

# Poster on Rule-based Technology for Schema Transformation

Yangjun Chen and Wolfgang Benn

Department of Computer Science, Technical University of Chemnitz-Zwickau,  
09107 Chemnitz, Germany

**Abstract** As the first step of database integration, a participating local database schema should be transformed into an abstract one to remove data model conflicts. For our project, the object-oriented schema is chosen to represent the integrated information and a semi-automatic method is developed to transform relational schemas into OO schemas. The main idea behind this method is to define two kinds of logics: a relational database logic  $L_{db}$  and an object-oriented logic  $L_o$  to formalize the corresponding data models.  $L_{db}$  is used to model a database structure as well as its state, while  $L_o$  is utilized for an object-oriented database. Then, based on these formalisms, a set of Horn-clause-like rules is constructed to do a meta-level reasoning for translating a formalism into another.

## 1. Outline

By the definition of  $L_{db}$ , the following issues concerning a relational database are considered:

- relation classification,
- null values,
- view definitions and
- integrity constraints.

That is, using our  $L_{db}$ , the above elements can be represented uniformly.

On the other hand, using our  $L_o$ , the following data can be formalized:

- object structure,
- derivation rules defined on object structures, and
- (inter and intra) semantic constraints.

Then, based on these two logics, a set of rules can be established to define the *relational-to-OO* transformation.

In our system, the relations are classified into four classes: *strong entity relations*, *weak entity relations*, *strong relationship relations* and *weak relationship relations*. Accordingly, we construct four subsets of rules to transform each of them into the corresponding object structures with different semantic constraints. In addition, in terms of our epistemological principles, two other subsets of rules are defined, respectively, to build *is-a* and *part-of* relationships, and to deal with more complicated inheritances in an OO schema generated by transforming a relational one. The detailed description about such rules can be found in [1]. In the following, we discuss the problems concerning the transformation of null values, view definitions and integrity constraints.

## 2. About null values, views and integrity constraints

In our system, the null value is treated as an applicable null value. This denotes that this value is presently unknown, but can be entered to the database once known. Then, an applicable null value can be represented using the fuzzy set concept. That is, a null value can be represented (explicitly or implicitly) as a multi-value of the form:  $\{(v_1, \frac{1}{n}), \dots, (v_n, \frac{1}{n})\}$ , where  $\{v_1, \dots, v_n\}$  is the attribute domain and  $\frac{1}{n}$  is the pro-

bability of some  $v_i$  becoming the corresponding attribute value. As a consequence, if any null value appears, the corresponding attribute will be transformed into a set-valued attribute by the schema transformation to accommodate the relevant values.

In order to transform a view definition, we have to extend the object-oriented model with deductive abilities. Exactly speaking, the derivation rules should be allowed in an object-oriented schema. In our system, for the complex object structures in  $L_o$ , we define derivation relations in a standard way, as implicitly universally quantified statements of the form:

$$\gamma_1 \& \gamma_2 \dots \& \gamma_l \Leftarrow \tau_1 \& \tau_2 \dots \& \tau_k,$$

where both  $\gamma_i$ 's and  $\tau_k$ 's are complex object structures or normal predicates of the first-order logic. For example, the rule  $(o_1, Empl)[e\_name: x, work\_in: o_2 Dept] \Leftarrow (o_2: Dept)[d\_name: y, manager: o_1 Empl]$  states that department managers work in the department they manage. Here *Empl* and *Dept* are classes,  $o_1$  and  $o_2$  are object variables (over object identifiers) and *work\_in* and *manager* are two aggregation functions. Universal quantifiers over  $o_1$  and  $o_2$  are omitted. As another example, consider the so-called 'interesting pair' problem. The problem is to find the pairs *employee-manager* such that the employee's department's manager's name coincides with the employee's name, which can be represented (using our method) as follows:

$$pair(o_1, manager(o_2)) \Leftarrow (o_1, Empl)[e\_name: x, work\_in: o_2 Dept], \\ manager(o_2).e\_name = x.$$

At last, in terms of  $L_o$ 's syntax, the integrity constraint w.r.t. an object-oriented schema can be defined to be a formula of the form:  $(Q_1x_1) \dots (Q_nx_n)e(x_1, \dots, x_n)$ , where each  $Q_i$  is either  $\forall$  or  $\exists$ ,  $n > 0$ ,  $e$  is a set of complex object structures or normal predicates of the first-order logic, connected with  $\wedge$ ,  $\vee$ ,  $\neg$ , or  $\Rightarrow$ , and  $x_1, \dots, x_n$  are all variables occurring in  $e$ . For example, formula  $(\forall o_1)(\forall o_2)\{(o_1, person)[name: \_, age: \_, sibling: o_2] \Rightarrow (o_2, person)[name: \_, age: \_, sibling: o_1]\}$  is a legal integrity constraint, representing that if  $o_1$  is the sibling of  $o_2$ , then  $o_2$  must be the sibling of  $o_1$ , too. Then, we can define some rules to do the integrity constraint transformation automatically. For example, a rule of the form:

$$Q(x, y)\{\tau_v(\bar{s}) \Rightarrow \tau_u(\bar{l})\}, x \in \bar{s}, y \in \bar{l}, generate\_class(v), \\ generate\_class(u), \delta_{v,w,v}(x, \_) , \delta_{u,z,u}(y, \_) \rightarrow \\ Q(x, y)\{(o_1, v)[w: x] \Rightarrow (o_2, u)[z: y]\}$$

may enable us to translate any relational integrity constraint of the form:  $Q(x, y)\{\tau_v(\bar{s}) \Rightarrow \tau_u(\bar{l})\}$  into an object-oriented one, where  $Q(x, y)$  stands for a sequence of quantifications containing variables  $x$  and  $y$ . It may be of the form:  $\forall x \forall y, \forall x \exists y, \exists x \exists y$  or  $\exists x \forall y$ . (See [1] for the definition of predicates appearing in the above rule.)

## References

1. Y. Chen and W. Benn, A Rule-based Strategy for Translating Relational Schemas into OO-Schemas, in: *Proc. of the first Int. Symposium on Cooperative Database Systems for Advanced Applications (CODAS'96)*, Kyoto, Japan, Dec. 1996, pp. 85-88.

The work reported here is supported by DFG under grant No. Be1786/1-1.