expert systems

IASTED

Proceedings of the Sixth IASTED International Symposium

EXPERT SYSTEMS

Theory & Applications

Los Angeles, U.S.A.
December 14-16, 1989

EDITOR: M.H. Hamza

A Publication of
The International Association of Science and Technology for Development - IASTED

ISBN 0-88986-137-4

ACTA PRESS

ANAHEIM * CALGARY * ZURICH

Knowledge Representation of Dynamic Features in the Semantic Binary Data Model [1]

Ragae Ghaly *Wang Ling*
Naphtali Rishe *Doron Tal*

School of Computer Science
Florida International University
University Park
Miami, Florida 33199

Abstract

In this paper, we select the Semantic Binary Database Model (SBDM) as a fact-oriented model and augment its features with activity representations. By adding this expansion to the SBDM, we have a system that meets all the requirements of a knowledge base system. The SBDM is based on a structured object scheme. It represents knowledge of the Universe of Discourse (UoD) as a collection of elementary facts of two types: unary and binary facts. Non-binary relations can be regarded as groups of elementary binary relationships.

Dynamic aspects are modeled by situation-action rules, dynamic constraints and a control mechanism for efficient triggering rules. Attached to each rule is a pair of dynamic constraints: one precondition and one postcondition, which are specified as predicate logic expressions. Triggers and constraints guarantee integrity and consistency of facts in the knowledge base. The enhanced model is a combination of production rules and structured object knowledge representation schemes.

Keywords: semantic data models, data abstraction, rule, object, event, category.

1 INTRODUCTION

Recently much attention has been devoted to the modeling of all types of constraints in a Universe of Discourse (UoD). The recent trend is to embed more intelligence into database systems. This assists in the deduction of facts and the extraction of knowledge from the database. A knowledge base system requires the complete definition of both fact specifications and activity specifications.

Therefore, the new generation of DBMS will manage not only a database of facts, but also a knowledge base of rules and heuristics regarding different applications, such as knowledge acquisition, semantic query optimization, user interfaces, and database design.

Semantic data models were introduced as schema design tools to accurately model the functional requirements that arise in a UoD. Later, they have been used as front-ends to some existing systems, and recently have become the basis for full-fledged DBMS [8]. A semantic data model is a high level of abstraction for modeling data, it is richer and has more expressive power to capture more semantics. Therefore, it helps the designers to think of data in a natural way. It also supports a framework for top-down, modular view of the schema design. It gives a complete and explicit description of the semantic properties of a database during its lifecycle (creation, use and maintenance) [2]. The enforcement of constraints and the optimization of queries lead to enhanced functionality and more efficiency by elimination of functional redundancy. Finally, it reduces the semantic overloading of data type constructor by shifting a substantial amount of schema information from the constraint side to the structural side and define behavioral properties implicitly in the schema as structural properties.

In order to explain our approach, we first introduce semantic model components and the Semantic Binary Data Model (SBDM), which are described in section 2. The remainder of the paper is organized as follows: section 3 presents background materials on dynamic features as presented in other semantic data models. In section 4, we select tools for modeling dynamic aspects to be used in our model. In section 5, we define the dynamic features in the Semantic Binary Data Model. Finally we give an example in order to demonstrate the applicability of our approach, and we conclude with a summary and points for future research.

2 COMPONENTS OF THE SEMANTIC DATA MODELS

Semantic data models are characterized by their rich semantics and their modeling constructs for describing the semantic features and properties of a UoD. The complete design and specification of semantic model must include both structure and behavior to form the completion of conceptual specification [3]. In practice, however, facts are more fundamental than activities to the users of an information system. The basic components of a semantic data model , as illustrated in Figure (1), are the following:

- **Static:** in which they describe the structural properties of the entities, relationships and the associations among the entities of the UoD.
  - An entity can be simple (irreducible) or composite (derived). A simple entity can be abstract or concrete (printable) types. A composite entity is an aggregate of simple ones.
  - Entity properties (attributes).
  - Relationships among entities.
- **Dynamic:** in which they describe the operational (behavioral) characteristics. Such operations can be classified as:
  - Simple (primitive) operations that can be applied on a single instance of entities at the low-level of data manipulation operations such as: *Select, Insert, Delete, Update, Project, Join and Intersect.*
  - Composite operations can be applied at high-level user defined operations such as: *Purchase, Reject_Order, Rotate, Register_Class, Change_Class.* Such operations are composed of several low-level operations, each operates on different entity.
- **Knowledge rules,** or semantic integrity constraints and security rules, that are general rules of inference to preserve the consistency of the data to be stored in the database. These rules are either static (on entities) or dynamic (on operations).
  - *Static constraints* specify the rules on different entities (as uniqueness integrity), rules of entity properties (as single, multivalued, null attributes) and rules of relationships among entities.
  - *Dynamic constraints* specify the rules on the operations apply on entities that keep its integrity and consistency. Since those rules applied on operations of entities, they are usually called action rules.

Static properties are defined in a schema, whereas dynamic properties are defined as specifications for programs, queries, and reports. A schema consists of a definition of all of the application's object types, including their attributes, relationships, and static constraints. An instantaneous database is a repository of data and facts, that corresponds to an instance of the schema [1]. The significance of the integration of knowledge rule system into the database is not only to enforce integrity and consistency

[1] This work has been supported in part by a grant from Florida High Technology and Industry Council.

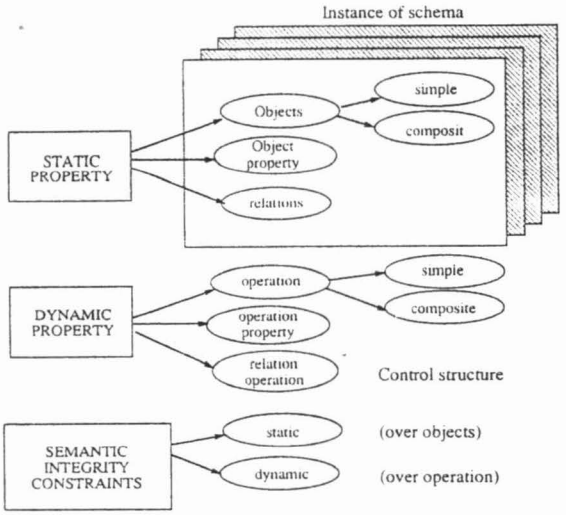

Fig 1. Semantic Data Models Components

constraints, but it also assists in the fact base design and in query opti mization process. Action rules also help to extend the semantics of the schema. They assist in the propagation of the effects of updates on the data model to maintain the assertive part of any predicate [9].

Semantic Binary Data Model (SBDM) SBDM [15,12,14,13], supports aggregation, classification and generalization abstraction mechanisms. It represents information of the UoD as a collection of elementary facts of two types:

unary facts: that categorize the objects of the UoD into entity sets where objects can have overlapped structural properties, but common properties are nonredundantly specified.

binary facts: that relate object_pairs with various kinds of relationships. Each binary relation is viewed as an inverse pair of possibly multivalued functions. The meaning of the relation remains unaltered in time, while the sets of pairs of objects corresponding to the relation may differ from time to time. A relation can be 1:1, 1:M, M:1 or M:M (many-to-many).

Non-binary relations can be regarded as groups of elementary relations.

3 BACKGROUND

Traditional databases were typically passive in the sense that a transaction or a query is executed when explicitly requested. No mechanism was provided for abstracting the consistent usage of the database entity throughout its lifetime. An active DBMS allows to specify actions to be taken automatically (without manual intervention) when certain conditions arise. The features and capabilities of active DBMSs can be traced back to CODASYL [4]. Constraints, which are predicates, are used to maintain integrity and consistency of data objects and can be associated with classes. Trigger (also called monitors or alerters) and constraint mechanisms were proposed for System-R to enforce consistency and integrity of data [5]. Triggers monitor the database for some conditions, other than those representing consistency violations. Triggers can be attached to object properties as well to their operations to generate whatever actions are desired. When a trigger predicate becomes true, the associated trigger action (body) is executed. In the following we review some models that support dynamic features:

Temporal Hierarchic Model (THM) is based on the Logical Database Model and the Entity Relationship Model. It introduces a clock, as a special entity type and maintains the current time as an entity outside the model [6]. Entities of this class in relation with other entities are used to express:

- the creation time of an entity,
- the insertion/ update/ delete time into classes,
- the time at which a relationship is established between an object and another construct in the model.

Also, using two entities of the TIME class, one can express duration *(from-to)* for entities or relations. The concept of time here is used to keep track of entity history. THM also provides a special mechanism called history_classes, to keep history information in history environments that is connected with time class.

TAXIS [3], synthesizes semantic modeling constructs with control and typing mechanisms from imperative languages. It provides some modeling concepts such as generalization (IS_A relationships), aggregation, classification with more general programming facilities (as ADT) and exception handling. TAXIS provides tools for modularizing the database transactions; they support a wide class of interactive data applications. Also, it incorporates most of the semantics associated with the specification of the operations on such data. It provides a composite operation (transaction) hierarchies and operation relationship control structures in generalization and aggregation hierarchies. It also deals with the exception handling in a hierarchical manner.

The Extended Semantic Hierarchy Model (SHM+) [3], has highly structured static constructs that are integrated with dynamic components in an elegant hierarchy. Its approach is based on an object-oriented paradigm of programming languages. SHM+ supports the abstraction mechanisms of classification, aggregation, generalization and association. Behavioral components forms an integral part of the structural building blocks. Transactions are designed at a higher level of the hierarchy.

The Event Model [10], is a semantic data model that combines static as well as dynamic primitives at a fundamental level. Each entity is treated as an ADT that has attributes. Entities flow between events. Events are either primitive (create/destroy entity, update attribute value of an entity and perusal events that make no state change) or composite (event with subevents) which are made by combining other events using sequencing, conditional, and iteration. Events aid in clarifying the meaning of operations. The Event Model also provides a hierarchy to organize entities and events. It supports aggregation, generalization and classification abstraction mechanisms. It also provides a unified support system for the entire life cycle of a database. Finally it gives a primary role to transactions, as a fundamental part of the initial functional specification of a system and is used to derive much of the refinement and implementation process.

Object_Oriented Semantic Association Model (OSAM*) [17] follows the object-oriented paradigm of modeling data where the abstract data type (ADT), property inheritance and metaclass concepts are emphasized. OSAM* allows also for rule-based knowledge components to be integrated with objects and object classes. It uses various association types and data constructors in a nested or recursive way in order to represent a complex classes of objects and semantic relationships among objects. An object, either it is concrete, abstract (conceptual) or an event, is represented by a globally unique object identifier (OID) which is generated by the DBMS at the creation time of that object. OSAM* object classes consist of a specification part and an implementation part. The specification part of a class consists of:

- its structural association relationship with other classes
- the high-level operations that provide access and manipulation of the class and its objects
- the rules and constraints that govern the actions taken.

The implementation part of a class consists of procedures and functions that implement operations and rules mentioned in the specification part in form of methods. This encapsulation mechanism that provides information hiding from the user is useful in dealing with aggregate of operations in an atomic fashion.

4 DYNAMIC ACTIVITIES IN THE SBDM

We select the Semantic Binary Data Model as a fact-oriented model and augment its features by activity representations to meet complete requirements of a knowledge system. Tools for representing dynamic activities are focused on the event-driven behaviors of the application environment. Office procedures, for example, are typically *triggered* upon completion of some awaited event: the modification of a document, the completion of a form or the arrival of a message. In our work, the representation of the dynamic features is embedded in the conceptual schema design and in the data manipulation language. For describing dynamic features in the conceptual schema, Petri-nets technique are used. The basic tool for

conceptual schema modeling of the static structures in SBDM is semantic nets. Petri-nets are used for modeling dynamic behaviors in the conceptual schema. The Petri-nets approach of modeling dynamic aspects is [16] defined by four-tuple

M = (P, T, I, O), where

P: the set of places (states), each is depicted by a circle.
T: the set of transitions (event), each is depicted by a bar.
I: the input function for each transition t_j, defines the set of places which input to it, depicted by the set of arrows from places coming to the transition, $I(t_j) = P_i$.
O: the output function for each transition t_j, defines the set of arrows from the transition going to the places, $O(t_j) = P_o$.

Attached to each transition is a pair of dynamic constraints one precondition and one postcondition, which are specified as predicate logic sentences and are represented as arcs to the transition and to an output, place respectively. The precondition specifies the requirement upon the input variables of the transition to occur. The postcondition specifies the

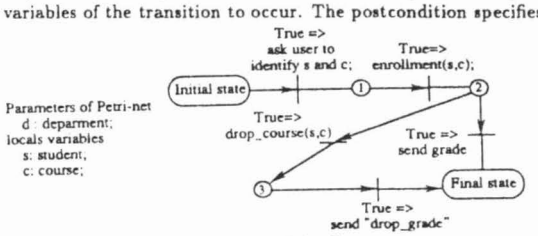

Fig 2 Function Enroll_In_Course

consequences of the event in terms of the description of the attribute values of the output variables. A transition is enabled if each of its places contains a token (simple entity). A transition can fire (trigger) whenever it is enabled and its preconditions are true. After firing of a transition a token is removed (consumed) from each of its input places and a token is placed into each of its output places, and its postconditions become true. The firing of a transition corresponds the occurrence of an event. Figure 2 shows an example of using Petri-nets.

On the other hand, predicate logic is used to assist in describing the dynamic behavior of the entities in the data manipulation laguage. Predicate logic provides more precise specifications to represent knowledge for the situations that occurs in the real world. A formal specification overcomes the ambiguities that arise when using natural languages or the graphical languages. An example of describing dynamic behavior in a business system is the situation of *hire and fire* an employee. This situation can be described with predicate logic as follow:

$S_i \models (x \in Person \wedge (x \notin Employee) \wedge hired(x)) \longrightarrow S_i + 1 \models (x \in Employee)$

$S_i \models (x \in Employee \wedge fire(x)) \longrightarrow S_j + 1 \wedge (x \notin Employee)$

S : state $\models$: is true $\longrightarrow$: if-then.

4.1 Description of dynamic features

Dynamic aspects in the SBDM are modeled by situation-action rules, dynamic constraints and a control mechanism for efficient triggering rules. Attached to each rule is a pair of dynamic constraints: precondition(s) and postcondition(s), which are specified as predicate logic expressions. A rule can fire whenever it is enabled and its preconditions are true. After firing a rule, an action will be executed and the postcondition will become true. An action may cause other rules to fire and can lead to a chain of events/actions in a specified hierarchy.

In order to describe the dynamic features of a UoD in the SBDM, the notions of event, action and transaction should be defined.

An event is a situation that occurs either during a database operation, a triggering action (of a certain rule), or an otherwise transaction operation (to be defined latter). This specifies *when* to check for a certain rule precondition(s) to be satisfied. Often more than one process is used to determine the occurrence of such an event, and more than one process may wait for its occurrence. Occurrence of an event type produces specific effects on related objects in the environment. Events can be internal or external. An internal event is a consequence of a previous event and is analogous to a *derived* event in [10]. On the other hand, external event is that occurs or is caused by something outside the scope of the UoD. An event hierarchy is specified when an event triggered by a user invocation of a single event.

An action is a behavioral property of an object and can be invoked to produce a state transition for an object. This specifies *how* to satisfy a certain rule condition. An action is composed of database primitives operations applied on an object. They are the only means to alter an object, which ensures the semantic integrity of entities. An action can also be viewed as a user's logical step applied on an entity. An object can have more than one action. Actions can be classified into:

- Insert a fact: this adds a new fact to an existing object. It also creates a new object if it does not exist and then relates it to the inserted fact and associates it with the specified relation.
- Delete a fact: this action deletes the specified fact, and if the associated object is not associated with another fact, it is also deleted.
- Replace a fact: this action is not elementary, since it is composed of both actions mentioned above. This type of actions is considered as a *transaction*.

A transaction is composed of a group of actions and produces a collection of logically related events on one or more objects. Transactions do not interact directly with database primitive operations and have a larger scope than actions do. A class of events is designed by modeling a transaction. From the conceptual view, a transaction is atomic in the sense that it is performed in an indivisible operation[15].

A function is a mechanism to describe all possible execution histories that an entity can go through[1]. A function is basically an extension of Petri-nets. Each node in the function represents a state, and each transition represents a transaction that can change the states of processes being represented. Functions are intended to model long-term events in the UoD. The main objective is to have such abstraction as a generalized process which requires communication and synchronization mechanisms of the system designer. In order to enable communication between different functions, message passing mechanisms are used which are based on the primitive operators of the communicating sequential processes (CSP) [7]. Functions are useful in enforcing dynamic integrity constraints on transaction call sequences (a student can not receive grades until he/she has enrolled in a course) and in defining the format and protocol of interactions with users. Functions provide a natural place for exception-handling, including execution arising due to time delays. Functions may be associated with different attributes which control protection and priority.

A rule fires only if the event occurs and the precondition evaluation is true (typically through a query functions on the fact base or on environment parameters). When a rule fires, its action will be *triggered*, and its postcondition is evaluated (to guarantee the database consistency). As a result, the action may cause the generation of new events that consequently cause other rules to fire. These cascading rule firings produce consequent assertions of additional information through the succession of rule activation in forward chaining expansion. Both actions and transactions are forms of procedural abstractions that causes appropriate invocation context. Before such invocations, certain preconditions must be fulfilled and actions on other entities may be necessary.

4.2 Control abstractions definition

In order to provide dynamic specifications, the SBDM is extended to support control abstractions. For high-level composite operations (actions, transactions) control abstractions are used to relate operations on objects. The three forms of control abstractions are:

- *Aggregation* of operations that can be executed either in sequence or in parallel on an object.
- *Generalization* of an operation on a generic is composed of a choice of operations, one for each subcategory (*case* is used for nonoverlapping subcategories, and sequences or parallel *if-then* control structures are used for overlapping subcategories).

- *Association* of an aggregate object is expressed as a repetition operation on all members of that aggregate (*do-while* or *for-each* control structures are used in the framework of either sequence or parallel control structures). An operation on a Set is composed of an operation that is applied in sequence or parallel to each member of the Set.

The operational hierarchy of transaction/action supports the operation modularity. Elementary action is regarded as the smallest unit of operations which change one fact. Actions can occur simultaneously and do not violate any integrity constraints of the conceptual schema. A transaction is a minimal set of actions and can be executed either in sequence or in parallel [15].

4.3 The Time Notion

The time concept (time axis) is a fundamental entity for any method attempting to model dynamic information and to express the proper behavior of activities [11]. In many applications the dynamics of the system can be modeled by the notion of causality (sequence of events caused by the invocation of a single event). In some information systems it is relatively important to express the occurrence of some events relative to another, but if the time axis is being represented as an entity, it can be useful in the conceptual schema design and is not considered at the internal level. Also in defining time axis, the DBMS needs to support the following:

- *Valid time:* the clock time that the event occurred in the real world, independent of the recording in database.
- *Transaction time:* concerns with the storage of information in the DB. Transaction time of an event is the transaction sequence number (an integer) of the transactions that already stored.
- *User defined time:* it is provided by the user or application program.

Another point of view of introducing the time axis is the question of time being absolute or relative to some events.

5 AN ILLUSTRATIVE EXAMPLE

As an illustrative example, the University Schema (Figure 4) is considered [12]. Events are represented by a bar and arrows comming from preconditions and going to postconditions. An important mechanism that can be implemented is the time stamping introduced in THM. Each event of CREATE NEW as a basic construct in SBDM [15,12] is augmented by <time> as an option parameter. The same can be performed while relationships are established among objects by RELATE , CATEGORIZE or DECATEGORIZE constructs.

Two basic actions are presented Take_Course and Drop_Course. Each action is specified by a set of parameters set of preconditions and a set of postconditions that specified the dynamic constraints. The actions Take_Course and Drop_Course are described as follow:

Action Take_Course (s : STUDENT, c : COURSE) ;
precondition:
c is offered? ;
limit not reached? ;
c is not taken before by s? ;
then:
insert c to the list taking course of s ;
insert s to the enrollment list of c;
increment the size of c;
postcondition:
s is in c enrollment list?;
c is in s course list?;
end_action Take_Course ;
Action Drop_Course (s : STUDENT, c : COURSE);
precondition:
s is taking course c? ;
then:
delete c from the list taking course of s;
delete s from the enrollment list of c ;
postcondition:
end_action Drop_Course;

The transaction Transfer_From_Course is composed of two actions, namely Take_Course and Drop_Course.

In representing the function Enroll_In_Course as a long term transaction it is important to specify all the states and transitions inside the function body.

Function Enroll_In_Course (d : DEPARTMENT);
Local Parameters:
s: STUDENT ;
c: COURSE ;
grade: {0..100} ;
States:
initial: initial_state ;
final: final_state ;
others: state_1 , state_2, state_3;
Transitions:
Obtain information:
from initial_state ;
to state_1 ;
Condition: none ;
Action: get c , s from user ;
Enrollment:
from state_1 ;
to state_2 ;

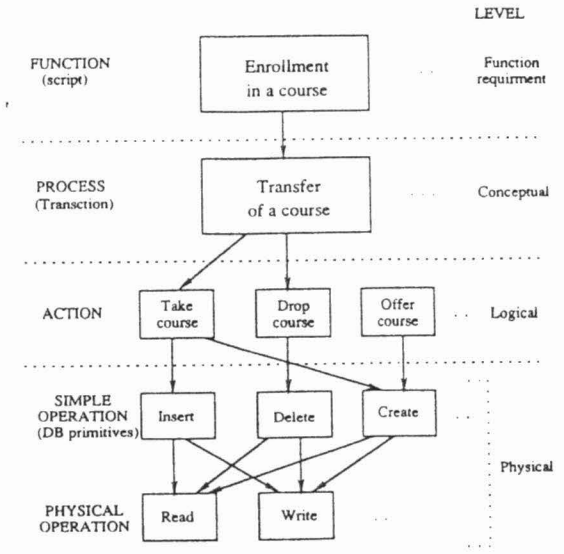

Fig 3 Operational hierarchy

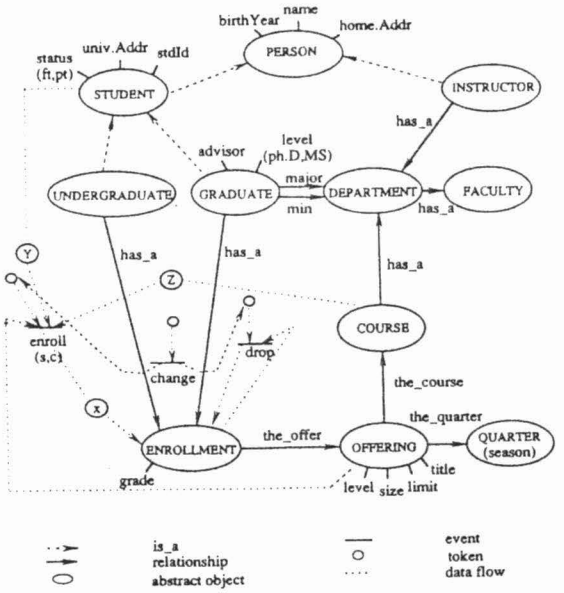

Fig 4 Activity representation in the university database schema

Condition: none;
Action: Take_Course(s,c);
Drop course:
from state_2 ;
to state_3 ;
Condition: none ;
Action: Drop_Course(s,c);
Grade:
from state_2 ;
to final_state ;
*Condition:*none;
Action: Send "drop-grade" message ;
Drop_Grade:
from state_3 ;
to final_state ;
*Condition:*none;
Action: Send grade ;
end Enroll_In_Course;

6 CONCLUSION

In this paper, we integrated the Semantic Binary Database Model (SBDM) as a fact-oriented model with dynamic activity representations in order to have a system that meets all of the requirements for a knowledge base system. Dynamic aspects in the SBDM are modeled by situation-action rules, dynamic constraints and a control mechanism for efficient triggering rules. Attached to each rule is a pair of dynamic constraints: one precondition and one postcondition, which are specified as predicate logic expressions. A rule fires whenever it is enabled and its preconditions are true. After firing a rule, an action will be executed and the postcondition will become true to establish integrity and consistency in the Fact Base. An action may cause other rules to fire and can lead to a chain of events/actions in a specified hierarchy. Triggers and constraints guarantee integrity and consistency of facts in the knowledge base. The enhanced model is a combination of production rules and structured objects knowledge representation schemes.

References

[1] A. Borgida, J. Mylopoulos, and H.K.T. Wang. Generalization / Specialization as a Basis for Software Specification. In M.L. Brodie, editor, *On Conceptual Modeling*, Spring-Verlag, 1984.

[2] M.L. Brodie. On the Development of Data Models. In M.L. Brodie, editor, *On Conceptual Modeling*, Spring-Verlag, 1984.

[3] M.L. Brodie and D. Ridjanovic. On the Design and Specification of Database Transactions. In M.L. Brodie, editor, *On Conceptual Modeling*, Spring-Verlag, N.Y., 1984.

[4] CODASYL Data Description Language Committee. *CODASYL Data Description Language Language Journal of Development.* NBS Handbook 113, June 1973.

[5] K.P. Eswaran. Specification, Implementations, and Interactions of a Trigger Subsystem in an Integrated Data Base System. *IBM Research Report RJ1820*, August 1976.

[6] A.L. Furtado and R.J. Neuhold. *Formal Techingues for Data Base Design.* Spriny-Verlay, Berlin, 1986.

[7] C.A.R Hoare. Communicating Sequential Processes. *CACM*, 21(8):666–677, August 1978.

[8] R. Hull and R. King. Semantic Database Modeling: Survey, Applications and Research Issues. *ACM Computing Surveys*, 19(3), September 1987.

[9] L. Kerchberg (Editor). Expert Database Systems. The Benjamin/Cummings Publishing Co. Inc., 1986.

[10] R. King and D. Mcleol. Semantic Data Models. In S.B. Yao, editor, *Principal of Database Design*, pages 115–146, Prentice Hall, 1985.

[11] N. Prabhakaran and E. Falkenberg. Representation of Dynamic Features in a Conceptual Schema. In *The Australian Computer Journal*, pages 98–104, August 1988.

[12] N. Rishe. *Database Design Fundamentals: a Structured Introduction to Databases and a Structured Database Design Methodology.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[13] N. Rishe. *Database Design: The Semantic Modeling Approach.* Prentice-Hall, Englewood Cliffs, NJ, to appear in 1990.

[14] N. Rishe. Semantic Database Management: from Microcomputers to Massively Parallel Database Machines. Keynote Paper, Proceedings of The Sixth Symposium on Microcomputer and Microprocessor Applications, Budapest, October 1989.

[15] N. Rishe. Transaction-management System in a Fourth Generation Language for Semantic Databases. In H.H. Hamza, editor, *Mini and Micro Computers: From Micro to Supercomputers. (Proc. of the ISMM Int'l Conf. on Mini and Microcomputers)*, pages 92–95, Acta Press, 1988.

[16] A. Solvberg and C.H. Kung. On Conceptual and Behavioral Modeling of Reality. In T.B. Steel Jr., editor, *Database Semantics (DS-1)*, pages 205–221, Elsevier Science Publishers B.V. (North Holland), 1986.

[17] S.Y.W. Su, V. Krishnamurthy, and H. Lam. An Object-Oriented Semantic Association Model (OSAM*). In S. Kumara, editor, *AI in Industrial Engineering and Manufacturing: Theoretical Issues and Applications*, American Institute of Industrial Engineers, 1988.