Normalization of Relations with Nulls in Candidate Keys

George C. Philip, University of Wisconsin-Oshkosh, USA

ABSTRACT

This paper discusses normalization of relations when the candidate keys of a relation have missing information represented by nulls. The paper shows that when the missing information is of the type “not applicable” or “does not exist,” problems and confusion can arise in normalizing relations. Candidate keys with missing information commonly are found in relations that represent information on two entities with a one-to-one relationship between them. The current definition of Boyce-Codd Normal Form (BCNF) is ineffective in identifying poor designs in such relations that may have insertion/deletion anomalies. It is shown that the above problem can be corrected by incorporating the concept of entity integrity rule into the definition of BCNF. This paper also shows that incorporating the entity integrity rule into the definition of either a relation or a candidate key does not provide a satisfactory solution to the problem.

Keywords: Relational databases, candidate keys, normalization of relations

The relational database design concepts were developed without considering missing information in relations (Codd, 1986; Levene, 1999; Date, 2000). Value of an attribute in a tuple may be missing for several reasons: 1) Value is applicable but it is unknown, 2) Value is not applicable, 3) Value does not exist, or 4) Other reasons such as value is undefined (Date, 2000). An example of value that is not applicable is the attribute driver license number for a ten-year-old child. If an adult does not have a driver’s license number, then the value does not exist. If an adult has a driver’s license number, but it is unknown, then the value is applicable but unknown.

A common method of representing
missing values is using nulls (Codd, 1986). Other methods include using default values (Date, 1990), using a subset of the attribute domain (Lipski, 1979), and using variables or many different “null values” (Imielinski and Lipski, 1984). Missing information can create problems in querying data from relations (Imielinski and Lipski, 1984; Date, 1990). Several methods have been proposed to extend the relational operators to deal with missing values (Codd, 1986; Reiter, 1986; Sutton and King, 1995).

Another group of studies examined the effect of nulls on the concept of functional dependency (Vassiliou, 1980; Vardi, 1986; Levene and Loizou, 1999). These studies have focussed primarily on missing values of the type, “applicable but unknown.”

The current paper examines the effect of nulls in candidate keys, on normalizing a relational schema. The nulls considered in this paper are of the type “not applicable” or “does not exist.” Specifically, this paper examines the effectiveness of Boyce-Codd Normal Form (BCNF) in identifying insertion/deletion anomalies if missing values in candidate keys are represented by nulls. Candidate keys with values that are not applicable or values that do not exist commonly are found in relations that represent information on two entities with a one-to-one relationship between them. It is shown that the current definition of Boyce-Codd Normal Form (BCNF) is ineffective in identifying poor designs in such relations. The paper identifies the source of the problem and offers a solution by incorporating the concept of entity integrity rule into the definition of BCNF. This paper also shows that incorporating the entity integrity rule into the definition of either a relation or a candidate key does not provide a satisfactory solution to the problem.

**DESCRIPTION OF THE PROBLEM**

To help explain the problem, we consider two entities, EMPLOYEE and COMPUTER, that have a (zero-or-one)-to-(zero-or-one) relationship between them. Thus, a computer has zero or one employee assigned to it at any given time. Similarly, an employee is assigned to zero or one computer at any time. Consider a relation, ASSIGNMENT (ID, NAME, TITLE, COMPUTER_NO, MODEL, RAM).

In the above relation, ID, NAME, and TITLE represent the identification number, the name, and the title of the employee, respectively. ID is the only unique identifier of the employee. COMPUTER_NO is the only unique identifier of the computer assigned to the employee. MODEL and RAM represent the model, and the amount of memory of the employee’s computer, respectively. Figure 1 shows a sample state of the relation.

Since some employees may not have a computer, the corresponding tuples in ASSIGNMENT do not have any value for COMPUTER_NO, MODEL, and RAM. Similarly, since some computers do not have employees assigned to them, the corresponding tuples do not have any value for ID, NAME and TITLE. Here, null may represent value that does not exist or value that is not applicable. For example, one employee may not be eligible for a computer (not applicable), while another employee may be eligible, but no computer was assigned (does not exist).

The above design is not a good one. The relation suffers from insertion and deletion anomalies. If ID is selected as the primary key, then information on a computer cannot be inserted if the computer is not assigned to an employee.
Related Content

Evaluation of Component-Based Development Methods
[www.igi-global.com/chapter/evaluation-component-based-development-methods/23021?camid=4v1a](www.igi-global.com/chapter/evaluation-component-based-development-methods/23021?camid=4v1a)

FuzzyEER: Main Characteristics of a Fuzzy Conceptual Modeling Tool
[www.igi-global.com/chapter/fuzzyeer-main-characteristics-fuzzy-conceptual/18761?camid=4v1a](www.igi-global.com/chapter/fuzzyeer-main-characteristics-fuzzy-conceptual/18761?camid=4v1a)
View Maintenance for Materialized Transitive-Closure Relations
www.igi-global.com/article/view-maintenance-materialized-transitive-closure/51188?camid=4v1a

Database Query Personalization
www.igi-global.com/chapter/database-query-personalization/11137?camid=4v1a