DATA ABSTRACTION TOOLS:

Design, Specification and Application

Joachim W. Schmidt

Fachbereich Informatik

Universität Hamburg

1. Introduction

Current research in data modeling is motivated by the following dilemma:

- At the application level - being confronted with "slices of reality" - details are perceived that, in general, cannot be represented.
- At the representation level - being confronted with "levels of machines" - details are represented that, in general, cannot be perceived.

Abstraction methods cope with that problem by suppressing unnecessary details and by formalizing and structuring the relevant information.

Most attempts to define a more abstract - less detailed - specification of systems use the notion of "domains of objects" for defining system states and of "operations on objects" for defining system transitions. Predicates are used to ensure that objects are "well-formed" and operations are "well-applied".

Domains and operations and the tools and techniques for their definition vary widely depending on the application area. Research and development projects in data modeling currently proceed along three lines:

- At the representational level:
  Data structures and data models are proposed that abstract further from low-level storage characteristics to meet data modeling requirements more directly.
- At the application level:
  Conceptual models are developed that support the design and formal definition of databases.
- At the tool level:
  Tools for the non-representation-oriented specification of programs and programming languages are extended and applied to data and data model definition.

Below, some of the motivations behind this research are given and some results are outlined.

2. Data Representation: Levels of an Abstract Relational Machine

Database applications can be characterized by the following observations on their object domains and on the operations on domains. Regardless of the specific structure of domains, structuring techniques like object classification, aggregation, and generalization [1] seem to be widely applicable. Operations on domains have to support content-oriented object selection, object creation, deletion and alteration and tests of general object properties. Predicates are needed to define constraints like uniqueness and existence of objects.

Therefore, an abstract relational machine that is intended to represent the objects of interest has to support at least the

- definition of relation structures with a general content-oriented selection mechanism;
- execution of relation operations for test, insertion, deletion, and replacement;
- maintenance of classes of predicates on relations, i.e. keys, existence predicates etc.

In Pascal/R [2], an extension of the programming language Pascal by constructs derived from the relational data model, structures and operations are provided at various levels; the levels are related by the fact that higher level operations are abstractions of sequences of operations and control statements at lower levels.

At the lowest level the Pascal/R relational machine provides a minimal set of three constructs operating on relation elements: a key-oriented selection mechanism, a test operation, and an assignment operation. It can be shown that all higher level operations on relations can be represented by a controlled application of these three constructs [3]. At higher levels the assign-

ment operator on relation elements is generalized to insert, delete, and replace operators with whole relations as operands. Relation equality and inclusion can be tested and the denotation for relation elements is extended to denote more than one element. The highest level is characterized by generalized relation test operations up to first-order predicates and by a generalized relation selection mechanism allowing any first-order predicate as a selection criterium.

3. Formal Specification of Abstract Database Machines

Now the questions arise: how should abstract machines be defined, compared, and evaluated?

Some of the formal approaches developed for the semantic definition of programs and programming languages have been applied to database and database model definition: operational approach, e.g. Weber, algebraic approach, e.g. Mayr, Paolini, denotational approach, e.g. Bjørner [4], Lamersdorf [5].

The benefits of a formal semantic specification of database models are numerous:

- An unambigious semantic definition is in any case needed for the intended use and implementation of a model.
- The attempt to define the semantics of a proposed model formally may show major design flaws, e.g., unnecessary restrictions, excessively complex constructs, non-intended side effects, poor interfaces.
- The development of semantic models provide tools, concepts, and principles that assist designers of new languages, e.g., Tennent [6] or database models.

Most semantic modeling techniques - e.g., the denotational approach [7] - require the definition of domains and operations. In the case of the Pascal/R specification [5] the "semantic" domains include types, keys, relations, database; the "syntactic" domains contain the operations, e.g., selection, test, assignment. The domains are defined in terms of sets, tuples, maps etc., and they are further restricted by "is-well-formed" and "is-well-applied" predicates. Finally, the effect of the operations is defined by elaboration functions operating on the objects of the semantic domains.

The crux in the design of any semantic modeling tool is the choice of the primitives for the definition of the object structures, the elaboration functions and the is-well-formed predicates. A good choice decides on the success of the modeling effort and depends heavily on the properties of the area to be modelled. For the definition of programming languages many of the primitives chosen in [7] are closely related to the primitives of programming languages. The effort to define them is done once - if at all?! - in terms of set theory, first-order logic, and lambda calculus. Though the specification of Pascal/R looks rather neat, in parts, the question of the "ideal" primitives for the specification of database models is not yet answered.

4. System Support for Database Modeling and Use

Experience from modeling complex systems indicates that the modeling tools have to be designed quite carefully on the basis of a deep ("theoretical") understanding of the essential concepts of an application area. Whether general-purpose conceptual models exist, that apply equally well to a sufficiently wide range of database applications, is still an open question. In this situation a more pragmatic ("experimental") approach, based on a few, well recognized modeling techniques and supported by an appropriate software environment, seems to be adequate.

A software environment for databases should provide at least the following features.

- A broad interface between database models and programming languages: Database applications require a carefully controlled manipulation of a database. The type compatibility rules, parameter mechanisms, scope and naming rules etc. of the programming language should be extended to apply to the database components. The data and control structures of the language should smoothly interact with the structures of the database model and its selection and test mechanisms. Perhaps, even the terminology (types, expressions, statements vs. schemata, queries, transactions) should be harmonized [3].
- Some form of schema design/redesign support: We have a system under development that supports interactive definition of objects and object structures by means of the so-called relationship techniques [1]. As a result a relational schema is produced together with a set of existence constraints. Procedures can be generated that enforce these constraints by overloading the generic operations on relations.
- Integrated software environment: In an environment supporting database design and use, there is a high demand for integrated editors, compilers, debugging aids, message and screen handling systems etc.

As the workshop session on "Application of Modeling Techniques" shows, there are many projects on the way that provide users of database models with this kind of environment.

5. Concluding Remarks

The three research areas addressed above are not isolated: The formal specification of the Pascal/R semantic led to a couple of revisions in its definition, mainly at the level of the relation-valued expressions (queries) and at the interface between the relational model and Pascal. Furthermore, our effort to support schema design provided hints on possible extensions to the types admitted to relation definition. These extensions enable the schema definition to capture certain existence constraints. Finally, experience in semantic modelling of database models may lead to a better insight into the requirements and limitations of conceptual database modeling in general.

References

[1] Smith J.M., Smith D.C.P. Database Abstractions: Aggregation and Generalization. ACM TODS 2, 2, June 1977

[2] Schmidt J.W. Some High Level Language Constructs for Data of Type Relation. ACM TODS 2, 3, Sept. 1977

[3] Schmidt J.W., Mall M. Pascal/R Report. Fachbereich Informatik, Universität Hamburg, Bericht No. 66, January 1980

[4] Bjørner D. Formalization of Database Models, Lecture Notes in Computer Science, vol. 86, pp. 144 - 228, Springer-Verlag, Heidelberg New York, June 1980

[5] Lamersdorf W., Schmidt J.W. Semantic Definition of Pascal/R, Fachbereich Informatik Universität Hamburg, Berichte No. 73 and 74, July 1980

[6] Tennent R.D. Language Design Methods Based on Semantic Principles. Acta Informatica, Vol 8, 2, 1977

[7] Bjørner D., Jones C.B. (Eds.) The Vienna Development Method: The Meta Language. Lecture Notes in Computer Science No. 61, Springer Verlag, Heidelberg New York, 1978