Ontology-Based Data Access and Integration

Diego Calvanese[1], Giuseppe De Giacomo[2], Domenico Lembo[2], Maurizio Lenzerini[2], and Riccardo Rosati[2]
[1]Research Centre for Knowledge and Data (KRDB), Free University of Bozen-Bolzano, Bolzano, Italy
[2]Dip. di Ingegneria Informatica Automatica e Gestionale Antonio Ruberti, Sapienza Università di Roma, Rome, Italy

Definition

An *ontology-based data integration* (OBDI) system is an information management system consisting of three components: an ontology, a set of data sources, and the mapping between the two. The ontology is a conceptual, formal description of the domain of interest to a given organization (or a community of users), expressed in terms of relevant concepts, attributes of concepts, relationships between concepts, and logical assertions characterizing the domain knowledge. The data sources are the repositories accessible by the organization where data concerning the domain are stored. In the general case, such repositories are numerous, heterogeneous, each one managed and maintained independently from the others. The mapping is a precise specification of the correspondence between the data contained in the data sources and the elements of the ontology. The main purpose of an OBDI system is to allow information consumers to query the data using the elements in the ontology as predicates.

In the special case where the organization manages a single data source, the term *ontology-based data access* (ODBA) system is used.

Historical Background

The notions of ODBA and ODBI were introduced in [3, 14] and originated from several disciplines, in particular, information integration, knowledge representation and reasoning, and incomplete and deductive databases.

OBDI can be seen as a sophisticated form of information integration [11], where the usual global schema is replaced by an ontology describing the domain of interest. The main difference between OBDI and traditional data integration is that in the OBDI approach, the integrated view that the system provides to information consumers is not merely a data structure accommodating the various data at the sources but a semantically rich description of the relevant concepts in the domain of interest, as well as the relationships between such concepts. Also, the distinction between the ontology and the data sources reflects the separation between the conceptual level, the one presented to the client, and the logical/physical level of the information system, the one stored in the sources, with the

© Springer Science+Business Media LLC 2017
L. Liu, M.T. Özsu (eds.), *Encyclopedia of Database Systems*,
DOI 10.1007/978-1-4899-7993-3_80667-1

mapping acting as the reconciling structure between the two levels.

The central notion of OBDI is therefore the ontology, and reasoning over the ontology is at the basis of all the tasks that an OBDI system has to carry out. In particular, the axioms of the ontology allow one to derive new facts from the source data, and these inferred facts greatly influence the set of answers that the system should compute during query processing. In the last decades, research on ontology languages and ontology inferencing has been very active in the area of knowledge representation and reasoning. Description logics [1] (DLs) are widely recognized as appropriate logics for expressing ontologies and are at the basis of the W3C standard ontology language OWL. (http://www.w3.org/TR/owl2-overview/) These logics permit the specification of a domain by providing the definition of classes and by structuring the knowledge about the classes using a rich set of logical operators. They are decidable fragments of mathematical logic, resulting from extensive investigations on the trade-off between expressive power of knowledge representation languages and computational complexity of reasoning tasks. Indeed, the constructs appearing in the DLs used in OBDI are carefully chosen taking into account such a trade-off [3, 12]. We observe that many research papers on reasoning in DLs for ODBA and OBDI actually concentrate on query answering in a setting where the data are assumed to reside in ad hoc repositories (often in RDF format), rather than in independent sources. Since such ad hoc repositories are designed for storing the instances of the elements of the ontology, mappings are not present in this simplified setting, which is often called *ontology-based query answering* (OBQA).

As we said before, the axioms in the ontology can be seen as semantic rules that are used to complete the knowledge given by the raw facts determined by the data in the sources. In this sense, the source data of an OBDI system can be seen as an incomplete database, and query answering can be seen as the process of computing the answers logically deriving from the combination of such incomplete knowledge and the ontology axioms. Therefore, at least conceptually, there is a connection between OBDI and the two areas of incomplete information [8] and deductive databases [4]. The new aspect of OBDI is related to the kind of incomplete knowledge represented in the ontology, which differs both from the formalisms typically used in databases under incomplete information (e.g., Codd tables) and from the rules expressible in deductive database languages (e.g., logic programming rules).

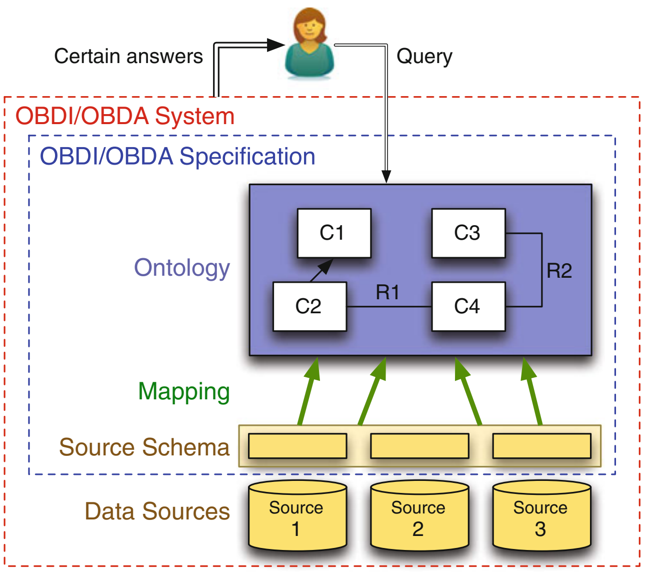
Scientific Fundamentals

We deal here with the semantic and computational aspects related to the use of an ontology and of mappings to data sources in query processing in OBDI. Thus, we do not address the problems that specifically pertain to accessing and querying *multiple*, *heterogeneous* data sources in an integrated way, such as wrapping non-relational data, distributed query evaluation, and entity resolution. For those problems, we refer to the "Information Integration" entry. The distinction between OBDI and OBDA is therefore not significant for our treatment of the scientific fundamentals, and we refer to OBDA only. Coherently, we assume to deal with a single relational data source, whose schema might represent the federated schema of multiple, heterogeneous data sources, wrapped as relational databases.

OBDA Framework

We distinguish between the specification of an OBDA system and the OBDA system itself (cf. Fig. 1). An *OBDA specification* $\mathcal{J}$ determines the intensional level of the system and is expressed as a triple $\langle \mathcal{O}, S, \mathcal{M} \rangle$, where $\mathcal{O}$ is an ontology, S is the schema of the data source, and $\mathcal{M}$ is the mapping between S and $\mathcal{O}$. Specifically, $\mathcal{M}$ consists of a set of mapping assertions, each one relating a query over the source schema to a query over the ontology. An *OBDA system* $\langle \mathcal{J}, D \rangle$ is obtained by adding to $\mathcal{J}$ an extensional level, which is given in terms of a database D, representing the data at the source, structured according to the schema S.

The formal semantics of $\langle \mathcal{J}, D \rangle$ is specified by the set $Mod_D(\mathcal{J})$ of its models, which is the set



Ontology-Based Data Access and Integration, Fig. 1 OBDI/OBDA specification and system

of (logical) interpretations $\mathcal{I}$ for $\mathcal{O}$ such that $\mathcal{I}$ is a model of $\mathcal{O}$ and $\langle D, \mathcal{I}\rangle$ satisfies all the assertions in $\mathcal{M}$. The satisfaction of a mapping assertion depends on its form, which is meant to represent semantic assumptions about the completeness of the source data with respect to the intended ontology models. Specifically, *sound* (resp., *complete*, *exact*) mappings capture sources containing a subset (resp., a superset, exactly the set) of the expected data.

In OBDA, the main service to be provided by the system is *query answering*. The user poses queries by referring only to the ontology and is therefore masked from the implementation details and the idiosyncrasies of the data source. The fact that the semantics of $\langle \mathcal{J}, D\rangle$ is defined in terms of a set of models makes the task of query answering involved. Indeed, query answering cannot be simply based on evaluating the query expression over a single interpretation, like in traditional databases. Rather, it amounts to compute the so-called *certain answers*, i.e., the tuples that satisfy the query in all interpretations in $Mod_D(\mathcal{J})$ and has therefore the characteristic of a logical inference task. Obviously, the computation of certain answers must take into account the semantics of the ontology, the knowledge expressed in the mapping, and the content of the data source. Designing efficient query processing algorithms is one of the main challenges of OBDA.

We discuss now computational issues connected to query answering in OBDA, with the aim of showing which are the sources of computational complexity. An OBDA framework is characterized by three formalisms: (1) the language used to express the ontology, (2) the language used for queries, and (3) the language used to specify the mapping. The choices made for each of the three formalisms affect semantic and computational properties of the system.

The axioms of the ontology allow one to enrich the information coming from the source with domain knowledge and hence to infer additional answers to queries. The language used for the ontology deeply affects the computational characteristics of query answering. For this reason, instead of expressing the ontology in first-order logic (FOL), one adopts tailored languages, typically based on description logics (DLs), which ensure decidability and possibly efficiency of reasoning.

Also, the use of FOL (i.e., SQL), as a query language, immediately leads to undecidability of query answering, even when the ontology consists only of an alphabet (i.e., it is a flat schema) and when the mapping is of the simplest possible form, i.e., it specifies a one-to-one

Ontology-Based Data Access and Integration, Table 1 *DL-Lite$_A$* assertions. Symbols in square brackets may or may not be present, and $R^-(x,y)$ stands for $R(y,x)$

Type of assertion	DL syntax	FOL semantics
ISA/disjointness between concepts	$C_1 \sqsubseteq [\neg]C_2$	$\forall x. C_1(x) \rightarrow [\neg]C_2(x)$
Domain/range of a role	$\exists R^{[-]} \sqsubseteq C$	$\forall x,y. R^{[-]}(x,y) \rightarrow C(x)$
Participation to a role	$C \sqsubseteq \exists R^{[-]}$	$\forall x. C(x) \rightarrow \exists y. R^{[-]}(x,y)$
ISA/disjointness between roles	$R_1^{[-]} \sqsubseteq [\neg]R_2^{[-]}$	$\forall x,y. R_1^{[-]}(x,y) \rightarrow [\neg]R_2^{[-]}(x,y)$
Functionality of roles	(funct $R^{[-]}$)	$\forall x,y,z. R^{[-]}(x,y) \wedge R^{[-]}(x,z) \rightarrow y = z$

correspondence between ontology elements and database tables. Hence, the language typically adopted is union of conjunctive queries (UCQs), i.e., FOL queries expressed as a union of select-project-join SQL queries.

With respect to mapping specification, the incompleteness of the source data is captured correctly by mappings that are sound. Moreover, allowing to mix sound mapping assertions with complete or exact ones leads to undecidability of query answering, even when only CQs are used in queries and mapping assertions and the ontology is simply a flat schema. As a consequence, all proposals for OBDA frameworks so far assume that mappings are sound. In addition, the concern above on the use of FOL applies also for the ontology queries in the mapping. Note instead that the source queries in the mapping are directly evaluated over the source database and hence are typically allowed to be arbitrary (efficiently) computable queries.

A Tractable OBDA Framework

Considering the discussion above, we present now a specific OBDA framework tailored towards efficiency and tractability. The framework makes use of a family of DLs, called *DL-Lite*, which has also given rise to the OWL 2 QL profile (http://www.w3.org/TR/owl2-profiles/) of the Web Ontology Language OWL standardized by the W3C.

DLs are class-based formalisms that represent the domain of interest in terms of classes, or *concepts*, and binary relationships, or *roles*, between classes. Here, we consider *DL-Lite$_A$*, which is able to capture essentially all features of entity-relationship diagrams and UML Class Diagrams, except for completeness of hierarchies. In *DL-Lite$_A$*, a concept is either an atomic concept C (i.e., a unary predicate) or the projection $\exists R$ or $\exists R^-$ of a role R on its first or second component, respectively. A role can be either an atomic role R or an inverse role R^-, allowing for a complete symmetry between the two directions. *DL-Lite$_A$* includes also *value attributes* relating objects in classes to domain values (such as strings or integers), but we do not discuss this aspect here. The *ontology* is modeled by means of axioms that can express *inclusion* and *disjointness* between concepts or roles and (global) *functionality* of roles (with some restrictions on the interaction between functionality and role inclusions to ensure tractability). In Table 1, we illustrate the conceptual modeling constructs captured by *DL-Lite$_A$* assertions and provide also their semantics expressed in FOL.

Mapping assertions are of type GAV, i.e., have the form $\varphi(\vec{x}) \rightarrow \psi(\vec{x})$, where $\varphi(\vec{x})$ is a domain-independent FOL query over the source schema, e.g., expressed in SQL, with answer variables $\vec{x}$, and $\psi(\vec{x})$ is a conjunction of atoms over the concepts and roles of the ontology, whose only variables are those in $\vec{x}$. However, we need to take into account the impedance mismatch between values in the data source and objects that populate classes in the ontology. To do so, the arguments of the atoms in $\psi(\vec{x})$ might be not only constants or variables but also terms constructed by applying *functors* to them. Such functors act as object constructors, like in object-oriented approaches. The meaning of a mapping assertion $\varphi(\vec{x}) \rightarrow \psi(\vec{x})$ is to extract the tuples satisfying $\varphi(\vec{x})$ and to use them to (partially) populate according to $\psi(\vec{x})$ the concepts and roles, constructing suitable objects through the functors. Actually, such extraction is typically only virtually performed during query answering.

The DLs of the *DL-Lite* family, including *DL-Lite*$_A$, combined with the GAV mapping assertions above, have been designed so as to enjoy the *FO-rewritability* property: given a UCQ q and an OBDA specification $\mathcal{J} = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, it is possible to compile q, $\mathcal{O}$, and $\mathcal{M}$ into a new FO query q' formulated over $\mathcal{S}$. Such query q' has the property that when evaluated over a database D for $\mathcal{S}$, it returns exactly the certain answers for q over the OBDA system $\langle \mathcal{J}, D \rangle$, for every data source D. Each such q' is called an (FO-) *perfect rewriting* of q w.r.t. $\mathcal{J}$. FO-rewritability immediately implies that the data complexity of computing certain answers is in AC^0, which is the same complexity as that of FOL query evaluation in relational databases.

Techniques for Query Answering via Rewriting

In the tractable OBDA framework previously described, one can think of a simple technique for query answering, which first retrieves an initial set of concept and role instances from the data source through the mapping and then, using the ontology axioms, "expands" such a set of instances deriving and materializing all the logically entailed concept and role assertions; finally, queries can be evaluated on such an expanded set of instances. Unfortunately, the instance materialization step of the above technique is not feasible in general, because the set of entailed instance assertions starting from even very simple OBDA specifications and small data sources may be infinite.

As an alternative to the above materialization strategy, most of the approaches to query answering in OBDA are based on query rewriting, where the aim is to first compute the perfect rewriting q' of a query q w.r.t. an OBDA specification $\mathcal{J}$ and then evaluate q' over the source database.

The above described OBDA framework allows for modularizing query rewriting. Indeed, the current techniques for OBDA consist of a phase of *query rewriting w.r.t. the ontology* followed by a phase of *query rewriting w.r.t. the mapping*. In the first phase, the initial query q is rewritten with respect to the ontology, producing a new query q_o, still over the ontology signature: intuitively, q_o "encodes" the knowledge expressed by the ontology that is relevant for answering the query q. In the second phase, the query q_o is rewritten with respect to the mapping $\mathcal{M}$, using the mapping assertions as rules for reformulating the query with respect to the source schema signature. We illustrate now the two phases more in detail.

Query Rewriting w.r.t. the Ontology

Most of the proposed techniques [3, 5, 13] start from a CQ or a UCQ and end up producing a UCQ (i.e., a set of CQs) expanding the initial query. They are based on variants of clausal resolution [10]: every rewriting step essentially corresponds to the application of clausal resolution between a CQ among the ones already generated and a concept or role inclusion axiom of the ontology. Each such step produces a new conjunctive query that is added to the resulting UCQ. The rewriting process terminates when a fixed point is reached, i.e., no new CQ can be generated.

A potential bottleneck of the rewriting approach is caused by the size of the rewritten query, and several research works aim at optimization techniques addressing this issue. For example, the first algorithm for query rewriting w.r.t. a *DL-Lite* ontology [3] has been improved in [5, 13] by refining and optimizing the way in which term unification is handled by the above resolution step. Notice that the sentences corresponding to the ontology axioms may be Skolemized (e.g., due to the presence of existentially quantified variables in the right-hand side of a concept inclusion): to compute perfect rewritings, the unification of Skolem terms during resolution can actually be constrained in various ways with respect to standard resolution.

Some recent proposals for optimizing query rewriting w.r.t. the ontology (e.g., [5, 7, 15]) are based on the use of Datalog queries besides CQs and UCQs, to express either intermediate results or the final rewritten query. The same idea has also been used to extend query rewriting to more expressive, not necessarily FO-rewritable ontology languages [2,5,6,13]. Other approaches take a more radical view and propose strategies based on partial materialization of instance assertions [9].

Query Rewriting w.r.t. the Mapping

It is well known by the studies on data integration that rewriting a query w.r.t. GAV mappings boils down to a simple unfolding strategy, which essentially means substituting every predicate of the input query with the queries that the mapping associates to that predicate [11]. In OBDA, however, query rewriting w.r.t. mappings is complicated by the following two aspects: (1) OBDA mappings allow for constructing objects that are instances of the ontology predicates from the values stored in the data source, in order to deal with the mentioned impedance mismatch problem; (2) the source queries in the mapping are expressed using the full expressive power of SQL, which is needed to bridge the large cognitive distance that may exist between the ontology and the source schema.

Solutions to the first problem depend on the strategy adopted to construct objects from values. When functors applied to values are used, as in the tractable framework for OBDA we presented above, logic terms constructed through such functors can be treated in the standard way in the unifications at the basis of the unfolding procedure: see, e.g., the algorithm proposed in [14], which relies on techniques from partial evaluation of logic programs. In the R2RML standard, (http://www.w3.org/TR/r2rml/) functors are realized through templates that construct W3C compliant URIs for objects from the values returned by the SQL query in the mapping assertion.

The second problem heavily affects the performance of the query-answering algorithm. Indeed, current SQL engines have hard times in optimizing the execution of queries expressed over virtual views, like those introduced by the unfolding that use complex SQL features such as union, nesting, or aggregation. Performance problems are of course amplified when there are several SQL queries mapping the same ontology predicate. Due to the abovementioned limitations, it is not realistic to group all such queries within a single mapping assertion for each predicate. However, without such grouping, the mapping associates several queries to the same predicate, and therefore, the size of the query obtained by rewriting w.r.t. the mapping may be exponential in the size of the input query. Indeed, in real-world applications, it may very well happen that the size of the produced rewriting is too large to be handled by current SQL engines. Techniques to avoid or mitigate these issues are currently under investigation.

Key Applications

The applications of OBDA and OBDI include all the real-world settings in which an organization needs a unified and transparent access to its data, based on a domain model. Examples are:

- Enterprise information systems, where data governance and data access can be greatly enhanced by the use of the ontology
- Scientific data management, at least in those fields where ontologies are available as unified representations of relevant meta-data
- Public administration and government data management, where the OBDI paradigm can be the enabling technology for information sharing and semantic interoperability
- Open data publishing, where the ontology can help determining what to publish and which strategy to follow in order to enrich the data with useful meta-data

Future Directions

OBDA and OBDI are young paradigms, and many problems related to them are still open. Here is an incomplete list:

- Although query processing in OBDI has been the main subject of investigation, there is still much room for optimization techniques, especially those aiming at making query processing feasible in the “big data” setting.

- Real-world applications often demand more expressive languages in the various components of the OBDI system. An interesting direction that is currently under investigation is to extend the query processing techniques to the case where both the ontology language and the mapping language go beyond the expressive power considered above and to the case of more powerful languages for specifying user queries.
- Although we concentrated on description logics, other types of ontology languages have been considered and studied, notably those based on extensions of Datalog [2].
- Since the ontology should reflect the conceptual model of the domain, and not the information at the sources, it is likely that source data are not fully coherent with the axioms in the ontology. How to design inconsistency-tolerant query answering methods is an important challenge in OBDI.
- In various applications of OBDI, e.g., enterprise information systems, there is the need to provide the user with update facilities. Obviously, the updates should be expressed at the level of the ontology, and the main challenge is to design techniques for translating the update requests into appropriate updates on the source data, similarly to the notorious problem of view update.
- Finally, interesting research developments aim at going beyond query processing and exploring the power of OBDI in more general data governance tasks, including data quality checking, data cleaning, data profiling, and data provenance.

Cross-References

▶ Description Logics
▶ Incomplete Information
▶ Inconsistent Databases
▶ Information Integration
▶ Ontology
▶ View-Based Data Integration

Recommended Reading

1. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF, editors. The description logic handbook: theory, implementation and applications. 2nd ed. Cambridge: Cambridge University Press; 2007.
2. Calì A, Gottlob G, Lukasiewicz T. A general datalog-based framework for tractable query answering over ontologies. J Web Semant. 2012;14:57–83.
3. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R. Tractable reasoning and efficient query answering in description logics: the *DL-Lite* family. J Autom Reason. 2007;39(3):385–429.
4. Ceri S, Gottlob G, Tanca L. Logic programming and databases. Berlin: Springer; 1990.
5. Chortaras A, Trivela D, Stamou GB. Optimized query rewriting for OWL 2 QL. In: Proceeding of the 23rd international conference on automated deduction (CADE). Wroclaw; 2011. p. 192–206.
6. Eiter T, Ortiz M, Simkus M, Tran T-K, Xiao G. Query rewriting for Horn-SHIQ plus rules. In: Proceeding of the 26th AAAI conference on artificial intelligence (AAAI). Toronto: AAAI Press; 2012.
7. Gottlob G, Kikot S, Kontchakov R, Podolskii VV, Schwentick T, Zakharyaschev M. The price of query rewriting in ontology-based data access. Artif Intell. 2014;213:42–59.
8. Imielinski T, Lipski W Jr. Incomplete information in relational databases. J ACM. 1984;31(4):761–91.
9. Kontchakov R, Lutz C, Toman D, Wolter F, Zakharyaschev M. The combined approach to ontology-based data access. In: Proceeding of the 22nd international joint conference on artificial intelligence (IJCAI). Barcelona; 2011. p. 2656–61.
10. Leitsch A. The resolution calculus. Berlin: Springer; 1997.
11. Lenzerini M. Data integration: a theoretical perspective. In: Proceeding of the 21st ACM symposium on principles of database systems (PODS). Madison; 2002. p. 233–46.
12. Levy AY, Rousset M-C. Combining Horn rules and description logics in CARIN. Artif Intell. 1998;104(1–2):165–209.
13. Pérez-Urbina H, Horrocks I, Motik B. Efficient query answering for OWL 2. In: Proceeding of the 8th internaional semantic web conference (ISWC). Volume 5823 of lecture notes in computer science. Springer; 2009. p. 489–504.
14. Poggi A, Lembo D, Calvanese D, De Giacomo G, Lenzerini M, Rosati R. Linking data to ontologies. J Data Semant. 2008;X:133–73.
15. Rosati R, Almatelli A. Improving query answering over *DL-Lite* ontologies. In: Proceeding of the 12th international conference on the principles of knowledge representation and reasoning (KR). Toronto; 2010. p. 290–300.