Ontology-Based Data Access to Slegge
(Extended Version of the ISWC 2017 Paper)

D. Hovland[1], R. Kontchakov[2], M. Skjæveland[1], A. Waaler[1], and M. Zakharyaschev[2]

[1] Department of Informatics, University of Oslo
[2] Department of Computer Science and Information Systems, Birkbeck, University of London

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 [9]. In the OBDA paradigm based on query rewriting [13], the data remains stored in the original (relational) 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 would presuppose detailed knowledge of the database schema and require an assistance of an IT expert. The OBDA system makes use of the mappings that relate the ontology vocabulary to the database schema in order 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., [1,2,4,5,7,14,16]. However, full details of an industrial use cases 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 schema of the Slegge SQL 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).

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. As 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 are available at `http://purl.org/slegge` and in the open git repository `https://gitlab.com/slegger/slegge-obda` published with the Creative Commons Attribution 4.0 International Public License.

Related Work. 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 (see Section 3) and the large and complex industrial database with hundreds of tables (see Section 4). We designed the ontology capturing the vocabulary of the needs in the context of oil and gas exploration (see Section 3.1) and the complex mappings to bridge the substantial conceptual gap between the ontology and the data (see Section 5).

Other publicly available OBDA specifications for databases with real data include FishMark [2][3], IMDb OBDA [14][4] and the NPD FactPages [18][5]. The first two have ontologies that are quite similar to the database schemas, and so their mappings are almost direct (with very few joins). There are no real, natural language end-user queries; only those based on existing SQL queries and queries invented by the authors are provided. The NPD FactPages specification was developed from a set of web reports in tabular format, which are generated primarily for human consumption, and a collection of simple queries that are generic for the oil and gas domain. The schema of the database obtained from the tabular reports is not very different from the ontology, whose design was also mostly driven by the structure of the data [17], and so again the mappings are similar to direct ones. The specification comes with 40MB of real data, which can be scaled up by the data pumper [12]. The Texas benchmark [15][6] and the OBDA extensions of the Berlin SPARQL Benchmark (BSBM) [15,3][7] and LUBM [14,10] are all examples with synthetic data based on existing non-OBDA benchmarks (e.g., Wisconsin database benchmark for Texas). The ontologies in the first two are class taxonomies and have no object and datatype properties; the mappings are almost direct: each SQL query refers to a single table and has one filter in the WHERE clause. The mappings in LUBM are also fairly simple, but more importantly, its database schema was designed specifically for the benchmark and quite closely follows the ontology.

We begin a brief description of the Slegge resources by outlining the process of data gathering at Statoil and the role OBDA can play in it.

---

[3] URL: `http://owl.cs.manchester.ac.uk/publications/supporting-material/fishmark`

[4] URL: `https://github.com/ontop/ontop/wiki/Example_MovieOntology`

[5] URL: `https://gitlab.com/logid/npd-factpages`

[6] URL: `http://www.obda-benchmark.org/texas`

[7] URL: `http://www.obda-benchmark.org/bsbm-obda`

2 Data Gathering at Statoil

The task of the exploration department at Statoil is to find exploitable deposits of hydrocarbons (oil or gas). Geoscientists in the department model the subsurface geography by classifying rock layers according to multiple stratigraphic hierarchies using information from a wide range of different sources. This model, which is basically a 3D map, largely determines the location of new wellbores for direct exploration and assessment of promising areas and, if successful, for exploitation of the hydrocarbon resource. Since wellbore drilling is a major expense,[8] 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 chronostrat 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 queries covering parts of the information need and then integrate their results, often with the help of primitive data management tools such as desktop speadsheet applications. The 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 that are not directly comparable. An alternative solution would be to ask the IT department to construct a custom SQL query. However, employing IT personnel tor the taks generally takes from a few days to a couple of weeks[9] 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 the correct database query. (In fact, as a quality measure, an informal policy is that, for any database, only a certain group of people are permitted to write custom queries.)

The OBDA paradigm [13] 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:

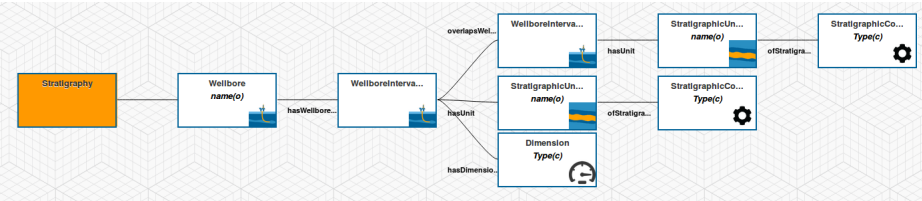
(001/02′) Give me the names of the available wellbores with the chronostratigraphic units and the top depths of the intervals they were found in; the depths should be in the standard units and from standard reference points (metres along the drill string). Also, return all lithostratigraphic units from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

[8] Personal communication cited 20–100 million euros, depending on the depth of sea, pressure, and time taken to drill.

[9] Personal communication.

query with, e.g., the visual query interface OptiqueVQS [19] of the Optique platform:



The formalised query (**001/02′**) is automatically translated into a SPARQL query:

```
SELECT ?wellbore ?chronostrat_unit ?top_md_m ?lithostrat_unit WHERE {
  ?w a :Wellbore; :name ?wellbore; :hasWellboreInterval ?intv.
  ?intv a :StratigraphicZone; :hasUnit ?cu; :hasTopDepth ?top.
  ?cu :name ?chronostrat_unit;
      :ofStratigraphicColumn [ a :ChronoStratigraphicColumn ].
  ?top a :MeasuredDepth ; :valueInStandardUnit ?top_md_m.
  ?intv :overlapsWellboreInterval ?litho_intv.
  ?litho_intv :hasUnit ?lu.
  ?lu :name ?lithostrat_unit;
      :ofStratigraphicColumn [ a :LithoStratigraphicColumn ].
}
```

Then, an OBDA tool such as Ontop [14,6] will use the *mappings* to 'rewrite' the *ontology-mediated query* into an SQL query over the database, optimise and execute it, returning the following results:

?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 below and appreciate the extreme differences in readability and in knowledge needed to write or edit this rewriting compared to the original SPARQL. Note also that the role of ontologies in OBDA is not only to provide a convenient vocabulary for the user queries but also to augment the (possibly incomplete) data with background knowledge.

```
SELECT
  QVIEW1."IDENTIFIER" AS "wellbore",
  QVIEW2."STRAT_UNIT_IDENTIFIER" AS "chronostrat_unit",
  QVIEW15."STRAT_UNIT_IDENTIFIER" AS "lithostrat_unit",
  QVIEW2."STRAT_ZONE_ENTRY_MD" AS "top_md_m"
 FROM
  "WELLBORE" QVIEW1,
  "STRATIGRAPHIC_ZONE" QVIEW2,
  "ROCK_FEATURE" QVIEW3,
  "COMPONENT_MATERIAL" QVIEW4,
```

```
  "DATA_COLLECTION" QVIEW5,
  "DATA_COLLECTION_CONTENT" QVIEW6,
  "ROCK_FEATURE" QVIEW7,
  "MATERIAL_CLASS" QVIEW8,
  "CLASSIFICATION_SYSTEM" QVIEW9,
  "DATA_COLLECTION_CONTENT" QVIEW10,
  "MATERIAL_CLASSIFICATION" QVIEW11,
  "STRATIGRAPHIC_ZONE" QVIEW15,
  "ROCK_FEATURE" QVIEW16,
  "COMPONENT_MATERIAL" QVIEW17,
  "DATA_COLLECTION" QVIEW18,
  "DATA_COLLECTION_CONTENT" QVIEW19,
  "ROCK_FEATURE" QVIEW20,
  "MATERIAL_CLASS" QVIEW21,
  "CLASSIFICATION_SYSTEM" QVIEW22,
  "DATA_COLLECTION_CONTENT" QVIEW23,
  "MATERIAL_CLASSIFICATION" QVIEW24
WHERE
  QVIEW1."REF_EXISTENCE_KIND" = 'actual' AND
  QVIEW1."IDENTIFIER" IS NOT NULL AND
  QVIEW1."IDENTIFIER" = QVIEW2."WELLBORE" AND
  QVIEW2."STRAT_ZONE_DEPTH_UOM" = 'm' AND
  QVIEW2."STRAT_COLUMN_IDENTIFIER" IS NOT NULL AND
  QVIEW2."STRAT_INTERP_VERSION" IS NOT NULL AND
  QVIEW2."STRAT_ZONE_IDENTIFIER" IS NOT NULL AND
  QVIEW2."STRAT_UNIT_IDENTIFIER" IS NOT NULL AND
  QVIEW2."STRAT_UNIT_IDENTIFIER" = QVIEW3."DESCRIPTION" AND
  QVIEW4."ENTITY_TYPE_NAME" = 'COMPONENT_MATERIAL' AND
  QVIEW3."ROCK_FEATURE_S" = QVIEW4."INCORPORATE_S" AND
  QVIEW3."ROCK_FEATURE_S" = QVIEW6."COLLECTION_PART_S" AND
  QVIEW5."DATA_COLLECTION_S" = QVIEW6."PART_OF_S" AND
  QVIEW2."STRAT_COLUMN_IDENTIFIER" = QVIEW5."NAME" AND
  QVIEW5."REF_DATA_COLLECTION_TYPE" = 'stratigraphic hierarchy' AND
  QVIEW9."KIND" = 'chronostratigraphy' AND
  QVIEW8."CLASSIFICATION_SYSTEM" = QVIEW9."NAME" AND
  QVIEW7."ROCK_FEATURE_S" = QVIEW10."COLLECTION_PART_S" AND
  QVIEW5."DATA_COLLECTION_S" = QVIEW10."PART_OF_S" AND
  QVIEW7."ROCK_FEATURE_S" = QVIEW11."MATERIAL_S" AND
  QVIEW8."MATERIAL_CLASS_S" = QVIEW11."MATERIAL_CLASS_S" AND
  QVIEW2."STRAT_ZONE_ENTRY_MD" IS NOT NULL AND
  QVIEW1."IDENTIFIER" = QVIEW15."WELLBORE" AND
  QVIEW15."STRAT_ZONE_DEPTH_UOM" = 'm' AND
  (((QVIEW15."STRAT_ZONE_EXIT_MD" >= QVIEW2."STRAT_ZONE_ENTRY_MD") AND (QVIEW15."
       STRAT_ZONE_ENTRY_MD" <= QVIEW2."STRAT_ZONE_EXIT_MD")) OR
      ((QVIEW2."STRAT_ZONE_EXIT_MD" >= QVIEW15."STRAT_ZONE_ENTRY_MD") AND (QVIEW2.
           "STRAT_ZONE_ENTRY_MD" <= QVIEW15."STRAT_ZONE_EXIT_MD"))) AND
  QVIEW15."STRAT_COLUMN_IDENTIFIER" IS NOT NULL AND
  QVIEW15."STRAT_INTERP_VERSION" IS NOT NULL AND
  QVIEW15."STRAT_ZONE_IDENTIFIER" IS NOT NULL AND
  QVIEW15."STRAT_UNIT_IDENTIFIER" IS NOT NULL AND
```

```
QVIEW15."STRAT_UNIT_IDENTIFIER" = QVIEW16."DESCRIPTION" AND
QVIEW17."ENTITY_TYPE_NAME" = 'COMPONENT_MATERIAL' AND
QVIEW16."ROCK_FEATURE_S" = QVIEW17."INCORPORATE_S" AND
QVIEW15."STRAT_COLUMN_IDENTIFIER" = QVIEW18."NAME" AND
QVIEW18."REF_DATA_COLLECTION_TYPE" = 'stratigraphic hierarchy' AND
QVIEW16."ROCK_FEATURE_S" = QVIEW19."COLLECTION_PART_S" AND
QVIEW18."DATA_COLLECTION_S" = QVIEW19."PART_OF_S" AND
QVIEW22."KIND" = 'lithostratigraphy' AND
QVIEW21."CLASSIFICATION_SYSTEM" = QVIEW22."NAME" AND
QVIEW20."ROCK_FEATURE_S" = QVIEW23."COLLECTION_PART_S" AND
QVIEW18."DATA_COLLECTION_S" = QVIEW23."PART_OF_S" AND
QVIEW20."ROCK_FEATURE_S" = QVIEW24."MATERIAL_S" AND
QVIEW21."MATERIAL_CLASS_S" = QVIEW24."MATERIAL_CLASS_S"
```

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.

3.1 Subsurface Exploration Ontology

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 9999ft. 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[10].

[10] URL: `http://www.opengeospatial.org/standards/geosparql`

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-$\mathcal{ALCHIQ}(\mathcal{D})$. Full GeoSPARQL is a $\mathcal{SHIF}(\mathcal{D})$ ontology; however, we only require a tiny part of it, which resides within OWL 2 QL.

3.2 SPARQL Queries

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 Section 2, we omitted the coordinates for simplicity).

4 Slegge Database

Slegge is an Oracle database with about 700 GB of data. The database 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 is currently maintained by Energistics.[11] The most recent

[11] URL: `http://www.energistics.org/`

specification is Epicentre v3.0.[12] It describes the logical database model and a standard way of projecting it to the physical model that can be implemented as an Oracle database.

The main features of the Epicentre object-oriented logical model and its relational 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.

We now illustrate these design principles and their ramifications for OBDA.

Entities and Inheritance. Epicentre extensively uses inheritance hierarchies. For example, `well` and `wellbore` are subtypes of `facility`; and `other_facility` along with a hundred of other entities are subtypes of `general_facility`, which in turn is a subtype of `facility`. Entities `facility` and `general_facility` are abstract and have no separate database tables, but each of the non-abstract entities such as `well` and `other_facility` is represented by a table in the database: `WELL` and `OTHER_FACILITY`. These tables have columns for the normal attributes of the entity: for example, the `wellbore` table has a `COMPLETION_DATE` column for the date when the wellbore was available for service. Tables for subtypes 'inherit' columns for the attributes of the supertypes: for example, tables for the various sorts of facilities contain `R_EXISTENCE_KD_NM` to specify whether the facility is actual, planned, etc. Also, `facility` is a subtype of the top-level entity `e_and_p_data`, and so all of these tables have a column `ROW_CREATOR` to record the person, company or application that created this instance.

Multiple inheritance is common in Epicentre: `wellbore` is a subtype of `facility`, and `field` is a subtype of `pfnu` (product flow network unit), but `well` is a subtype of both `facility` and `pfnu` (which are incomparable). In such cases, the subtype table inherits columns from all of its supertypes.

Instances of entities (objects) are identified by their (permanent) unique IDs, which are surrogate primary keys in the tables: e.g., columns `WELLBORE_S` and `WELL_S` for wellbores and wells, respectively. On the other hand, the (user-friendly) well and wellbore identifiers (attribute `identifier` of the supertype entity `facility`) are represented in columns `WELL_ID` and `WELLBORE_ID`, respectively; the tables for the subtypes of `general_facility` inherit column `GENERAL_FCL_NAME` for the general facility identifier.

The projection of the Epicentre logical model (and Slegge, in particular) also implements an alternative approach to dealing with hierarchies: all subtypes of an entity can be mapped to the same database table, in which case a *discriminating column* contains the name of the subtype of the object. For example, the four subtypes of entity `stratigraphic_marker` (including itself) are represented by the same table, `STRAT_MRK`, where column `ENTITY_TYPE_NM` distinguishes between the subtypes.

[12] URL: `http://w3.energistics.org/archive/Epicentre/Epicentre_v3.0/`

Properties and Denormalisation. To model composite attributes (such as 2D coordinates or quantities with units of measure), Epicentre uses so-called *properties*, which, like entities, have instances (auxiliary objects) and are arranged in an extensive hierarchy. Properties are normally mapped to database tables. However, one of the important features of the Slegge implementation is the use of *denormalised attributes*: for example, table `WELL_SURFACE_PT` for the locations of wells (`well_surface_point` entity) contains columns `WATER_DEPTH` and `WATER_DEPTH_U` to store the value and the unit of measure of the water depth property, respectively. So, values are stored directly in the 'entity' table rather than in a special table `P_WATER_DEPTH` for property `pty_water_depth` (which is present in the database but is *empty*). The same applies to the azimuth and inclination of interesting points along a wellbore: both compound attributes are modelled as pairs of columns in the table `WELLBORE_POINT` for the entity `wellbore_point` (rather than in separate tables `P_AZIMUTH` and `P_INCLINATION`). In fact, only 20 out of the 543 property tables are non-empty in Slegge. For example, coordinates of wells' surface points are stored in a dedicated table `P_LOCATION_2D` that represents the property `pty_location_2d` (and not in `WELL_SURFACE_PT`).

Reference Entities. The Epicentre data model also features *reference entities* such as `ref_unit_of_measure`: this entity is projected to the table `R_UOM` that contains acronyms (e.g., `'m'`), identifiers (e.g., `metre`), descriptions (e.g., `'The metre is the length...'`) and other information about units of measure. The primary key in such a table is then referenced by numerous foreign keys of property and entity tables: e.g., `WATER_DEPTH_U` in `WELL_SURFACE_PT` is among 814 columns referencing `ACRONYM` in `R_UOM`.

One-to-Many Relationships. In the Epicentre data model, relationships are represented as attributes of entities (with collections of instances as possible values), so entities at both end-points of a relationship have a respective attribute: one for the relationship, the other for its inverse. One-to-many relationships are normally projected as columns in the tables. For example, the column `WELL_S` in the table `WELLBORE` specifies the identifier of the well containing this wellbore (recall that `WELL_S` is the surrogate primary key in `WELL`). In this particular case, the database also 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 various tables. For example, the relationship between `activity` and `facility` is represented by the column `FACILITY_S` in `ACTIVITY`. But since various facilities are covered in a number of tables, there cannot be a simple foreign key. In fact, `ACTIVITY` contains another column, `FACILITY_T`, that provides the specific type of the facility it refers to (e.g., `'WELLBORE'` or `'WELL'`), and so `FACILITY_T` and `FACILITY_S` together identify the referenced table and the row in it, respectively.

Another complication in the Slegge implementation is denormalised attributes for reducing the number of joins in queries: some tables contain copies of columns from the tables they refer to. For example, table `WELLBORE`, along with the reference to the primary key `WELL_S` of `WELL`, also contains a column duplicating `WELL_ID` from `WELL`.

Many-to-Many Relationships are modelled in Epicentre as association entities, which are then projected to separate tables in the database. For example, the topological relationships of the 9-intersection model [8] between instances of `topological_object`

(including its subtypes: field, core and facility, and so both wellbore and well) are modelled as instances of the topological_relationship association entity, which is represented as the table TOPOLOGICAL_REL. The table contains two pairs of columns, PRIM_TOPLG_OBJ_S/_T and SEC_TOPLG_OBJ_S/_T, for the two arguments of a topological relation such as 'inside' (reference entity ref_object_intersection provides a list of possible topological relations). As above, the _T column identifies the table and the _S column the row in that table; but again, there are no foreign keys in the database.

The Slegge Oracle database has 6 schemas: SLEGGE_EPI, SLEGGE_SNP, SIS_CATALOG, MDS_COORD, ENTITLED and SLEGGE. The SLEGGE_EPI schema is a dated implementation of the POSC Epicentre data model and consists of 1545 tables with 19 719 columns. Note that 1141 of these tables are empty because large portions of the POSC Epicentre data model are not used by the tools. There are 221 tables that contain 1–100 rows (these mostly represent reference entities; even though R_UOM, for example, contains 974 rows). On the other hand, there are 9 tables with more than a million rows each. The SLEGGE_SNP, MDS_COORD and SIS_CATALOG schemas are much smaller (21, 14 and 1 table, respectively) and related to other applications. The schema 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—these are simply used to rename tables and columns to make Slegge more compliant with Epicentre 3.0. Two views additionally limit access in accordance with the user privileges stored in ENTITLED. The remaining 88 views vary in complexity from two-table joins to 31-way joins with additional WHERE and GROUP BY clauses. Many of the views also contain ORDER BY clauses, which suggests their primary use for reporting and user interfaces. In addition, the SLEGGE schema contains 102 tables for various purposes. The most interesting ingredient, however, is the five *materialised views*, two of which are joins of 12 and 15 tables, respectively, while the other three use calls to complex 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 the database objects, which required detailed knowledge of both components. Unfortunately, the Slegge implementation does not fully comply with any particular version of Epicentre, and the important documentation on Slegge has become either unavailable or hard to obtain at Statoil. The lack of integrity constraints (foreign keys)[13] 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 database schema was the 2996 queries found in the configuration files of ProSource, a proprietary tool developed in-house and

[13] 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.

the *de facto* way of accessing the Slegge database. It is worth mentioning that even the view definitions of the SLEGGE schema often contain ProSource-generated fragments (hand-crafting SQL queries is quite complex, error-prone, and generally avoided).

It turned out, however, that a significant number of the ProSource queries, and in particular, the most useful ones for mapping ontology classes and properties were quite large (up to 91 joins). The following table illustrates the distribution of the number of tables and views per query:

# 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 interesting 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 connected by object properties, with data properties providing additional information such as names, measured values, etc. The user will then be able, in a SPARQL query, to assemble individual triple patterns featuring classes and properties into the required complex object.

We used the query catalogue and the vocabulary of the ontology to identify relevant ProSource queries. These were then carefully analysed and split into subqueries that match classes and properties of the ontology. In this process, we also extracted the fragment of the database schema relevant to the obtained mappings. The integrity constraints (both explicitly defined in the database and those that follow from the Epicentre specifications) were used to validate and simplify joins in the resulting 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 contain 324 mapping assertions. The logical tables vary from simple tables to 6-way joins with up to 5 additional filter expression in the WHERE clause. More detailed statistics is collected in the two tables below, where – in the number of filters means R2RML base table and 0 means an R2RML view with a single table and no WHERE clause but with a function call in the SELECT clause:

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

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

It is also worth noting that two SQL queries in the mappings also have GROUP BY and 8 contain calls to a stored procedure.

For the purposes of testing the OBDA setup outside Statoil, we also identified a small fragment of the database schema that supports the mappings. This fragment includes tables, views and materialised views that occur in the mappings and/or the relevant integrity constraints. We also reduced the number of columns and left only those that are required to execute the SQL queries of the mappings. The identified fragment of the database schema consists of two schemas, SLEGGE and SLEGGE_EPI, which contain 66 tables with 379 columns, 55 views, 5 materialised views and 4 stored procedures (functions). This schemas contain only 47 foreign keys, 11 of which refer to R_UOM and only 3 foreign keys refer 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 [11]. 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 the EU IP project Optique FP7-318338 and the UK EPSRC project iTract EP/M012670. We are grateful to Statoil[14] for allowing publication of these resources, and to everyone who helped us, 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: Proc. of the 8th Int. Conf. on Formal Ontology in Information Systems, FOIS 2014. Frontiers in Artificial Intelligence and Applications, vol. 267, pp. 372–385. IOS Press (2014), http://dx.doi.org/10.3233/978-1-61499-438-1-372
2. 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 SSWS+HPCSW 2012. CEUR Workshop Proceedings, vol. 943, pp. 1–15. CEUR-WS.org (2012), http://ceur-ws.org/Vol-943/SSWS_HPCSW2012_paper1.pdf
3. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. International Journal on Semantic Web and Information Systems (IJSWIS) 5(2), 1–24 (2009), http://dx.doi.org/10.4018/jswis.2009040101

[14] URL: https://www.statoil.com

4. 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: Proc. of the 31st AAAI Conf. on Artificial Intelligence, AAAI 2017, pp. 1070–1076. AAAI Press (2017), http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14881
5. 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. Semantic Web 2(1), 43–53 (2011), https://doi.org/10.3233/SW-2011-0029
6. 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 8(3), 471–487 (2017), http://dx.doi.org/10.3233/SW-160217
7. 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. Engineering Applications of Artificial Intelligence 51, 212–229 (2016), http://dx.doi.org/10.1016/j.engappai.2016.01.005
8. Clementini, E., Di Felice, P., van Oosterom, P.: A small set of formal topological relationships suitable for end-user interaction. In: Proc. of the 3rd Int. Symposium on Advances in Spatial Databases, SSD'93. Lecture Notes in Computer Science, vol. 692, pp. 277–295. Springer (1993), http://dx.doi.org/10.1007/3-540-56869-7_16
9. 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 Computer 48(3), 60–67 (2015), http://dx.doi.org/10.1109/MC.2015.82
10. Guo, Y., Pan, Z., Heflin, J.: Lubm: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web 3(2-3), 158–182 (2005), http://dx.doi.org/10.1016/j.websem.2005.06.005
11. Kharlamov, E., Hovland, D., Jiménez-Ruiz, E., Lanti, D., Lie, H., Pinkel, C., Rezk, M., Skjæveland, M.G., Thorstensen, E., Xiao, G., Zheleznyakov, D., Horrocks, I.: Ontology based access to exploration data at Statoil. In: Proc. of the 14th Int. Semantic Web Conf. ISWC 2015, Part II. Lecture Notes in Computer Science, vol. 9367, pp. 93–112. Springer (2015), http://dx.doi.org/10.1007/978-3-319-25010-6_6
12. Lanti, D., Rezk, M., Slusnys, M., Xiao, G., Calvanese, D.: The NPD benchmark for OBDA systems. In: Proc. of the 10th Int. Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 13th Int. Semantic Web Conf. (ISWC 2014). pp. 3–18 (2014), http://ceur-ws.org/Vol-1261/SSWS2014_paper1.pdf
13. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. Journal on Data Semantics X, 133–173 (2008), https://doi.org/10.1007/978-3-540-77688-8_5
14. Rodríguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: Ontop of databases. In: Proc. of the 12th Int. Semantic Web Conf., ISWC 2013, Part I. Lecture Notes in Computer Science, vol. 8218, pp. 558–573. Springer (2013), http://dx.doi.org/10.1007/978-3-642-41335-3_35
15. Sequeda, J.F., Arenas, M., Miranker, D.P.: OBDA: query rewriting or materialization? In practice, both! In: In Proc. of the 13th Int. Semantic Web Conf., ISWC 2014. Lecture Notes in Computer Science, vol. 8796, pp. 535–551. Springer (2014), https://doi.org/10.1007/978-3-319-11964-9_34
16. Sequeda, J.F., Miranker, D.P.: A pay-as-you-go methodology for ontology-based data access. IEEE Internet Computing 21(2), 92–96 (2017), http://dx.doi.org/10.1109/MIC.2017.46
17. Skjæveland, M.G., Giese, M., Hovland, D., Lian, E.H., Waaler, A.: Engineering ontology-based access to real-world data sources. Web Semantics: Science, Services and Agents on the World Wide Web 33, 112–140 (2015), http://dx.doi.org/10.1016/j.websem.2015.03.002

18. Skjæveland, M.G., Lian, E.H., Horrocks, I.: Publishing the Norwegian Petroleum Directorate's FactPages as semantic web data. In: Proc. of the 12th Int. Semantic Web Conf., ISWC 2013, Part II. Lecture Notes in Computer Science, vol. 8219, pp. 162–177. Springer (2013), http://dx.doi.org/10.1007/978-3-642-41338-4_11
19. 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. Universal Access in the Information Society 15(1), 129–152 (2016), http://dx.doi.org/10.1007/s10209-015-0404-5