Using Multivalued Logic in Relational Database Containing Null Value

Ma Zongmin (马崇民)
Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110015

Yan Li
Liaoning Workers Movement College, Shenyang 110168

Received January 11, 1995; revised June 19, 1995.

Abstract
In this paper, several kinds of multivalued logic for relational database and their developing process are presented on the basis of null value's semantics. A new 5 valued logic is led into relational database containing null value. The feasibility and necessity of using 5 valued logic are expounded. Comparative calculation and logical calculation under 5 valued logic are defined at the end of the paper.

Keywords: Null value, relational database, multivalued logic.

1 Introduction

In classical relational databases, the value recording the result of comparative calculation can only be T (true) and F (false), and the result of logical calculation AND and OR is also T and F. \{T, F\} is traditional 2 valued logic (2VL).

After incomplete information on tuple’s attribute—null value is led into relational model, the result of comparative calculation is concerned with the null value’s appearance and its semantics directly. As 2VL is not applicable, it is necessary to introduce new logical value system that can fully embody the whole semantics of null value and its difference in order to exactly record the results of comparative calculation and logical calculation in relational database under null value circumstance.

2 Basic Knowledge

According to the semantics of null value, null value has three kinds: impossible value, existing value and placeholder. In this paper, they are presented by \(\phi^0\), \(\phi^+\) and \(\phi^-\), respectively. \(\phi^0\) is the particular attribute value showing that it is impossible to have common value on the attribute. \(\phi^+\) is the particular attribute value showing the range of common value, and the range is expressed by range \((\phi^*)\). The upper limit of this range is expressed by \(sup(\phi^*)\) and lower limit of this range is expressed by \(sub(\phi^*)\). If \(\phi^+\) is a value on the attribute \(A\), there is \(range(\phi^+) \subseteq DOM(A)\), in which \(DOM(A)\) is the attribute scope. \(\phi^-\) is the particular attribute value which is either \(\phi^0\) or \(\phi^+\), and as to \(\phi^-\) of attribute \(A\), its semantics is \(\{\phi^0\} \cup DOM(A)\).
The tuple containing null value is called incomplete tuple. There is relationship of equivalence and compatibility between null values or incomplete tuples. Equivalence is expressed by \( \equiv \), and compatibility is expressed by \( \approx \).

**Definition 1.** All \( \phi_i \)'s are equivalent. For \( \phi_i^+ \) and \( \phi_j^+ \), if \( \phi_i^+ \equiv \phi_j^+ \), \( i = j \), i.e. when replacing \( \phi_i^+ \) and \( \phi_j^+ \) with a common value, the same value is used. For \( \phi_i^- \) and \( \phi_j^- \), if \( \phi_i^- \equiv \phi_j^- \), \( i = j \), i.e. when replacing \( \phi_i^- \) and \( \phi_j^- \) with more definite value (including null value, but not placeholder), the same value is used.

That two common values are equal is one particular feature of equivalence.

**Definition 2.** For \( \phi_i^+ \) and \( \phi_j^+ \), if range(\( \phi_i^+ \)) \( \cap \) range(\( \phi_j^+ \)) \( \neq \) \( \emptyset \), \( \phi_i^+ \approx \phi_j^+ \). For \( \phi_i^- \) and \( \phi_j^- \) belonging to attribute A and attribute B respectively, if \( \text{DOM}(A) \cap \text{DOM}(B) \neq \emptyset \), \( \phi_i^- \approx \phi_j^- \). For common value a and \( \phi \) on attribute A, if \( a \in \text{DOM}(A) \), \( a \approx \phi^- \), and for common value b and \( \phi^* \), if \( b \in \text{range}(\phi^*) \), \( b \approx \phi^* \).

**Definition 3.** For complete or incomplete tuples t and s, if the values on the same attribute name are equivalent correspondently, \( t \equiv s \), and if the values on the same attribute name are compatible correspondently, \( t \approx s \).

### 3 Introducing 3VL and 4VL

According to E. F. Codd's wording in [1,2], null value has two kinds of semantics: inapplicable data and applicable but unknown data. These semantics correspond to impossible value and existing value.

Because existing value has a value range, the result except for T and F should have a third state—maybe value which may be either T or F and is expressed by M when it participates in comparative calculation. T, M and F comprise 3VL under two kinds of semantics of null value.

The case in which existing value participates in comparative calculation is considered above. Because impossible value has no concept of big or small, the result is not T, F or M but should be another—no existing, which is expressed by I when impossible value participates in comparative. I and M are not the same in strict meaning. I, F, M and T make up 4VL under two kinds of semantics of null value.

As pointed out by E. F. Codd, though 4VL is more complex than 3VL, it can exactly distinguish among impossible information, false information and applicable but unknown information, so the use of 4VL is proper under semantics of impossible value and existing value. On the other hand, 4VL should become 3VL while let I equal F.

### 4 Appearance of 7VL

According to the discussion above, null value has the semantics of placeholder besides the semantics of impossible value and existing value. Two kinds of semantics of null value result in 3VL, what multivalued logic should derive under the whole semantics of null value?

See the following comparative calculation \( X = a \) based on the whole semantics of null value, in which \( X \) is attribute name and \( a \) is common value. The value of \( X = a \) has several possibilities as follows:

1. The attribute value of \( X \) is common value and is equal to \( a \). The value of expression is true, expressed by T.