Utilization Of Query Rewriting Over Ontology Change: A Review

Helna Wardhana, Yohanes Suyanto, Sigit Priyanta

Abstract: There are two approaches to view-based query processing, called query rewriting and query answering. Query rewriting is one of the most important reasoning techniques for answering queries on data represented utilizing ontologies or ontology-based data access applications. The last years a wide variety of query rewriting systems has been proposed, including the research about ontology change. In this paper, we present the discussion of the utilization of query rewriting on ontology change. We divide the use of query rewriting in this paper into 3 parts i.e. proposing a new algorithm, research development in Description Logics (DL) family and complexity of conjunctive query. Furthermore, we also present a brief description of the advantages and disadvantages of using query rewriting algorithm only on 3 types of ontology, ontology contraction, ontology evolution and ontology change. Finally, we show the list of query rewriting algorithm publication based on their approaches.

Index Terms: query rewriting, query answering, conjunctive query, ontology, ontology change, description logics

1. INTRODUCTION

There are two approaches to view-based query processing, called query rewriting and query answering [1]. One of the most important reasoning techniques for query answering is query rewriting. In query rewriting, query processing is divided in two steps, where the first re-expresses the query in terms of a given query language over the alphabet of the view names and the second evaluates the rewriting over the view extensions. In query answering, we do not pose any limit to query processing and the only goal is to exploit all possible information, in particular the view extensions, to compute the answer to the query [1]. Some recent research indicated that the problem of view-based query processing is relevant in many aspects of database management including query answering with incomplete information. There are two approaches to view based query processing, called query rewriting and query answering [1]. Query rewriting is an important technique for answering queries on data represented utilizing ontologies or ontology-based data access applications [2].

- *Helna Wardhana is currently pursuing doctoral degree in Study Program of Computer Science & Electronic Department, Gadjah Mada University, Yogyakarta. Helna Wardhana is also a lecturer of Study Program of Informatic Engineering, STMIK Bumigora Mataram, Lombok, Indonesia*
  E-mail: helna.wardhana@mail.ugm.ac.id
- *Yohanes Suyanto is a lecturer of Study Program of Computer Science & Electronic Department, Gadjah Mada University, Yogyakarta.*
  E-mail: yanto@ugm.ac.id
- *Sigit Priyanta is a lecturer of Study Program of Computer Science & Electronic Department, Gadjah Mada University, Yogyakarta.*
  E-mail: seagatejogja@ugm.ac.id

2. DEFINITION

2.1 Query Rewriting

Query rewriting is a technique used in mediation based data integration systems for translating the queries formulated over the mediated schema to a query over the various sources by making use of the view definitions. According to [1], given a query Q and a set of view definitions, and the goal is to reformulate the query into an expression, the rewriting, that refers only to the views, and provides the answer to Q. Typically, the rewriting is formulated in the same language used for the query and the views. [1] also given the extensions of the views and the aim is to calculate the set of tuples that are implied by these extensions, i.e., the set of tuples that are in the answer set of Q in all the databases that are consistent with the views. Query rewriting is a common reasoning technique of choice. A rewriting of a query *Q* with respect to an ontology *T* is another query (typically a union of CQs or a data-log query) that captures the information in *T* relevant for answering *Q* w.r.t. *T* [2].

2.2 Ontology

There are many definitions of ontology from various research. An ontology is an explicit specification of a conceptualization [3]. An ontology is a representation vocabulary, often specialized to some domain or subject matter. More precisely, it is not the vocabulary as such that qualifies as an ontology, but the conceptualizations that the terms in the vocabulary are intended to capture [4]. An ontology became important because without ontologies, or the conceptualizations that underlie knowledge, there cannot be a vocabulary for representing knowledge [4]. Furthermore, the first step in designing an effective knowledge representation system and vocabulary, is to perform an effective ontological analysis of the field or domain. Weak analyses lead to incoherent knowledge bases. According to [5], ontology in computer science is defined as formal and explicit specifications of a shared conceptualization of a domain of discourse and is the main driving force behind semantic web vision. Ontologies enable knowledge to be made explicit, formalize the relevant underlying view of the world (domain model) and make such models machine processable and interpretable [6]. In [7], ontologies to be effective, need to change as fast as the parts of the world they describe. There are two main challenges in adapting ontologies, the evolution of ontologies should reflect both the changing interests of people and the changing data,

for example the documents stored in a digital library. Some of the reasons to develop an ontology are: to share common understanding of the structure of information among people or software agents, to enable reuse of domain knowledge, to make domain assumptions explicit, to separate domain knowledge from the operational knowledge and to analyze domain knowledge [8].

2.3 Ontology Change

Ontologies are often large and complex structures whose development and maintenance emerge several interesting research problems. One of the most important such problems is ontology change, which refers to the generic problem of changing an ontology in response to a certain requirement [9]. Some of the aspects that will initiate a change when requested for accommodation in the ontology [5]:

- New Concept: This is the most common change in any ontology. New concepts emerge and have to be accommodated in the concept hierarchy.
- Concept with Changed Properties: This is the case when the concept in focus is already present in the ontology but its properties and restrictions are different from those associated with existing concepts.
- Simple vs. Aggregated Concept: The concept in focus might be a combination of two or more existing concepts (or vice versa). The ontology framework shall preferably detect and act accordingly to accommodate the change.
- Concept vs. Property: Different modeling approaches are followed by ontology engineers for building ontologies. One such case is modeling the same concept either as a class in OWL or as a property of some other existing class.
- Concept with Changed Hierarchy: Different modeling approaches may fix the same concept in different hierarchical locations in two different ontologies.

According to [10], the functional requirement specifies which functionality must be accommodated for changing the ontology. This functionality heavily depends on the underlying ontology model. The more powerful and expressive model requires a richer set of modelling primitives. The notion of an ontology has to be clarified before discussing about functional requirements. There is no standard ontology model appropriate to the variety of ontology models. Due to differences in ontology models, the author focuses on the common features of ontology models, namely concepts, properties, instances, as well as concept inheritance. Each of these ontology entities can be updated by one of the meta change transformations: add, remove, modify. A full set of changes (Tab. 1) can thus be defined by the cross product of the set of entities of the ontology model, which form meta schema, and the set of meta-changes.

Table 1. *Changes in the ontology [10]*

Meta changes Meta entities	**Add**	**Remove**	**Modify**
Concept	Add concept	Remove concept	Rename concept
Concept hierarchy	Add subConceptOf relationship	Remove subConceptOf relationship	Set subConceptOf relationship
Property	Add property	Remove property	Rename property
Property Domain	Add property domain	Remove property domain	Set property domain
Property Range	Add property range	Remove property range	Set property range
Instance	Add instance	Remove instance	Rename instance
Property Instance	Add property instance	Remove property instance	Set property instance

2.4 Description Logics

Description logics (DLs) are a family of logic-based knowledge representation (KR) formalisms designed to represent and reason about the knowledge of an application domain in a structured and well-understood way [11]. Most KR applications of DLs are of this kind, and also the majority of ontologies focusses on conceptual modeling. In contrast, more recent applications of DLs additionally involve (potentially large amounts of) instance data, particularly, instance data plays an important role when using DL ontologies for data integration and in ontology-mediated data access. In DLs, a TBox is used to represent conceptual information, and instance data is stored in the ABox [12]. Description logics (DLs) are logical languages for expressing and modelling ontologies. In description logics, the ontological axioms are usually divided into two sets: The A-Box (assertional box) and a T-Box (terminological box) which contains axioms and constraints that allow us, on the one hand, to infer new facts from those given in the A-box and on the other hand, to express restrictions such as keys [13]. Description Logics (DL) are constraint languages that have been extensively studied in the past with the goal of providing useful modeling constructs while keeping the query answering problem decidable [14].

3. UTILIZATION OF QUERY REWRITING

Many researches have talked about the utilization of query rewriting over ontology change and several approaches have been proposed in the literature. Some focus on proposing a new algorithm [15], [16], [17], [18], [19], [20], [21], [13], [2], [14], research development in DL-family [22], [23], [24], [25], [26], and complexity of conjunctive queries [12], [13], [2], [27], [28], [14], [29].

3.1 Proposing A New Algorithm

In [15] learned the issue of calculating a rewriting for a conjunctive query (CQ) over an ontology that has been changed-that is, for ontologies which some of its axioms have been appended or existing ones deleted. That issue particularly concentrated on resolution-based query rewriting system. The primary objective of their approach is to use the information generated for the premier rewriting, so that evade repeating many of the calculations. [15] have proposed a new algorithm for ontology revision and ontology contraction. In [15] have also provided a comprehensive experimental evaluation utilizing the famous query rewriting systems Requiem and Rapid. They have also executed the algorithm that calculates a new rewriting immediately from the input rewriting. They have evaluated those prototypes over a range of test and commonly used big-scale ontologies. The result of their experiments recommended that it would be extremely useful for modern state-of-the-art systems to combine their

techniques, particularly when a relatively small portion of the ontology is appended or deleted. Previously, [16] & [17] have proposed a practical algorithm just to append a new axiom to ontology and to delete existing ones. Those algorithms suitable with rewriting calculation as well as those underpinning Requiem, QuOnto, Nyaya and Rapid. They have acquired pleasing results when they have executed and assessed their algorithm on the rewriting system Requiem. [18] have provided a new algorithm for the efficient calculation of query rewriting on DL-Lite$_R$ ontologies, called Rapid. An ontology representation language of Description Logics (DL), called DL-Lite$_R$, which provided from the research of description logics, supports profile of the OWL 2 QL. Rapid using benefit of the specific feature from the problem of query rewriting via the first-order resolution in OWL 2 QL. Rapid also calculated the collection of the non-redundant rewritings of a user query efficiently, by keeping away unknowledgeable and copious inferences, such as by reducing the requirement for expanded query subsumption inspects. Venetis et al. [19] have dedicated a new algorithm for query rewriting on DL-Lite$_R$ ontologies, which is based on a new approach that cultivates every atom respectively. Furthermore, the algorithm integrates the output to calculate the last union of conjunctive queries (UCQ) rewriting. [20] have compared algorithm proposed, called Requiem, with the basic algorithm introduced by [24], then executing both and making an empirical evaluation utilizing ontologies and queries obtained from the realistic applications. The results show that their algorithm generates significantly smaller rewriting in majority cases, which could be prominent for practicality in realistic applications. According to [21], the answer to the query for every extensional database D was obtained by evaluating the rewritten query over D. They have recommended a novel algorithm that calculates the rewriting of a conjunctive query for Linear Datalog± and DL-Lite, which the primary concept underpinning the algorithm was the construction of a bounded Datalog query. Although if the acquired rewriting was syntactically recursive, it can be interpreted into an SQL query. [13] have talked about query rewriting and query optimization then proposed a novel rewriting algorithm for slightly general kind of description logics. The algorithm proposed indicates that a conjunctive query (CQ) toward an enterprise ontology may be collected into a union of conjunctive queries (UCQ) toward the underpinning database. Moreover, they have proposed an effective new method that involves famous description logics of the DL-Lite family and runs for Linear Datalog±, called Nyaya. A test query generation algorithm where can be implemented to Horn ontologies have been proposed by [2]. Moreover, [2] provided the technique of correctness evaluation which has already been attested helpful to discover critical implementation mistakes in rewriting systems. This technique was not expressed by available benchmarks. [14] have proposed a new algorithm of query rewriting which manages constraints modeled in the DL ELHIO¬ and utilize it to reveal that answering conjunctive queries is PTime-complete with respect to data complexity. Some of description logics from the EL and DL-Lite families were handled by this algorithm.

3.2 Research development in DL-family

An initiatory research of belief contraction in the EL family of description logics has been provided by declaring the primary definitions for contraction in a description logic and show the primary issues embroiled when addressing belief set contraction in EL [22]. The foundation for contraction in the EL family of description logics, particularly for basic contraction, have located by providing a formal account of basic contraction of a TBox [22]. [23] have been inspected the properties of existing model and formula-based semantics to contraction for the description logics DL-Lite and EL, which underpin the QL and EL profiles of OWL 2. Consequently, the author suggested that these contraction semantics, which are well-understood and well behaved for propositional logics, are intrinsically problematical in the context of ontology languages [23]. [24] have introduced the DL-Lite family, a new family of Description Logics particularly appropriated to catch conceptual data models and basic ontology languages, while keeping low complexity of reasoning. The approach has focused on two members of the family (i.e., DL-LiteR and DL-LiteF). Reasoning means not only calculating subsumption between concepts and verifying satisfiability of the whole knowledge base, but also answering complex queries (specifically unions of conjunctive queries) over the instance level (ABox) of the DL knowledge base [24]. According to [24], the usual DL reasoning tasks for the DLs of the DL-Lite family, are polynomial in the size of the TBox, and query answering is LogSpace in the size of the ABox (i.e., in data complexity) and this is the first result of polynomial time data complexity for query answering over DL knowledge bases. According to the logic, allowed for a separation between TBox and ABox reasoning during query evaluation: the part of the process requiring TBox reasoning is independent of the ABox, and the part of the process requiring access to the ABox can be executed by an SQL engine, so as taking advantage of the query optimization strategies provided by existing database management system. According to [25], the DL-Lite family of Description Logics has been designed with the specific aim of allowing for answering complex queries (in particular, conjunctive queries) over ontologies with very large instance sets (ABoxes) but this aim has been actually reached only for relatively simple (short) conjunctive queries of existing DL-Lite systems. A new query rewriting method for DL-Lite ontologies, called Presto, and an experimental comparison of Presto with the main previous approaches to query answering in DL-Lite have been provided by [25]. Presto is actually able to handle conjunctive queries of up to 30 atoms in real ontologies, whereas existing techniques are only able to answer conjunctive queries of less than 7-10 atoms (depending on the complexity of the TBox). Ontology evolution will advantage from the incorporation and utilize of belief revision techniques and theories, but the most influential belief revision theory, the AGM (Alchouron, Gardenfors and Makinson) theory have been indicated to be incompatible with many ontology representation formalisms [26]. The AGM theory cannot be used as the basis for defining contraction operators for several ontology representation languages. Furthermore [26] test the postulate of relevance which has been proposed in the belief revision literature as a more intuitive alternative to the AGM postulate of recovery. [26] proposed the ontology representation languages which are compatible with the relevance and the recovery postulate respectively, and the proposed set of postulates (i.e., with relevance instead of recovery) is far more sufficient than the original AGM set as far as ontology evolution is concerned.

3.3 Complexity of Conjunctive Query

Conjunctive query answering becomes an important part in applications of description logics (DLs) that comprise instance data. The complexity of conjunctive query answering in expressive DLs between ALC and SHIQ established by [12]. [12] have been revealed that conjunctive query answering is 2ExpTime-complete in the presence of inverse roles and only ExpTime-complete without them. According to [13], query rewriting consists of the compilation of an ontological query into an equivalent query against the underlying relational database. The focus here is on soundness and completeness. [13] have been introduced a new rewriting algorithm for rather general types of ontological constraints (description logics) after analysis the previous results. The author particularly denotes how a conjunctive query (CQ) against an enterprise ontology can be compiled into a union of conjunctive queries (UCQ) against the underlying database. Ontological query optimization tries to fix this process in order to obtain possibly small and cost-effective output UCQ. Furthermore, [13] review existing optimization methods and propose an effective new method that works for Linear Datalog±, a description logic that encompasses well-known description logics of the DL-Lite family. According [2], query rewriting systems constitutes complex software programs although based on appropriate algorithms. Sophisticated optimizations make the systems more complex and errors become more likely to happen. [2] proposed an algorithm which generates relevant test queries when its given an ontology as input. Each of these queries can be applied to verify whether the system exactly presents a certain set of inferences, each of which can be traced back to axioms in the input ontology. Moreover, [2] proposed techniques which enable to decide whether a system is unsound and/or incomplete for a given test query and ontology. The result from this approach indicates that most available query rewriting systems are unsound and/or incomplete, even on commonly used benchmark ontologies. The techniques proposed also revealed the precise causes of their correctness issues and the systems were then corrected based on our feedback. In [27], the author consider positive rules, called ∀∃-rules, which the conclusion could contain existentially quantified variables, which makes reasoning assignments (such as conjunctive query answering or entailment) undecidable. ∀∃-rules have the same logical form as tuple-generating dependencies in databases and as conceptual graph rules. [27] have been presented more obvious view of the limits between decidability and non-decidability of reasoning with those rules. The author also introduces decidable classes based on backward chaining whereas previous known decidable classes were based on forward chaining. Another outcome from this approach is the definition of a backward mechanism that takes the complex structure of ∀∃-rules conclusions into account. [28] have been introduced a framework for tractable ontology querying and for a variety of other applications, called Datalog+/-. That framework enlarges plain Datalog by characteristics such as existentially quantified rule heads and limits the rule syntax thus to achieve decidability and tractability. [28] also discussed three paradigms assuring decidability: chase termination, guardedness, and stickiness. A number of languages in Datalog+/- family have been reviewed by [28] and the results showed these languages are appropriated for various jobs, such as data exchange or ontological query answering. Moreover, Datalog+/- are simple, easy to understand, easy to analyze, decidable, and they have good complexity properties, even for ontological reasoning and query answering they become to be very multipurpose and expressive. According to [14] Answering queries over an incomplete database in connection with a set of constraints is a prominent computational task with applications in various domains for example information integration and metadata management in the Semantic Web. For many DLs, query answering under constraints can be reduced to query rewriting: given a conjunctive query Q and a set of DL constraints T, the query Q can be transformed into a datalog query Q_T that takes into account the semantic consequences of T; then, to get answers to Q w.r.t. T and some (arbitrary) database instance A, one can simply evaluate Q_T over A using existing (deductive) database technology, without taking T into account. Ontology evolution is an incontestable need in ontology-based data integration. However, few research efforts have focused on addressing the need to describe the evolution of ontologies used as global schema into the underlying data integration systems. A solution that permit query answering in data integration systems on evolving ontologies without mapping redefinition provided by [29], achieved by rewriting queries between ontology versions and after that forwarding them to the underlying data integration systems to be answered. The technique proposed automatically find and represent the changes between ontology versions using a high-level language of changes which are interpreted as sound global-as view (GAV) mappings, and they are utilized so as to generate equivalent rewritings among ontology versions. There are two steps will be done when equivalent rewritings cannot be generated: guide query redefinition or provide the best over-approximations i.e., the minimally-containing and minimally-generalized rewritings [29].

4. VARIOUS ALGORITHMS OF QUERY REWRITING

Based on the discussion of section 3.1, thus we try to present a brief description of the advantages and disadvantages of using query rewriting algorithm only on 3 types of ontology, ontology contraction [17], ontology evolution [16] and ontology change [15]. According to [17], there is no previous study or research of the problem of query rewriting over contracted ontologies. The author believe that such an algorithm can be beneficial in cases where computing the rewriting for a big and complex ontology is time-consuming. In such cases, an initial rewriting can be computed once while then rewritings for contractions of the input ontology can be computed using a lightweight algorithm. Additionally, ontology contraction can also be interesting in designing ontologies for practical applications. More precisely, given an ontology O and query q the proposed method can be used to investigate which of the axioms of O affect the size of the rewriting for q; O. Algorithm Delete proposed [17] accepts as input a labelled datalog rewriting Q_D for a query Q and ontology R and a set $A \subseteq R$ of axioms to be removed from R and produces a new datalog rewriting for Q, R \ A. Furthermore, the author presented Add algorithm in [16]. Such a detailed algorithm that relies on a query-oriented or an axiom-oriented inference system. Add algorithm accepts as input the ontology O, the set of clauses R, the set of new clauses O_N = O'\O and a sound inference system Γ [16]. Hereafter, the author developed their research with implemented the algorithm that computes a new rewriting

directly from the input rewriting, which, as it does not perform any additional inferences, is independent of any system. [15] evaluated these prototypes on a range of test and commonly used large-scale ontologies. All their experiments suggest that, especially when a relatively small part of the ontology is added or removed, it would be highly beneficial for modern state-of-the-art systems to integrate our techniques. Additionally, the experiments that concern the ontology contraction show that in many cases one can compute a large part of the target rewriting by simply deleting clauses from the initial rewriting [15]. The simple summary description of advantages and disadvantages of query rewriting algorithm depicted in table 2.

Table 2. *Summary of advantages and disadvantages of query rewriting algorithm*

Refs	Algorithm proposed	Advantages of algorithm proposed	Disadvantages of algorithm proposed
[17]	Delete	applied on large ontology, it is possible to reuse the previously computed information	rewriting needs to contain all minimal subsets for a member of the rewriting. If not sufficient, it will lack information or nothing will be contracted
[16]	Add	the worst-case complexity of Add algorithm is the same with the complexity of the query rewriting system. computation time is better than Requiem	atoms that have many descendants cause computation time to increase significantly
[15]	G-Delete	can extend more ontology with new rewriting query. Ontology is not given query one by one	Looping on the Add algorithm can grow exponentially depending on input size. According to the complexity theory, something that grows exponentially means it may be inefficient [30].

Furthermore, following table show the list of query rewriting algorithm publication based on their approaches. This table described the supported of ontology language, capability and characteristic from each query rewriting algorithm on ontology change which has been proposed in the last year.

Table 1. *List of Query Rewriting Algorithm publication based on their approaches*

Refs	Year	Algorithm Name	The Supported of Ontology Language	Capability	Characteristic
[25]	2010	Presto	DL-Lite	• able to handle conjunctive queries of up to 30 atoms • providing a new upper bound on the size of the perfect reformulation of a UCQ in DL-Lite • the algorithm can be improved in the case when there are no role inclusion assertions in the TBox (e.g., for DL-LiteF TBoxes): in fact, the problem of computing perfect reformulations in this setting is significantly simplified by this assumption.	• Presto does not generate a union of conjunctive queries, but a non-recursive datalog program • Presto applies expansion rules driven by the goal of eliminating existential joins from the query based on the computation of most general subsumes of concept and role expressions, which turns out to be a much smarter strategy than previous approaches.
[14]	2010	Requiem	DL ELHIO¬	• handling constraints modeled in the DL ELHIO¬ and use it to show that answering conjunctive queries in this setting is PTime-complete w.r.t.	The algorithm deals with various description logics of the EL and DL-Lite families and is worst-case

				data complexity. • obtaining the certain answers of a conjunctive query Q over an ELHIO¬ knowledge base K	optimal w.r.t. data complexity for all of them.
[31]	2011	Mastro	DL-Lite$_A$	• providing optimized algorithms for answering expressive queries as well as features for intentional reasoning and consistency checking • providing a proprietary API, an OWLAPI compatible interface, and a plugin for the Protege 4 ontology editor • providing a comprehensive solution to such a problem by offering features both for specifying and reasoning on an ontology, and for mapping external data sources to it	• Mastro is developed in Java and can be connected to any data management system allowing for a JDBC connection, e.g., a relational DBMS. • A Java tool for ontology-based data access (OBDA)
[32]	2012	Quest	DL-Lite	• improving the overall performance of query answering systems. • creating ABox repositories that are complete w.r.t. a significant portion of DL-Lite TBoxes, including those expressed in RDFS, but where the data is not explicitly expanded. • characterizing ABox completeness by means of dependencies, and how to use these and equivalence to optimize DL-Lite TBoxes • reducing the cost of query rewriting and generating highly efficient queries.	Focus on redundancy in reasoning, in particular due to completeness of the ABox w.r.t. the TBox and the role of RDBMSs in OBQA systems.
[33]	2012	Clipper	DL-Lite and EL family	• rewriting a CQ into a union of CQs but generating (possible recursive) Datalog rules to capture the TBox. • incorporating the input CQ into the TBox before saturation, after which the TBox is translated into Datalog. • this setting not only subsumes both DL-Lite and EL, but also generates an algorithm for answering (limited) recursive queries over Horn-SHIQ ontologies (an undecidable problem for full recursive queries)	A rewriting-based algorithm for conjunctive query (CQ) answering over Horn-SHIQ ontologies
[34]	2012	Blackout	RDF	• examining the performance and scalability of producing unions of conjunctive queries versus datalog queries as rewritings. • providing an empirical comparison between two representative approaches that consider very expressive ontology languages.	Considering advantages and disadvantages of producing DQs over UCQs in terms of scalability of query asnwering
[35]	2012	Pure	Datalog+/-	• The proposed algorithm accepts any set of existential rules as input and stops for so-called finite unification sets of rules. • the rewriting step relies on a graph notion, called a piece, which allows to identify subsets of atoms from the query that must be processed together. • the rewriting method computes a minimal set of CQs when this set is finite.	focus on the backward chaining paradigm, which involves rewriting the query (assumed to be a conjunctive query, CQ) into a set of CQs (seen as a union of CQs).
[36]	2013	Ontop	OWL 2 QL,	• providing the architecture and technologies underpinning the OBDA system Ontop and taking full advantage of storing data in relational databases. • analyzing the performance of Ontop in a series of experiments and demonstrate that, for standard ontologies, queries and data stored in relational databases • despite the negative theoretical results on the worst-case OWL 2 QL query rewriting and sometimes disappointing experiences of the first OBDA systems high-performance OBDA is achievable in practice when applied to real-world ontologies, queries and data stored in relational databases. • Ontop is fast, efficient and produces SQL rewritings of high quality.	• the theoretical foundations of Ontop: the tree-witness query rewriting, T -mappings and optimizations based on database integrity constraints and SQL features.
[37]	2013	Kyrie	DL-Lite	• compiling data, dimensions and measurements that have been used to evaluate some of the most recent systems, • analyzing and characterizing these assets • providing a unified set of them that could be used as a starting point towards a more systematic benchmarking process for such systems. • applying this initial benchmark with some of the	Kyrie modifies REQUIEM by adding some engineering optimizations to speed up the process of query rewriting with the side effect of a possible reduction in the Datalog program generated when the output is set to Datalog.

				most relevant OBDA approaches	
[38]	2014	Iqaros	DL-Lite	• solving the problem of computing the rewriting of an extended query by 'extending' a previously computed rewriting of the initial query and avoiding recalculation. • computing the rewriting of a fixed query. More precisely, the query can be 'decomposed' into its atoms and then each atom can be processed incrementally. • providing detailed algorithms, several optimizations for improving the performance of query rewriting algorithm proposed	The techniques for rewriting extended queries imply a novel approach for computing a rewriting for fixed queries.
[39]	2015	Rapid	Horn-SHIQ & UOBM	• modifying the unfolding and n-shrinking rules to be applicable on clauses with equality that are a distinctive feature of Horn-SHIQ • dealing with equality reasoning	Rapid is consistently faster than Clipper. The sizes of the computed rewritings are roughly the same and differences are attributed to the different structures of the computed datalog rewritings.

Several of them, like Presto, Quest, Rapid, and IQAROS employ sophisticated optimizations in order to reduce either the size of the computed rewriting or the computation time.

5. CONCLUSION

Conjunctive query (CQ) answering is a key reasoning service for ontology-based data access. There are two approaches to view-based query processing, called query rewriting and query answering. One of the most prominent approaches to conjunctive query answering is query rewriting where a wide variety of systems has been proposed the last years. In the current paper we have studied and presented the discussion of the utilization of query rewriting on ontology change. We divide the use of query rewriting into 3 parts i.e. proposing a new algorithm, research development in description logics (DL) family and complexity of conjunctive query. Moreover, we also show the list of query rewriting algorithm publication based on their approaches.

6. REFERENCES

[1] D. Calvanese, G. De Giacomo, M. Lenzerini, and V. M.Y., What is QueryRewriting? 2000.

[2] M. Imprialou, G. Stoilos, and B. C. Grau, "Benchmarking Ontology-based Query Rewriting Systems," Twenty-Sixth AAAI Conf. Artif. Intell., pp. 779–785, 2012.

[3] B. T. Gruber, "What is an Ontology ?," pp. 1–11, 1993.

[4] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What Are Ontologies , and Why Do We Need Them ?," 1999.

[5] A. M. Khattak, R. Batool, Z. Pervez, A. M. Khan, and S. Lee, "Ontology evolution and challenges," J. Inf. Sci. Eng., vol. 29, no. 5, pp. 851–871, 2013.

[6] S. Editors, P. Bernus, and M. J. Shaw, International Handbooks on Information Systems. 2007.

[7] S. Bloehdorn, P. Haase, Y. Sure, and J. Voelker, "Ontology Evolution," Semant. Web Technol. Trends Res. Ontol. Syst., pp. 51–70, 2006.

[8] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Stanford Knowl. Syst. Lab., p. 25, 2001.

[9] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, and G. Antoniou, "Ontology change: classification and survey," Knowl. Eng. Rev., vol. 23, no. 2, pp. 117–152, 2008.

[10] L. Stojanovic and B. Motik, "Ontology evolution within ontology editors," EKAW02 Work. Eval. Ontol. Tools, pp. 53–62, 2002.

[11] U. Sattler, "Description Logics for Ontologies," pp. 1–21, 2003.

[12] A. Armando and P. Baumgartner, The Complexity of Conjunctive Query Answering in Expressive Description Logics. 2008.

[13] G. Gottlob, G. Orsi, A. Pieris, and Q. Finance, "Ontological Queries : Rewriting and Optimization," Proc. - Int. Conf. Data Eng., pp. 2–13, 2011.

[14] H. Pérez-Urbina, B. Motik, and I. Horrocks, "Tractable Query Answering and Rewriting under Description Logic Constraints," J. Appl. Log., vol. 8, no. 2, pp. 186–209, 2010.

[15] E. Tsalapati, G. Stoilos, A. Chortaras, G. Stamou, and G. Koletsos, "Query rewriting under ontology change," Comput. J., vol. 60, no. 3, pp. 389–409, 2016.

[16] E. Tsalapati, G. Stoilos, G. Stamou, and G. Koletsos, "Query Rewriting Under Ontology Evolution," 2013.

[17] E. Tsalapati, G. Stoilos, G. Stamou, and G. Koletsos, "Query Rewriting Under Ontology Contraction," 2012.

[18] A. Chortaras, D. Trivela, and G. Stamou, "Optimized query rewriting for OWL 2 QL," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 6803 LNAI, pp. 192–206, 2011.

[19] T. Venetis, G. Stoilos, and G. Stamou, "Incremental Query Rewriting for OWL 2 QL," 2012.

[20] I. Horrocks, B. Motik, H. Perez-Urbina, I. Horrocks, and B. Motik, "Efficient Query Answering for OWL 2," Iswc'09, pp. 489–504, 2009.

[21] G. Orsi, A. Pieris, and A. P. Giorgio Orsi, “Optimizing Query Answering under Ontological Constraints,” Pvldb, vol. 4, no. 11, pp. 1004–1015, 2011.

[22] A. Bundy, “First Steps in EL Contraction,” Autom. Reason. about Context Ontol. Evol., 2009.

[23] B. C. Grau, E. Kharlamov, and D. Zheleznyakov, “Ontology Contraction : Beyond the Propositional Paradise,” 2012.

[24] D. Calvanese et al., “Tractable Reasoning and Efficient Query Answering in Description Logics : The DL-Lite Family,” J. Autom. Reason., vol. 39, no. 3, pp. 385–429, 2007.

[25] R. Rosati and A. Almatelli, “Improving Query Answering over DL-Lite Ontologies,” Kr, vol. 10, no. Kr, pp. 51--53, 2010.

[26] M. M. Ribeiro, R. Wassermann, G. Antoniou, G. Flouris, and J. Pan, “Belief contraction in web-ontology languages,” CEUR Workshop Proc., vol. 519, no. August, 2009.

[27] J. F. Baget, M. Leclère, M. L. Mugnier, and E. Salvat, “On rules with existential variables: Walking the decidability line,” Artif. Intell., vol. 175, no. 9–10, pp. 1620–1654, 2011.

[28] A. Cal et al., “Datalog + / - : A Family of Logical Knowledge Representation and Query Languages for New Applications Keynote Lecture,” Proc. - Symp. Log. Comput. Sci., pp. 228–242, 2010.

[29] H. Kondylakis and D. Plexousakis, “Ontology Evolution without Tears,” J. Web Semant., vol. 19, pp. 42–58, 2013.

[30] M. Sipser, “Introduction to the Theory of Computation,” 2013.

[31] D. Calvanese et al., “The MASTRO system for ontology-based data access,” Semant. Web, vol. 2, no. 1, pp. 43–53, 2011.

[32] M. Rodrıguez-Muro and D. Calvanese, “High Performance Query Answering over DL-Lite Ontologies,” Proc. Thirteen. Int. Conf. Princ. Knowl. Represent. Reason., pp. 308–318, 2012.

[33] T.-K. T. and G. X. Thomas Eiter, Magdalena Ortiz, Mantas Simkus, “Query Rewriting for Horn-SHIQ plus Rules.pdf.” 2012.

[34] G. K. and E. S. Hector Perez-Urbina, Edgar Rodriguez-Diaz, Michael Grove, “Evaluation of Query Rewriting Approaches for OWL 2.pdf.” 2012.

[35] M. T. Melanie Konig, Michel Leclere, Marie-Laure Mugnier, “A Sound and Complete Backward Chaining Algorithm for Existential Rules.pdf.” 2012.

[36] M. Rodríguez-Muro, R. Kontchakov, and M. Zakharyaschev, “Ontology-based data access: Ontop of databases,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8218 LNCS, no. PART 1, pp. 558–573, 2013.

[37] J. Mora and O. Corcho, “Towards a systematic benchmarking of ontology-based query rewriting systems,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8219 LNCS, no. PART 2, pp. 376–391, 2013.

[38] T. Venetis, G. Stoilos, and G. Stamou, “Query Rewriting Under Query Extensions for OWL 2 QL Ontologies,” J. Data Semant., vol. 3, pp. 1--23, 2014.

[39] D. Trivela, G. Stoilos, A. Chortaras, and G. Stamou, “Query Rewriting in Horn- SHIQ (Extended Abstract),” pp. 5–8, 2015.