RELATIVISM AND VIEWS IN A CONCEPTUAL DATA BASE MODEL

Peter Kreps

Department of Computer Science and Applied Mathematics
Lawrence Berkeley Laboratory, University of California
Berkeley, California 94720

The purpose of the Pingree Park Workshop was to bring together practitioners from the fields of Artificial Intelligence (AI), Database Management (DB), and Programming Languages (PL), to discover common issues, and to explore commonalities and differences in approaches to these issues. At the risk of being superficial let me try to summarily characterize the three fields and point to where I think they may fruitfully interact.

It seems to me that at its best, AI is an interdisciplinary science of cognition. It attempts to understand the bases for natural cognition, primarily by developing models of structures and processes that underlie cognition. By incorporation into interactive systems these models can be both exploited as artificially intelligent technology and explored for their adequacy in explaining natural cognition.

On the other hand, at their best, DB and PL are technology-oriented engineering disciplines. DB is primarily concerned with the creation of hardware/software tools for dealing with large quantities of data, including aspects of physical organization, access and caching strategies, logical representation, techniques for specifying associative reference (i.e. query languages), etc. PL is primarily concerned with the creation of hardware/software tools for dealing with problem definition and the specification, verification, and realization of processes and problem solutions.

While DB has focused more heavily on the representation of objects and relationships and PL has focused more on the representation of processes we are seeing that as we become more ambitious about matching technology to high-level applications we cannot focus exclusively on either aspect separately. I see several ways in which AI, as cognitive science, and DB and PL, as engineering areas, can interact and cross-fertilize.

(1) DB and PL technologies suggest representational paradigms for cognitive science to explore.

(2) DB and PL technologies provide support for the implementation of AI models. That is, as the representational power of DB and PL technologies increase it becomes more tractable to implement more refined and complex knowledge bases and models of cognition.

(3) In application environments which require technology that engages, complements, and supports human intelligence in the management of complexity, AI provides models toward which that technology can aim.

The last point is especially important and has a two-fold meaning. The first is that when the problem of managing complexity becomes an issue in an application, models of natural cognition can be of some guidance, since the management of complexity (e.g. by using abstraction) is obviously an important component of natural cognitive activity. Second, one measure of the effectiveness of an intellectual tool is the extent to which that tool matches our own natural cognitive structures. Thus AI models of cognition can help DB and PL technologies achieve a better fit to users.

In the remainder of this paper, I present some work in the area of database conceptual modelling oriented around an application environment which is in many ways exemplary of the remarks above.

A complex application. As part of our work we are attempting to understand the semantic issues in the design of a socio-geographic or "ecological" information system which provides an integrated view of data derived from such sources as the decennial U.S. Censuses of population and socio-economic characteristics, periodic censuses of business and agriculture, and databases of environmental quality, health, and energy use and production. In such a system, complexity arises in two ways.

First, the objects modelled are themselves abstract and intricately interconnected. The data explicitly represent information about various population groupings and geographic regions. However, in the real world many of the attributes of these (abstract) entities are derived via statistical summarizations of the attribute values of underlying entity instances such as individual persons, specific locations, or observations at specific points in time. In addition, entities representing differently defined groupings and summarizations of those underlying entity-instances are present in the collective database (e.g. as a result of different collection methodologies).

Second, this database supports applications (or ad hoc interactions) whose intended views of the data vary widely. Typically, these views involve either further summarization or else statistical redistribution followed by regrouping and summarization.

This kind of system shares with decision support and knowledge representation systems the characteristic that it attempts to provide a structural description of a real world environment that is used as a tool to draw inferences, ascertain implicit relationships (e.g. functional dependencies), and generate and test hypotheses. In the "open" modelling environments typical of such systems it is quite difficult to draw the boundary between what should be included (and modelled) in the database and what can be left out. Such environments usually require an evolutionary approach to the database design task.

Views and relativism. In any database system whose intent is to capture real world meaning, a major requirement is the accommodation of the multiplicity of possible user views. Contemporary proposals for data base system architecture (for example ANSI [1]) recognize this requirement by providing for two separate levels of logical data base description or schema: one which represents an integrated conceptual view of the overall database, and another higher level through which external interactions with the database occur and at which the view of the database may be particularized to a given user or application. It is, of course, a requirement that the idiosyncratic external views and the integrated conceptual view all be mutually consistent, and furthermore, that the external views must be logically derivable from the conceptual view. It is the central goal of logical or conceptual data base design to reflect the mutual consistency of external views in some "minimized" but complete representation of the database which can serve as the conceptual schema.

What then should go into this representation and how can it be structured in such a way as to make the maintenence of external views as easy as possible? This is complicated by something that has come to be called (semantic) relativism. (Cf. the tutorial on semantic database models in this proceedings.) That term denotes the idea that the role that a class of objects seems to play within a database actually varies depending on the perspective of the user or external viewer. Objects which appear in one view as (attribute) values can appear in another as entities and in another as instances of a relationship. Further, an object which is an individual from one perspective may denote a whole class of individuals from another perspective. This implies that in the conceptual model the representation of the semantic relationships in the data should be independent of the labels that may be applied in any given user view. A specific application schema, which may limit certain objects to certain roles, arises from the interaction of a specific set of application goals with this core.

Besides providing the semantic basis for the elaboration of user views, the representation also acts as a specification for physical implementation. Both considerations are best served by relying on a model that is both semantically neutral and structurally simple.

A semantic core model. I would like to propose a model, which I call the "semantic core model", whose goal is to compactly and unambiguously represent the intrinsic semantic relationships in a database such as described above; i.e., to serve as a vehicle for expressing a conceptual schema. In it the sole modelling structures are classes and mappings from one class to another. It draws heavily on the work of Shipman [3], McLeod [2], and Smith and Smith [4,5]. A class both represents and contains a set of instances. Each instance refers to a single real-world individual. This individual may have representative instances in the database in multiple classes. Those instances may be thought of as values, entities, relationships, or classes, depending on context and usage. Each class represents a type. Each class is itself represented as an individual in the database by at least one instance in some meta-class. Mappings define associations between instances of one class and instances of another. Logically a mapping represents an undirected binary relationship. (Of course at lower levels of database description only one or even neither direction may have a direct implementation.) However, even at the logical level, it is convenient to have a separate representation for each direction. Therefore, henceforth, I will use the term "mapping" to refer to one particular direction of a relationship between instances of a "source" class and instances of a "target" class.

Three special interpreted types of mappings are provided. IS relates an instance to another instance of the identical individual in a different class context. ISMEMBER relates an individual to another individual (regarded as a class) to which the first individual belongs. ISSUBSET relates an individual (regarded as a class) to another individual (regarded as a class) whose member instances form a superset of the member instances associated with the first individual. The set of ISSUBSET mappings put classes into a confluent hierarchy of types and sub-types. All other mappings are uninterpreted and may be thought of as defining the target class to be an attribute of the source class. With each mapping also will be associated a set of characteristics (e.g. single/multi-valued, partial/total, into/onto) that will determine certain inter-class constraints related to insertion, deletion and modification of individual instances.

The virtues of this model are structural simplicity, which allows a streamlined physical implementation; and semantic neutrality, which allows a freer expression of higher order semantic abstractions in user views.

It may be noted that there is a convenient and natural mechanism in this model for integrating a description of the schema (that is, the collection of classes and mappings) into the schema itself. Since there is an explicit mechanism for associating sets of individuals with an individual playing a generic role, we can represent class membership itself as a specific instance of this.

Views and roles. I have used the term "view" somewhat loosely up to now. Let me pin it down a bit more precisely. In this model, a view is defined to be a class or a mapping derivable from or definable in terms of operations on other existing classes or mappings via some computation not involving the introduction of new data from outside

the system. An external schema can thus be seen as some (finite) subset of the set of possible views.

A view has two possible motivations from the point of view of a particular application: 1) abstraction, reducing representational detail to emphasize the essential components of the database model; and 2) transformation of perspective, bringing important components of the model close together. The model described above provides a handle on the management and maintenance of views since families of views derivable from a core model schema depend directly on the distinct roles any given class plays within the model. A class potentially plays the following roles:

1. member role: as a set of (abstract) instances of an associated generic individual (see no. 6);

2. source role: as a set of instances of relationship among instances of the target classes of its mappings;

3. target role: as a set of values; i.e., the target class of mappings from some source class;

4. superset role: as a class partitioned into some collection of subset classes according to values of one of its attribute mappings;

5. subset role: as a subset of some superset class;

6. generic role. When the instances of a class have been partitioned into subsets according to the value of a mapping to a target class, then we call that target class a generic class with respect to each of the subsets. Each instance of that generic class can be seen as a generic object which represents (and whose attributes may summarize) the instances of a particular subset.

An example. The example that follows shows how, given a particular semantic core schema, families of derivable user views can be characterized according to the roles played by the different classes. The figure diagrams a hypothetical mortality database giving mortality rates for population groups identified by County, Sex, and Race. The wavy envelope encloses two subset classes, MaleGroups and FemaleGroups, of the class PopGroups, which has been partitioned according to the attribute Sex. Each subset class is also shown with an attribute specific to itself: ProstateMortalityRate for MaleGroups and ChildbirthMortalityRate for FemaleGroups. County is shown having an attribute Location. One set of views, arising from the subset role played by the MaleGroups and FemaleGroups classes defines for them all the attributes (and all functions of attributes) of the superset class PopGroups (e.g. MortalityRate.) Further, for each subset class, the viewed attribute Sex has a single uniform value for all instances. PopGroups can also be viewed as having a mapping to Location, arising from the fact that County is a target for PopGroups and a source for Locations. From the role of Sexes as a grouping class for the subsets of PopGroups, families of views can be inferred that define attributes of the grouping class Sex in terms of aggregate functions over the class PopGroup; e.g., a prototype view GenericAvg could be defined as

```
GenericAvg  =   Avg( A(PopGroup) )
                PER Sex OVER PopGroup
```

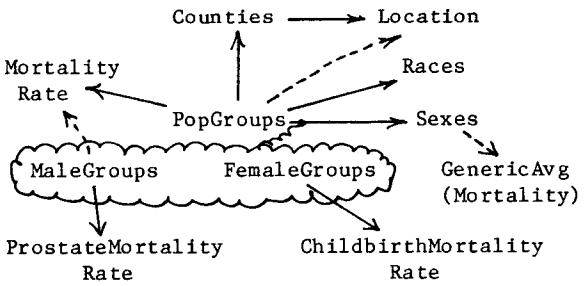

Fig. 1. Hypothetical Mortality Data Base Schema

where A(PopGroup) represents some attribute mapping defined on PopGroup such as MortalityRate.

Summary. In a core schema that attempted to faithfully represent the inter-connectedness of the real world, each class role would need to be made explicit, naming its respectively associated classes and mappings. From each role a characteristic family of derived views can be inferred. In practice, this program need not be fully carried out as there are always connections in the real world which we choose not to model. But it is interesting to imagine an information system that behaved as if all (or at least all obvious) possible views had been automatically defined. It appears that humans have a system for doing some such automatic schema transformation in our natural semantic model of the world. As we constantly shift perspectives we seem always to be able to readily call up whatever role is appropriate for any given object in (mental) view.

The work described here falls in an area that is just on the edge of currently developing DB technologies. It is motivated by a desire to achieve systems that are more natural, more intelligent, and which better complement and support our own intellectual activity in the management and modelling of complex environments. There are, of course, many issues involved here which are far from the arena of cognitive science (especially the more purely technical issues of physical system implementation.) But there are many aspects of the interface toward which these efforts are aimed which remain open questions that AI and cognitive science in general can help to clarify.

[1] ANSI/X3/SPARC Study Group on Data Base Management Systems Interim Report, FDT 7, #2 (1975).

[2] McLeod, D., and R. King, "Applying a Semantic Model," in Entity-Relationship Approach to Systems Analysis and Design, North-Holland, 1980.

[3] Shipman, D., "The Functional Data Model and the Data Language DAPLEX", to appear in ACM TODS (1980).

[4] Smith, J.M., and D.C.P. Smith, "Database Abstractions: Aggregation," CACM, vol. 20, no. 6, June 1977.

[5] Smith, J.M. and D.C.P. Smith, "Database Abstractions: Aggregation and Generalization", ACM TODS, vol. 2, no. 2, June 1977.

This work was supported by the Applied Mathematical Sciences Research Program, Office of Energy Research, Department of Energy under Contract W-7405-ENG-48.