Ontology-Based Data Access to Slegge

Dag Hovland[1], Roman Kontchakov[2(✉)], Martin G. Skjæveland[1], Arild Waaler[1], and M. Zakharyaschev[2]

[1] Department of Informatics, University of Oslo, Oslo, Norway
{hovland,martige,arild}@ifi.uio.no
[2] Department of Computer Science and Information Systems, Birkbeck, University of London, London, UK
{roman,michael}@dcs.bbk.ac.uk

Abstract. We report on our experience in ontology-based data access to the Slegge database at Statoil and share the resources employed in this use case: end-user information needs (in natural language), their translations into SPARQL, the Subsurface Exploration Ontology, the schema of the Slegge database with integrity constraints, and the mappings connecting the ontology and the schema.

1 Introduction

We present the resources developed for ontology-based data access (OBDA) to the Slegge database at the international oil and gas company Statoil using the OBDA system Optique platform [7]. In the OBDA paradigm based on query rewriting [9], the data remains stored in the original DBMS, while user queries are formulated in terms of an OWL 2 QL ontology designed specifically for the end-users—rather than directly over the database, which presupposes detailed knowledge of the database schema and requires an assistance of an IT expert. The OBDA system makes use of the mappings that relate the ontology vocabulary to the database schema to transform the ontology-mediated queries into standard SQL queries, which are then executed by the DBMS.

OBDA has been an active research area since the mid 2000s, with OBDA systems used in a variety of projects within both academia and industry, e.g., [1–4,6,11,13]. However, full details of an industrial use case have never been made publicly available. The main aim of this paper is to fill this gap by publishing the following complete set of OBDA specifications for the Slegge use case:

- the Subsurface Exploration OWL Ontology specifically designed to capture the terms used by the geologists at Statoil when querying subsurface exploration data;
- the typical information needs (in natural language) and respective SPARQL queries;
- the SQL schema of the Slegge database;
- the R2RML mappings connecting the ontology vocabulary to the schema;
- statistics on the Slegge data (the data is private and cannot be made public).

C. d'Amato et al. (Eds.): ISWC 2017, Part II, LNCS 10588, pp. 120–129, 2017.
DOI: 10.1007/978-3-319-68204-4_12

The Slegge schema demonstrates the intrinsic complexity of industrial databases, which is reflected in the mappings that encode the semantics of the data using, in particular, multiple joins. The large body of queries are collected from domain experts at Statoil, totalling 73 natural language information needs and 96 corresponding SPARQL queries. The ontology captures the vocabulary of the SPARQL queries and describes parts of the petroleum subsurface exploration domain represented in Slegge. Since the resources we publish include all the intricacies and peculiarities of a large industrial setup, we believe they will be useful to the developers of OBDA technologies and to the wider semantic technologies and information systems communities. In particular, the resources could be used for benchmarking query rewriting and optimising engines and for development of methods and tools for ontology and mapping construction and analysis. The Slegge resources and an extended version of this paper are available at http://purl.org/slegge under the CC Attribution 4.0 International Public License.

The main feature distinguishing the Slegge resources from other available OBDA specifications is that Slegge straddles the long distance between two industrial artefacts. Our starting points were the Statoil geologists' information needs (Sect. 3) and the large industrial database with hundreds of tables (Sect. 4). We designed the ontology capturing the vocabulary of the needs in the context of oil and gas exploration (Sect. 3) and the complex mappings bridging the substantial conceptual gap between the ontology and the data (Sect. 5). Other publicly available OBDA specifications for databases with real data include FishMark [2], IMDb OBDA [11] and NPD FactPages [14]. Their ontologies, however, are quite similar to their database schemas, and so the mappings are almost direct (with very few joins). The NPD FactPages comes with simple queries generic for the oil and gas domain, while FishMark and IMDb OBDA only contain queries stemming from existing SQL queries or invented by the authors. The Texas benchmark [12] and the OBDA extensions of the Berlin SPARQL Benchmark (BSBM) [12] and LUBM [11] are all examples with synthetic data based on existing non-OBDA benchmarks, where the mappings are also almost direct.

2 Data Gathering at Statoil

The task of the exploration department at Statoil is to find exploitable deposits of oil and gas. Geoscientists model the subsurface geography by classifying rock layers according to multiple stratigraphic hierarchies using information from various sources. This model largely determines the location of new wellbores for direct exploration and possible exploitation of the hydrocarbon resource. Since wellbore drilling is a major expense, the quality of the model, which depends on the availability of and the ease of accessing the relevant data, becomes a crucial factor for efficiency of the exploration process.

We illustrate the current workflow with a typical domain expert *information need*, which is an informal natural language description of a user question:

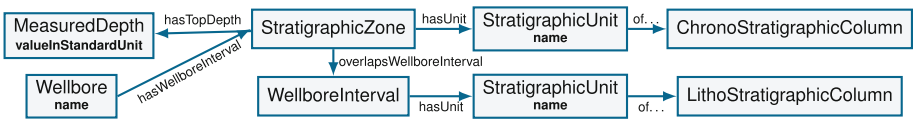
(001) In my area of interest, return the wellbores penetrating a given chronostratigraphic unit X and return information about the lithostratigraphy and the hydrocarbon content in the wellbore interval that penetrates X. Also return information about other wellbore intervals with hydrocarbon content in the wellbores with hydrocarbon in X.

To find answers, the geologist will use pre-defined SQL queries covering parts of the information need and then integrate their results, often with primitive data management tools such as spreadsheets. This process is onerous and error-prone: for example, the comparison of depths in a wellbore can easily go wrong as there are multiple units, reference points and types of depth measurements. An alternative solution would be to ask the IT department to construct a custom SQL query. However, employing IT personnel for the task generally takes days (or even weeks) because there are very few people with the rare combination of intimate knowledge of the geological domain *and* database structure required to translate the information needs into database queries.

The OBDA paradigm [9] offers a third alternative, where an *ontology* describes the geologists' vocabulary. For example, given an ontology in the W3C standard language OWL 2 QL containing classes such as `Wellbore`, `StratigraphicUnit`, `MeasuredDepth`, and properties such as `name`, `hasUnit`, `hasWellboreInterval`, `valueInStandardUnit`, the geologist can recast information need **(001)** more formally in the following way:

(001/02′) Give me the names of the available wellbores with the chronostratigraphic units and the top depths of the intervals they were found in; the depths should be in the standard units and from standard reference points (metres along the drill string). Also, return all lithostratigraphic units from depths overlapping the depths at which the chronostratigraphic units were found.

And, following the structure of the ontology, the geologist can easily formalise such a query with, e.g., the visual query interface OptiqueVQS [15] of the Optique platform:



The formalised query **(001/02′)** is automatically translated into a SPARQL query, and an OBDA tool such as Ontop [5,11] will use the *mappings* to 'rewrite' the *ontology-mediated query* into an SQL query over the database, optimise and execute it, returning

?wellbore	?chronostrat_unit	?lithostrat_unit	?top_md_m
"NO 1/2-1"	"Jurassic"	"Fisk"	"1234.5"

So, in the OBDA paradigm, the geologist does not need to know the structure of the database to create new queries. And, instead of supporting geologists directly (by translating their information needs into SQL), the IT expert has an easier task of constructing and maintaining mappings that populate the ontology classes and properties with data from the database. Thus, mappings explicate the IT expert's knowledge of the database. The reader can find the rewritten SQL query in the full paper and appreciate the knowledge required from the IT expert to produce an SQL query over Slegge by hand.

3 From Information Needs to Ontology and SPARQL Queries

The starting point of the Slegge use case was a list of 73 **information needs** collected from end-users at Statoil over a period of four years. It turned out that 39 information needs are beyond the scope of the Slegge database: they concern user interface configuration, data entry processes or require data unavailable in Slegge. The remaining 34 information needs provided the basic *competency questions* for creating the Subsurface Exploration Ontology, which gives the vocabulary (ontology classes and properties) for translating the information needs into SPARQL queries. We publish all 73 information needs as they can be useful for the research in natural language processing and for the future work on other data sources at Statoil.

The **Subsurface Exploration Ontology** describes parts of the petroleum subsurface exploration domain and captures the classes and properties from the user information needs. Class `Wellbore` represents a path drilled through the Earth crust. Rock samples (class `Core`) are normally extracted from the wellbore during drilling. Smaller samples (`CoreSample`) are drilled out of the core and used for direct visual and experimental observations. A `WellboreInterval` is a depth interval along a wellbore, defined by its top and bottom depths. It has two natural subclasses: `Reservoir` and `StratigraphicZone`.

Numerous measurements taken from wellbores are modelled by the taxonomy under the class `Measurement`, with subclasses such as `TrueVerticalDepth`, `Permeability` and `FormationPressure`. Each measurement provides a value in the *standard* and in the *original* units because translation from a variety of units in the database to the standard ones may mask suspicious values, e.g., depth 9999 ft. Since wellbores are not necessarily vertical, there are two types of depth for relating points along them: `MeasuredDepth` refers to the length along the wellbore or drill string, while `TrueVerticalDepth` is the length of the normal to the reference surface, usually the mean sea level.

To represent geographical objects and connect Slegge to other Statoil data sources, we imported class `SpatialObject` (with its subclasses) from GeoSPARQL 1.1.

The resulting Subsurface Exploration Ontology has 71 classes, 46 object properties and 34 data properties. The depth of the class hierarchy (without `SpatialObject`) is 5, and the depth of the property hierarchy is 4. The existential depth, which measures the length of chains of labelled nulls (caused by

existential quantifiers in the ontology), is 5: for instance, every `Permeability` must be related by the inverse of property `hasPermeability` to some `CoreSample`, which in turn is related by the inverse of `hasCoreSample` to a `Core`; every `Core` is `extractedFrom` some `WellboreInterval` linked by the inverse of `hasWellboreInterval` to a `Wellbore` and then to a `Well` via the inverse of `hasWellbore`. Note, however, that the structure of the mappings and database integrity constraints make sure that wherever there is a `Permeability`, the data itself contains the required chain of length 5 as above, and so the corresponding labelled nulls in the chase are not needed. This fact substantially simplifies query rewriting.

We incorporated some background knowledge in the ontology even if it required constructs unavailable in OWL 2 QL such as functionality of properties and local range constraints of the form $\texttt{DrillingOperation} \sqsubseteq \forall \texttt{hasActivityPart}.\texttt{WellboreDrilling}$. Fortunately, the structure of the mappings and database imply most of the non-OWL 2 QL axioms, while the remaining ones are not relevant for the mappings. The smallest standard description logic capable of representing the ontology is Horn-$\mathcal{ALCHIQ}(\mathcal{D})$.

Each of the 34 information needs in the scope of Slegge was recast in SPARQL. The resulting 96 **SPARQL queries** (some information needs are vague and can be interpreted in SPARQL in different ways) were constructed manually, either by hand or using OptiqueVQS. These queries have an average of 13 triple patterns, ranging from 3 to 30; 16 queries use OPTIONAL and 3 use FILTER NOT EXISTS. Most queries capture only part of the corresponding information need, often because some data is not available in Slegge. There is also a considerable overlap among the SPARQL queries because the information needs overlap too; these mainly include different features of wellbores and their surroundings. Many information needs, e.g., **(001)**, contain the expression 'for my area of interest', which could be interpreted as 'restrict the query to the geographical area I am interested in'. There is no general translation of such needs into SPARQL, but many queries such as **(001/02′)** in the query catalogue use concrete geographical areas in the North Sea identified by coordinates (in the example in Sect. 2, we omitted the coordinates for simplicity).

4 Slegge Database

Slegge is an Oracle database with about 700 GB of data. Its schema was initially constructed in the late 1990s on the basis of Epicentre v2.2. The Epicentre data model had been developed by the Petrotechnical Open Standards Consortium (POSC) since the early 1990s, and its latest v3.0 is maintained by Energistics [10]. It defines the object-oriented logical database model and its standard projection to the physical model in an Oracle database. The main features of Epicentre and its implementation in Slegge are:

– extensive inheritance hierarchies are projected by two methods: (a) a table per subtype and (b) a single table for all subtypes with a discriminating column;

– denormalisation: many columns are duplicated to avoid joins when querying;
– lack of foreign keys: many relationships involve multiple tables for subtypes, and so foreign keys would have to be *conditional*, which is not supported by the DBMS.

Entities `well` and `wellbore` are subtypes of `facility`. Abstract entities such as `facility` have no database tables, but each of the non-abstract entities is represented by a table: e.g., `WELL` for entity `well`. These tables contain columns for the normal attributes of the entity: `WELLBORE` has a `COMPLETION_DATE` column for the date when the wellbore was available for service. Tables for subtypes 'inherit' columns from supertypes: e.g., tables for all sorts of facilities inherit column `R_EXISTENCE_KD_NM` to specify whether the facility is actual, planned, etc. Instances of entities (objects) are identified by their unique IDs, which are surrogate primary keys in the tables: e.g., column `WELL_S` in table `WELL`. On the other hand, the (user-friendly) well identifiers (attribute `identifier` of the supertype entity `facility`) are represented in column `WELL_ID`.

Epicentre also follows an alternative approach to hierarchies, where all subtypes of an entity are projected to the same table, and a *discriminating column* specifies the object's subtype: e.g., the four subtypes of `stratigraphic_marker` are mapped to the same table, `STRAT_MRK`, with column `ENTITY_TYPE_NM` containing the subtype name.

For composite attributes (such as quantities with units of measure), Epicentre uses *properties*, which, like entities, have instances and are arranged in an extensive hierarchy. Properties are normally projected to tables. However, many attributes in Slegge are *denormalised*: e.g., table `WELL_SURFACE_PT` for the `well_surface_point` entity has columns `WATER_DEPTH` and `WATER_DEPTH_U` to store the value and the unit of measure of the water depth property `pty_water_depth`. So, values are stored directly in the 'entity' table rather than in the property table `P_WATER_DEPTH`, which is empty in the database. Only 20 (out of 543) property tables are non-empty in Slegge: e.g., coordinates of wells' surface points are stored in the table `P_LOCATION_2D` for property `pty_location_2d`.

Reference entities collect standard values: e.g., `ref_unit_of_measure` is projected to table `R_UOM` with information about 974 units of measure (others are much smaller). The primary key in such a table is often referenced by foreign keys of entity and property tables: e.g., `WATER_DEPTH_U` is one of 814 columns referencing `ACRONYM` in `R_UOM`.

In the Epicentre data model, relationships are in fact attributes of entities: both end-points of a relationship have an attribute—one for the relationship, the other for its inverse. *One-to-many relationships* are normally projected as columns in the tables. For example, column `WELL_S` in `WELLBORE` specifies the identifier of the well containing this wellbore. In this exceptional case, the database contains a *foreign key*: `WELL_S` of `WELLBORE` references `WELL_S` of `WELL`. However, most of such foreign keys are missing because the subtypes are distributed over tables. For example, the relationship between `activity` and `facility` is represented by column `FACILITY_S` in `ACTIVITY`. Since facilities are covered

in a number of tables, there is no foreign key. Instead, ACTIVITY has another column, FACILITY_T, to specify the facility subtype it refers to ('WELLBORE' or 'WELL'), and so the pair FACILITY_T/_S identifies the referenced table and the row in it.

Denormalised attributes are quite common in Slegge—they reduce the number of joins in queries: e.g., table WELLBORE, along with the reference to the primary key WELL_S of WELL, contains a column duplicating WELL_ID from WELL.

Many-to-many relationships in Epicentre use association entities, which are then projected to tables: e.g., topological relationships such as 'inside' between instances of topological_object (and its subtypes field, core and facility) are modelled by the topological_relationship association entity in the table TOPOLOGICAL_REL.

The Slegge Oracle database has 6 schemas. The SLEGGE_EPI schema is a dated implementation of the Epicentre data model and consists of 1545 tables with 19 719 columns: 1141 tables are empty because large portions of the data model are not used by the tools; 221 tables contain only 1–100 rows (mostly for reference entities), but 9 tables have more than a million rows each. Schemas SLEGGE_SNP, MDS_COORD and SIS_CATALOG are much smaller (21, 14 and 1 table, respectively) and related to other applications; ENTITLED has two tables modelling user privileges. The main SLEGGE schema integrates the other five schemas and defines 1722 views to their tables. However, most of them (1632) contain no joins and no WHERE clauses—they simply rename tables and columns; two views additionally limit access in accordance with the user privileges from ENTITLED. The remaining 88 views vary from two-table joins to 31-way joins with additional WHERE and GROUP BY clauses; many also contain ORDER BY clauses, which suggests their primary use for reporting and user interfaces. In addition, SLEGGE contains 102 tables for various purposes. Finally, there are five *materialised views*, two of which are joins of 12 and 15 tables, respectively, while the other three use calls to stored procedures (with more queries and even Java code inside).

5 Mapping Ontology to Slegge Database

One of the main challenges in the project was to map the classes and properties of the ontology to database objects, which required detailed knowledge of both components. Unfortunately, the Slegge implementation does not fully comply with any version of Epicentre, and the documentation on Slegge has become either unavailable or hard to obtain at Statoil. The lack of integrity constraints (foreign keys)[1] and abundance of denormalisation made any initial attempts at automated schema analysis inefficient. The sheer size of the database (1727 views and 1685 tables) only exacerbated the problem. Our main source of information about the schema was the 2996 queries found in the configuration files of

[1] Even though SLEGGE_EPI has 3112 foreign keys, 2727 of them refer to just 18 reference tables such as R_UOM; in fact, most of the remaining 385 foreign keys refer to 'single-purpose' reference tables such as R_OBJECT_NTRS listing topological relations.

ProSource, a proprietary tool developed in-house and the *de facto* way of accessing Slegge. It turned out, however, that a significant number of these queries, in particular, the most useful ones for mappings were quite large (up to 91 joins). The distribution of the number of tables and views per query is illustrated below:

# tables/views	1	2	3	4–10	11–20	21–30	31–40	41–50	51–92
# ProSource queries	1801	545	327	228	51	18	7	10	9

Such large queries are necessary to provide the geologist users with all the information about complex domain objects such as wellbores and their litho- and chronostratigraphic columns. However, the ontology 'decomposes' such objects into a number of classes linked by object properties, with data properties providing 'scalar' attributes such as names, measured values, etc. The user will then be able, in a SPARQL query, to assemble triple patterns featuring classes and properties into required complex objects.

We used the query catalogue and ontology vocabulary to identify relevant ProSource queries. These were carefully analysed and split into subqueries that match classes and properties of the ontology. The integrity constraints (both declared in the database and implied by the Epicentre model) were used to validate and simplify joins in the queries.

As a result, we obtained an R2RML specification with 62 logical tables and 180 mapping assertions (combinations of subject, predicate and object maps); the ontology-saturated mappings have 324 assertions. The logical tables vary from base tables to 6-way joins with up to 5 additional filter expression in the WHERE clause:

# tables/views	1	2	3	4	5	6
# mappings	32	8	13	2	2	5

# filters	0	1	2	3	4	5
# mappings	10	24	15	8	3	2

Also, two SQL queries have GROUP BY and 8 contain stored procedure calls.

For testing the OBDA specification outside Statoil, we identified a small fragment of the database schema that supports the mappings. It consists of tables, views and materialised views occurring in the mappings and/or relevant integrity constraints (with only necessary columns). The result consists of schemas `SLEGGE` and `SLEGGE_EPI` and contains 66 tables with 379 columns, 55 views, 5 materialised views and 4 stored procedures (functions). It also has 47 foreign keys, with only *three* referring to entity tables.

6 Conclusions and Future Work

The application of OBDA technologies at Statoil dramatically reduces the amount of time for information gathering by allowing the geologists to express their needs as ontology-mediated queries and efficiently execute them over the database [8]. This paper presents the *complete OBDA specification* of the Statoil use case and includes the geologists' queries, the Subsurface Exploration Ontology, the schema of the Slegge database, and the mappings between the ontology

and the database. We are planning to develop a synthetic data generator for the OBDA specification, where the main challenge will be the faithful modelling of implicit domain constraints.

Our work on the mappings revealed a lack of tools to support the following tasks (taking account of the database integrity constraints):

- checking whether a mapping assertion is implied by the ontology and other mapping assertions (e.g., as the property `:overlapsWellboreInterval` is symmetric, the mapping assertions obtained by swapping the object and subject are redundant);
- checking whether a mapping assertion for a property generates all the triples of the assertions for the subclasses of its domain/range; a negative answer (even though is not an error) may indicate incorrect modelling if similar SQL queries are used.

Routine tasks such as checking whether IRI templates of classes/properties match could also be automated: for example, a Protégé plugin could list all IRI templates for the currently selected class or property (with ontology inferences taken into account). Developing tool support for such reasoning tasks is an important direction of future work.

Acknowledgements. This work was supported by EU IP Optique FP7-318338 and UK EPSRC project iTract EP/M012670. We are grateful to Statoil for allowing publication of the resources, and to everyone's help, especially Toralv Nordtveit and Hallstein Lie.

References

1. Antonioli, N., Castanò, F., Coletta, S., Grossi, S., Lembo, D., Lenzerini, M., Poggi, A., Virardi, E., Castracane, P.: Ontology-based data management for the Italian public debt. In: Proceedings of FOIS 2014, FAIA, vol. 267, pp. 372–385. IOS Press (2014)
2. Bail, S., Alkiviadous, S., Parsia, B., Workman, D., van Harmelen, M., Goncalves, R.S., Garilao, C.: Fishmark: a linked data application benchmark. In: Proceedings of SSWS+HPCSW (2012)
3. Brandt, S., Güzel Kalaycı, E., Kontchakov, R., Ryzhikov, V., Xiao, G., Zakharyaschev, M.: Ontology-based data access with a Horn fragment of metric temporal logic. In: AAAI, pp. 1070–1076. AAAI Press (2017)
4. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semant. Web **2**(1), 43–53 (2011)
5. 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. Semant. Web **8**(3), 471–487 (2017)
6. Calvanese, D., Liuzzo, P., Mosca, A., Remesal, J., Rezk, M., Rull, G.: Ontology-based data integration in EPNet: production and distribution of food during the Roman Empire. Eng. Appl. Artif. Intell. **51**, 212–229 (2016)

7. Giese, M., Soylu, A., Vega-Gorgojo, G., Waaler, A., Haase, P., Jiménez-Ruiz, E., Lanti, D., Rezk, M., Xiao, G., Özçep, Ö.L., Rosati, R.: Optique: zooming in on big data. IEEE Comput. **48**(3), 60–67 (2015)
8. Kharlamov, E., et al.: Ontology based access to exploration data at Statoil. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 93–112. Springer, Cham (2015). doi:10.1007/978-3-319-25010-6_6
9. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semant. **10**, 133–173 (2008)
10. POSC Epicentre v3.0. http://w3.energistics.org/archive/Epicentre/Epicentre_v3.0
11. Rodríguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: *ontop* of databases. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 558–573. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41335-3_35
12. Sequeda, J.F., Arenas, M., Miranker, D.P.: OBDA: query rewriting or materialization? In practice, both! In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 535–551. Springer, Cham (2014). doi:10.1007/978-3-319-11964-9_34
13. Sequeda, J.F., Miranker, D.P.: A pay-as-you-go methodology for ontology-based data access. IEEE Internet Comput. **21**(2), 92–96 (2017)
14. Skjæveland, M.G., Lian, E.H., Horrocks, I.: Publishing the Norwegian Petroleum Directorate's FactPages as semantic web data. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8219, pp. 162–177. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41338-4_11
15. Soylu, A., Giese, M., Jiménez-Ruiz, E., Vega-Gorgojo, G., Horrocks, I.: Experiencing OptiqueVQS: a multi-paradigm and ontology-based visual query system for end users. Univ. Access Inf. Soc. **15**(1), 129–152 (2016)