Frames, Objects and Relations: Three Semantic Levels for Knowledge Base Systems*

M. C. Norrie[1], U. Reimer[2], P. Lippuner[2], M. Rys[1], H.-J. Schek[1]

[1]Dept. of Computer Science, Swiss Federal Institute of Technology (ETH),
CH-8092 Zürich, Switzerland
{norrie, rys, schek}@inf.ethz.ch
[2]Swiss Life, Informatik-Forschungsgruppe, CH-8022 Zürich, Switzerland
{reimer, lippuner}@swssai.uu.ch

Abstract

We propose an architecture for large-scale knowledge base systems based on database technologies and the three levels of semantic construct - frames, objects and relations. The intermediate object level retains the structural semantics of the frame level and is therefore beneficial in bridging the semantic gap between the frame and relational levels and enabling the use of semantic information in query optimisation. Specifically, we outline how this approach has been adopted in the hybrid knowledge base system, HYWIBAS.

1 Introduction

For knowledge base systems to be effective for large-scale applications, it is essential that they support efficient retrieval and update operations on large, shared knowledge bases. Database system research has focussed on issues of performance and concurrent access to large data sets and we wish to exploit the resulting technologies for the storage and management of knowledge bases.

Past research in this area has tended to use relational systems for the persistent storage of knowledge bases. While this strategy does meet the requirements of controlled data sharing, the large semantic gap between the knowledge representation structures and the relational structures makes it more difficult to utilise data semantics in query optimisation. We therefore adopt a two-level mapping. The first level maps a frame-based knowledge representation model, FRM [Rei 89; RL 94], to an object data model, COCOON [SLR+92], which retains much of the data semantics. The second level then maps COCOON to a relational system which is used as a simple storage system with query and update strategies controlled primarily at the object system level.

Here, we present an overview of how this approach is utilised in the (hybrid) knowledge base system HYWIBAS [RRS+93] (the hybrid aspects are not elaborated here). Section 2 introduces the three level architecture and discusses its merits. The mappings from FRM to COCOON and from COCOON to a relational system are discussed in Sections 3 and 4, respectively. Some remarks on the current status of HYWIBAS and future research plans are given in Section 5.

2 Three Level Architecture

Knowledge base systems research has tended to concentrate on issues of semantic expressiveness and inference mechanisms. For knowledge base systems to be used for large-scale applications, issues of efficient update and retrieval operations on large, shared knowledge bases must be addressed. Database systems research has focussed on these very issues in dealing with efficient, concurrent access to large data sets. The question then becomes one of how best to exploit database technologies in knowledge base systems.

Relational database technologies now have established and well-understood mechanisms to support efficient access to large sets of value tuples with techniques for concurrency control and recovery. The problem of mapping a knowledge model directly to a relational storage structure is the large semantic gap due to the lack of semantic expressiveness of the relational data model. As described in [RS 89], this can in part be overcome by mapping a knowledge model to a nested relational model which can represent complex structures directly. However, the nested relational model does not support notions of type inheritance and concept hierarchies which are fundamental to knowledge models such as FRM.

Object data models have been developed to support notions of semantic data modelling and thereby increase the semantic expressiveness of the data model. They have constructs to represent both complex structures and relationships between structures – including those that arise in classification structures, often known as *isa* hierarchies. In addition, a number of object data models have been proposed that specify operations over collections of objects in terms of an object algebra. By mapping the frame knowledge model to an object data model rather than to a relational data model, the semantic gap is reduced. However, object-oriented database management systems are not yet as well established as

* The work presented here was supported by the Swiss Priority Programme in Computer Science under Grant No. 5003-34347.

relational database management systems in terms of efficient processing of set-oriented retrieval and update operations and supported transaction mechanisms. For this reason, we choose to map our object data model to a relational storage system. This mapping is specifically tailored to support the retrieval and update patterns initiated by the frame model. As a result, we have a three level architecture as indicated in Figure 1.

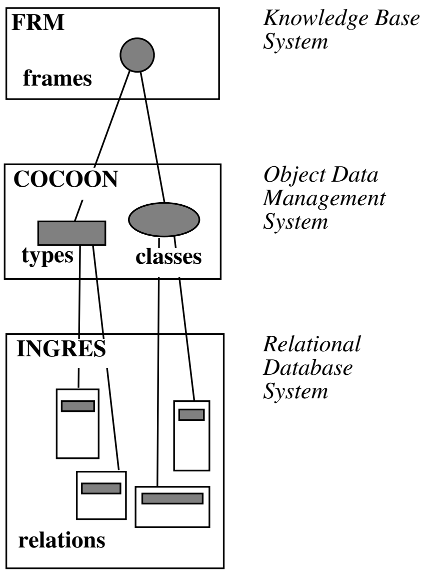

Figure 1: Three Level Architecture

The knowledge model FRM is mapped to the object data model COCOON which in turn is mapped to a relational system. At present, we use the relational data base management system INGRES, but the mapping can easily be altered for other relational systems.

3 From Frames to Objects

A discussion of the differences between the knowledge representation and semantic data modelling approaches is given in [Bor 91]. One of the main differences often quoted is that database models tend to be prescriptive rather than descriptive. Thus the underlying assumption is that the database provides a complete, current and consistent description of the application domain; any attempt to input data which is not consistent with the database model will be rejected. Knowledge models tend to be descriptive and it is quite acceptable that the model may have to be revised according to new information received into the system. This is most clearly visible in a knowledge-based system with some learning capabilities (see e.g. [Mor 91]).

A further general distinction between data models and knowledge representation languages is the fact that data models have a much clearer separation between intensional and extensional information. Intensional information is given by a database schema which is relatively stable and thus plays a predominant role in determining efficient storage, retrieval and update strategies for operations on extensional data.

Ideally, for the support of knowledge base systems, we wish to have the latter property of database models (i.e. efficiency) but not necessarily the former (i.e. being prescriptive). In this respect the COCOON object data model is a good candidate for the support of the frame model FRM.

In this paper we consider only a subset of FRM which corresponds to the common frame constructs: slots, slot entries, and cardinality restrictions. For example,

$$
\begin{array}{l}
\textit{Skilled-Person} \doteq \\
\quad (\textbf{and}\ \textit{Person} \\
\qquad (\textbf{all}\ \textit{has-skills}\ \textit{Skill}) \\
\qquad (\textbf{exist}\ \textit{has-skills}\ \textit{Rare-Skill}) \\
\qquad (\textbf{atleast}\ \textit{has-skills}\ 3))
\end{array}
$$

defines a frame class *Skilled-Person* as a subclass of *Person* with the slot *has-skills* that represents the relationship *has-skills* to the class *Skill*. The slot requires at least 3 values at an associated class instance; one of those entries must be an instance of the class *Rare-Skill*.

COCOON has a strong influence from both semantic data models and knowledge representation languages (especially KL-ONE [BS 85]) in terms of semantic expressiveness. It supports not only complex object structures but also rich classification structures and high-level operations over collections of objects. As a result, the semantic expressiveness of COCOON is at a similar level to that of FRM with the main difference between the two models stemming from the fact that FRM supports more specialised inference mechanisms. In some sense COCOON may be considered as lying somewhere between the prescriptive and descriptive paradigms. A COCOON class represents a semantic grouping of objects and may have an associated predicate condition. For example

define class *Youngsters* : *person* **some** *Persons*
  where *age* < 30;

defines a class *Youngsters* which contains objects of type *person* and is a subclass of *Persons*; further there is an associated predicate condition that specifies that its members should be less than 30 years old. The object type *person* declares what functions are applicable to an object of that type and may look like the following

define type *person* = *age* : **integer**,
  name : **string**, *has-skills* : **set-of** *skills*;

A formal mapping from frame structures to object structures and from query operations on frame knowledge bases to object bases has been defined and implemented. While concept class descriptions in FRM are based on a single representation structure – the frame, COCOON has two basic representation structures – the type and the class. Types describe what properties and relationships to other objects an object can have whereas, as stated above, classes deal with semantic groupings of objects. Only a small number of the frame constructs for concept class descriptions can be mapped to COCOON

FRM concept class description:

$$\begin{array}{lcll} Comp\_Delivery & \doteq & (\textbf{and} & (\textbf{all}\ supplier\ Company) \\ & & & (\textbf{exist}\ supplier\ Computer\_Company) \\ & & & (\textbf{all}\ recipient\ Company\ Person) \\ & & & (\textbf{atmost}\ recipient\ 1) \\ & & & (\textbf{all}\ ispart\ Workstation) \\ & & & (\textbf{all}\ price\ [0,100]) \\ & & & (\textbf{atmost}\ price\ 1)) \end{array}$$

Corresponding COCOON type definition:

$$\begin{array}{lcl} \textbf{define type}\ comp\_delivery & = & supplier : \textbf{set-of object}, \\ & & recipient : \textbf{object}, \\ & & ispart : \textbf{set-of object}, \\ & & price : \textbf{integer}; \end{array}$$

Corresponding COCOON class definition:

$$\begin{array}{rcl} \textbf{define class}\ Comp\_Delivery & : & comp\_delivery \\ \textbf{where} & & supplier \subseteq Company\ \textbf{and} \\ & & \emptyset \neq (supplier \cap Computer\_Company)\ \textbf{and} \\ & & recipient \subseteq (Company \cup Person)\ \textbf{and} \\ & & ispart \subseteq Workstation\ \textbf{and} \\ & & \emptyset = \textbf{select}\ [(i < 0)\ \textbf{or}\ (i > 100)]\ (i : price); \end{array}$$

Figure 2: Example of Mapping an FRM Concept Class Description to COCOON Types and Classes

type definitions but all of them to COCOON class definitions. As a consequence, frames of FRM are mapped to some combination of types and classes in COCOON. To increase the possibilities for compile-time optimisation, we designed the mapping such that as much information as possible is provided on the type level.

Figure 2 shows an example of mapping an FRM concept class description to COCOON types and classes. In a first step the object type *comp_delivery* is derived from the FRM class *Comp_Delivery* such that for every **all** construct (i.e. for every slot) we have a function with the same name. In case of a slot with a maximal cardinality of 1 the function is single-valued, otherwise set-valued. In a second step the COCOON class *Comp_Delivery* of type *comp_delivery* is generated from the frame class *Comp_Delivery*. With the type reference we ensure that the class will contain only objects with the right functions being applicable. With the associated class predicate we cover the remaining features of the FRM concept class description. As a result, the COCOON class defines the same necessary and sufficient conditions on class membership as the frame class does. Note that the three object-valued functions in the type definition *comp_delivery* are all of type **object**. This is because providing more specialised function ranges (e.g. *supplier* : **set-of** *Company*) would not lead to a simpler class predicate. As this would not reduce the amount of dynamic type checking necessary we decided to keep the mapping to the type level simple and to map always to object-valued functions of type **object**. For details see [LNR+94].

The establishment of the mapping from frames to types and classes has also proved useful in providing an insight into the similarities and differences in the fundamental concepts of terminological models such as FRM and object data models.

In knowledge base systems a query for objects with certain properties is usually established as a class description. The result of the query is all the objects subsumed by that class so that in this case query evaluation amounts to inferencing. To support such queries on our COCOON-based FRM we have specified a second mapping that transforms a frame class description to be interpreted as a query into an equivalent expression of the COCOON object algebra (cf. example in Figure 3). This algebra expression is then evaluated on the COCOON object base derived from the original frame knowledge base. At that point query optimisation techniques, which are highly developed in the database area, can be employed. We hope that this will lead us to a query processing that is much more efficient than evaluating a query frame by the inference mechanism of FRM.

4 From Objects to Relations

In mapping an object data model onto a relational system, there are many choices to make concerning both the representation of objects and also of classes. For example, all the properties of an object may be stored together in a single relation or split over several relations. In the former case, there are problems of how to represent multi-valued properties. In the latter case, several join operations may be required to reconstruct an object.

With the representation of classes, the choices arise because an object may belong to many classes and the prime decision is whether to store an object only with its most specific class – or to store it in all classes – or to have some form of compromise between the two extremes. Further, some COCOON classes have associated predicates which specify necessary and sufficient conditions for membership of that class. In such a case, there is no need to store

Query Frame:

(**and** (**all** *supplier Company*)
(**all** *recipient Company*)
(**exist** *recipient Insurance_Company*)
(**all** *product*
(**and** *Workstation*
(**all** *has-cpu Sparc*) (**atleast** *has-cpu* 2))))

Corresponding Algebra Expression:

$$\mathbf{select}[supplier(o_1) \subseteq Company](o_1 : Objects) \cap$$
$$\mathbf{select}[recipient(o_1) \subseteq Company](o_1 : Objects) \cap$$
$$\mathbf{select}[recipient(o_1) \cap Insurance\_Company \neq \emptyset](o_1 : Objects) \cap$$
$$\mathbf{select}[product(o_1) \subseteq \mathbf{select}[has\text{-}cpu(o_2) \subseteq Sparc] \cap \mathbf{select}[\#(has\text{-}cpu(o_2)) \geq 2](o_2 : Workstation)](o_1 : Objects)$$

Figure 3: Example of Mapping a Query Frame to an Object Algebra Expression (still to be Optimised)*

the class explicitly as it can be derived at access time. The trade-off here is between fast access to explicitly stored classes versus high update overheads if data is replicated unnecessarily.

In our mapping of COCOON onto a relational storage system, we employ extensive replication to minimise retrieval costs. For example, all classes are represented explicitly even those which could be specified in terms of a query expression (view) over other classes. Since an object may belong to many classes, an object representation may be replicated in several relations. The penalty associated with such an approach of massive replication is the cost of update operations; a single update operation on a specific object may require updates on a large number of relations involved in the representation of that object.

The problem then becomes one of how to speed up the time for updates. This is achieved by implementing the update operation as a number of simpler update operations which can be executed in parallel. The exploitation of intra-transaction parallelism together with multi-level transactions is a key technique towards such improved performance [WS 92].

We are currently evaluating the above approach to see under what conditions the overheads of parallelisation are compensated by the corresponding speed-up of the operations. In the future, we shall investigate dynamic methods of mapping the object data model COCOON to relational systems such that good performance is attained under various retrieval and update patterns (which finally stem from specific retrieval and update operations on the knowledge base system).

5 Conclusions

In the HYWIBAS project, we are using database technologies to support large, shared knowledge bases. We employ a three level architecture corresponding to three semantic levels of frames, objects and relations. The introduction of the object level is beneficial in reducing the semantic gap between the frame level and the relational level and enabling the utilisation of structural semantic information for query and update processing. The mapping from the object level to the relational level allows the use of well-established, efficient mechanisms for data storage, data access, data sharing and recovery under failure.

At present, we have implemented mappings for structural information from the frame model, FRM, to the object model, COCOON and from COCOON to the multiprocessor relational database system, INGRES. We also have a mapping from frame query classes to COCOON algebra. Moreover, there are some early results on the parallelisation of update operations over a COCOON database represented in INGRES [Rys 94]. Currently, we are working on the mapping of the remaining operational components and on the mapping of frame class instances to objects.

* For reasons of readability we have slightly simplified the algebra expression: The select statements should apply to classes of objects for which the functions referred to are really defined, rather than operating on the most general class *Objects*. This requires an additional meta-schema query, which we have omitted.

References

[Bor 91] A. Borgida, "Knowledge Representation, Semantic Modeling: Similarities and Differences", In *Entity-Relationship Approach: The Core of Conceptual Modelling*, ed. H. Kangassalo, North-Holland, 1991, pp. 1-24.

[BS 85] R. J. Brachman and J. G. Schmolze, "An overview of the KL-ONE knowledge representation system", *Cognitive Science*, Vol. 9, No. 2, 1985, pp. 171-216.

[LNR+94] P. Lippuner, M. Norrie, U, Reimer and M. Rys, "Mapping a Frame Model, FRM, to an Object Data Model, COCOON", HYWIBAS Working Paper, 1994. (in preparation)

[Mor 91] K. Morik, "Underlying Assumptions of Knowledge Acquisition and Machine Learning", *Knowledge Acquisition*, Vol. 3, 1991, pp. 137-156.

[Rei 89] U. Reimer, "FRM: Ein Frame-Repräsentationsmodell und seine formale Semantik. Zur Integration von Datenbank- und

Wissenrepräsentationsansätzen", Springer, 1989.

[RL 94] U. Reimer, P. Lippuner, "Syntax und Semantik von FRM", Working Paper, 1994, Informatik-Forschungsgruppe, Swiss Life, CH-8022 Zurich).

[RRS+93] U. Reimer, M. Rys, H.-J. Schek and R. Marti, "Datenbankbasierung eines Frame-Modells: Abbildung auf ein Objektmodell und effiziente Unterstützung komplexer Operationen", Beitrag zum Workshop "Verwaltung und Verarbeitung von strukturierten Objekten" während der KI 93, (also available as Technical Report 5/93, Informatik-Forschungsgruppe, Swiss Life, CH-8022 Zurich).

[RS 89] U. Reimer and H.-J. Schek, "A Frame-Based Knowledge Representation Model and its Mapping to Nested Relations", *Data and Knowledge Engineering*, Vol. 4, No. 4, 1989, pp. 321-352.

[Rys 94] M. Rys, "Parallelising Generic Update Operations in COCOON Using Multilevel Transactions". (in preparation)

[SLR+92] M. H. Scholl, C. Laasch, C. Rich, H.-J. Schek and M. Tresch, "The COCOON Object Model", Technical Report 211, Dept of Computer Scince, ETH Zurich, CH-8092 Zurich, Switzerland.

[WS 92] G. Weikum, H.-J. Schek, "Concepts and Applications of Multilevel Transactions and Open Nested Transactions", In *Database Transaction Models for Advanced Applications*, ed. A.K. Elmagarmid, Morgan Kaufmann, 1992.