

Semantic Data Modeling of Hypermedia Associations

John L. Schnase
John J. Leggett
David L. Hicks
Ron L. Szabo

Hypermedia Research Lab

OCTOBER 1991
TAMU-HRL 91-005

Table of Contents

1. INTRODUCTION	1
2. SEMANTIC DATA MODELING	2
2.1 Characteristics of Semantic Models	3
2.1.1 <i>Early History.</i>	3
2.1.2 <i>Organization of Semantic Models.</i>	3
2.1.3 <i>Existing Semantic Models.</i>	8
2.2 The Saberel Semantic Network Database System	9
3. HB1: A CASE STUDY IN SEMANTIC DATA MODELING	10
3.1 The Hypertext Research Lab Perspective	11
3.2 HB1 Organization	12
3.2.1 <i>Storage Manager.</i>	12
3.2.2 <i>Object Manager.</i>	12
3.2.3 <i>Association Set Manager.</i>	13
3.3 Semantic Modeling of Associations	13
3.3.1 <i>Semantic Modeling of OM Data.</i>	14
3.3.2 <i>Semantic Modeling of ASM Data.</i>	14
3.4 HB1 Operation	15
4. RELATED WORK	16
5. DISCUSSION	16
5.1 An Evaluation of Semantic Modeling	16
5.1.1 <i>Semantic vs. Relational Approaches.</i>	17
5.1.2 <i>Semantic vs. Object-Oriented Approaches.</i>	19
5.2 An Evaluation of Saberel	20
6. FUTURE WORK	20
7. SUMMARY	21

Table of Contents

ACKNOWLEDGMENTS	21
REFERENCES	22

Semantic Data Modeling of Hypermedia Associations

JOHN L. SCHNASE
JOHN J. LEGGETT
DAVID L. HICKS
RON L. SZABO

1. INTRODUCTION

Interconnected information is fundamental to hypermedia. As a result, many important issues in the design and implementation of hypermedia system functionality focus on the way these connections, or associations, are represented, manipulated, and stored. In related work, we have described the initial design and implementation of a hyperbase management system (HBMS) called HB1 [41]. Among HB1's distinctions is its use of the Sabrel Semantic Network Database System to manage physical storage. In this paper, we discuss our experiences using a semantic database and show how semantic modeling has been useful for representing the structurally complex interrelationships that commonly arise in hypermedia.

The HB1 hyperbase management system supports an extensible, object-based system architecture for distributed, inter-application linking. This is an architectural organization that we feel is likely to characterize the next generation of hypermedia systems. HB1 is referred to as a *hyperbase* management system because it supports not only the storage and manipulation of hypermedia information, but the storage and manipulation of connectivity data as well. HB1 and its underlying data model make a strong distinction between the structural and behavioral aspects of hypermedia, and it is this emphasis on structural abstraction that has prompted our interest in semantic modeling.

Semantic data models attempt to provide more powerful mechanisms for structuring objects than are typically provided by traditional approaches. Relational methods, for example, emphasize information structures that promote efficient storage and retrieval rather than those that accommodate inter-object relationships. Object-oriented databases, on the other hand, provide predominantly behavioral abstractions. In the organization of HB1, we wanted to be able to manage

the inter-object connectivity of hypermedia separate from the objects involved in the connectivity. In addition, there was a requirement of separating connectivity data from those objects in the database that implement hypermedia functionality. Building HB1 on top of a semantic database system facilitated such a separation and has made the structural aspects of hypermedia conveniently accessible to independent manipulation.

We continue our introduction in Section 2 by presenting an overview of semantic data modeling. We also describe Saberel in this Section. Section 3 presents a case study in the use of a semantic database system to support hypermedia. Here, we present our concept of hypermedia and show how it has been effectively mapped to a semantic representation in HB1. In Section 4, we show how our work relates to other research appearing in the hypermedia literature and, in Section 5, consider the advantages and disadvantages of semantic modeling. Section 6 outlines areas for future research, and Section 7 summarizes the major points of the paper.

2. SEMANTIC DATA MODELING

Traditional database management approaches tend to emphasize techniques that promote efficient storage and retrieval of fixed information structures. For example, the relational, hierarchical, and network models use a simple record-based format, and they generally lack direct support for relationships, data abstraction, inheritance, constraints, and unstructured objects [27]. Recently, however, greater consideration has been given to the user's perception of data rather than its physical representation. This trend has resulted in a requirement for richer, more expressive data structuring capabilities and has led, in turn, to a renewed interest in semantic data models. Semantic models attempt to provide more powerful abstractions for specifying database schemas than are supported by traditional models [27, 37, 38].

In this Section, we summarize the general characteristics of semantic data models, then describe the Saberel Semantic Network Database System which has been used in our research. Later in the paper, we show how semantic modeling differs from other data modeling approaches, and, more importantly, we identify the advantages it holds for hypermedia system designers. Most of the following discussion is distilled from an excellent overview of the area provided in [27, 37].

2.1 Characteristics of Semantic Models

2.1.1 *Early History.* Semantic models were initially developed to facilitate the design of database schemas [9, 24, 40, 45]. A schema could first be formulated using the higher-level abstractions of a semantic model, then translated into one of the traditional database models for implementation. Since the emphasis of semantic models was to capture data relationships as they exist in "real-world" settings, they were often more complex and, as shown below, tended to encourage a navigational view of the data.

Semantic models were introduced in the early 1970s [2]. Since then, research has resulted in the development of powerful mechanisms for representing the *structural* aspects of data, and, more recently, its dynamic or behavioral characteristics as well. In addition, these advances have accompanied a growing interest in developing semantic models into full-fledged database management systems. A number of these systems now exist and offer a wide range of data modeling capabilities.

2.1.2 *Organization of Semantic Models.* The piece of the "real-world" represented by a semantic model is referred to as an *enterprise*. A *semantic schema* represents important elements within an enterprise and shows the structural interrelationships among those elements. Virtually all semantic data models offer a diagrammatic construct for conceptualizing schemas. Here we use the Hypertext Community Database schema shown in Fig. 1 to illustrate the basic organization of semantic models. Our example is loosely based on Hull and King's [27] Generic Semantic Model (GSM) constructs.

The fundamental components used by semantic models to structure data are objects, atomic and constructed types, attributes, ISA relationships, and derived schema components. We begin by discussing objects.

(1) **Objects** – Elements within an enterprise are represented by objects or entities. It is generally assumed that these objects can be any unstructured data such as strings, integers, and reals, or, in the case of multimedia enterprises, text, voice and image data [10, 55].

(2) Atomic types – The direct representation of object types, distinct from their attributes, is essential to semantic modeling. As the name implies, atomic types correspond to classes of simple, nonaggregate objects in an enterprise.

The Hypertext Community schema in Fig. 1 shows the organization of object types in a hypothetical enterprise. In it, two atomic types, PERSON and EMPLOYER, are represented. These are depicted using triangles to indicate that they are *abstract* or nonprintable types. Abstract types generally correspond directly to physical or conceptual objects in an enterprise. Depending upon a particular model's implementation, these types may not be visible to the user. In contrast, *printable* types, such as PNAME and OCCUPATION, would have an external representation accessible to the user. These appear as ovals in Fig. 1. The distinction between abstract and printable types is important because, as we will show, objects in a semantic data model may not be uniquely identifiable by their printable attributes.

Conceptually, in an *instance* of a schema, an *active domain* is associated with each node in the schema. The active domain of an atomic type holds all objects of that type currently in the database. In our example, a set of objects of type PERSON would be associated with the PERSON node, objects of type EMPLOYER would be associated with the EMPLOYER node, and objects of type PNAME would be associated with the PNAME node, etc.

(3) Constructed types – Perhaps the most important feature of semantic models is their ability to construct complex object types from atomic types. *Aggregation* and *grouping*, also called association, are the most common *type constructors* in the semantic literature.

An aggregation is a composite type formed from other types already existing in the schema. For example, ADDRESS, represented in Fig. 1 by a \otimes -node, is an aggregation of three printable types, STREET, CITY, and ZIP. Mathematically, an aggregation is an ordered n -tuple, and, in an instance of the schema the active domain of an aggregation type will be a subset of the Cartesian product of the active domains assigned to its underlying nodes. The aggregation abstraction allows users to focus on a complex data type while ignoring its component parts.

The grouping constructor, on the other hand, is used to represent sets of objects of the same type. In Fig. 1, HYPERTEXT SYSTEMS, represented by a \oplus -node, consists of sets of HT SYSTEMs.

Mathematically, a grouping is a finitary powerset. In an instance, the active domain of a grouping type will consist of a set of objects, each of which is a finite subset of the active domain of the underlying node. Although aggregation and grouping define new object types from previously defined types, they are fundamentally distinct abstractions. Aggregation, in effect, provides a means for specifying the attributes of a new object type (see below), whereas grouping defines a type whose value will be a set of objects of a particular type. In most semantic models supporting aggregation and grouping, there can be no directed or undirected cycles of type constructor edges.

(4) Attributes – Another fundamental aspect of semantic models is their ability to represent interrelational dependencies or connections among object types. These properties are called attributes or relationships. Attributes are generally specified explicitly by the user to correspond to facts about an enterprise.

In the Hypertext Community schema, PERSON has three attributes: HAS-NAME, LIVES-AT, and USES. In this schema, attributes are represented by arrows that originate at the domain of each attribute and terminate at its range. That is to say, the HAS-NAME attribute maps an object of type PERSON to a printable object of type PNAME with similar mappings implied by the other attributes. A range object can be referred to as the *value* of the mapping attribute.

In a more formal sense, there are several ways attributes can be expressed in semantic models. The HAS-NAME attribute is an example of a *one-argument* attribute, i.e. it is a directed, binary relationship between two types. Some models allow *n-argument* attributes, which define a directed relationship between a set of n types and one type. This situation is not shown in our example, but could be denoted by an n -tailed arrow. Attributes can also be either *single-valued* or *multivalued*. An ADDRESS, for example, can be the residence of several PERSONs. In our schema, single- and multivalued relationships are distinguished by one- and two-headed arrows respectively.

In an instance of the schema, a mapping is assigned to each attribute. The mapping will consist of either a binary or $(n+1)$ -ary relation, depending on the number of arguments n in the attribute. The domain of the mapping is the Cartesian product of the active domain(s) of the attribute's source(s), and the range is the active domain of the attribute's target. In the case of single-valued attributes, such as HAS-NAME, the mapping must be a function in the strict mathematical sense.

The distinctions between various forms of type constructors and attributes help explain some of the variability one sees in the expressiveness of semantic models. For example, there is a close correspondence between the semantics of a multivalued attribute and the semantics of a single-valued attribute whose range is a constructed grouping type. Likewise, a one-argument attribute whose domain is an aggregation and an n -argument attribute are similar. It should be clear that several ways of representing essentially the same data interrelationships can exist given a full range of modeling functionality. Most existing models, however, only support multivalued attributes and do not permit an attribute to map to a grouping type.

What may not be quite so obvious is that the semantics of object identity are strongly influenced by the abstractions chosen in a representation. All COMPUTER SCIENTISTs in our schema work for an EMPLOYER at an OCCUPATION. In this particular representation, the underlying identity of a COMPUTER SCIENTIST is independent of its associated EMPLOYER or OCCUPATION values. That is, changing a COMPUTER SCIENTIST's OCCUPATION would not, in a strict sense, change their identity. However, if we had chosen to represent COMPUTER SCIENTIST as an aggregate type having EMPLOYER and OCCUPATION as coordinate types, then a COMPUTER SCIENTIST's identity would be determined completely by an ordered pair of EMPLOYER-OCCUPATION values. Changing OCCUPATIONs in this case would change the identity of the COMPUTER SCIENTIST.

There is a final point about attributes. Some semantic models draw a fine distinction between attributes and relationships. In these cases, attributes are considered relationships in which the values are atomic. In addition, some models allow relationships, but not attributes, to have attributes of their own, just like other objects. In the remainder of this paper, we use the two terms interchangeably.

(5) ISA relationships – Virtually all semantic models have the ability to represent ISA or supertype/subtype relationships. Generally speaking, an ISA relationship from a subtype to a supertype indicates that each object associated with the subtype is also associated with the supertype.

In our example schema, subtypes are represented by circles. The ISA edge from COMPUTER SCIENTIST to PERSON indicates that each COMPUTER SCIENTIST is a PERSON. In an

instance of the schema, the active domain of COMPUTER SCIENTIST would be contained in the active domain of PERSON. In most semantic models, subtypes *inherit* their supertype's attributes and can have attributes that are not shared by their supertype. Since semantic models usually allow undirected or weak cycles to exist among these relationships, they form what is often referred to as the schema's *ISA network*. For example, many PSYCHOLOGISTS in the Hypertext Community are also COMPUTER SCIENTISTS. Our schema would be more accurate if we were to introduce a new type, perhaps called RESEARCHER, having ISA relationships to both PSYCHOLOGIST and COMPUTER SCIENTIST. Doing so would create a weak cycle.

ISA relationships are used in semantic models for two closely related purposes. First, they can represent one or more overlapping subtypes of a type, as with the subtypes of PERSON in our example. Second, they can be used to form types that contain a union of disjoint types already present in the schema. Historically, semantic models have used a single kind of ISA relationship for both purposes. Recent research, however, has differentiated several kinds of ISA relationships, such as *subset* and *generalization*, that allow the subtle semantics of set containment and partitioning to be expressed.

(6) *Derived schema components* – Derived schema components, also called *derived data*, are a basic mechanism for data abstraction and encapsulation in many semantic models. Derivation allows information to be incorporated into a database schema that is itself computed from other information in the schema.

Derived schema components consist of a *structural specification* for holding the derived information and a *derivation rule* describing how the structure is to be filled. The structure, in most cases, is either a *derived subtype* or a *derived attribute*. For example, in our Hypertext Community schema, we could specify a derived subtype of PERSON called EXPERT with a derivation rule that any PERSON who USES more than one HYPERTEXT SYSTEM be included in EXPERT. Generally speaking, derived data are automatically updated as required by updates to other parts of the schema.

Objects, atomic and constructed types, attributes, ISA relationships, and derived schema components are the fundamental abstractions offered by the semantic modeling paradigm. It is important to note that much of the expressive power of semantic models comes from the fact that these constructs can usually be used recursively. From a broader perspective, though, the effectiveness of semantic

modeling is influenced by several other important factors. These include such things as combining restrictions, static integrity constraints, and the manipulation languages that enable users to interact with data.

(1) *Combining restrictions* – Most semantic models do not allow arbitrary combinations of the basic constructs. The most prominent of these restrictions affect the combining of ISA relationships. For example, directed cycles of ISA edges really make no sense and are usually not permitted. Generally speaking, combining restrictions assure that schemas are not redundant or ambiguous and capture the natural intuition about objects.

(2) *Static integrity constraints* – In general, semantic models express in a structural manner the most important types of relational integrity constraints. There are, however, a variety of relationships and properties of relationships that cannot be directly represented with the abstractions presented thus far. If objects are connected through relationships, then insertion, deletion, or modification of one object can still impact others. For this reason, many semantic models provide mechanisms for specifying integrity constraints, which in essence, assure that data associated with different parts of a schema are consistent according to some criteria. We will return to this issue in our discussion section.

(3) *Manipulation languages* – The data structuring capabilities discussed so far would normally be supported by a *data definition language* associated with a specific model. In addition, the model would be expected to have a corresponding *data manipulation language* that allows users to interact with the database. There are three fundamental capabilities that differentiate a semantic data manipulation language from a manipulation language for a traditional, record-oriented model: its ability to query abstract types, manipulate attributes, and manage derived data.

2.1.3 *Existing Semantic Models.* Since the mid-1970s, several semantic data models have been proposed. We will not survey them here, see instead Hull and King [27] and Peckham and Maryanski [37]. It is useful, however, to note some of the major philosophical dimensions along which existing models vary.

Models differ in the emphasis they place on the various constructs for interrelating object classes. For example, the Semantic Binary Data Model (SBDM) [2], Functional Data Model (FDM) [30, 44], and IRIS Model [12] stress the use of attributes to interrelate objects. In contrast, the Entity–Relationship (ER) Model [9], Semantic Database Model (SDM) [24], Semantic Association Model (SAM^{*}) [48], and the IFO Model [1] tend to emphasize type constructors.

Existing models also vary in the extent to which they try to accommodate all possible data modeling requirements of database applications. Those mentioned above attempt to be comprehensive. In contrast, the Structural Model [52], RM/T [11], and GEM [50, 57] support complex data as an extension of the relational model. A few models even address the issue of modeling the dynamic aspects of data. The Event Model [32] and SHM+ [6, 7] are two such examples.

The semantic models presented here are largely research vehicles. Only recently have fledgling commercial products begun to emerge. SIM, for example, is a commercial database system based on SDM [28]. An analysis of the commercial potential of semantic models is provided by Peckham and Maryanski [37].

Finally, we close this section by mentioning an important trend in a closely related area, object-oriented database design. Some development efforts on object-oriented systems have roots in semantic modeling in contrast to the majority which derive more from work on object-oriented languages [34]. As we will show later, these *structural object-oriented systems*, such as SOOM [4], DAMOKLES [15], and Cactis [25, 26] have important implications for hypermedia system designers.

2.2 The Saberel Semantic Network Database System

Saberel¹ is a prototypic semantic network database system under development by IBM Corporation. As we explain in the next section, Saberel forms the Storage Manager subsystem of the HB1 hyperbase management system presently under development in the Hypertext Research Laboratory. Saberel is representative of a family of semantic models [2, 5, 13, 21, 43], all of which represent data using two primary constructions: entity sets and binary relations. As shown in Fig. 2, schemas of these models generally consist of labeled nodes for entity sets and labeled arcs corresponding to binary relationships between them.

Saberel only supports atomic objects, called *subjects*. Subjects can carry a printable name and any arbitrarily sized string of data, referred to as a *user managed string* or *UMS*. Virtually all of

¹ Saberel is not a product.

Saberel's higher-level abstractions are derived from subjects. For example, entity sets, or *categories*, are themselves subjects as are relationship names. As a result, the semantic distinction between objects, types, and attributes is not directly enforced by Saberel's modeling primitives. Each binary relation is viewed as an inverse pair of single-valued functions.

Saberel's semantic model does not support type constructors, ISA relationships, or derived schema components and enforces few combining restrictions or integrity constraints. For example, the aggregation type ADDRESS is simulated in Fig. 2 as a category with three binary relationships. Saberel itself would not enforce the constraint that each STREET, CITY, ZIP tuple be unique. ISA relationships can be simulated as well. The philosophical approach of Saberel, and similar models, is to provide a small, universal set of constructs that can be used to build more powerful structures [19, 20]. These models tend to be "minimalist" in the sense that they require the database designer to understand fewer constructs [27].

The Saberel System runs on top of the Unix file system. Saberel subjects are stored in a *repository* and can be partitioned among any number of *areas*. Repositories and areas are implemented as Unix files, and, while they assist the user in organizing a Saberel database, they do not participate as abstractions in Saberel's underlying semantic model. Saberel has two programming interfaces: a low-level procedural interface to the library routines that implement a Saberel database and a declarative, high-level language that can be precompiled into C applications.

3. HB1: A CASE STUDY IN SEMANTIC DATA MODELING

We now turn our attention to a case study that demonstrates the use of semantic modeling to support hypermedia functionality. As we indicated earlier, we have been working on the design and implementation of a hyperbase management system called HB1. This HBMS is intended to satisfy the storage requirements of an extensible, object-based system architecture for distributed, inter-application linking. In this Section, we briefly describe our conceptual view of hypermedia and show how HB1 supports it. We also summarize HB1's overall organization and show how the system utilizes its underlying semantic database. The architectural setting for HB1 as well as its basic operations are also summarized. A more complete presentation of this material can be found in [41].

3.1 The Hypertext Research Lab Perspective

Fig. 3 summarizes our conceptual model of hypermedia. The model consists of six elements, the first three of which are: applications, components, and persistent selections. *Applications* are programs. *Components* are the data or information manipulated by applications. Text editor applications, for example, typically manipulate ASCII components. *Persistent selections* are selections within components that persist between application sessions and can be accessed at a later time. This functionality is implemented by applications maintaining persistent selection data structures appended to components (Fig. 4). Within our conceptual framework, hypermedia results from connections forged among persistent selections.

The remaining three components of our model are essential for hypermedia functionality. They are: anchors, links, and associations. As shown in Fig. 3, *anchors* are associated with persistent selections. *Links* join anchors, thereby completing connections among persistent selections. The relationships between these elements, depicted as arcs in Fig. 3, are *associations*. The distinguishing features of the model are: (1) anchors and links are processes in our model – behavioral entities capable of arbitrarily complex computations, (2) associations are structural entities, and (3) anchors, links, and associations all have first class status.

We clearly have a computational perspective on hypermedia. Anchors and links, for example, can be arbitrarily powerful agents. In addition, we make a strong distinction between structure and behavior. A general system architecture implemented along these lines allows the integration of diverse applications under a common hypermedia model. Specifically, non-monolithic, inter-application linking can be realized by moving hypermedia data and functionality into a distinct "link services" subsystem of a computing environment (Fig. 3). Services can then be provided to participating applications through interprocess communication (IPC).

Such an approach accommodates extensibility by allowing new functionality – i.e. new anchor and link processes – to be incorporated at any time without affecting the underlying structure of hypermedia. Massive distribution of functionality is accommodated since application, anchor, and link processes, as well as processes affecting associations, can conceptually reside anywhere within the IPC domain of the architecture. Simply put, the model provides a powerful framework for building the next generation of hypermedia systems.

3.2 HB1 Organization

As shown in Fig. 5, HB1 is built on top of the Unix file system and is composed of three distinct subsystems: The Object Manager (OM), Association Set Manager (ASM), and Storage Manager (SM). OM and ASM are both network-accessible server processes. OM is an object server. ASM manages structural data applicable to objects within OM's repository that are involved in hypermedia connections. Together, these two components implement HB1's data model. Physical storage is managed by SM which is currently a semantic network database system. OM and ASM bind to SM's library routines which, in turn, provide an interface to the underlying Unix file system. We begin by looking at HB1's Storage Manager.

3.2.1 Storage Manager. As we have indicated, HB1's Storage Manager is a prototypic semantic network database system under development by IBM called Saberel. Saberel's basic modeling constructs were presented earlier in the paper. The system was chosen because it allows unstructured objects to be stored, manipulated, and accessed by way of conveniently defined inter-object relationships. Traditionally, Saberel has been used for CAD and VLSI design applications, but we are extending and modifying this storage system to better support distribution, large object size, and hypermedia transactions.

3.2.2 Object Manager. HB1's Object Manager is a server process that provides persistent and sharable storage for applications. As shown in Fig. 5, OM is built on top of HB1's Storage Manager and provides an IPC interface to clients. OM, in effect, manages a Saberel repository of unstructured objects.

HB1 objects are simple. They are arbitrary size byte strings having unique identity and optional type and name attributes. HB1 objects have no class structure, methods, or inheritance. Each object in the database has a unique, 32-bit identifier (OID). These OIDs are sequentially allocated by the server upon request from clients creating objects. Applications use objects by copying them from the server into the virtual memory space of their own processes where they are manipulated locally and written back to the server.

3.2.3 *Association Set Manager.* The Association Set Manager is a server process that provides persistent and sharable storage for Associations and manages their run-time representation in support of a link services subsystem. To clarify what is meant by this, we must first explain ASM's data model. In [41], we present a formal definition for Association. Here, we provide an informal description along with an example. ASM manages the structural or connectivity data applicable to those objects within OM's repository that are involved in hypermedia associations. In a sense, it manages the static, backing store representation of hypermedia. ASM is built on top of HB1's Storage Manager and provides its own IPC interface to clients (Fig. 5).

In Fig. 6, we present an example of inter-application linking. Here, a hypermedia connection exists between persistent selections in two different components each of which is handled by a different application. The persistent selections, components, and applications all have unique IDs. AppIDs and CompIDs correspond to the OIDs for these objects on backing store. That is to say, these identifiers are generated and maintained by HB1's Object Manager, and the objects are stored in the Storage Manager's repository. Like AppIDs and CompIDs, AnchorIDs and LinkIDs correspond to the OIDs for these executable objects. PSIDs are generated and managed by applications, and the scope of their uniqueness extends only to the components in which they reside. The other IDs are unique across the world served by HB1.

To represent the structural information involved in such a connection, ASM's data model defines three collections of IDs: Bridges, Sides, and Associations (Fig. 6). A *Bridge* consists of a {PSID, CompID, AppID} triple and imparts global uniqueness to a PSID. It is a bridging collection of IDs in the sense that the triple spans the distinct address domains managed by HB1 and individual applications. An AnchorID associated with one or more Bridge triples forms a *Side*. Finally, the connection is completed by a LinkID being associated with two or more Sides. We call this an *Association*. A context is the *set of associations* available at a given moment, hence the server's name. Notice that our definition allows inter-application linking to occur at the level of anchors and at the level of links (Fig. 7).

3.3 Semantic Modeling of Associations

Fig. 8 shows how HB1's Object Manager and Association Set Manager data are represented in the Storage Manager. This is a complete Saberel instance schema for the simple linking example of Fig. 6. It is important to understand that the OM and ASM servers operate on the same Saberel repository. The shaded area in Fig. 8 corresponds to that part of the schema managed by OM. The subjects and relationships lying outside the shaded region are the part of the schema manipulated by ASM. This example shows quite graphically how information, structure, and behavior have been separated in HB1.

3.3.1 Semantic Modeling of OM Data. In the shaded area of Fig. 8, we see that three categories are explicitly represented in the database: OBJECT, NAME, and TYPE. The chunks of memory that constitute HB1 objects are members of the OBJECT category. Ovals in Fig. 8 represent Saberel subjects, and, in this instance schema, we show the binary relationships that exist between example subjects in each category.

3.3.2 Semantic Modeling of ASM Data. Outside the shaded area of Fig. 8, we see how Association Set Manager data are modeled by the Storage Manager. The OBJECT category in Fig. 8 contains applications, components, links and anchors as well as other objects handled by OM. In addition, ASM uses the categories PSID, BRIDGE, SIDE, and ASSOCIATION. ASM stores any PSIDs involved in connections in the PSID category. Subjects in the BRIDGE category tie together objects identified by {PsID, CompID, AppID} triples. Subjects in the SIDE category reference BRIDGES that have been grouped together by anchor attachment, and ASSOCIATION subjects join SIDES that have been grouped by link attachment. It is important to notice that the Storage Manager, by virtue of its underlying semantic database, directly represents associations among linked objects in our system. Or, said another way, the collection of IDs that define BRIDGE, SIDE, and ASSOCIATION entities need not be repeated in the database because these Saberel subjects have relationships that point directly to the HB1 objects involved. The connectivity information maintained by the Association Set Manager, in effect, overlays the object structure maintained by the Object Manager.

3.4 HB1 Operation

HB1 currently backends the SP1 hypermedia system prototype under development in the Hypertext Research Lab. SP1 allows hypermedia connections to be forged among applications that are able to participate in our distributed link services protocol. Its functional capabilities are realized by client/server relationships among Participating Applications, the Link Services Manager, and HB1's Object Manager and Association Set Manager servers (Fig. 5).

The Link Services Manager (LSM) is a server process that provides run-time support for inter-application linking. It coordinates the interprocess communication required to implement the hypermedia functionality of Participating Applications. Participation, in short, requires that applications be able to handle persistent selection and respond appropriately to messages from the LSM. A more complete description of the SP1 prototype and its Participating Applications can be found in [16, 41].

Although the details of SP1's overall behavior is beyond the scope of this paper, the basic notion is that hypermedia connections can be authored and browsed through coordinated events between Participating Applications and the Link Services Manager. As shown in Fig. 5, Participating Applications communicate with HB1's Object Manager and the LSM, and the LSM communicates with HB1's Object Manager and Association Set Manager. The messages that pass among the various processes essentially consist of type information and IDs.

At the present time, HB1's Object Manager performs basic back-end operations such as create, delete, store, and retrieve. Attribute operations have also been implemented. The Association Set Manager, in contrast, supports operations such as attach and detach anchor and link. In addition, there is a follow link operation that returns the identifiers of all objects that are reachable in the hyperbase from some input {PSID, CompID, AppID} triple. This query is intended to support traversal operations. OM and ASM do not currently support transaction management, concurrency control, ownership or access control, or assure database integrity in the event of failures. A complete description of HB1 operation can be found in [41].

4. RELATED WORK

There is currently widespread interest in developing formal reference models for hypermedia. Among the most prominent are those described in [3, 14, 17, 18, 23, 33, 42]. Not surprisingly, these models vary along many dimensions. Unlike the approach we are taking, most of the current models tend to focus on abstracting the connectivity of hypermedia from its underlying information rather than abstracting structure from functionality.

An important parallel interest is the development of effective database support for these models. Prominent hyperbase projects include work on the Hypertext Abstract Machine (HAM) [8, 14], the University of Aalborg's HyperBase [53, 54], and Schutt and Streitz' [42] HyperBase. Most current generation hypermedia systems implement backend functionality themselves without the support of a commercial DBMS [29]. Intermedia is a notable exception [46, 56]. Earlier versions of Intermedia were implemented on top of the INGRES relational DBMS [47]. Schutt and Streitz' [42] HyperBase is currently built on top of the Sybase relational DMBS.

The Dexter model [23] and that of Afrati and Koutras [3] have not yet been implemented. This is also true of Lange's [33] model, although he has shown how it could be implemented using object-oriented techniques. HAM [8, 14] and the University of Aalborg's HyperBase [53, 54] have been implemented without the assistance of an underlying DBMS. As far as we can tell, HB1 is unique in its use of a semantic network database system to manage physical storage.

5. DISCUSSION

In this section, we reflect on what we have learned so far in our use of semantic modeling. We begin with general comments, then specifically evaluate Saberel.

5.1 An Evaluation of Semantic Modeling

Many of the concepts offered by semantic modeling have important implications for hyperbase designers. Here, we examine some of semantic modeling's advantages and disadvantages by comparing semantic techniques to relational and object-oriented approaches.

5.1.1 *Semantic vs. Relational Approaches.* Clearly, the greatest single benefit semantic data modeling has for hypermedia applications is its capacity to express complex data interrelationships. Semantic models generally have several constructs that allow such relationships to be specified, whereas traditional, record-oriented models typically have only two or three [27]. Having few constructs that capture global structure complicates the modeling of hypermedia connectivity by making it necessary to derive inter-object relationships from other information in the database. It also has the effect of obscuring data dependencies.

For example, there are only two ways of representing relationships between objects in the relational model: within a relation and by using the same values in two or more relations. Fig. 9 shows a third normal form (3NF) relational schema [51] that corresponds to the semantic schema in Fig. 8. To capture the semantics of the original schema, key and inclusion dependencies are included. Key dependencies, indicated by the underlined attributes, state that the value of one or more fields of a relational tuple determines the remaining field values of the tuple. Inclusion dependencies state that all of the values occurring in one or more columns of one relation also occur in some columns of another relation. This representation requires one or more relations for each original object type. (A single relation could, in principle, be used to model associations. However, since set-valued attributes, such as BRIDGEID and SIDEID, are present, the relations would not be in 3NF.)

The important points for hypermedia are these. First, within the relational framework, the combined significance of relations and key and inclusion dependencies are difficult to comprehend. Generally speaking, semantic models capture in a structural manner these two important types of relational integrity constraints. Key dependencies can be represented using single-valued attributes, and inclusion dependencies that serve as referential constraints are modeled implicitly by the relationships that connect objects in a semantic schema. In short, the flat nature of relations makes it difficult to model complex object networks [46].

The second point of interest for hypermedia concerns the inter-object relationships that must be computed within the relational framework. For example, a hyperbase management system that stores its link information in a relational database as shown in Fig. 9 has no direct representation of hypermedia connectivity. It will likely use its underlying database almost exclusively for selection, then use join operations performed within its server's program memory to build up the inter-object

relationships required for browsing, etc. When Intermedia was running on top of INGRES, it built its document's block and link hierarchy in this way [46]. A session thus begins by causing part of the disk to be copied to another and ends by converting internal data structures into relational tuples which are then written back to disk. Not only does this impart a batch-processing flavor to hypermedia interactions, it requires many database exchanges and complicates distribution and transaction management [35, 46].

The capacity of a semantic database to directly represent connectivity off-loads from an HBMS's servers much of the work involved in representing link information. Said another way, information of a structural nature is manipulated with greater ease [37]. In fact, recent trends in the design of object-oriented databases suggest that this type data can remain entirely under database control and yield better overall performance than memory-resident data structures constructed using relational methods [34].

From another perspective, notice that the access paths to physical data that are made available through the relational model mimic the logical structure of the database schemas. In contrast, the attributes of semantic models may be used as direct conceptual pointers. Attributes are applied directly to objects to move through the schema to other objects. This results in an increased separation of logical and physical components, and, more important for hypermedia, encourages a navigational view of data.

In contrast to traditional approaches, semantic models tend to permit information to be viewed at many different levels of abstraction. For example, schemas can be considered at various levels of detail. It is easy to isolate information of a given type within a semantic schema, and it is often easy to follow semantic connections to closely related object types. Systems that support derived schema components often allow arbitrary "user views" of the data to be computed. Overall, the strong emphasis that semantic models place on abstraction enhances their capacity for modeling "real-world" situations, since viewing data on many levels is consistent with the way people view the world [27, 37].

In summary, the relational model does not provide sufficiently rich data structuring capabilities for

problems that do not readily map into tables. Hypermedia, with its emphasis on arbitrarily complex connectivity, appears to have much to gain from the use of semantic methods.

5.1.2 *Semantic vs. Object-Oriented Approaches.* Since they are so closely related, it is appropriate to clarify the distinction between the semantic and object-oriented database paradigms. Semantic databases attempt to provide *structural abstractions* [27, 31] whereas object-oriented databases provide *behavioral abstractions* [36]. Semantic models, as we have seen, provide constructors for creating complex types, while behavioral issues are often left undefined. Object-oriented models take an abstract data type approach of embedding operations within types. An object, therefore, is in control of its own behavior, behavior that is invoked by "messages" that are sent to the object.

One of the drawbacks we see in using object-oriented techniques for hypermedia is that encapsulation can complicate certain types of structural operations. For example, the addressing information involved in a hypermedia connection would typically be stored in the instance variables of a "link" object. Access to these instance variables is obtained only through the methods that have been encapsulated within these objects or the link object class. Since there is no externally accessible representation of global structure, operations such as structural queries can be difficult to implement. Our impression is that current hypermedia data models coupled with object-oriented database techniques tend to encourage implementation approaches that encapsulate structural information with structure-independent functionality. Doing so hampers the implementation of crucial next-generation system features such as structure search.

The distinction between object-oriented and semantic models is often not well defined. Some researchers refer to semantic models as being "object-oriented" in order to stress that they provide mechanisms for structuring complex types. Likewise, some experimental object-oriented database systems attempt to provide enhanced support for structural abstractions [31]. In fact, the synergy between semantic and object-oriented paradigms is becoming apparent to many people. Together they provide ways of expressing the structural *and* behavioral aspects of complex objects.

Which brings us to the only major drawback we see to the use of semantic modeling for hypermedia. While we want to avoid embedding structure in behavioral entities, we would, in fact, like objects in our architecture to exist in a rich class hierarchy and be capable of complex functionality and interactions. Semantic techniques alone do not provide this. The ideal approach appears to be a *structural object-oriented* methodology which integrates behavioral and structural techniques [4, 15, 25, 26, 39]. In a sense, we have already begun to move down this path in that HB1's Association Set Manager is a behavioral layer that has been applied to the hyperbase's underlying structure.

5.2 An Evaluation of Saberel

Simple "binary" semantic models, such as Saberel, attempt to make schema development easier by reducing the number of modeling constructs a user must learn [27]. Our impression is that a more complex semantic model would have been helpful in several cases. For example, Saberel lacks constructed types. BRIDGES in Fig. 8 would clearly be modeled more accurately as an aggregation type. In fact, only through aggregation does one capture the sense that a BRIDGE is uniquely identified by its coordinate {PsID, CompID, AppID} triple. Likewise, it would be more accurate to model SIDES as a grouping type comprised of sets of BRIDGES, and ASSOCIATIONS should really be a grouping of SIDES. Besides giving the semantics of object identity greater precision, constructed types would have simplified the language mechanisms required to manipulate the database. Garg [18] has demonstrated other ways that constructed types are useful with respect to hypermedia.

Several operations, such as the Association Set Manager's attach and detach operations, would have been easier to implement if Saberel had the capacity to enforce integrity constraints at the database level. Since it does not, it was necessary to explicitly include these constraints in the procedures that implemented the operations. Although we have not at this point worked with Saberel enough to know, we suspect that consistency constraints will become even more important when the questions of versioning and composition are addressed.

6. FUTURE WORK

Future work will be directed toward addressing HB1's current shortcomings. In particular, we are interested in examining the issue of versioning within a semantic database and the implementation of structure-related operations. Several of these operations – such as computing virtual structures, constructing graphical overviews, and performing structural searches – are recognized to be crucial next-generation features [22]. One of the benefits of representing hypermedia structure the way we have is the way it facilitates the implementation of these types of operations. HB1 is also being extended to include transaction management and controlled sharing, and we are enhancing its distributed performance through intelligent caching protocols.

7. SUMMARY

Advanced hypermedia environments pose several unique data management problems. Most of these ultimately derive from the fact that hypermedia is a complex amalgam of information, structure, and behavior. In this paper, we have described the basic concepts underlying semantic data modeling. We have also shown how our conceptual model of hypermedia has been instantiated within a hyperbase management system that uses a semantic database system to manage physical storage. Among the most important of our conclusions so far are:

- semantic data models are particularly effective in capturing the complex inter-object relationships that characterize hypermedia;
- semantic techniques allow structure, information, and behavior to be abstracted from hypermedia;
- the implementation of structure-related operations, such as structural queries, is facilitated by a semantic representation;
- hyperbase management systems have much to gain from the use of integrated semantic object-oriented techniques.

ACKNOWLEDGMENTS

The authors would like to thank the team that constructed the software described in this paper: Peter Nuernberg, Doug Miller, Joe Drufke, Don Dylla, and Tim Roden. Thanks also to Ed Cunnius for help with figures and to Cindy Kunz for help preparing the manuscript. Bob Griffith, Jeff Barber, and Ken Briskey provided valuable assistance in our work with Saberel.

REFERENCES

1. Abiteboul, S., and Hull, R. IFO: A formal semantic database model. *ACM Trans. Database Syst.* 12, 4 (Dec. 1987), 525–565.
2. Abrial, J. R. Data semantics. In *Data Base Management*, J. W. Klimbie and K. L. Koffemen, (Eds.), North-Holland, Amsterdam, (1974), pp. 1–59.

3. Afrati, F., and Koutras, C. A hypertext model supporting query mechanisms. In *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext*, (France), A. Rizk, N. Streitz, and J. Andre, (Eds.), Cambridge University Press, Cambridge, U.K., (Nov. 1990), pp. 52–66.
4. Berre, A. J. SOOM and Tornado^{*}: Experience with database support for object-oriented applications. In *Advances in Object-Oriented Database Systems: Proceedings of the 2nd International Conference on Object-Oriented Database Systems*, K. Dittrich, (Ed.), Springer-Verlag, New York, (Sept. 1988), pp. 104-109.
5. Bracchi, G., Paolini, P., and Pelagatti, G. Binary logical associations in data modelling. In *Modelling in Data Base Management Systems*, North Holland, Amsterdam, (1976), pp. 125-148.
6. Brodie, M. L. On the development of data models. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, (Eds.), Springer-Verlag, New York, (1984), pp. 19–48.
7. Brodie, M. L., and Ridjanovic, D. On the design and specification of database transactions. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*, M. L. Brodie, J. Mylopoulos, and J. W. Schmidt, (Eds.), Springer-Verlag, New York, (1984), pp. 277–320.
8. Campbell, B., and Goodman, J. HAM: A general-purpose hypertext abstract machine. *Commun. ACM* 31, 7 (July 1988), 856-861.
9. Chen, P. P. The entity-relationship model – Toward a unified view of data. *ACM Trans. Database Syst.* 1, 1 (March 1976), 9–36.
10. Christodoulakis, S., Ho, F., and Theodoridou, M. The multimedia object presentation manager of MINOS: A symmetric approach. In *Proceedings of the ACM International Conference on the Management of Data (SIGMOD)*, (Washington, DC), ACM, New York, (1986), pp. 295–310.
11. Codd, E. F. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.* 4, 4 (Dec. 1979), 397–434.
12. Darrett, N., Kent, W., and Lyngbaek, P. Some aspects of operations in an object-oriented database. *IEEE Database Eng. Bull.* 8, 4 (Dec. 1985).

13. Deheneffee, C., Hennebert, H., and Paulus, W. Relational model for a database. In *Proceedings of the IFIP Congress*, (1974), pp. 1022–1025.
14. Delisle, N., and Schwartz, M. Neptune: A hypertext system for CAD applications. In *Proceedings of the ACM International Conference on the Management of Data (SIGMOD)*, (1986), pp. 132–143.
15. Dittrich, K. R., Gotthard, W., and Lockemann, P. C. DAMOKLES – A database system for software engineering environments. In *Proceedings of the IFIP Workshop on Advanced Programming Environments*, (Trondheim, Norway), R. Conradi, R. M. Didriksen, and D. H. Wanvik, (Eds.), Springer-Verlag, New York, (June 1986), pp. 353-371.
16. Drufke, J. E., Leggett, J. J., Hicks, D. L., Schnase, J. L. 1991. The derivation of a hypertext widget class from the Athena text widget. Department of Computer Science Technical Report No. TAMU 91-002, Texas A&M University, College Station, TX, (1991).
17. Furuta, R., and Stotts, D. The Trellis hypertext reference model. In *Proceedings of the NIST Hypertext Standardization Workshop*, NIST, Gaithersburg, MD, (1990), pp. 83–93.
18. Garg, P. Abstraction mechanisms in hypertext. In *Commun. ACM* 31, 7 (July 1988), 862–870.
19. Griffith, R. L. 1982. Three principles of representation for semantic networks. *ACM Trans. Database Syst.* 7, 3 (Sept. 1982), 417-442.
20. Griffith, R. L. 1989. Semantic-network databases for VLSI design. *Proceedings of the Seventh British National Conference on Databases (BNCOD 7)*, (Heriot-Watt University), M. H. Williams, (Ed.), Cambridge University Press, Cambridge, U.K., (July 1989), pp. 5-36.
21. Hainaut, J. L., and Lecharlier, B. An extensible semantic model of database and its data language. In *Proceedings of the IFIP Congress*, (1974), pp. 1026–1030.
22. Halasz, F. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Commun. ACM* 31, 7 (July 1988), 836–852.
23. Halasz, F., and Schwartz, M. The Dexter hypertext reference model. In *Proceedings of the NIST Hypertext Standardization Workshop*, NIST, Gaithersburg, MD, (1990), pp. 95–133.

24. Hammer, M., and McLeod, D. Database description with SDM: A semantic database model. *ACM Trans. Database Syst.* 6, 3 (Sept. 1981), 351–386.
25. Hudson, S. E., and King, R. The Cactis project: Database support for software environments. *IEEE Trans. Softw. Eng.* 14, 6 (June 1988), 709–719.
26. Hudson, S. E., and King, R. Cactis: A self-adaptive, concurrent implementation of an object-oriented database management system. *ACM Trans. Database Syst.* 14, 3 (Sept. 1989), 291-321.
27. Hull, R., and King, R. Semantic database modeling: Survey, applications, and research issues. *ACM Comput. Surv.* 19, 3 (Sept. 1987), 201–260.
28. Jagannathan, D., Fritchman, B. L., Gluck, R. L., Thompson, J. P., and Tolbert, D. M. SIM: A database system based on the semantic data model. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, (Chicago), ACM, New York, (June 1988), pp. 46-55.
29. Kacmar, C. J., Leggett, J., Schnase, J. L., and Boyle, C. Data management facilities of existing hypertext systems. Department of Computer Science Technical Report No. TAMU 88-018, Texas A&M University, College Station, TX, (1988).
30. Kerschberg, L. and Pacheco, J. E. S. A functional database model. Technical Report, Pontificia Univ. Catolica do Rio de Janeiro, Rio de Janeiro, Brazil, (1976).
31. King, R. My cat is object-oriented. In *Object-Oriented Concepts, Databases, and Applications*, W. Kim and F. H. Lochovsky, (Eds.), Addison-Wesley Publishing Co., Reading, MA, (1989), pp. 23–30.
32. King, R., and McLeod, D. The event database specification model. In *Proceedings of the 2nd International Conference on Databases: Improving Usability and Responsiveness* Jerusalem, Israel, IIPA, (1986), pp. 299–322.
33. Lange, D. A formal model for hypertext. In *Proceedings of the NIST Hypertext Standardization Workshop*, NIST, Gaithersburg, MD, (1990), pp. 145–166.
34. Maier, D. Why isn't there an object-oriented model? Computer Science and Engineering Technical Report No. CS/E-89-002, Oregon Graduate Center, Beaverton, Oregon, (1989).

35. Maier, D. Making database systems fast enough for CAD applications. In *Object-Oriented Concepts, Databases, and Applications*, W. Kim and F. H. Lochovsky, (Eds.), Addison-Wesley Publishing Co., Reading, MA, (1989), pp. 573–582.
36. Maier, D., Stein, J, Otis, A. and Purdy, A. Development of an object-oriented DBMS. In *Proceedings of the First ACM Conference on Object-Oriented Programming Systems, Languages and Applications, OOPSLA '86*, ACM, New York, (1986), pp. 472–482.
37. Peckham, J., and Maryanski, F. Semantic data models. *ACM Comput. Surv.* 20, 3 (Sept. 1988), 153–189.
38. Potter, W. D., and Trueblood, R. P. Traditional, semantic, and hyper-semantic approaches to data modeling. *Computer* 21, 6 (June 1988), pp. 53–63.
39. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorenzen, W. *Object-oriented Modeling and System Design*. Prentice Hall, Englewood Cliffs, (1990).
40. Schmid, H. A., and Swenson, J. R. On the semantics of the relational data model. In *Proceedings of the ACM International Conference on the Management of Data (SIGMOD)*, (San Jose, CA), ACM, New York, (1975), pp. 211–223.
41. Schnase, J. L., Leggett, J. J., and Hicks, D. L. HB1: Initial design and implementation of a hyperbase management system. Department of Computer Science Technical Report No. TAMU-HRL 91-003, Texas A&M University, College Station, TX, (1991).
42. Schutt, H. A., and Streit, N. A. Hyperbase: A hypermedia engine based on a relational database management system. In *Hypertext: Concepts, Systems, and Applications. Proceedings of the European Conference on Hypertext*, (France), A. Rizk, N. Streit, and J. Andre, (Eds.), Cambridge University Press, Cambridge, U.K., (Nov. 1990), pp. 95–108.
43. Senko, M. E. Information systems: Records, relations, set, entities, and things. *Inf. Syst.* 1, 1 (1975), 3–13.
44. Shipman, D. The functional data model and the data language DAPLEX. *ACM Trans. Database Syst.* 6, 1 (March 1981), 140–173.
45. Smith, J. M., and Smith, D. C. P. Database abstractions: Aggregation and generalization. *ACM Trans. Database Syst.* 2, 2 (March 1977), 105–133.

46. Smith, K., and Zdonik, S. Intermedia: A case study of the differences between relational and object-oriented database systems. In *Proceedings of the ACM OOPSLA '87 Conference*, ACM, New York, (1987), pp. 452–465.
47. Stonebraker, M., Wong, E., Kreps, P. and Held. G. The design and implementation of INGRES. *ACM Trans. Database Syst.* 1, 3 (Sept. 1976), 189-222.
48. Su, S. Y. W. SAM^{*}: A semantic association model for corporate and scientific statistical databases. *Inf. Sci.* 29 (1983), 151–199.
49. Tanenbaum, A. S., van Renesse, R., van Staveren, H., Sharp, G. J., Mullender, S. J., Jansen, J., and van Rossum, G. Experiences with the AMOEBA distributed operating system. *Commun. ACM* 33, 12 (Dec. 1990), 46–63.
50. Tsur, S., and Zaniolo, C. An implementation of GEM – Supporting a semantic data model on a relational back-end. In *Proceedings of the ACM International Conference on the Management of Data (SIGMOD)*, ACM, New York, (1984), pp. 286–295.
51. Ullman, J. D. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, Rockville, MD, (1988).
52. Wiederhold, G., and El-Masri, R. Structural model for database design. In *Entity-Relationship Approach to System Analysis and Design*. North Holland, Amsterdam, (1980).
53. Wiil, U. K. Design and implementation of a hyperbase. Department of Mathematics and Computer, Institute for Electronic Systems Technical Report No. IR 90-03, The University of Aalborg, Aalborg, Denmark, (1990).
54. Wiil, U. K., and Osterbye, K. Experiences with HyperBase – A multi-user back-end for hypertext applications with emphasis on collaboration support. Department of Mathematics and Computer, Institute for Electronic Systems Technical Report No. IR 90-38, The University of Aalborg, Aalborg, Denmark, (1990).
55. Woelk, D., Kim, W., and Luther, W. An object-oriented approach to multimedia databases. In *Proceedings of the ACM International Conference on the Management of Data (SIGMOD)*, (Washington, DC), ACM, New York, (1986), pp. 311–325.
56. Yankelovich, N., Haan, B., Meyrowitz, N., and Drucker, S. Intermedia: The concept and the construction of a seamless information environment. *Computer* 21, 1 (Jan. 1988), 81–96.

57. Zaniolo, C. The database language GEM. In *Proceedings of the ACM International Conference on the Management of Data (SIGMOD)*, ACM, New York, (1983), pp. 207–217.

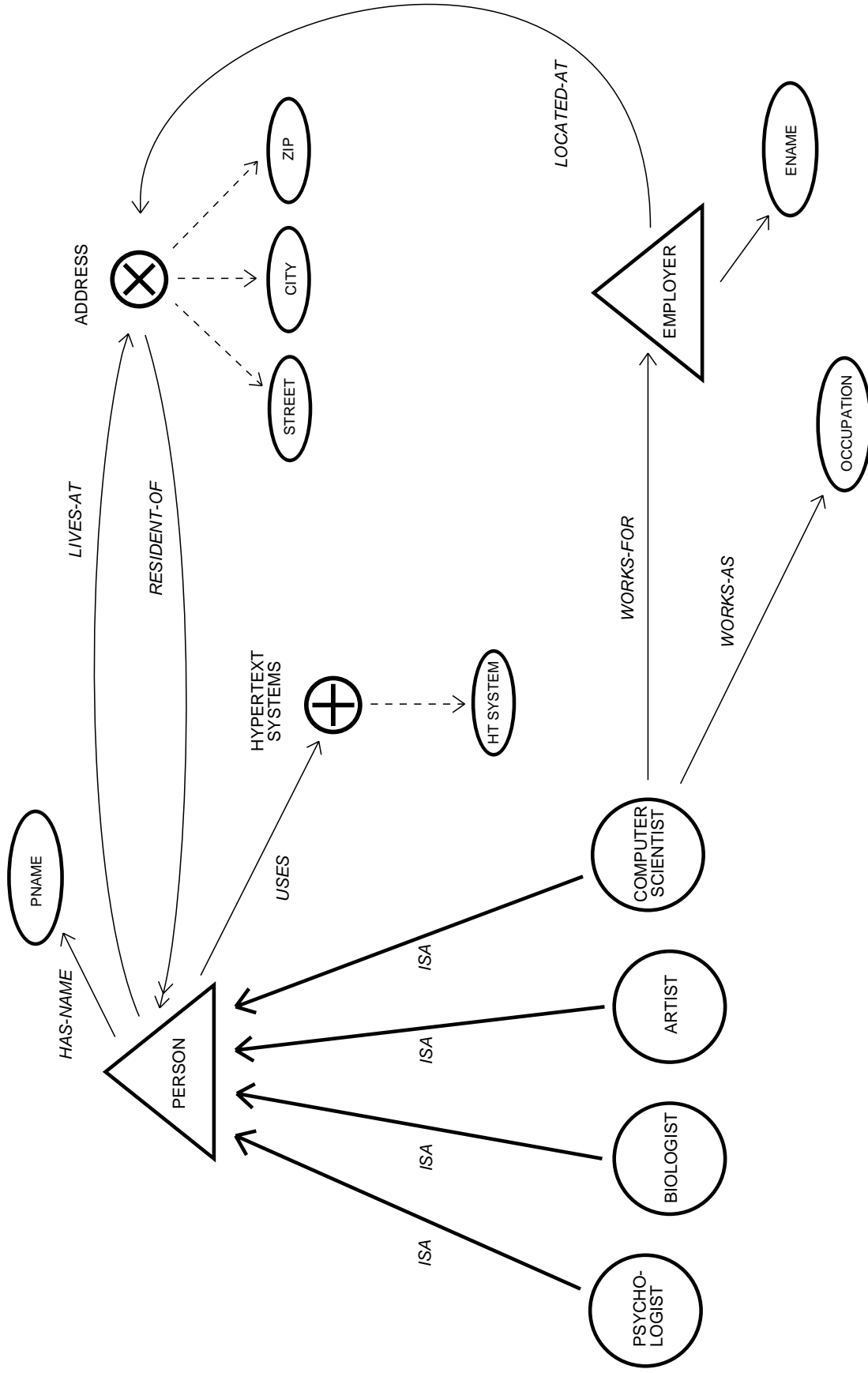


Figure 1. The Hypertext Community Database schema.

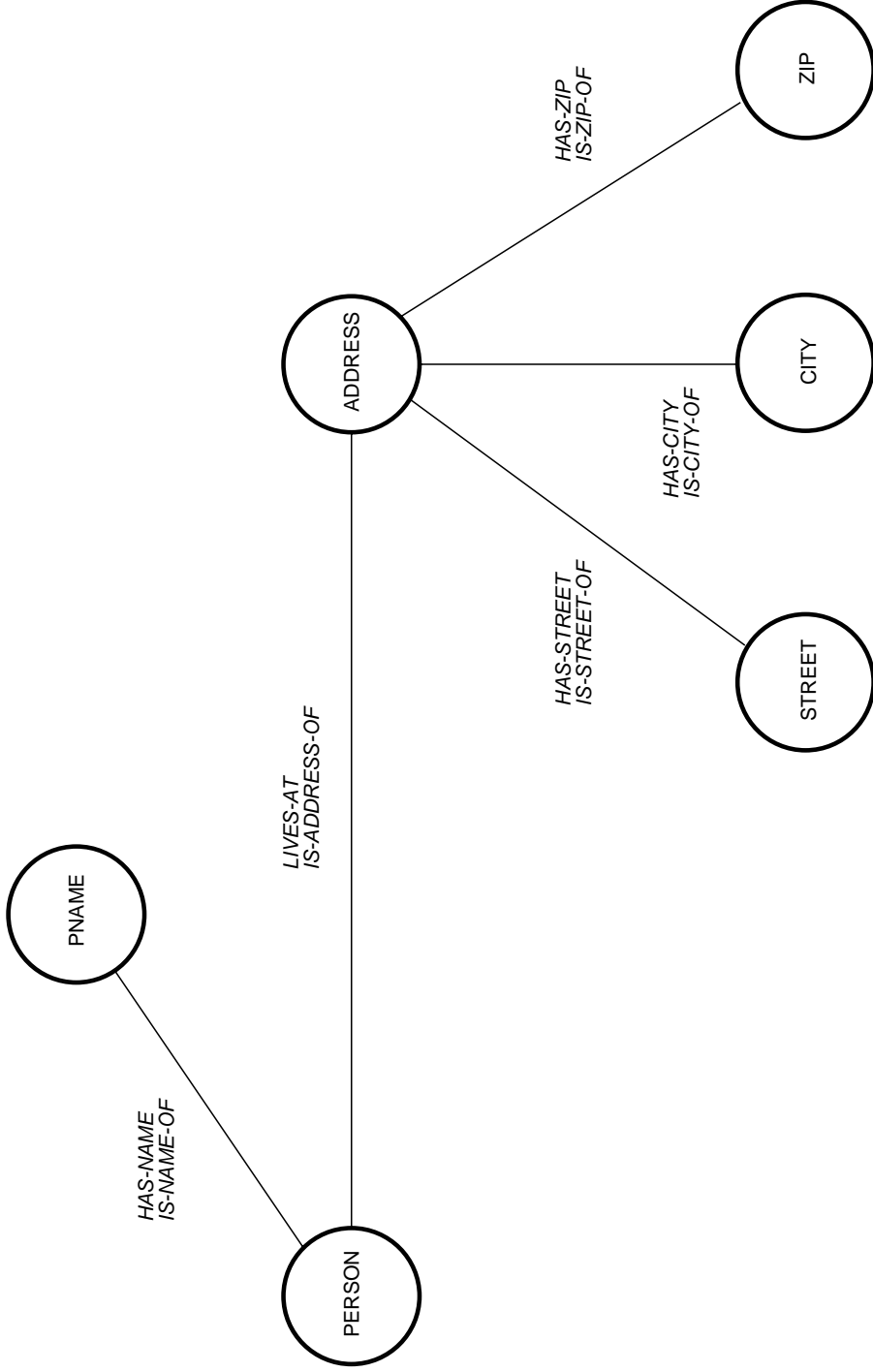


Figure 2. Saberel representation of part of the Hypertext Community schema.

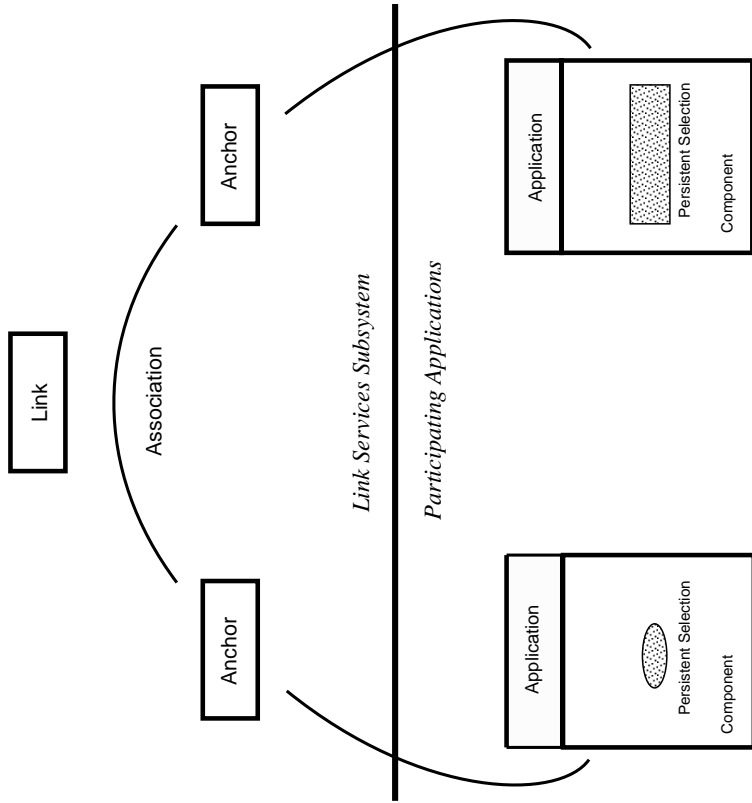


Figure 3. Conceptual model of hypermedia underlying HB1.

PS ID	PS Value	Anchor Bit	Link Bit
00000000	0021:0003	0	0
00000001	0000:0013	1	1
00000002	014A:0013	1	1
00000003	RasterSelect(fig1,x,y,001B)	1	1
00000004	12FA:008C	1	0
•••	•••	•••	•••
n	0000:0000	0	0

Parental investment may be defined as "any investment by the parent of an individual offspring that increases the offspring's chance of surviving at the cost of the parent's ability to invest in other offspring" (Trivers, 1972). Although it is difficult to distill a single metric to represent the time, energy, and risk associated with **parental investment**, a rough estimate can be obtained by considering the combined qualitative and energetic contributions of parents to the reproductive effort (Biedeweg, 1983).

In Cassin's Sparrow, territorial defense is the responsibility of the male and primarily involves the allocation of energy resources with little associated risk. Egg production, requiring energy, and incubation, requiring time, are done entirely by the female. Feeding of offspring demands time and energy and appears to be skewed toward the female during both the nestling and fledging care phases.

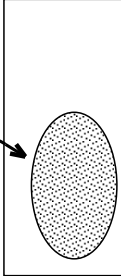


Figure 1. Cassin's Sparrow fledging foraging in a Honey mesquite tree (*Prosopis glandulosa*).

Figure 4. Component and persistent selection data structure. In this case, we see a Persistent Selection Table.

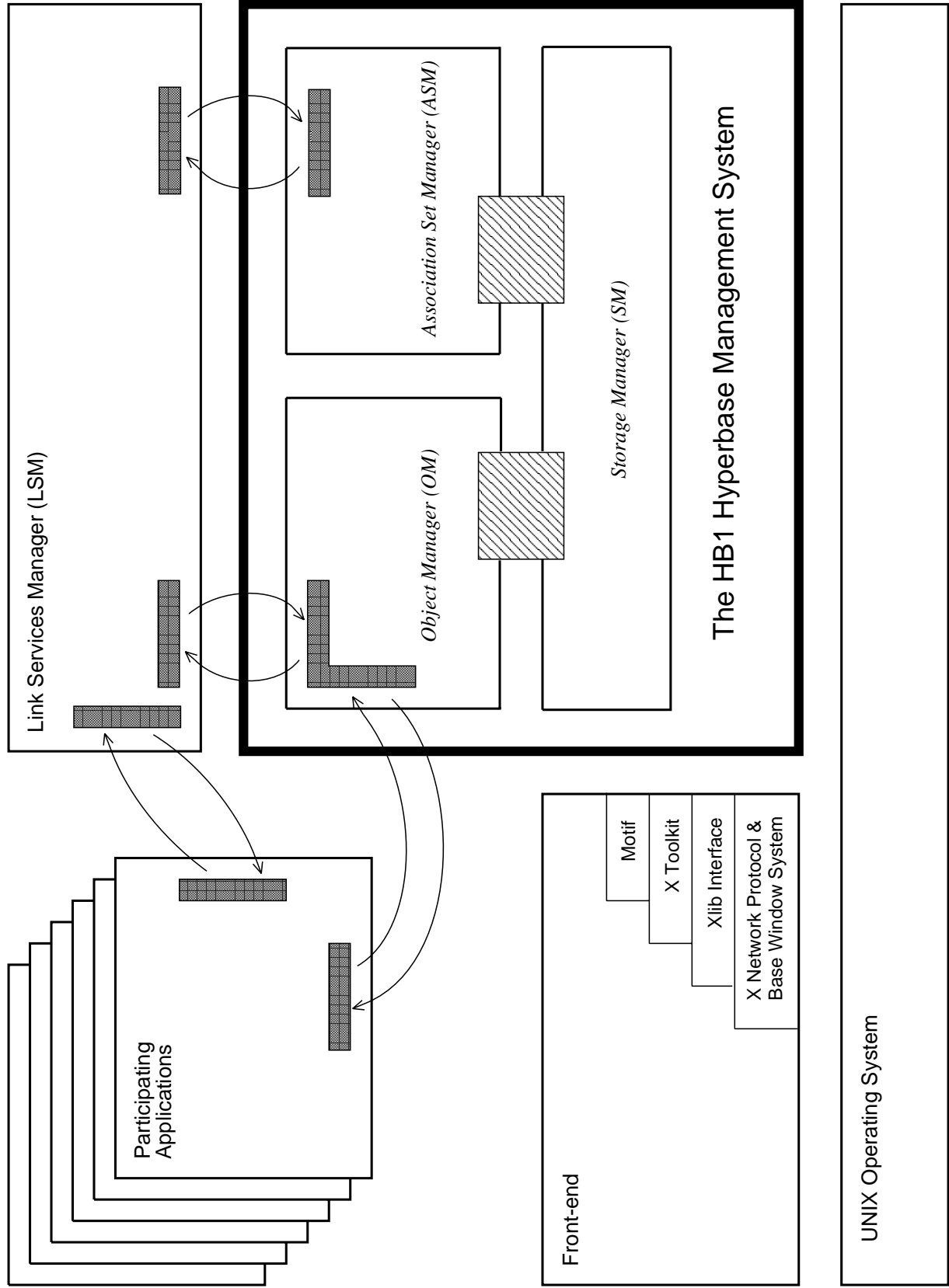


Figure 5. System architecture for the SP1 hypermedia system prototype.

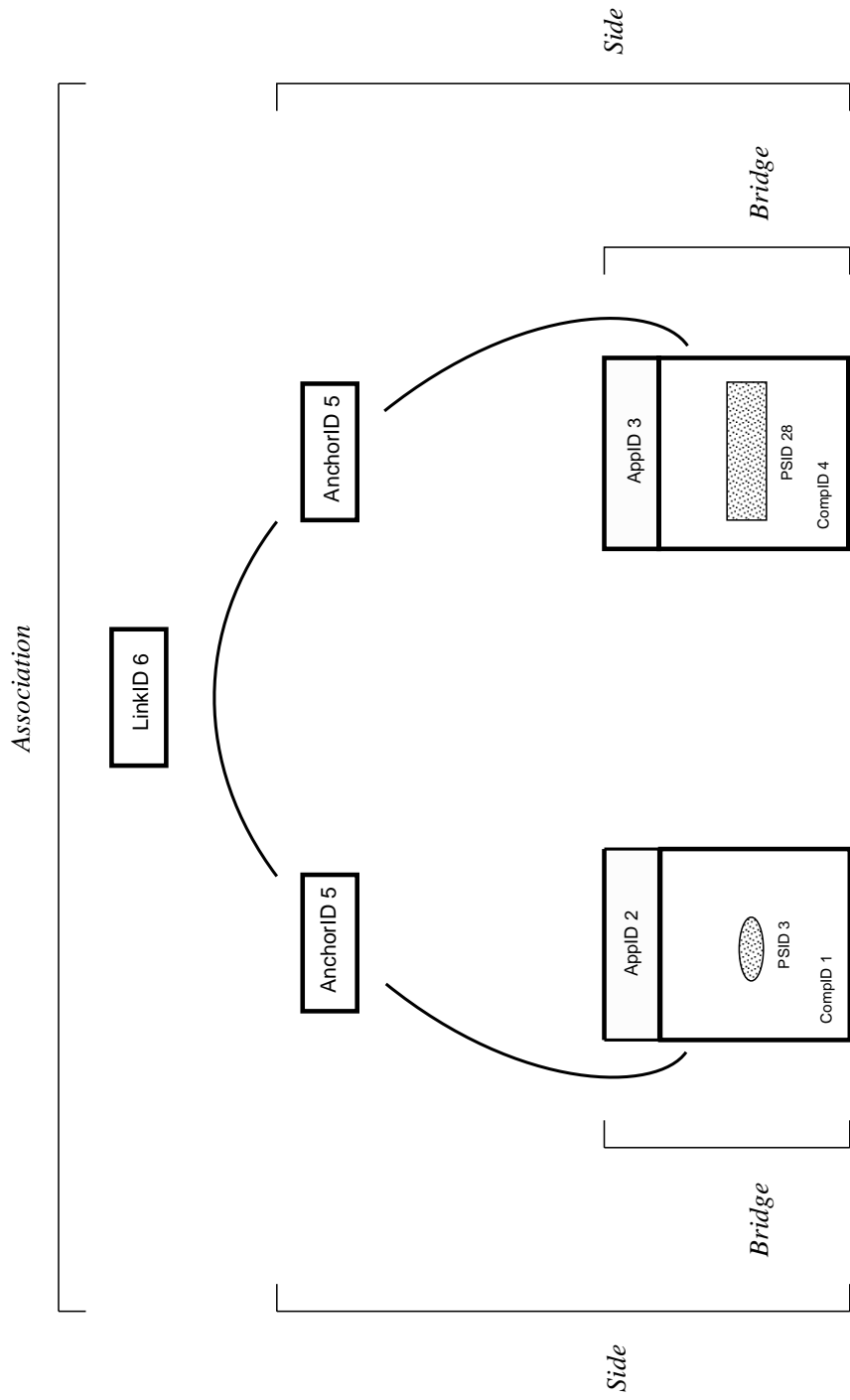


Figure 6. Simple example of inter-application linking showing ASM's 3 major abstractions.

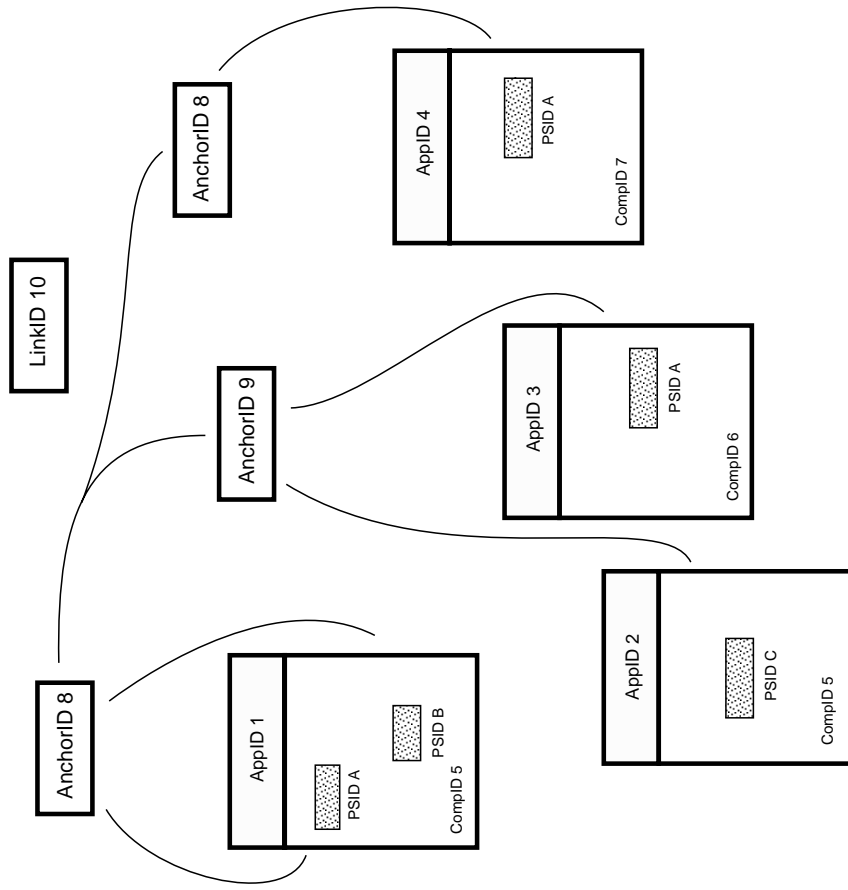


Figure 7. Complex example of inter-application linking.

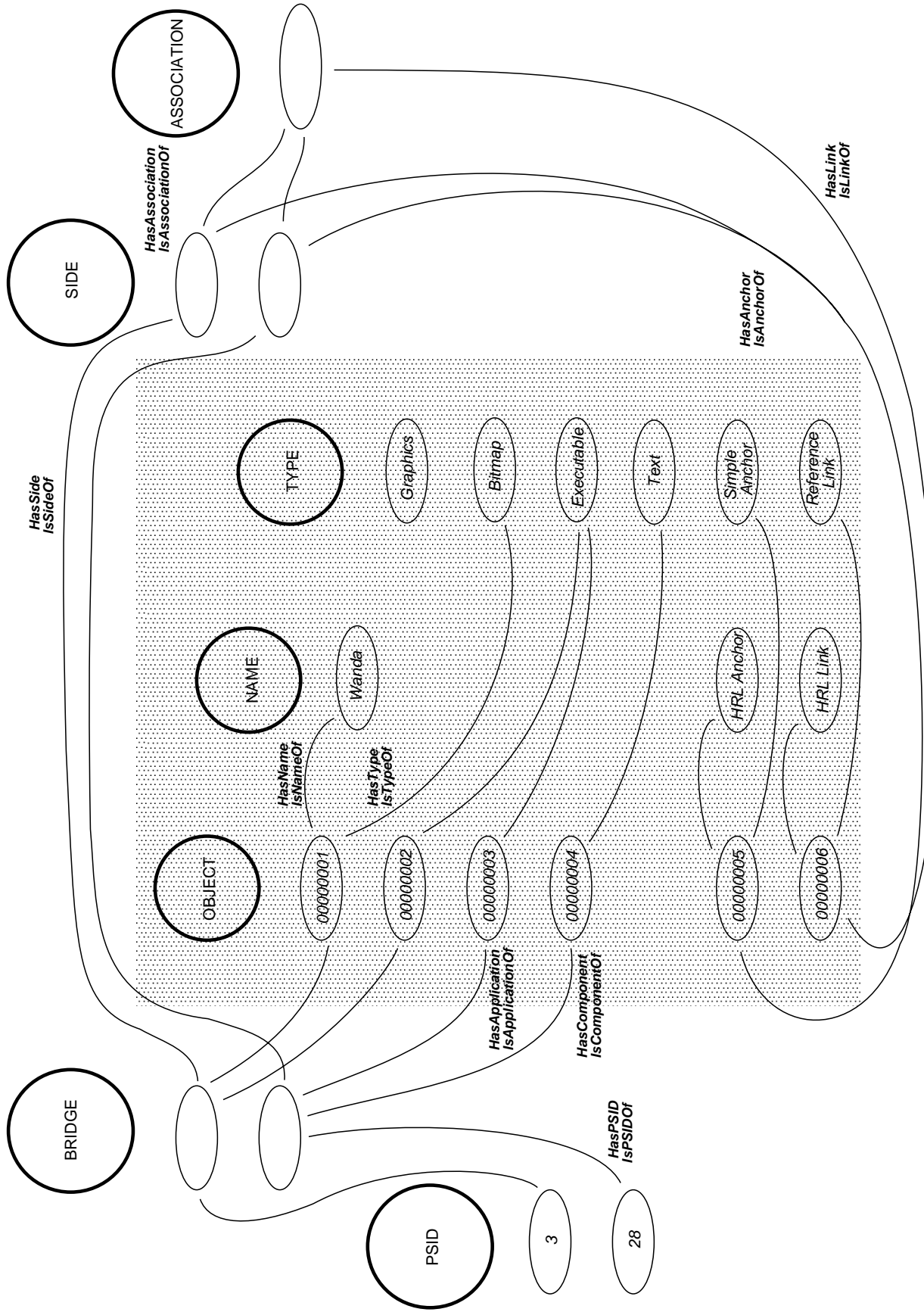


Figure 8. An instance schema showing how Object Manager and Association Set Manager data are modeled by HBI's Storage Manager. The schema corresponds to the inter-application linking example in Fig. 6.

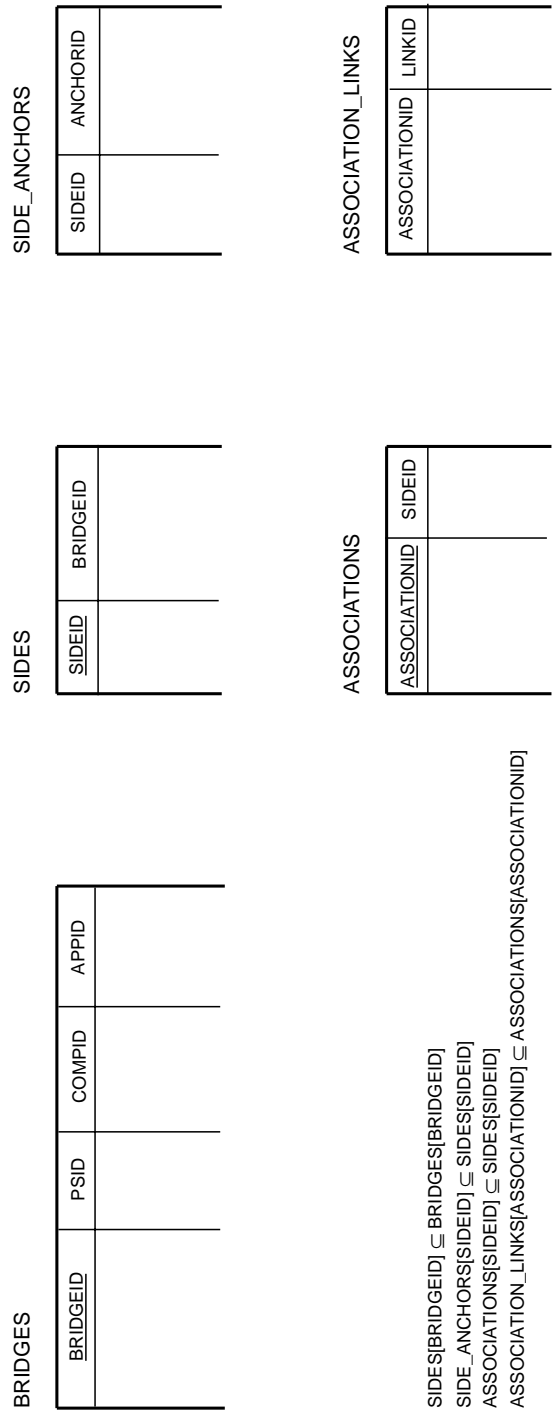


Figure 9. 3NF relational schema corresponding to the semantic schema in Fig. 8.