Efficient Ontology-Based Data Integration with Canonical IRIs

Guohui Xiao[1], Dag Hovland[2], Dimitris Bilidas[3], Martin Rezk[4], Martin Giese[2], and Diego Calvanese[1]

[1] Faculty of Computer Science, Free-University of Bozen-Bolzano, Italy,
{xiao,calvanese}@inf.unibz.it
[2] Department of Informatics, University of Oslo, Norway,
{hovland,martingi}@ifi.uio.no
[3] National and Kapodistrian University of Athens, Greece,
d.bilidas@di.uoa.gr
[4] Rakuten, Tokyo, Japan
martin.rezk@rakuten.com

Abstract. In this paper, we study how to efficiently integrate multiple relational databases using an ontology-based approach. In ontology-based data integration (OBDI) an ontology provides a coherent view of multiple databases, and SPARQL queries over the ontology are rewritten into (federated) SQL queries over the underlying databases. Specifically, we address the scenario where records with different identifiers in different databases can represent the same entity. The standard approach in this case is to use `sameAs` to model the equivalence between entities. However, the standard semantics of `sameAs` may cause an exponential blow up of query results since all possible combinations of equivalent identifiers have to be included in the answers. The large number of answers is not only detrimental to the performance of query evaluation, but also makes the answers difficult to understand due to the redundancy they introduce. This motivates us to propose an alternative approach, which is based on assigning canonical IRIs to entities in order to avoid redundancy. Formally, we present our approach as a new SPARQL entailment regime and compare it with the `sameAs` approach. We provide a prototype implementation and evaluate it in two experiments: in a real-world data integration scenario in Statoil and in an experiment extending the Wisconsin benchmark. The experimental results show that the canonical IRI approach is significantly more scalable.

1 Introduction

Large organizations, both public and private, typically need to manage, in multiple information systems, large amounts of data stored across multiple heterogeneous data sources. To support decision making in such settings, there is the need to access in an integrated way the data managed by the different systems, which in general are stored in different databases. A key challenge for data integration in such settings is that the different systems have often been designed separately serving different purposes, and thus lack a coherent view of the underlying data.

Ontology-based data access (OBDA) [25] is a successful paradigm that addresses this challenge by relying on semantic technologies to provide a uniform conceptual view over heterogeneous data. Specifically, an ontology describing the domain of interest is connected to a data source through a declarative specification, given in terms of *mappings* [9] that relate symbols (i.e., classes and properties) in the ontology to (SQL) views over the data. The ontology layer can be queried using SPARQL, and queries are automatically rewritten by an OBDA engine into SQL queries expressed over the underlying database. Thus, users no longer need an understanding of the structure and organization of the data in the source. *Ontology-based data integration* (OBDI) is an extension of OBDA in which the data is not stored in a single database, but in a multitude of databases that need to be queried in an integrated way, while still maintaining the same conceptual architecture based on mappings [4]. OBDI has been successfully deployed in several domains, such as oil and gas [19], maritime security [2], and cultural heritage [3].

An important aspect in OBDI is the fact that the same conceptual entity may be stored in different databases but represented by different identifiers. Consider e.g., the same person, represented in one database through name, surname, and date of birth, in a second database through the social security number, and in a third database through the taxcode. In order to combine the information coming from different systems so as to produce coherent results to queries, we need a way to relate different representations so that they can be recognized as representing the same entity. This represents a challenge both from the theoretical and from the practical point of view.

The standard approach in the Semantic Web of modeling the fact that different IRIs actually represent the same entity is to use `owl:sameAs`[5], which defines an equivalence relation between entities. In our previous work [6], we have shown how an approach based on the use of `sameAs` can be adopted also for OBDI. A serious drawback of this approach is that the semantics of `sameAs` may cause an exponential blowup of query results, since all possible combinations of equivalent identifiers have to be included in the answer to a query. The large number of answers is detrimental to the performance of query evaluation, and also introduces redundancy making answers difficult to understand.

To address this problem, we propose here an alternative approach, based on assigning to each entity a *canonical IRI*. Such canonical IRI is the one of choice for relating occurrences of the entity that are identified differently in different data sources, and also for returning the entity to the user. The idea of exploiting a canonical representation of IRIs internally, for optimization purposes, has already been used in some reasoning engines. E.g., RDFox [23] uses an optimized approach for rule evaluation by forward chaining based on propagating a canonical representation for entities. Also Stardog adopts a similar strategy in query answering, by only returning a canonical representation of equivalent entities, to avoid the combinatorial explosion in query results[6].

However, lifting this idea to the OBDI scenario is non-trivial. On the one hand, the choice on which source should provide the canonical IRI for an entity depends on the actual (type of) entity, and on the data sources in which it appears. A more critical issue is due to the fact that in OBDI, entities with the corresponding IRIs are not materialized but kept *virtual*, since they are generated dynamically at query answering time from the

[5] For space reasons, from now on we will use `sameAs` as an abbreviation for `owl:sameAs`.
[6] `https://www.stardog.com/docs/#_same_as_reasoning`

data in the sources through the mappings. Hence it is not really feasible to explicitly substitute the entities by canonical ones. Instead our proposal is based on embedding such information in the mapping specification, while delegating to the designer of the OBDI system the responsibility of choosing which canonical IRIs to adopt.

For such a setting, we provide the following contributions:

- We formally define the *canonical IRI semantics* as a SPARQL entailment regime. We show that such semantics is equivalent to the `sameAs` semantics (taking into account equivalence of IRIs), thus guaranteeing soundness and completeness of the associated query answering algorithm. (See Section 3.)
- We show that query answering and the canonical IRI semantics can be reduced to a simple entailment regime by means of query rewriting. However, this query rewriting step is only of theoretical interest since the rewritten query is not efficiently executable. (See Section 4.)
- We propose and alternative technique for query answering, inspired by algorithms for mapping saturation [3], that is based on *mapping rewriting*. (See Section 5.)
- We provide a prototype implementation of our technique and evaluate it in two experiments. The first experiment is based on a real-world data integration scenario in the oil company Statoil [19]. The second experiment is based on the Wisconsin benchmark [10]. The experimental results show that the canonical IRI approach is significantly more scalable. (See Section 6.)

We start our technical development by introducing in Section 2 the necessary background. Due to space limits, omitted proofs are provided in an online appendix[7].

2 Preliminaries

We present now the preliminary notions on RDF graphs, SPARQL, and ontology-based data access on which we build in the rest of the paper.

2.1 RDF and SPARQL

SPARQL [15] is the W3C standard language designed to query RDF graphs. We consider a vocabulary of three pairwise disjoint and countably infinite sets of symbols: $\mathbf{I}$ for *IRIs*, $\mathbf{L}$ for *RDF literals*, and $\mathbf{V}$ for *variables*. In line with previous work on ontology-based data access, we do not consider blank nodes. The elements of $\mathbf{T} = \mathbf{I} \cup \mathbf{L}$ are called *RDF terms*, and those of $\mathbf{T} \times \mathbf{I} \times \mathbf{T}$ are called *(RDF) triples*. An *(RDF) graph* is a set of triples. A *triple pattern* is an element of $(\mathbf{T} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{V}) \times (\mathbf{T} \cup \mathbf{V})$. A *basic graph pattern* (*BGP*) is a finite set of triple patterns. We consider the fragment of SPARQL queries defined by Q in the following EBNF grammar[8]:

$$\begin{array}{rcl} P & ::= & B \mid Q \mid P\, \texttt{FILTER}\, F \mid P\, \texttt{UNION}\, P \mid (P, P) \mid P\, \texttt{OPT}\, P \\ Q & ::= & \texttt{SELECT}\, \{\, \mathbf{V}\, \texttt{AS}\, \mathbf{V}\, \}\, \texttt{WHERE}\, P \end{array}$$

where B is a BGP and F is a *filter expression* (we refer to [15] for details). An

[7] https://www.dropbox.com/s/h9xsagzhh7mxpy3/eswc-18-canonical-iri-extended.pdf

[8] Recall that in EBNF "{ A }" means any number of repetitions of A.

expression $\{?y_1 \texttt{ AS } ?x_1, \ldots, ?y_n \texttt{ AS } ?x_n\}$ is called a *projection* with *answers variables* $\{?x_1, \ldots, ?x_n\}$. We abbreviate "$?x \texttt{ AS } ?x$" with $?x$. For a SPARQL query $Q = \texttt{SELECT } R \texttt{ WHERE } Q_1$, we use $sig(Q)$ to denote the answer variables of R.

The semantics of SPARQL queries is given in terms of *solution mappings*, which are *partial* maps $s\colon \mathbf{V} \to \mathbf{T}$ with (possibly empty) domain $dom(s)$. Here, following [24,21,27], we use the set-based semantics for SPARQL (rather than the bag-based one, as in the W3C specification). More specifically, for a BGP B, the *answer* $[\![B]\!]_G$ to B over a graph G is $[\![B]\!]_G = \{s\colon var(B) \to \mathbf{T} \mid s(B) \subseteq G\}$, where $var(B)$ is the set of variables occurring in B and $s(B)$ is the result of substituting each variable u in B by $s(u)$. Then, the *answer* to a SPARQL query Q over a graph G is the set $[\![Q]\!]_G$ of solution mappings defined by induction using the SPARQL algebra operators (filter, join, union, optional, and projection) starting from BGPs; cf. [18]. This semantics is known as *simple entailment*.

2.2 SPARQL Entailment Regimes

SPARQL entailment regimes allow for querying RDF graphs with richer reasoning capabilities [13]. Specifically, an entailment regime E specifies how to obtain from an RDF graph G an entailed graph $eg^E(G)$. Then, the answer $[\![B]\!]_G^E$ to a BGP B under the entailment regime E is defined as $[\![B]\!]_{eg^E(G)}$. Similarly, The answer $[\![Q]\!]_G^E$ to a SPARQL query Q under the entailment regime E is defined as $[\![Q]\!]_{eg^E(G)}$. Note that entailment regimes only modify the evaluation of BGPs but not that of other SPARQL operators.

We present now the standard W3C semantics for SPARQL queries over OWL 2 ontologies under different entailment regimes. Under the *OWL 2 direct semantics entailment regime*, one can query an RDF graph G that consist of two parts: the *intensional* sub-graph (i.e., TBox or ontology) $\mathcal{T}$ representing the background knowledge in terms of class and property axioms, and an *extensional* sub-graph (i.e., ABox) $\mathcal{A}$ representing the data as class and property assertions. We write such a graph G, which represents a *knowledge base*, as $(\mathcal{T}, \mathcal{A})$ to emphasize the partitioning when necessary. Moreover, for convenience, we use the triple notations $(s, \texttt{rdf:type}, C)$ and (s, p, o) and the ABox assertion notations $C(s)$ and $p(s, o)$ interchangeably.

We work with ontologies expressed in the OWL 2 QL profile [22] of OWL 2. Such profile induces the *OWL 2 QL* (or simply, *QL*) *entailment regime* in which, for an OWL 2 QL knowledge base G we have $eg^{QL}(G) = \{t \mid G \models_{DL} t\}$, where $\models_{DL}$ denotes the standard OWL 2 entailment, defined in terms of description logics semantics, cf. [1]. Under the QL entailment regime, a SPARQL query Q formulated over an ontology $\mathcal{T}$ is *first-order rewritable*, i.e., Q can be rewritten into a query $Q_\mathcal{T}$ such that for every ABox $\mathcal{A}$, evaluating Q over $(\mathcal{T}, \mathcal{A})$ (under the QL entailment regime) is equivalent to evaluating $Q_\mathcal{T}$ over $\mathcal{A}$ (under the simple entailment regime) [5,21].

2.3 SPARQL Entailment Regimes for `sameAs`

In addition to the input graph $G = (\mathcal{T}, \mathcal{A})$, we consider now a set $\mathcal{A}^s$ of `sameAs` triples of the form $\texttt{sameAs}(a, b)$, specifying the equivalence between individuals a and b.

We now define an entailment regime that interprets `sameAs` as the equivalence closure (i.e., the reflexive, transitive, and symmetric closure) of $\mathcal{A}^s$. For a set $\mathcal{A}^s$ of

$\texttt{sameAs}$ assertions, we denote by $(\mathcal{A}^s)^*$ the equivalence closure of $\mathcal{A}^s$. Given an RDF graph $\mathcal{A}$ and a set $\mathcal{A}^s$ of $\texttt{sameAs}$ assertions, we define the *sameAs entailed graph* $sag(\mathcal{A}, \mathcal{A}^s)$ of $\mathcal{A}$ with respect to $\mathcal{A}^s$ as

$$\mathcal{A} \cup \{C(o') \mid C(o) \in G, \texttt{sameAs}(o, o') \in (\mathcal{A}^s)^*\}$$
$$\cup \{R(o'_1, o'_2) \mid R(o_1, o_2) \in G, \texttt{sameAs}(o_1, o'_1) \in (\mathcal{A}^s)^*, \texttt{sameAs}(o_2, o'_2) \in (\mathcal{A}^s)^*\}.$$

Given an entailment regime E, we can derive an extended entailment regime E+$sameAs$, which takes into account equivalences between entities inferred through $\texttt{sameAs}$ statements, as follows:

$$eg^{E\text{+}sameAs}(\mathcal{T}, \mathcal{A} \cup \mathcal{A}^s) \;=\; eg^{E}(\mathcal{T}, sag(\mathcal{A}, \mathcal{A}^s))$$

In this paper, we are mostly interested in the SPARQL entailment regimes QL and QL+$\texttt{sameAs}$, but our results also apply to other entailment regimes E in which $\texttt{sameAs}$ is treated as a standard object property, and new $\texttt{sameAs}$ triples *cannot* be inferred.

2.4 Ontology-Based Data Access and Integration

In ontology-based data access (OBDA), we start from an *OBDA specification* $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \mathcal{S})$, consisting of a set $\mathcal{T}$ of OWL 2 axioms, a set $\mathcal{M}$ of *mapping assertions*, and a relational database schema $\mathcal{S}$. An *OBDA instance* $(\mathcal{P}, D)$ is given by an OBDA specification $\mathcal{P}$ and a relational database instance D compliant with $\mathcal{S}$.

Mapping assertions allow one to define how an ABox should be populated with values retrieved by means of SQL queries. Each mapping assertion has one of the forms

$$C(f(\boldsymbol{x})) \leftarrow \texttt{sql}(\boldsymbol{y}) \qquad\qquad P(f(\boldsymbol{x}), f'(\boldsymbol{x'})) \leftarrow \texttt{sql}(\boldsymbol{y})$$

such that $\boldsymbol{x} \subseteq \boldsymbol{y}$ and $\boldsymbol{x'} \subseteq \boldsymbol{y}$, $\texttt{sql}$ is an arbitrary SQL query over $\mathcal{S}$ projecting columns $\boldsymbol{y}$, and f and f' are functions constructing RDF terms out of values retrieved from the database. In a concrete mapping language like R2RML [9], these functions are specified as templates for IRIs and literals. For example, `<http://statoil.com/wellbore/{id}>` is an IRI template where "`{id}`" is a placeholder, and it generates, e.g., the IRI `<http://statoil.com/wellbore/25>` when `{id}` is instantiated with `"25"`. In practice, it is desired that each IRI can be constructed by at most one IRI template, and we make such assumption here; formally, the union of all the IRI templates occurring in an OBDA specification is an injective function. By applying all mapping assertions in $\mathcal{M}$ to D, one can derive a (*virtual*) RDF graph $\mathcal{A}_{\mathcal{M},D}$ [25]. Then, SPARQL query answering over an OBDA instance $(\mathcal{P}, D)$ is defined as query answering over $(\mathcal{T}, \mathcal{A}_{\mathcal{M},D})$.

We recall that, ontology-based data integration (OBDI) considers a set of data sources instead of a single one, but otherwise the formal treatment is identical to that of OBDA.

The inspiration for working with equality in OBDI came from problems we encountered in the context of the EU-funded project Optique [12], when querying data in the Norwegian oil company Statoil. This is also the background of the *real world* experiments in Section 6.1. In the following, we present our running example, which provides a simplified version of the experimental setting of Optique. The content of the tables below is real data from the Norwegian Petroleum Directorate FactPages[9].

[9] `http://factpages.npd.no`

Example 1. Assume two databases `national` and `corporate` with one table each:

national.wellbore

name	opPurp	wlbFld
1/3-1	WILDCAT	
2/4-2	WILDCAT	EKOFISK
1/3-10	APPRAISAL	OSELVAR
1/2-1	WILDCAT	BLANE

corporate.drillingops

name	driStDt	reason
NO_1/3-1	06-07-1968	WILDCAT
NO_2/4-2	18-09-1969	
NO_1/2-1	20-03-1989	WILDCAT
NO_1/3-A-1_H	22-07-2011	PRODUCTION

In both tables, the column `name` is the only key (i.e., names are unique within each table). These columns have almost the same values in the two databases, except that in the `corporate` database they are prefixed with "`NO_`" (country code for Norway). An example set of mapping assertions is:

```
:NationalWellbore/{name} :inField {wlbFld} ; :purpose {opPurp} .
← SELECT name, wlbFld, opPurp FROM national.wellbore

:CorporateWellbore/{name} :drillingStarted {driStDt} ; :purpose {reason} .
← SELECT name, driStDt, reason FROM corporate.drillingops
```

Examples of triples in the ABox defined by this OBDA scenario are[10]

$$\begin{array}{lll}(\texttt{:NationalWellbore/1/2-1}, & \texttt{:inField}, & \texttt{BLANE}),\\ (\texttt{:CorporateWellbore/NO\_1/3-A-1\_H}, & \texttt{:drillingStarted}, & \texttt{22-07-2011}),\\ (\texttt{:NationalWellbore/1/3-10}, & \texttt{:purpose}, & \texttt{APPRAISAL}),\\ (\texttt{:CorporateWellbore/NO\_1/2-1}, & \texttt{:purpose}, & \texttt{WILDCAT}).\end{array}$$

Compare the property :`purpose`, which is mapped to several databases, with the properties :`inField` and :`drillingStarted`, which are mapped to a single database each. Properties like :`purpose` may or may not have values for the same entity in different datasets. So the SPARQL query $(?w, \texttt{:purpose}, ?p)$ returns both $\{?w \mapsto \texttt{:NationalWellbore/1/3-1},\ ?p \mapsto \texttt{WILDCAT}\}$ and $\{?w \mapsto \texttt{:CorporateWellbore/NO\_1/3-1},\ ?p \mapsto \texttt{WILDCAT}\}$ as answers, while the query $((?w, \texttt{:inField}, ?f), (?w, \texttt{:drillingStarted}, ?d))$ has no answers.

There are different ways of improving this situation. Perhaps the most obvious and direct one is to just change the mapping set such that the wellbores are mapped into equal IRIs. This is not always possible, as a data source may not be under the control of the mapping author; consider, e.g., Linked Open Data. Forcing the same IRIs also makes it hard to scale mapping construction, as the equal IRIs must be enforced everywhere. Another approach is to explicitly declare mappings so that to virtually generate the `sameAs` relation like $(\texttt{:NationalWellbore/1/3-1}, \texttt{sameAs}, \texttt{:CorporateWellbore/NO\_1/3-1})$. ■

3 Canonical IRI Semantics

We observe that the standard `sameAs` semantics is not scalable for query answering in general. Suppose that $\mu = \{?x_1 \mapsto a_1, \dots, ?x_n \mapsto a_n\}$ is a solution mapping of a SPARQL query q over an ABox that includes assertions $\{\texttt{sameAs}(a_i, b_i) \mid 1 \leq i \leq n\}$. Then one can replace any a_i by b_i in μ, resulting in a semantically equivalent solution mapping, which is still a valid answer to q. In this case, there are 2^n such possibilities.

[10] The IRI encoding of special symbols like "/" is omitted for readability.

This example shows that in general, `sameAs` could cause an exponential blowup of query results. Such large numbers of answers are not only detrimental to the performance of query evaluation, but also make the query answers difficult to understand due to the redundancy they introduce. Indeed, our motivation is to integrate multiple datasets, and what we care most are the entities, not the IRIs that represent them. This motivates us to propose an alternative approach, which is based on assigning *canonical IRIs* to entities.

We assume that, in addition to $\mathcal{A}$, we have a set $\mathcal{A}^c$ of *canonical IRI assertions* using the property `canIriOf`. We make the following assumption on $\mathcal{A}^c$, which states that one IRI cannot have more than two canonical representations:

Assumption 1 *The property* `canIriOf` *is inverse functional in* $\mathcal{A}^c$*:*

$$\{\ \mathtt{canIriOf}(c_1, i),\ \mathtt{canIriOf}(c_2, i)\ \} \subseteq \mathcal{A}^c \quad \textit{implies} \quad c_1 = c_2.$$

Definition 1 (Canonical IRI and Canonical Graph). *For an IRI o, the* canonical IRI $\mathit{can}_{\mathcal{A}^c}(o)$ *of o is c_o, if* $\mathtt{canIriOf}(c_o, o) \in \mathcal{A}^c$ *for some c_o, and o otherwise. Given $\mathcal{A}$ and $\mathcal{A}^c$, the* canonical graph $\mathit{cg}(\mathcal{A}, \mathcal{A}^c)$ *is*

$$\{C(\mathit{can}_{\mathcal{A}^c}(o)) \mid C(o) \in \mathcal{A}\} \ \cup\ \{P(\mathit{can}_{\mathcal{A}^c}(o_1), \mathit{can}_{\mathcal{A}^c}(o_2)) \mid P(o_1, o_2) \in \mathcal{A}\}.$$

Given an entailment regime E without special treatments of canonical IRI assertions, we can now define the *canonical IRI entailment regime* $E{+}\mathit{can}$ as:

$$\mathit{eg}^{E+\mathit{can}}(\mathcal{T}, \mathcal{A} \cup \mathcal{A}^c) \ = \ \mathit{eg}^{E}(\mathcal{T}, \mathit{cg}(\mathcal{A}, \mathcal{A}^c))$$

Now we study the relationship between the canonical IRI semantics and the `sameAs` semantics. To do so, we first define the correspondence between $\mathcal{A}^c$ and $\mathcal{A}^s$.

Definition 2. *We say that* $\mathcal{A}^c$ is compliant with $\mathcal{A}^s$ *if the following two conditions hold:*
1. *if* $\mathtt{canIriOf}(o_1, o_2) \in \mathcal{A}^c$*, then* $\mathtt{sameAs}(o_1, o_2) \in (\mathcal{A}^s)^*$*;*
2. *if* $\mathtt{sameAs}(o_1, o_2) \in (\mathcal{A}^s)^*$*, then* $\mathit{can}_{\mathcal{A}^c}(o_1) = \mathit{can}_{\mathcal{A}^c}(o_2)$*.*

Next, we extend canonical graphs to SPARQL queries. More precisely, given a SPARQL query Q, we obtain $\mathit{cg}(Q, \mathcal{A}^c)$ by replacing each IRI o in Q by $\mathit{can}_{\mathcal{A}^c}(o)$.

In the following, we assume that SPARQL queries contain neither `sameAs` nor `canIriOf`. The next proposition shows that query answering under the canonical IRI and the `sameAs` entailment regimes are equivalent in the following sense:

Theorem 1. *Let $(\mathcal{T}, \mathcal{A})$ be a KB, $\mathcal{A}^c$ compliant with $\mathcal{A}^s$, Q a SPARQL query, $\mathit{sig}(Q) = \{?x_1, \ldots, ?x_n\}$, and E an entailment regime that does not imply new* `sameAs` *or* `canIriOf` *assertions. It holds that*

- *if* $\{?x_1 \mapsto o_1, \ldots, ?x_n \mapsto o_n\} \in [\![Q]\!]^{E+\mathit{sameAs}}_{\mathcal{T}, \mathcal{A}\cup\mathcal{A}^s}$*, then*
$$\{?x_1 {\mapsto} \mathit{can}_{\mathcal{A}^c}(o_1), \ldots, ?x_n {\mapsto} \mathit{can}_{\mathcal{A}^c}(o_n)\} \in [\![\mathit{cg}(Q, \mathcal{A}^c)]\!]^{E+\mathit{can}}_{\mathcal{T}, \mathcal{A}\cup\mathcal{A}^c};$$
- *if* $\{?x_1 \mapsto o_1, \ldots, ?x_n \mapsto o_n\} \in [\![Q]\!]^{E+\mathit{can}}_{\mathcal{T}, \mathcal{A}\cup\mathcal{A}^c}$*, and* $\mathtt{sameAs}(o_j, o'_j) \in (\mathcal{A}^s)^*$*, for $j \in J \subseteq \{1, \ldots, n\}$, then*
$$\{?x_j \mapsto o_j \mid j \notin J\} \cup \{?x_j \mapsto o'_j \mid j \in J\} \in [\![Q]\!]^{E+\mathit{sameAs}}_{\mathcal{T}, \mathcal{A}\cup\mathcal{A}^s}.$$

Example 2. Referring to Example 1, a possible set $\mathcal{A}_c$ of canonical IRI assertions (satisfying Assumption 1) is:

$\{$ (:`Wellbore/1`, `canIriOf`, :`NationalWellbore/1/3-1`),
(:`Wellbore/1`, `canIriOf`, :`CorporateWellbore/NO_1/3-1`),
(:`Wellbore/2`, `canIriOf`, :`NationalWellbore/2/4-2`),
(:`Wellbore/2`, `canIriOf`, :`CorporateWellbore/NO_2/4-2`) $\}$

An example of an $\mathcal{A}_s$ such that $\mathcal{A}_c$ is compliant with $\mathcal{A}_s$ is:

$\{$ (:`NationalWellbore/1`, `sameAs`, :`CorporateWellbore/1`),
(:`NationalWellbore/2`, `sameAs`, :`CorporateWellbore/2`) $\}$

Here we have chosen the canonical representative to always be of the (new) form :`Wellbore/`$\{id\}$. This is a choice made by the mapping author, and the canonical representative could also have been one of the existing IRIs. If we let D be the database and $\mathcal{M}$ the mappings from Example 1, the query
$Q = \texttt{SELECT}\ ?w, ?f, ?d\ \texttt{WHERE}\ ((?w, \texttt{:inField}, ?f), (?w, \texttt{:drillingStarted}, ?d))$
has a non-empty answer set over the canonical graph $cg(\mathcal{A}_{\mathcal{M},D}, \mathcal{A}_c)$, including the answer $\{?w \mapsto$: `Wellbore/1`, $?f \mapsto$ `EKOFISK`, $?d \mapsto$ 06-07-1968$\}$. In Section 5 we will see mapping assertions that populate this ABox. ∎

4 Handling Canonical IRI Semantics by Query Rewriting

In this section, we develop a query rewriting algorithm for canonical IRI semantics so that the canonical graph does not need to be materialized. However, this rewriting algorithm is mostly of theoretical interest but not meant to be implemented, since the structure of rewritten queries is too complex.

We recall that BGP queries under the simple entailment regime are monotonic (in the input graph G): if $s \in [\![Q]\!]_G$ then $s \in [\![Q]\!]_{G \cup \Delta G}$. However, observe that answering BGP queries under canonical IRI semantics are non-monotonic: for example let $G = \{(a, \texttt{rdf:type}, C)\}$, $\Delta G = \{(c_a, \texttt{canIriOf}, a)\}$, and $q = (?x, \texttt{rdf:type}, C)$, then $\{?x \mapsto a\} \in [\![Q]\!]^{can}_G$ but $\{?x \mapsto a\} \notin [\![Q]\!]^{can}_{G \cup \Delta G}$. This means that for any BGPs Q, there is no union of BGPs Q' such that for all RDF graphs G, $[\![Q]\!]^{can}_G = [\![Q']\!]_G$. The reason is that $[\![Q]\!]^{can}_G$ is always non-monotonic in G, while $[\![Q']\!]_G$ is always monotonic in G. Thus we cannot rewrite a BGP under the canonical IRI semantics to a (union of) BGPs under the standard SPARQL semantics and expect to obtain the same answers. In the following, we show that query answering under the E+can entailment regime can be reduced to query answering under the E entailment regime, by using non-monotonic SPARQL construct like `NOT EXISTS`.

To define such a translation, we first introduce a subquery, denoted $qc[?xc, ?x]$, that returns $can_{\mathcal{A}^c}(?x)$ for $?xc$. This is achieved by defining

$$\begin{aligned} qc[?xc, ?x] = {} & \texttt{SELECT}\ ?xc, ?x\ \texttt{WHERE}\ (\ (?xc, \texttt{canIriOf}, ?x)\ \texttt{UNION} \\ & (\texttt{SELECT}\ ?x\ \texttt{AS}\ ?xc, ?x\ \texttt{WHERE}\ ((?x, ?p_1, ?o)\ \texttt{UNION}\ (?s, ?p_2, ?x)) \\ & \ \texttt{FILTER NOT EXISTS}\ (?y, \texttt{canIriOf}, ?x))) \end{aligned}$$

Lemma 1. *Let $\mathcal{A}$ and $\mathcal{A}^c$ be as above and a an individual occurring in $\mathcal{A}$. Then $\{?xc \mapsto c_a, ?x \mapsto a\} \in [\![qc[?xc, ?x]]\!]_{\mathcal{A} \cup \mathcal{A}^c}$ iff $c_a = can_{\mathcal{A}^c}(a)$.*

Definition 3. *The* canonical-iri rewriting *$\psi(Q)$ of a SPARQL query Q is obtained from Q by replacing each triple pattern t with the sub-query $\psi(t)$ obtained from t as follows:*

central.aliasTable

id	nationalName	corporateName
1	NO_1/3-1	1/3-1
2	NO_2/4-2	2/4-2
3	NO_1/2-1	1/2-1
4	NO_1/3-A-1_H	
5		1/3-10

```
:Wellbore/{id} canIriOf
  :NationalWellbore/{nationalName} .
← SELECT id, nationalName FROM central.aliasTable

:Wellbore/{id} canIriOf
  :CorporateWellbore/{corporateName} .
← SELECT id, corporateName FROM central.aliasTable
```

Fig. 1: Table and mappings for Example 5

(1) *for each variable* $?x$*, introduce a fresh variable* $?x_c$*,*
(2) *for each* occurrence *of a variable* $?x$*, introduce a fresh variable* $?x_o$*, change* $?x$ *to* $?x_o$*, and join with* $qc[?x_c, ?x_o]$*, and*
(3) *add a projection* $?x_c$ AS $?x$*.*

Example 3. Consider Example 1 and the query
$Q = \texttt{SELECT}\ ?w, ?f, ?d\ \texttt{WHERE}\ ((?w, \texttt{:inField}, ?f), (?w, \texttt{:drillingStarted}, ?d))$.
Then

$$\begin{aligned}\psi(Q) = {} & \texttt{SELECT}\ ?w, ?f, ?d\ \texttt{WHERE}\ (\\ & \quad (\texttt{SELECT}\ ?wc1\ \texttt{AS}\ ?w, ?fc\ \texttt{AS}\ ?f\ \texttt{WHERE} \\ & \quad\quad ((?wo1, \texttt{:inField}, ?fo),\ qc[?wc1, ?wo1],\ qc[?fc, ?fo])), \\ & \quad (\texttt{SELECT}\ ?wc2\ \texttt{AS}\ ?w, ?dc\ \texttt{AS}\ ?d\ \texttt{WHERE} \\ & \quad\quad ((?wo2, \texttt{:drillingStarted}, ?d),\ qc[?wc2, ?wo2],\ qc[?dc, ?do]))) \quad \blacksquare\end{aligned}$$

We note that Step 2 in Definition 3 is necessary to deal with each *occurrence* of a variable separately, as shown in the following example:

Example 4. Let $\mathcal{A} = \{(a, \texttt{P}, b)\}$, $\mathcal{A}^c = \{(c, \texttt{canIriOf}, a), (c, \texttt{canIriOf}, b)\}$, and $B = (?x, \texttt{P}, ?x)$ a BGP. It is easy to see that the following query returns the expected answers:
$\psi(B) = \texttt{SELECT}\ ?xc\ \texttt{AS}\ ?x\ \texttt{WHERE}\ ((?x', \texttt{P}, ?x''),\ qc[?xc, ?x'],\ qc[?xc, ?x''])$. $\blacksquare$

Theorem 2. *For any* $\mathcal{T}$*,* $\mathcal{A}$*,* $\mathcal{A}^c$*,* E*, and* Q*,* $[\![Q]\!]^{E+can}_{\mathcal{T},\mathcal{A}\cup\mathcal{A}^c} = [\![\psi(Q)]\!]^{E}_{\mathcal{T},\mathcal{A}\cup\mathcal{A}^c}$.

We observe that although the size of the rewritten query is linear in the size of the original one, the approach is impractical, considering that answering each subquery qc requires enumerating all individuals in $\mathcal{A}$.

5 Handling Canonical IRI Statements in OBDA

To handle ontology-based integration of cross-linked datasets, we assume that the mapping set $\mathcal{M}$ may include a subset $\mathcal{M}^c$ consisting of all mapping assertions that populate `canIriOf` triples.

Example 5. We extend the OBDA scenario from Example 1 with mappings for `canIriOf` into tables in a database `central`, both shown in Figure 1.

The canonical IRI mapping assertions relate wellbores in databases `national` and `corporate` by employing an existing aliasing table in database `central`. Materializing these mappings and the database we obtain the assertions in Example 2. *Master tables* like `central` are available in the Statoil use case, and are typical in corporate scenarios.

The canonical IRI mappings do not depend on the existence of such master tables, as arbitrary SQL queries in the source part of mappings can be used. However, regarding both maintainability and performance, our experience is that materializing the resolution of equality into a master table is a better choice. ■

In an OBDA setting, we propose a practical method based on compiling the consequences of canonical IRI semantics into mappings. This method is inspired by the mapping saturation algorithm used for classical OBDA [21,26].

In order to make sure that the RDF graph constructed from the mappings satisfies Assumption 1, we state a stronger assumption on the mappings:

Assumption 2 *For each IRI template iri, there is at most one mapping assertion of the form:* $iri_c(\boldsymbol{a})\ \texttt{canIriOf}\ iri(\boldsymbol{b}) \leftarrow \mathit{sql}(\boldsymbol{a}, \boldsymbol{b})$

Lemma 2. *If $\mathcal{M}^c$ is a set of mapping assertions satisfying Assumption 2, then $\mathcal{A}_{\mathcal{M}^c,D}$ satisfies Assumption 1 for any database instance D.*

Now we are ready to present the mapping rewriting algorithm. Intuitively, it replaces all the individuals and IRI-templates in the mapping by their canonical representation.

Definition 4. *Let $\mathcal{M} = \mathcal{M}' \cup \mathcal{M}^c$ be a set of mapping assertions. The* canonical-iri rewriting *$cm(\mathcal{M}', \mathcal{M}^c)$ of $\mathcal{M}$ is obtained by processing each $m \in \mathcal{M}'$ as follows:*

- *for each IRI template $iri(\boldsymbol{a})$ occurring in m, if $\mathcal{M}^c$ contains a mapping* $iri_c(\boldsymbol{b_0})\ \texttt{canIriOf}\ iri(\boldsymbol{b_1}) \leftarrow \mathit{sql}(\boldsymbol{b_0}, \boldsymbol{b_1})$ *then we replace $iri(\boldsymbol{a})$ in the target of m by $iri_c(\boldsymbol{b_0})$ and join the source query with $\mathit{sql}(\boldsymbol{b_0}, \boldsymbol{b_1}), \boldsymbol{a} = \boldsymbol{b_1}$.*
- *for each occurrence of an IRI o, if $o = iri(\boldsymbol{a})$ for some IRI template iri, then we process it as in the IRI template case.*

Example 6. Let $\mathcal{M}'$ be the mappings in Example 1 and $\mathcal{M}_c$ the mappings in Example 5. Then $cm(\mathcal{M}', \mathcal{M}_c)$ obtained according to Definition 4 is as follows:

```
:Wellbore/{id} :inField {wlbField} ; :purpose {opPurp} .
← SELECT wlbFld, opPurp, id FROM national.wellbore, central.aliasTable
   WHERE name = nationalName

:Wellbore/{id} :drillingStarted {driStDt} ; :purpose {reason} .
← SELECT driStDt, reason, id FROM corporate.drillingops, central.aliasTable
   WHERE name = corporateName                                            ■
```

It is easy to see that the algorithm described in Definition 4 has the same effect as applying cg to the RDF graph:

Lemma 3. *Let $\mathcal{M} = \mathcal{M}' \cup \mathcal{M}^c$ be a set of mapping assertions, and D a database instance. Then $cg(\mathcal{A}_{\mathcal{M}',D}, \mathcal{A}_{\mathcal{M}^c,D}) = \mathcal{A}_{cm(\mathcal{M}',\mathcal{M}^c),D}$*

It follows that the mapping rewriting algorithm is sound and complete.

Theorem 3. *Let $(\mathcal{P}, D)$ be an OBDA specification, where $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \mathcal{S})$ and $\mathcal{M} = \mathcal{M}' \cup \mathcal{M}^c$, and Q a SPARQL query. Then $[\![Q]\!]^{QL+can}_{\mathcal{T},\mathcal{A}_{\mathcal{M},D}} = [\![Q]\!]^{QL}_{\mathcal{T},\mathcal{A}_{cm(\mathcal{M}',\mathcal{M}^c),D}}$.*

This theorem shows that supporting canonical IRI semantics in OBDA can be implemented by simply adding a canonical-iri rewriting step into the existing workflow during the startup of an OBDA system.

The canonical IRI approach shown above shifts effort from the rewriting to the mapping construction. That is, compared with the approach using `sameAs` triples, more information must be encoded in the mappings. Specifically, one must encode transitivity, and choose a representative for each equivalence class. An important question is therefore if this is applicable in practical scenarios. Our experience from the Statoil use case is that one can exploit existing *master tables* correlating the names used for objects in different databases. The mappings for canonical IRIs are simple mapping into these tables.

6 Implementation and Experiments

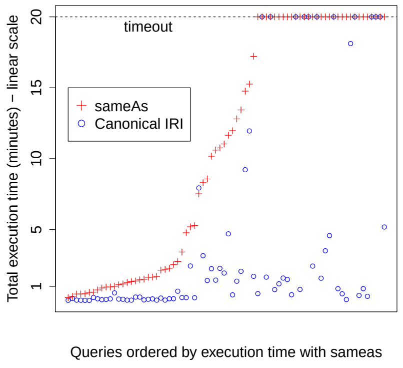
The canonical IRI approach described above has been implemented in the standard distribution of *Ontop* [3], which now supports both sameAs and canonical IRI semantics. Observe that *Ontop* does not perform SQL federation, therefore it usually relies on systems such as Teiid [11] or Exareme [8] to integrate multiple databases. These systems act as *mediators* and expose to *Ontop* a set of tables coming from the different databases. In particular, *Ontop* has been tested with Exareme intensively. Exareme was initially developed as an engine for complex dataflow processing on elastic clouds [20], and was subsequently enriched with data federation capabilities. As a result, the Exareme SQL federation engine is able to decompose complex relational queries, use common sub-expression identification techniques in order to save processing costs (e.g., to process only once identical query fragments coming from different subqueries of a union query), and decide which query fragment should be sent to each external database. Exareme then processes intermediate results coming from different databases in parallel in order to produce the final results.

We now present two sets of experiments evaluating the performance of queries over crossed-linked datasets. One experiment was conducted on a production environment of databases in Statoil, based on queries from geoscientists at the company. Since all the data sources are on production servers with confidential data, the load changes, and the OBDA setting is too complex to isolate different features of this approach, we also created a controlled OBDA environment in our own server to study our technique.

6.1 Real-world experiments

We integrated 7 data sources (relational databases) used in Statoil, extending an existing ontology and the set of mappings, and creating the tables necessary for `sameAs` and `canIriOf`. The queries and ontology are published in [16] and a description of the corporate use case is given in [19]. One of the data sources is the *slegge* database, which is also described in [16] together with the mappings toward this database.

[11] `http://teiid.jboss.org`



	sameAs	Canonical IRI
Total queries	76	76
Timeouts	31	11
Min exec. time	12s	0.50s
Mean exec. time	11m	4.3m
Median	11m	0.77m

Fig. 2: Execution time and statistics for the queries in the federated setting at Statoil, comparing the sameAs and the canonical IRI approaches

The experiments in Statoil were run with a catalogue of 76 SPARQL queries constructed from information needs written down by geologists and geoscientists in the company. The domain of the queries is that of subsurface exploration, with a focus on wellbore information. The most complex query had 23 triple patterns, using object and data properties coming from 5 data sources. The queries were executed with a 20 minute timeout, both with sameAs approach and with the canonical IRI approach.

The *Ontop* rewriting engine and Exareme SQL federation engine all run on virtual machines deployed on the company intranet, as the data cannot be moved out. *Ontop* ran on a single machine, while the Exareme SQL federation ran on 8 other machines. The oracle databases are version 10g, and run on separate machines.[12]

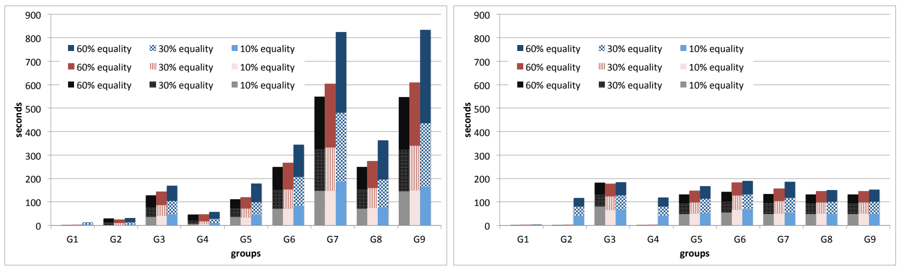
We realize that this setup does not comply with the normal *clean* setup of a database experiment. However, the complexity (7 datasources) and realism (real questions and production databases) of the setup means the results have great value, although their precision is sub-optimal. Compare this with biology, where the *in vivo* experiments on live creatures, dealing with the full complexity of the organisms, may lead to results that cannot be seen in the *in vitro* experiments, and therefore are considered superior.

The total query execution times, for both the sameAs and canonical IRI approaches are shown in Figure 2. The improvement from sameAs to canonical IRI is drastic. With the canonical IRI approach all queries, with three exceptions, are faster, there are fewer timeouts, and the majority of the queries execute within 3 minutes.

6.2 Controlled experiments using Wisconsin Benchmark

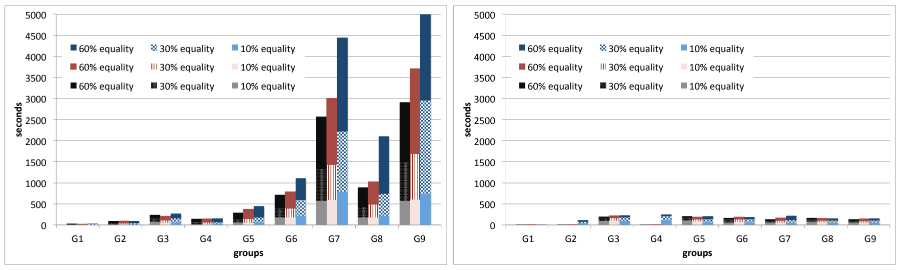
We also evaluated the canonical IRI approach in a more controlled setting. All files needed to reproduce this example are provided online[13]. This example is a simulated federated scenario with a single database consisting of 4 Wisconsin tables [10], representing

[12] Typical machine: HP ProLiant Server, 24 Intel Xeon CPUs (X560@2.67GHz), 283 GB RAM.
[13] `https://github.com/ontop/ontop-examples/tree/master/eswc-2018-canonical-iri`



(a) sameAs (b) canonical IRI

Fig. 3: Execution times of most expensive queries with 2 datasets



(a) sameAs (b) canonical IRI

Fig. 4: Execution times of most expensive queries with 3 datasets

different datasets, and 6 *linking tables*, see [6]. We reused the Wisconsin Benchmark tables, each of them containing 100M rows, and we created 3 new *canonical IRI tables* out of existing linking tables. We added the columns for provenance and canonical id, so that the mappings can generate `canIriOf` relations out of these 3 tables.

To evaluate the overhead of equality reasoning when answering SPARQL queries, we considered the following three parameters:

1. number of linked datasets (2–3);
2. selectivity of the query (returning 0.001%, 0.01%, 0.1% of the dataset);
3. number of equal objects between datasets (10%, 30%, 60%).

In total, we ran 1332 queries, grouped in 9 groups: (G1) no properties, (G2) 1 data property and 0 object properties, (G3) 0 data properties and 1 object property, ..., (G9) 2 data properties and 2 object properties.

The results confirm an improvement, reducing the high cost of execution of the queries. In the previous setting based on sameAs, with the 6 linking tables and 2 linked-datasets scenario, with 120M equal objects (60%), in the worst case, most of the queries ran in around 3 min, while with the new canonical IRI approach, all queries can be executed in around 1 min. The query that performed worst in the previous setting (4 joins, 2 data properties, 2 object properties) returned 480 000 results and took around 6 min. In the new setting, for the same query, the number of results reduced to 60 000, avoiding duplicates. The resulting SQL query is simpler, and can be executed in only 50 sec. In the 3 linked-datasets scenario, the improvement is even more visible: the slowest executions took around 9 min in the previous setting, and less than 1.5 min in the new setting. The worst query in the previous setting took around 1.5 hours, and returned 1 620 000 results. This query can now be executed in 53 sec, and the number of results returned is significantly reduced to 60 000. The number of linked datasets is the variable that has most impact on query performance, but it is less influential in the canonical IRI setting. Another observation is that compared to the sameAs setting, the optimization for canonical IRI semantics is done only in the off-line phase and additional startup time is negligible. In fact, the startup time is only around 5 sec.

In Figures 3 and 4, we visualize the comparison of the execution times in these two settings. For each group, we show the execution times of the most expensive queries. The results confirm that under the canonical IRI semantics, we are able to run the queries significantly faster and the query answering times are more uniform.

7 Conclusions

This work is a natural continuation of [6], which was the first work to study the issue of equivalence of IRIs in OBDI and used the standard `sameAs` construct to model equivalence between entities. Our proposal based on canonical IRIs improve on such an approach in terms of efficiency, and avoids the drawback of redundant answers.

Another important aspect in OBDI is how to efficiently evaluate the rewritten SQL queries over federated databases. Several federated database systems and prototypes have been presented in the literature. Early approaches include TSIMMIS [7], Garlic [14], and Tukwila [17]. Unpredictability regarding processing is the main issue that these systems have to cope with, using techniques such as adaptive query planning and query caching. BigDAWG [11] is a more recent approach that lays emphasis on the “one size does not fit all” principle, by trying to take advantage of specialized engines used as endpoints in order to efficiently process different types of data. How to natively deal with federated query evaluation in the setting of OBDI is subject of further work.

References

1. J. Bao et al. OWL 2 Web Ontology Language document overview (second edition). W3C Recommendation, W3C, Dec. 2012. Available at `http://www.w3.org/TR/owl2-overview/`.
2. S. Brüggemann, K. Bereta, G. Xiao, and M. Koubarakis. Ontology-based data access for maritime security. In *Proc. of ESWC 2016*, volume 9678 of *LNCS*, pages 741–757. Springer, 2016.
3. D. Calvanese, B. Cogrel, S. Komla-Ebri, R. Kontchakov, D. Lanti, M. Rezk, M. Rodriguez-Muro, and G. Xiao. Ontop: Answering SPARQL queries over relational databases. *Semantic Web J.*, 8(3):471–487, 2017.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, and M. Ruzzi. Data integration through $\mathit{DL\text{-}Lite}_{\mathcal{A}}$ ontologies. In K.-D. Schewe and B. Thalheim, editors, *Revised Selected Papers of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008)*, volume 4925 of *LNCS*, pages 26–47. Springer, 2008.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

6. D. Calvanese, M. Giese, D. Hovland, and M. Rezk. Ontology-based integration of cross-linked datasets. In *Proc. of ISWC 2015*, volume 9366 of *LNCS*, pages 199–216. Springer, 2015.
7. S. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of the 10th Meeting of the Inf. Proc. Society of Japan (IPSJ 1994)*, pages 7–18, 1994.
8. Y. Chronis, Y. Foufoulas, V. Nikolopoulos, A. Papadopoulos, L. Stamatogiannakis, C. Svingos, and Y. E. Ioannidis. A relational approach to complex dataflows. In *Proc. of the EDBT/ICDT Workshops*, volume 1558 of *CEUR*, `ceur-ws.org`, 2016.
9. S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. W3C Recommendation, W3C, Sept. 2012. Available at `http://www.w3.org/TR/r2rml/`.
10. D. J. DeWitt. The Wisconsin benchmark: Past, present, and future. In J. Gray, editor, *The Benchmark Handbook*. Morgan Kaufmann, 1992.
11. J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. Zdonik. The BigDAWG polystore system. *SIGMOD Record*, 44(2):11–16, 2015.
12. M. Giese, A. Soylu, G. Vega-Gorgojo, A. Waaler, P. Haase, E. Jiménez-Ruiz, D. Lanti, M. Rezk, G. Xiao, Ö. L. Özçep, and R. Rosati. Optique: Zooming in on Big Data. *IEEE Computer*, 48(3):60–67, 2015.
13. B. Glimm and C. Ogbuji. SPARQL 1.1 entailment regimes. W3C Recommendation, W3C, Mar. 2013. Available at `http://www.w3.org/TR/sparql11-entailment/`.
14. L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proc. of VLDB 1997*, pages 276–285, 1997.
15. S. Harris and A. Seaborne. SPARQL 1.1 query language. W3C Recommendation, W3C, Mar. 2013. Available at `http://www.w3.org/TR/sparql11-query`.
16. D. Hovland, R. Kontchakov, M. G. Skjæveland, A. Waaler, and M. Zakharyaschev. Ontology-based data access to Slegge. In *Proc. of ISWC 2017*, volume 10588 of *LNCS*, pages 120–129. Springer, 2017.
17. Z. G. Ives, D. Florescu, M. Friedman, A. Levy, and D. S. Weld. An adaptive query execution system for data integration. *SIGMOD Record*, 28(2):299–310, 1999.
18. M. Kaminski, E. V. Kostylev, and B. Cuenca Grau. Query nesting, assignment, and aggregation in SPARQL 1.1. *ACM Trans. on Database Systems*, 42(3):17:1–17:46, 2017.
19. E. Kharlamov et al. Ontology based data access in Statoil. *J. of Web Semantics*, 44:3–36, 2017.
20. H. Kllapi, E. Sitaridi, M. M. Tsangaris, and Y. Ioannidis. Schedule optimization for data processing flows on the cloud. In *Proc. of ACM SIGMOD 2011*, pages 289–300, 2011.
21. R. Kontchakov, M. Rezk, M. Rodriguez-Muro, G. Xiao, and M. Zakharyaschev. Answering SPARQL queries over databases under OWL 2 QL entailment regime. In *Proc. of ISWC 2014*, volume 8796 of *LNCS*, pages 552–567. Springer, 2014.
22. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, W3C, Dec. 2012. Available at `http://www.w3.org/TR/owl2-profiles/`.
23. B. Motik, Y. Nenov, R. E. F. Piro, and I. Horrocks. Handling owl:sameAs via rewriting. In *Proc. of AAAI 2015*, pages 231–237. AAAI Press, 2015.
24. J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. on Database Systems*, 34(3):16:1–16:45, 2009.
25. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, 10:133–173, 2008.
26. F. Priyatna, O. Corcho, and J. F. Sequeda. Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph. In *Proc. of WWW 2014*, 2014.
27. M. Rodriguez-Muro and M. Rezk. Efficient SPARQL-to-SQL with R2RML mappings. *J. of Web Semantics*, 33:141–169, 2015.