Journal of Intelligent Information Systems, 3, 185–203 (1994)

Representation of Highly-Complex Knowledge in a Database

AVIGDOR GAL AVIGAL@IE.TECHNION.AC.IL

OPHER ETZION IERETZN@IE.TECHNION.AC.IL
Information Systems Engineering Group, Faculty of Industrial Engineering and Management, Technion-Israel Institute of Technology, Haifa 32000, Israel

ARIE SEGEV SEGEV@CSR.LBL.GOV
Haas School of Business, University of California and Information & Computing Sciences Division, Lawrence Berkeley Laboratory, Berkeley, CA 94720, USA

Abstract. This paper presents a unified framework for representing highly-complex knowledge in a database as a new paradigm for handling large and complex information in an easy and efficient manner. The framework provides a database with the capabilities to support next generation databases for decision support systems through the use of derivation rules, temporal information, knowledge from multiple sources with different measures of quality and epistemic knowledge. The model integrates concepts from both the *database* and the *artificial intelligence* disciplines.

Keywords: Complex Information Modeling, Intelligent information systems, Temporal databases, Active databases

1. Introduction

Data stored in database management systems (DBMSs) are supposed to represent portions of the real world for the use of computerized applications. Most DBMS models and tools were designed to support information with relatively simple structure, thus limiting the scope of applications that can be naturally supported by this technology. This restriction has become an obstacle with the increasing sophistication of applications such as decision support systems and the need for fast applications development. The lack of support for highly complex knowledge in DBMS products either deters organizations from developing such applications or forces system developers to create ad-hoc solutions. The latter case is usually not general enough to be reusable; it is also expensive, time consuming and hard to verify.

This research is intended to bridge the gap between current DBMS technology and the technology required to support applications which use highly complex knowledge. The proposed strategy is to extend the database schema to include various types of meta data that will enable to represent complex knowledge and reason about it. The main components of these meta data entities that were identified in the context of decision support systems are:

Derivation Rules: rules that derive the value of a data item as a function of values of other data items.

Temporal Knowledge: time points or intervals that are associated with each data item. Examples: the time point in which the value becomes known or believed, an interval during which the value is believed to be valid etc.

Data Quality: knowledge about the quality of each data item's value; this knowledge may either be related to the information source or be specified according to some ordinal scale.

Epistemic Knowledge: different concurrent viewpoints of the same knowledge.

The use of these meta data entities as DBMS primitives extends DBMS functionality. This extension enables to capture complex decision support systems in a more natural fashion using high level language and structure.

The model presented in this paper is aimed to support the different elements discussed above, where each element is called a *dimension*. The PARDES model (Etzion, 1993a) that supports the first requirement (derivation rules), was chosen as a basis, while other requirements are extensions of this model. Each data item is represented in the database as an ordered pair $< d, e >$ where d is a data item and e is the extension of a data item, including knowledge associated with the different dimensions.

We assume an append-only database where changes can occur to data, meta-data, rules and constraints. Any of those changes can also be retroactive or proactive. The active property of the model is more general than the common active databases approach of Event-Condition-Action (ECA) (McCarthy, 1989) in that it allows a general definition of statements called *invariants* (Etzion, 1993a). These statements enforce the database to maintain consistency at all times without the need to explicitly define the triggering events. The work done so far in other studies, as will be shown in Section 1.2, did not introduce suitable solutions to problems resulting from integrating the above elements in a database.

The construction of a model supporting all these elements is not a trivial extension of existing models. This was demonstrated in (Etzion, 1994), where the combination of *active* and *temporal* database functionalities are discussed.

1.1. A motivating Example

As a motivating example we present an application that requires the combination of both active and multi-dimensional knowledge in a database. The case study is based on the Cournot game (Tirole, 1989); (Cournout, 1997):

There are three manufacturers of instant coffee (Snowhite, LRR (Little Red Ridinghood) and Goldilox) that have to decide each month on the quantity to be produced for the following month. Each manufacturer makes the decision about its production quantity based on its assessment of the quantities produced by other manufacturers (Estimated-Production), its own strategy (maximum revenue, a certain market share etc.) and general knowledge about the market's behavior. Each manufacturer has its own deadline for the production decision.

We assume in this paper that there is a single market price for this type of instant coffee that is determined periodically as a function of the total quantity produced in that period, and that the following equation represents that function:

$$\textit{Total-Quantity} * \textit{Market-Price} = \textit{Market-Constant}$$

Historical information is needed in order to obtain an accurate estimation of the *Market-Constant.*

Each manufacturer attempts to estimate its competitors' decisions prior to making its own decision. For example, assume that all three manufacturers have an objective function of maximum revenue. Each of the manufacturers wants to manufacture as much as possible, yet without decreasing the price of the instant coffee to such a level that would cause a decline in its total revenue. If Snowhite knows the competition's strategy of both LRR and Goldilox, it can estimate their production decisions, and maximize its revenue through an optimal production decision.

The use of the different dimensions in the manufacturers' domain is demonstrated in the following examples:

1. The production decision of each manufacturer is derived using an algorithm that is automatically triggered by the modification of the estimated production of any of the manufacturers.

2. In order to assess the competitor's decisions, a manufacturer needs to have the information that was available to a competitor at a given time point.

3. A new information regarding a competitor's Estimated-Production may re-activate (possibly proactively) the production decision algorithm.

4. Information may arrive from many sources, each of them has a different reliability level. A measure of confidence based on past performance is associated with each of them.

5. Each competitor should be able to reason about a data item as it is known by other manufacturers. For example: In Snowhite's databases, it is recorded that a knowledge-item α about Snowhite is known to LRR since May 1992 and to Goldilox since July 1992. LRR is believed to refer to α as a fact, while Goldilox is believed to assign α a certainty value of 0.5.

1.2. Related work

The active aspects of databases have been investigated in research such as (Dayal, 1988), (Etzion, 1993a), and (Stonebraker, 1991). Active databases extend the modelling capability of a database schema by adding the *rule* construct. A *rule* is a database element consisting of two major components: the *trigger* component and the *action* component. The *trigger* component defines the prerequisites for the execution of the rule's operational part. Most of the current active database research follow the E-C-A (Event-Condition-Action) architecture (McCarthy, 1989) in which the triggering component consists of two parts: event detection and condition evaluation. The *action* component contains the operational part of the rule applied in most contemporary active database models as a database operation or a user defined program.

Temporal semantics was dealt with in works on temporal databases, such as: (Clifford, 1987); (Navathe, 1989); (Gadia, 1988); (Snodgrass, 1987); (Su, 1991); (Rose, 1991) and many others, yet the capabilities of the model with the presence of retroactive and proactive updates have not been investigated. Most of the research in this area focused on the structural semantics, assuming that the update process is applied either in a procedural manner (Wuu, 1992) or as part of the retrieval language. Some models (such as (Ariav, 1986); (Shoshani, 1986) and (Wiederhold, 1991)) enforce a single value for each time point. As a result, a mechanism to handle retroactive and proactive updates is either disabled or applied in an unnatural manner.

Information quality has been discussed in the context of AI, motivated by the fact that in the absence of information quality, decisions are taken based on an inaccurate or an out-of-date data ((Bonoma, 1985); (Johnson, 1990) etc.). The majority of research efforts on information quality has focused on providing *quality indicators* (Jang, 1992), data about data, from which the information quality can be derived. The decision-analytic approach (e.g., (Keeny, 1976)) and utility analysis under multiple objectives (e.g., (Chankong, 1983)) describe solution approaches for specifying preferences and resolving multiple objectives. The preference structure of the user is specified using an hierarchy of objectives. Through decomposition of objectives the hierarchy is reduced to a single value. The decision-analytic approach assumes the existence of continuous utility function. Later research (Jang, 1992) lessens this requirement to local dominance relationships between quality parameters.

Reasoning about the world is contingent on the reasoner's knowledge known to those who process the reasoning. This knowledge is referred to as *epistemic knowledge*. Research of the ability to identify the epistemic knowledge and to maintain conclusions based on this knowledge, are mostly based on different versions of non-monotonic logic ((Gardenfors, 1988); (Poole, 1988); (Levesque, 1989) etc.). For example, in (Levesque, 1989), two modal operators are defined, B and O, where Bα is read as "α *is believed*" and Oα is read as "α *is all that is believed*". These operators are used to develop a proof theory, by which derived knowledge is maintained, based on epistemic knowledge.

The combination of the temporal and active aspects has been investigated in (Su, 1991); (Wuu, 1992); (Edera, 1993) and (Sistla, 1993). The OSAM*/T presented in (Su, 1991) is an object-based temporal knowledge representation model which combines update rules with temporal characteristics, extending the object oriented model OSAM* (Su, 1989). Rules are used to capture temporal semantics other than the valid start and end times of a tuple. Corrections result in overwrites or deletions so in this model information may be lost. The combination employs restricted update protocol, with no proactive and retroactive updates. Another model which do not allow proactive and retroactive updates is the model presented in (Sistla, 1993), which combines temporal triggers in a database.

The update is done by replacing the current value with a new one, adding the old one to the history of the variable. Issues related to processing rules where the event component is a temporal calendric expression are discussed in (Chandra, 1993) and (Chandra, 1994).

The model presented in (Wuu, 1992) describes a database supporting a planning system; it is based on the EER model in which the database components change states as a result of external events. Unfortunately, the active part uses an imperative language, thus reasoning about the application's flow is not possible. Furthermore, the well documented problems of imperative programming (time consuming, difficult to verify etc.), (Abiteboul, 1988) and (Gal, 1992), are still present in this model.

An interesting work regarding performance issues is presented in (Edera, 1993); it extends a technique for incremental recomputations of active relations (Qian, 1991) to handle temporal active relations.

An example of a combination of several aspects of the data is given in ((Gadia, 1993)). In that paper data has spatial, temporal and belief aspects organized in a relational database. Unfortunately, the relational model, due to it simplicity and the normalization processes, can hardly satisfy a semantic model of this sort and would probably require a lot of unnecessary maneuvers to maintain the required functionality. The lack of the active aspect significantly decreases the functionality of the model.

As a conclusion, there is no unified model that combines all of the above aspects along with sufficient database support and decision tools. Despite having useful parts of the required solution, current models cannot properly support the functionality we require. nevertheless some of them contain useful ideas for the required solution.

2. The data model

2.1. The basic model

This Section introduces briefly an *active object-oriented* database model that follows the ideas of the PARDES model (Etzion, 1993a) as a basis for the temporal extension.

A database consists of a collection of **objects**. Similar objects are instances of the same **class**, while classes are organized in a **generalization lattice**. A class definition contains the specification of **properties** that are applicable to its instances, along with their types. An object has a set of associated **variables**, each variable is an instance of a property. A **variable state** is a value belonging to the range of the relevant property, bounded to the variable. The value can be an atom, a set, a sequence, a tuple or a reference to another object. An **object state** is a set of all its variable states. Although each object has a unique **object identity**, for convenience reasons, an object is identified by a subset of the object's state. This subset is referred to as the **object identifier**.

Figure 1 shows the database schema of a manufacturer's knowledge about manufacturers (including itself).

Periodic-Info and *Global-Periodic-Info* are nested properties that consist of the required information for each production period.

class=	*Manufacturer*
properties=	*Name*
	Unit-Cost
	Competition-Strategy
	Periodic-Info:set of:
	Period
	Estimated-Production
	Actual-Production
	Decision-Deadline
Class =	*Self*
Generalizations =	*Manufacturer*
Properties =	*Periodic-Decision: set of:*
	Production-Decision
class=	*Global-Knowledge*
properties=	*Global-Periodic-Info: set of:*
	Period
	Total-Quantity,
	Market-Price
	Market-Constant
Class =	*Competitor*
Generalizations =	*Manufacturer*

Figure 1. Schema of Manufacturer Knowledge

Self and *Competitors* are specializations of *Manufacturer*. They inherit all the properties of *Manufacturer*; in addition *Self* contains the property *Periodic-Decision*, designating the decision as derived by the system's rules.

The underlined property *Name* is the object-identifier of the class *Manufacturer*. *Global-Knowledge* is a singleton class, hence no object-identifier is required.

Competition-Strategy denotes the strategy by which the manufacturer makes its decision (example: maximum revenue). In the "self" case, this strategy is the actual competition strategy while in the competitors case it is only a conjectured one.

Estimated-Production represents the estimated production quantities of the competitors and *Actual-Production* is the actual production decision as known for that period in retrospect.

A class description describes the structure of its instances. A partial example of an instance of the class *Manufacturer* in *Snowhite*'s database referring to *LRR* designating *Snowhite*'s knowledge about *LRR* is presented in Figure 2.

The rules in our example are shown in Figure 3. The rules are expressed in the form of invariants. An *invariant* is a declarative definition of dependencies that should hold for any instance in a consistent database.

Rules (d1) and (d2) are *data driven*, that is, any change in one of the derivers (e.g., *Competitor.Estimated-Production* in (d1)) requires recalculation of the rule. Rules (d3)

Class= *Competitor*
Name= *LRR*

Unit-Cost=6
Periodic-Info:
  Period=Dec 1991
  Estimated-Production=280
  Actual-Production=280
  Decision-Deadline=Nov 1991

Periodic-Info:
  Period=Jan 1992
  Estimated-Production=260
  Actual-Production=280
  Decision-Deadline=Dec 1991

Figure 2. An Instance Example

Derivations

(d1) Competitors-Total-Estimation := sum(Competitor.Estimated-Production)
(d2) Production-Decision :=
sqrt(Market-Constant*Competitors-Total-Estimation/Unit-Cost) –
Competitors-Total-Estimation
when Competition-Strategy = max-profit
(d3) Total-Quantity := sum (Actual-Production)
(d4) Market-Constant := avg(Total-Quantity*Market-Price)

Constraints

(c1) LRR.Production-Decision$\leq$600

Figure 3. Rules Definitions

and (d4) are *event driven*, that is, they are calculated as a response to an event. Events are not shown in Figure 3, but an example event is *End-Of-Period.* After the event is specified it can be attached to more then one rule, in our case to (d3) and (d4).

(d1) sums, for each manufacturer, the Estimated-Production of all its competitors.
The formula presented in (d2) is the result of maximizing the revenue function of each manufacturer. Different goal functions would yield other formulae.
In (c1) the production quantity of LRR is limited to 600 units per period.

Manufacturer	Dec 1991
LRR	280; 300
Goldilox	300; 350

Figure 4. Estimation of Production Quantities

2.2. *Variable States and Extensions*

In our framework, unlike a conventional database where each variable has a unique value, several values of the same variable (with different dimensional characteristics) can coexist simultaneously.

VS(α) (**variable state of a variable** α) is a sequence of pairs representing different variable's values. The symbol α designates a variable of a given object; each pair is a *state-element.*

A **state element** is a an ordered pair $< d, e >$ designating the combination of data and the knowledge associated with it. d is the variable's value and e is the *extension.*

The **extension** is the set of all the *dimensional variables* associated with the value.

A **dimensional variable** is a set of variables associated with a particular dimension. Example: $\{$*source, confidence value*$\}$ is the dimensional variable associated with the quality dimension.

Figure 4 displays a table of Snowhite's estimations of its competitors' production in December 1991. Values of each manufacturer's Estimated-Production are all state elements of the variable Estimated-Production. Along the temporal dimension we might have different values of Estimated-Production, for example, for LRR, the two values may be the history as we see it from the observation time December 1991, where the value 280 represents a belief in September 1991, and 300 represents the change of belief in December 1991. Along the quality dimension we might have different values as well; for example, we know Snowhite's estimation of Goldilox's production is 300 units, but a reliable informer notified Snowhite that the production is 350 units. 300 and 350 are both values of Estimated-Production that had the same temporal value, yet each one of them has a different value in the quality dimension.

In some cases, two or more state elements overlap in the sense that several values can be retrieved as a result of a query. For example, the result of the query: "What is Snowhite's estimation of Goldilox's production quantity for December 1991?" could be either 300 or 350. There is no trivial way of deciding which value is the "correct" one. Furthermore, queries may yield different results when executed with respect to different observation times.

In order to select the desired value among overlapping state elements a preference relation has to be devised. Different assumptions may yield different preference relations. Preference relations may be defined according to various criteria: the source of information, a confidence value that is associated with the information, temporal information etc.

For example, a preference relation with respect to the temporal dimension, introduced in the context of active temporal databases (Etzion, 1994), is based on the assumption that a data item α is preferred to a data item β if it was decided at a later time. It represents the belief that knowledge monotonically improves with time, and that all the previous decisions are available when a later decision is made, thus a later decision is based on a better knowledge than an earlier one, and can override all previous decisions. The support of several dimensions requires a combination of preference relations from different dimensions (such as temporal and quality dimensions). For example, if a database retrieval operation selects data items based solely on their decision time, it might chooses a data item with very low confidence value, rather then a less current but more reliable data item.

2.2.1. The Dimensional Variables

Each aspect of the data is represented by a *dimensional variable*, which is a set of variables defining the dimension. In this section we describe the different dimensional variables:

The Temporal Dimension: The fundamental assumption is that for each object, several time types (Snodgrass, 1986) are required to model the desired functionality. A basic set of time types, as defined in (Etzion, 1994) consists of:

Transaction Time (t_x)- The commit time of the transaction which updates the variable state.

Decision Time (t_d) - The time in which the variable's value has been decided in the database's domain of discourse.

Valid Time (t_v) - The time points in which the decision maker believes that this value reflects the object's value in the real world. t_v is expressed by a *temporal element* (Gadia, 1988) which is a time-point or an interval $[t_s, t_e]$ or a collection of intervals and time-points. If t_v is an interval, it is believed that the object value is constant in this interval. This case has been classified as *stepwise change* (Segev, 1987). Other possible cases are *discrete events change* where the value is defined only in given time-points and *continuous change* where the value changes continuously according to some function. In this paper we consider the stepwise case only, while the other cases are natural extension of this discussion.

Observation Time (t_o)- a time point associated with a retrieve operation that designates the time point from which the retrieve operation is viewed.

These time types are restricted by the following constraints:

1. $t_s \leq t_e$. (negative intervals are not allowed).
2. $t_x \geq t_d$. (decisions cannot be speculated).
3. $t_o \leq NOW()$. (future observation times are undefined).

The **temporal dimensional variable** is a set of variables that represent transaction time, decision time and valid time. Observation time is discussed in Section 3.

The Quality Dimension: The quality dimension associates a data item with its source and the degree of confidence in it. The dimensional variable is a set of the following variables:

Source (c_s)- an identifier of the source that provided the information. A source may denote a specific agent, or a general source such as: a newspaper, a rumor, an industrial espionage, inside information, etc.

Confidence value (c_v)- a value which designates the degree of confidence in the variable's value, expressed in some ordinal scale such as [0,1]. The confidence value may be attributed to the source; in this case, the confidence values of all the information provided by a certain source are defaulted to a given value.

The Epistemic Dimension: The epistemic dimension associates a knowledge item with a set of viewpoints. Each of this viewpoints is assumed to have access to the knowledge item, possibly under certain conditions.

The dimensional variable of the epistemic dimension is a set of pairs A_s. Each pair $a_t \in A_s$ is of the form <*w*, *cond*>, where w is a *world* and *cond* is a *condition.*

A **world** is a collection of viewpoints. Each viewpoint may belong to a single world. For example, in our case study the knowledge of LRR is a world and so is the knowledge of Goldilox.

A **condition** is an assertion that restricts the accessibility to the knowledge only when the assertion is satisfied. For example, the condition $t_x < (NOW() - 1)$ stands for the fact that a world is entitled to a knowledge item only one month after it was committed in the database. The default condition is "null" designating unconditional accessibility. A condition may refer to variable values as well as dimensional variables such as temporal types $(t_x, t_d$ and $t_v)$, confidence values $(c_s$ and $c_v)$ or members of A_s. Circular conditions are considered as a system design error.

Worlds are ordered in an inheritance lattice. The lattice imposes a partial order relation denoted as $\leq_w$, that is, $w_1 \leq_w w_2$ stands for the fact that w_1 inherits all the knowledge accessible to w_2.

2.2.2. An Example

Figure 5 presents an example of state elements. All state elements consist of a value and dimensional information. The state elements *s1* and *s2* belong to the variable Estimated-Production of LRR in the Period December 1991. The state elements *s3, s4, s5* and *s6* belong to the variable Estimated-Production of LRR in the Period January 1992.

Period=Dec 1991
Estimated-Production:
(s1) 280, t_x=Oct 1991, t_d=Sep 1991, t_v=[Sep 1991, ∞)
c_s=LRR, c_v=1, $A_s = (< \mathit{Snowhite}, \mathit{null} >, < \mathit{LRR}, \mathit{null} >,$
$< \mathit{Goldilox}, (t_x < (\mathit{NOW}() - 1)) >)$
(s2) 300, t_x=Nov 1991, t_d=Nov 1991, t_v=[Dec 1991, ∞)
c_s=LRR, c_v=1, $A_s = (< \mathit{Snowhite}, \mathit{null} >, < \mathit{LRR}, \mathit{null} >)$
Period=Jan 1992
Estimated-Production:
(s3) 260, t_x=Oct 1991, t_d=Sep 1991, t_v=[Sep 1991, ∞)
c_s=LRR, c_v=1, $A_s = (< \mathit{Snowhite}, \mathit{null} >, < \mathit{LRR}, \mathit{null} >,$
$< \mathit{Goldilox}, (t_x < (\mathit{NOW}() - 1)) >)$
(s4) 250, t_x=Nov 1991, t_d=Nov 1991, t_v=[Oct 1991, ∞)
c_s=LRR, c_v=1, $A_s = (< \mathit{Snowhite}, \mathit{null} >, < \mathit{LRR}, \mathit{null} >)$
(s5) 270, t_x=Nov 1991, t_d=Nov 1991, t_v=[Nov 1991, ∞)
c_s=Snowhite, c_v=0.8, $A_s = (< \mathit{Snowhite}, \mathit{null} >,$
$< \mathit{Goldilox}, (t_x < (\mathit{NOW}() - 1) >))$
(s6) 500, t_x=Dec 1991, t_d=Dec 1991, t_v=[Nov 1991, ∞)
c_s=rumor, c_v=0.7, $A_s = (< \mathit{Snowhite}, \mathit{null} >)$

Figure 5. The Estimation of LRR's Production

3. Retrieving highly-complex knowledge from a Database

As demonstrated in Figure 4 and Figure 5, a single variable may include more than one state element. The functionality of operations in a highly-complex knowledge environment requires filtering out some state elements or creating new state elements based on the aggregation of existing ones. The automatic elimination of state elements using predefined preference relations is especially important for novice users, which do not comprehend the complicated processes involved in query processing of such a database.

In our model we use a single primitive named *filter*, introduced in Section 3.1, to select state elements based on a given selection criteria. The existence of a single primitive eases the task of query optimization (see Section 3.2). On the other hand, the use of a single primitive as a query language is tedious, hence a higher query language that is automatically translated to filters is required. An example of such a query language in the temporal active context is discussed in (Etzion, 1993b).

3.1. Multi-dimensional Filters

A substantial amount of work has been done on the optimization of queries in databases. Alas, these optimizers assume the simple structure of the relational model, whereas optimizers for databases that support complex objects are still evolving (Lanzelotte, 1992). Since our model employs a very complex structure, our goal is to simplify the retrieval mechanism in order to ease the task of optimizing queries. In this section we shaw a retrieval mechanism that is based on a single retrieval primitive, called *filter*.

A *filter* is a function that maps a set of state elements to a set of state elements based on a given condition or operation. A filter is defined as:

$$f(sse, arg) : sse \rightarrow sse1$$

where sse and $sse1$ stand for a set of state elements and arg stands for an argument that is passed to the filter. Each filter has an associated fo (filter operation). Operation types that are associated with filters are *SELECT* and *GENERATE.* [1]

sse may be the entire database or a result of another filter. In all the examples (unless mentioned otherwise) we assume that sse is the set of all state elements of the *Estimated-Production* of *Manufacturer* LRR's production for the period of January 1992 as presented in Figure 5.

3.1.1. Atomic Filters

1. **Variable Filters:** (VF) returns a set of state elements of variables included in a variable list vl and satisfying a set of conditions *cond.*
   VF: arg=<vl,cond>; fo=SELECT $\{s \in v \mid v \in vl \wedge$ *v satisfies cond*$\}$. Example:
   VF(DB, Estimated-Production,
   {Manufacturer-Name=LRR, Period=Jan 1992})=
   $\{s3, s4, s5, s6\}$ where DB stands for "the entire database".

2. **Temporal Filters:**

   Observation Time: (OT) is a filter that defines a variable state relative to an observation time t_o to be the collection of all state elements that were committed until t_o (persisted in the database no later than t_o). Decisions that were made prior to t_o but not committed in the database by t_o are not included in OT. The use of observation time enable us to phrase queries about what would be the result of a retrieval operation if it had been issued in a given time point that is not necessarily NOW(). Queries of this type are useful in applications where tracing of decisions or actions relative to a given knowledge is required. Examples of such applications are auditing systems and decision analysis systems.
   OT: arg=t_o*; fo=SELECT* $\{s \mid t_x(s) \leq t_o\}$. Example:
   OT(sse, Nov 1991)=$\{s3, s4, s5\}$

   Relevant Time: (RT) is a filter that selects all the state elements whose validity intervals intersect with a given temporal element t_l.
   RT: arg=t_l*; fo= SELECT* $\{s \mid t_l \cap t_v(s)) \neq \emptyset\}$. Example:
   RT(sse, [Aug 1991, Oct 1991))=$\{s3, s4\}$

   Periodical Average: (PA) is a filter that creates new state elements whose values are calculated by averaging the values of all state elements with overlapping validity time. t_v of the created state element is the intersection of t_v's of the participating state elements. Let $t_{min} = min(t_s(se)) \mid se \in sse$; $t_{max} = max(t_e(se)) \mid se \in sse$.

PA: fo=GENERATE state-elements $\{q_1, \ldots, q_n\}$ *s.t.*

(A) T=$\{\tau_1, \ldots, \tau_n\}$ is a partition on the set of time-points TP, where $\forall t \in TP : t_{min} \leq t \leq t_{max}$.

(B) $\forall t_a, t_b \in \tau_i : [\forall se \in sse \mid t_a \in t_v(se) \rightarrow t_b \in t_v(se)]$.
We denote the set of all se satisfying this condition as SE_i

(C) $\forall i : t_v(q_i) = \tau_i$.

(D) $\forall i : value(q_i) = avg_{se \in SE_i}(value(se))$.

Example: PA(sse)=

(q1)	260,	t_v=[Sep 1991, Oct 1991)
(q2)	255,	t_v=[Oct 1991, Nov 1991)
(q3)	320,	t_v=[Nov 1991, ∞)

In this example we omit the rest of the extension variables.

3. **Quality Filters**

Weighted Average: (WA) is a filter that creates a new state element whose value is calculated by averaging the values of all state elements based on their confidence measure. c_v of the created state element is the average of c_v's of the participating state elements.

WA: fo=GENERATE state-element q s.t.:

(A) $value(q) = \frac{\sum_{se \in sse} value(se) * c_v(se)}{\sum_{se \in sse} c_v(se)}$

(B) $c_v(q) = avg_{se \in sse}(c_v(se))$.

Example: WA(sse)=307, c_v=0.88.

4. **Epistemic Filters**

Observer View: (OV) is a filter that selects all the state elements that are accessible from a certain viewpoint (v).
OV: arg=v; fo= SELECT $\{s \mid \exists a_t \in A_s(s), \exists w' : v \in w' \wedge w' \leq_w a_t.w \wedge a_t.cond$ *is satisfied*$\}$. Example:
OV(sse, Goldilox)=$\{s3, s5\}$.

3.1.2. *Compound Filters*

The complex retrieval task requires the use of *compound filters*, non-atomic filters that uses the results of other filters. A compound filter is represented either in an explicit formula or in the form of:

$$cf(sse, arg_1, arg_2 \ldots, arg_n) = f_1(f_2(\ldots(f_n(sse, arg_n), \ldots, arg_2), arg_1)$$

where $f_1, \ldots, f_n$ are basic or compound filters and arg_i represents the argument list of the i-th filter.

The following compound filters are useful in both retrieval and update processes.

Candidate State Elements: (CSE) is the set of all state elements in time t as observed in time t_o by a viewpoint v, that is, all the state elements, as known in t_o by v, such that t is included in their validity interval. Since knowledge is usually referred in the context of viewpoint and observation time, this filter is essential in many queries. *CSE(sse, t, t_o, v)=OV(RT(OT(sse,t_o),t,v)))*. Example:
CSE(sse, Nov 1991, Nov 1991, Goldilox)=$\{s3\}$.

Multi Accessible Filter: (MAF) is the set of all state elements accessible to several viewpoints. Let $vl = \{v_1, \ldots, v_n\}$ be a sequence of viewpoints.
MAF(sse, vl)=$\bigcap_{v \in vl} OV(sse, v)$. Example:
MAF(sse, $\{LRR, Goldilox\}) = \{s3\}$.
MAF can be used to check the knowledge coordination of a group of competitors and discover de-facto cartels.

Temporal Average: (TAG) is a filter that creates a new state element whose value is calculated by exponential smoothing of the historical state elements[2]. TAG can be used for the calculation of the *Market-Constant* as defined in Section 1.

TAG(sse, coef, $t_s, t_e) =$
$coef * [V(t_m) + (1 - coef) * V(t_{m-1}) + (1 - coef)^2 * V(t_{m-2}) + \ldots + (1 - coef)^{m-2} *$
$V(t_2)] +$
$(1 - coef)^{m-1} * V(t_1)$,
where $t_1 = t_s, t_m = t_e, V(t_i) = value(q_i \in PA(sse) \mid t \in t_v(q_i))^3$.
Example: TAG(sse, 0.5, Sep 1991, November 1991)=289.

The filters mentioned above are only a subset of the filters needed for retrieval and update operations and often new filters need to be defined. A new filter is defined either by an explicit definition of the fo and the arg (e.g., TAG) or by using existing filters (e.g., CSE). In the latter case, the definition of fo and arg is implied by the predefined filters.

3.2. *Improving retrieval time of highly-complex knowledge*

In this section we observe the special properties that should be considered while optimizing queries in this model.

3.2.1. The database level

The database model defined in this paper has two outstanding properties:

1. The database is an *append only* one. Due to this property, we can maintain in a single database all its previous versions. This is done using the transaction time attached to each state element in the database and the observation time filter that returns only the relevant knowledge as of a certain time point.

2. For each data item in the database, in order to persist its dimensional knowledge values, the required storage space is considerably larger than the space required for representing a data item in conventional database.

The combination of these two properties implies that the type of storage media should support high quantities of data, fast retrieval capabilities and no in-place update capabilities. An optical storage media may be used to satisfy these requirements.

3.2.2. The object level

The dimensional variables have the following properties:

1. Due to the append only property, the data items are ordered in the database according to their insert time (t_x).

2. Some of the dimensional variables (c_s in the confidence dimension and w in the epistemic dimension) usually have a small domain set of values.

3. Although c_v has an infinite domain set of values, for practical purposes it can be transformed into a small set of ranges. The granularity of these sets is application dependent.

4. We assume that there is a close relationship between t_x and t_d, that is, $t_x - t_d < M$, where M is an application dependent constant.

These properties enable developing specific optimization mechanisms that cannot be applied on general data. For example: the values of dimensional variables such as c_s and w can be pointed instead of actually being written in the extensions of all the relevant state elements.

3.2.3. The filter level

Optimization of filters compiler can be performed by replacing the order of filters' evaluation. For example, consider the CSE filter:

$$CSE(sse, t, t_o, v) = OV(RT(OT(sse, t_o), t, v)))$$

In the CSE case, the database first filters out state elements based on the OT filter. This is appropriate, since the data are physically ordered according to the t_x. However, if we define a new filter, CSE':

$$CSE'(sse, t_o, t, v) = OV(OT(RT(sse, t), t_o, v)))$$

we cannot rely on the physical order.

In this case, since the result of the CSE' filter is equal to the result of the CSE filter, we can use the CSE filter instead of the CSE'. This type of filter is called a *commutative filter*:

Commutative filter $cf(sse, arg_1, \ldots, arg_n) = f_1(\ldots(f_n(sse, arg_n), \ldots, arg_1)$ is a filter that for each two atomic filters $f_i, f_j \in cf$: $f_i(f_j(sse, arg_j), arg_i) = f_j(f_i(sse, arg_i), arg_j)$

Not all filters are commutative. For example, VSE is a compound filter, used in the update process, to determine the preferred state element among several state elements:

$$VSE(sse, t, t_o, v) = CP(TP(CSE(sse, t, t_o, v)))$$

where TP (Temporal Preference) is an atomic filter that chooses the state element with the higher t_d and CP (Confidence Preference) is an atomic filter that chooses the state element with the higher confidence value.

Based on Figure 5:
CSE(sse, Nov 1991, Dec 1991, Snowhite)=s3, s4, s5, s6
VSE(sse, Nov 1991, Dec 1991, Snowhite)=s6.

Note that VSE is not commutative, since TP(CP(CSE(sse, Nov 1991, Dec 1991, Snowhite)))=s4. VSE is a *semi-commutative filter*:

Semi-commutative filter
$cf(sse, arg_1 \ldots, arg_n) = f_1(\ldots(f_n(sse, arg_n), \ldots, arg_1)$
is a filter that for some $k < n$, $f_k(\ldots(f_n(sse, arg_n), \ldots, arg_1)$ is a commutative filter.

In this example, the VSE is semi-commutative filter since CSE is a commutative filter.

4. Conclusion

This paper has presented a unified framework for representing a highly-complex knowledge in a database. Such a model extends the capabilities of database technology to cope with applications that use derivations rules, temporal information, and knowledge from multiple sources with different measures of quality and epistemic knowledge. The model enables the support of features which we believe are essential for the next generation of decision support and decision management systems. Notable features that are supported by this model, and are not easily supported by contemporary models are:

1. The ability to "go back to the past" and reason about the information that was available to a decision maker at that time.

2. The ability to issue retroactive updates, and get an automatic propagation of the consequences over the temporal space.

3. The ability to group data items according to different criteria, such as epistemic and quality.

4. The ability to evaluate data using intra-dimensional criteria as easy as inter-dimensional criteria in the retrieval process.

The number of applications using these features increase with the introduction of collaborative or competitive decision support systems, and intelligent auditing systems. These applications are currently implemented using conventional technologies that require the user to use self defined procedures to achieve these functionalities. Furthermore, in many cases the application's functionality is compromised due to conceptual or technical limitations.

The introduction of this framework is only one step in a long way. Further extensions to this research include:

1. Extending the model to support adaptable extension, by eliminating some dimensions and defining new ones (such as space).

2. Devising a complete query language and inference mechanism using the data and the dimensional variable.

3. Extending the active temporal database update algorithm to support multi-dimensional update.

4. Dealing with performance issues. This includes a variety of optimization problems, such as: storage management, detecting possible cases of incremental updates, using flexible transaction protocol to allow asynchronous subtransactions and query optimization based on the observations given in Section 3.2.

Acknowledgements

The case study was adjusted with the help of Dov Monderer.
The work of the second author was supported by the fund for the promotion of research at the Technion.
The work of the third author was supported by NSF Grant Number IRI-9000619 and by the Applied Mathematical Sciences Research Program of the Office of Energy, U.S. Department of Energy under Contract DE-AC03-76SF00098.

Notes

1. The GENERATE operation creates virtual state elements, for query uses only. Modification of the database is done through the update mechanism.
2. Exponential smoothing is a weighing method giving exponentially higher weights to more recent periods.
3. The result of PA assigns each t to a unique q_i.

References

Abiteboul S.- Update, The New Frontier, *Lecture Notes on Computer Science, 326*, pp. 1-18, 1988.

Ariav G.- A Temporally Oriented Data Model, *ACM Transactions on Database Systems, 11(4)*, pp. 499-527, Dec 1986.

Bonoma T.V.- Case Research in Marketing: Opportunities, Problems and a Process, *Journal of Marketing Research, 22*, pp. 199-208, 1985.

Chandra R., Segev A., - Managing Temporal Financial Data in Extensible Databases, Proceedings of the *19th International Conference on Very Large Databases*, Dublin, Ireland, Aug 1993.

Chandra R., Segev A., Stonebraker M.- Implementing Calendars and Temporal Rules in Next Generation Databases, Proceedings of the *IEEE Conference on Data Engineering*, Houston, Texas, February 1994.

Chankong V., Haimes Y.Y.- Multiobjective Decision Making: Theory and Methodology, *New York, N.Y.: Elsevier Science*, 1983.

Clifford J., Crocker A. - The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans, *Proc. International Conference on Data Engineering*, pp. 528-537, Feb 1987.

Cournot A. - Researches into the Mathematical Principles of the Theory of Wealth (ed. N. Bacon), *Macmillan, New York*, 1897.

Dayal U., Buchmann A.P., McCarthy D.R. - Rules Are Objects Too: A Knowledge Model for an Active Object-Oriented Database Model, *Proc. 2nd Int'l Workshop on Object-Oriented Databases*, pp. 140-149, Sep 1988.

Edara M.L., Gadia S.K.- Updates and Incremental Recomputation of Active Relational Expressions in Temporal Databases, *Proceedings of the International Workshop on an Infrastructure for Temporal Database, Arlington,TX*, June 1993.

Etzion O. - PARDES-A Data-Driven Oriented Active Database Model, *SIGMOD RECORD, 22(1)*, pp. 7-14, Mar 1993a.

Etzion O., Gal A., Segev A - Decision Support using Temporal Active Database, *Technion - Israel Institute of Technology, Technical Report ISE-TR-93-5*, Nov 1993b.

Etzion O.,Gal A., Segev A - Retroactive and Proactive Database Processing. *Proceed. RIDE- ADS 1994.*

Gadia S.K.- The Role of Temporal Elements in Temporal Databases, *Data Engineering Bulletin, 7*, pp. 197-203, 1988.

Gadia S.K. - Parametric Databases: Seamless Integration of Spatial, Temporal, Belief and Ordinary Data , *SIGMOD RECORD, 22(1)*, pp. 15-20, Mar 1993.

Gal A., Etzion O.- Updating Databases with an Invariant Language, *Technion-Israel Institute of Technology, Technical Report ISE-TR-92-4*, Feb 1992.

Gardenfors P., Makinson D.- Revisions of Knowledge Systems Using Epistemic Entrenchment, *Proc. 2nd Conference on Theoretical Reasoning About Knowledge*, 1988.

Jang Y., Kon H.B., Wang R.Y.- A Knowledge-Based Approach to Assisting in Data Quality Judgment, *Proc. WITS '92*, pp. 179-188 , Sep 1992.

Johnson J.R. - Hallmark's Formula for Quality, *Datamation*, pp. 119-122, 1990.

Keeney R.L., Raiffa H.- Decision with Multiple Objectives: Preferences and Value Tradeoff, *New York: John Wiley & Son*, 1976.

Lanzelotte R.S.G., Valduriez P., Zait M.- Optimizing of Object-Oriented Recursive Queries Using Cost-Controlled Strategies, *Proc ACM SIGMOD 1992*, pp. 256-265, June 1992.

Levesque H.J.- All I Know: A Study in Autoepistemic Logic, *University of Toronto, Technical Report KRR-TR-89-3*, Jan 1989.

McCarthy D., Dayal U.- The Architecture of an Active Data Base Management System, *Proc. ACM SIGMOD 89*, pp. 215-224 , June 1989.

Navathe S.B., Ahmed R.- A Temporal Relational Model and Query Language, *Information Sciences, 49*, pp. 147-175, 1989.

Poole D.- A Logical Framework for Default Reasoning, *Artificial Intelligence, Vol. 36*, 1988.

Qian X., Wiederhold G.- Incremental Computations of Active Relational Expressions, *IEEE Transactions on Knowledge and Data Engineering, 3(3)*, pp. 337-341, 1991.

Rose E., Segev A.- TOODM-a Temporal, Object-Oriented Data Model with Temporal Constraints, *Proc. International Conference on the Entity-Relationship Approach, San Mateo, California*, pp. 205-229, 1991.

Segev A., Shoshani A.- Logical Modeling of Temporal Data, *Proc. ACM SIGMOD 87*, pp. 454-466, May 1987.

Shoshani A., Kawagoe K.- Temporal Data Management, *Proc VLDB 86*, pp. 79-88, Aug 1986.

Sistla A.P., Wolfson O.- Specification and Management of Temporal Triggers in Active Databases, *Technical Report, Univ of Il at Chicago*, 1993.

Snodgrass R., Ahn I.- Temporal Databases, *IEEE Computer 19*, pp. 35-42, Sep 1986.

Snodgrass R.- The Temporal Query Language TQUEL, *ACM Transactions on Database Systems*, pp. 247-298 , June 1987.

Stonebraker M., Kemnits G.- The POSTGRES Next-Generation Database Management System, *CACM, 34(10)*, pp. 78-93, Oct 1991.

Su S.Y.W, Lam H., Krishnamurthy V. - An Object-Oriented Semantic Association Model (OSAM*), *in: S.T. Kumera et al. (eds)-AI: Manufacturing Theory and Practice, Chap. 17, Norcross, GA*, 1989.

Su S.Y.W., Chen H.M.- A Temporal Knowledge Representation Model OSAM*/T and its Query Language OQL/T, *Proc VLDB*, 1991.

Tirole J.- The Theory of Industrial Organization, *the MIT press*, 1989.

Wiederhold G., Jajodia S., Litwin W. - Dealing with Granularity of Time in Temporal Databases, *Lecture Notes in Computer Science, 498, (R. Anderson et al. eds.), Springer-Verlag*, pp. 124-140, 1991.

Wuu G., Dayal G.U.- A Uniform Model for Temporal Object-Oriented Databases, *Proc. International Conference on Data Engineering*, 1992.