DATA DICTIONARIES IN OPEN SYSTEM COMMUNICATION

Peter Hitchcock

University of Victoria
P.O. Box 1700
Victoria, British Columbia
CANADA V8W 2Y2

0. Introduction

My current work is based on the premise that one should be able to access a foreign database system with the same ease that one can move from one type of telephone system to another. Data dictionaries must form the focal point in any architecture which would support such a concept. Data from a foreign system must be presented to the local system in such a way that the local system can understand it if the databases are disjoint, or merge the foreign conceptual model with the local one if the data bases have some common structure. Once such a rapport has been established, other aspects of the conceptual models, such as constraints and operations on the data, must be translated from one system to the other. This raises the following issues which I would like to see discussed at the workshop.

1. The Need for a Standard Representation for Conceptual Models.

There are two approaches which can be followed in transforming from one representation of a conceptual model to another, either sufficient mappings are developed linking all pairs of representation techniques, or a single mapping is produced for each modelling technique, which goes to and from the standard. There are many different data modelling techniques and so the second alternative is preferable.

This standard representation is purely for the purpose of information interchange, and as such need not be concerned with such aspects as user friendliness. This means that it can follow a principle of explicitness. For example, rules about hierarchies which are implicit in the constructs provided in one modelling technique, should be spelt out in detail in the standard representation. In some sense, the mapping to the standard representation will become a formal definition of the particular modelling technique which has been chosen.

The standard representation should not replace other modelling techniques. If it did so, the convenient shorthand of data modelling, which is so essential to commercial systems, would be lost.

2. Canonical Conceptual Models.

If the two database systems are holding some information about the same things, and the transformations above have been followed so that the same representation is now used for both the conceptual models, then some problems still remain. It will be necessary to merge the two conceptual schema. This is essentially a database design problem, and one that is not capable of automatic solution.

Homonyms and synonyms must be recognised and resolved. There may also be representational differences. This is best illustrated by the following example, using the relatively simple binary model of Mike Senko to describe an expense claim. The expense claim has an identifier and is made up of amounts of money which are charged to various codes. We have to make a design decision as to whether these codes are entities in their own right, or whether they are the names of associations between the entities 'expense claim' and 'amount of money'. In the predicate calculus this appears as a decision between whether we use predicate names, or make these names the instances of variables. It is my belief that this

is a real problem, and that no canonical representation of a particular situation exists.

Design decisions like the above are ultimately made in the light of the operations to be performed against the database. The languages we have for operating on data make it easy to work with entities, less so to manipulate associations. It is easy for most database systems to answer the question: tell me who is in this department; less easy to reply to the question: tell me all you know about John Smith. In predicate calculus terms, over what do we quantify?

Yet we have the problem in the database world that we are trying to build models that will support not only the processes that we know about today, but also the applications that will come along tomorrow. Ideally our models should not be so affected by the operations to be performed.

3. Modelling Processes

The processes that we have to support are the reason for storing data in the database. If we are to avoid violating the constraints of the database rules, then the operations on the data should themselves be constrained, and should become the only operations allowed against the data. Clearly they must form part of any conceptual model. It should be possible in many cases to use their descriptions to show that transactions built only from these operations cannot violate the database constraints.

4. Coping with Change

Databases evolve over time. If the initial design is correct then they should evolve by modelling the enterprise to greater levels of detail. This has the advantage that the previous less detailed views can always be maintained for reading, but there will be an awkward transition period until the whole database is converted to the same level of detail.

5. The Place for a Data Dictionary:

It is my belief that the division of a system into the data dictionary and the database is artificial. The two should merge together imperceptibly. Instances of the data should not be disjoint from descriptions of the data. To a large extent this is a legacy from our record based past which is still with us in many data modelling techniques.