Towards a New Generation of Ontology Based Data Access

Oscar Corcho [a,*], Freddy Priyatna [a] and David Chaves-Fraga [a]
[a] *Ontology Engineering Group, Universidad Politécnica de Madrid, Spain*
E-mails: ocorcho@fi.upm.es, fpriyatna@fi.upm.es, dchaves@fi.upm.es

Abstract. Ontology Based Data Access (OBDA) refers to a range of techniques, algorithms and systems that can be used to deal with the heterogeneity of data that is common inside many organisations as well as in inter-organisational settings and more openly on the Web. In OBDA, ontologies are used to provide a global view over multiple local datasets; and mappings are commonly used to describe the relationships between such global and local schemas. Since its inception, this area has evolved in several directions. Initially, the focus was on the translation of original sources into a global schema, and its materialisation, including non-OBDA approaches such as the use of Extract Transform Load (ETL) workflows in data warehouses and, more recently, in data lakes. Then OBDA-based query translation techniques, relying on mappings, were proposed, with the aim of removing the need for materialisation, something especially useful for very dynamic data sources. We think that we are now witnessing the emergence of a new generation of OBDA approaches, driven by the fact that a new set of declarative mapping languages, most of which stem from the W3C Recommendation R2RML for Relational Databases (RDB), are being created. In this paper, we discuss the reasons why new mapping languages are being introduced, and why we think that it may be relevant to work on translations among them, so as to benefit from the engines associated to each of them whenever one language and/or engine is more suitable than another. In this vision paper, we discuss the emerging concept of "mapping translation", the basis for this new generation of OBDA, together with some of its desirable properties: information preservation and query result preservation. We also discuss several scenarios where mapping translation can be or is being already applied, even though this term has not necessarily been used in existing literature.

Keywords: OBDA, Data Translation, Query Translation, Mapping Translation

1. Introduction

Database technologies play a vital role in the development of information systems for all sorts of organisations. So far, relational databases (RDB) are still the dominating type of structure and technology used for data management inside organisations, although other formats (e.g., JSON, spreadsheets, XML) and types of databases (e.g., noSQL, graph databases) have also emerged as alternatives for data representation and management in the last decades.

In the early days of information system development, it was natural for organisations to develop their own data models, which were strongly aligned with their activities. This led to a large heterogeneity across organisations, and even across different departments inside the same organisation. Such heterogeneity was especially evident in the case of organisational changes, merges, etc. Similarly, data warehouses were also used in order to align and materialise data from different sources, normally from the same organisation, so as to provide support for analytical queries and for the generation of reports. These situations made researchers and professionals start working on solutions for data integration, where data from several sources needed to be accessible according to a unified and global view over such local heterogeneous data sources. Popular technologies used in production systems worldwide included the use of Extract-Transform-Load (ETL) workflows to overcome heterogeneity and ensure the availability of data in such

*Corresponding author. E-mail: ocorcho@fi.upm.es.

data warehouses or on integrated databases. Indeed, these approaches are still strongly used nowadays.

In the meantime, data integration challenges became even more relevant since two decades ago, since organisations started using Web technologies to provide access to their data (via Web Services, REST APIs or using Semantic Web and Linked Data approaches), both for their own information system development as well as for data sharing, and later on when public administrations started publishing open data according to public-sector information reuse initiatives. Availability and heterogeneity of data (both in terms of content and format) is nowadays present at an unprecedented level. Following the aforementioned ETL approaches, the term data lake has been rather recently coined to refer to an evolution of data warehouses that considers not only structured data but also the other types of structured, semi-structured and unstructured data formats in which data is made available nowadays, as discussed above.

Over these decades, several approaches have been proposed to tackle data integration challenges. We are specially interested in those that fall under the area of Ontology Based Data Access (OBDA) and Integration (OBDI) [1]. From now on we will refer to both of them, in a general manner, as OBDA. In OBDA, ontologies are used as a global view over heterogeneous data sources. It is quite common to use a mediator-based approach [2], where mediators and wrappers are used as intermediaries to overcome the differences between the local schemas and the global view. In this setting, mappings are commonly used to describe such relationships in a declarative manner. These mappings may be normally exploited in two directions: for **data translation**, so that the original data is transformed and materialised according to the global view (in a similar way as with the use of the aforementioned ETL workflows); and for **query translation** [3, 4], where queries written according to the global schema are transformed into the query language supported by the original data sources and evaluated in the original data management systems, with the results being transformed back according to the global view.

Many different types of OBDA mapping languages have been proposed over the last decades, with a large variety of syntaxes and formats especially in the early ones. Since the standardisation of languages like RDF and OWL, several languages were proposed focused on the transformation from relational databases into RDF (e.g., D2R, R2O). This led to the creation of the RDB2RDF W3C Working Group, which published two recommendations for transforming the content of relational databases into RDF: Direct Mapping [5] and R2RML [6]. The Direct Mapping approach specifies simple transformations that require no intervention from users. R2RML allows specifying transformation rules, such as how URIs should be generated, which columns to be used for the transformation, etc. A bit after R2RML was recommended, and because of its use in different types of contexts, new needs and requirements arose, especially in relation to supporting other formats beyond relational databases, and this resulted in the creation of many new mapping languages, such as RML [7] (to deal with CSVs, JSON and XML data sources), xR2RML [8] (to deal with MongoDB), KR2RML [9] (to deal with nested data), CSVW[1] (to describe CSV files on the Web), or D2RML [10] (for XML, JSON and REST/SPARQL endpoints). In addition to declarative languages, non-declarative mapping languages have also been proposed, such as SPARQL-Generate [11].

There are several reasons why new mapping languages are needed. The first and main reason is that a typical mapping language is designed to work with a specific **data format** (e.g. R2RML is focused on relational databases). Even for a more generic purpose mapping language, such as RML, there may still be a need to extend it to support a more specific technology, such as xR2RML. Another reason is **readability and compactness**. Most mapping languages are designed in a format to be parsed by machines and they do not take into account human readability. Examples of languages created to account for this are RMLC-Iterator (for statistical CSV files) [12] or YARRRML [13]. Lastly, many of existing mapping languages **lack formalisation**, making it difficult to apply query translation techniques.

Therefore, the current situation of an OBDA practitioner that needs to provide access to a varied set of heterogeneous data sources is that there are many different options to select from, and it is difficult to determine which one is better for each situation. Languages are not necessarily interoperable, and many of them come associated with a very specific engine that supports them. However, at the same time, it is clear that most of these languages share many common aspects, such as the description of where the data comes from, how URIs can be created for resources, how triples need to be generated (in a materialised or virtual way),

[1] https://www.w3.org/ns/csvw

etc. Having the possibility of translating among these different languages, covering at least those common characteristics that are shared across languages, would allow practitioners to have the possibility of selecting a wider set of engines to implement their OBDA.

In this paper, we lay out our vision that the next generation of OBDA systems should take advantage of this proliferation of mapping languages. In other words, in addition to the data translation and query translation techniques that have been widely addressed in the state of the art of OBDA so far, the OBDA research community will need to think carefully about how to address mapping translation (See Figure 1).

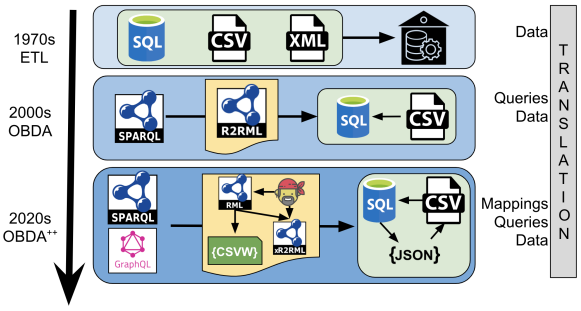

Fig. 1. Timeline for techniques of ETL, current generation and next generation of OBDA systems.

The paper is organised as follows. In Section 2 we informally discuss the concept of mapping translation and some of its desirable properties. A deeper formalisation of the concept and properties is out of the scope of this paper, although we consider it a relevant topic to better understand and characterise ongoing activities in this area. Several scenarios where mapping translation is already being applied or where we think that mapping translation would be clearly applicable are presented in 3. Finally, conclusions and practical implications of this vision are discussed in Section 4.

2. Mapping Translation: Concept and Properties

We define mapping translation as a function that transforms a set of mappings described in one language (we call them original mappings) into a set of mappings described in another language (we call them target mappings).

Our next step is to attach desirable properties for such a function. In this line, we propose to use and adapt some properties that have been described by [14] and [15] in their works. To be more specific, those properties are *information preservation* and *query result preservation* (Figure 2).

The **Information Preservation Property (IPP)** applied to a mapping translation function states that at least there is a function so that the application of such function over the information generated by the application of the target mappings over the original data source returns the same information generated by the application of the original mappings over the same data source.

The **Query Result Preservation Property (QRPP)** applied to a mapping translation function states that for any query that can be evaluated over the information generated from the application of the original mappings to the original data source, there is at least a function to generate another query that can be evaluated over the information generated by the application of the target mappings to the same data source, in such a way that both queries return the same results.

Finally, using these two properties, we define the concepts of weak and strong semantics preservation for a mapping translation function, as follows: a mapping translation function exhibits **weak semantics preservation** if only IPP holds. If both IPP and QRPP are satisfied, then we say that it holds the **strong semantics preservation** property.

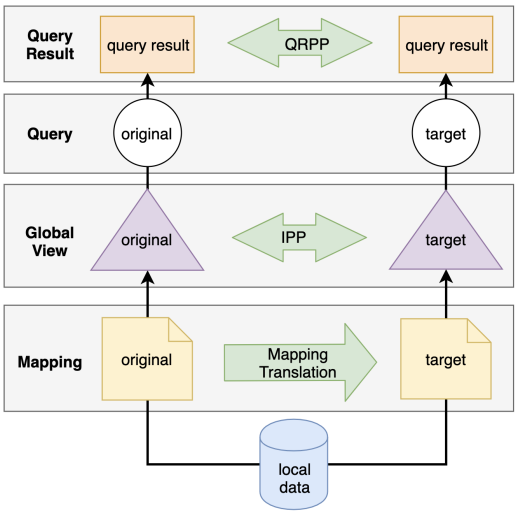

Fig. 2. **Mapping Translator Properties.** The results (triangles) may satisfy the IPP property after the application of the source and target mappings over the same data. In the same way, query results (rectangles) may satisfy the QRPP property when equivalent queries are evaluated over the source and target results.

3. Mapping Translation Scenarios and Challenges

In this section we identify a set of scenarios and challenges in the creation and use of OBDA mapping languages, where mapping translation is relevant. We describe the challenge and provide some references to some of the work presented in the literature addressing or acknowledging it.

3.1. Improving mapping creation and maintenance.

Creating and maintaining OBDA mappings is usually difficult, since mapping languages have been created so that they can be consumed by the corresponding OBDA engines, and they commonly suffer from readability and compactness problems. With respect to **readability**, several approaches have focused on providing mapping editors (e.g., [16]) so that mappings are easier to create by non experts. However, these editors are usually limited to some features of the mapping language or to a specific version of the mapping language specification, and in general they still require knowledge about the underlying mapping language syntax. With respect to **compactness**, there are cases where the generated mapping documents are very long and repetitive, making it difficult to create and maintain [12]. For instance, this is the case when an OBDA approach is used to provide access to multidimensional data sources, such as the ones commonly used to publish statistical data.

We describe two cases where mapping translation ideas are already being applied to address these issues.

YARRRML [13] is a serialisation of RML mappings that uses the YAML (a human-readable data serialization language) format[2]. It is designed with the objective to reduce the size and verbosity of RML. There is no specific engine or parser to exploit YARRRML mappings in an OBDA setting. Instead, the tool Matey[3] is in charge of translating YARRRML mappings into RML, so that any RML-compliant OBDA engine can be used to exploit them.

In the case of multidimensional data (e.g., official statistics data), the W3C RDF DataCube recommendation is the ontology that is commonly used as a global view in an OBDA setting. In most cases, the amount of mappings that would need to be created to link the original data source with the ontology will be rather large and with similar structure. Therefore, there is a high risk that the [R2]RML mapping document(s) generated in the end will contain clerical errors due to copy&paste&edit operations. Furthermore, they will be difficult to maintain. As a result, RMLC-Iterator [12] is proposed as a simplified mapping language specifically designed for this type of data, including two new properties in the R2RML specification: a property to define the array of access columns and a corresponding dictionary if the value of a header needs to be changed. With this approach, the mapping size is drastically reduced. Additionally, a tool is provided to translate RMLC-Iterator mappings into R2RML, hence allowing the use of any R2RML-compliant OBDA engine.

3.2. From declarative mappings to programmed adapters.

Introduced in 2000, REST [17] has become now the most popular architecture for the provision of web services and the implementation of Web-based applications. However, the complexity of software development continues evolving, and aspects that received little attention, such as the size of data being exchanged/transmitted or the number of API calls being made, are now becoming more relevant in the context of mobile application development. As a result, problems like *over-fetching* (a REST endpoint returns more data than what is required by the client) and *under-fetching* (a single REST endpoint does not provide sufficient information requested by the client) are now being discussed. In order to address these problems, Facebook proposed the GraphQL query language [18], used internally since 2012 and released for public use in 2015. Since then it has been increasingly adopted, and GraphQL is now supported by multiple GraphQL engines for major programming languages (e.g. JavaScript, Python, Java, Golang, Ruby).

The two main components of a GraphQL server are the **schema** and the **resolvers**. The GraphQL schema specifies the type of an object together with the fields that can be queried. GraphQL resolvers are data extraction functions implemented in a programming language that are responsible to translate GraphQL queries into queries supported by the underlying datasets (e.g. GraphQL to SQL). In addition, query planning tools have been developed in order to

[2] https://yaml.org/
[3] http://rml.io/yarrrml/matey/

translate GraphQL queries into other query languages (e.g., dataloader[4], joinmonster[5]).

From this basic description of the GraphQL framework, the analogy with the OBDA architecture is clear. Typically, the following tasks need to be done to setup a GraphQL server:

1. A domain expert will analyse the underlying datasets, propose a unified GraphQL schema and describe how the source data sources will need to be mapped into it. Note that there is no standard mechanism to represent these mappings.
2. A software developer will then implement those mappings as GraphQL resolvers. Generating GraphQL resolvers is difficult even for a standard-sized dataset which typically contains more than a handful tables and hundreds of properties. This situation is even worse if the underlying dataset evolves, considering that the corresponding resolvers have to be updated as well.

In a recent paper [19] we proposed the use of the mapping translation concept to facilitate the generation of GraphQL resolvers. We propose specifying mappings in R2RML, which is a well-defined and formalised mapping language, and apply mapping translation to generate automatically the corresponding GraphQL resolvers in different programming languages. Our intuition is that following this approach, GraphQL resolver will be easier to maintain, as they are declarative and independent from any programming language.

3.3. Providing access to semi-structured data.

Semi-structured data formats are one of the most widely used formats to publish data on the Web. Although existing mapping languages provide support for this type of data sources, existing engines are mostly focused on the generation (materialisation) of RDF-based Knowledge graphs, with only a few proposals (e.g. xR2RML [8]) focused on the application of query-translation techniques (virtualisation) over such types of data sources.

In the specific case of spreadsheets (CSV), providing access to this format is difficult for two main reasons: (i) CSV does not provide its own query language, (ii) there are some transformations that are commonly needed when treating data available in CSVs. For solving the first issue, query translation techniques have been applied over such data format by considering a CSV file as a single table that can be loaded in an RDB. For the second issue, some extensions of well-known mapping languages (RML together with the Function Ontology [20]) and annotations following the CSVW specification [21] can be used.

Morph-CSV[6] applies the concept of mapping translation for enabling an efficient access to CSV from SPARQL. It exploits the information of CSVW annotations and RML+FnO mappings to create an enriched RDB representation of the CSV files together with the corresponding R2RML mappings, enabling the use of existing query translation (SPARQL-to-SQL) techniques implemented in R2RML-compliant OBDA engines. The main reason for using two approaches in this case is that individually they are not able to resolve all the CSV challenges identified to enable a efficient query-translation process over the data. When we started this work we noticed that there are enough proposals for dealing with the heterogeneity of the CSV files (CSVW together with RML+FnO) and the SPARQL-to-SQL techniques and optimisation have been studied and implemented for a decade. Finally, we imposed one requirement: our approach should use as much as possible existing resources and not introduce yet another mapping language nor require a new SPARQL-to-SQL implementation. The result is a solution that, considering mappings and annotations that deal with the heterogeneity of the CSV files, builds an enriched RDB instance that represents the original data and translates the RML+FnO input mapping into the corresponding R2RML mapping so that it can take advantage of existing SPARQL-to-SQL optimisations.

3.4. Understanding the semantics of mapping languages

To the best of our knowledge, there has not been yet any formal study of the relationship between R2RML and the Direct Mapping recommendations, and among the many different mapping languages that have arisen recently, as pointed out in Section 1.

For the first case (R2RML and Direct Mapping), intuitively we may consider the Direct Mapping is a subset of R2RML, given the expressive power provided by the latter. However, it would be interesting to know how expressive Direct Mapping may be in case that

[4] https://github.com/facebook/dataloader
[5] https://join-monster.readthedocs.io/en/latest/
[6] https://github.com/oeg-upm/morph-csv

views are generated for the underlying data sources, for instance. Our intuition is that given the possibility of creating a database view from an existing database, there exists a fragment of R2RML that can be translated into Direct Mapping, such that the application of Direct Mapping over the view generates equivalent results as the application of R2RML mappings over the original database. Finding such fragment brings a practical implication because it would lower down the barrier for transforming data into RDF and enable people to use Direct Mapping engines, which are in general easier to use than R2RML engines for those people who are used to manage databases.

Similarly, this analysis may be extended to other combinations of mapping languages, so as to allow mapping translations among them that would allow exploiting the specific characteristics of each associated implementation, as well as describing formally their semantics, especially for those cases where no formal specification of the semantics has been provided yet.

Ontop [4] is an OBDA system that comes with both data and query translation techniques. Ontop translates R2RML mappings into its own mapping called "OBDA mappings". These mappings are represented as datalog rules, allowing the formalisation and semantic optimisation techniques to be performed, and generating a more efficient SQL queries (e.g., self-join elimination) that can be evaluated in less time by the underlying databases.

4. Conclusions and Practical Implications

In this vision paper, we have discussed the concept of mapping translation, which had not been addressed before in the literature. We have shown how this concept has been actually implemented in some existing approaches addressing the readability and maintenance of mappings, the generation of programming code to provide access to heterogeneous data sources, or the enrichment of original data sources, among others.

We think that this concept needs to be explored further, and this would allow a new range of OBDA approaches that may be part of a new OBDA generation, as claimed in the title of this paper. In our opinion, the OBDA community should see this variety of mapping languages not only as challenges (e.g., interoperability) but also, and mainly, as an opportunity for further research and development in this area, to address the need to cover more types of data sources while taking advantage of all the work that has been done in advanced aspects like query translation. Providing mapping translator services across mapping languages would bring further benefits and increase the availability of ontology-based data for its exploitation by search engines and query answering systems at Web scale.

Acknowledgements The work presented in this paper is supported by the Spanish Ministerio de Economía, Industria y Competitividad and EU FEDER funds under the DATOS 4.0: RETOS Y SOLUCIONES - UPM Spanish national project (TIN2016-78011-C4-4-R) and by an FPI grant (BES-2017-082511).

References

[1] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini and R. Rosati, Linking data to ontologies, in: *Journal on data semantics X*, Springer, 2008, pp. 133–173.

[2] G. Wiederhold, Mediators in the architecture of future information systems, *Computer* **25**(3) (1992), 38–49.

[3] F. Priyatna, O. Corcho and J. Sequeda, Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph, in: *Proceedings of the 23rd international conference on World wide web*, ACM, 2014, pp. 479–490.

[4] M. Rodriguez-Muro and M. Rezk, Efficient SPARQL-to-SQL with R2RML mappings, *Journal of Web Semantics* **33** (2015), 141–169.

[5] M. Arenas, A. Bertails, E. PrudâĂŹhommeaux and J. Sequeda, A Direct Mapping of Relational Data to RDF, W3C Recommendation 27 September 2012, 2013.

[6] S. Das, S. Sundara and R. Cyganiak, R2RML: RDB to RDF Mapping Language, Accessed: 2018-12-07.

[7] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens and R. Van de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data., in: *LDOW*, 2014.

[8] F. Michel, L. Djimenou, C. Faron-Zucker and J. Montagnat, Translation of relational and non-relational databases into RDF with xR2RML, in: *11th International Confenrence on Web Information Systems and Technologies (WEBIST'15)*, 2015, pp. 443–454.

[9] J. Slepicka, C. Yin, P.A. Szekely and C.A. Knoblock, KR2RML: An Alternative Interpretation of R2RML for Heterogenous Sources., in: *COLD*, 2015.

[10] A. Chortaras and G. Stamou, D2RML: Integrating heterogeneous data and web services into custom RDF graphs, *Proceedings of the LDOW. CEUR, ceur-ws. org* **2073** (2018).

[11] M. Lefrançois, A. Zimmermann and N. Bakerally, A SPARQL extension for generating RDF from heterogeneous formats, in: *European Semantic Web Conference*, Springer, 2017, pp. 35–50.

[12] D. Chaves-Fraga, F. Priyatna, I. Perez-Santana and O. Corcho, Virtual Statistics Knowledge Graph Generation from CSV files, in: *Emerging Topics in Semantic Technologies: ISWC 2018 Satellite Events*, Studies on the Semantic Web, Vol. 36, IOS Press, 2018, pp. 235–244.

[13] P. Heyvaert, B. De Meester, A. Dimou and R. Verborgh, Declarative Rules for Linked Data Generation at your Fingertips!, in: *Proceedings of the 15th ESWC: Posters and Demos*, 2018.

[14] J.F. Sequeda, M. Arenas and D.P. Miranker, On directly mapping relational databases to RDF and OWL, in: *Proceedings of the 21st international conference on World Wide Web*, ACM, 2012, pp. 649–658.

[15] O. Hartig, Foundations of RDF* and SPARQL*:(An Alternative Approach to Statement-Level Metadata in RDF), in: *AMW 2017 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017.*, Vol. 1912, Juan Reutter, Divesh Srivastava, 2017.

[16] P. Heyvaert, A. Dimou, A.-L. Herregodts, R. Verborgh, D. Schuurman, E. Mannens and R. Van de Walle, RMLEditor: a graph-based mapping editor for linked data mappings, in: *European Semantic Web Conference*, Springer, 2016, pp. 709–723.

[17] R.T. Fielding and R.N. Taylor, *Architectural styles and the design of network-based software architectures*, Vol. 7, University of California, Irvine Doctoral dissertation, 2000.

[18] Facebook, Inc., GraphQL, 2018, Accessed: 2018-12-07.

[19] F. Priyatna, D. Chaves-Fraga, A. Alobaid and O. Corcho, morph-GraphQL: GraphQL Resolvers Generation from R2RML Mappings., in: *SEKE*, 2019.

[20] B. De Meester, W. Maroy, A. Dimou, R. Verborgh and E. Mannens, Declarative data transformations for Linked Data generation: the case of DBpedia, in: *European Semantic Web Conference*, Springer, 2017, pp. 33–48.

[21] J. Tennison, G. Kellogg and I. Herman, Model for tabular data and metadata on the web. W3C recommendation, *World Wide Web Consortium (W3C)* (2015).

[22] P. Vassiliadis, A survey of extract–transform–load technology, *International Journal of Data Warehousing and Mining (IJDWM)* **5**(3) (2009), 1–27.

[23] T. Berners-Lee, J. Hendler, O. Lassila et al., The semantic web, *Scientific american* **284**(5) (2001), 28–37.

[24] M. Hert, G. Reif and H.C. Gall, A comparison of RDB-to-RDF mapping languages, in: *Proceedings of the 7th International Conference on Semantic Systems*, ACM, 2011, pp. 25–32.