Realizing Ontology-based Reusable Interfaces for Data Access via Virtual Knowledge Graphs

Diego Calvanese
calvanese@inf.unibz.it
Faculty of Computer Science
Free University of Bozen-Bolzano
Bolzano, Italy
Department of Computing Science
Umeå University
Umeå, Sweden

Linfang Ding
Alessandro Mosca
Guohui Xiao
ding@inf.unibz.it
alessandro.mosca@unibz.it
xiao@inf.unibz.it
Faculty of Computer Science
Free University of Bozen-Bolzano
Bolzano, Italy

ABSTRACT

In this paper, we present a comprehensive framework, which we call VKG-UI, for realizing ontology-based reusable user interfaces (UIs) for data access via virtual knowledge graphs (VKGs). The VKG approach uses an ontology to model the domain of interest and to hide the heterogeneity of the underlying data sources. Reusable UIs can be built by relying on queries that are issued to the VKG system and that use the high level vocabulary from the ontology layer. This use of VKGs allows for decoupling the data from the UIs, and brings great reusability in designing the latter. To illustrate our approach, we introduce significant use cases with various types of UIs, including programming, graphic, natural language, and voice interfaces.

KEYWORDS

ontology, virtual knowledge graph, data access, user interface

ACM Reference Format:
Diego Calvanese, Linfang Ding, Alessandro Mosca, and Guohui Xiao. 2021. Realizing Ontology-based Reusable Interfaces for Data Access via Virtual Knowledge Graphs. In *CHItaly 2021: 14th Biannual Conference of the Italian SIGCHI Chapter (CHItaly '21), July 11–13, 2021, Bolzano, Italy.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3464385.3464744

1 INTRODUCTION

The idea of exploiting ontologies as formal models to lower the cost of designing, developing, and maintaining user interfaces (UIs) of information systems dates back to the end of the 1990s and represented a natural evolution of the so-called model-based approach in software engineering [15]. Since then, ontologies have been used to provide non-ambiguous, machine-readable representations of the main elements characterising the visualisation capabilities, interaction possibilities, and the development process of a UI. Among other elements, ontologies have been proposed to model *users*, *roles*, *contexts*, *real world domains*, *devices*, and *software modules*, including the concept of interface itself and its main ingredients (e.g., input controls, navigational and informational components, widgets). Paulheim and Probst [11] provide a comprehensive survey on the topic of enhancing user interfaces with ontologies.

In this work, we rely on a substantially different use of ontologies in information systems, namely as a means to provide high-level access to different kinds of data sources. In this approach, known as *ontology-based data access* (OBDA) [18], which recently became also popular as *Virtual Knowledge Graphs* (VKGs) [19], the attention has been put on two orthogonal aspects: on the one hand, on the computational efficiency in processing information requests, i.e., queries, posed over the ontology to extract specific data from the underlying sources; and on the other hand, on the conceptual simplicity in formulating such requests. If we keep in mind the goal of supporting the re-use of already existing UI software components for data access [3, 10], the greatest advantage of embracing a VKG approach resides in the neat distinction between the (back-end) functionalities for data management and the (front-end) functionalities for data access and exploration. The VKG back-end exposes a formal declarative representation of the data sources in terms of a pair made of an ontology and a mapping: the ontology, in particular, is the only part of the specification that a front-end component needs to be aware of in order to work as an interface between the user requests and the integrated data. Under these conditions, it is clear that the definition of both the ontology and the SPARQL queries [8] is fully independent from any specific UI that can be built on top of them, provided that the UIs have a mechanism to manage and issue the execution of these queries. Let us then observe that in a VKG-based system:

(1) changing an interface component or an entire UI does not require any further modification or code writing in the component/UI except for the specification of those functions that are responsible of loading and issuing the appropriate SPARQL queries;
(2) the integration of further data sources or the update of the ones that were already present, does not affect the behaviour of the UI (as long as these changes leave the ontology layer unchanged).

CHItaly '21, July 11–13, 2021, Bolzano, Italy

ACM ISBN 978-1-4503-8977-8/21/07…$15.00
https://doi.org/10.1145/3464385.3464744

Due to the above considerations, we claim that our proposal to rely on VKG systems represents an original contribution to the design and implementation of ontology-enhanced user interfaces for data access: being a high-level declarative representation of the system data sources, ontologies in VKG systems not only provide end users with a conceptual interface they can directly exploit to access the data by formulating queries in a vocabulary they are familiar with, but they also constitute a programming-oriented interface for dedicated UI modules whose aim is to support users with more advanced or customised data exploration features (e.g., visual, faceted, or natural language-based).

2 VIRTUAL KNOWLEDGE GRAPHS

We provide now the necessary background on ontologies and virtual knowledge graphs. To illustrate the relevant notions, we use as a running example the Open Data Hub-Virtual Knowledge Graph (ODH-VKG) project, which is a joint project between NOI Techpark[1] and Ontopic[2] for publishing South Tyrolean tourism data as a knowledge graph[3]. The *Virtual Knowledge Graph* paradigm is a popular paradigm that enables end users to access data sources through an ontology. The VKG framework, illustrated in the lower part of Figure 1, consists of the following key elements:

- *Data sources.* These are the data sources to be accessed, which normally are relational databases. Other data formats (e.g., Excel files and CSV files) can be loaded in a database (e.g., PostgreSQL) or accessed through a data federation tool (e.g., Dremio), which exposes them in the form of relational tables. A data federation tool is also used to wrap a collection of (possibly heterogeneous) data sources and present them as a single relational source.
- *Ontology.* The VKG defines high-level concepts that model the domain of interest in terms of an OWL 2 [9] or RDFS ontology. The ontology models the domain of interest, hides the heterogeneity of the underlying data sources, and can be used to guide the formulation of appropriate queries.
- *Mapping.* The ontology is semantically linked to the data sources by means of a mapping, consisting of a set of mapping assertions [13]. The mapping language standardized by the W3C and typically adopted in the VKG setting is R2RML [4]. In the following, however, we provide examples of mappings in the native mapping language of the Ontop system, which is more compact and readable than R2RML.

The ontology and mapping, which together with the (relational) schema of the underlying data sources are called a *VKG specification*, expose the data source as a *virtual* RDF graph [16], and make it accessible at query time through queries expressed in the SPARQL language [8]. The approach is called *virtual* because it actually avoids to materialize the RDF graph. Instead, queries formulated over the ontology vocabulary are answered by being translated on the fly into queries over the original sources, while performing also ontological reasoning. VKG systems implementing this paradigm include Mastro[4] [2], Morph[5] [14], Ontop [1, 20], Stardog[6], and Ultrawrap [17].

In the following example, we illustrate how mappings work, by describing the RDF graph that the mapping exposes (virtually). For readers who are not familiar with RDF, we recall that an RDF graph consists of a set of *triples* of the form (s, p, o), where the *subject* s is an individual, denoted by an IRI (a form of identifier on the Web), the *object* o is either an individual or a literal (i.e., a value such as a string, integer, etc.), and the *predicate* p denotes a binary relation connecting s to o. In addition, a triple of the form $(s, \texttt{rdf:type}, C)$, using the special predicate '`rdf:type`', denotes that the individual s is an instance of the class C. We use here the *turtle syntax*, where a triple (s, p, o) is written as "$s\ p\ o$.".

A mapping is a set of *mapping assertions*, each of which consists of a *source* part, which is a SQL query over the data source, and a *target* part, which is a *triple template*, i.e., a set of triples written in the turtle syntax with placeholders (enclosed in '`{`' and '`}`') that are answer variables from the source query. Intuitively, when evaluating the SQL source query over the data source, for each answer in the query result, the mapping assertion instantiates the template by replacing the placeholders with concrete values from the answer, and exposes the set of triples generated in this way.

Example 2.1. In OHD-VKG, since the main information about the concept `schema:FoodEstablishment` in the project ontology is stored in the relational table `v_gastronomiesopen`, the following mapping assertion constructs instances of the class `schema:FoodEstablishment`, their names in German, and their geometries:

```
target:
  data:gastronomy/{Id} rdf:type
    schema:FoodEstablishment .
  data:gastronomy/{Id} schema:name {Detail-de-Title}@de .
  data:gastronomy/{Id} geo:hasGeometry
    data:geo/gastronomy/{Id} .
  data:geo/gastronomy/{Id} rdf:type sf:Point .
  data:geo/gastronomy/{Id} geo:asWKT
    "POINT␣({Longitude}␣{Latitude})"^^geo:wktLiteral .
source:
  SELECT Id, Longitude, Latitude, Detail-de-Title
  FROM v_gastronomiesopen
```

If the source SQL query returns the answer (`'GASTROE9316', 11.448191, 46.495352, 'Gasthof Schlosshof'`), the above mapping assertion (virtually) produces the following RDF triples:

```
data:gastronomy/GASTROE9316 rdf:type
  schema:FoodEstablishment .
data:gastronomy/GASTROE9316 schema:name
  "Gasthof␣Schlosshof"@de .
data:gastronomy/GASTROE9316 geo:hasGeometry
  data:geo/gastronomy/GASTROE9316 .
data:geo/gastronomy/GASTROE9316 rdf:type sf:Point .
data:geo/gastronomy/GASTROE9316 geo:asWKT
  "POINT␣(11.448191␣46.495352)"^^geo:wktLiteral .
```

Considering that the classification of `schema:FoodEstablishment` is stored in the `Shortname` column of the table `v_CategoryCodes`, the following mapping assertion constructs beer gardens, i.e., instances of `:BeerGarden`:

[1] https://noi.bz.it/en/
[2] http://ontopic.ai/
[3] The complete source code of ODH-VKG is available under the AGPL-3.0 License on GitHub: https://github.com/noi-techpark/it.bz.opendatahub.sparql

[4] http://www.obdasystems.com/it/mastro/
[5] https://github.com/oeg-upm/morph-rdb/
[6] https://www.stardog.com/

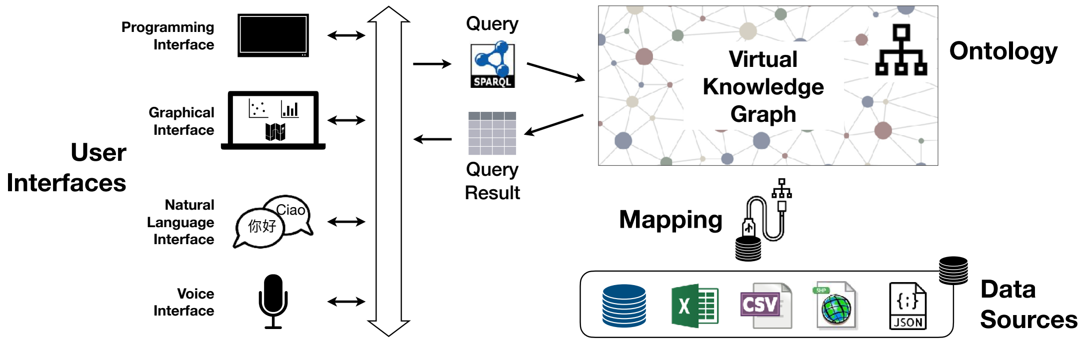

Figure 1: The VKG-UI framework of ontology-based reusable interfaces for data access via Virtual Knowledge Graphs.

```
target: data:gastronomy/{gastronomiesopen_Id} rdf:type
           :BeerGarden .
source: SELECT gastronomiesopen_Id FROM v_CategoryCodes
        WHERE Shortname = 'Braugarten'
```

Since 'GASTROE9316' is an answer to the above source query, the above mapping assertion produces the RDF triple "data:gastronomy/GASTROE9316 rdf:type :BeerGarden". The VKG is then exposed as a standard SPARQL endpoint, which implies that clients can communicate with the endpoint using the standard HTTP protocol [6]. E.g., the following SPARQL query retrieves all the beer gardens, their names in German, and their locations:

```
SELECT ?b ?pos ?posLabel
WHERE {
  ?b rdf:type :BeerGarden .   ?b geo:hasGeometry ?g .
  ?g geo:asWKT ?pos .   ?b schema:name ?posLabel .
  FILTER (lang(?posLabel) = 'de')
}
```

The Ontop system translates this SPARQL query to the following SQL query (which we have simplified for readability):

```
SELECT CONCAT('data:gastronomy/', Id) AS b,
  CONCAT("POINT␣(", Longitude, "␣", Latitude, ")") AS pos,
  Detail-de-Title AS posLabel
FROM v_gastronomiesopen v1, v_CategoryCodes v2
WHERE v1.Id = v2.gastronomiesopen_Id AND
      v2.Shortname = 'Braugarten'
```

One answer to this SPARQL query is:

```
?b = data:gastronomy/GASTROE9316,
?pos = "POINT␣(11.448191␣46.495352)"^^geo:wktLiteral,
?posLabel = "Gasthof␣Schlosshof"@de
```

3 THE VKG-UI FRAMEWORK

We present now the VKG-UI framework, for realizing ontology-based reusable interfaces for data access via VKGs. Figure 1 depicts the structure of the framework, where arrows indicate information flow. The right part of the diagram is a VKG providing an ontological representation of the underlying data through mappings. The left part are various kinds of user interfaces for accessing the information in the data sources through the VKG. The interaction between the UIs and the VKG is via SPARQL queries formulated over the vocabulary of the ontology.

We stress that decoupling the data and the user interfaces using VKGs brings great reusability in designing the user interfaces themselves. Since the user interfaces only rely on the ontological representation, the framework is robust with respect to changes in the data source layer. Indeed when the ontology is stable, adding new data sources only requires adding more mappings from the new sources to the established concepts in VKGs, but the user interfaces can stay unchanged. Similarly when some data sources change. Below we discuss possible user interfaces with examples:

- *Programming interfaces.* The SPARQL query language already provides a programming interface. This interface can be accessed using the SPARQL HTTP protocol either directly or through libraries wrapping the HTTP protocol for a programming language, e.g., RDF4J for Java and RDFLib for Python[7]. Other programming interfaces can also be implemented over the SPARQL API. For example, the ODH-VKG project also implemented a Web API[8] generating JSON-LD[9] snippets in the schema.org vocabulary over the SPARQL endpoint. The generated snippets can be embedded into a web page to help search engines to extract structured data from the page[10].
- *Graphical interfaces.* Graphical user interfaces can be implemented on top of VKGs using SPARQL queries. For example, Ontop comes with a generic interface for writing SPARQL queries and visualizing query results using the YASGUI library[11] embedded in a SPARQL endpoint. In the ODH-VKG project we have also built *Web Components*, which are a technology that allows one to create reusable custom elements and utilize them in web apps. With the newly developed Web Components[12], when users want to embed such custom elements in their websites, they only need to write two lines of code: one to import the library, and one to use the component with attributes specifying the endpoint of the Knowledge Graph and the SPARQL query to retrieve the data.
- *Natural language* and *voice interfaces.* Users can interact with VKGs using natural language by typing or speaking. The

[7] https://rdf4j.org/, https://github.com/RDFLib/rdflib
[8] https://github.com/ontopic-vkg/odh-web-api
[9] https://json-ld.org/
[10] https://developers.google.com/search/docs/guides/intro-structured-data
[11] https://yasgui.triply.cc/
[12] https://webcomponents.opendatahub.bz.it/webcomponent/567cb2e2-3e5d-421a-bf85-b8ecc500aab9

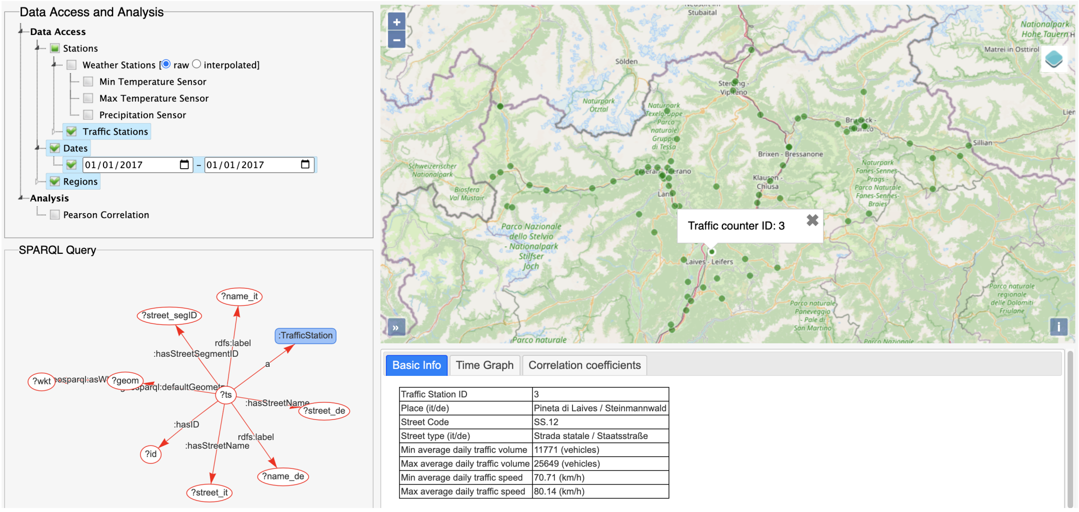

Figure 2: The visual interface for sensor data analysis.

interface will then translate the natural language input into one or multiple SPARQL queries over VKGs. For example, in the ODH-VKG project, we have prototyped a voice interface as an Amazon Alexa Skill[13]. Users can talk to this ODH Skill and get some replies. As an example, when a user says that she is hungry, the skill will recommend a few restaurants nearby, by sending appropriate SPARQL queries to the VKG.

3.1 Sensor Data Integration and Analysis: A Use Case

The research project about Sensor Data Integration and Analysis [5] aims to integrate and analyse various sensor data from the region of South Tyrol using VKGs and geovisual analytics approaches[14]. The project uses data from the Open Data Portal (ODP)[15] and the State Institute for Statistics (ASTAT)[16] of the province of South Tyrol. ODP collects and publishes data on a variety of topics (e.g., meteorology, culture, health) from local authorities, companies, and relevant stakeholders. These data and their metadata are provided in different formats, e.g., JSON, XML, CSV, and PDF. The portal also features a Geocatalog portal[17], providing massive geodata on administrative boundaries, satellite images, and transportation networks. These geodata are available in the formats of ESRI SHP, AutoCAD, Google KML, or GeoJSON. ASTAT coordinates the official statistical activities in the province and provides an interactive database[18], where users can view and download socioeconomic data.

The knowledge about sensor data is represented by means of two standard ontologies, namely GeoSPARQL (with prefix `geo:`) [12] for spatial features and relations, and Semantic Sensor Network (SSN, with prefixes `ssn:` and `sosa:`) [7] for sensors and observations. The core classes are `geo:Feature`, `sosa:Platform`, `sosa:Sensor`, `sosa:ObservableProperty`, and `sosa:Observation`. The resulting ontology is further enriched with domain-specific classes, e.g., the two classes `:WeatherStation` and `:TrafficStation`, as subclasses of both `sosa:Platform` and `geo:Feature`.

Over the VKG, a web-based graphical user interface has been developed for the analysis of meteorological and traffic sensor data. The interface, shown in Figure 2, consists of four basic linked visual components:

(1) A *data access and analysis view* (upper left) listing the core concepts as information items, which connects the ontology model and SPARQL. Users can click/check the intended features to formulate a query to access data. The design of this view is done according to the core vocabularies in the ontology, including stations, sensors, and observable properties. A time window is added to select data in a certain time slot.
(2) A *SPARQL query view* (bottom left) linked to the data access view and showing directly the basic graph patterns of the query. It allows an intuitive perception of the involved concepts and their relations when a query is formulated and issued to the SPARQL endpoint.
(3) A *map view* (upper right) linked with the data access view and the statistical view and showing the spatial distribution of queried objects, e.g., the locations of all the meteo-stations, and the precipitation distribution. Users can interactively select a feature on the map to investigate its characteristics in the linked statistical view.
(4) A *statistical result view* (bottom right) linked to the data access view and the map view, and showing relevant statistics of the selected feature on the map in the selected time period. The three tabs show the basic information of the selected feature (e.g., traffic station ID, and the min and max traffic volumes), time series of the observations, and the correlation coefficients of the weather and traffic data at this station.

[13] https://github.com/ontopic-vkg/odh-alexa
[14] https://github.com/dinglinfang/suedTirolOpenDataOBDA/
[15] http://daten.buergernetz.bz.it/
[16] http://astat.provinz.bz.it/
[17] http://geokatalog.buergernetz.bz.it/geokatalog/
[18] http://astat.provinz.bz.it/de/datenbanken-gemeindedatenblatt.asp

4 CONCLUSIONS

In this paper, we have presented the VKG-UI framework for realizing ontology-based reusable user interfaces (UIs) for data access via virtual knowledge graphs (VKGs). We have shown that the idea of VKG-UI has been successfully applied in a number of use cases with various types of user interfaces. We believe that our work provides a good basis and gives the necessary insights to adopt the VKG-UI framework also in other settings and for different domains of interest.

ACKNOWLEDGMENTS

This research has been partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Italian Basic Research (PRIN) project HOPE, by the EU H2020 project INODE, grant agreement 863410, by the project IDEE (FESR1133) funded by the Eur. Reg. Development Fund (ERDF) Investment for Growth and Jobs Programme 2014-2020, and by the Free University of Bozen-Bolzano through the projects KGID, GeoVKG, and STyLoLa.

REFERENCES

[1] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. 2017. Ontop: Answering SPARQL Queries over Relational Databases. *Semantic Web J.* 8, 3 (2017), 471–487. https://doi.org/10.3233/SW-160217

[2] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. 2011. The Mastro System for Ontology-based Data Access. *Semantic Web J.* 2, 1 (2011), 43–53.

[3] Florian Daniel and Maristella Matera. 2014. *Mashups – Concepts, Models and Architectures*. Springer. https://doi.org/10.1007/978-3-642-55049-2

[4] Souripriya Das, Seema Sundara, and Richard Cyganiak. 2012. *R2RML: RDB to RDF mapping language*. W3C Recommendation. World Wide Web Consortium. Available at http://www.w3.org/TR/r2rml/.

[5] Linfang Ding, Guohui Xiao, Diego Calvanese, and Liqiu Meng. 2020. A Framework Uniting Ontology-Based Geodata Integration and Geovisual Analytics. *Int. J. of Geo-Information* 9, 8 (2020). https://doi.org/10.3390/ijgi9080474

[6] Lee Feigenbaum, Gregory Todd Williams, Kendall Grant Clark, and Elias Torres. 2013. *SPARQL 1.1 Protocol*. W3C Recommendation. World Wide Web Consortium. Available at http://www.w3.org/TR/sparql11-protocol.

[7] Armin Haller, Krzysztof Janowicz, Simon Cox, Danh Le-Phuoc, Kerry Taylor, and Maxime Lefrançois. 2017. *Semantic Sensor Network Ontology*. W3C Recommendation. World Wide Web Consortium. Available at https://www.w3.org/TR/vocab-ssn/.

[8] Steve Harris and Andy Seaborne. 2013. *SPARQL 1.1 Query Language*. W3C Recommendation. World Wide Web Consortium. Available at http://www.w3.org/TR/sparql11-query.

[9] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. 2012. *OWL 2 Web Ontology Language: Primer (Second Edition)*. W3C Recommendation. World Wide Web Consortium. Available at http://www.w3.org/TR/owl2-primer/.

[10] Anis Nouri and Florian Daniel. 2015. Interactive, Live Mashup Development Through UI-Oriented Computing. In *Rapid Mashup Development Tools – Revised Selected Papers of the 1st Int. Rapid Mashup Challenge (RMC 2015) (Communications in Computer and Information Science, Vol. 591)*, Florian Daniel and Cesare Pautasso (Eds.). Springer, 31–49. https://doi.org/10.1007/978-3-319-28727-0_3

[11] Heiko Paulheim and Florian Probst. 2010. Ontology-enhanced User Interfaces: A Survey. *Int. J. on Semantic Web and Information Systems* 6, 2 (2010), 36–59.

[12] Matthew Perry and John Herring. 2011. *GeoSPARQL - A Geographic Query Language for RDF Data*. OGC Candidate Standard OGC 11-052r3. Open Geospatial Consortium.

[13] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. *J. on Data Semantics* 10 (2008), 133–173. https://doi.org/10.1007/978-3-540-77688-8_5

[14] Freddy Priyatna, Oscar Corcho, and Juan F. Sequeda. 2014. Formalisation and Experiences of R2RML-based SPARQL to SQL Query Translation using morph. In *Proc. of the 23rd Int. World Wide Web Conf. (WWW)*. 479–490. https://doi.org/10.1145/2566486.2567981

[15] Alberto Rodrigues da Silva. 2015. Model-driven Engineering: A Survey Supported by the Unified Conceptual Model. *Computer Languages, Systems & Structures* 43 (2015), 139–155. https://doi.org/10.1016/j.cl.2015.06.001

[16] Guus Schreiber and Yves Raimond. 2014. *RDF 1.1 Primer*. W3C Working Group Note. World Wide Web Consortium. Available at http://www.w3.org/TR/rdf11-primer/.

[17] Juan F. Sequeda and Daniel P. Miranker. 2013. Ultrawrap: SPARQL Execution on Relational Data. *J. of Web Semantics* 22 (2013), 19–39.

[18] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. 2018. Ontology-Based Data Access: A Survey. In *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. IJCAI Org., 5511–5519. https://doi.org/10.24963/ijcai.2018/777

[19] Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. 2019. Virtual Knowledge Graphs: An Overview of Systems and Use Cases. *Data Intelligence* 1 (2019), 201–223.

[20] Guohui Xiao, Davide Lanti, Roman Kontchakov, Sarah Komla-Ebri, Elem Güzel-Kalayci, Linfang Ding, Julien Corman, Benjamin Cogrel, Diego Calvanese, and Elena Botoeva. 2020. The Virtual Knowledge Graph System Ontop. In *Proc. of the 19th Int. Semantic Web Conf. (ISWC)*, Vol. 2. 259–277.