressing the four requirements for Big Data Access from Section I: (i) *Prerequisites:* OBDA requires an ontology and mappings. In a Big Data setting, these can become enormous artefacts that need to be authored and maintained. (ii) *Usability:* most end users cannot be expected to know a formal query language to formulate their information needs. They need the support of a user interface for query formulation, and one that can help them to find their way in an ontology with thousands of concepts. (iii) *Scalability:* existing OBDA approaches do not deal well with very large ontologies and mappings. (iv) *Scope:* much of the relevant data is often more than 'relational,' it may be temporal, streaming, geospatial, etc., and a system needs to be equipped to handle such data types appropriately.

The Optique project[1] is addressing these issues, driven by real-world case studies provided by Siemens AG and Statoil ASA. The project is developing an integrated end-to-end platform for end user access to Big Data, according to the architecture in Figure 1. The configuration of the platform is done by IT-experts, using a bespoke ontology and mapping management component. An important part of this is the 'bootstrapping' of ontologies and mappings from existing data models and ontologies, and the source schemas, as described in Section III. Configuration artefacts such as the ontologies and mappings, but also previously formulated queries are stored in the platform's central semantic repository.

[1] http://www.optique-project.eu/

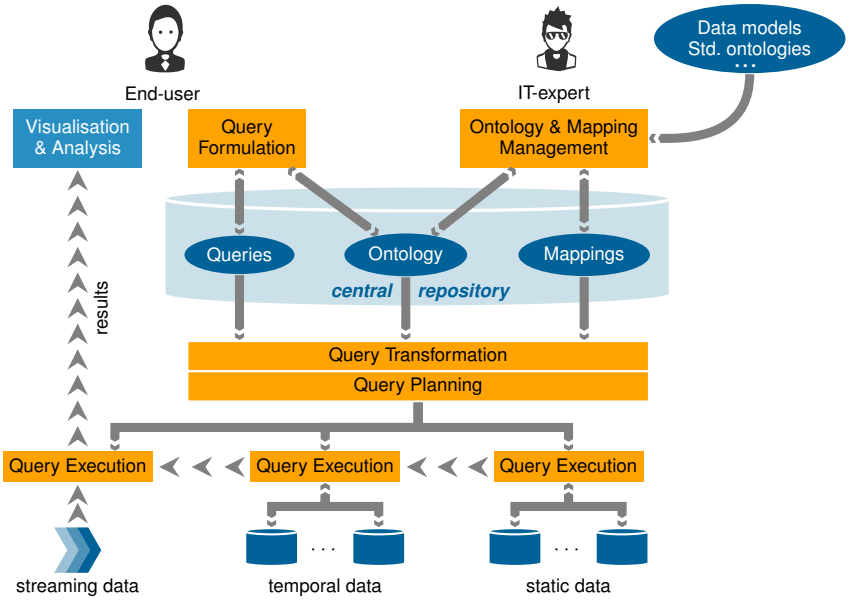

Figure 1: The Optique platform can access static, temporal, and streaming data in diverse sources; artefacts like queries, ontologies, and mappings are kept in a central semantic repository.

End users formulate queries using the query formulation interface described in Section IV. End-user queries then get transformed into queries over the data sources, using the techniques of Section V. A planning component will then prepare them for execution; possibly by federation over several data sources, and possibly by including stream processing as outlined in Section VI. As results become available, they are passed on to the end-user's existing visualisation and analysis tools.

The Optique platform is designed to provide end user access to data without moving it out of the storage in which it currently resides. Typically this will be a RDBMS. In addition to connecting to standard RDBMS, the Optique platform includes the Exareme system [3] as query execution component. Exareme is a massively parallel database back-end that will improve scalability whenever the application allows relocating the data to an elastic cloud. Moreover, the capabilities of Exareme are used for federation between multiple data sources, and to combine streaming and static data. Exareme also makes it easy to interface with alternative storage layers like NoSQL or graph-based stores. This way, the Optique platform is not limited to RDBMS backends.

In the remainder of this article we present the different components of the Optique platform, stressing how the Big Data challenges are addressed. To illustrate our points we employ the following running example: *Which hydrocarbon fields contain a wellbore with gas content*.

We assume that the data is stored according to the following relational schema:

- ***Well***(*id*, *name*, *type*)
- ***Field***(*id*, *name*)
- ***Wellbore***(*id*, *name*, *content*, *well_fk*, *field_fk*)
- ***DevelopmentWellbore***(*wellbore_fk*, *production_facility*)

where primary keys are underlined, and foreign keys' names end in *_fk*.

This example is based on a case study we performed on the Norwegian Petroleum Directorate's FactPages data set [4]. This is one of the information sources routinely used by domain experts in the Optique project's Statoil case study. The query is similar to real information needs in the Statoil case study, but kept

simple for the purpose of illustration. Note that there is a dedicated table in the schema for development wellbores, which are a special kind of wellbore.

III. Reducing Installation Overhead

Building an ontology and connecting it to the data sources via mappings is a costly process, especially for large and complex databases. To aid this process, tools that can extract a preliminary ontology and mappings from the source schema are useful. To improve the quality of the ontology and mappings, but also to maintain them, e.g. when source schemas change, tool support for editing ontologies and mappings is crucial. While ontology editing is well covered by existing tools, mapping editors are not as readily available, which is why such an editor [5] is developed as part of the Optique platform.

In order to ease the production of initial versions of the ontology and mappings, the Optique platform includes a *bootstrapping* component that takes a set of database schemata as input, and constructs an ontology and a set of mappings that connect the terms occurring in the ontology to the schema elements. 'Guessing' an ontology from a database schema is no easy task, since the database modelling step that produces a database schema from (explicit or implicit) knowledge of a domain is typically lossy. Automatic bootstrapping means to make a best effort at reverse engineering this step and is necessarily imperfect. However, depending on the quality of the data source schemata, the results often provide a very good starting point for later manual optimisations that can be applied as required.

Our current implementation is based on the approach called 'direct mapping' by the W3C:[2] every table in the database (except for those representing many-to-many relationships) is mapped to one class in the ontology; every data attribute is mapped to one data property; and every foreign key to one object property.

Additionally, explicit and implicit database constraints from the schema can be used to enrich the ontology with axioms about the classes and properties from these direct mappings. For instance, in the example schema, the DevelopmentWellbore table's primary key is also a foreign key pointing to the Wellbore table. It turns out to be an effective heuristic in such cases to add class inclusion axioms between the directly mapped classes, in this case $\mathsf{DevelopmentWellbore} \sqsubseteq \mathsf{Wellbore}$.

Even after applying such heuristics to enrich the ontology, it will still usually be too close to the source schema. It is however increasingly often the case that a high-quality ontology of (parts of) the domain already exists, that captures the domain experts' vocabulary better than the directly mapped ontology.

When such a high-quality ontology is available, the bootstrapping component allows importing it and using it in the bootstrapping process. This is achieved through a heuristic alignment of the directly mapped and the imported ontologies, using the LogMap system [6]. LogMap is a highly scalable ontology matching system that discovers ontology-to-ontology mappings, e.g. class equivalence axioms, between the vocabularies of the input ontologies.

Special care needs to be taken to avoid introducing unwanted consequences: for instance the bootstrapper will avoid adding alignment axioms that would lead to inconsistencies, or faulty consequences like $\mathsf{Well} \sqsubseteq \mathsf{WellBore}$ that are not supported by the domain ontology. This is based on novel techniques to avoid violations of the so-called consistency and conservativity principles [7].

We evaluated our bootstrapper on Statoil's corporate exploration and production data store, an SQL database comprising over 1500 tables with a total of over 19000 columns. Parts of the directly mapped ontology were automatically aligned with an independently handcrafted ontology based on the NPD FactPages data set. The resulting ontology fully covered 30%[3] of the terms of a previously collected catalogue of representative information needs of petroleum exploration experts.

---

[2] http://www.w3.org/TR/rdb-direct-mapping/

[3] 50% if considering partially covered terms (i.e. semi-true positives).

This is a very good coverage for a fully automated process that only requires a few seconds. However, to achieve a really high coverage of end-user information needs, the ontology and mappings will have to be improved further; and both have to be maintained when end user requirements or data sources change. An important insight is that the data sources, ontology and mappings together are a complex combination of artefacts where it is easy to introduce mistakes. To support the maintenance and evolution of mappings, advanced analysis techniques are required that are based on the equivalence between the SQL queries in mappings and formulae of first-order logic [8].

IV. Eliciting Information Needs

Figure 2: Snapshot of the query interface of OptiqueVQS

As discussed above, end users should be able to pose expressive queries for an arbitrary domain in order to exploit available Big Data sources. Ideally, the query specification mechanism should be usable even for users without specialised IT skills. Accordingly, the following approaches can be found in the literature:

- Query language editors allow highly expressive queries through query languages such as SPARQL[4]. However, knowledge of a query language is a too exigent requirement for end users in most cases.
- Information retrieval approaches mimic traditional keyword queries. Although simple to use, expressiveness is rather limited.
- Natural language interfaces aim to interpret a query as a whole in order to give more relevant responses than with keywords. However, ambiguities and linguistic variability limit their effectiveness.
- Visual approaches try to achieve good usability and sufficient expressiveness, although finding a good trade-off is difficult. For example, visual SPARQL query builders are still too complicated for mainstream users, while typical facet-based systems limit queries to one class.

[4] http://www.w3.org/TR/sparql11-query/

To address this challenge we have proposed a novel visual query interface named OptiqueVQS [9]. It allows the formulation of rather expressive queries through a visual user interface that hides the syntax of the query language – see Figure 2. The bottom-left widget can be used to browse the concept taxonomy and include new concepts to the query. This is shown as a tree in the top widget and can be manipulated, e.g. to select or delete a concept variable, reacting after each query change. A selected concept can be further refined using the controls of the bottom-right widget, which are built from the concept properties in the ontology. Figure 2 represents the query built with OptiqueVQS for the information need presented in Section II.

Overall, OptiqueVQS integrates concept browsing, facet-based search and visual query manipulation in the query interface. Behind the scenes, a SPARQL query is built and sent to the query transforming component of the Optique platform – for our running example we obtain:

```
SELECT ?field WHERE {
  ?field a npdfp:Field .
  ?wb a npdfp:Wellbore ;
      npdfp:wellboreForField ?field ;
      npdfp:wellboreContent "GAS"^^xsd:string .
}
```

Since ontologies can be very large in a Big Data context, OptiqueVQS addresses this challenge by gradually loading on demand the information about classes, and offering input fields to find classes and properties without having to navigate endless lists. Moreover, its widget-based architecture can be customized for different needs, e.g. we have developed widgets to select geospatial objects on a map, or to specify a time window for streaming data.

We have tested OptiqueVQS in four preliminary user studies, two of which were performed with targeted end-users in Siemens AG and Statoil ASA. Participants were able to formulate non-trivial information needs using this search interface, even without previous training. After using the system for about an hour, all participants were able to author queries joining up to 8 relations. These results support the central hypothesis of query formulation in Optique: that end users can interact with a query, instead of directly with the data, if the query interface is implemented with care.

V. Scaling Up Query Transformation

Query transformation is the task of rewriting end-user queries over the data sources by taking into account the ontology and mappings. With this approach there is no need to chase GBs of data to generate all the facts derived by the ontology. However, query transformation can be challenging and even prohibitive in presence of large ontologies, complex schemas, and numerous mappings. The *Ontop* query transformation system[5] used in Optique handles this problem by embedding the consequences of the ontology into the mappings, generating so called $\mathcal{T}$-mappings. This is done off-line, i.e. independently of any concrete query. End-user queries can then be rewritten disregarding the ontology and considering only a small set of necessary mappings.

For instance, in the running example, we have the axiom

$$\text{DevelopmentWellbore} \sqsubseteq \text{Wellbore}$$

and the following mappings to the relational schema

Wellbore(*id*) ←SELECT *id* FROM `Wellbore`

DevelopmentWellbore(*wellbore_fk*) ←SELECT *wellbore_fk* FROM `DevelopmentWellbore`

[5] http://ontop.inf.unibz.it/

Ontop will embed the axiom above in the mappings by generating the following new $\mathcal{T}'$-mapping:

Wellbore(*id*) ←
SELECT *id* FROM `Wellbore`
UNION
SELECT *wellbore_fk* AS *id* FROM `DevelopmentWellbore`

Then to answer the triple pattern

```
?wb a :Wellbore
```

in our running example, we only need to consider the above $\mathcal{T}'$-mapping and the data source.

The architecture of *Ontop* is depicted in Figure 3. The input consists of an ontology, a database—with relative integrity constraints, and the mappings connecting the ontology and the database.

The *off-line* stage of *Ontop,* in the right part of the Figure, pre-processes the ontology, mappings and database schema as follows:

1) The ontology is classified using an ontology reasoner.
2) $\mathcal{T}'$-mappings are constructed to simplify query rewritings drastically.
3) The $\mathcal{T}'$-mappings are then optimised in order to eliminate redundancies.

The *on-line* stage in the left part of the Figure takes a SPARQL query and unfolds it with respect to the $\mathcal{T}'$-mappings. Then, it translates this query into SQL and optimizes it. The optimized query is executed by the data source.

We have shown using a range of standard benchmarks that *Ontop* outperforms many commercial triple stores with and without reasoning enabled, see e.g. [10].

In order to test the efficiency of *Ontop* in a challenging OBDA setting, we have created a scalable benchmark based on the previously mentioned NPD FactPages data set, ontology, and mappings [4], as well as a set of real-world queries created by users of the FactPages. The benchmark is equipped with a data generator able to increase the initial dataset in order to allow a scalability analysis [11]. We took the OWL 2 QL fragment of the NPD ontology, and we obtained 343 classes, 142 object properties, 238 data properties, 1451 axioms, and maximum hierarchy depth of 10.

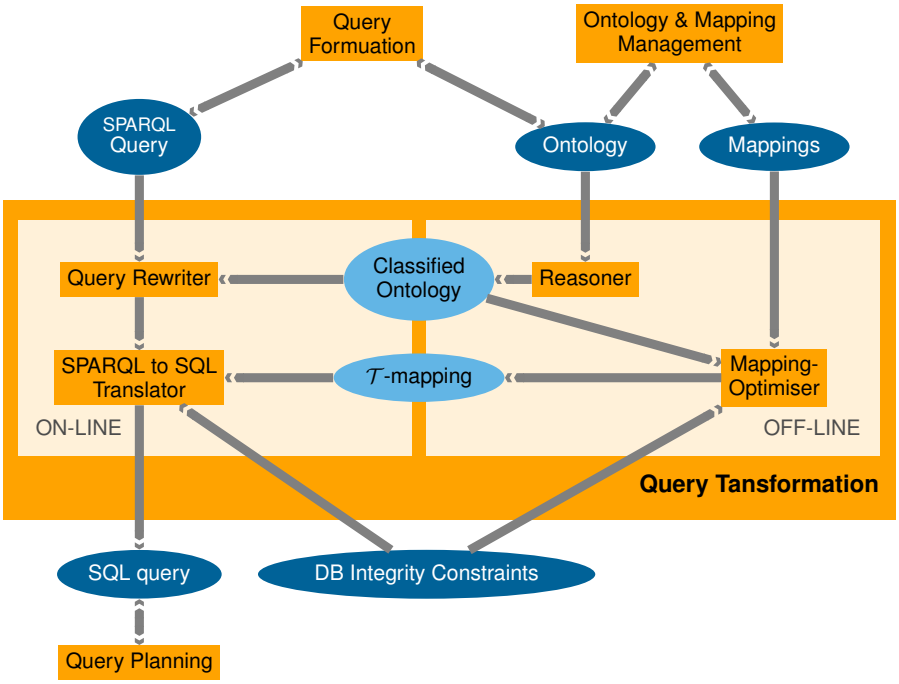

Figure 3: The architecture of *Ontop*—the query transformation component of the Optique platform. Ontology Classification and Mapping Optimisation can be carried out off-line, independent of concrete queries.

The results for *Ontop* show that most of the SPARQL queries are translated into a single select-project-join SQL query. Those SPARQL queries contain between 3 and 7 joins, filters and modifiers. This set of queries can be evaluated efficiently—11000 queries per hour on the original 2**M** triples dataset—and execution times scale linearly w.r.t. the size of the dataset—14 queries per hour on the 4**B** triples dataset. The benchmark also contains more challenging queries, which translate to a union of multiple SQL queries. Most of these take 1–2 seconds to execute already in the 2**M** dataset.[6]

VI. Streaming Queries

Most Big Data processing involves a temporal dimension—be it as time attributes within a static DB, or as timestamps on data items arriving in streams. Hence, as the Optique platform is intended to be used in a wide range of scenarios it comes with means for querying temporal and streaming data.

In contrast to classical OBDA systems, the Optique platform does not constrain the backend sources to relational DBMS but allows them to be relational data stream management system (DSMS). Relational DSMS are equipped with SQL like query languages that offer stream operators (such as window operators) for querying relational streams, i.e., potentially infinite sets of time stamped relational tuples. In the Optique prototype, the DSMS used is the highly distributable SQL backend system Exareme [3].

Due to this extension of data sources to DSMS, the mappings in Optique do not generate (virtual) static data only, which can be processed by ontology-level query languages such as SPARQL. But they may also generate (virtual) ontology-level streams, which cannot be processed with SPARQL. Trying to meet the need for a query language over ontology-level streams, one faces challenges that are different from the ones

[6] The benchmark is available online: https://github.com/ontop/npd-benchmark/

known from classical stream analytics. In fact, for Optique, none of the recent ontology-level stream engines relying on SPARQL extensions was adequate. The reason is that these engines implement time-oblivious window operators which lead to potential inconsistencies with natural functionality constraints the engineer wants to formulate: For example, the amount of gas produced by a wellbore is unique at every time point but may be different at different time points. So, in Optique the new query language framework STARQL (Streaming and Temporal ontology Access with a Reasoning-Based Query Language) [12] was developed and integrated into the architecture.

In order to illustrate the functions and use of STARQL, we extend the running example from the beginning of the paper: For all fields containing a well with gas content, compute a moving 3 month average over the field's gas production. This can be done using the following STARQL query:

```
CREATE STREAM Sout AS
SELECT ?field, AVG(?p), NOW
FROM proddata[NOW-3month, NOW]->1 month, npdfp
WHERE {
       ?field a :Field .
       ?wb a :Wellbore ;
       :wellboreForField ?field ;
       :wellboreContent "GAS"^^xsd:string .}
SEQUENCE BY StdSeq AS SEQ
HAVING EXISTS i in SEQ:
       GRAPH i : { ?wb :producedGas ?p }
```

In STARQL, querying historical data and querying streaming data proceeds in an analogous way. In both cases, the query may refer to static data, i.e., data that is not equipped with timestamps as they are assumed to be non-temporal or to hold at every time point. In general, the static part of a STARQL query is formulated in the WHERE clause. For instance, the example query asks for wellbores containing gas, which is assumed to be a static property for this query. Answers produced by the static sub-query are used as a pre-filter for the stream processing in the remainder of the query. This separation facilitates processing high volume static data and high velocity streaming data within one query.

Relevant slices of the temporal data are specified with a window. It contains a reference to the developing time NOW and a sliding parameter that determines the rate at which snapshots of the data are taken. The example query has a three-month sliding window over gas production data stored in `proddata`. The contents of the temporal data are grouped according to a sequencing strategy into a sequence of small graphs that represent different states. In the example, the strategy used is standard sequencing according to which assertions with the same timestamps come into the same graph.

On top of the sequence, relevant patterns, and aggregations are formulated in the `HAVING`-clause, using an expressive template language. The example query asks for every three-month snapshot whether there is a state i such that field ?f produced gas volume ?p at that state.

The sequencing idea underlying STARQL is a necessary component for a proper treatment of ontology-level streams. Though this is not a direct contribution to the challenges of classical stream analytics (e.g., the problems of concept drift for reliable prediction), the sequencing strategy parameter in combination with the template language of the `HAVING` clause provides the place for implementing various adaptive strategies.

VII. CONCLUSION

Giving end users with limited IT expertise flexible access to Big Data is a major bottleneck in data-intensive industries. We have argued how ontology-based data access (OBDA) can provide a solution: By capturing the end users' vocabulary in a formal model (ontology), and maintaining a set of mappings from this vocabulary to the data sources, we can automate the translation work previously done by the IT-experts. Yet current OBDA systems lack a holistic approach to the problem of establishing an integrated, end-to-end connection from the end user to large scale and distributed data sources. Specific problems pertain to usability, prerequisites, scope, and scalability. We have shown how these problems can be overcome by a novel combination of techniques, encompassing an end user oriented query interface for

eliciting information needs, semi-automated methods for managing ontologies and mappings, new techniques for scalable query rewriting, temporal and streaming data processing.

These techniques are implemented in an integrated platform, in which all components ranging from low-level distributed query execution to visual end user interfaces interact seamlessly. To tackle the issue of cross-component communication and optimization, components communicate through a unified semantic layer of abstraction. Knowledge artefacts such as ontologies, mappings, metadata and queries are stored in the central semantic repository based on open standards such as OWL 2 QL for ontologies, SPARQL as a query language, and R2MRL for the mappings. As a result, the Optique platform provides a single point of entry for administrative tasks (e.g. management of mappings and ontologies) as well as visual components through which end users can satisfy their information needs in interacting with Big Data.

The authors would like to thank all other members of the Optique project for their contributions to the project; and in particular Diego Calvanese, Ian Horrocks, Evgeny Kharlamov, Ralf Möller, Domenico Fabio Savo, Evgenij Thorstensen, and Dmitriy Zheleznyakov for their contributions to and comments on drafts of this manuscript.

Acknowledgement

This work has been funded by the European Commission under FP7 Grant Agreement 318338, "Optique."

REFERENCES

[1] M. Giese *et al*., "Scalable end-user access to Big Data," in *Big Data Computing*, R. Akerkar, Ed. CRC Press, 2013.

[2] E. Kharlamov, E. Jiménez-Ruiz, D. Zheleznyakov *et al*., "Optique: Towards OBDA systems for industry," in *Eur. Sem. Web Conf. (ESWC) Satellite Events*, 2013, pp. 125–140.

[3] H. Kllapi, E. Sitaridi, M. M. Tsangaris, and Y. E. Ioannidis, "Schedule optimization for data processing flows on the cloud," in *SIGMOD Conference'11*, 2011, pp. 289–300.

[4] M. G. Skjæveland, E. H. Lian, and I. Horrocks, "Publishing the Norwegian Petroleum Directorate's FactPages as semantic web data," in *Proc. ISWC 2013*, H. Alani *et al*., Eds., vol. 8219. Springer, 2013, pp. 162–177.

[5] K. Sengupta, P. Haase, M. Schmidt, and P. Hitzler, "Editing R2RML mappings made easy," in *ISWC 2013 Posters & Demos*, 2013, pp. 101–104.

[6] E. Jiménez-Ruiz, B. C. Grau, Y. Zhou, and I. Horrocks, "Large-scale interactive ontology matching: Algorithms and implementation," in *Proc. ECAI 2012*, 2012, pp. 444–449.

[7] A. Solimando, E. Jiménez-Ruiz, and G. Guerrini, "Detecting and correcting conservativity principle violations in ontology-to-ontology mappings," in *Proc. ISWC 2014*, 2014.

[8] D. Lembo, J. Mora, R. Rosati, D. F. Savo, and E. Thorstensen, "Towards mapping analysis in ontology-based data access," in *Proceedings of RR 2014*, 2014, to appear.

[9] A. Soylu *et al*., "OptiqueVQS – towards an ontology-based visual query system for Big Data," in *Proc. MEDES 2013*. ACM, 2013.

[10] M. Rodriguez-Muro *et al*., "Evaluating SPARQL-to-SQL translation in ontop," in *Proc. OWL Reasoner Evaluation (ORE 2013)*, ser. CEUR, vol. 1015, 2013, pp. 94–100.

[11] D. Lanti, M. Rezk, M. Slusnys, G. Xiao, and D. Calvanese, "The NPD benchmark for OBDA systems," in Proc. 10th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2014). CEUR-WS.org, 2014, pp. 3–18.

[12] Özgür. L. Özçep, R. Möller, and C. Neuenstadt, "A stream-temporal query language for ontology based data access," in *KI 2014*, ser. LNCS, vol. 8736. Springer, 2014, pp. 183–194.

AUTHOR BIOGRAPHIES

Martin Giese is an associate professor at the Univ. of Oslo, Norway. He acts as Assistant Scientific Director of the Optique project.

Peter Haase is the head of R&D at fluid Operations. He is responsible for the technical integration of the Optique platform.

Ernesto Jiménez-Ruiz is a postdoc at the University of Oxford. He leads the ontology and mapping bootstrapping activities of Optique.

Davide Lanti is a PhD student at the Free Univ. of Bozen-Bolzano, Italy. He conducts research on benchmarking and optimization of OBDA systems.

Özgür Özçep is a postdoc at the Univ. of Lübeck doing research on spatio-temporal and stream reasoning.

Martin Rezk is a postdoc at the Free Univ. of Bozen-Bolzano, Italy. He leads the development of *Ontop*, the query transformation component of Optique.

Riccardo Rosati is an associate professor at Sapienza University of Rome. He leads the ontology and mapping management workpackage of Optique.

Ahmet Soylu is a postdoc at Univ. of Oslo and an associate professor at Gjøvik University College. He conducts research on ontology-based visual query formulation.

Guillermo Vega-Gorgojo is a postdoc at the Univ. of Oslo. He researches on Semantic Web user interfaces and Big Data.

Arild Waaler is a professor at the Univ. of Oslo. He is the Coordinator of the Optique project.

Guohui Xiao is an assistant professor at the Free Univ. of Bozen-Bolzano, Italy. He conducts research on query transformation in OBDA.