InteGraal: a Tool for Data-Integration and Reasoning on Heterogeneous and Federated Sources

Jean-François Baget, Pierre Bisquert, Michel Leclère, Marie-Laure Mugnier, Guillaume Pérution-Kihli, Florent Tornil, Federico Ulliana

InteGraal: a Tool for Data-Integration and Reasoning on Heterogeneous and Federated Sources

Jean-François Baget, Pierre Bisquert, Michel Leclère, Marie-Laure Mugnier,
Guillaume Pérution-Kihli, Florent Tornil, Federico Ulliana
first.last@inria.fr
LIRMM, Inria, Univ Montpellier, CNRS, INRAE
Montpellier, France

Abstract

We propose to demonstrate InteGraal, a tool for reasoning over heterogeneous and federated data sources. InteGraal is a highly modular tool constituted by two main components. The first is the data-integration layer which allows the users to build a federated factbase over a collection of sources. The second is the automated reasoning layer, which provides powerful means for the declarative exploitation of data through the expressive formalism of *existential rules*. This demonstration proposes to showcase the use of the tool in use-cases of data exploitation as well as to present its architecture and its dedicated query answering mechanisms.

1 The Data-Variety Issue and Knowledge-Based Data-Management

Data is today generated at vertiginous rates and across many different application domains. This richness opens for many opportunities and organizations are constantly seeking for new insights from their data to support their next decision. Yet, the multiplication of models, languages, platforms, as well as the number of independent sources which provide data, makes its exploitation increasingly challenging. Concretely, when dealing with data which is *heterogeneous* (i.e., diverse) and *federated* (i.e., crosslinked), basic tasks for feeding analytic tools such as running explorative queries or building a training set become the bottleneck as users are often unable to formulate the "right" queries to the sources.

Example 1. *Consider the scenario depicted by Figure 1 (blue part). A data-analyst wants to retrieve the list of clients of a given company starting from two heterogeneous sources. The first is a relational database which contains purchases made by clients. The second is a JSON database which contains the description of products that have been sold. In this case, to retrieve the desired information, a data-analyst will typically have to* (*i*) *formulate and issue SQL and JSON queries* manually *to two separated systems (which requires to first understand both data models)* (*ii*) *understand and possibly enrich the data coming from the sources by relying on expert knowledge and then* (*iii*) *write an external script to cross the resulting data.*

A principled solution to this issue is to provide users a *unified high-level vision* of data, which simplifies the writing of queries and frees the users from mastering the variability

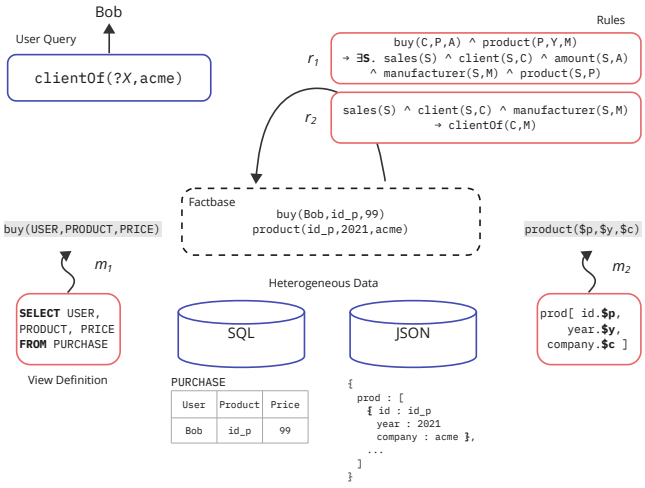

Figure 1. Querying Heterogeneous and Federated Data

of data by hiding the specificities of each data source behind a common vocabulary. The unification of data is achieved through a *data-integration* system, where heterogeneous data in the sources is transformed and linked to a global integration schema that users can query directly. To cope with the heterogeneity, however, all data must first be brought to a common model; this is achieved via the definition of *views*.

In this work, we consider data-integration systems where the notion of global integration schema is replaced by a more sophisticated knowledge layer, which not only acts as a unifying layer but also enables reasoning. More specifically, instead of resulting in a (possibly virtual) global database, the integration results in a *knowledge base*. A knowledge base is constituted of a set of facts (called factbase) made of *factual knowledge* on the application-domain (here, resulting from the integration of heterogeneous sources) as well as a set of rules (called a rulebase) containing *universal knowledge* on the application-domain. Rules are a powerful declarative paradigm for representing knowledge and express recursive queries which can serve to many tasks in data processing and governance. More precisely, we are focusing on knowledge bases built around sets of *existential rules* [8, 14] (a.k.a. Datalog$^{\pm}$ and Tuple-Generating-Dependencies). The interest

in Existential rules stems from the fact that they are ubiquitous in many domains because they enable to implement fundamental mechanisms needed to process data and knowledge. An essential feature of these rules is that they allow to create entities that may not appear in the factbase, hence to reason on open domains (a feature also called *value invention* [21], see e.g. Rule r_1 in Example 1). As a matter of fact, existential rules can be used as a uniform language to express relational GLAV mappings [20, 25] and expressive data dependencies [9], as well as domain knowledge in the form of semantic constraints and ontologies [28]. In particular, they generalize many description logics used to reason on data, a.k.a. Horn Description-Logics [28]. Finally, they can be used to express rich recursive queries and generalize the Datalog language [1]. Overall, the combined use of existential rules with relational views of heterogeneous data provides great expressivity and integration power.

Example 1 (Continued). *Figure 1 illustrates the use of* views *and* rules *to exploit heterogeneous and federated data (red part). View definitions m_1 and m_2 export informations related to purchases and products. Both issue a* native *query, either on the SQL or JSON source. The native query results then instantiate the ternary predicates* `buy` *and* `product`. *Rules r_1 and r_2 provide high-level relations that the users can easily query. Rule r_1 defines a high-level concept of* `sale` *through a reification design pattern which groups all relevant information coming from* `buy` *and* `product` *(notably, its* `manufacturer`*) by using an* existentially quantified variable *S (which then represents a sale in itself). On top of this, Rule r_2 defines the relation* `clientOf` *which is instantiated whenever a client buys a product of a certain manufacturer. Overall, the combined use of m_1, m_2, r_1, r_2 allows the system to provide answers to the user query retrieving the list of clients of the given company.*

1.1 Related Work

The combined use of data integration and knowledge bases leads to the notion of *Knowledge-based Data-Management* (KBDM) system that we shall define in the next section. These kinds of systems are at the crossroads of several lines of work: *federated query answering* (FQA), *ontology-based data-access* (OBDA), and *ontology-mediated query answering* (OMQA). Federated Query Answering is the problem of providing users with the ability of querying multiple heterogeneous data sources (database systems, structured files, web services) under a uniform data model and query language by relying on virtualization techniques which leaves the data at the level of sources. This problem has been studied in different contexts and we refer the reader to [18] for a recent survey on the topic. Ontology-based data-access (OBDA) is a data-integration paradigm where the global integration schema is an ontology [28]. This paradigm distinguishes itself for the use of ontologies supporting virtualization, notably DL-Lite [16]. Ontology-mediated query answering is the problem of answering queries in the presence of a logical theory modelling domain knowledge [11]. The problem has been vastly studied for two families of languages: from one side Description Logics [3] and from the other side existential rules [6, 14]. While many implementations exist, they all focus either only on FQA (e.g., [2, 22]), or OBDA [15] or OMQA [5, 17, 26]. The only system we are aware of which combines FQA and OBDA is Obi-Wan [13]. Obi-Wan considers however RDF as the data-integration model and RDF-Schema ontologies (which express very basic knowledge on classes and properties) on top of the Tatooine mediator [12]. More generally, the combination of data federation and reasoning is still an open field of research, as illustrated by the recent work of [19].

1.2 Novelty and Contributions.

This work proposes to demonstrate InteGraal: *a Java tool for integrating and querying heterogeneous and federated data through existential rules knowledge-bases.* InteGraal is an open source tool available at (gitlab.inria.fr/rules/integraal). The distinctive elements of the tool are the following.

1. **Framework and Languages.** InteGraal is the only system considering a framework for integrating and reasoning on data which englobes that of FQA, OBDA, and OMQA. Also, it provides support for a very expressive rule language, notably *existential rules*.
2. **Architecture.** InteGraal has a generic and extensible architecture. Instead of fixing a single query answering approach, it hosts multiple algorithmic strategies which can be combined in a *modular* way to cover a larger landscape of applicative scenarios.

InteGraal is a major revision of the Graal tool for reasoning with existential rules [5]. With respect to its predecessor InteGraal features a completely refactored core model and reasoning algorithms as well as data-integration facilities which were not present in Graal.

The remainder of the paper is dedicated to the formal definition of KBDM systems in Section 2 as well as a presentation of the architecture and features of InteGraal in Section 3. Section 4 outlines the demonstrating scenario.

2 Answering Queries in Knowledge-Based Data-Management Systems

In this section, we formally define KBDM systems as well as the semantics of *query answering*, which is the ultimate task for InteGraal. As already said, in a KBDM system, heterogeneous and federated data is integrated through a knowledge base (KB) that users can query. A KB is typically composed by a factbase (factual knowledge) and a rulebase (general knowledge, including ontologies). As such, the classic notion of KB is lifted up to account for factbases that are not directly given but *defined* from the federation of heterogeneous databases, thereby leading that of a KBDM system.

Formally, a KBDM system is a pair $K = (\mathcal{F}, \mathcal{R})$ where $\mathcal{F}$ is a *federated factbase* and $\mathcal{R}$ is a set of *existential rules* as follows:

- A federated factbase is a pair $\mathcal{F} = (S, M)$ constituted by a set of heterogeneous data *sources* $S = \{s_1, \ldots, s_k\}$ and a set of *relational view-definitions* M transforming (possibly not relational) data into a relational signature exploitable by rules. A view definition $m \in M$ for a source $s \in S$ is of the form
$$q_{\text{native}}(\bar{x}) \rightsquigarrow P(\bar{x})$$
where $q_{\text{native}}(\bar{x})$ is a query (or more generally, a function) on the native language[1] of source s. The evaluation of the native query yields a set of tuples obtained by affecting its answer variables $\bar{x}$ to constant values. Then, every answer tuple $\mathbf{t}$ is exported as a logical atom of the form $P(\mathbf{t})$, where P is a predicate seen as a view name.
- A set of existential rules $\mathcal{R}$ holds general knowledge on the domain of data, expressed as a logical theory. An existential rule is a first-order logic formula of the form
$$\forall \bar{x}\bar{y}.\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}.\psi(\bar{x}, \bar{z})$$
where $\phi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ are positive function-free conjunctions of atoms, called the *body* and *head* of the rule, respectively. All variables in the rule are universally quantified except those in $\bar{z}$ which are existentially quantified. Next, we simplify the notation by omitting universal quantifiers.

Abusing notation, we shall denote by $\mathcal{F}$ both the federated factbase and the resulting data federation instance, that is, the set of all $P(\mathbf{t})$, seen as a logical conjunction. Note that this federated data instance $\mathcal{F}$ can be *materialized* or *virtual*; this is discussed in Section 3.1.

The view definitions we consider can be assimilated to GAV (global-as-view) mappings as studied in data integration [25], with the important addition of not being restricted to a relational source. Then, our formalism naturally allows one to reach the expressivity of more powerful GLAV (global-local-as-view) mappings. This is achieved by composing a view definition $q_{\text{native}}(\bar{x}) \rightsquigarrow P(\bar{x})$ with an existential rule $P(\bar{x}) \rightarrow \exists \bar{z}.\psi(\bar{x}', \bar{z})$ (and $\bar{x}' \subseteq \bar{x}$). In this vision, a GLAV mapping is decomposed into low-level (view definition) and high-level (existential rule) assertions.

We are finally ready to define query answering. Simply put, answers to queries are logically *entailed* from the KB associated with the KBDM system. More specifically, a Boolean first-order query Q answers true on a KBDM system $K = (\mathcal{F}, \mathcal{R})$ when $\mathcal{F}, \mathcal{R} \models Q$, where $\models$ denotes standard logic entailment; and a tuple of constants $\mathbf{t}$ is an answer to a query $Q(\bar{x})$, with $\mathbf{t} = |\bar{x}|$, if $\mathcal{F}, \mathcal{R} \models Q[\bar{x}/\mathbf{t}]$, where $Q[\bar{x}/\mathbf{t}]$ is the Boolean query obtained from Q by substituting each variable in $\bar{x}$ by the corresponding constant in $\mathbf{t}$.

[1] For instance SQL, SPARQL, NoSQL, Web API, etc.

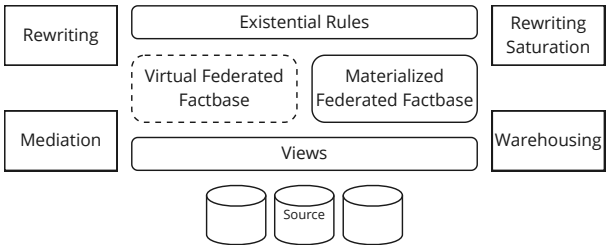

Figure 2. Data-Integration and Reasoning Strategies

3 InteGraal's Architecture and Algorithms for Knowledge-Based Data-Management

The landscape of applicative scenarios for KBDM system is *vast*. From one side, because of the types of data sources users may want to integrate. From the other side, because of the type of data processing and governance tasks aimed by users. This is a significant challenge, which makes that it is unlikely for a single approach to be able to tackle all cases. In light of this, InteGraal has been designed with genericity in mind thereby making the tool able to adapt to very different applicative scenarios. The term genericity here refers to two main aspects: *data heterogeneity* and *reasoning algorithms*.

Data Heterogeneity. InteGraal provides support for very heterogeneous data through a generic integration module which allows users to integrate multiple type of sources. Currently, this includes RDBMS (PostgreSQL, MySQL, HSQLDB), Triplestores (Virtuoso, RDF4J), Document-stores (MongoDB), and Web APIs. Most importantly, however, the tool can be *extended* so as to easily add novel types of data sources seamlessly with respect to the query answering algorithms. This module serves at establishing relational views over the heterogeneous sources as previously outlined and constitutes the first brick for handling heterogeneous and federated data.

Reasoning Algorithms. InteGraal provides a relevant collection of *algorithmic strategies* for storing, integrating, querying, and reasoning on data. All of these algorithms provide concrete implementations for high-level *services* which are offered by the tool (e.g., store, query, reason). The genericity of the tool also makes extending and calling new algorithmic strategies easy. Overall, the architecture modularity and extensibility paves the way to a modular combination of algorithmic solutions which best fit the considered applicative scenario.

3.1 Algorithmic Strategies

Answering queries over KBDM systems requires to solve two problems at once: data-integration (to account for views) and automated-reasoning (to account for rules). Unfortunately, the large variety of situations that can arise implies that it is impossible to settle on a one-rules-them-all approach.

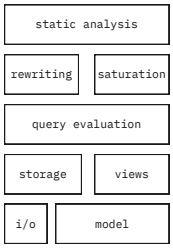

Figure 3. InteGraal Modules

Data-integration. The heterogeneity of data concerns many aspects. Beside the data model employed by the sources, data can be static or frequently updated as well as voluminous or of moderate size. Furthermore, sources may consist of platforms offering efficient query facilities as well as web services introducing undesirable latency. Depending on the data sources to integrate, InteGraal provides the possibility to choose between *materializing* the data resulting from a source in the federation or to keep this *virtual*. Materializing data follows the *warehousing* approach which aims at centralizing data into a single database. The net advantage of this method is that it gives control over the factbase storage and hence the query answering facilities. The approach can however be inconvenient when data is frequently updated or more efficiently queried via the source API. In this case, it may be preferable to keep the data virtual thereby following the *mediation* approach. With this method, data is left at the level of sources and, at query time, an access plan is built by using the native queries in the view definitions.

In a nutshell, InteGraal supports both the mediation and the warehousing approach for handling views and enables the use of proper combinations of these approaches.

Automated Reasoning. The dynamicity and the volume of data also play an important role when choosing reasoning algorithms. Two main reasoning approaches for query answering are saturation (also called forward-chaining) and query rewriting (also called backward-chaining).

Saturation consists at extending the factbase with the inferences produced by rules. Typically, saturation may be chosen when the database does not change. Once the saturation has been computed, the user queries can be directly evaluated on the saturated factbase. However, saturation is not always relevant: the size of the obtained database may be prohibitive, maintenance can be costly under updates, and finally the method applies only to federated factbases that are materialized.

Query rewriting, in contrast, consists at propagating the rules into the query thereby leaving the data untouched. With this, a user query is rewritten into a union of queries. The main inconvenient with this approach is that it can produce very large rewriting sets but it still remain interesting when this compensates for computing and maintaining a saturation. Because of its virtual nature, query rewriting can be used both in combination with warehousing and mediation approaches. In the first case, the rewriting set of queries is evaluated on the materialized federated factbase. In the second case, a rewriting is further translated into an access plan using native queries targeting the sources.

In a nutshell, InteGraal provides a large set of saturation and rewriting techniques for existential rules, and enables the use of proper combinations of these approaches.

3.2 Main Modules

InteGraal is an open source project consisting of 25K lines of Java code. Its main modules are the following (see Figure 3).

- `model`. This module contains all objects needed for representing KBDM systems and notably the knowledge base.
- `views`. This module provides facilities to the creation of relational views (or as we said, low level mappings) over heterogeneous data. As described before, for now, the tool supports views over an SQL, SPARQL, MongoDB and Web APIs as datasources.
- `storage`. This module provides facilities for storing *locally* materialized knowledge bases. Note that it complements the module presented before, which focus on querying data which sits in an autonomous system. More precisely it allows one to choose between several types of storage facilities for sets of logical atoms [7]. This includes a native InteGraal in-memory implementation of a graph database, as well as a local RDBMs or a local Triplestore system.
- `query-evaluation`. The module provides support for evaluating a conjunctive query over a federated factbase. InteGraal provides two alternative strategies for this task. The first is through a general backtrack algorithm; this can be used to query any federated factbase. The second is through a bridge with the Tatooine mediator [12] which is viable only for views that are supported by Tatooine (SQL, Triplestore, MongoDB).
- `saturation`. This module provide facilities for saturation based reasoning. It offers parametrizable algorithm allowing one to deploy different variants of the known *chase* algorithm, notably restricted, skolem and core chases [10]. Each of them offers a different tradeoffs between complexity and termination (see [27] for some details on these tradeoffs).
- `rewriting`. This module provide facilities for rewriting based reasoning. It offers a general algorithm for query rewriting as well as an optimized algorithm relying on rule compilation [23].
- `static-analysis`. This module provides tools for analyzing the KBDM system and pointing to terminating reasoning strategies [24].

- `i/o`. This module provides support for import and export of data from textual formats. For now the tool provides support for DLGP [4], RDF and CSV.

4 Demonstration Scenario

During the demonstration, we will show how to use our tool for two types of tasks. First, we focus on the construction of a KBDM system. More precisely, this includes the definition of views and rules for integrating and exploiting data. We will then look at the reasoning tasks and show how, in the presence of rules, the queries formulated by users can be addressed by saturation, query rewriting, as well as proper combinations of these techniques. We will in particular zoom on mediation techniques and show how queries are evaluated over federated databases by using the different approaches provided by our tool. The overall demonstration will outline an end-to-end process for the exploitation of heterogeneous and federated data. We will focus on data-integration scenarios inspired from use-cases stemming from agronomy where views and rules are used to support the creation of training sets at the support of machine learning tasks. These include the exploitation and reasoning on relational, RDF, JSON, and Web API sources. Details on our tool and use-cases will be available at gitlab.inria.fr/rules/integraal-examples.

Acknowledgments

We are grateful to Julien Cufi and Patrice Buche for their help setting up the demonstration scenario. This work was partially supported by the Inria-DFKI project R4Agri and the ANR project CQFD (ANR-18-CE23-0003).

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of Databases*. http://webdam.inria.fr/Alice/

[2] Rana Alotaibi, Bogdan Cautis, Alin Deutsch, Moustafa Latrache, Ioana Manolescu, and Yifei Yang. 2020. ESTOCADA: towards scalable polystore systems. *Proceedings of the VLDB Endowment* (2020).

[3] Franz Baader, Ian Horrocks, and Ulrike Sattler. 2004. *Description logics*. Springer.

[4] Jean-François Baget, Alain Gutierrez, Michel Leclère, Marie-Laure Mugnier, Swan Rocher, and Clément Sipieter. 2015. Datalog+, RuleML and OWL 2: Formats and Translations for Existential Rules. In *Proceedings of the RuleML 2015 Challenge*.

[5] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, Swan Rocher, and Clément Sipieter. 2015. Graal: A Toolkit for Query Answering with Existential Rules. In *9th International Symposium, RuleML 2015, Berlin, Germany, August 2-5, 2015, Proceedings (Lecture Notes in Computer Science)*.

[6] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. 2011. On rules with existential variables: Walking the decidability line. *Artif. Intell.* (2011).

[7] Jean-François Baget, Madalina Croitoru, and Bruno Paiva Lima da Silva. 2013. Alaska for ontology based data access. In *The Semantic Web: ESWC 2013 Satellite Events: ESWC 2013 Satellite Events, Montpellier, France, May 26-30, 2013, Revised Selected Papers 10*. Springer.

[8] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat. 2009. Extending Decidable Cases for Rules with Existential Variables. In *IJCAI*.

[9] Catriel Beeri and Moshe Y. Vardi. 1984. A Proof Procedure for Data Dependencies. *J. ACM* (1984).

[10] Michael Benedikt, George Konstantinidis, Giansalvatore Mecca, Boris Motik, Paolo Papotti, Donatello Santoro, and Efthymia Tsamoura. 2017. Benchmarking the Chase. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*.

[11] Meghyn Bienvenu. 2016. Ontology-mediated query answering: Harnessing knowledge to get more from data. In *IJCAI 2016*.

[12] Raphaël Bonaque, Tien Duc Cao, Bogdan Cautis, François Goasdoué, Javier Letelier, Ioana Manolescu, Oscar Mendoza, Swen Ribeiro, Xavier Tannier, and Michaël Thomazo. 2016. Mixed-instance querying: a lightweight integration architecture for data journalism. In *VLDB*.

[13] Maxime Buron, François Goasdoué, Ioana Manolescu, and Marie-Laure Mugnier. 2020. Obi-Wan: ontology-based RDF integration of heterogeneous data. *Proceedings of the VLDB Endowment* (2020).

[14] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2009. A General Datalog-Based Framework for Tractable Query Answering over Ontologies. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009*.

[15] 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* (2017).

[16] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. 2007. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *Journal of Automated reasoning* (2007).

[17] David Carral, Irina Dragoste, Larry González, Ceriel Jacobs, Markus Krötzsch, and Jacopo Urbani. 2019. Vlog: A rule engine for knowledge graphs. In *International Semantic Web Conference*. Springer.

[18] Zhenzhen Gu, Francesco Corcoglioniti, Davide Lanti, Alessandro Mosca, Guohui Xiao, Jing Xiong, and Diego Calvanese. 2022. A systematic overview of data federation systems. *Semantic Web* (2022).

[19] Zhenzhen Gu, Davide Lanti, Alessandro Mosca, Guohui Xiao, Jing Xiong, and Diego Calvanese. 2022. Ontology-based data federation. In *35th International Workshop on Description Logics, DL 2022*.

[20] Alon Y Halevy. 2001. Answering queries using views: A survey. *The VLDB Journal* (2001).

[21] Richard Hull and Masatoshi Yoshikawa. 1990. ILOG: Declarative Creation and Manipulation of Object Identifiers. In *VLDB*.

[22] Boyan Kolev, Carlyna Bondiombouy, Patrick Valduriez, Ricardo Jiménez-Peris, Raquel Pau, and José Pereira. 2016. The cloudmdsql multistore system. In *SIGMOD '16*.

[23] Mélanie König, Michel Leclère, Marie-Laure Mugnier, and Michaël Thomazo. 2015. Sound, complete and minimal UCQ-rewriting for existential rules. *Semantic Web* (2015).

[24] Michel Leclère, Marie-Laure Mugnier, and Swan Rocher. 2013. Kiabora: an analyzer of existential rule bases. In *Web Reasoning and Rule Systems: 7th International Conference, RR 2013, Mannheim, Germany, July 27-29, 2013. Proceedings 7*. Springer.

[25] Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*.

[26] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. 2015. RDFox: A highly-scalable RDF store. In *International Semantic Web Conference*. Springer.

[27] Adrian Onet. 2013. The Chase Procedure and its Applications in Data Exchange. In *Data Exchange, Integration, and Streams*, Phokion G. Kolaitis, Maurizio Lenzerini, and Nicole Schweikardt (Eds.). Dagstuhl Follow-Ups, Vol. 5. Schloss Dagstuhl, 1–37.

[28] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev. 2018. Ontology-based data access: A survey. International Joint Conferences on Artificial Intelligence.