High-level Data Access Based on Query Rewritings

Ekaterina Stepalina
National Research University – Higher School of Economics
33/5 Kirpichnaya, st., Moscow, Russian Federation
estepalina@mail.ru

***Abstract*—This paper describes the ODBA problem solution based on query rewriting techniques, introduces the DL-Lite logics for knowledge representation and the query rewriting algorithms for high-level data access. The RQR algorithm's optimization capabilities are considered.**

Keywords- ODBA;description logic; DL-Lite; query answering, query rewriting,OWL 2 QL

I. Introduction

A conceptual interface for accessing the data stored in existing relational databases can be implemented via query rewriting techniques. Built on these techniques, the interface may be independent from DBMS and as well as from particular DB schemes[6]. The development of such interface is an actual problem of raising the abstraction level for working with data and high-level integration of information systems. The ontology representation format OWL 2 QL has been specially designed to use actual database technology for query answering via query rewriting. An efficient query rewriting algorithm RQR[1] was introduced at the international OWLED-2009 workshop: it translates queries to ontologies into the queries to ordinal databases. The data complexity of RQR is no higher than P in the worst case. The additional advantage of the algorithm is that it can be used for more expressive descriptive logics – DL-Lite and higher. This paper describes the ODBA problem, introduces DL-Lite logics and query rewriting techniques, then analyses the RQR optimization capabilities.

II. ODBA Problem

With conceptual modeling progress (OOP, UML) and system sophistication the need of providing an information system with a high-level interface for working with large amounts of data is appeared. Such interface may be provided if the knowledge domain is represented in an ontology description form (knowledge base). The data access problem though a high-level conceptual interface is called ontology-based data access (ODBA) [1]. The solution must satisfy the following requirements: 1) efficient query processing, which must be ideally executed with the same speed as the SQL queries over existing RDB, and 2) the query processing must use all advantages of relational technologies already used to store data.

III. Ontology-based Knowledge Representation

Knowledge base, KB - is the knowledge domain description saving the relationships' semantics between concepts. KB allows extracting data stored in a database (ABox), taking into account the constraints expressed at a higher conceptual level (TBox)[4]:

$$\text{KB} = \text{TBox}+\text{ABox, or } K = (T, A) \qquad (1)$$

Where TBox (T) – terminological box – the conceptual data model, for instance, Entity-Relationship;

ABox (A) – assertional box – data set stored in a database.

An ontology can be called a particular instance of KB, represented on a formal KB description language. Description logic (DL) of a special expressivity power can be used as a knowledge representation language. The expressivity power is defined by the set of axioms allowed in TBox and ABox. On the one side, the language should be as more expressive as possible to completely describe the knowledge domain. On the other side, the reasoning problems over KB must have an acceptable computational complexity.

IV. Syntax and Axioms of the DL-Lite Family

The DL-Lite[1] language family is proposed for conceptual modeling in addition to UML and ER. The DL-Lite syntax:

$$\mathrm{R} ::= \mathrm{P_k} \mid {}^{-}\mathrm{P_k}, \qquad (2)$$

$$\mathrm{B} ::= \perp \mid \mathrm{A_k} \mid \geq \mathrm{qR}, \qquad (3)$$

$$\mathrm{C} ::= \mathrm{B} \mid \neg \mathrm{C} \mid \mathrm{C_1} \sqcap \mathrm{C_2}, \qquad (4)$$

$$\mathrm{inv(R)} = \begin{cases} \mathrm{P_k^-}, \text{if } \mathrm{R} = \mathrm{P_k}, \\ \mathrm{P_k}, \text{if } \mathrm{R} = \mathrm{P_k^-} \end{cases} \qquad (5)$$

TBox is a finite set of $\mathrm{C_1} \sqsubseteq \mathrm{C_2}$, $\mathrm{R_1} \sqsubseteq \mathrm{R_2}$ - concept and role inclusion axioms.

ABox is a finite set of $\mathrm{A_k(a_i)}$, $\neg\mathrm{A_k(a_i)}$, $\mathrm{P_k(a_i, a_j)}$ and $\neg\mathrm{P_k(a_i, a_j)}$ - assertions.

Where $\mathrm{a_i}$- object name, A – concept name, P – role name, q – integer number.

Interpretation $\mathcal{I}$ (essentially, the particular instance of KB) is a pair if non-empty domain and an interpretation function

$$(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}}) : \mathrm{a_i^{\mathcal{I}}} \in \Delta^{\mathcal{I}}, \mathrm{A_k^{\mathcal{I}}} \subseteq \Delta^{\mathcal{I}} \text{ and } \mathrm{P_k^{\mathcal{I}}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \qquad (6)$$

For each interpretation the unique name assumption (UNA) status is also specified. UNA affects on the computational complexity characteristics of $\mathcal{I}$:

$$\mathrm{a_i^{\mathcal{I}}} \neq \mathrm{a_j^{\mathcal{I}}}, \text{for all } \mathrm{i} \neq \mathrm{j} \qquad (7, \text{UNA})$$

Languages of different expressive power are produced by restricting the set of allowed axioms. The main axioms:

$$(P_k^-)^{\mathcal{J}} = \{(y,x) \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} | (x,y) \in P_k^{\mathcal{J}}\}, \quad (8)$$

$$\perp^{\mathcal{J}} = \emptyset, \quad (9)$$

$$(\geq qR)^{\mathcal{J}} = \{x \in \Delta^{\mathcal{J}} | \aleph\{y \in \Delta^{\mathcal{J}} | (x,y) \in R^{\mathcal{J}} \} \geq q\}, \quad (10)$$

$$(\neg C)^{\mathcal{J}} | \Delta^{\mathcal{J}} \setminus C^{\mathcal{J}}, \quad (11)$$

$$(C_1 \sqcap C_2)^{\mathcal{J}} = C_1^{\mathcal{J}} \cap C_2^{\mathcal{J}} \quad (12)$$

Where $\aleph$ means the cardinality of the following set.

Additional axioms reflect various relationships used in conceptual modeling:

$$(\geq qR.C)^{\mathcal{J}} = \{x \in \Delta^{\mathcal{J}} | \aleph\{y \in C^{\mathcal{J}} | (x,y) \in R^{\mathcal{J}} \} \geq q\} \quad (13)$$

$$\mathcal{J} \vDash \mathrm{Dis}(R_1, R_2) \text{ iff } R_1^{\mathcal{J}} \cap R_2^{\mathcal{J}} = \emptyset, \quad (14)$$

$$\mathcal{J} \vDash \mathrm{Asym}(P_k) \text{ iff } P_k^{\mathcal{J}} \cap (P_k^-)^{\mathcal{J}} = \emptyset, \quad (15)$$

$$\mathcal{J} \vDash \mathrm{Sym}(P_k) \text{ iff } P_k^{\mathcal{J}} = (P_k^-)^{\mathcal{J}}, \quad (16)$$

$$\mathcal{J} \vDash \mathrm{Irr}(P_k) \text{ iff } (x,x) \notin P_k^{\mathcal{J}} \text{ for all } x \in \Delta^{\mathcal{J}}, \quad (17)$$

$$\mathcal{J} \vDash \mathrm{Ref}(P_k) \text{ iff } (x,x) \in P_k^{\mathcal{J}} \text{ for all } x \in \Delta^{\mathcal{J}}, \quad (18)$$

Where $\vDash$ is a satisfaction relation in KB.

The common denominators of DL-Lite logics are the following – 1) it is not possible to assign particular roles only to certain concepts, that means all roles can be applied to every concept ($\exists R.C), C = \top$; 2) TBox axioms are only concept inclusions and cannot represent any kind of disjunctive information, for instance, that several concepts cover the whole domain.

V. Main Problems of Working With Knowledge Bases

Given a KB $K = (T, A)$ one may consider the following fundamental reasoning problems [5]:

A. *Satisfiability*

Check whether a model of K exists.

B. *Instance checking*

Given an object a and a concept C, check whether $K \vDash C(a)$, or, in other words, whether $a^{\mathcal{J}} \in C^{\mathcal{J}}$ for each $\mathcal{J}$ of K.

C. *Query answering*

Given a query $q(\vec{x})$ and a tuple $\vec{a}$ of objects from A, check whether $K \vDash q(\vec{a})$, or, in other words, whether $\vec{a}$ is an answer to the $q(\vec{x})$ query w.r.t. K.

The computational complexity of these problems depends on a number of variable and fixed input parameters. The input parameters are: the TBox size, $|T|$, the ABox size, $|A|$, the $K = (T, A)$ size, the query $q(\vec{x})$ size – the number of query parameters, $|\vec{x}| = N$.

The combined and data (by the amount of data to be processed) complexity are separately considered w.r.t. reasoning problems. The data complexity is the most important in ODBA problem context, so the TBox size is considered fixed, and the query size is negligible w.r.t. the size of ABox.

VI. Efficient Query Answering in DL-Lite KBs

The maximal expressive language for conceptual modeling, for which the query answering complexity (data) will not exceed P, is $DL - Lite_{horn}^{HF}$[1]. If UNA is accepted, then query answering in $DL - Lite_{horn}^{(HF)|(HN)}$ will have the least computational complexity by the amount of data - AC^0. This feature causes a very important fact:

Given a knowledge base $K = (T, A)$ satisfying $DL - Lite_{horn}^{(HF)}$ with UNA and a conjunctive (with no disjunctions) query $q(\vec{x})$. Then $q(\vec{x})$ and TBox can be rewritten into a union of conjunctive $SQL(q(\vec{x}))$ queries over ABox only, and the answer for this new query will be sound and complete[3].

Based on this fact, query rewriting allows one to obtain a knowledge base over a traditional database, as well as to work with data at the conceptual level independently from a certain database scheme, and effectively use all advantages provided by modern relational DBMS.

VII. Query Rewriting Algorithms for OWL 2 QL and Higher

For information systems working with large amounts of data, mostly performing the query answering problems, the W3C consortium's proposed the OWL 2 QL standard. This standard based on less expressive, than $DL - Lite_{horn}^{(HF)}$, the $DL - Lite_{core}^{H}$ subset of axioms (another designation - $DL - Lite_R$). The complexity of all reasoning problems over $DL - Lite_{core}^{H}$ ontologies does not exceed polynomial. This significant restriction's been added because the equality or inequality of objects in OWL is to be specified explicitly with no UNA (or not UNA) implicit assumption. To keep the reasoning problems' complexity constant and UNA-independent for ontologies built in compliance with the OWL 2 QL standard, it's been decided not to include axioms, which allow one to define function dependencies and numeral restrictions over concepts. These axioms strongly affect the reasoning complexity, which depends on the fact whether UNA or not UNA is assumed in the ontology.

Query rewriting techniques and algorithms are intensively developed for OWL 2 QL to provide mechanisms for high-level conceptual query answering over existing databases.

Currently two algorithms have been designed and implemented[2]: CGLLR and RQR.

The CGLLR algorithm for DL-Lite has been implemented in several systems, such as QuOnto, Owlgres, ROWLKit. The RQR algorithm for DL-Lite+ was introduced in 2009 and implemented in REQUIEM. Both algorithms, CGLLR and RQR, retrieve the same results of query rewriting. During the rewriting process each algorithm produces a large number – about several thousand - UCQ (unique conjunctive query). This results in complicated SQL queries with too many unions, which can be impracticable to DBMS.

The algorithms have been tested on computers with equal configuration. The testing data included 9 ontologies of the $DL - Lite_R$ [2] expressivity level, corresponding to the OWL 2

QL profile and used in real applications, such as VICODI project, LUBM, SANAP and other.

An active optimization work on these algorithms is conducted in the following directions:

- Simplifying the initial query $q(\vec{x})$ through query subsumption check;
- Excluding UCQ, which have no corresponding OWL-RBD mappings.

The experiments[2] showed that in some cases RQR with subsumption checking generates less UCQ, than CGLLR. Moreover, unlike CGLLR, the RQR algorithm can be used for more expressive description logic languages, than DL-Lite.

In whole, RQR works more effectively than CGLLR, supports large amounts of data, complex queries and qualified existential restrictions (∃). With subsumption checking applied to initial queries both RQR and CGLLR generate an equal number of UCQ. However, the subsumption check itself takes time and practically equalizes the result efficiency of RQR and CGLLR in the worst case.

VIII. FUTURE WORK

The experiment results demonstrate that RQR is more preferable for query rewriting, than CGLLR[2].

For researching into practical usage aspects of these algorithms, first of all, one should find out how much query answering based on described query rewriting techniques is efficient on real databases. The obvious obstacle for query rewriting approach is the need of mapping a conceptual model to a particular database for each database and for each unique model. However, it is an additional abstraction layer requirement, which is inevitable to raise the abstraction level of data access interface.

In further experiments the testing data must include queries, which are to be transformed into SQL queries to real databases based on prerequisite mappings. One may suppose that the query rewriting algorithm efficiency may also significantly depend on a particular mapping representation. Currently, there are no standards and examined formalisms to define such mappings.

Further optimizations can be applied to RQR: forward and backward subsumption check, query condensation and other. Additional experiments with these optimizations are needed. Besides, full features of OWL 2 QL (especially, data types) must be supported in RQR, and a new series of experiments will be required to get reliable results of checking the RQR efficiency with the complete support for $\mathrm{DL-Lite}_{R}$.

[1] Artale, A.; Calvanese, D.; Kontchakov, R. and Zakharyaschev, M. (2009) The DL-Lite family and relations. Journal of Artificial Intelligence Research 36 (1), pp. 1-69. ISSN 1076-9757.

[2] H.P´erez-Urbina, I.Horrocks, and B.Motik. Efficient Query Answering for OWL 2. In Proceedings of the 8th International Semantic Web Conference (ISWC2009), Chantilly, Virginia, USA, 2009.

[3] H.P´erez-Urbina, B.Motik, and I.Horrocks. Tractable Query Answering and Rewriting under Description Logic Constraints. JournalofAppliedLogic, 2009.

[4] F. Baader. Logic-Based Knowledge Representation. In M.J. Wooldridge and M. Veloso, editors, Artificial Intelligence Today, Recent Trends and Developments, number 1600 in Lecture Notes in Computer Science, pages 13–41. Springer Verlag, 1999.

[5] The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2002. ISBN 0521781760. Edited by F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. F. Patel-Schneider.

[6] Semantic Future by SWUG. [Online]:http://semanticfuture.net.