

PROPERTY DRIVEN DATA BASES

Jean Paul BARTHES, Michel VAYSSADE, and Monika MIACZYNSKA*
Department of Applied Mathematics and Computer Science
University of Technology of Compiègne
60206 Compiègne Cédex, France

In this paper we suggest how, focusing on the concept of property, one can use a very small number of operators (functions) to organize a factual data base whose information structures are constructed as simple association lists. The approach was used to design and implement a data base management system, operating in a small-core/slow-secondary-storage environment, as a part of a low cost multimicroprocessor robot system.

1. INTRODUCTION

The A.I. community has developed efficient techniques for in core handling of symbolic knowledge representation. On the other hand commercial data bases use extensively secondary storage, but for various reasons suffer from very rigid data structures. Both sides have proposed abstract representation models with the hope to find an efficient solution [1,2,3,4]. However difficulties still subsist when one has to actually implement the models, in particular when secondary storage is involved. Our recent work aimed at building a low cost multimicroprocessor robot system led us to address this problem in a small-core/large-but-slow secondary storage environment. Our objective was to implement a data base using a 16 bit microprocessor. We then reconsidered the simple property value representation which proved to be surprisingly powerful. Indeed it is possible to build easily a data base management system as an interpreter of a set of properties, each being described in turn as a property - value list (introducing readily a metalevel of description). Then, while processing data, the presence of any given property in the data stream can trigger associated functions, which renders the overall approach somewhat analogous to a production rule system. Hence the name of Property Driven Data Base.

The main characteristics of our approach are
- no distinction between physical and logical

*This work was supported in part by CNRS under ATP granted by the Committee for Industrial Robotics and by the University of Technology of Compiègne.

models, which avoids the so-called "data independence" problem plaguing most commercial systems [6].

- no a priori information (or object) classes ; each entity stands for itself. It is represented as a property list and possesses an internal name. The representation is very similar to the one found in Abrial [1].
- operators imbedding semantic meaning are attached to properties.
- relational information is implemented at property level.
- access to the data base is done through specific entry points unlike in McDermott [4].
- privacy and user classes are dealt with using the notion of model and related access paths.
- access paths are computed dynamically.

2. DATA BASE MODEL

2.1 Static View of the Data Structure

It is built from 3 distinct ensembles, E , the entity set or set of objects, P , the property set or set of attributes, and V , the value set. It must be noticed that we do not consider any "relation set". Relations are implemented via the property set which accordingly is partitioned into 2 subsets, T , the terminal property set, and S , the structural property set, with $P = T \cup S$. To each entity e_i of E is associated a subset $P_i = T_i \cup S_i$ of P . To each property t_j of T we associated a subset V_{ij} of V . To every property s_j of S we associate a subset E_{ij} of E . Hence t_j describes local attributes of e_i , whereas s_j gives the connections of e_i with other entities (equivalently the relations). Furthermore for each entity $|P_i| = |S_i| + |T_i|$

may be different and depends solely on what is known about e_i ($|X|$ denotes the cardinality of X).

Elementary Operations: Three basic operators are needed:

- π associating each entity to its corresponding properties: $\pi : E \rightarrow P$ (where P denotes the power set of E)
- τ yielding the values associated to a terminal property t_i of e_i (a restricted case of the LISP function GET): $\tau : E \times T \rightarrow V$
- γ yielding all entities linked to e_i (a different use of GET): $\gamma : E \times S \rightarrow E$

Additional Definitions:

Successor: An entity e_k is a "successor" of a given entity e_i if $e_k \in \gamma(s, e_i)$.

Predecessor: A "predecessor" e_k of e_i is an entity for which e_i is a successor.

Access and Entry Port: Every entity having no predecessor is called an access of the data base.

The data base, G , will be said to be normalized if all accesses are made the successor of a single entity e_0 which thus becomes the only access of G .

However e_0 could be linked to other entities in G . All successors of e_0 will be called entry-ports, abbreviated ports.

2.2 Exploring Mechanisms

Information can be obtained incrementally from any given location in the data base without any a priori knowledge of its organization, simply using operators π , τ , γ . In particular from a static point of view it is possible to obtain full information solely from the elements (e_0, π, τ, γ) e_0 being the data base access. This possibility should be reserved to the data base manager, normal users being excluded.

2.3 Modifying the Data Base

Deleting Information: Deleting values or links at the entity level is straightforward. However deleting an entity requires updating of all its predecessors.

Adding Information: is a much more interesting operation. Indeed the overall structure of the data base must remain consistent. It is convenient to associate to each terminal property t_i of P a screening operator ξ_j such that for any external value W :

$$\xi_j(W) = \begin{cases} W & \text{if } W \text{ is consistent with } V_{ij} \\ \emptyset & \text{otherwise} \end{cases}$$

In practice the corresponding function carries on such tasks as validity checks and external/internal format translation. This permits to associate a specific data type to any given property, and to support easily heterogeneous collections of data.

Retrieving Information: Let us consider a subset E_i of E , a property p_i of P and an external set of values W consistent with P_i , i.e. such that $\xi_j(W) \neq \emptyset$. One can define one or more selection operators ω_i associated with p_i , extracting from E_i all elements satisfying a given criterium: $\omega_i(E_i, W) = \{e \in E_i \mid \omega_i(e, W) \neq \emptyset\}$. ω_i is a mapping $\omega_i : E \times W \rightarrow E^j$. The involved criterium can be represented by a Boolean function β_j , e.g.

$$E = \{e_i \mid \beta_j(V_{ij}, W)\} \quad \text{if } p_j \in T_i$$

The selection operator ω_i is similar to Lindgreen's detection operator [3] but is not constrained to "HIS" (Homogeneous Information Classes). More will be said concerning its implementation.

2.4 Accessing Structures

Map and Models: A large data base especially in A.I. type of applications may contain many chunks of information stored in a very disorderly fashion. Whenever secondary storage is used it is necessary to minimize the number of accesses. We introduce to this purpose the notion of map and models. In addition to increasing access efficiency they permit to solve neatly some of the problems related to privacy and user's view of the data base. An interesting feature of such models is the possibility to construct them a posteriori using pattern recognition techniques. In order to define them we need some additional concepts.

Following Property: Property P_i will be said to be following a property P_j associated with entity e_i , if e_i is a successor of e_j and $P_i \in \pi(e_j)$.

We can define mapping Γ_a such that $\Gamma_a : S \rightarrow P$ and extend it over the set of properties $\Gamma_a^* : P \rightarrow P$.

Map: It is the graph $G_a = (P, \Gamma_a^*)$ connecting properties.

Model: A model is any subgraph of the map. A model implements a user's view of the data base.

Exploring Using Criteria: Let E_i a subset of E be named a position. Let χ be an operator which when applied to E_i will yield a subset E_q deri-

ved from a given property p_k and a given model M^n : $\chi(E_i, p_k, M^n) = E_q$ where $\chi: E \times P \times M \rightarrow E$

E is the set of entities that can be reached from E_i by using some existing path in M^n leading to a property p_k . Access paths are dynamically computed in a model.

Clearly some entities in the data base having property p_k will not be reached, even from e_o , if there are no existing paths in the given model M^n . Furthermore if, instead of M^n , one considers subgraphs of M^n , access to entities will be further restricted. For example if E_i is a set of entities each one having property p_i , one can easily obtain all entities having both properties p_i and p_k by applying $\chi(E_i, p_k, \{p_i\})$. An analogous result can be obtained from any position E_i using $\chi(\chi(E_i, p_j, M^n), p_k, \{p_i\})$.

Associative Access: It is then possible to combine χ with previously defined selection operators. Using a screening operator (or filter) ω , associated with a property p_i , one can distinguish 2 ways of filtering information:

- cumulative filtering $E_q = \omega_k(\omega_k(E_i, W_k), W_i)$ yielding all entities from position E_i satisfying 2 (or possibly more) criteria associated with one (or more) properties.

- repetitive filtering $E_q = \omega_k(\chi(E_i, p_i, M^n), W_i)$ yielding all entities that can be reached from position E_i according to the model M^n and that satisfy criteria implemented by ω . Obviously such filtering can be applied recursively.

Classes of Users, each having various access privileges, can be defined by means of domains: $D_u = (M_u, P_u, \Omega_u, Z_u)$ including a set of models M_u , a set of properties P_u , associated filters Ω_u and screening operators Z_u .

3. IMPLEMENTATION AND DISCUSSION

An implementation was done at the UTC on a PDP 11. The host language is a shallow binding LISP dialect, functions were developed allowing to store and to access directly variable length character strings identified by variable length keys, sequential access was added to the LISP I/O subsystem for reading bulk data and for producing printed forms. The volume of object code limited us to 5000 list cells and 1250 atom hash cells. This realizes in effect a small processing window over the data base held in secondary storage. An additional primitive for reclaiming atom space had to be developed for dealing with the clogging of the hash table.

Data is structured as association lists. Entities linked by structural properties are reversely chained. Ports are implemented via the hash table. Operators are implemented as LISP functions

A present restriction in our approach is the lack of possible deduction when the search fails. However the general structure makes it rather straight-forward to introduce "advising functions" at the property level. Furthermore screening operators could be implemented as microprograms to be loaded into "stream processors" to take full advantage of the recent developments in secondary storage architecture [6]. On the other hand, although path access leads to iterative retrieval of entity subsets, it offers a solution to the privacy problem and to the implementation of specific user's view of the data base.

REFERENCES

- [1] Abrial, J.R. "Data Semantics", in Data Base Management, ed. J.W. Klimbie and K.L. Koffeman North Holland, 1974.
- [2] Codd, E.F. "A Relational Model of Data for Large shared Data Banks", CACM 13:6 (1970) 377
- [3] Lindgreen, P. "Basic Operations on Information as a Basis for Data Base Design", in Information Processing, North Holland, 1974
- [4] Mc Dermott, D.V. "Very Large Planner-Type Data Base", MIT - AI Memo 306, 1974.
- [5] Date, C.J. "An introduction to Data Base Systems" Addison-Wesly, 1977.
- [6] Computer Issue of March 1979.