ISSN 0361-7688, Programming and Computer Software, 2012, Vol. 38, No. 6, pp. 315–326. © Pleiades Publishing, Ltd., 2012.
Original Russian Text © L.A. Kalinichenko, 2012, published in Programmirovanie, 2012, Vol. 38, No. 6.

# Effective Support of Databases with Ontological Dependencies: Relational Languages instead of Description Logics

**L. A. Kalinichenko**

*Institute of Informatics Problems, Russian Academy of Sciences, ul. Vavilova 44/2, Moscow, 117333 Russia*
*e-mail: leonidk@synth.ipi.ac.ru*

Received December 12, 2011

**Abstract**—The paper provides a brief survey of the languages for efficient support of access to the databases satisfying the ontological dependencies. The first part of the paper retraces the development of description logics intended for application in the database and information systems context, as well as experimental results of building of the "ontologically based" data access systems. The main part of the paper is devoted to the survey of the development of relational database languages providing broader and more efficient support of queries over databases with the ontological dependencies compared to the description logics.

**DOI:** 10.1134/S0361768812060059

## 1. INTRODUCTION

Ontology in informatics is understood as a formal knowledge representation in the form of a set of concepts of some subject domain and relations between them. Such a representation is used for reasoning about entities of the subject domain, as well as for the domain description. Thus, ontology is a vocabulary agreed in the community that is used for modeling the subject domain. Ontologies are used in various fields of informatics, including artificial intelligence, Semantic Web, system engineering, software engineering, biomedical informatics, library sciences, information architectures, etc., as a form of representation of knowledge about the discourse space. Independent of a particular knowledge representation language, modern ontologies possess great structural similarity. For example, the majority of the ontologies define individuals (instances), classes (concepts), attributes, and relations. Examples of ontologies developed for various subject domains (such as, for instance, cultural heritage, linguistics, Earth sciences, genomics, protein systematics, pharmaceutics, structure of plants, education, biology, enterprise architecture, biomedicine, and so on) can be found at http://en.wikipedia.org/wiki/Ontology_(information_science). These examples, as well as the development of various ontology modeling languages, testify of practical need in ontologies for structuring and representing knowledge in various subject domains.

During last years, ontologies have been actively expanded in information systems and databases. This expansion is based on the simple idea to use ontologies as conceptual database schemas. Recently, ontological languages and systems have been rather frequently used for representation and support of conceptual schemas over (relational) databases. Such a construct makes it possible to use axioms of conceptual schemas and facilities of ontological inference machines upon interpreting queries to databases. In practice, this approach is used for the development of relatively small applications, basically, in scientific subject domains. The activity of this kind is accompanied by the introduction of new terms, such as Ontology Based Information Systems (OBIS), Ontology Based Data Access (OBDA), and Ontology Based Data Integration (OBDI) [1–4]. The objective of this paper is to give a brief analysis of the state of the art in the field of the ontology-based systems and to compare description logic based facilities used in them with functionally similar facilities based on the traditional database tools. The latter are extensions of the relational database model by special dependencies presented in the Datalog± language family [5–12] and their implementations. Discussion of capabilities of these languages is the main subject of this paper.

The paper is organized as follows. In Section 2, relationships between the ontological and conceptual modeling and the corresponding language facilities are discussed. Analysis of the development of the description logics languages that are used as conceptual modeling facilities in the framework of databases and information systems is given in Section 3. In Section 4, a brief survey of experimental implementations of the "ontology-based" data access systems is presented. Sections 2–4 discuss issues that were earlier considered in [13], including analysis of evolution of the ontology definitions and the place of the ontological languages and models in informatics. In Section 5, the problem of efficient support of ontological dependencies in databases based on the database technology is formulated. A brief discussion of types of dependences supported by the databases (Section 6) is followed by a

brief survey of the state of the art of the family of the Datalog± languages, which ensure effective support of ontological dependencies in the database schemas upon query answering. In Section 8, relationship between the families of the Datalog± languages and description logics DL-Lite is discussed. In Section 9, estimates of efficiency of implementation of the Datalog± languages are given. Results of the analysis presented in the paper are summarized in the concluding section.

## 2. ONTOLOGIES AS CONCEPTUAL SCHEMAS

First of all, it is interesting to retrace the evolution of the definition of the concept of ontology in informatics. In the early 1990s, due to efforts aimed at the creation of interoperability standards for information system components, ontologies were identified as necessary components of knowledge-based systems. By T. Gruber's definition, which is widely used until now, ontology is a specification of conceptualization [14], or, in other words, ontologies are formal definitions of concepts in some subject domain and relations between them. As early as 1998, N. Guarino analyzed possible approaches to the development of ontology-driven information systems (ISs) and outlined variants of ontology use in the course of IS design and functioning [15]. The key idea of this work is that ontology-based specifications do not depend on any implementation aspects. Unfortunately, in this study, no parallel was run with conceptual modeling methods for semantically interoperable systems, information resource integration systems, etc., the inherent feature of which is that the specification level used does not depend on the implementation.

The evolution of the concept of ontology from the early Gruber's definition during last years is also worth noting. The later point of view of T. Gruber on the ontology [16], which is shared by many researchers, seems to be most representative. It can be characterized by the following statements. In the context of informatics, ontology is a set of representation primitives by means of which a knowledge domain or discourse are modeled. The representation primitives are usually classes (or sets), attributes (or properties), and links (or relations) between instances of classes. In particular, in the context of the database systems, ontologies can be viewed as an abstraction level of data models[1] expressed in hierarchical and relational structures but designed for modeling knowledge on individuals, their attributes, and relations to other individuals.[2] Ontologies are said to be at the "semantic" level, whereas database schemas are data models at the "logical" or "physical" levels.[3] In view of their independence on the lower-level data models, the ontologies are sometimes used for integrating heterogeneous databases, for ensuring interoperability of various systems, and for specifying interfaces of independent knowledge-based services.[4]

There also are definitions similar to the following one given by D. Calvanese [17]. Ontology is a representation schema that determines formal conceptualization of the subject domain. In this case, ontology specification usually includes two different levels. The intensional level specifies the set of conceptual elements and rules for defining conceptual structures of the subject domain (in terms of the description logics, such definitions are said to be collected in the TBox). The extensional level specifies the set of instances of the intensional elements[5] (which are concentrated in the ABox).

Ontologies can also be used as metalevel specifications for describing model categories whose instances are conceptual elements. Along with the analysis of the extension of the concept of ontology itself and the extension of the set of functions assigned to the ontologies, it is also interesting to consider evolution of the relationship (determined by the ontological community) between ontological languages and various traditional languages of informatics.

The ontological languages themselves are relatively simple. To represent elements from the intensional level, one needs facilities for representing concepts, concept properties, relations between the concepts and their properties, and axioms. The axioms are logical formulas that express conditions of the intensional level that the extensional elements must meet. The axioms that express relation of "being a subclass," an equivalence relation for classes or their compositions, a relation of class non-overlapping, cardinality relations, etc. can also be represented.

On the extensional level, individuals and facts are represented. An individual, or object, is represented as an instance of an extension of a concept. Facts are represented as relations between instances of concepts or properties. It is not difficult to see that such simple languages (or their subsets) can naturally be identified with subsets of various languages used in some fields of informatics. Examples of such identification can easily be found in publications of the ontological community. First of all, there is a tendency to consider ontologies as diagrams (represented as semantic nets, schemas of entity–relationship models, and UML class diagrams) [18].

[1] In the given case, we mean database schemas.

[2] When speaking about the abstraction level, data is often turn to knowledge.

[3] T. Gruber intentionally forgets about conceptual database schemas.

[4] This phrase sounds very strange: as if there do not exist more advanced languages (compared to ontological ones) for integrating databases, for ensuring semantic interoperability, and for specifying mediators that make it possible to implement the above-listed functions.

[5] In the deductive databases, it is commonly accepted to call these levels intensional (IDB) and extensional (EDB) databases.

The point of view of the ontological community on the relationship between ontological and other languages can be summarized as follows [17]:

• with respect to knowledge representation languages, ontologies are knowledge representation schemas;

• with respect to logic, logic is a tool imparting semantics to ontological languages;

• with respect to conceptual data models, conceptual schemas are special ontologies that can be used for conceptualization of a particular logical data model;

• as for programming languages, definitions of classes in programs are special ontologies serving for conceptualization of particular structures used in calculations.

This scheme reflects the desire to classify ontological languages among various languages of informatics and to determine the place of the latter languages among ontological ones[6]. It often happens that some languages that have nothing to do with the ontological languages are still classified among the latter. Below is the example of such classification of ontological languages given in [17].

1. Graph languages:

• Semantics nets;

• Conceptual graphs;

• UML class diagrams, entity–relationship model schemas.

2. Frame languages:

• Systems on frames;

• OKBC, XOL.

3. Logical languages:

• Description logics (e.g., SHOIQ, DLR, DL-Lite, OWL);

• Rule languages (e.g., RuleML, LP/Prolog, F-Logic);

• First-order logics (KIF);

• Non-classical logics (e.g., non-monotonous and probabilistic logics).

Let us try to understand what new such expansion of ontological terminology gives. It may seem that ontologies open a way to new research directions, after decades of studies in the field of conceptual data models, deductive databases, integration systems for heterogeneous databases, semantic interoperability, and so on. However, the required theoretical foundation and particular high-level models and languages in these fields have been created long time ago. In fact, it is better to focus on revealing usefulness of application of ontological languages in various fields of informatics, including conceptual modeling in the field of information systems and databases. What new do studies in the field of ontological languages bring to the theory and practice of databases and information systems?

[6] This list can easily be extended. Say, database schemas are special ontologies for conceptualization of database structures, and the like.

Ontological models considered in publications devoted to their use in databases and information systems are based on the first-order predicate logic or, more often, on its subsets—description logics. Essentially, in the DB and IS context, ontological languages (in particular, languages based on description logics) play role of data models and nothing more. Thus, it is better to focus on analysis of specific properties of languages based on the description logics and on the novelty introduced by them into the DB and IS context (what particularly data models based on the description logics can yield compared to relational, object, and other data models).

In so doing, the relationship between the conceptual and ontological modeling is fundamentally important. This issue is discussed in more detail in [19]. Here, we confine ourselves to only several basic considerations.

The conceptual modeling implements abstract semantic modeling of the subject domain (definition of classes of the subject domain, their relationships, and constraints), which does not depend on a particular implementation and serves as a facility for generating a reference specification reflecting consensus in the community, which includes the developers, IS users, and ISs themselves.

Conceptual schemas are also used as global schemas when integrating information resources (databases), in the course of the IS design, and upon interpreting queries. The conceptual schema determines structure of the subject domain, whereas ontology should be focused mainly on defining concepts used in the subject domain. It is important to note that conceptual schemas of databases and information systems contain not only descriptions of the subject domain classes but also descriptions of behavior of objects (methods, functions, and processes), whereas ontologies do not contain the latter. On the other hand, an ontological model makes it possible to define dependencies between classes of the subject domain objects and their attributes that assume logical inference when responding to a query over the conceptual schema. Besides, ontological definitions of concepts may annotate the corresponding definitions of the conceptual schema (that is where ontologies show their worth in an original way).

## 3. ELABORATION OF LANGUAGES BASED ON DESCRIPTION LOGICS IN THE DATABASE AND INFORMATION SYSTEM CONTEXT

In this section, we briefly analyze development of languages based on description logics that bear a relation to the DB and IS context. The analysis involves two parts. In the first part, we consider languages that were created before they had been included in the W3C stack. In the second part, the basic attention is focused on how languages based on description logics

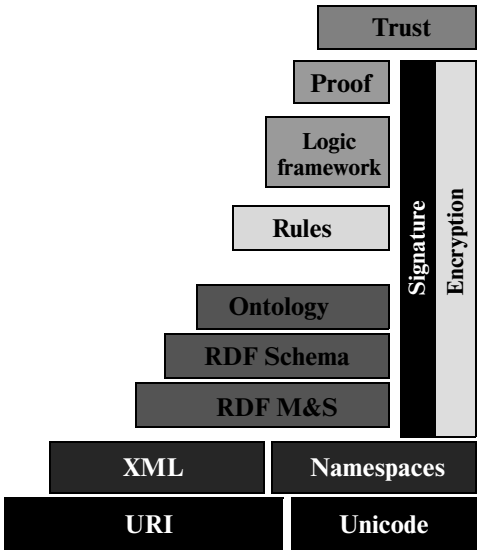


**Fig. 1.** Single-stack W3C architecture.

(as well as other languages) in the W3C stack relate to one another.

Since the end of 1980s, studies and development in the field of description logics aimed at their use in the database context were carried out. One of the first languages of this kind was CLASSIC [20]. This language allows one to define structures of database objects not only in terms of their relations with other objects but also in intensional terms and is capable of inferring, for example, definitions of classes to which a given object belongs. Even at that time, objectives and results of language development were discussed in terms similar to those currently met in works on ontology-based database access. True, the term "ontology" was not used. Languages oriented on support of subject domain concepts and terminological integration appeared later. Among them are LOOM [21], GRAIL [22], SIMS [23], and OIL [24]. These languages support practically all operations of concept formation. LOOM provides also support of reasoning over ABox and TBox.

A milestone of this period is creation of languages combining capabilities of description logics with programs on rules. The AL-log [4, 25] language is a combination of a simple description logic with the Datalog language. The CARIN language [26] extends the description logic by the Horn rules without functions. Works on integration of description logics with reasoning on rules revealed necessity of restricting expressive capabilities of the terminological language part in order to preserve decidability. It is interesting to note that, even at that time, the description logics were used in experiments on data integration. For example, the CARIN language and description logic DLR on n-ary relations were used in works on integration of relational databases [26].

This period was also marked by works on advanced information representation languages based on the first-order logic for object and frame models. Among these languages are the ontological language Ontolingua [27]; the SYNTHESIS language [28], which was designed for providing semantic interoperability and integration of heterogeneous information resources; logic-based frame language F-Logic [29], and the OKBC language [30], which was designed for providing interoperability of databases. Compared to the languages based on the description logics, the languages that are based on the first-order logic and object–frame models possess greater capabilities in what concerns defining types, classes, and logical statements.

As the activity of W3C in the field of the Semantic Web increases, there appear more and more works on ontological languages in this context. The languages accepted by W3C as a technological basis of the Semantic Web are organized in the form of the Semantic Web Stack (Fig. 1). This stack determines hierarchy of languages, in which each level relies on the facilities of the underlying levels. It also demonstrates evolution of the Semantic Web as an extension of the classical hypertext World Web. The main languages of the Semantic Web that have direct relation to ontological specifications include (1) RDF, a simple language for representing data in the form of triples that refer to objects (resources) of the Web and links determining them (data structures expressible in RDF can be represented in the XML syntax); (2) the RDF Schema language, which extends RDF and allows one to define schemes of the resource properties and classes that can be represented in RDF; (3) the family of the OWL languages (OWL 2) for representing ontologies possessing formal semantics, as well as RDF/XML-based serialization of such ontologies.

Recently, viability of the single-stack architecture (SSA) developed by the W3C is doubted. The main objective is that hardly will the single language hierarchy turn out sufficient for all future needs of Web (especially taking into account infancy of the Semantic Web) [31]. Any technology, including language development technology, eventually becomes obsolete. Moreover, no technology can cope with all problems. A multi-stack architecture (MSA), in which many stacks coexist with one another, seems to be more realistic. In the course of the implementation of the RIF (Rule Interchange Format) project and under the influence of the rule languages included in the W3C stack, the stack presented in Fig. 1 bifurcated into two and turned to the multi-stack shown in Fig. 2.

The MSA is extendable, so that other stacks can be added to it when needed. Each level in the multi-stack is a syntactic and semantic extension of the previous level. In this aspect, it is interesting to observe what happens at the stack bifurcation point. Unlike the OWL languages based on description logic, rule languages are logical programming languages relying on

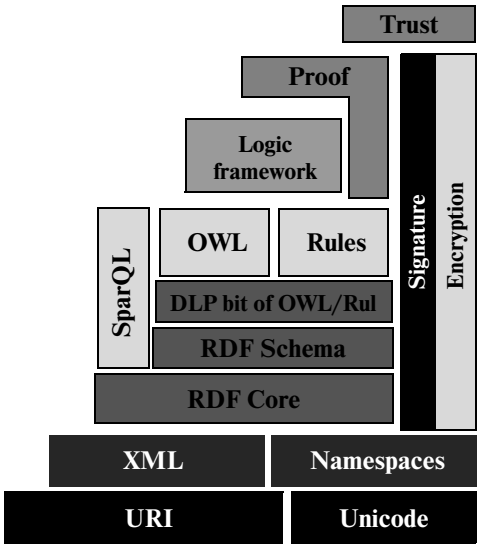


**Fig. 2.** Multi-stack W3C architecture.

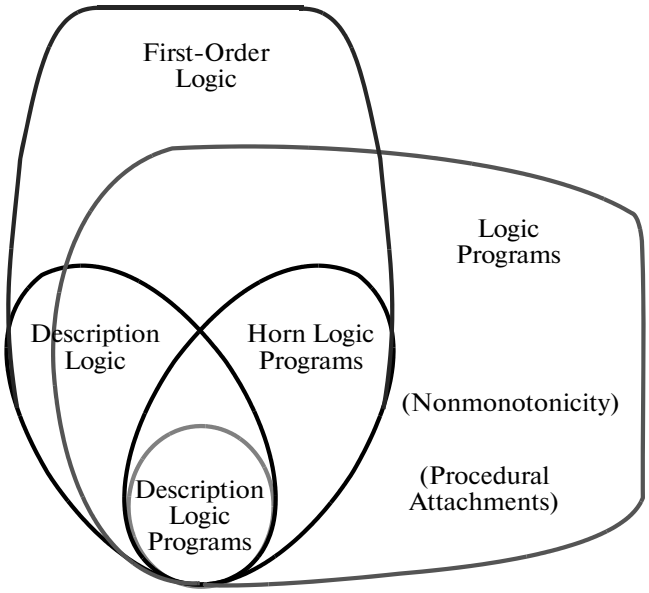


**Fig. 3.** Relationships between logical languages.

non-monotonous reasoning. The specific feature of the MSA shown in Fig. 2 is the (implicit) presence of the description logic program (DLP) level, all operators of which are mapped to the Horn rules (a subset of the first-order logic (Fig. 3)) [32].

This level should be present (at least, implicitly) in the stack to ensure compatibility in the stack from the bottom to the top: the DLP semantics in the stack OWL and in the stack of rules should be identical. At the same time, logical programs based on the subset of the first-order logic (e.g., OWL) do not support concepts of the integrity constraints and their violations. Instead, dependencies are used as statements determining the desirable state of the environment. For example, given the monogamy condition, if an inference result is that Ivan has two wives, Ann and Vera, OWL concludes that Ann and Vera are one and the same person. Unlike this, programs based on rules with non-monotonic reasoning will treat this situation as an inconsistency in the database.

Moreover, there is an opinion that the statement that the rule-based language Datalog is an extension of the DLP level is incorrect. For example, if the database has the only fact *knows(ivan, ann)*, then DLP and Datalog will give different answers to the query on whether *ivan* knows exactly one person. According to the OWL semantics, the answer will be "*unknown*," whereas, in the semantics of the rule-based languages, the answer will be "*yes*." Actually, both answers are correct. Everything depends on what stack—the OWL stack or that of rules—is selected by the user.

It should be noted that the development of the MSA is still underway and needs considerable efforts. For example, taking into account contradictions between constraints and dependencies mentioned above, the question about the place of profiles of the OWL 2 language remains open. It is interesting to note that the creation of profile OWL 2 QL [33] turned out possible owing to studies related to the description logic family DL-Lite [3, 34].

The goal of the development of DL-Lite was support of basic facilities of ontological languages while preserving complexity of reasoning at an acceptable level (the reasoning functions include also answers to queries that can be represented as a union of conjunctive queries over the extensional level (ABox)). The complexity of reasoning in DL-Lite is polynomial in the size of TBox, and the complexity of an answer to a query is estimated as AC0 in the size of ABox.

It is important that, upon query processing, DL-Lite allows one to carry out reasoning in TBox and ABox independently, so that, first, reasoning over TBox is performed, and, then, the query over ABox can be performed by means of SQL.

Logics included in the DL-Lite family are *maximal description logics* that ensure efficient responding to queries over large-scale databases.

The DL-Lite family is formed as follows (Fig. 4).

DL-Lite$_{core}$ supports ISA statements over concepts, concept non-overlapping requirements, role typing, and inclusion constraints. In DL-Lite$_F$, facilities to express functional dependence conditions on roles are added. DL-Lite$_R$ is extended by ISA assertions over roles and role non-overlapping conditions. In DL-Lite$_A$, possibilities of joint use of role inclusion assertions and functionality assertions are added.

Profile OWL 2 QL is based on DL-Lite$_R$ [33]. It is important that this profile includes about one third of the set of axioms supported by OWL 2 [33]. It should be noted that simple description logics, like DL-Lite$_R$,

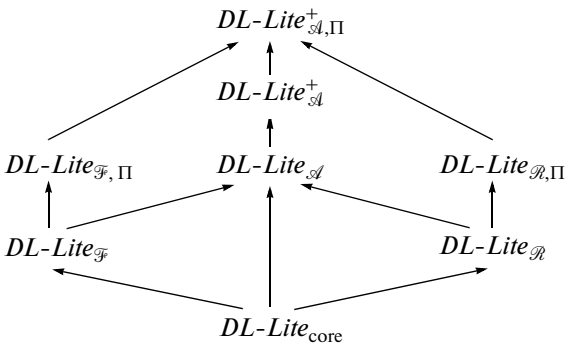


**Fig. 4.** The DL-Lite family of description logics.

are still acceptable for support of simple ontological languages and conceptual data models (like entity–relationship model and a subset of the UML class diagram).

## 4. ONTOLOGY-BASED DATA ACCESS

Ontology-based technologies suggest using information resources based on conceptualization of subject domains and ontology-mediated (by a conceptual point of view on data) data access. As a matter of fact, there are few research works in this field. The approaches adopted in these works are discussed in this section. It should be emphasized that the results of these works are applicable to only relational databases. To ensure data access when using ontologies, it is required (1) to determine mapping of relational database schemas to ontological (conceptual) schemas, (2) to transform a conjunctive query in terms of a conceptual schema to a query over a relational database, and (3) to implement the query obtained. Let us consider in detail how this is done. The approach discussed is quite applicable for the implementation of the OWL 2 QL profile.

Relational databases store data, whereas concept (class) instances are objects. The basic mechanism to overcome this impedance mismatch is well-known in the field of object databases. In this case, the unique object identifier is produced by the Skolem function whose parameter is the value of the database tuple. Mapping of a relational data model into a conceptual one is constructed as a set of conjunctive queries over atomic concepts, attributes, role attributes, and the corresponding queries in the SQL language. Formally, the mapping consists of assertions of type $\varphi \rightsquigarrow \psi$, where $\varphi$ is an arbitrary SQL query of arity $n > 0$ over the database and $\psi$ is the union of conjunctive queries of arity $n > 0$ in terms of ontology $T$ (e.g., in TBox' terms in DL-Lite$_R$).

To answer the query, each assertion of mapping $\varphi \rightsquigarrow \psi$ is divided into several assertions of form $\varphi \rightsquigarrow p$, with one assertion corresponding to each atom $p$ in $\psi$. Atoms of query $q$ are unified with the atoms on the right-hand side of the mapping, resulting thus in a union of conjunctive queries. Then, each atom turns using the left-hand side of the mapping to obtain an SQL query.

It should be noted that, before performing the above-described procedure, the query is deductively extended based on the axioms specified in the schema and the DL-Lite$_R$ inference machine. Note also that, according to the ontological model, information about the subject domain is not complete. The queries obtained as a result of the transformation ensure receiving *certain* answers, independent of how the incomplete information can be augmented.

The approach described is implemented in QuOnto [35], an instrumental environment supporting reasoning in ontologies of family DL-Lite, as well as responding to queries represented in the UCQ (union of conjunctive queries) form. QuOnto permits using external relational DBMSs (such as Oracle, DB2, IBM Information Integrator, SQL Server, MySQL, etc.).

Instead of a relational DBMS, QuOnto can work jointly with a federated database, the result of integration of relational databases. Such a configuration of QuOnto is called MASTRO-I [36]. One should take into account that the federated database is presented for QuOnto by a single relation schema. To form the federated database, the GAV approach can be used. The way QuOnto functions in MASTRO-I does not differ from that described above in this section.

## 5. STATING OF THE ONTOLOGY-BASED DATA ACCESS PROBLEM IN TERMS OF THE DATABASE TECHNOLOGY

Efficient support of queries to large-scale data is a fundamental problem for the database management systems. To remove the gap between the database world and the world of ontologies represented in terms of the description logics, which can efficiently be processed, ontological data dependencies should be taken into account when answering a query (in this paper, a minimal set of such dependencies is assumed to include those expressible in the description logics of the DL-Lite family). In terms of the relational data model, this means the following [5]. An extensional relational database $D$ (which is called ABox in the community of experts on description logics) is considered jointly with an ontological theory $\Sigma$ (called TBox), which contains rules and dependencies based on which inference of new intensional data from $D$ is performed. The response to a query is formed with respect to the logical theory $D \cup \Sigma$ rather than with respect to the database $D$. In other words, for a conjunctive query $q$, it is decided whether $D \cup S \models q$ is performed instead of just checking $D \models q$. It is known that the answer to a conjunctive query $q$ over $D \cup \Sigma$ is equivalent to the answer to the same query with

respect to the extension of $D$ obtained as a result of *chase* applied to $\Sigma$, which is denoted as *chase*($D$, $\Sigma$).

The chase procedure (or simply chase) is a tool created on the basis of a fundamental algorithm suggested for checking implication dependencies [37] and, later, for checking query containment [38]. In informal terms, chase is a process of correcting database $D$ with respect to the dependency set $\Sigma$ that makes the resulting database satisfy these dependencies. However, procedure *chase*($D$, $\Sigma$) may be endless and, thus, incomputable. In order to work with large-scale databases, it is required that the answer to the query be not only decidable but also tractable in the data size (given that $\Sigma$ and $q$ are fixed) and realizable on the basis of relational query processors. Since a conjunctive query in DL-Lite can be rewritten into a query in the first-order logic, a conjunctive query $q$ based on an ontology $\Sigma$ can be rewritten as an SQL query over the original database $D$.

A query $q$ over a dependency class $C$ is said to be rewritable into a query in the first-order logic (*FO-rewritable*) if and only if, for any dependency set $\Sigma$ in the class $C$, there exists a query $q_\Sigma$ in the first-order logic such that, for any database $D$, $D \cup \Sigma \models q$ holds if and only if $D \models q_\Sigma$.

Recently, a number of studies in the framework of the relational data model were carried out [5–13] to reveal expressive data dependency classes with the aim of generalization of DL-Lite and other advanced ontological formalisms (e.g., to provide possibilities of using joins in bodies of rules (with certain constraints ensuring decidability)), as well as with the aim of preserving the possibility of FO-rewritability of queries or, at least, achieving tractable query computability with respect to the data size. Results of studies of such dependency classes performed by G. Gottlob and his colleagues are presented in a number of publications as a family of special rules extending the Datalog language and proofs showing tractability of queries over databases with such dependencies and considerable extensions of capabilities of such description logics as DL-Lite, DLR-Lite, and EL. There are implementations of the proposed approach that substantiate its high efficiency compared to the description logic implementations.

In what follows, a brief survey of tractable extensions of the Datalog language is given.

## 6. TYPES OF DEPENDENCIES IN DATABASES

Since the case in point is extensions of the Datalog language by constructs that make it possible to express various data dependencies, we briefly recall types of constraints met in databases. These are relation keys, external keys–reference constraints (a set of relation attributes corresponding to the key of another relation), inclusion constraints, functional dependencies, multi-valued dependencies, and joint dependencies. It is known that, in a unified form, these dependencies are represented as statements of the first-order predicate logic [39].

In so doing, one should distinguish between *embedded* dependencies of the form $\forall x_1 \ldots \forall x_n[\phi(x_1, \ldots, x_n) \longrightarrow \exists z_1 \ldots \exists z_k \psi(y_1, \ldots, y_m)]$, where $\{z_1, \ldots, z_k\} = \{y_1, \ldots, y_m\} - \{x_1, \ldots, x_n\}$, $\varphi$ is a (possibly, empty) conjunction of atoms, $\psi$ is a nonempty conjunction of atoms, and *total* dependencies whose heads do not contain existential quantifiers (in other words, their heads do not contain variables that are not contained in the rule body). Both formulas $\phi$ and $\psi$ contain *relation atoms* of the form $R(w_1, \ldots, w_l)$ and *equality atoms* of the form $w = w'$, where $w, w', w_1, \ldots, w_l$ are variables.

Two special types of constraints—TGDs (Tuple Generating Dependencies) and EGDs (Equality Generating Dependencies)—are frequently used, since they are naturally express the majority of constraints in relational databases.

The TGDs [40] are dependencies that require presence of certain tuples in the database. For a relational schema $R$, a TGD $\Sigma$ is a first-order logic formula of the form $\forall X \forall Y \phi(X, Y) \longrightarrow \exists Z \psi(X, Z)$, where $\phi(X, Y)$ and $\psi(X, Z)$ are conjunctions of atoms over $R$ called the rule body and rule head, respectively [8]. For example, a TGD of the form $r_1(X, Y) \longrightarrow \exists Z r_2(X, Z)$, which does not contain variables that occur more than once both in the body and head, is called the inclusion dependence (ID) (see, e.g., [39,41]). Usually, the universal quantifiers in TGDs and EGDs are omitted. Such TGDs are embedded ones. If $Z$ is empty, then the TGD is a total dependence, since it completely determines variables in the rule head.

The TGDs (together with appropriate EGDs) make it possible to define external keys, inclusion dependencies, joins, multi-valued dependencies, and other dependencies.

The EGDs are dependencies over $R$ of the form $\forall X \phi(X) \longrightarrow X_i = X_j$, where $\phi(X)$ is a conjunction of atoms, and $X_i, X_j \in X$. The EGDs are generalizations of keys (KDs) and functional dependencies [39].

In the general case, receiving an answer to a query upon presence of TGDs is undecidable [40]. Therefore, restricted variants of the TGDs are used. Similarly, when a set of EGDs is considered together with a set of TGDs, the problem of receiving an answer to a query is, generally speaking, also undecidable. Therefore, the relationship between the EGDs and TGDs should be controled to preserve decidability. This can be achieved if the EGDs and TGDs *do not conflict* [6].

One of the important constraints is the *negation dependency* (ND) given by a first-order logic formula of the form $\forall X \phi(X) \longrightarrow \bot$, where $\phi(X)$ is a conjunction of atoms (not containing constraints) and $\bot$ denotes the logical constant *false*. For example, the requirement that a person ID cannot occur in the relations *employee*(*ID*, *Name*) and *retired*(*ID*, *Name*) simultaneously is written as *employee*($X$, $Y$) $\wedge$

*retired*$(X, Y) \longrightarrow \bot$. In other words, the negation dependency assigns false values to certain formulas in all models of the given theory. Similar NDs turned out useful in modeling various ontological structures, including the case where the latter are used as conceptual schemas.

## 7. DATALOG± LANGUAGE FAMILY

The Datalog± language family [5, 6] originated in 2009 and was designed for creating efficient algorithms for answering queries in more advanced ontological languages compared to the DL-Lite family of description logics. Languages from the Datalog± family are based on the Datalog language rules that permit using variables under the existential quantifier in the rule head, like in Datalog, which possesses the capability of *generating new values* [42, 43]. As noted in [44], the absence of facilities for generating new values in the standard Datalog makes it impossible to use this language for ontological reasoning. The basic extension of Datalog± is rules in the form of embedded TGDs. Taking into account that obtaining an answer to a conjunctive query when TGDs are used is undecidable [40], certain syntactic constraints are needed. The two fundamental paradigms of introduction of constraints ensuring decidability and tractability of obtaining an answer to a query with respect to the data size are as follows: *weak acyclicity* [45] and *guardedness* [46]. When using weakly acyclic sets of TGDs, the chase procedure always terminates, resulting thus in a finite instance of the database. Clearly, query answering over such an instance is decidable. Decidability of query answering under the condition of guarded TGDs, i.e., TGDs containing in their bodies atoms (the so-called guards) that include all variables occurred in the body, follows from the fact that the tree of the database instance constructed upon chase has restricted width.

### *7.1. Guarded and Linear Languages of the Family*

Guarded TGDs (the version of the Datalog language extended by such TGDs is called guarded Datalog±) are special TGDs for which query answering is decidable and even tractable with respect to the data size [8, 11]. Upon using such TGDs, query are calculated over a finite chase result.

A TGD $\sigma$ is a *guarded* TGD if and only if its body contains an atom that includes all universally quantified variables of the $\sigma$ body. The leftmost atom of this kind is said to be a *guarding atom* (or a *guard*) of $\sigma$. Atoms of the $\sigma$ body that are not guards are called auxiliary atoms of $\sigma$.

**Example.** The TGD $r(X, Y), s(Y, X, Z) \longrightarrow \exists W\, s(Z, X, W)$ is a guarded TGD (with the guard $s(Y, X, Z)$), whereas $r(X, Y), r(Y, Z) \longrightarrow r(X, Z)$ is not a guarded TGD.

**Linear** Datalog± is a version of the guarded Datalog± in which queries are also FO-rewritable while preserving tractability [8, 11]. A TGD is linear if and only if it contains only one atom in the body. Linear Datalog± generalizes the well-known class of inclusion constraints and is more expressive. For example, the following linear TGD, which cannot be expressed by means of inclusion constraints, states that each person that manages itself is a manager:

$$supervises(X, X) \longrightarrow manager(X).$$

Generalizations of linear TGDs are multi-linear TGDs. A TGD $\sigma$ is a multilinear TGD if each atom of the body $body(\sigma)$ is a guard, i.e., includes all variables of $\sigma$ located under the universal quantifier. Clearly, any linear TGD is a multilinear TGD, since the only atom of its body is automatically a guard. The use of multilinear TGDs makes expressible statements like, for example, the following one:

$$employee(X), manager(X) \longrightarrow \exists Y\, supervises(X, Y),$$

which states that each employee that is a manager manages other employees. Clearly, this TGD neither is linear nor is an inclusion constraint.

**Table**

| Axioms in the description logic DL-Lite | Datalog± dependencies |
|---|---|
| Inclusion of concepts | |
| $employee \sqsubseteq person$ | $employee \longrightarrow person$ |
| Inclusion of roles and their inversions | |
| $reports^{-} \sqsubseteq manager$ | $reports(X, Y) \longrightarrow manager(Y, X)$ |
| Role transitivity | |
| $trans(manager)$ | $manager(X, Y), manager(Y, Z) \longrightarrow manager(X, Z)$ |
| Participation | |
| $employee \sqsubseteq \exists\, report$ | $employee(X) \longrightarrow \exists Y\, report(X, Y)$ |
| Non-overlapping | |
| $employee \sqsubseteq \neg\, customer$ | $employee(X), customer(X) \longrightarrow \bot$ |
| Functionality | |
| $funct(reports)$ | $reports(X, Y), reports(X, Z) \longrightarrow Y = Z$ |

Examples of the correspondence of Datalog± dependencies to their analogues in DL-Lite are presented in the table (see p. 322) [47].

The next example from [8] shows some axioms in the description logic DL-Lite and the result of their transformation into Datalog±. These are concept inclusion axioms, which state that (a) papers published in conference proceedings and journals are publications, (b) papers published in conference proceedings are not journal papers, (c) any scientist is an author of publications, and (d) *isAuthorOf* relates scientists to publications:

$CPaper \sqsubseteq Article$, $JPaper \sqsubseteq Article$,

$CPaper \sqsubseteq \neg JPaper$, $Scientist \sqsubseteq isAuthorOf$,

$\exists isAuthorOf \sqsubseteq Scientist$, $\exists isAuthorOf^{-} \sqsubseteq Article$.

These axioms are transformed into the following TGDs and constraints (atomic concepts and roles are identified with the corresponding predicates):

$CPaper(X) \longrightarrow Article(X)$, $JPaper(X) \longrightarrow Article(X)$,

$CPaper(X), JPaper(X) \longrightarrow \bot$,

$Scientist(X) \longrightarrow \exists Z\, isAuthorOf(X, Z)$,

$isAuthorOf(X, Y) \longrightarrow Scientist(X)$,

$isAuthorOf(Y, X) \longrightarrow Article(X)$.

The algorithm of translation from DL-Lite into Datalog± is presented in [6].

### *7.2. Sticky Language of the Datalog± Family*

In another advanced language version from the Datalog± family, a class of sticky sets of TGDs [10] is introduced. The latter include TGDs with constraints on multiple occurrences of variables in the rule bodies. An informal treatment of the concept of stickiness in terms of chase is as follows. In the course of chase, each value occurring in a new atom $a$ upon performing join in the TGD body must present in all subsequent atoms generated from $a$. Whether this requirement is followed is effectively recognized by the condition that includes special marking of variables in rules [10]. In particular, this implies that the body variables on which the join occurs must also be included in the rule head.

The class of rules reducible to the first-order logic that contains rules possessing simultaneously properties of sticky TGD sets and linear TGDs is called the class of sticky join TGD sets [7, 12].

Sticky TGDs and sticky join TGDs [7, 10, 12] are special classes of TGDs that generalize linear TGDs and make it possible to naturally express ontological dependencies that are not expressible in the description logics corresponding to OWL. For example, one of the limitations of the DL-Lite logics is impossibility of using joins in the rule bodies, like in the following example of a sticky TGD. The rule determining that, for each project that is run by some department, there should exist an external auditor that is an expert in the project application domain (in_area) is expressed by the following TGD [7]:

$runs(W, X), in\ area(X, Y) \longrightarrow \exists Z\, external(Z, Y, X)$.

Sticky sets of TGDs can be used in relational database schemas of arbitrary arity. Unlike this, languages of the DL-Lite family, as the majority of the description logics, are applicable to only binary relations. In particular, sticky sets of TGDs generalize class of inclusion dependencies, whereas DL-Lite constraints generalize only unary or binary inclusion dependencies. DLR-Lite [48], the generalization of DL-Lite on relations of arbitrary arity, can also be strictly extended by sticky sets of TGDs. Sticky sets of TGDs can represent constraints and rules containing joins. Since query answering in the case of using TGDs that include joins is generally undecidable, the class of rules used is restricted by sticky requirements. On the whole, these constraints do not make the class of dependencies used in practice too narrow.

Sticky sets of TGDs greatly generalize other constructs containing joins introduced with the aim of extending capabilities of description logics. To be more specific, in [49], the notion of Cartesian product of concepts is introduced. For example, the statement that elephants are bigger than mice is represented as

$elephant(X) \wedge mouse(Y) \longrightarrow bigger\ than(X, Y)$.

It is evident that the notion of concept product is just a weak subclass of sticky sets of TGDs.

## 8. RELATIONSHIP BETWEEN DATALOG± LANGUAGE FAMILY AND DESCRIPTION LOGICS DL-LITE

Basic languages of the DL-Lite family, such as DL-LiteF, DL-LiteR, and DL-LiteA, can be reduced to linear TGDs and sticky sets of TGDs augmented by negation constraints and EGDs. These additional facilities do not increase complexity of query answering [5]. Moreover, description logics DL-Lite$_F$, DL-Lite$_R$ and DL-Lite$_A$ obtained from DL-Lite$_F$, DL-Lite$_R$, and DL-Lite$_A$, respectively, by adding conjunctions to the left-hand side of axioms can be reduced to multilinear TGDs (with NCs and KDs). Analogues of such description logics with binary roles in the DLR-Lite family with $n$-ary roles are also reducible to multilinear TGDs and sticky sets of TGDs (with NCs and KDs) [12].

On the whole, by combining linear (or sticky) sets of TGDs and EGDs with NCs, we obtain more expressive formalisms compared to those in the majority of tractable ontological languages widely used in practice, such as, in particular, DL_Lite$_A$, DL_Lite$_F$, and DL_Lite$_R$. It is important that, in this case, the FO-rewritability is preserved and tractability of query answering under the condition of large volume of data is ensured.

The relationship between different languages of the Datalog± family and description logics is depicted in Fig. 5 [11].

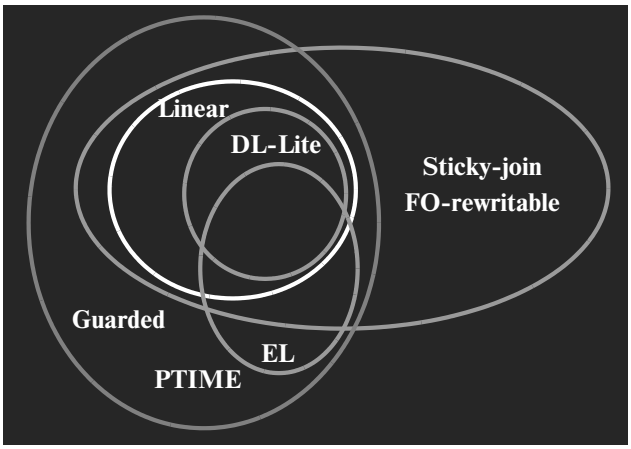


**Fig. 5.** Relationship between languages of the Datalog± family and description logics.

The complexity estimate of query answering with respect to the database volume for languages from the Datalog± family is as follows: for linear, sticky, and sticky-join languages, the complexity class is $AC_0$, and, for the guarded version of the language, the complexity is polynomial [6, 10].

## 9. EFFICIENCY OF THE DATALOG± IMPLEMENTATION

Efficiency of the Datalog± implementation for data management is demonstrated by the Nyaya[7] system [50, 51], which possesses advanced tools for ontological reasoning and capability of expressing conjunctive queries and user-defined dependencies over data collections. For such collections, various data sets (including those represented in RDF) are used. The latter sets are, possibly, related to metadata determining content constraints on data sets in such languages of the Semantic Web as RDF(S) and OWL (or their variants). Each set of this kind provides an interface for expressing queries to the data set and the possibility of accessing metadata. To estimate efficiency of the inference facilities in Datalog±, the metadata of the original sets are represented in Nyaya as Datalog± constraints. The collection of such sets is presented in a uniform way, so that the users may express queries and extend the set of the original constraints by new dependencies formulated in the Datalog± language.

The users work in such an environment by using a high-level interface (e.g., SPARQL or Query by Example). Users' queries are rewritten in Datalog± and reformulated with regard to dependencies specified in Datalog±. Since Nyaya uses only FO-rewritable dependencies, such a transformation results in unions of conjunctive queries (UCQs). Then, these queries are rewritten in terms of the data sets involved with regard to relations between entities supported in such sets. As a result, the UCQs are transformed into actual queries over sets of data stored (including SQLs if relational sets are used). Efficiency of the Nyaya system[8] was tested on commonly accepted test data sets LUBM[9] and UOBM [52] and was compared with that of two other known semantic data management systems BigOWLim[10] and IODT[11]. Since the expressive capability of metadata in these sets exceeds capabilities of Datalog± (LUMB uses OWL-Lite, and UOMB uses OWL-DL), the best approximation of metadata acceptable for all three systems was employed. Estimates of time spent on reasoning and query answering showed advantages of Nyaya compared to the other two systems [50, 51]. For example, query answering time in Nyaya is less than that in BigOWLim and IODT by the factors of 1.6 and 1.9, respectively.

An important characteristic of query rewriting methods is the number of queries obtained as a result of the rewriting. Nyaya ensures considerable reduction of this number compared to the QuOnto and Requiem systems [53].

## 10. CONCLUSIONS

The paper surveys development of relational languages and facilities for efficient support of queries over database schemas with ontological dependencies versus languages and facilities based on description logics. Development of description logics designed for use in the database and information systems and well-known experimental results of the development of "ontology-based" data access systems resulted in the creation of the family of description logics DL-Lite, which are maximal subsets of facilities possessing acceptable efficiency of work with databases. These results found their application in the OWL 2 QL profile [33] based on DL-Lite$_R$. Works on "ontology-based" information systems are aimed at the use of ontological languages for conceptual modeling, i.e., at the creation of conceptual data models based on description logics. Languages based on description logics and the corresponding conceptual models can be mapped onto the corresponding data models of database integration systems (e.g., language SYNTHESIS [28]), with their semantics being preserved [54]. Moreover, databases with schemas in OWL can be integrated with other, traditional, databases in database integration systems. Semantics-preserving mapping of OWL onto SYNTHESIS was constructed in [54, 55].

It seems that, from the point of view of databases and information systems, the idea of the so-called "ontology-based" systems is the result of terminolog-

[7] Nyaya is the name of a logics school in the philosophy of Hinduism; it literally means "recursion" in Sanskrit.

[8] http://pamir.dia.uniroma3.it:8080/nyaya/
[9] http://swat.cse.lehigh.edu/projects/lubm/
[10] http://www.ontotext.com/owlim/big/
[11] http://www.alphaworks.ibm.com/tech/semanticstk

ical expansion of ontologies into the field of databases. At the same time, studies carried out in this field and developments in the field of description logics are important from the point of view of inclusion of relational databases into the context of the Semantic Web.

The alternative approach to efficient support of queries over database schemas with ontological dependencies developed by G. Gottlob and his colleagues in the framework of traditional technologies seems to be quite promising. The approach consists in identifying maximal fragments of the first-order logic language that are capable of expressing ontological dependencies in the database schemas and preserve possibility of efficient support of queries to databases over such schemas, including the FO-rewritability property. A family of languages Datalog± possessing properties of perception simplicity, decidability, and efficiency has been constructed. The languages obtained are extremely flexible and expressive in the field of ontological reasoning. It is proved that the expressive capability of the simplest language of the Datalog± family (linear Datalog± with the negation dependency and non-conflicting keys) exceeds capabilities of logics of the DL-Lite family. Moreover, unlike the description logics, the languages from the Datalog± family are not confined to only binary relations and can be effectively extended by non-monotonous stratified negation, which is an important expressive facilities not available in the description logics.

The existing implementations of the Datalog± languages substantiate possibility of creating more efficient implementations in the framework of database technologies than those obtained with the help of the description logics.

## ACKNOWLEDGMENTS

This work was supported in part by the Russian Foundation for Basic Research (project nos. 10-07-00342-a and 11-07-00402-a) and by the Presidium of the Russian Academy of Sciences (the fundamental research program no. 15, project 4.2).

## REFERENCES

1. Calvanese1, D., Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., and Rosati, R., Ontology-based database access, *Proc. of the 15th Italian Conf. on Database Systems (SEBD 2007)*, 2007.
2. Calvanese, D., Ontology-based Data Management Masters Ontology Spring School, September 2009, http://ksg-projects.meraka.csir.co.za/krr-projects/events/moss09-1/MOSS-09-OBDM-calvanese-draft.pdf.
3. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro1, M., and Rosati, R., Reasoning Ontologies and Databases: The DL-Lite Approach Web 2009, *LNCS*, vol. 5689, Springer, 2009, pp. 255–356.
4. Donini, F., Lenzerini, M., Nardi, D., and Schaerf, A., Al-log: Integrating Datalog and Description Logics, *J. Intelligent Information Systems (JIIS)*, 1998, vol. 27, no. 1.
5. Calì, A., Gottlob, G., and Lukasiewicz, T., Datalog±: A Unified Approach to Ontologies and Integrity Constraints, *ICDT 2009*, St. Petersburg, 2009.
6. Calì, A., Gottlob, G., and Lukasiewicz, T., A General Datalog-Based Framework for Tractable Query Answering over Ontologies, *PODS'09*, June 2009.
7. Calì, A., Gottlob, G., and Pieris, A., Advanced Processing for Ontological Queries, *Proc. of the 36th Int. Conf. on Very Large Data Bases*, Singapore, 2010.
8. Calì, A., Gottlob, G., Lukasiewicz, T., Marnette, B., and Pieris, A., Datalog±: A Family of Logical Knowledge Representation and Query Languages for New Applications, *Proc. of the 25th Ann. IEEE Symp. on Logic in Computer Sci.*, 2010, pp. 228–242.
9. Calì, A., Gottlob, G., and Pieris, A., Query Answering under Expressive Entity–Relationship Schemata, *ER 2010*, *LNCS*, 2010, vol. 6412, pp. 347–361.
10. Calìi, A., Gottlob, G., and Pieris, A., New Expressive Languages for Ontological Query Answering, *Proc. of the Twenty-Fifth AAAI Conf. on Artificial Intelligence*, 2011.
11. Gottlob, G., Ontological Queries Rewriting and Optimization, *Proc. of the ICDE,* 2011.
12. Gottlob, G., Orsi, G., and Pieris, A., Ontological Query Answering via Rewriting, *ADBIS 2011*, *LNCS*, 2011, vol. 6909, pp. 1–18.
13. Kalinichenko, L.A., Onology Expansion: Ontologies in Information Systems, *Trudy Vtorogo Simpoziuma "Ontologicheskoe modelirovanie"* (Proc. of the Second Symp. "Ontological Modeling"), Kazan, 2010, Moscow: IPI RAN, 2011, pp. 29–44.
14. Gruber, T.R., Toward Principles for the Design of Ontologies Used for Knowledge Sharing, *Int. J. Human–Comput. Studies*, 1995, vol. 43, pp. 907–928.
15. Guarino, N., Formal Ontology in Information Systems, *Proc. of FOIS'98,* Trento, Italy, 1998, Amsterdam: IOS, pp. 3–15.
16. Gruber, T., Ontology, in *Encyclopedia of Database Systems*, Ling Liu and *Ozsu*, M.T., Eds., Springer, 2008.
17. Clavanese, D., Knowledge Bases and Databases. Part 2: Ontology-Based Access to Information, *Presentation*, http://www.inf.unibz.it/~calvanese/teaching/08-09-kbdb/lecture-notes/p2-obda-2up.pdf, 2009.
18. Clavanese, D., Ontology-based Data Management, *Masters Ontology Spring School*, September 2009, http://ksg-projects.meraka.csir.co.za/krr-projects/events/moss09-1/MOSS-09-OBDM-calvanese-draft.pdf.
19. Kogalovsky, M.R. and Kalinichenko, L.A., Conceptual and Ontological Modeling in Information Systems, *Programming Computer Software*, 2009, vol. 35, no. 5, pp. 241–256.
20. Borgida, Brachman, R.J., McGuinness, D.L., and Resnick, L.A., CLASSIC: A Structural Data Model for Objects, *ACM SIGMOD Record*, 1989, vol. 18, no. 2.
21. MacGregor, R.M., Using a Description Classifier to Enhance Deductive Inference, *Proc. of the Seventh IEEE Conf. on AI Applications*, 1991, pp. 141–147.
22. Rector, A.L., Bechofer, S., Goble, C.A., Horrocks, I., Nowlan, W.A., and Solomon, W.D., The Grail Concept Modelling Language for Medical Terminology, *Artificial Intelligence Medicine*, 1997, vol. 9, pp. 139–171.

23. Arens, Y., Chun-Nan Hsu, and Knoblock, C.A., Query Processing in the Sims Information Mediator, in *Advanced Planning Technology*, AAAI, 1996.
24. Fensel, D., Horrocks, I., Van Harmelen, F., Decker, S., Erdmann, M., and Klein, M., Oil in a Nutshell, *Proc. of the 12th Int. Conf. on Knowledge Engineering and Knowledge Management EKAW2000*, France, 2000.
25. Clavanese, D., Giacomo, G., and Lenzerini, M., Description Logics for Information Integration, in *Computational Logic: From Logic Programming into the Future*, *LNCS*, Springer, 2001.
26. Goasdou è, F., Lattes, V., and Rousset, M.-C., The Use of Carin language and Algorithms for Information Integration: The Picsel Project, *Int. J. Cooperative Information Systems (IJCIS)*, 1999, vol. 9, no. 4, pp. 383–401.
27. Gruber, T., A Translation Approach to Portable Ontology Specifications, *Knowledge Acquisition*, 1993, vol. 5, no. 2, pp. 199–220.
28. Kalinichenko, L.A., SYNTHESIS: The Language for Defining, Designing, and Programming of Heterogeneous Interoperable Information Resource Environments, Moscow: IPI RAN, 1993.
29. Kifer, M, Lausen, G., and Wu, J., Logical Foundations of Object-Oriented and Frame-based Systems, *J. ACM*, 1995, vol. 42, no. 4.
30. Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D., and Rice, J.P., Open Knowledge Base Connectivity (OKBC) Specification Document 2.0.3, *Technical Report*, SRI International and Stanford University (KSL), April 1998.
31. Kifer, M., de Bruijn, J., Boley, H., and Fensel, D., A Realistic Architecture for the Semantic Web, *Proc. of the 1st Int. Conf. on Rules and Rule Markup Languages for the Semantic Web (RuleML2005)* (Galway, Ireland, November 2005), *LNCS*, vol. 3791, Springer, pp. 17–29.
32. Grosof, B.N., Horrocks, I., Volz, R., and Decker, S., Description Logic Programs: Combining Logic Programs with Description Logic, *WWW2003,* Budapest, May 20–24, 2003.
33. OWL 2 Web Ontology Language: Profiles, W3C, 2009.
34. Clavanese, D., Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R., Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family, *J. Automated Reasoning*, 2007, vol. 39.
35. Acciarri, Clavanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., and Rosati, R., QUONTO: QUerying ONTOlogies, *Proc. of AAAI 2005*, pp. 1670–1671.
36. Clavanese, D., Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., and Rosati, R., MASTRO-I: Efficient Integration of Relational Data through DL Ontologies, *Proc. of the 2007 Int. Workshop on Description Logic (DL 2007)*, CEUR Electronic Workshop Proceedings, 2007.
37. Maier, D., Mendelzon, A.O., and Sagiv, Y., Testing Implications of Data Dependencies, *ACM Trans. Database Systems*, 1979, vol. 4, no. 4, pp. 455–469.
38. Johnson, D.S. and Klug, A.C., Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies, *J. Comput. Syst. Sci.*, 1984, vol. 28, no. 1, pp. 167–189.
39. Abiteboul, S., Hull, R., and Vianu, V., *Foundations of Databases*, Addison-Wesley, 1995.
40. Beeri, C. and Vardi, M.Y., The Implication Problem for Data Dependencies, *Proc. of ICALP*, 1981, pp. 73–85.
41. Kanellakis, P.C., Elements of Relational Database Theory, *Handbook of Theoretical Computer Science*, Van Leeuwen, J., Ed., Amsterdam: Elsevier, 1991, pp. 1074–1156.
42. Mailharrow, D., A Classification and Constraint-based Framework for Configuration, *Artificial Intelligence Engineering Design, Analysis Manufacturing*, 1998, vol. 12, no. 4, pp. 383–397.
43. Cabibbo, L., The Expressive Power of Stratified Logic Programs with Vvalue Invention, *Inf. Comput.*, 1998, vol. 147, no. 1, pp. 22–56.
44. Patel-Schneider, P.F. and Horrocks, I., A Comparison of Two Modelling Paradigms in the Semantic Web, *J. Web Sem.*, 2007, vol. 5, no. 4, pp. 240–250.
45. Fagin, R., Kolaitis, P.G., Miller, R.J., and Popa, L., Data Exchange: Semantics and Query Answering, *Theor. Comput. Sci.*, 2005, vol. 336, pp. 89–124.
46. Calì, A., Gottlob, G., and Kifer, M., Taming the Infinite Chase: Query Answering under Expressive Relational Constraints, *Proc. of KR-2008*, 2008, pp. 70–80.
47. Gottlob, G., Ontological Queries Rewriting and Optimization, *Proc. of ICDE-2011*, 2011.
48. Calvanese, D., Giacomo, G., Lembo, D., Lenzerini, M., and Rosati, R., Data Complexity of Query Answering in Description Logics, *Proc. of KR,* 2006, pp. 260–270.
49. Rudolph, S., Krotzsch, M., and Hitzler, P., All Elephants are Bigger than all Mice, *Proc. of DL*, 2008.
50. De Virgilio, R, Orsi, G., Tanca, L., and Torlone, R., Semantic Data Markets: A Flexible Environment for Knowledge Management, *CIKM'11*, October 2011, Glasgow, Scotland, UK.
51. De Virgilio, R, Orsi, G., Tanca, L., and Torlone, R., Reasoning over Large Semantic Datasets, *Technical Report RT-DIA-149*, University Roma Tre, 2009.
52. Ma, L., Yang, Y. Qiu, Z., Xie, G.T., Pan, Y., and Liu, S., Towards a Complete OWL Ontology Benchmark, *Proc. of the 3rd Europ. Semantic Web Conf. (ESWC)*, 2006, pp. 125–139.
53. Perez-Urbina, H., Motik, B., and Horrocks, I., Tractable Query Answering and Rewriting under Description Logic Constraints, *J. Applied Logic*, 2009, vol. 8, no. 2, pp. 151–232.
54. Kalinichenko, L. and Stupnikov, S., OWL as Yet Another Data Model to Be Integrated, *Proc. of the 15th Int. Conf. ADBIS 2011*, Vienna, Austria, September 2011.
55. Stupnikov, S.A. and Skvortsov, N.A., Mutual Mapping of a Canonical Information Model and OWL 2 Language, *Trudy 12-oi Vserossiiskoi nauchnoi konferentsii "Elektronnye biblioteki: perspektivnye metody i teknologii, elektronnye kollektsii"* (Proc. of the 12th All-Russian Conf. "Digital Libraries: Prospective Methods and Technologies, Digital Collections" RCDL'2010), Kazan: KGU, 2010, pp. 392–398.