The Description Logic for Relational Databases

Ma Yue, Shen Yuming, Sui Yuefei, Cao Cungen

The description logic for relational databases [1]

Ma Yue[1,2], Shen Yuming[1,2], Sui Yuefei[1], Cao Cungen[1]

[1] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China

[2] Graduate University of Chinese Academy of Science, Beijing, 100039, China

Abstract: Description logics are widely used to express structured data and provide reasoning facility to query and integrate data from different databases. This paper presents a many-sorted description logic $\mathcal{MDL}$ to represent relational databases. We give a translation from relational databases to the description logic $\mathcal{MDL}$, and show this translation completely and faithfully captures the information in the relational database. Moreover, we show that some relational algebra operations could be expressed in $\mathcal{MDL}$.

Keywords: Relational database, Description logic, Translation

1. Introduction

The Relational model is an important theoretic model of database management system. In a relational database, data are manipulated as a collection of relations or tables. In addition, a relational database provides a collection of methods to access the data in tables, and to modify and combine tables by using a set of relational algebra operations.

Description logic is a logic-based formalism of knowledge, focusing on concepts, individuals, roles, and the relationship between them. It is a proper choice to represent relational databases in DL, using its reasoning facility to implement the query of the information in databases, and to integrate different sources of data. There are many papers on represent structured databases in DL. Calvanese et al.[2] proposed a translation from ER-model to a description logic $\mathcal{DLR}$. In this translation, tuple entities are implemented as constants in the model of $\mathcal{DLR}$ and the sets of entities are regarded as concepts. Moreover, the values of attributes are also described as constants, and there are concepts for attribute values. Unfortunately, in $\mathcal{DLR}$ these two kind of concepts could not be syntactically distinguished. There should be statements in $\mathcal{DLR}$ with form "an attribute value a is an instance of entity-concept R", which is meaningless in the corresponding databases.

Similar to Calvanese's work, we describe both the sets of attribute values and the sets of tuples in a relational database as concepts in DL. To distinguish

[1] The work is supported by the National Natural Science Foundation of China (60496326, 60573063, 60573064 and 60773059), the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z325.

these two kind of objects, we propose a many-sorted description logic $\mathcal{MDL}$, in which the constants of attribute value and the constants of tuples are taken as different types of constant symbols. Similarly, we use different types of concept names representing concepts for attribute values and tuples. Based on the definition of $\mathcal{MDL}$, we shall present a translation to represent a relational database in $\mathcal{MDL}$: from a given database, we construct an $\mathcal{MDL}$ model and a knowledge base containing $\mathcal{MDL}$ statements which are satisfied in the $\mathcal{MDL}$ model. We will show that this translation completely and faithfully captures the information in the relational database. Moreover, we show that some relational algebra operations could be expressed in $\mathcal{MDL}$ using concept and role constructors.

This paper is organized as follows: in section 2 we give a brief introduction on relational databases and description logics; in section 3 we propose a many-sorted description logic $\mathcal{MDL}$, and give its formal definitions ; in section 4 we present the translation from relational database to $\mathcal{MDL}$, and express relational algebra operations in $\mathcal{MDL}$; in the last section we conclude the paper.

2. Preliminaries

This section will give the basic definitions on relational databases and description logics. The relational model is an important theoretic model of database management system. We present a brief formal definitions in relational database and relational algebra [7,8]. Let U be a set of attributes called an universe. For each attribute $A \in U$, we assign a set of values $\mathrm{dom}(A)$ called the domain of A. Let $\mathrm{dom} = \bigcup_{A \in U} \mathrm{dom}(A)$.

Definition A *relation schema* R on U is a subset of U. A *database schema* D is a set of relation schema. □

Definition Let $D = \{R_1, ...R_n\}$ be a database schema, R be a relation in D, and X be a set of attributes in U. An *X-tuple* t is a mapping from X to dom, such that for each $A \in X$, $t(A) \in \mathrm{dom}(A)$. A *relation* r over relation schema R is a finite set of R-tuples. A *database* d over database schema D is a set of relations $\{r_1, ..., r_n\}$, each r_i is a relation over R_i. We use $\alpha(r)$ to denote the set of attributes of r. Given a database $d = \{r_1, ..., r_n\}$, each $r_i \in \{r_1, ..., r_n\}$ is an *atomic relation* in d.

□

Based on the definition of tuples and relations, we now give the definition of the relational algebra. The relational algebra is an algebra on the set of relations, including several primitive operations on relations. We give the formal definitions of these operations as follows:

Definition Let t be an X-tuple and $Y \subseteq X$. The projection of tuple t on Y, denoted $t[Y]$, is the restriction of t on Y.

Given relations r_1, r_2 and a set of attributes Y, let $\alpha(r_1) = X_1, \alpha(r_2) = X_2$,

- the *projection* of r_1 on Y, denoted $\pi_Y(r_1)$, fulfills:
  i) $\pi_Y(r_1)$ is defined if $Y \subseteq X_1$. If $\pi_Y(r_1)$ is defined, $\alpha(\pi_Y(r_1)) = Y$,
  ii) $\pi_Y(r_1) = \{t[Y] | t \in r_1\}$.
- the *natural join* of r_1, r_2, denoted $r_1 \bowtie r_2$, fulfills:

i) $\alpha(r_1 \bowtie r_2) = X_1 \cup X_2$,
ii) $r_1 \bowtie r_2 = \{t | t$ is $X_1 \cup X_2$-tuple, such that $t[X_1] \in r_1, t[X_2] \in r_2\}$.
• the *union* of r_1, r_2, denoted $r_1 \cup r_2$, fulfills:
i) $r_1 \cup r_2$ is defined if $X_1 = X_2$. If $r_1 \cup r_2$ is defined, $\alpha(r_1 \cup r_2) = X_1$,
ii) $r_1 \cup r_2 = \{t | t \in r_1$ or $t \in r_2\}$.
• the *difference* of r_1, r_2, denoted $r_1 - r_2$, fulfills:
i) $r_1 - r_2$ is defined if $X_1 = X_2$. If $r_1 - r_2$ is defined, $\alpha(r_1 - r_2) = X_1$,
ii) $r_1 - r_2 = \{t | t \in r_1$ and $t \notin r_2\}$.
• the *renaming* of r_1 from A to B, denoted $\rho_{B|A}(r_1)$, fulfills:
i) $\rho_{B|A}(r_1)$ is defined if $A \in X_1, B \notin X_1$. If $\rho_{B|A}(r_1)$ is defined, $\alpha(\rho_{B|A}(r_1)) = X_1 - \{A\} \cup \{B\}$,
ii) $\rho_{B|A}(r_1) = \{t |$ there exists $t' \in r_1$ such that $t'[A] = t[B]$, $t'[X_1 - \{A\}] = t[X_1 - \{A\}]\}$.
• the *selection* on r_1 by φ , denoted $\theta_\varphi(r_1)$, fulfills:
i) φ has the form $A = B|A = v|A$ **op** v, where A, B are attributes in $\alpha(r_1)$, $v \in \mathrm{dom}(A)$, **op** is a binary relation over $\mathrm{dom}(A)$.
ii) $\alpha(\theta_\varphi(r_1)) = X_1$, $\theta_\varphi(r_1) = \{t | \varphi$ is satisfied on $t\}$.

□

From above we give the basic notions of relational database and relational algebra. Now we present a basic introduction of description logic. Description logic is a logic-based formalism for knowledge representing, focusing on describe individual, concept, role and the relationship between them. We give formal definition of a simple description logic $\mathcal{ALC}$.

Definition (syntax and semantics of $\mathcal{ALC}$) The language $L_{\mathcal{ALC}}$ for $\mathcal{ALC}$ contains the following primitive symbols:

- object names $\mathbf{c_0}, \mathbf{c_1}, ...$;
- concept names $\top, \mathbf{C_0}, \mathbf{C_1}, ...$;
- role names $\mathbf{r_0}, \mathbf{r_1}, ...$;
- concept constructors: $\neg, \sqcap, \exists$;
- the subsumption relation: $\sqsubseteq$; and
- logical connectives: $\neg, \rightarrow$.

All the concept names and $\top$ are atomic concepts; If $\mathbf{C}$ and $\mathbf{D}$ are concepts, $\mathbf{c}, \mathbf{d}$ are object names, φ and ψ are statements, and $\mathbf{r}$ is a role name, then

(i) $\neg\mathbf{C}, \mathbf{C} \sqcap \mathbf{D}, \mathbf{C} \sqcup \mathbf{D}, \exists\mathbf{r}.\mathbf{C}, \forall\mathbf{r}.\mathbf{C}$ are concepts;

(ii) $\mathbf{C}(\mathbf{c}), \mathbf{r}(\mathbf{c}, \mathbf{d}), \mathbf{C} \sqsubseteq \mathbf{D}, \neg\varphi, \varphi \rightarrow \psi$ are statements.

A model M is a pair (Δ, I) such that Δ is a non-empty set, and I is an interpretation such that for each concept name $\mathbf{C}$, $\mathbf{C}^I \subseteq \Delta$; for each role name $\mathbf{r}$, $\mathbf{r}^I \subseteq \Delta \times \Delta$; for each object name $\mathbf{c}$, $\mathbf{c}^I \in \Delta$.

The interpretation of concepts are defined as follows:

$$\begin{aligned}
\top^I &= \Delta; \\
\mathbf{C}^I &= I(\mathbf{C}); \\
(\neg\mathbf{C})^I &= \Delta - \mathbf{C}^I; \\
(\mathbf{C} \sqcap \mathbf{D})^I &= \mathbf{C}^I \cap \mathbf{D}^I; \\
(\exists\mathbf{r}.\mathbf{C})^I &= \{x : \mathbf{E}y \in \Delta((x, y) \in \mathbf{r}^I \& y \in \mathbf{C}^I)\};
\end{aligned}$$

The satisfaction of statements are defined as follows:

$$\begin{aligned} M \models \mathbf{C}(\mathbf{c}) &\text{ iff } \mathbf{c}^I \in \mathbf{C}^I; \\ M \models \mathbf{r}(\mathbf{c}, \mathbf{d}) &\text{ iff } (\mathbf{c}^I, \mathbf{d}^I) \in \mathbf{r}^I; \\ M \models \mathbf{C} \sqsubseteq \mathbf{D} &\text{ iff } \mathbf{C}^I \subseteq \mathbf{D}^I; \\ M \models \neg\varphi &\text{ iff } M \not\models \varphi; \\ M \models \varphi \to \psi &\text{ iff } M \models \varphi \Rightarrow M \models \psi. \end{aligned}$$

□

By adding concept and role constructors, we could extend $\mathcal{ALC}$ to some complicated DL systems. In next chapter we present an extended DL for relational database.

3. Description logic $\mathcal{MDL}$

In this section we present a description logic $\mathcal{MDL}$ for relational databases, then we propose a translation from relational database to $\mathcal{MDL}$, and show that the translation preserves the information of tuples, relations in relational database.

Calvanese et al.[2,3] proposed a translation from ER-schema to $\mathcal{DLR}$. In his translation, a tuple entity is taken as a constant in DL, and entity sets are translated into concepts in DL. For each attribute, there is a special 2-arity role in $\mathcal{DLR}$, whose first component is an entity and the second component is the attribute value of that entity. We use an similar way to express tuples and relations from relational databases in description logic. Tuples in relational database are interpreted as constants in DL, that is, elements of the domain of DL model. A relations, as a set of tuples with the same attributes set, is interpreted as a concept. Like the translation from ER-schemas to $\mathcal{DLR}$, we take attribute values as constants, and for each attribute name there is a role associated. Binary relations on attribute values are also interpreted as roles.

Note that there are two kind of constants, one for tuple and the other for attribute value. Assume that there is a concept $\mathbf{R}$ in DL, associated with a relation R in a relational database. For a tuple constant $\mathbf{t}$, statement $\mathbf{R}(\mathbf{t})$ means that tuple t is an element of relation R. But for a attribute value constant $\mathbf{v}$, $\mathbf{R}(\mathbf{v})$ is just meaningless in the corresponding relational database, though that statement has no syntax error in the traditional DL system. For the similar reason, the role for the binary relations of attribute name, and the role for the tuple-attribute value association cannot be syntactically distinguished.

All of above show that it is necessary to propose a many-sorted logic to distinguish different types of object syntactically. We now present a many-sorted description logic $\mathcal{MDL}$, the formal description is as follows:

Definition (syntax of $\mathcal{MDL}$) The language $L_{\mathcal{MDL}}$ for $\mathcal{MDL}$ contains the following primitive symbols:

- tuple constant names $\mathbf{t_0}, \mathbf{t_1}, ...$;
- value constant names $\mathbf{v_0}, \mathbf{v_1}, ...$;

- relation concept names $\top_{\mathbf{R}}, \mathbf{R_0}, \mathbf{R_1}, ...$;
- value concept names $\top_V, V_0, V_1, ...$;
- attribute role names $\mathbf{a_0}, \mathbf{a_1}$,...;
- value role names $\mathbf{op_0}, \mathbf{op_1}, ...$;
- concept and role constructors: $\neg, \sqcap, \exists, \{\}$;
- the subsumption relation: $\sqsubseteq$; and
- logical connectives: $\neg, \rightarrow$.

All the relation concept names are atomic relation concepts, all the value concepts are atomic value concepts, all the attribute role names are atomic attribute roles. If $\mathbf{R}$ and $\mathbf{S}$ are relation concepts, $\mathbf{V}, \mathbf{W}$ are value concepts, $\mathbf{a}, \mathbf{b}$ are attribute roles , $\mathbf{op}$ is value role, φ and ψ are statements, then

(i) $\neg\mathbf{R}, \mathbf{R} \sqcap \mathbf{S}, \exists \mathbf{a}.\mathbf{V}$ are relation concepts;

(ii) $\neg\mathbf{V}, \mathbf{V} \sqcap \mathbf{W}, \exists \mathbf{op}.\mathbf{V}, \{\mathbf{v_0}, ..., \mathbf{v_n}\}$ are value concepts;

(iii) $\mathbf{a} \sqcap \mathbf{b}, \neg\mathbf{a}$ are attribute role names;

(iv) $\mathbf{V}(\mathbf{v}), \mathbf{R}(\mathbf{t}), \mathbf{a}(\mathbf{t}, \mathbf{v}), \mathbf{op}(\mathbf{v_1}, \mathbf{v_2}), \mathbf{V} \sqsubseteq \mathbf{W}, \mathbf{R} \sqsubseteq \mathbf{S}, \neg\varphi, \varphi \rightarrow \psi$ are statements. □

Definition (semantics of $\mathcal{MDL}$)

A model M is a tuple (Δ, Σ, I) where Δ, Σ are non-empty sets such that $\Delta \cap \Sigma = \emptyset$ and I is an interpretation such that for each relation concept name $\mathbf{R}$, $\mathbf{R}^I \subseteq \Delta$; for each value concept name $\mathbf{V}$, $\mathbf{V}^I \subseteq \Sigma$; for each attribute role name $\mathbf{a}$, $\mathbf{a}^I \subseteq \Delta \times \Sigma$; for each value role name $\mathbf{op}$, $\mathbf{op}^I \subseteq \Delta \times \Delta$; for each tuple constant name $\mathbf{t}$, $\mathbf{t}^I \in \Delta$ and for each value constant name $\mathbf{v}$, $\mathbf{v}^I \in \Sigma$;

The interpretation of concepts and roles are defined as follows:

$$
\begin{aligned}
\top^I_{\mathbf{R}} &= \Delta; \\
\top^I_{\mathbf{V}} &= \Sigma; \\
(\neg\mathbf{R})^I &= \Delta - \mathbf{R}^I; \\
(\mathbf{R} \sqcap \mathbf{S})^I &= \mathbf{R}^I \cap \mathbf{S}^I; \\
(\neg\mathbf{V})^I &= \Sigma - \mathbf{V}^I; \\
(\mathbf{V} \sqcap \mathbf{W})^I &= \mathbf{V}^I \cap \mathbf{W}^I; \\
(\exists \mathbf{a}.\mathbf{V})^I &= \{x | \text{exists } y \in \Sigma, (x, y) \in \mathbf{a}^I \& y \in \mathbf{V}^I\}; \\
(\exists \mathbf{op}.\mathbf{V})^I &= \{x | \text{exists } y \in \Sigma, (x, y) \in \mathbf{op}^I \& y \in \mathbf{V}^I\}; \\
(\{\mathbf{v_0}, ..., \mathbf{v_n}\})^I &= \{x | x = \mathbf{v}^I_i \text{ for some } i\}; \\
(\neg\mathbf{a})^I &= \Delta \times \Sigma - \mathbf{a}^I; \\
(\mathbf{a} \sqcap \mathbf{b})^I &= \mathbf{a}^I \cap \mathbf{b}^I;
\end{aligned}
$$

The satisfaction of statements are defined as follows:

$$
\begin{aligned}
M \models \mathbf{R}(\mathbf{t}) &\text{ iff } \mathbf{t}^I \in \mathbf{R}^I; \\
M \models \mathbf{V}(\mathbf{v}) &\text{ iff } \mathbf{v}^I \in \mathbf{V}^I; \\
M \models \mathbf{a}(\mathbf{t}, \mathbf{v}) &\text{ iff } (\mathbf{t}^I, \mathbf{v}^I) \in \mathbf{a}^I; \\
M \models \mathbf{p}(\mathbf{v_1}, \mathbf{v_2}) &\text{ iff } (\mathbf{v_1}^I, \mathbf{v_2}^I) \in \mathbf{p}^I;
\end{aligned}
$$

$$M \models \mathbf{R} \sqsubseteq \mathbf{S} \text{ iff } \mathbf{R}^I \subseteq \mathbf{S}^I;$$
$$M \models \neg\varphi \text{ iff } M \not\models \varphi;$$
$$M \models \varphi \rightarrow \psi \text{ iff } M \models \varphi \Rightarrow M \models \psi.$$

□

4. Represent information from relational databases in $\mathcal{MDL}$

Based on the formal definition of $\mathcal{MDL}$, We now present a translation σ from relational database to $\mathcal{MDL}$, which could be divided into syntactical and semantical parts: given a database d, the syntactical part of σ translate d to a collection of $\mathcal{MDL}$ statements $KB(d)$; the semantical part of σ translate d to an $\mathcal{MDL}$ model $\sigma(d)$.

Definition (syntactical part of σ)

Given a database $d = \{r_1, ..., r_n\}$, for each tuple $t \in \cup r_i$, the $\mathcal{MDL}$ sublanguage L_d for d includes a corresponding tuple constant name $\mathbf{t}$. Similarly, for each attribute value v, attribute name A, binary relation op and relation r_i; we have $\mathbf{v}$ as value constant, $\mathbf{a}$ as attribute role, $\mathbf{R_i}$ as relation concept and $\mathbf{op}$ as value role in L_d respectively. In addition, for each attribute A, we have $\mathbf{D_A}$ as a value concept, denotes the concept containing values in the domain of A. For a relation r in database, we use $\sigma(r)$ to denote the corresponding tuple concept in $\mathcal{MDL}$; if r is the atomic relation in the original database, then $\sigma(r) = \mathbf{R}$.

We construct $KB(d)$ as follows:

- For each relation $r \in d$ and tuple t such that $t \in r_i$ in relational database, the $\mathcal{MDL}$ statement $\mathbf{R_i(t)} \in KB(d)$;
- For each tuple t, attribute value v and attribute A such that $t(A) = v$, $\mathbf{a(t, v)} \in KB(d)$ and $\mathbf{D_A(v)} \in KB(d)$;
- For each relation $r \in d$ such that $\alpha(r) = \{A_1, ..., A_n\}$ and $U - \alpha(r) = \{B_1, ..., B_m\}$,

$\mathbf{R_i} \sqsubseteq \exists \mathbf{a_1}.\mathbf{D_{A_1}} \sqcap ... \sqcap \exists \mathbf{a_n}.\mathbf{D_{A_n}} \sqcap \neg\exists \mathbf{b_1}.\mathbf{D_{B_1}} \sqcap ... \sqcap \neg\exists \mathbf{b_m}.\mathbf{D_{B_m}} \in KB(d)$.

□

From the definition of $KB(d)$, we know that all the information about how tuples belong to relations and which value associated as an attribute value of a tuple, are translated into $KB(d)$, $KB(d)$ captures the complete static information of a relational database.

Definition (semantical part of σ)

Given a database $d = \{r_1, ..., r_n\}$, we construct an **MDL** model $\sigma(d) = (\Delta, \Sigma, I)$, such that:

- $\Delta = \bigcup_{1 \leq i \leq n} r_i$, that is, for each tuple t in database, t is an element of the domain of $\sigma(d)$;
- $\Sigma = \bigcup_{1 \leq i \leq n} \bigcup_{A \in \alpha(r_i)} \mathrm{dom}(A)$, including all the attribute value in the database;
- I is the interpretation fulfills:

- for each tuple constant $\mathbf{t}$, $I(\mathbf{t}) = t$, t is the corresponding tuple in relational database;
- for each value constant $\mathbf{v}$, $I(\mathbf{v}) = v$;
- for each tuple concept $\mathbf{R}$, $I(\mathbf{R}) = r$;
- for each attribute role $\mathbf{a}$, $I(\mathbf{a}) = \{(t, v)|t(A) = v\}$
- for value concept $\mathbf{D_A}$, $I(\mathbf{D_A}) = \mathrm{dom}(A)$;
- for each value role $\mathbf{op}$, $I(\mathbf{op}) = \{(v_1, v_2)|v_1\ op\ v_2\}$.

□

The translation σ is faithful, that is, all the statements in $KB(d)$ are satisfied in the $\mathcal{MDL}$ model $\sigma(d)$. Formally, we have following theorem:

Theorem For each $\varphi \in KB(d)$, $\sigma(d) \models \varphi$.

Proof. By induction on the construction of $\mathcal{MDL}$ statements, the detailed proof is omitted here. □

The theorem above shows that we could use $\mathcal{MDL}$ to express the static information of a relational database. We now show some relational algebra operations could also be represented in $\mathcal{MDL}$.

Firstly, we consider the set-theoretic operations: union $\cup$ and difference $-$. Let r_1, r_2 be relations in relational database, and $\mathbf{R_1}, \mathbf{R_2}$ be the corresponding tuple concept in $\mathcal{MDL}$. Use the concept constructor $\sqcap$, we obtain the tuple concept $R_1 \sqcap R_2$. It is straightforward to show the relation $r_1 \cup r_2$ is the corresponding relation of tuple concept $\neg(\neg R_1 \sqcap \neg R_2)$. Formally we have following proposition:

Proposition $\sigma(r_1 \cup r_2) = \neg(\neg\mathbf{R_1} \sqcap \neg\mathbf{R_2})$. That is, if a tuple t is in relation $r_1 \cup r_2$, then $(\neg(\neg\mathbf{R_1} \sqcap \neg\mathbf{R_2}))(\mathbf{t}) \in KB(d)$, $\sigma(d) \models (\neg(\neg\mathbf{R_1} \sqcap \neg\mathbf{R_2}))(\mathbf{t})$. □

Similarly, we have the following proposition:

Proposition $\sigma(r_1 - r_2) = \mathbf{R_1} \sqcap \neg\mathbf{R_2}$.

□

Note that the above proposition is satisfied only if $\alpha(r_1) = \alpha(r_2)$. If $\alpha(r_1) \neq \alpha(r_2)$, the relation $r_1 - r_2$ is undefined, while $\mathbf{R_1} \sqcap \neg\mathbf{R_2} \equiv \mathbf{R_1}$.

Besides of set-theoretic operations, the selection of relations could also be expressed in $\mathcal{MDL}$. Given a relation r, an attribute $A \in \alpha(r)$ and a value $v \in \mathrm{dom}(A)$, we consider $\theta_{A=v}(r)$. Using enumerate constructor $\{\}$ of concept, we could obtain $\{\mathbf{v}\}$ as a singleton concept of value. It is easy to show the concept $\exists\mathbf{a}.\{\mathbf{v}\}$ corresponds to the set of tuple in relational database, whose element has value v of attribute A. According to the definition of selection operator, $\mathbf{R} \sqcap (\exists\mathbf{a}.\{\mathbf{v}\})$ is the concept corresponding to $\theta_{A=v}(r)$. We could construct concepts for $\theta_{A\ op\ v}$ and $\theta_{A=B}$in a similar way. We have following proposition:

Proposition Selection operation could be expressed in $\mathcal{MDL}$ as follows:

$$\sigma(\theta_{A=v}(r)) = \mathbf{R} \sqcap (\exists\mathbf{a}.\{\mathbf{v}\});$$
$$\sigma(\theta_{A\ op\ v}(r)) = \mathbf{R} \sqcap (\exists\mathbf{a}.(\exists\mathbf{op}.\{\mathbf{v}\}));$$
$$\sigma(\theta_{A=B}(r)) = \mathbf{R} \sqcap (\exists(\mathbf{a} \sqcap \mathbf{b}).\mathbf{D_A}).$$

□

Notice that the set-theoretical operationa and selection are closed to the set of tuples in the original database. That is, there will be no new tuples due to these operations. We now consider the join operation. Given relations r_1, r_2, tuples in $r_1 \bowtie r_2$ are differs from those tuples from the original relations r_1, r_2: the attribute set of $r_1 \bowtie r_2$ is $\alpha(r_1) \cup \alpha(r_2)$. Since the domain Δ of tuple only includes tuples which are from the original database, We cannot express the new tuple gained from join operation, like $t \in r_1 \bowtie r_2$. Similarly, the rename and projection operations are not closed to tuples. That means without making any extension to $\mathcal{MDL}$, we cannot express these operations.

5. Conclusion

In this paper we present a many-sorted description logic $\mathcal{MDL}$ for relational databases. Based on $\mathcal{MDL}$, we present a translation σ from relational database to $\mathcal{MDL}$, which could be divided into syntactical and semantical parts: given a database d, the syntactical part of σ translate d to a collection of $\mathcal{MDL}$ statements $KB(d)$; the semantical part of σ translate d to an $\mathcal{MDL}$ model $\sigma(d)$. This translation is faithful, which means all the static information of tuples, relations and attribute values in database could be expressed in $\mathcal{MDL}$. In addition, we show some relational algebra operations could be expressed in $\mathcal{MDL}$ as concepts.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi and P. F. Patel-Schneider (eds.), *The Description Logic Handbook,* Cambridge University Press, 2002.

[2] A. Borgida, M. Lenzerini and R. Rosati, Description logics for data bases, in [1], 472-494.

[3] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi and R. Rosati, Description logic framework for information integration, KR98, 2-13.

[4] A. Borgida, Description logics for querying databases, in: DL-94, Bonn, Germany, 1994.

[5] A. Borgida, Description logics in data management, *IEEE Transactions on Knowledge and Data Engineering* 7(1995), 671-682.

[6] E. F. Codd, Relational completeness of database sublanguages, R. Rustin (ed.), *Data Base Systems,* Prentice Hall, 1972, 65-98.

[7] D. A. Simovici and R. L. Tenney, *Relational Database Systems,* Academic Press, 1995.

[8] P. C. Kanellakis, Elements of relational database theory. *Brown U. Technical Report,* 1988.

[9] M. Lenzerini, Description logics and their relationships with databases, ICDT'99, LNCS 1540, 32-38.