THE ASSOCIATIVE DATA MODEL*

Ronald S. King and Stephen B. Rainwater
Department of Computer Science
The University of Texas at Tyler
College of Engineering and Computer Science
3900 University Blvd.
Tyler, Texas 75799
Phone#: (903)-566-7403
rking@mail.uttyl.edu
srainwat@mail.uttyl.edu

1 INTRODUCTION

The associative data model, ADM, is a new data model implemented in Lazy Software Inc.'s new associative data modeling system trademarked as *Sentences* [8]. *Sentences* is a multi-user, web-enabled database modeling system, useable for internet applications, especially application service providers (ASPs). This model is an excellent topic to include on web-based components from the introductory to advanced database classes as described by [7]. An overview of the ADM and *Sentences* is provided for in this paper.

2 BACKGROUND

Currently most database applications are implemented in either the relational, object, or object-relational data models. Even though the relational model is well suited to transaction processing, it cannot manage complex data structures typical of multimedia applications. Also, the relational model does not easily support the distribution of one database across a number of servers, which is the natural model for the internet. Object-oriented databases store small numbers of large objects more efficiently than large numbers of small ones. The latter feature of the object-oriented data model is counter to the Internet needs of modern database processing where lightweight technologies have components that can be distributed across the Internet itself and embedded in browsers and web sites. Additionally, object databases

---

currently lack essential features such as SQL support, authorization and access control, performance tuning, and interfaces to transaction monitors and other databases. The object-relational model however abandons SQL which is felt by many professionals in the field to be the relational data model.

The associative data model has the “look-and-feel” of Object Role Modeling, ORM, developed in Europe in the mid-1970s [1,5]. Falkenberg [6] proposed the fundamental ORM framework which he called the “object-role model”. This framework allowed n-ary and nested relationships for modeling information systems. The framework was eventually modified to a method renamed "Natural language Information Analysis Method" (NIAM) [4]. An overview of ORM noting its advantages over entity relationship and traditional object oriented modeling is discussed in [2]. ORM is normally implemented as a set of relational tables, clearly a level of indirection. An in-depth coverage of ORM is presented in the new text by Halpin [3].e

The Associative Data Model developed by Simon Williams simply implements things and associations between them via R-trees. Thus the Associative Data Model has been implemented in an associative data modeling system directly by Lazy Software, Inc. *Sentences* is a multi-user, web-enabled DBMS written in Java complete with a full set of development tools, interfaces and applications. The first general release of the product, 1.1 *Sentences*, was in October 2000; it runs under SUN Solaris, Windows NT or Linux. The ancillary tools that provide the user interface, schema specification facilities and query support can make use of Microsoft, Apache, WebSphere and Tomcat Web servers, with Windows clients running either Microsoft or Netscape browsers. Release 2.0 includes XML support and allows for stored procedures and triggers.

3 COMPARISON OF THE RELATIONAL AND ASSOCIATIVE DATA MODELS

The relational data model has many disadvantages which can be overcome by the associative data model. For instance, when employing the relational data model, each time a new application is constructed, the applications programmer would have to construct or modify a new set of tables. The latter practice is both time consuming and complex. Omnicompetent programming, a feature of the associative data model, allows for a single set of programs to implement many different applications. Also, companies or businesses often want to record information that is relevant only to a particular customer. In the relational data model, the programmer or database designer would accomplish the latter task via text or through null values in the relevant table column. Neither of these two solutions is truly satisfactory. Through the instance schema feature, the *Sentences* DBMS enables schema and rules to apply to a single instance in the database with no overhead. Additionally many company databases are reflecting mergers taking place worldwide. In *Sentences*, distinct databases can be viewed together or amalgamated at any time without special tools for data correlation and analysis through the Schema aggregation feature. *Sentences* also allows for feature programming whereby features can be made visible or invisible to individual users. This latter feature is ideally suited to the needs of today's Application Service Providers (ASPs).

4 THE ASSOCIATIVE DATA MODEL

In the associative data model, a database comprises two types of data: entities and associations. Entities are things, or objects, that have discrete, independent existence. Associations are things whose existence depends on one or more other things, such that if any of those things ceases to exist, then the original thing itself ceases to exist or becomes meaningless. Associations are allowed to depend upon other associations. In the associative model, all attributes are represented as links between things within the database, and the target of every attribute is another thing that is represented within the database in its own right. In contrast, the relational data model has tuples as the source of an attribute and the target is the value contained by the cell under the column heading in the tuple. Thus in the associative data model, attributes are represented as links between the entity or association whose attribute that is being recorded as the source, a verb to express the nature of the attribute, and an entity or association as the target.

The impact of the latter representation is that things and associations are no different from associations in general. At any time the database designer or programmer may decide to describe an attribute by giving it attributes of its own. In the relational data model this would require restructuring the database, replacing a value by a foreign key and adding a new relation. The associative data model can be viewed as a vertically defined model on variable length data whereas the relational data model processes fixed length tuples horizontally.

4.1 Example Sentences Database

Consider a mail order database application. Suppose the database designer needed to store the following piece of information: "Bob ordered a recordable compact disk on July 7th, 2001 from Alpha, Inc." This piece of information contains four entities: Bob, recordable compact disk, (July 7th, 2001) and (Alpha, Inc.). Additionally this information contains three links: ordered, on and from. Three links are required to store the data, as represented in the *Sentences* DBMS, by:

Bob ordered recordable compact disk
... on July 7, 2001
... from Alpha, Inc.

or we have:

(((Bob ordered recordable compact disk)
on July 7, 2001) from Alpha, Inc.)

This information could be implemented with following tables:

Items	
Identifier	**Name**
23	Bob
14	recordable compact disk
77	July 7, 2001
81	Alpha, Inc
92	ordered
101	on
16	from

Links			
Identifier	**Source**	**Verb**	**Target**
19	23	92	14
21	19	101	77
36	21	16	81

4.2 Associative Data Model for the Example *Sentences* Database

A segment of a corresponding associative data model would involve items (ovals) and links (lines) in a semantic network diagram.

The semantic network can be implemented in the *Sentences* DBMS. Every association has the following properties: a name, source type, target type, cardinality, inverse cardinality, being sequenced or sorted, and a default target. The associative data model permits an associative type that is specific to a single entity or association. Additionally, an entity or association type may be a subtype or supertype of another type. Besides subtypes and supertypes, the associative data model allows for the use of inferred subsets and supersets. For example,

Good Customer subset of **Customer**.

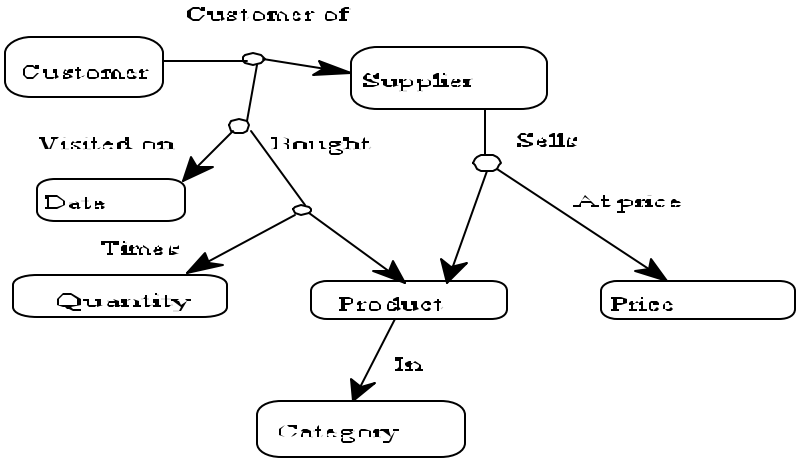
The *Sentences* Explorer is the GUI for defining the schema. Both schema and data panes are included on the GUI. Menu and tool bars allow for filtering and positioning. Drill-down and dataforms enable dynamic mechanisms for creating, displaying and manipulating data.

Sentences schema and data information are stored in chapter files. One or more chapters is combined to make a profile. Essentially a profile is a view of a Sentences database, the mechanism which facilitates modular database design. Chapters may be added to and removed from a user's profile at any time, without deleterious effects on the integrity of the database.

Sentences allows for a variety of means for importing and exporting schema and data information: Comma Separate Variate (CSV) files, a Sentence's specific file format for importing and exporting Profiles. Loading and extraction of data from Sentences can also be accomplished with API in a custom Java program.

4.3 Query Language for the Associative Data Model

Associative algebra is the basis for query processing in the *Sentences* DBMS, which is derived directly from SQL. The following operators are available, for release 1.1: union, intersection, difference, product, select, project, join, divide, extend, summarize, rename, and recursive closure. Only extend, summarize, and recursive closure differ from the traditional SQL operators. Extend forms in *Sentences* include the associative type that has the original type as source and a new type, instances of which are derived from the source as target. Summarize forms a type whose instances are formed by grouping instances of the original type that have the same sub-tree as source, and creating one instance of the new type for each subgroup, with the sub-tree as source and an instance aggregating corresponding sub-trees of the group as target. The recursive closure forms a relation by joining a self-referencing type



with itself, taking the result and joining it again with the original type, as many times as necessary.

The schema for the latter associative data model for the mail-order business would be:

Customer customer of **Supplier**
... visited on **date**

... bought P**roduct**
... time **Quantity**
Supplier sells **Product**
... at Price
P**roduct** in **Category**

An instance of sample data would be:

Bob customer of **Best Buy**
... visited on **25-June-01**
... bought **ink cartridge**
... times **10**
... bought **diskettes**
... times **200**

Example queries in the Sentences DBMS are:

"Who shops at Best Buy?"

Q: Select (**Customer** customer of **"Best Buy"**)
A: **Bob** customer of **Best Buy**
"What computer facilities has Bill bought?"
Q: Select ((((**"Bill"** customer of **Supplier**) visited on **date**) bought **Product**) join (**Product** category **"computer facility"**))
A: (((**Bill** customer of **Ables Land**) visited on **6-July-01**) bought **diskettes**) join (**diskettes** category **computer facilities**)

The *Sentences* Query Editor allows you to create and execute *Sentences* queries with a GUI. The Editor is comprised of three panes: a query pane, a schema pane, and a results pane. Queries are built using query operators and entity plus association types from the schema. The Editor supports "drag and drop" and copy/paste features. Basically the Select source retrieves the source entity or association instances from an association input set. The Select target retrieves the target entity or association instances from an association input set. A Join operator joins two input sets into a temporary output set with a source and target. Join operators return a "cross-product" output set if no join parameters are present. A Group operator allows for the traditional relational DBMS aggregation operations. Ordered results from a query are available through the use of an Order By operation.

5 CONCLUSION

The associative data model is capable of being implemented as a multi-user web-enabled DBMS, as illustrated by Sentences, and at the same time overcoming several significant limitations of the relational model. One program on a Sentences ADM database can implement many applications, or what is normally referred to as omni-competent programming; a feat not easily solved by the relational data model. A reduced emphasis on database integrity issues is part of the ADM since the links for both entities and associations are internal to the database.

Metacode for the database is implemented, using the ADM implementation in Sentences, which enables programs to be written that will operate on business entities without modification.

The web-enabled database component for database classes should benefit from the inclusion of both ORM and ADM. Sentences, representing a Web implementation of ORM suitable for ASPs, enables the future small to medium sized business a low cost and efficient means of providing their customers with a web-based tool. The ADM, inplemented in Sentences, frees the information technology model to directly represent the business process: the gap between business model and database implementation has been dramatically closed.

REFERENCES

[1] Abrial, J. R, Data Semantics, *Data Base Management*, eds J. W. Klimbie and K. L. Koffeman. North-Holland, Amsteerdam, TheNetherlands, pp. 1-60, 1974.

[2] Halpin, T., *Handbook on Architectures of Information Systems*, Chapter 4: eds P. Bernus, K. Mertins, and G. Schmidt, Springer, 1998.

[3] Halpin, T., *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*, Morgan Kaufmann, 2001.

[4] Nijssen, G. M. A gross architecture for the next generation database management systems, *Proceedings 1976 IFIP Working Conference on Modelling in Data Base Management Systems*, ed G.M. Nijssen, Freudenstadt, Germany, North Holland Publishing, 1-24, 1976.

[5] Senko, M. E. Information systems: records, relations, sets, entities and things, *Information Systems*, Pergamon Press, 1, (1), 3-12, 1975.

[6] Falkenberg, E. D. Concepts of modeling information, *Proceedings 1976 IFIP Working Conference on Modelling in Data Base Management Systems*, ed G.M. Nijssen, Freudenstadt, Germany, North Holland Publishing, 95-109, 1976.

[7] Springsteel, F., Robert, M. A., and Ricardo, C. M., The Next Decade of the Database Course: Three Decades Speak to the Next. *Proceedings of 31st SIGCSE Symposium (March, 2000)*, ACM Press, 41.

[8] Williams, S., *The Associative Data Model*, Lazy Software Ltd., 2000.