# Ontology-Mediated Queries for NOSQL Databases
## (Extended Abstract)

Marie-Laure Mugnier[1], Marie-Christine Rousset[2], and Federico Ulliana[1]

[1] University of Montpellier, LIRMM, INRIA
[2] University Grenoble Alpes, LIG, IUF

## 1 Introduction

Ontology-based data access (OBDA) is a well-established paradigm for querying incomplete datasources while taking into account knowledge provided by a domain ontology [PLC+08]. Today, the main applications of OBDA can be found in data integration as well as in querying the Semantic Web. So far, OBDA has been studied for relational structures and deployed on top of relational databases, and then transposed to RDF. The database queries used in OBDA are (unions of) conjunctive *relational* queries (or basic graph pattern queries), while the ontologies are specified in either a description logic (e.g., the lightweight DL-Lite [CGL+07][KLT+10], or the expressive Horn-$\mathcal{SHIQ}$ [EOS+12]), or a rule-based fragment of first-order logic (e.g., Datalog$\pm$ [CGP12] and existential rules [BLMS11]). Decidability and complexity of ontology-mediated query answering within this framework have been extensively studied and many algorithms have been designed and implemented.

Whether this paradigm can be used in conjunction with other kinds of query languages is a still an open question. The naive way to deal with non relational datasources is to define mappings for translating them into relational structures, and then use the classic OBDA framework as it is. However, this approach would induce a significant performance degrade as it would add a step for converting the data using the mappings and, most importantly, it would make impossible to take advantage of the low level query optimizations provided by native systems. This can be particularly acute for NOSQL systems, like *key-value (KV) stores*, that have been specifically designed to scale when dealing with very large collections of data.

Our goal is to study ontology-based data access directly on top of NOSQL systems. The term NOSQL (NotOnly SQL) defines a broad collection of languages. KV stores are NOSQL systems adopting the data model of *KV records* (also called JSON records). These records are processed on distributed systems, but also increasingly exchanged on the Web thereby replacing semistructured XML data and many RDF formats (see JSON-LD [SLK+04]). KV records are non-first normal forms where values are not only atomic (in contrast with relational databases) and nesting is possible [AHV95].

This extended abstract summarizes our work on OBDA for KV stores [MRU16]. We introduce NO-RL, a rule language to express lightweight ontologies on top of KV records, and provide first (un)decidability and data complexity results for ontology-mediated query answering in our setting. The NO-RL rule language operates directly on KV records and can be compared to existing languages designed for reasoning on nested structures like F-Logic [KLW95,Kif05], Elog [BFGK01,BCG+03,GK04] and Active XML [ABM04]. However, up to our knowledge, none of this existing work follows an OBDA approach.

## 2 Data and Queries

Our first contribution is a formalization of the data model of KV stores (sets of KV records) and the associated basic queries. In this abstract, we will just illustrate these notions by an example. The KV record below contains ten different keys: most of them are associated with atomic values, except `phone` associated with a record, `professor` and `course` associated with sequences (of records and of sequences, respectively) and `director` associated with an unspecified value $null$.

{ `department` : *"CS"*,
`professor` : [
{ `name` : *"Alice"*, `reachable` : *"yes"*, `boss` : *"Charles"* }
{ `name` : *"Bob"*, `phone` : {`office` : *"5-256"*}, `speciality` : *"Logics"*} ]
`course` : [[*"C123"*, *"Java"*] , [*"C310"*, *"C++"*]]
`director` : *null* }

KV stores feature a limited set of low-level highly parallelizable queries based on *paths*, and leave other functionalities (such as joins between records) for the application accessing the data. An example of query retrieving the names of professors is get(`professor.name`). This expression returns the values "*Alice*" and "*Bob*" once evaluated on the example record. Another feature supported by KV stores is the possibility to check structural properties of a record before selecting some content. Thus, we further look at queries with a check() construct like the expression check(`director`).get(`professor.name`). The get part of the query is evaluated at the only condition that the key `director` also exists in the record. The check() construct is particularly useful when dealing with incomplete information expressed by null values, like for the key `director` in the example.

## 3 The NO-RL Rule Language

To define rules on KV records, we consider atoms of the form $\kappa(x)$, where $\kappa$ is a (finite) sequence of keys, called a key-path, and $x$ a variable. NO-RL rules can be of two kinds: $\kappa'(x) \longrightarrow \kappa(x)$ or $\kappa'(x) \longrightarrow \exists y \kappa(y)$, as illustrated below:

1. "*every phone number is a contact (any value for the key phone is a value for the key contact)* "
$\texttt{phone}(x) \longrightarrow \texttt{contact}(x)$
2. "*if there is a reachable field then there is a phone number*"
$\texttt{reachable}(x) \longrightarrow \exists y.\, \texttt{phone}(y)$
3. "*if a professor has a boss, then this last one is a director of the department*"
$\texttt{professor.boss}(x) \longrightarrow \texttt{director}(x)$
4. "*the department defines a teaching specialty for all of its professors*"
$\texttt{department}(x) \longrightarrow \texttt{professor.specialty}(x)$

We define three rule fragments: NO-RL(1), composed of rules that employ only key-paths of length one (like Rules 1 and 2 in the example), NO-RL(2) and NO-RL(3), which both extend NO-RL(1) by allowing respectively for rule bodies of arbitrary length (like Rule 3) and for rule heads of arbitrary length (like Rule 4).

The following record is obtained from the previous one by application of the four above rules until fixpoint (process called "saturation").

{ department : *"CS"*,
  professor : [
  {name : *"Alice"*, reachable : *"yes"*, boss : *"Charles"*, **phone :** *null*, **specialty :** ***"CS"*** },
  {name : *"Bob"*, phone : {office : *"5-256"*},
  **contact :** {**office :** ***"5-256"***}, specialty : [***"CS"***, *"Logics"*]} ]
  course : [[*"C123"*, *"Java"*], [*"C310"*, *"C++"*]]
  director : ***"Charles"*** }

Rule application inserts a path into the existing record, which requires to merge information (e.g., the application of Rule 4 inserts the path professor.specialty : *"CS"*). We show that the inference operator associated with rule application is confluent (up to a notion of record equality), hence the order in which rules are applied is irrelevant. Finally, a value is an answer to a query $Q$ over a knowledge base $(I, \Sigma)$, where $I$ is a set of records and $\Sigma$ a set of NO-RL rules, if it is an answer to $Q$ on a record inferred from a record in $I$ by the rules in $\Sigma$.

## 4 Decidability and Complexity Results

Despite the apparent simplicity of the NO-RL language, answering check/get queries under NO-RL rules is generally undecidable (which we prove by reduction from the word problem in a semi-Thue system). Intuitively, this is due to the unrestricted interaction between NO-RL(2) and NO-RL(3) rules. In particular, decidability can be recovered by restricting the rule language to NO-RL(2) or NO-RL(3) only.

A soon as a single NO-RL(3) rule is involved, saturation may be infinite. If we consider NO-RL(2) rules, saturation is always finite since these rules do not increase the depth of records. However, with a single NO-RL(1) rule already, saturation can be exponential in the size of the input record. This observation advocates for query reformulation techniques in the spirit of the OBDA paradigm. We define a sound and complete reformulation technique (i.e., a value is an answer to $Q$ over $(I, \Sigma)$ iff if it is an answer to a $\Sigma$-reformulation of $Q$ over $I$). Dually to saturation with NO-RL(2) rules, query reformulation with NO-RL(3) rules is always finite since these rules do not increase the depth of records. It follows that the data complexity of (the decision problem associated with) query answering with NO-RL(3) rules is the same as basic query answering on KV stores, i.e., it is in the low complexity class $AC^0$. Query reformulation can be infinite with NO-RL(2) rules, however, given a record of depth $d$, only reformulations of depth less or equal to $d$ are useful. There is an exponential number of useful reformulations, nevertheless each of them can be produced with a polynomial number of steps ($(d + 1) \times |\Sigma|$), hence query answering over NO-RL(2) rules is in NP (for both data and combined complexities —the exact lower bounds being open).

Future work includes further clarifying query answering complexity, identifying decidable / tractable cases mixing NO-RL(2) and NO-RL(3) rules, as well as designing algorithms that would exploit the parallelization features of KV stores.

**Acknowledgments.** This work has been partially supported by projects PAGODA (ANR-12-JS02-007-01) and Labex PERSYVAL-Lab (ANR-11-LABX-0025-01).

## References


[ABM04] Serge Abiteboul, Omar Benjelloun, and Tova Milo. Positive Active XML. In *PODS*, 2004.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu, editors. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1995.

[BCG+03] Robert Baumgartner, M. Ceresna, Georg Gottlob, M Herzog, and V. Zigo. Web information acquisition with lixto suite: a demonstration. In *ICDE*, 2003.

[BFGK01] Robert Baumgartner, Sergio Flesca, Georg Gottlob, and Christoph Koch. Visual web information extraction with lixto. In *VLDB*, 2001.

[BLMS11] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. On Rules with Existential Variables: Walking the Decidability Line. *Artificial Intelligence*, 175(9-10):1620–1654, 2011.

[CGL+07] 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. Autom. Reasoning*, 39(3):385–429, 2007.

[CGP12] Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence*, 193:87–128, 2012.

[EOS+12] T. Eiter, M. Ortiz, M. Simkus, T.-K. Tran, and G. Xiao. Query Rewriting for Horn-SHIQ Plus Rules. In *AAAI.*, 2012.

[GK04] Georg Gottlob and Christoph Koch. Monadic datalog and the expressive power of languages for web information extraction. *J. ACM*, 10, 2004.

[Kif05] Mickael Kifer. Rules and Ontologies in F-logic. In *RW*, 2005.

[KLT+10] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. The Combined Approach to Query Answering in DL-Lite. In *KR*, 2010.

[KLW95] Mickael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 1995.

[MRU16] Marie-Laure Mugnier, Marie-Christine Rousset, and Federico Ulliana. Ontology-mediated queries for nosql databases. In *AAAI*, 2016.

[PLC+08] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[SLK+04] Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindstrom. JSON-LD 1.0, A JSON-based Serialization for Linked Data. Technical Report http://www.w3.org/TR/json-ld/, W3C, 2004.