Towards Restructuring and Normalization of Types in Databases

Yang Li (杨 立)

Institute of Automation, The Chinese Academy of Sciences, Beijing 100080

Received June 2, 1993.

Abstract

The equivalence-preserving transformation and normalization of types in object-oriented databases are discussed. Specifically, a normal form of types based on set-theoretic equivalence is proposed, rewrite rules which transform types into normal forms are presented, and the uniqueness of normal form and the completeness of rewrite rules are proved. The emphasis of this work is on normal forms and corresponding rewrite rules. It provides a new formal approach for the study of restructuring of database schema and other manipulations in object-oriented databases.

Keywords: Object-orientation, object-restructuring, database normalization.

1 Introduction and Preliminary

In the study of object-oriented database systems, a large variety of semantic data models\cite{1,2} provide flexible means for data modeling. However, many models do not give the corresponding modeling methods or tools to manipulate typed hierarchical objects. In the database schema design, on the other hand, an object can be described in different ways by types of different structures. Obviously, this leads to the restructuring and transformation of types. Furthermore, similar to the normalization of relations, object-oriented databases also need the normalization of types.

As the preliminary of our discussion, we present the definition of static types in databases. Types are constructed from atomic types by constructors \{*, x, + \}.

There are two kinds of atomic types. One is printable (denoted by $P$) which corresponds to objects of predefined types such as string, integer, real, and Boolean. The other is called abstract (denoted by $A$) which has no underlying structure from the point of view of database designers or users, such as various multimedia data.

Set constructor, denoted by $(\ast)$, can construct sets of objects of a type, $T = \{T'\}$. It supports the semantic modeling concept “association”.

This work is partially supported by grants of the National Natural Science Foundation of China and National Postdoctoral Science Foundation of China.
The Cartesian product of types is represented by tuple constructor \( ( \times ) \), \( T = [T_1, \ldots, T_n] \). It supports “aggregation”.

Union constructor \((+)^\) means that the type constructed as the plus of several types will have a domain equal to the union of domains of these types, that is \( T = (T_1, \ldots, T_n) \).

**Definition 1.** Assume \( t \in P \cup A, P \cap A = \phi \), \( t \) is called atomic type. \( \text{dom}(t) \) denotes the set of all atomic objects that belong to \( t \).

Starting with atomic types, more “complex” types are built by three constructors.

**Definition 2.** A type is a directed tree \( T = (V, E) \) such that
1. \( V \) is the disjoint union of sets \( V_P \) (printable vertices), \( V_A \) (abstract vertices), \( V_* \) (\(*\)-vertices), \( V_{\times} \) (\(\times\)-vertices) and \( V_{+} \) (+-vertices);
2. elements in \( V_P, V_A \) are leaves of the tree;
3. each \(*\)-vertex has only one child;
4. \( \times\)-vertex and +-vertex have more than one ordered child.

The domain of type \( T \) can be defined as follows.

**Definition 3.** The set of objects of type \( T \), denoted by \( \text{dom}(T) \), is defined as follows.
1. If \( T \) is an atomic type, \( \text{dom}(T) \) is defined as in Definition 1.
2. If the root \( r \) of \( T \) is a \(*\)-vertex, and \( T_1 \) is the subtree of \( r \), \( T = \{T_1\} \), then
   \[
   \text{dom}(T) = \{\{O_1, \ldots, O_m\}| m \geq 0, O_i \in \text{dom}(T_1), 1 \leq i \leq m}\]
3. If the root \( r \) of \( T \) is a \( \times\)-vertex, and \( T_1, \ldots, T_n \) are the ordered subtrees of \( r \), \( T = [T_1, \ldots, T_n] \), then
   \[
   \text{dom}(T) = \{\{O_1, \ldots, O_n\}| O_i \in \text{dom}(T_i), 1 \leq i \leq n}\]
4. If the root \( r \) of \( T \) is a +-vertex, and \( T_1, \ldots, T_n \) are the ordered subtrees of \( r \), \( T = (T_1, \ldots, T_n) \), then
   \[
   \text{dom}(T) = \bigcup_{i=1}^{n} \text{dom}(T_i)\]

## 2 Equivalence of Types and Normal Form

In the aspect of type equivalence, Abiteboul and Hull proposed a partial order on the structure of constructed types\([4]\). In this section, we define a set-theoretic equivalence between types. A normal form is introduced, and rewrite rules which convert types into normal forms are given. By three theorems, it is shown that each equivalence class of types has a unique member (up to isomorphism) that is in normal form, and rules are complete to convert each type in the class into that normal form.