Integrating a Database System and Programming / Information Environment

R. G. G. Cattell
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, California

Vast differences in terminology aside, there is considerable overlap between work in knowledge representation, programming language data types, and database models. Our current work on the Cedar programming environment has necessitated integrating a number of ideas in these areas.

As part of the Cedar project in the Computer Science Lab at Xerox PARC, we have been constructing a database management system. The goal of the Cedar environment is to greatly increase our productivity by combining the best currently known principles from programming languages, programming tools, and user interfaces into a single integrated system. Cedar is based on the Mesa programming language [1]. The goal of our database subproject of Cedar is to provide a uniform prepackaged way to perform access to data structures, as contrasted to the current state of affairs in which Mesa programmers repeatedly re-invent the facilities we intend to provide to type, structure, index, link, robustly store, concurrently access, and cache data stored in the primary or secondary memory of one or more computers on a network.

An initial version of our database system is now operational; we are testing it with some simple applications. This initial system provides to Mesa programs concurrent transaction-based access to distributed data at the level of tuples (records) and their fields, with iteration over tuples satisfying some simple types of queries. In the second phase of our database work, which we have now begun, we are extending this low-level system to make it more useful to both programs and end users. Three tasks are involved in this extension, all of which involve some aspect of data abstraction:

1. The Data Model. We have not concerned ourselves with providing a "semantic" data model to users of the initial system, providing instead a set of primitives on which we believe a number of data models can be constructed. We are conducting longer-term research on the higher-level conceptual data model we will subsequently implement. This research has led us through a number of papers in the database literature as well as some programming language and artificial intelligence work, in an attempt to integrate the wide range of views and terminology into a single satisfactory data model. This model must satisfy our local Cedar clients from all three areas. For example, our database should be suitable for knowledge representation as well as providing an integrated extension of the data types of Mesa. Our survey relating the views of data models, data types, and knowledge representation has formed the basis for the model which is now taking shape.

2. A User Interface. Direct user access to data is as important as program access for many applications. We believe we can provide a single user interface to examine and modify data independent of the type of data and its application, by using meta-data describing the database to control the form of the user display. The user interface should be integrated with the programming environment. Integration means, for example, that a user can call up system utilities such as an editor or compiler on items in the database system; thus the user interface performs the functions of an operating system executive, with the database management system substituted for the file system. The current experimental user interface has been influenced in our lab by Goldstein and Bobrow [2] who constructed an interface of this kind in the Smalltalk programming language, by the author's previous experience with database user interfaces [3], and by other database user interfaces (e.g., [4]).

3. Language Integration. We are providing two types of language access to the database system: a query language, which is connected with the user interface above, and an extension of the Mesa language to incorporate database access at a low level. The latter extension requires redesign of Mesa records, record field access, and record declarations. New control constructs are also necessary to fetch and index through tuples satisfying constraints. The goal here is to unify the features of records (e.g., random record access and fast compiled access code) with those of tuples (e.g., indexes on tuples and dynamically defined tuple types) in an elegant but practical way. ("Practical" means good performance relative to the features used.)

The first of the above tasks, data modelling, was a central topic of the conference. There were few ideas discussed that we had not already encountered in our survey of data models in the literature, but this fact simply reinforces the belief that researchers in the three areas are concerned with much the same problems.

There was surprising commonality in the data models underlying the various representation languages, in all three areas. Some

common features include the basic record type mechanisms equivalent to the relational model with domains, augmented by the introduction of entities (also called atoms, or objects), generalization and aggregation (also called subclasses or subtypes), sets of legal operations for each type, representation of the data schema (type definitions) as data, and automatic invocation of procedures upon data access (views, demons).

It is fortunate that there *was* some agreement on representation mechanisms, because criteria for choosing between data models were not a topic of discussion. Most of the participants in all three areas were certainly designers of such mechanisms. In our own data model design we are again faced with a choice between the alternatives, and we need a basis on which to make this decision. Some criteria might be:

1. *Understandability.* To a first approximation, understandability is inversely proportional to the number of distinct representation primitives necessary to use the model. A word commonly used in the data modelling literature is "naturalness", but this may be so undefinable as to be useless.

2. *Expressive Power.* More sophisticated models incorporate more of the real world's semantics and data integrity constraints within the model framework. To the extent this is achieved by the introduction of more primitives, expressive power may conflict with understandability.

4. *Decomposition of Assumptions.* In particular, a desirable property is the separation of implementation from specification (data independence).

5. *Implementability.* An abstraction that cannot be implemented reasonably efficiently might better be expressed in some other way.

There was less concern at the conference with user interfaces and the view of data provided the end user (usually different from the programmer's interface). The ultimate level of abstraction is the one closest to the user, and it is at this level that all abstractions must come together in an integrated way.

The variety of ways in which the participants combined programming languages and database access was enlightening for our work integrating Cedar database access into Mesa. Some chose to equate programming language records with relational tuples. Others chose to introduce new orthogonal types and operators for database access, feeling that integration with the language was not reasonable at this time. Another dimension of choice is whether to provide aggregate operations on tuples through (1) algebraic operations such as the relational algebra integrated with the language, (2) run-time routines that accept constructed queries (strings or data structures) passed as arguments, or (3) control constructs or generators to iterate through tuples in relations, indexes, and so on. The first of these alternatives is probably the most desirable but also the most difficult.

Artificial Intelligence work has faced issues that have only recently been of serious concern to the database and programming language worlds, and there is some quantity of technology that might be transferred here. Some examples of mechanisms in this category are automatic procedure invocation (views, triggers, demons), complex type systems (multiple inheritance of properties, etc.), complex integrity constraints, and data inference. The database work has in the past been concerned with issues less important to AI, e.g., large, concurrently-accessed, secure, distributed, long-term, efficient, highly reliable database management systems. The idea of building AI systems on top of database systems was suggested more than once. This would allow AI systems to take advantage of the lower-level work in databases, for applications for which this is desirable; however, often the higher-level type mechanisms influence the lower-level implementation. The programming language work has been involved with some of the database concerns as well as others, e.g., more heterogenous data and operations and ease of program development.

A point called to mind by comparison of areas but not always made clear at the workshop is that combining many of the ideas presented is much more complex than a simple juxtaposition of the implementations or syntax. The high-level system models and low-level system implementations just mentioned are just one example of this. Much of our work on the Cedar database is concerned with integration issues of this kind.

Acknowledgements

Mark Brown and Nori Suzuki are co-authors of the database system that is now operational. Peter Deutsch and Jerry Popek assisted us in the design of the system. Mark provided comments on an earlier version of this paper. I would also like to thank John Smith, Dennis McLeod, Steve Zilles, Richard Brodie, Joachim Schmidt, and the other spirited hikers for their absorbing conversations enroute to Stormy Peaks Pass and Cirque Meadows. John Smith deserves particular commendation for remembering a compass.

Bibliography

[1] Morris, J. et al. Early Experience with Mesa, *SIGPLAN Notices*, March 1977.

[2] Goldstein, I., Bobrow, D. Personal communication. Also see paper in this proceedings.

[3] Cattell, R. G. G. An Entity-based Database User Interface. *Proceedings ACM SIGMOD Conference*, Santa Monica, May 1980.

[4] Herot, C. The Spatial Data Management System. *Proceedings of the Conference on Very Large Databases*, Rio de Janeiro, 1980.