An Unified Dynamic Description Logic Model for Databases: Relational Data, Relational Operations and Queries

Guoshun Hao[1] **Shilong Ma**[1] **Yuefei Sui**[2] **Jianghua Lv**[1]

[1] National Lab of Software Development Environment,
Beihang University, Beijing, China,
Email: { haogs, slma, jhlv } @nlsde.buaa.edu.cn

[2] Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China,
Email: suiyyff@hotmail.com

Abstract

The paper presents an unified Description Logic (DL) model for databases. Describing database models using DLs is a fundamental problem in many areas because it turns databases to logical systems with enriched semantics and enhanced reasoning mechanism. A typical relational database model comprises three components: relational data model, relational operations, and queries. Therefore, a DL model for databases should also unify these three components together. However, most available DL models for databases only describe one or two components of database models. After pointing out the two key factors resulting in the absence of such a model, the paper develops an extended dynamic description logic language $\mathcal{DALC}_P$ based on $\mathcal{PDLC}$ with enriched form of action modal operators. Based on $\mathcal{PALC}_P$, an unified DL models for databases is presented. It represents relational data by $\mathcal{ALC}$, represents relational operations by atomic modal operators representation; and represents queries by complex modal operators. In addition, translation functions are provided to transform arbitrary database systems into DL systems automatically. Because the model unifies data and operations together, query processing could be done based on the logical inference mechanism which is very different from other work.

Keywords: Dynamic Description Logics; Relational Databases; Query Processing; Data Integration.

1 Introduction

The paper presents an unified Description Logic (DL) model for database systems. The model unifies DL representations of relational data, of relational operations, and of user queries together.

Building a DL model for a database system is a fundamental problem in many database related applications, such as data integration (Giacomo and Lenzerini, 1995), knowledge representation, and data warehouse, etc.. Such a DL model for a database system transforms each database system into a logical system with enriched semantics and enhanced reasoning mechanism. Therefore, operations in database systems, such as query processing, could be done based on the logical inference mechanism.

Take the data integration problem as an example. The goal of data integration is to provide an uniform interface to enable users to access data resided in distributed database sources transparently. Building a model for the global virtual database is one of the core issues, and the model should have inference capabilities to answer queries by generating a "service flow" based on sub-queries to physical databases. Only if a logical model for a single database model is presented, can a logical model for the global virtual database be built based on the lower-layer database model, and can the enhanced reasoning mechanism of DLs be utilized.

A typical relational database model comprises three components: relational data model, relational operations, and queries (without loss of generality, update is not concerned in the paper). Therefore, a DL database model should also describe these three components and unify them together. Describing database using DLs has been studied since 80's (Chakravarthy et al., 1984) (Gallaire et al., 1984) and many progress have been achieved. Available works could be roughly divided into the following two classes:

(I) represented relational data by DLs, and described queries by concepts: Calvanese et al. (1998) presented a conceptual model for databases using $\mathcal{ALCQI}$, and provided reasoning mechanism to check some relevant properties of schemata; Borgida (1995) and Baader et al. (2003) presented $\mathcal{DLR}$ models for database schema and represented queries as concepts; Calvanese et al. (2002) added a conceptual layer over the database model layer for the convenience of being integrated with other applications.

(II) represented relational data by DLs, and processed queries using other logical tools(such as datalog and Horn logic): This kind of studies were mainly focused on query answering over multi-databases. Calvanese et al. (2000), Beeri et al. (1997), and Motik and Volz (2003) studied the problem of answering queries using views (Halevy, 2000) over DL Knowledge Bases (KBs): user queries were translated into logical views (Local As View or Global As View) over DL schema, and based on datalog, the views were rewritten to queries to physical databases. The details of datalog and its theoretical basics, Horn logic, were introduced in references (Ullman and Widom, 1997) (Ullman, 1997).

Although the problem have been studied for many years, such an unified model comprising above three

The paper is supported by the National 973 Project of China under the grant number 2005CB321902.

components has not been proposed. We consider that two key factors result in the absence of an unified DLs model for databases: 1) operations were not modeled as higher layer entities above the data model layer; 2) even the expressivity of dynamic DLs was not adequate for representing the operations in a database system, some extensions are needed.

Focusing on these two key issues, the paper represents operations by action modal operators and extends the form of action modal operators. The paper proposes an unified DLs database model comprising three layers consistent with relational database model: describing relational data model by $\mathcal{ALC}$; describing relational operations by atomic action modal operators; and describing queries by combined modal operators. Necessary translation functions are also provided.

Based on the model, query processing could be done in a thoroughly logical way: (1) SQL queries could be reformulated to modal operators based on the translation function; (2) by decomposing complex modal operators to atomic modal operators whose behaviors are formally defined, queries could be answered. The logical model of database systems could aid the study of data integration and other related applications.

The paper is organized as follows: after the analysis of the background of the problem in the introduction, the next section will present the preliminaries including a relational database model and a dynamic description logic language $\mathcal{PDLC}$; the extended dynamic DL $\mathcal{DALC}_P$ based on $\mathcal{PDLC}$ will be presented in section 3; an unified model for databases using $\mathcal{DALC}_P$ will be presented in section 4 in three detailed subsections; section 5 will conclude the paper and give the future work.

2 Preliminary

2.1 Relational Database Model

A typical relational database model consists three components: relational data, relational operations, and queries.

2.1.1 Relational Data

A relational data model can be defined as follows.

- A relation T is described by (U, A), where U is a non-empty universe, and A is a set of attributes. A schema of table T is $Sch(T) = A$.
- A database D is a set $\{T_1, T_2, \cdots, T_n\}$ of relations. A schema of database D is $Sch(D) = \{Sch(T_1), Sch(T_2), \ldots, Sch(T_n)\}$.

2.1.2 Relational Operations

Relational operations take relations as both inputs and outputs. Here, we give definitions for some typical relational operations.

- Union: $R \cup S = \{t \mid t \in R \vee t \in S\}$
- Difference: $R - S = \{t \mid t \in R \wedge t \notin S\}$
- Intersection: $R \cap S = \{t \mid t \in R \wedge t \in S\}$
- Cartesian Product: $R \times S = \{t_r \frown t_s \mid t_r \in R \wedge t_s \in S\}$
- Selection: $\sigma_F(R) = \{t \mid t \in R \wedge F(t) =' true'\}$
- Projection: $\pi_A(R) = \{t[A] \mid t \in R\}$
- Natural Join: $R \bowtie_B S = \{t_r \frown t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[B] = t_s[B]\}$

2.1.3 Queries

For simplicity, the paper discusses SELECT expressions defined as follows which include the core functions of SQL queries despite the simple form.

$$\textbf{select } T_1AS[,\ T_2AS] \textbf{ from } T_1[,\ T_2] \ [\textbf{where } CS]$$

where

$$\begin{array}{rcl} TAS & ::= & T.AS \mid T.* \\ AS & ::= & A \mid A; AS \\ CS & ::= & C \mid (C) \mid CS \textit{ and } C \mid CS \textit{ or } C \\ C & ::= & T.A \textbf{ comp } v \mid T.A \textbf{ comp } T.A \\ \textbf{comp} & ::= & > \mid < \mid = \mid \geq \mid \leq \mid \neq \end{array}$$

Sec.4 will focus on how to describe the SELECT expressions by modal operators, how to translate the expressions to corresponded modal operators, and how to decompose them to atomic modal operators.

2.2 Dynamic Description Logics

Description Logics have been designed to work with concepts and concept hierarchies. Their principle entities are concepts which are unary predicates, representing sets of individuals, and roles which are binary predicates, representing binary relations over individuals (Grun, 1998). In addition to atomic concepts and atomic roles (concept names and role names), users could build complex descriptions of concepts and roles (defined concepts and defined roles).

Different description logic languages are distinguished by the concept and role constructors provided, which results in different expressive capability and different computing complexity. The paper use DL language $\mathcal{ALC}$ to describe data model. More contents of DLs and $\mathcal{ALC}$ can be obtained in references (Baader et al., 2003) (Chomicki and Saake, 1998).

Description Logics were originally designed for representing static knowledge. To take into account changes in time or under certain actions and retain the simplicity of the language, it is natural to extend them by the corresponding modal operators (Emerson, 1990) (Wolter and Zakharyaschev, 1999). Wolter and Zakharyaschev (1998) introduced the language $\mathcal{PDLC}$ which combined the DL language $\mathcal{ALC}$ with dynamic logic. The syntax and semantic definitions of $\mathcal{PDLC}$ are as follows (according to the needs of the paper, some minor modifications are made).

Definition 2.1 *The **primitive symbols** of $\mathcal{PDLC}$ are:*

- *concept names $C_0,\ C_1,\ \ldots$;*
- *role names $R_0,\ R_1,\ \ldots$;*
- *object names $a_0,\ a_1,\ \ldots$;*
- *the booleans $\sqcap,\ \sqcup,\ \neg,\ \top,\ \bot$ and the quantifier $\exists R_i,\ \forall R_i$, for every role name R_i;*
- *action variables $\alpha_0,\ \alpha_1,\ \ldots$;*
- *action constructs: $;,\ \cup,\ *$.*

Definition 2.2 ***Concepts, roles, action terms, and formulas** are defined as: all concept names as well as $\top,\ \bot$ are atomic concepts. All role names are atomic roles. Every action variable is an atomic action. If C and D are concepts, R a role, a and b object names, φ and ψ formulas, α and β action terms, then*

- *$C \sqcap D,\ C \sqcup D,\ \neg C,\ \exists R.C,\ \forall R.C,\ [\alpha]C$ are concepts;*

- $C(a),\ R(a,b),\ C \equiv D,\ C \sqsubseteq D,\ \varphi \sqcap \psi,\ \neg\varphi,\ [\alpha]\varphi$;
- $\alpha;\beta,\ \alpha \cup \beta,\ \alpha^*$ *are action terms.*

Remark 2.1 *The action term α in $\mathcal{PDLC}$ denotes a class of actions and $[\alpha]$ denotes an action instance. The paper will extend the form of α to represent operations in databases.*

The semantics of pure description part of $\mathcal{PDLC}$, i.e. $\mathcal{ALC}$, was defined by the following interpretation:

Definition 2.3 *An **interpretation** is defined as*

$$\mathcal{I} = \langle \Delta,\ a_0^{\mathcal{I}},\ \ldots,\ C_0^{\mathcal{I}},\ \ldots,\ R_0^{\mathcal{I}},\ \ldots \rangle$$

where Δ is a non-empty set of objects, $a_i^{\mathcal{I}} \in \Delta$ interprets the object name a_i, $R_i^{\mathcal{I}}$ is a binary relation on Δ (i.e. $R_i^{\mathcal{I}} \subseteq \Delta \times \Delta$) interpreting the role name R_i, $C_i^{\mathcal{I}}$ a subset of Δ (i.e. $A^{\mathcal{I}} \subseteq \Delta$) interpreting the concept name C_i.

The semantics of the dynamic component of $\mathcal{PDLC}$ was interpreted based on the following definitions of Frame and Model.

Definition 2.4 (Frame) *A Kripke **Frame** is defined as*

$$\mathcal{F} = \langle W, \triangleright_{\alpha_0},\ \triangleright_{\alpha_1},\ \ldots \rangle \qquad (1)$$

where W is a non-empty set of worlds, and $\triangleright_{\alpha_i}$ is a binary accessibility relation on W corresponding to the action variable α_i.

Definition 2.5 *A **model** of $\mathcal{PDLC}$ based on a frame $\mathcal{F}$ of the form (1) is a pair $\mathcal{M} = \langle \mathcal{F}, I \rangle$ in which I is a function associating with each world $w \in W$ an $\mathcal{ALC}$-interpretation*

$$I(w) = \langle \Delta,\ R_0^{\mathcal{I},w}, \ldots,\ C_0^{\mathcal{I},w}, \ldots,\ a_0^{\mathcal{I},w}, \ldots \rangle \qquad (2)$$

such that $\alpha_i^{\mathcal{I},u} = \alpha_i^{\mathcal{I},v}$ for any $u, v \in W$. Note that the set of object Δ is the same for every world in W, it is called the domain of $\mathcal{M}$.

Definition 2.6 *Given a $\mathcal{PDLC}$-model $\mathcal{M} = \langle \mathcal{F}, I \rangle$ and a world w, the value $C_i^{\mathcal{I},w}$ of a concept C in w, the truth-relation $(M, w) \models \varphi$, and the relation $\triangleright_\alpha$ are defined inductively as follows:*

$$\begin{aligned}
\top^{\mathcal{I},w} &= \Delta \\
\bot^{\mathcal{I}} &= \phi \\
(\neg C)^{\mathcal{I}} &= \Delta \backslash C^{\mathcal{I},w} \\
(C \sqcap D)^{\mathcal{I},w} &= C^{\mathcal{I},w} \cap D^{\mathcal{I},w} \\
(C \sqcup D)^{\mathcal{I},w} &= C^{\mathcal{I},w} \cup D^{\mathcal{I},w} \\
(\exists R.C)^{\mathcal{I},w} &= \{a \in \Delta \mid \exists b.(a,b) \in R^{\mathcal{I},w}\} \\
(\forall R.C)^{\mathcal{I},w} &= \{a \in \Delta \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I},w}\} \\
([\alpha]C)^{\mathcal{I},w} &= \{a \mid \forall v \in W\ (w \triangleright_\alpha v \Rightarrow a \in \Delta)\} \\
w \models C \equiv D &\Leftrightarrow (C^{\mathcal{I},w} \subseteq D^{\mathcal{I},w})\ \&\ (D^{\mathcal{I},w} \subseteq C^{\mathcal{I},w}) \\
w \models C \sqsubseteq D &\Leftrightarrow C^{\mathcal{I},w} \subseteq D^{\mathcal{I},w} \\
w \models C(a) &\Leftrightarrow a^{\mathcal{I},w} \in C^{\mathcal{I},w} \\
w \models R(a,b) &\Leftrightarrow (a^{\mathcal{I},w}, b^{\mathcal{I},w}) \in R^{\mathcal{I},w} \\
w \models \neg\varphi &\Leftrightarrow w \not\models \varphi \\
w \models \varphi \wedge \psi &\Leftrightarrow w \models \varphi\ \&\ w \models \psi \\
w \models [\alpha]\varphi &\Leftrightarrow \forall v \in W\ (w \triangleright_\alpha v \Rightarrow v \models \varphi) \\
\triangleright_{\alpha;\beta} &= \triangleright_\alpha \circ \triangleright_\beta \\
\triangleright_{\alpha \cup \beta} &= \triangleright_\alpha \cup \triangleright_\beta \\
\triangleright_{\alpha^*} &= \triangleright_\alpha^*
\end{aligned}$$

3 $\mathcal{DALC}_P$ – Extending $\mathcal{PDLC}$ by Parameters

To represent actions in dynamic systems like databases, we need to give the detailed form of the action terms and make some extensions to describe various aspects of attributes of actions. For example, if we represent queries using action modal operators α, the query conditions should be described in α. If we denote relational operations using action terms α, also some operation attributes should be described by α, we should extend the simple form of α to represent operation attributes.

The paper extends $\mathcal{PDLC}$ to $\mathcal{DALC}_P$ by increasing the expressivity of action terms from α to $\alpha(P_0, \ldots)$ where $P_0, \ldots$ denote parameters for the action α. The syntax and semantic definitions are defined as follows based on that of $\mathcal{PDLC}$.

3.1 Syntax of Action Terms in $\mathcal{DALC}_P$

Definition 3.1 ***Atomic Action terms*** *α in $\mathcal{DALC}_P$ is defined as follows:*

$$\begin{aligned}
\alpha \ ::= \ & \varnothing \mid 1 \mid -(C) \mid \sqcup(C) \mid \sqcap(C) \mid \bowtie(C) \\
& \mid \times(C) \mid ?(CS) \mid \pi(AS) \mid \sigma(CS)
\end{aligned}$$

where the action terms are provided to represent the relational operations in database systems:

- *$\varnothing$ denotes an action of creating an empty concept, and 1 denotes an action of retrieve the same concept as the applied concept;*
- *$\sqcup$, $\sqcap$, $-$, $\times$, $\bowtie$ denote the relational operations of union, intersection, difference, times, and join respectively;*
- *π, σ denote projection and selection, and ? denotes an action of checking if a concept meet some conditions.*
- *and CS, C, AS are parameters, where C denotes record concepts and CS, AS denotes condition set and attribute set whose definitions are provided in Sec.2.1.3.*

Definition 3.2 *If α, β are atomic action terms, then $(\alpha),\ \alpha;\beta,\ \alpha \cup \beta,\ (\alpha)^*$ are **complex action terms**. Both atomic action terms and complex action terms are **action terms** (Complex Action Modal Operator Concept).*

Definition 3.3 *Assume that each parameter P_i in the action term $\alpha(P_0, \ldots)$ is assigned a value p_i, we say that the action term generates an action instance $\alpha(p_0, \ldots)$, and we call the instance an **action (or an action modal operator)**.*

Here, symbol P_i denotes a signature of a parameter while p_i denotes concrete value.

3.2 Semantic Definitions

The semantic definitions of $\mathcal{DALC}_P$ are basically same with that of $\mathcal{PDLC}$ except two aspects:

1) the binary accessibility relations should correspond to each action $[\alpha(pSet)]$, other than action term $[\alpha(PSet)]$;

2) the object domain of each world may be variable, therefore, we should use Δ^w denoting the domain of world w.

Here, we use symbol $m_0,\ m_1,\ \ldots$ to represent action modal operators corresponding to each action instances, the frame and the interpretation function definitions are modified to :

$$\mathcal{F} = \langle W, \triangleright_{m_0}, \triangleright_{m_1}, \ldots \rangle \quad (3)$$

$$I(w) = \langle \Delta^w, R_0^{\mathcal{I},w}, \ldots, C_0^{\mathcal{I},w}, \ldots, a_0^{\mathcal{I},w}, \ldots \rangle \quad (4)$$

and the satisfactions relating to modal operators and accessible relations are modified to or added by:

$$\begin{aligned} (m_i C)^{\mathcal{I},w} &= \{a | \ \forall v \in W \ (w \triangleright_{m_i} v \Rightarrow a \in \Delta^w)\} \\ w \models m\varphi &\Leftrightarrow \forall v \in W \ (w \triangleright_m v \Rightarrow v \models \varphi) \\ \triangleright_{(\alpha)} &= \triangleright_\alpha \end{aligned}$$

4 An Unified $\mathcal{DALC}_P$ Model for Database

The logical tools prepared, this section will focus on how to utilize $\mathcal{DALC}_P$ to build up an unified database model. Corresponding to the relational database model, the model should include three components and unify them together. In addition, two translation mechanisms will be also provided which could automatically transform DBs into KBs and queries to modal operators respectively.

4.1 Modeling Relational Data

4.1.1 $\mathcal{ALC}$ Data Model.

We use $\mathcal{ALC}$ to represent relational data. As the basic instance, a record individual is described by its value in each attribute which is described by the corresponding role. A table concept represents a set of records contained by the table, and a database concept represents a set of records contained by the database.

* concept C_R represents all record instances included in the model;
* associate with every single record r a record concept C_r which has only one instance i_r;
* associate with every table t a table concept C_t, such that $\forall r \in t, \ C_r \sqsubseteq C_t$;
* associate each attribute a of table a role R_a represents the binary relations between record individuals and attribute values;
* associate with every database d a database concept C_d, such that $\forall t \in d, \ C_t \sqsubseteq C_d$

In a word, the model represents relational data in the concept hierarchy: record concepts C_r, table concepts C_t, database concepts C_d, and the top concept of records C_R.

4.1.2 Translation Function from Database to $\mathcal{ALC}$ KB.

The translation $\sigma(D)$ from a database to a DL knowledge base is defined as follows:

Given a database d,

- $\sigma(d)$ includes a record concept C_R, whose instances are all records contained in the model;
- $\sigma(d)$ includes a concept C_d, whose instances are all the records contained in database d;
- $\forall t \in d$
  – $\forall a \in t$, a is translated to a role R_a;
  – t is translated to a concept C_t whose instances are all the records contained in table t:
  if $A_t = \{a1, a2, \cdots, an\}$ then C_t has definition: $C_t \sqsubseteq \exists R_{a1}.\top \sqcap \exists R_{a2}.\top \sqcap \cdots \exists R_{an}.\top$;
- $\forall r \in t$
  – r is translated into a concept C_r which has an instance i_r;
  – $\forall a \in t : (r(a) = v)$, $r(a) = v$ is translated into an assertion $R_a(i_r, v)$;
  – if $r(a_1) = v_1, \ldots, r(a_n) = v_n$, then $C_r \equiv \exists R_{a_1}.v_1 \sqcap \exists R_{a_2}.v_2 \sqcap \cdots \exists R_{a_n}.v_n$.

In the resulted KB, the concepts satisfy the subsumption relation: $\forall r, t, d(r \in t \in d \iff i_r \in C_r \sqsubseteq C_t \sqsubseteq C_d \sqsubseteq C_R)$

Once the translation function $\rho_1 : DB \to KB$ is formally defined, an arbitrary database could be automatically transformed into a DL KB.

4.2 Modeling Relational Operations

In relational database model, relational operations are important because upper layer operations, like queries, updates, and modifies, could be represented by the combinations of relational operations. To build an unified DL database model, representing relational operators is also a key issue. This subsection provides representations for relational operations using above atomic action modal operators. Notice that the domain is not the same as the world changes since some actions create new instances(records) and new KBs.

4.2.1 Basic Modal Operators for Record-Level Operations

Before describing relation operations, we define some low-level modal operators. These basic modal operators will be applied to *record concepts* to provide *record-level* operations on the KB. We regard them as low level operations which are necessary to the DL database model.

I $[\varnothing]C_r = \varnothing$

II $[1]C_r = C_r$

III Join Two Records, $[\bowtie (C_R)]C_R \to C_R$
Join two source records to one new record. Here, we assume that there is no same attribute belonging to both source records.
$(C_{r_1} \equiv \exists R_{a_1}.v_1 \sqcap \ldots \sqcap \exists R_{a_m}.v_m) \wedge$
$(C_{r_2} \equiv \exists R_{b_1}.u_1 \sqcap \ldots \sqcap \exists R_{b_n}.u_n) \wedge$
$([\bowtie (C_{r_2})]C_{r_1} = C_{r_{12}}) \to$
$C_{r_{12}} \equiv \exists R_{a_1}.v_1 \sqcap \ldots \sqcap \exists R_{a_m}.v_m \sqcap R_{b_1}.u_1 \sqcap \ldots \sqcap \exists R_{b_n}.u_n$

IV Projection over a Record, $[\pi(as)]C_R \ \to C_R$
$(C_r \equiv \exists R_{a_1}.v_1 \sqcap \ldots \sqcap \exists R_{a_n}.v_n) \wedge$
$([\pi(as)]C_r = C_{r'}) \wedge$
$(as = a_1; \ldots; a_k; b_1; \ldots; b_j, k < n) \to$
$C_{r'} \equiv \exists R_{a_1}.v_1 \sqcap \cdots \exists R_{a_k}.v_k$

V Condition Check On a Record, $[?(CS)]C_R \to \{'true', 'false'\}$
According to the different forms of the condition set cs, the operation behavior is defined respectively as follows:
(a) $[?\ (cs = \ t.a \ \mathbf{comp} \ v)]C_r =$
$(C_r \sqsubseteq C_t) \wedge (\exists R_a \exists v'(R_a(i_r, v'))) \wedge (v' \ \mathbf{comp} \ v)$
(b) $[?(t.a1 \ \mathbf{comp} \ t.a2)]C_r =$
$(C_r \sqsubseteq C_t) \wedge (\exists R_{a_1} \exists v1(R_{a_1}(i_r, v_1))) \wedge$
$(\exists R_{a_2} \exists v_2(R_{a_2}(i_r, v_2))) \ \wedge \ (v1 \ \mathbf{comp} \ v2)$
(c) $[?(t1.a1 \ \mathbf{comp} \ t2.a2)]C_r =' false'$
(d) $[?(cs' \ \mathbf{and} \ c)]C_r = ([?(cs')]C_r) \ \wedge \ ([?(c)]C_r)$
(e) $[?(cs' \ \mathbf{or} \ c)]C_r = ([?(cs')]C_r) \ \vee \ ([?(c)]C_r)$
(f) $[?((cs'))]C_r = [?(cs')]C_r$

VI Selection Over a Record: $[\sigma(CS)]C_R \rightarrow C_R$
$[\sigma(CS)]C_r = C_r \quad$ if $[?(CS)]C_r =' true'$
$[\sigma(CS)]C_r = \bot \quad$ else

4.2.2 Modal Operators for Relational Operations.

With record-level operations defined, we can define modal operators for table-level relational operations. Here, we present modal operators for the frequently used relational operations.

Notice that for operators that has a table concept as parameter and has a table concept as applied concept, such as union, intersection, difference, and cartesian times, we assume that the two operated tables have the same attribute set.

1. $[\varnothing]C_t = \varnothing$
2. $[1]C_t = C_t$
3. Union: $[\sqcup(C_{t2})]C_{t1} \sqsubseteq C_{t1} \sqcup C_{t2}$
4. Intersection: $[\sqcap(C_{t2}]C_{t1} \sqsubseteq C_{t1} \sqcap C_{t2}$
5. Difference $[-(C_{t2}]C_{t1} \sqsubseteq C_{t1} \sqcap (\neg C_{t2})$
6. Cartesian Times: $[\times(C_T)]C_T \rightarrow C_T$
   $([\times(C_{t_2}]C_{t_1} = C_{t1_2}) \wedge$
   $(C_{t_1} \sqsubseteq \exists R_{a_1}.\top \sqcap \ldots \sqcap \exists R_{a_m}.\top) \wedge$
   $(C_{t_2} \sqsubseteq \exists R_{b_1}.\top \sqcap \ldots \sqcap \exists R_{b_n}.\top) \rightarrow$
   $C_{t_{12}} \sqsubseteq \exists R_{a_1}.\top \sqcap \ldots \exists R_{a_m}.\top \sqcap \exists R_{b_1}.\top \sqcap \ldots \exists R_{b_n}.\top$
7. Projection on a Table: $[\pi(AS)]C_T \rightarrow C_T$
   $(C_t \sqsubseteq \exists R_{a_1}.\top \sqcap \ldots \exists R_{a_n}.\top) \wedge$
   $([\pi(as)]C_t = C_{t'}) \wedge$
   $(as = a_1; a_2; \cdots; a_k; b_1; b_2; \ldots; b_j, k < n) \rightarrow$
   $C_{t'} \sqsubseteq \exists R_{a_1}.\top \sqcap \cdots \exists R_{a_k}.\top$
8. Selection on a Table: $[\sigma(CS)](C_T) \rightarrow C_T$
   $[\sigma(cs)]C_t = \{[\sigma(cs)]C_r \mid \forall C_r \sqsubseteq C_t\}$
9. Selection on a Database: $[\sigma(CS)]C_D \rightarrow C_D$
   $[\sigma(cs)]C_d = \{[\sigma(cs)]C_t \mid \forall C_t \sqsubseteq C_d\}$
10. Natural Join: $[\bowtie_B (C_T)]C_T \rightarrow C_T$
    $(C_{t_1} \sqsubseteq \exists R_{a_1}.\top \sqcap \ldots \exists R_{a_n}.\top) \wedge$
    $(C_{t_2} \sqsubseteq \exists R_{b_1}.\top \sqcap \ldots \exists R_{b_m}.\top) \wedge$
    $([\bowtie_B (C_{t_2})]C_{t_1} = C_{t_{12}}) \rightarrow$
    $C_{t_{12}} \sqsubseteq \exists R_{a_1}.\top \ldots \exists R_{a_n}.\top \sqcap \exists R_{b_1}.\top \ldots \exists R_{b_m}.\top$

4.3 Modeling Queries

With atomic modal operators defined, including the low level operators and modal operators for relational operations, we will define Modal Operators for Queries (MOQ) which are complex modal operators. Firstly, we will give the syntax of MOQ, secondly we will give the translation function from SELECT query expression presented in Sec. 2.1.3 to the modal operators, and after that, we will provide the mechanism of decomposing MOQ to atomic modal operators.

4.3.1 Syntax of MOQ

Definition 4.1 (Modal Operators for Queries) *Modal operators for queries is defined as follows:*

$$
\begin{aligned}
MOQ \ ::= \ & [\textbf{\textit{Sel}}\ (Cont,\ SrcT)] \\
& \mid [\textbf{\textit{ConSel}}(Cont,\ SrcT,\ CS)] \\
& \mid [\textbf{\textit{USel}}\ (ContLst,\ SrcT)] \\
& \mid [\textbf{\textit{ConUSel}}\ (ContLst,\ SrcT,\ CS)]
\end{aligned}
$$

where

- [***Sel***] *represents selection(query) over a single table concept;*
- [***USel***] *represents conjunction queries over two table concepts(queries over more than two tables could be induced to queries over two tables);*
- [***ConSel***] *and* [***ConUSel***] *represent* [***Sel***] *and* [***USel***] *selections with condition check enabled.*
- *Cont represents attribute set of a single table, ContLst represents attribute sets of one or two tables:*
  $Cont ::= Table.AttrSet \mid Table.*$
  $ContLst ::= Cont \mid Cont; Cont$
- *SrcT denotes source table concept(s):*
  $SrcT ::= Table \mid Table; Table$

Remark 4.1 *1. The form of MOQ,* [***ConUSel***]*, is the general form of all MOQ, all other forms can be degenerated from it.*

2. Another thing need to pointed out is why we introduce MOQ to the model. Just like complex concept names are used to represent the long and complex concept definitions, MOQ assign a symbol name for complex modal operators for the same concern. In addition, MOQ has similar form with SELECT clauses, we can take MOQ as a bridge between the SELECT clause and the atomic modal operators, i.e., we can translate SELECT clause to MOQ, and then we can decompose the MOQ to atomic modal operators. The following content will focus on these two interesting issues.

4.3.2 Translation from SELECT Clauses to MOQ

In addition to describing SELECT expressions using modal operators, a translation function translating SELECT expressions automatically to MOQ can be formulated. Based on the formal expressions of SELECT clause in Sec.2.1.3 and the syntax form of MOQ in Sec.4.3.1, we define the translation function $\rho_2 : \textbf{SelectClause} \rightarrow MOQ$ as follows.

$$
\begin{aligned}
&\rho_2(select\ tas\ from\ t') = [\textbf{Sel}(tas,\ t')] \\
&\rho_2(select\ tas\ from\ t\ where\ cs) = \\
&\qquad [\textbf{ConSel}(tas,\ t,\ cs)] \\
&\rho_2(select\ t1as, t2as\ from\ t1,\ t2) = \\
&\qquad [\textbf{Usel}(t1as; t2as,\ t1; t2)] \\
&\rho_2(select\ t1as, t2as\ from\ t1, t2\ where\ cs) = \\
&\qquad [\textbf{ConUSel}(t1as; t2as, t1; t2,\ cs)]
\end{aligned}
$$

Based on the translation, all the core SQL query clauses can be translated into MOQ. All queries can be done in relational database models can also be done in this model.

4.3.3 Decomposing MOQ to Atomic Modal Operators.

After we have translated SQL queries to MOQ, we should give the behavior definitions of MOQ by decomposing them to atomic modal operators whose action behaviors have been defined in Sec.4.2.

- $[\textbf{Sel}(t.as,\ t)]C_d = [\pi(as)]C_t$
- $[\textbf{USel}(t1as; t2as,\ t1; t2)]C_d =$
  $[\textbf{Sel}(t12.(t1as; t2as),\ t12)]C_{t12}$, where
  $C_{t12} = [\times(C_{t2})]C_{t1}, (C_{t1} \sqsubseteq C_d) \wedge (C_{t2} \sqsubseteq C_d)$

- $[\mathbf{ConSel}(tas,\ t,\ cs)]C_d = ([\mathbf{Sel}(tas,\ t)] \circ [\sigma(cs)])C_d$
- $[\mathbf{ConUSel}(t1as; t2as,\ t1; t2,\ cs)]C_d = [\mathbf{ConSel}(t12.t1as; t2as, t12, cs)]C_{12}$, where $C_{t12} = [\times(C_{t2})]C_{t1}, (C_{t1} \sqsubseteq C_d) \wedge (C_{t2} \sqsubseteq C_d)$

5 Conclusion and further work

The paper presents an unified DLs model for databases, which unifies model for relational data, model for relational operations, and model for queries together. This is an interesting work that has not been done before, especially representing queries by modal operators.

After reviewed the literature, the paper points out two key issues of building up an unified DL model for a databases system: modeling queries as action modal operators, and extending the expressivity of actions of available dynamic DLs. Modeling queries as action model operators provides important features: the resulted model is consistent with the three layer structure of relational database model; the modal is a pure Description Logic system; it could utilize the reasoning mechanism to decompose modal operators (queries) to atomic modal operators(relational operators).

Tab.1 shows the correspondence between two models, it also shows the work of the paper which are closed by boxes: all three layers is modeled and translations are formulated. The proposed model represents relational data using $\mathcal{ALC}$, and, based on the provided translation function, existed databases could be translated to DL KBs automatically. As the foundation of relational databases, relational operations are described using atomic action modal operators. Based on atomic modal operators and modal constructors, complex modal operators MOQ is provided to represent queries. We also provide the translation function from SELECT expressions to MOQs and the decomposing mechanism from MOQ to atomic modal operators.

Table 1: Correspondence Between Two Models

Relational DB Model		*A DL DB Model*
Queries	$\Rightarrow$	Complex Modal Operators
Relational Operations	$\rightarrow$	Atomic Modal Operators
Relational Data Model	$\Rightarrow$	$\mathcal{ALC}$ Model

The model is a pure Description Logical model, query processing could be done based on inferences in DLs, which is very different from other works. In conclusion, the paper provide a mechanism of transforming a database systems into a DL system.

Certainly, the paper provides only a framework, and some details need to be studied in particular applications. Take data integration as an example, we may need to extend the model to describe the model of distributed databases other than that of a single database; we may need to enrich the model by expressing more constraints (Calvanese et al., 1998) (Baader et al., 2003); and more importantly, we may need to develop equal or other relations between concepts or attributes of different databases.

References

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. and Patel-Schneider, P. F., eds (2003), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press.

Beeri, C., Levy, A. Y. and Rousset, M.-C. (1997), Rewriting queries using views in description logics., *in* 'PODS', pp. 99–108.

Borgida, A. (1995), 'Description logics in data management.', *IEEE Trans. Knowl. Data Eng.* **7**(5), 671–682.

Calvanese, D., Giacomo, G. D. and Lenzerini, M. (2000), Answering queries using views over description logics knowledge bases., *in* 'AAAI/IAAI', pp. 386–391.

Calvanese, D., Giacomo, G. D. and Lenzerini, M. (2002), Description logics for information integration., *in* 'Computational Logic: Logic Programming and Beyond', pp. 41–60.

Calvanese, D., Lenzerini, M. and Nardi, D. (1998), Description logics for conceptual data modeling., *in* (Chomicki and Saake, 1998), pp. 229–263.

Chakravarthy, U. S., Fishman, D. H. and Minker, J. (1984), Semantic query optimization in expert systems and database systems., *in* 'Expert Database Workshop', pp. 659–674.

Chomicki, J. and Saake, G., eds (1998), *Logics for Databases and Information Systems (the book grow out of the Dagstuhl Seminar 9529: Role of Logics in Information Systems, 1995)*, Kluwer.

Emerson, E. A. (1990), Temporal and modal logic., *in* 'Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)', pp. 995–1072.

Gallaire, H., Minker, J. and Nicolas, J.-M. (1984), 'Logic and databases: A deductive approach.', *ACM Comput. Surv.* **16**(2), 153–185.

Giacomo, G. D. and Lenzerini, M. (1995), What's in an aggregate: Foundations for description logics with tuples and sets., *in* 'IJCAI (1)', pp. 801–807.

Grun, G. A. (1998), 'Description logics', Available at: http://www.cs.sfu.ca/people/GradStudents/grun/personal/s1.ps.

Halevy, A. Y. (2000), 'Theory of answering queries using views.', *SIGMOD Record* **29**(4), 40–47.

Motik, B. and Volz, R. (2003), Optimizing query answering in description logics using disjunctive deductive databases., *in* 'KRDB'.

Ullman, J. D. (1997), Information integration using logical views., *in* F. N. Afrati and P. G. Kolaitis, eds, 'ICDT', Vol. 1186 of *Lecture Notes in Computer Science*, Springer, pp. 19–40.

Ullman, J. D. and Widom, J. (1997), *A First Course in Database Systems*, Prentice Hall, Inc.

Wolter, F. and Zakharyaschev, M. (1998), Dynamic description logics., *in* M. Zakharyaschev, K. Segerberg, M. de Rijke and H. Wansing, eds, 'Advances in Modal Logic', CSLI Publications, pp. 431–446.

Wolter, F. and Zakharyaschev, M. (1999), 'Modal description logics: Modalizing roles.', *Fundam. Inform.* **39**(4), 411–438.