ANITA GUDELJ, M.Sc.
E-mail: anita@pfst.hr
MAJA KRCUM, M.Sc.
E-mail: mkrcum@pfst.hr
University of Split, Faculty of Maritime Studies
Zrinsko-Frankopanska 38, HR-21000 Split,
Republic of Croatia

Section: Traffic Management
Review
Accepted: Dec. 15, 2005
Approved: Feb. 21, 2006

MANAGING TEMPORAL KNOWLEDGE IN PORT MANAGEMENT SYSTEMS

ABSTRACT

Large ports need to deal with a number of disparate activities: the movement of ships, containers and other cargo, the loading and unloading of ships and containers, customs activities. As well as human resources, anchorages, channels, lighters, tugs, berths, warehouse and other storage spaces have to be allocated and released. The efficient management of a port involves managing these activities and resources, managing the flows of money involved between the agents providing and using these resources, and providing management information. Many information systems will be involved.

Many applications have to deal with a large amount of data which not only represent the perceived state of the real world at present, but also past and/or future states. These applications are not served adequately by today's computer management and database systems. In particular, deletions and updates in such systems have destructive semantics. This means that previous database contents (representing previous perceived states of the real world) cannot be accessed anymore.

A review of how define temporal data models, based on generalizing a non-temporal data model in to a temporal one to improve port management is presented. This paper describes a practical experiment which supports managing temporal data along with the corresponding prototype implementations.

KEY WORDS

strategies, temporal knowledge, temporal database, deductive database

1. INTRODUCTION

Most database applications are temporal in nature, e. g., financial applications such as portfolio management, accounting (stock values, exchange rates, sales, etc.), record-keeping applications such as personnel, medical-record, and inventory management, scheduling applications such as ship, hotel reservations and project management, and scientific applications such as weather monitoring.

A temporal database records time-varying information. They assume that all data values are associated with timestamps which represent the time during which this data value was valid in the real world and/or the time during which this data value was recorded in the database. Time-varying data are of great importance in many database applications, such as data mining.

Temporal databases are often of huge size because of the large quantity of historical data. Thus, mining strategies is not a trivial work.

The work presented in this paper generalizes temporal extensions proposed for relational databases to deductive databases. Our approach also lends itself nicely to an efficient implementation of a deductive database system which provides all typical database services such as persistence, concurrency control, recovery etc.

On a practical level, our implementation of temporal deductive database relies on a translation of one relation database to temporal, using AmsiProlog, which is based on $Datalog_{1S}$. This allows a concise formulation of most types of temporal queries which were executed.

In the following section, we outline the time basis features of Port Management Systems. In section 3, we outline the formal framework for temporal databases, in section 4 we formalize the syntax and the semantics of $Datalog_{1S}$ (temporal deductive databases) and, in section 5, we survey some applications that have been considered in the context of port management system. Finally, section 6 summarizes the contributions of our work.

2. PORT MANAGEMENT SYSTEM

2.1. Financing of port management bodies

Most port management bodies form bureaus of local governments. The general accounts of the local public entity budget for major port construction projects and the special port development budgets in-

clude items for the management, operation, and construction of functional facilities.

Both of these budgets are based on cash accounting, not accrual (corporate accounting). The underlying idea of this practice is that the goal of port management and operations is for ports to serve as public establishments available (offering services) to the public, not to turn a profit.

Port management bodies assess such fees as port dues and fees for use of port facilities (including quay wall usage fees and cargo handling equipment usage fees). These fees are determined on a cost accounting basis and set forth by port management bodies through regulations. Unlike the tonnage tax and the special tonnage tax, port dues are levied on all vessels in exchange for the use of the port as a whole; the port management body levies these dues in accordance with enacted regulations. Port dues may be calculated and assessed based on the expenses necessary for managing the water area facilities (excluding anchorages), outlying facilities and environmental development facilities for which it is difficult to recoup the expenses incurred for offering services from fees. Note that port dues are those fees authorized by the government (the Minister of Land, Infrastructure and Transport) for specially designated major ports.

To promote the use of ports and harbors, some port management bodies offer incentive systems whereby fees for use of port facilities and port dues are reduced.

2.2. Port Management body functions and duties

Port management bodies consist of proprietary--type organizations. In addition to building, maintaining, and managing port facilities (navigation channels, breakwaters and other basic facilities and quay walls, cargo handling and other functional facilities), port management bodies formulate policies for basic development plans in consideration of the development of the inland regions.

Port facilities (functional facilities) are leased to the private sector under the management of the port management bodies. Actual operation (port transport, storage, land-based transport, etc.) is entrusted to the private sector, as stipulated by the relevant laws and regulations.

The Port and Harbor Law forbids port management bodies to interfere with the ventures of the private sector or to conduct any business that competes with the private sector. In the course of managing and operating the port, they are also forbidden to make any prejudicial distinctions in their treatment of persons or entities connected with the port. Systematic guarantees of operation by the private sector are thought to provide greater efficiency than direct operation by local government.

3. BASICS OF TEMPORAL DATABASE SYSTEMS

Conventional database systems are designed to capture current data. As soon as new data values become known, existing data values are replaced. Although, such databases serve many applications well, they are insufficient for applications which have to consider past and (or) possible future states. Consequently, interest in research concerning the study of time in databases has been growing steadily over the past few years. Many researchers have incorporated time into conventional databases using various schemes providing different capabilities for handling temporal information. In particular, work has been done on adding temporal information to conventional databases to turn them into temporal databases. However, considerable research efforts were put into the development of temporal database systems.

Now, we are going to survey these efforts to establish a common base for our work. The first part illustrates different possibilities of how time can be associated with information. Two approaches have become popular: attribute timestamping and tuple timestamping. While attribute timestamping leads to a more compact representation, tuple timestamping is advantageous in terms of simplicity.

The second part addresses topics related to the different notions of time in a database system: user-defined time, transaction time and valid time. The different notions of time allow the distinction of different types of temporal databases: snapshot relations, transaction time relations, valid time relations, and bitemporal relations, which are described in [7].

3.1. Associating time with information

Temporal Data Model is an extension of relational model by adding temporal attributes to each relation. In that way, it becomes possible to store different database states. Changes are viewed as additions to the information stored in the database.

One approach is that a temporal database may timestamp entities with time periods. Each tuple contains time flag which expresses when it was, is or will be valid. Let us assume there is the table

No	Name	Year_birth	Salary	ValidTime
100	Ante	1953	2000	[1981/3 - 1991/8)
101	Tonko	1934	3000	[1985/7 - ∞)

On January 1, 1997, each employee was given a salary rise of 250 KN. The result is the following table.

Table 1 - Attribute timestamping does not eliminate all redundancies

No	Name	Year_birth	Salary	ValidTime
100	Ante	1953	2000	[1981/3 - 1991/8)
101	Tonko	1934	3000	[1985/7 - 1997/1)
101	Tonko	1934	3250	[1997/1 - ∞)

Although only one attributes value changes, an entire tuple has to be inserted. However, apart from the changed attribute value and differing time flags, all information in the new tuple is identical!

The main disadvantage of tuple timestamping is the fact that information about a real world entity is spread over several tuples. Each tuple represents a state the real world entity in during a certain period of time. Tuple timestamping, additionally, introduce data redundancy.

Another approach is timestamping of the property values (attributes) of the entities. Now it is possible to overcome the disadvantage of data redundancy introduced when applying tuple timestamping. In data models, each attribute value is timestamped. Values in a tuple, which are not affected by modification, do not have to be repeated. So, the history of values is stored separately for each attribute. For example, we can store the data about employee in a single tuple.

No	Name	Year_birth	Salary
[1985-∞) 101	[1985-∞) Tonko	[1985-∞) 1934	[1985-1997) 3000 [1997-∞) 3250

3.2. Notions of time

Snodgrass and Ahn [8] describe a classification of databases depending on their ability to represent temporal information. They identify three notions of time: user defined time, transaction time and valid time. Depending on the respective application, all of these time dimensions or just some of them are necessary.

Valid time denotes the time period during which a fact is true with respect to the real word. Transaction time is the time period during which a fact is stored in the database. These two time periods do not have to be the same for a single fact.

No	Name	Year_birth	Salary	ValidTimeStart	ValidTimeStop
100	Ante	1953	2000	1981	1991
100	Tonko	1934	3000	1985	∞
101	Ana	1957	3000	1986	∞
102	Edo	1960	3200	1989	1998

The table contains the history of employees with respect to the real world. The attributes **ValidTimeStart** and **ValidTimeStop** denote a time interval, which is closed at the lower and opened at the upper bound. The upper bound ∞ is used to represent the special values ***until changed.***

Now it is possible to store the information about the past states. Thus, we see that Edo was employed from 1989 until 1998. In the corresponding non-temporal table this information was deleted when Edo left the company.

Databases which represent only the latest snapshot of the world being modeled are called snapshot databases. All conventional databases come under this category. Databases which represent transaction time alone and therefore treat valid time and transaction time as identical are called rollback databases since it is possible to rollback to a past state of the database and pose a query with respect to that state. The problem with representing transaction time alone is that a history of the database activities is recorded (since every database state is stored, in effect), rather than the history of the world being modeled. Thus it is not possible to make proactive/retroactive updates and errors in a past state cannot be corrected.

Databases which store the history of the real world as is best known are called historical databases. These databases have the concept of valid time alone but it is possible to make changes to the history of each tuple. Finally, databases which store all the past history as is best known at every state of the database are called temporal databases.

4. TEMPORAL DEDUCTIVE DATABASE

In the classical temporal database approach, temporal extensions are represented explicitly, that is, by adding a set of values which represent the time during which this data value was valid in the real world and/or the time during which this data value was recorded in the database. This representation is usable only if the number of fact is finite and, in practice, if the database is not too large. An alternative is to use an implicit representation from which the extensions can be computed. We will show how such implicit representation of historical data can be used in financial application in port management system.

Temporal databases are especially useful when updates have retrospective effect. For example, in [13], Sergot et al. discuss the representation of the British Nationality Act as a logic program. A database might be used to record the details of a person. The logic program can then be used to deduce whether or not that person is, according to the act, a British citizen. As the person's details change, their status can change accordingly. It is possible for a person to become a

British citizen and for this status to have effect from the date of their birth. A temporal database preserves the distinction between what was previously considered to be their status at the time of their birth and what is now considered to be their status at that time. This capability is achieved by an explicit storage of the time at which information becomes available. In the case of databases, this time is referred to as the transaction time.

In this section, we define this implicit representation by using $Datalog_{1S}$ [3], temporal deductive language. We consider a Horn clause deductive language which consists of a finite set of temporal facts and temporal rules. A temporal fact is a temporal ground atom while a temporal rule has the form H← B where H is an atom and B is a temporal formula. We call H the head and B the body of the rule. A temporal query has the form Q, where Q is a temporal formula.

This deductive language is used for defining the ***intensional*** database (IDB) relations. When we consider the evaluation of this deductive language, we will consider it in conjunction with the generalized database formalism used for providing the ***extensional*** database (EDB) relations.

4.1. $Datalog_{1S}$

Syntax

We assume that language contains infinitely many variable, function and predicate symbols. Logic symbols are Boolean operators ∧ (conjunction), ∨ (disjunction), ¬ (negation), → (*implication*) and *the quantifiers* (∀, ∃).

Constants are either strings or numbers (a, b, true). *An identifier* is an arbitrary character sequence starting with a letter and containing letters, digits and underscores only. *Variables* are denoted by identifiers starting with a capital letter.

Predicate symbols are identifiers starting with a lower case letter. Some predicates symbols are predefined, called built-in, namely the usual comparison operators =, >, <, ≥, ≤, ≠. *A term* is either a constant or a variable. Further, if $t_1, t_2, \ldots, t_n$ are terms and f is a n-ary function symbol then $f(t_1, t_2, \ldots, t_n)$ is also a term.

Atom is formula where p is predicate and each t_i is a term. A term or an atom is said to be ***ground*** if it is variable free. A ***fact*** is ground atom. A ***formula*** is defined recursively. First, every atom is a formula. Then, if F_1 and F_2 are formulas, x is a variable then $F_1 \wedge F_2$, $F_1 \vee F_2$, $\neg F_1$, $F_1 \rightarrow F_2$, $\exists X F_1$ i $\forall X F_1$ and are formulas as well. *A* ***database*** is finite set of facts.

A rule has the form $H \leftarrow B$ where the atom H is called the head and the formula B is called the body of the rule.

A ***logical program*** is finite set of rules together with database. *A* ***goal*** is a formula written as $\leftarrow B_1, \ldots, B_m$. *A* ***query*** is formula built from finite set of rules together with a goal.

We distinguish between *intensional database* (IBP) *and extensional* database (EBP) predicate symbols, which are built from *intensional/extensional atoms*. *Intensional* atoms are the only ones that can be appear in the head and in the body of clauses. *Extensional atoms* appear only in the body of clauses..

Datalog is a logical program in which the only terms are constants or variables. ***$Datalog_{1S}$*** is an extension of Datalog and it involves temporal and non-temporal variables and constants. We assume that there is only one temporal constant 0. A predicate symbol is either temporal or non-temporal. Temporal predicates have a single distinguished parameter, called temporal, in addition to the usual data parameters. Temporal parameters are temporal terms.

A ***temporal term*** is defined inductively:

1. the temporal constant is temporal term;
2. a temporal variable is temporal term;
3. if v temporal term, then v+1 is a temporal term;
4. there are no other temporal terms.

Example 1 If T is variable, then 0, T, T+1 are temporal terms.

We will write $\boldsymbol{k}$

$$\underbrace{(\cdots((0+1)+1)\cdot+1)}_{k-times}$$

and $\boldsymbol{T+k}$ instead of

$$\underbrace{(\cdots((T+1)+1)\cdot+1)}_{k-times}.$$

The integer $\boldsymbol{k}$ corresponds to the temporal term $\boldsymbol{k}$ in a natural way.

A *temporal deductive database* consists of a finite set of temporal facts and temporal rules. A temporal fact is a temporal ground atom while a temporal rule has the form H← B where H is an atom and B is a temporal formula. A *temporal query* has the form $Q(x_1, \cdots, x_k)$, where Q is a temporal formula built from temporal and non-temporal atoms, logical connectives and quantifiers.

Example 2 Consider the following rules for scheduling backups in one distributed ship system

backups (T+24, X) ← ***backups*** (T, X)
backups (T, Y) ← ***dependent*** (X, Y), ***backups*** (T, X)
...

The first rule defines that the backup on machine would be taken every 24 hours. The second rule requires that the backup should be taken simultaneously on all dependent machines.

Semantics

The semantics of $Datalog_{1S}$ is given with respect to two-sorted domains. Temporal terms are interpreted

over the set of integers, whereas the data terms are interpreted over the set of constants.

The declarative semantics of deductive program considered together with an extensional database is captured by its minimal model which can be obtained by the intersection of all its minimal models. Minimal model contains all the ground atoms that are logical consequences of the program and it is usually infinite when the program contains temporal terms. .

However, since *Datalog* does not allow function symbols in programs, every minimal model is finite. It uses *Bottom-up* algorithm for finding a minimal model. This algorithm starts from a finite set of facts and from it the algorithm derives new facts using deductive Horn rules. The algorithm finishes when no new facts are derived.

Example 3 The following rules schedule the arriving destination.

IBP	*arrive (T, X) ← first_arrive(T, X)* *arrive(T+1, Y) ← follows(X, Y), arrive(T, X)*

T: "*if the customer X arrives the destination in given time T and Y is the customer which follows after the customer X and Y arrives the destination in time T+1*".

Assume that database contains the following rules:

EBP	*first_arrive (0, AMERICAN HAWAII CRUISES)* *follows (AMERICAN HAWAII CRUISES, ARCALIA SHIPPING)* *follows (ARCALIA SHIPPING, AMERICAN HAWAII CRUISES).*

The rules can be used to derive the following infinitely many facts of database.

arrive (0, AMERICAN HAWAII CRUISES)
arrive (1, ARCALIA SHIPPING)
arrive (2, AMERICAN HAWAII CRUISES)
arrive (3, ARCALIA SHIPPING)
.... where 1=1+0, 2=(0+1)+1, ...

On the practical level, we are interested only in finite part of the given database. In given time, all predicates represent finite relations.

5. EXPERIMENTAL RESULTS

In this section we will apply the proposed framework for temporal deductive database to one existing non-temporal database model for Transportation and shipment. It is intended to support the development of port management systems. The experiments were done on a PC running Windows XP. Figure 1 presents non-temporal data model [14] for transportation and shipment in financial applications of port management system.

5.1. Quering temporal database

In this section, we will describe a few examples of querying on temporal data. We used AmziProlog for creating database and querying data.

In temporal deductive database, EBP contains the relations *Transportation_Agency* (time, agency_id, agency_name, other_agency_details), *Customers* (time, customer_id, date_become_customer, penalty_charge, other_customer_details), *Shipment_Order* (time, order_id, agency_id, customer_id, oder_status_code, shipping_from_adress_id, shipping_to_address_id, order_received_date). The logical time is interpreted as one month. The month "0" is time when the database is building.

Let us consider a part of database containing the following facts.

first_Transportation_Agency (0, A101, ***'Wilson Shipping'***, ' ').
first_Transpottation_Agency (0, A102, 'Split Shipping', ' ').
first_Transporttation_Agency (0, A103, ' ***Maritime Agency Intorg***, ' ')
...
first_Customers (0, C101, ' AMERICAN HAWAII CRUISES', 07-Feb-1999, ' ', ' ').
first_Customers (0, C102, ' DOLPHIN HELLAS SHIPPING', 21-Jan-2000, ' ', ' ').
first_Customers (0, C103, ' AMERICAN HAWAII CRUISES', 28-May-2000, ' ', ' ').
...
first_Shipment_Order (0, 001, A101, C101, O1, line1#, line3#, 10-Feb-1999, ' ', ' ', ' ').
first_Shipment_Order (0, 002, A101, C103, O1, line2#, line3#, 01-Feb-2000, ' ', ' ', ' ').
first_Shipment_Order (0, 003, A103, C102, O1, line1#, line2#, 10-Sep-2000, ' ', ' ', ' ').

IMP contains the following rules:

Transportation_Agency (Nt, A_id, Aname, _): -
first_Transportation_Agency (1, A_id, Aname, _).
Transportation_Agency (Nt , A_id, Aname, _): - Nt > 0, Ntt is Nt-1,
first_Transportation_Agency (Ntt, A_id, Aname, _).
Customers (Nt, C_id, _, penalty_charge, _) : -
first_Customers (Nt, C_id, _, penalty_charge, _) .
Customers (Ntt , C_id, _, P_charge, _) : - Ntt > 0, Nt is Ntt-1, *Customers* (Nt, C_id, _, P_charge, _).
Shipment_Order (Nt, order_id, A_id, C_id, oder_Scode, _, _, _) : -
first_Shipment_Order(Nt, order_id, A_id, C_id, oder_Scode, _, _, _).
Shipment_Order (Ntt, order_id, A_id, C_id, oder_Scode, _, _, _) : - Ntt > 0, Nt is Ntt - 1,
Shipment_Order (Nt, order_id, A_id, C_id, oder_Scode, _, _, _).

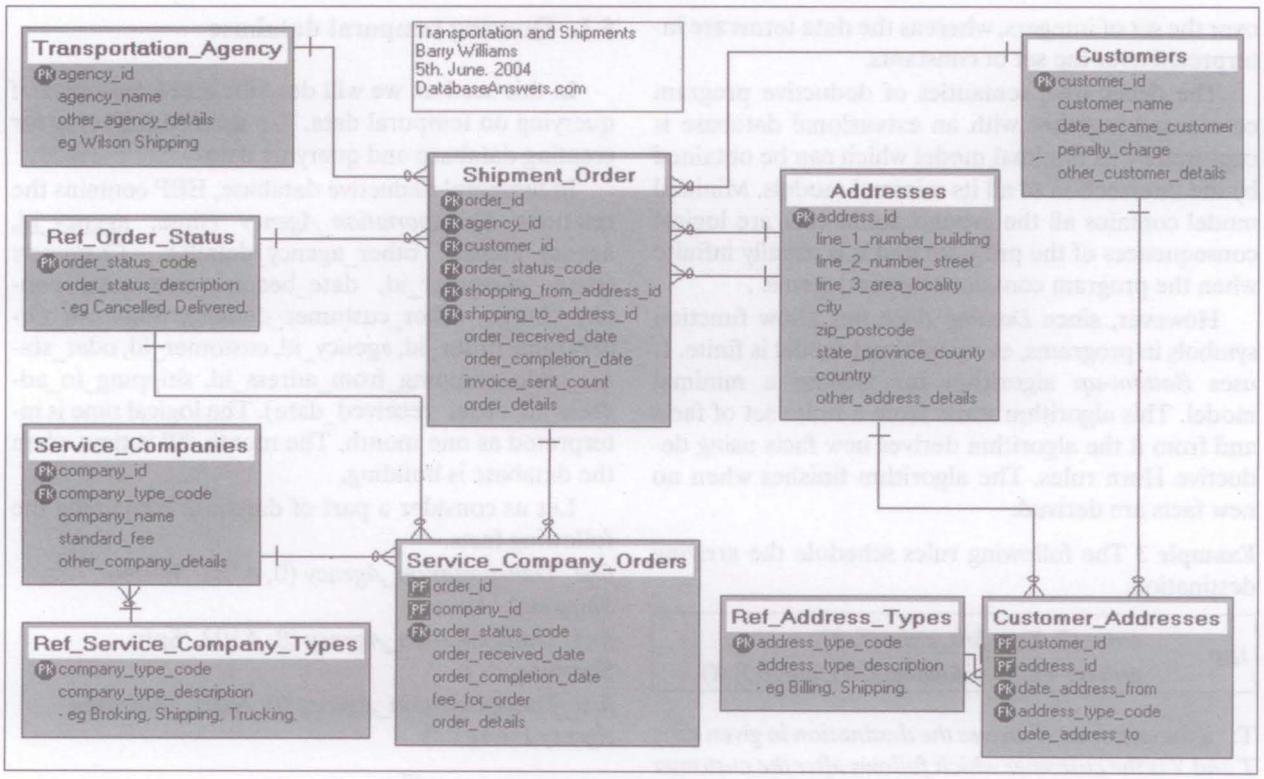

Figure 1 - Non-temporal ER diagram

Also, this framework allows defining business rules. Now we are introducing just three rules.

(R1) The Customer has one month to pay the charge, after the Company send "Charge Invoice" them. If the customer doesn't pay the charge, the Company will send a penalty charge to the Customer.

Pay_charge(T, X, A#, C#, 0) : - *Shipment_Order* (T, X, A#, C#, _, _, _, _, invoice_sent_count).

Pay_charge(*t+1*, X, A#, C#, y) : - *Pay_charge*(*t*, X, A#, C#, y).

Sent_penalty(Tt, A#, C#, penalty_charge) : - *Shipment_Orde* (T, X, A#, C#, _, _, _, _, invoice_sent_count), Tt>T+2, *Pay_charge*(Tt, X, A#, C#, 0).

(R2) For every month, the Company will calculate the cost of each shipment by summing the charge of shipment service companies.

Shipment_Cost(T, X, Tcost) : - *Shipment_Order*(T, X, _, _, _, _, _, _, _), Tcost= sum{fee : *Service_Company_Orders* (T, X, _, _, _, _, fee; _),

(R3) If the Customers pay the charges, the Company will calculate the profit for one month by using the charge minus the cost fee for each shipment.

Profit (T, X, agency_id, P) : - *Shipment_Order* (T, X, agency_id, _, _, _, _, _), fee= Service_Company_Orders (T, X, _, _, _, _, fee; _), charge=*Pay_charge*(*t*, X, agency_id , C#, charge) ∧ charge>0, P = charge – fee).

Temporal selection

Example 4 Let us assume that we want to identify the profit month 5. The answer should be the same whatever the query is asked. The query follows and the value of variable P in the given time is the answer.
← *Profit*(5, _, A101, P).

Temporal join

Temporal data which are applied to different time moments can be combined together using temporal operators.

Example 5 *For the query " Identify all shipment orders for one agency for the last three months ", the following rule is corresponded:*
← *Shipment_Order* (T, order_id, A_id, C_id, _, _, _, _) ,
Shipment_Order (T+1, order_id, A_id, C_id, _, _, _), *Shipment_Order* (T+2, order_id, A_id, C_id, _, _, _, _) .

Explicit time manipulation

In temporal deductive databases, it is natural to ask queries to determine when the case happened.

Example 6 If we want to know the profit of the Company A101 for time period from month 0 to month3.

The query should be defined as follows:
average_Profit(T, agency_id, W, Y) ← T>0, Tt is T-1, Profit(T, _, agency_id, P),
write (Tt : P), nl,
XX is P+W,

average_Profit(Tt, agency_id, XX, Y).
and goal
← average_Profit(3, A101, P, Y).

6. CONCLUSIONS

In this paper, we have pointed out the framework for representing the knowledge rules which make reference to temporal events and the management of histories of these rules in a knowledge base management system. The formalization is done in the Horn clause and it is executable as a logic program. We have restricted our attention to ground unit clauses as the relationships that are derivable from a description of events. The methods for updating, retrieving and evaluating temporal rules have also been presented. It is the authors' belief that the modeling and language concepts and implementation considerations presented in this paper will contribute to the realization of future active knowledge base in port management systems.

Mr. sc. **ANITA GUDELJ**
E-mail: anita@pfst.hr
Mr. sc. **MAJA KRCUM**
E-mail: mkrcum@pfst.hr
Sveučilište u Splitu, Pomorski fakultet
Zrinsko–Frankopanska 38, 21000 Split, Republika Hrvatska

SAŽETAK

UPRAVLJANJE TEMPORALNIM ZNANJEM U LUČKIM SUSTAVIMA UPRAVLJANJA

Velike luke moraju obavljati mnoge različite aktivnosti: kretanje brodova, kontejnera i drugog tereta, utovar i istovar brodova i kontejnera, carinske formalnosti. Kao i ljudski resursi, tako se i sidrenja, kanali, maune, tegljači, privezišta, skladišta i ostali skladišni prostori moraju dodijeliti i osloboditi. Učinkovito upravljanje lukom podrazumijeva upravljanje ovim aktivnostima i resursima, upravljanje novčanim tokom do kojeg dolazi između agenata koji pružaju i koriste te resurse, i koji pružaju informacije o upravljanju. Pri tome su uključeni mnogi sustavi informiranja.

Mnoge se primjene moraju baviti velikim količinama podataka koji ne predstavljaju samo zatečeno stanje stvarnog svijeta toga trenutka, nego također i prošla i/ili buduća stanja. Današnji računalni menadžment i sustavi baza podataka ne opslužuju u prikladnoj mjeri ove primjene. U ovakvim sustavima, brisanja i ažuriranja imaju naročito destruktivnu semantiku. To znači da se više ne može doći do prijašnjih sadržaja baza podataka (koje predstavljaju prijašnja zatečena stanja stvarnog svijeta).

U radu je predstavljen pregled načina definiranja modela privremenih podataka, na temelju poopćavanja modela ne-privremenih podataka u privremene kako bi se poboljšalo upravljanje lukom. Ovaj rad opisuje praktični eksperiment koji podržava upravljanje privremenim podacima zajedno sa odgovarajućim primjenama prototipa.

KLJUČNE RIJEČI

strategije, temporalno znanje, baza temporalnih podataka, deduktivna baza podataka

LITERATURE

[1] **M. Baudinet,, J. Chomicki, P. Wolper**, *Temporal Deductive Databases*

[2] **M. Baudinet, M. Niezettte, P. Wolper**, *On the Representation of Infinite Temporal Data and Queries*. Proc. 10th, ACM PODS, 280-290, 1991.

[3] **M. H. Böhlen**, *Managing Temporal Knowledge in Deductive Databases*, A dissertation submitted to the Swiss Federal Instituote f Technology Zurich, Department Informatik, pages 27-4, 9ETH Zürich, 1994.

[4] **M. H Böhlen** and **C. S. Jensen**, A *Seamless Integration of Time into SQL*, ACM Transactions on Database Systems, 1996.

[5] **J. Chomicki**, *Temporal Qery Language*, A Survey, *12. ACM PODS*, 1993.

[6] **J. Chomicki**. *Temporal deductive databases,* In A. Tansel, J. Clifford, S. Gadia, S. Jagodia, A. Segev, and R. Snodgrass, editors, *Temporal Databases: Theory, Design and Implementation*, pages 294-320, Benja Min/Cummings, 1993.

[7] **A. Steiner**, A *Generalization Approach To Temporal Data Model And Their Implementations*, Ph. D. Thesis, ETH Zürich, 1997.

[8] **R. Snodgrass, I. Ahn**, *Temporal Databases*, IEEE Computer, 19, No. 9, Sep. 1986, pp. 35-42

[9] **R. Snodgrass, M. Böhlen, C. Jensen** and **A. Steiner**, *Adding Valid Time to SQL/Temporal. ANSI X3H2-96-501r2, ISO/IECC JTC1/SC 21/WG 3 DBL-MAD-146r2*, 1996.

[10] **R. Snodgrass, M. Böhlen, C. Jensen** and **A. Steiner**. *Adding Transaction Time to SQL/Temporal. ANSI X3H2-96-501r2, ISO/IECC JTC1/SC 21/WG 3 DBL-MAD-146r2.*

[11] **R. Snodgrass**, *Temporal Database: Status and Research Directions*, SIGMOD RECORD, 1990.

[12] **R. Snodgrass**, *Temporal Databases,* University of North Carolina at Chapel Hill, 1986.

[13] **S. M. Sripada**, *A logical framework for temporal deductive databases*, Proceedings of the 14th VLDB Conference Los Angeles, California 1988, pp. 171-182

[14] http://www.databaseanswers.org/data_models/transportation_and_shipments/index.htm