Abstract Data Types and Data Bases

by
Paolo Paolini
Politecnico of Milano

Several researchers have been working, recently, at the attempt of using the ADT approach in the Data Base area. Different specific problems have attacked and we will review some of them.

Data Models Definition

Examples of what we mean by Data Model, in this context, are the "Relational Model", the "Codasyl Model",ecc. with their general properties. Historically such properties were identified with "how the data are represented in the Data Base". The ADT approach emphasizes, instead, "how the data of the Data Base can be operated with".

The ADT approach to the definition of a Data Model consists of the following steps:

1 - identification of the basic operations which can be performed with the data, instances of the Data Model (e.g. projection, join,selection ecc. for the Relational Model).
2 - Specification of the sinctactic properties of the operations, i.e. the types of the input and output parameters.
3 - Specification of the semantic properties of the operations.

Different techniques differ essentially at the third step. Various approaches were presented at the workshop by F.Cristian, H.C.Mayr,J.W.Schimdt,M.Shaw,S.N.Zilles and J.W.Thatcher. Our preference is for the Algebraic specifications, where the properties of the operations are expressed through equations. But it is also our belief that not significant advantage or difficulty will arise from using one techinique or another. The real issue is whether formal definitions of Data Models are useful or not. Data Base Models are, in fact, very complicated (by ADT standards) and their definitions become easily messy and obscure. The only readable (and therefore utilizaable) definitions are those where drastic simplifications have been introduced.These simplifications alterate so much the features of the Data Model itself, that they can't be acceptable for "real life" use and they become a mere technical excercise. Accurate definitions, instead, are so complicated and detailed (our shortest definition of the Relational Model runs 20 pages long), that precision is achieved at the expenses of clearness.

Moreover a careful look at the DBMS's makes evident that each system has "its own version" of the Data Model. Therefore it seems impossible to speak, for example, of the Relational Model in general, and to describe how the Model is implemented in SYSTEM-R, INGRES, PASCAL-R etc.etc. Since the available operations differ significantly, it seems necessary to describe how the Relational model is defined and implemented by those systems. And more, the actual implementations are so tricky that, even looking at them with an "abstract eye", it seems impossible to formally prove their correctness.

An actractive alternative is to use some kind of automatic tool, to handle formal definitions of ADT's. The CLEAR language is a good candidate for this purpose. Its actual usefulness, however, must be demonstrated by an effective use.

Our conclusion is therefore that formal definitions of Data Models are not very likely to have a strong impact on the Data Base field. The strongest influence is, pheraps, a change of the attitude of people:

i.e. people now realize better that the characterization of a Data Model is given by the set of operations available for creating, modifying and accessing the Data Base (and not by the static description of the representation of data). Therefore, when, for example, Data Models or Data Base descriptions are compared, operations are also taken into account.

Data Base Languages

Several researchers are currently working at the definition of ADT-based programming languages for Data Bases applications.

At the workshop this approach was described by M.L.Brodie,L.A.Rowe and J.W.Schimdt.

The basic idea is that a Data Base object should be incorporated in a program as an "abstract object" (in Programming Languages terminology) ,i.e. an object the representation of which is unkown, but the properties of which (available operations included) are well known.

The objects are manipulated by the program through a set of "abstract operations", the semantic of which is well defined.

The integration of an application program with a Data Base is treated as the "import" of one or more abstract objects in the program itself. Therefore no special syntax and semantics are required.

A discussion point is whether the objects imported by the application program should be of general nature (e.g. relations) with generalized operations (e.g. projection, join etc.), or should be tailored to the application (e.g.the table of suppliers, the list of customers etc.) with ad-hoc operations (e.g.find-next-customer,find-supplier-with-a-given name etc.).

Most of the researchers seem to prefer the first type of solution; we are personally more inclined toward the second solution, since it seems to provide a better Data Independence to the application programs and to be more coherent with the ADT approach, which suggests that each program should build the abstract objects more suitable to it.

We believe that this approach to the design of Data Base languages is very promising, but, obviously, their advantages can only be proved by implementing them and experiencing their use.

Formal Definition of Views and Data Base Design

In tha ADT terminology views can be considered abstract objects, which use, as representation, the underlying Data Base. Again the abstractions defined by the views can be of general nature (e.g.relations) or tailored to the application.

Taking the second approach, we have tried to develop a technique for the formal verification of application programs. The application programmer should be provided with:

a - a formal specification of the view(s) he is working with
b - a formal specification of what the program is supposed to do
c - the assurance that the view(s) is (are) correctly implemented.

Once this information are provided, the application programmer will be able, using standard techniques, to prove the correctnes of its application program.

For the formal specification of views we have developed a technique, which is a combination of the ALPHARD technique and of other ADT approaches (such as the algebraic technique). Each operation of the view is formally specified through a precondition and a postcondition predicate. These predicates have a predifined special structure and use as "building blocks" elementary operations proper of the Data Model being used. These elementary operations are assumed to be already defined and proved to be correctly implemented. With this technique the task of proving the correctness of a view is greatly reduced.

Interesting new problems arise when several views and several application programs are considered. Two views defined over the same Data Base can be considered as two abstract objects which share their representation. This situation is very bad for the verification of the programs which use the two objects, since they interfere each other in a rather tricky way. And in fact this situation is usually forbidden by the Programming Languages people (and M.Shaw argue in this sense at the workshop). In Data Base though, this situation can't be avoided,and has to be dealt with.

Our conclusion is that the three types of information defined above are not sufficient anymore. Cooperation (or interference of views must, somehow, be taken into accoun In other words a set of separate specifications of views and application programs is not sufficient to fully characterize the behaviour of the system.

This fact has a strong impact on the Data Base design process. It is impossible to proceed defining external schemas (views) and providing specifications for the application programs, delaying the problem of implementing the views at a later moment. It is impossible, in fact, to provide specifications for the application programs whithout considering the implementation of the views, since this implementation defines how the views interact each other.

Moreover, and more disturbing, whenever the implementation of a view is modified, (even whithout modifying its external specifications), the interactions of the programs,using it, with othr programs are modified.

Our conclusion is therefore that the notion of view (or external schema) must be revisited, at the light of these facts, and must be adapted to the need of verification.

In our opinion the interaction between Data Base design is one of one of the most promisini area of research.