Chapter XXV

Data Dependencies in Codd’s Relational Model with Similarities

Radim Belohlavek
Binghamton University–SUNY, USA and Palacky University, Czech Republic

Vilem Vychodil
Binghamton University–SUNY, USA and Palacky University, Czech Republic

ABSTRACT

This chapter deals with data dependencies in Codd’s relational model of data. In particular, we deal with fuzzy logic extensions of the relational model that consist of adding similarity relations to domains and consider functional dependencies in these extensions. We present a particular extension and functional dependencies in this extension that follow the principles of fuzzy logic in a narrow sense. We present selected features and results regarding this extension. Then, we use this extension as a reference model and compare it to several other extensions proposed in the literature. We argue that following the principles of fuzzy logic in a narrow sense, the same way we can follow the principles of classical logic in the case of the ordinary Codd relational model, helps achieve transparency, versatility, conceptual clarity, and theoretical and computational tractability of the extension. We outline several topics for future research.

INTRODUCTION

Ordinary Codd Relational Model and Data Dependencies

Codd’s relational model of data is one of the most important contributions to computer science and perhaps the most important concept in data management: “A hundred years from now, I’m quite sure, database systems will still be based on Codd’s relational foundation” (Date, 2000, p. 1). Among the main virtues of the model are logical and physical data independence, access flexibility, and data integrity. They are mainly due to the reliance of
the model on a simple yet powerful mathematical concept of a relation and on first-order logic: “The relational approach really is rock solid, owing (once again) to its basis in mathematics and predicate logic” (p. 138). Codd’s relational model represents the theoretical foundations for relational databases.

Data dependencies represent an important tool in Codd’s relational model (see, e.g., Maier, 1983). Functional dependencies, multivalued dependencies, inclusion dependencies, and join dependencies are perhaps the most important data dependencies. They serve as constraints and are being used in the design of relational databases. Data dependencies are a classic topic in relational databases that has been thoroughly studied in the past (Ullman, 1988). In addition to that, data dependencies have been used for data mining purposes (Manilla & Räiha, 1994).

Codd’s relational model, data dependencies, and functional dependencies in particular, are the main subject of this chapter. We will now recall these notions. The central notion in Codd’s model is that of a relation (table) over a relation scheme. This concept is illustrated by the table in Figure 1. The corresponding relation scheme \( Y \) consists of the attributes name, age, and education, denoted for brevity by \( n, a, \) and \( e \). For each attribute \( y \), the set \( D_y \) of all possible values of \( y \) is called a domain of \( y \). For instance, \( D_n \) consists of all names (character strings). The table in Figure 1 can be thought of as a relation \( \Delta \) between domains \( D_n, D_a, \) and \( D_e \). That is, \( \Delta \subseteq D_n \times D_a \times D_e \). The tuples that belong to \( \Delta \) are just the rows of the table. That is, \{Adams, 30, Comput. Sci.\} \( \in \Delta \), …, \{Francis, 39, Business\} \( \in \Delta \), but, for instance, \{Adams, 30, Business\} \( \notin \Delta \). A functional dependency (FD) over a relation scheme \( Y \) is an expression:

\[
A \Rightarrow B,
\]

where \( A \) and \( B \) are sets of attributes from \( Y \), that is, \( A, B \subseteq Y \). An FD \( A \Rightarrow B \) can be true (valid) or false (not valid) in a given table \( \Delta \) over a relation scheme \( Y \). By definition, \( A \Rightarrow B \) is true in \( \Delta \), denoted by \( \| A \Rightarrow B \|_\Delta = 1 \), iff (if and only if) the following applies.

\[
\text{IF } t_1, t_2 \in \Delta: \quad \text{IF } t_1, t_2 \text{ have the same values on attributes from } A, \quad \text{THEN } t_1, t_2 \text{ have the same values on attributes from } B.
\]

Otherwise, if there are \( t_1, t_2 \in \Delta \) such that \( t_1 \) and \( t_2 \) agree on attributes from \( A \) but disagree on some attribute from \( B \), \( A \Rightarrow B \) is not true in \( \Delta \), denoted by \( \| A \Rightarrow B \|_\Delta = 0 \). If one thinks of \( A \Rightarrow B \) as a constraint, then \( \| A \Rightarrow B \|_\Delta = 1 \) means that the data in \( \Delta \) satisfy the constrained \( A \Rightarrow B \). If we denote the fact that \( t_1 \) and \( t_2 \) have the same values on all attributes from \( C \) in table \( \Delta \) by \( t_1(C) =_\Delta t_2(C) \), we can rewrite Equation 1 as follows.

\[
\text{for any } t_1, t_2 \in \Delta: \quad \text{IF } t_1(A) =_\Delta t_2(A) \quad \text{THEN } t_1(B) =_\Delta t_2(B).
\]

As an example, \{age\} \( \Rightarrow \) \{education\} is an FD that is not true in the table in Figure 1 because Adams and Black have the same age but differ in their education. On the other hand, \{name\} \( \Rightarrow \{ \text{age, education}\} \) is an FD that is true because, in our table, name uniquely determines both age and education.

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>education</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chang</td>
<td>28</td>
<td>Accounting</td>
</tr>
<tr>
<td>Enke</td>
<td>36</td>
<td>Electric. Eng.</td>
</tr>
<tr>
<td>Francis</td>
<td>39</td>
<td>Business</td>
</tr>
</tbody>
</table>