Visual Query Operators for Temporal Databases

Sônia Fernandes*, Ulrich Schiel** and Tiziana Catarci^

*Departamento de Engenharia Elétrica, **Departamento de Sistemas e Computação
Universidade Federal da Paraíba, C.P. 10106 Campina Grande - PB, Brasil, e-mail: sonia@dsc.ufpb.br
^Dipartimento di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza"
Via Salaria, 113 - 00198 Roma, Italy, e-mail: catarci@dis.uniroma1.it

Abstract

Many recent proposals in the literature present easy-to-use query languages, based on visual representations of the database and direct manipulation mechanisms. Such proposals essentially deals with conventional query operations, whereas little effort is devoted to investigate friendly environments for querying temporal databases. Nevertheless, temporal aspects are generally considered extremely relevant in several database applications. Thus, the various users should be provided with powerful and simple query systems also exhibiting temporal features. While attempting to meet this need, we propose a visual query system for temporal databses. It adopts a diagrammatic representation of the database schema (including temporal classes and relationships), on which conventional as well as temporal visual query operators can be applied. In this paper we concentrate on the temporal operators, showing both examples of usage and formalization.

1. Introduction

As a consequence of the user-centered approach to the development of information systems, several research activities on easy-to-use visual query languages have been carried out in the past years (see [6] for a survey on visual query systems). Conventional query languages, such as SQL, are hardly aproachable by naive users, for both their intrinsic syntatical complexity and the lack of a global view of the data of interest and their interrelationships. Indeed, it is difficult to capture complex structural information through relational tables [24], where the objects have to be decomposed into many parts and reassembled through sophisticated join operations.

Similar problems also occur with textual query languages for temporal databases. Proposals exist based on SQL extensions, such as TSQL[22], HSQL[23], TSQL2[29], and object-oriented languages, such as OQL/T[30] where special clauses and predicates are added to the original language in order to manipulate the temporal aspects. However, there is little research concerning user-centered query interfaces for temporal databases in spite of numerous papers that were published considering the temporal factor relevant in several modern database aplications (see [34] for the last temporal database bibliography update).

As efforts were made to find new visual query mechanisms for accessing conventional databases, this should be done for temporal databases. Adequate conceptual schemata are needed and new visual mechanisms must be found in order to manipulate the temporal aspects. Such temporal aspects must be integrated with other components of traditional semantic modeling. The visual operators should be as powerful as the textual languages, and much easier to use in order to meet the needs of the various classes of users.

This paper presents a visual query language for historical databases (**TVQOs**, Temporal Visual Query Operators), based on a temporal extension of the Graph Model [3], [5], [7]. The language is presently being implemented on top of an object-oriented database. The Graph Model has been originally proposed in [3] as a graph-based formalism for representing and querying databases. This formalism is suitable to give a precise semantics to complex visual representations and is general enough to formalize, in principle, a database expressed in any of the most common data models.

A Graph Model DataBase (GMDB) is a triple $<g, c, m>$, where g is a Typed Graph, c is a set of Constraints, and m is an Interpretation. The schema of a database, i.e. its intensional part, is represented by the Typed Graph and the set of Constraints. The instances of a database, i.e. its extensional part, are represented by the notion of Interpretation. The querying primitives of the formalism, although constituted solely by two elementary graphical actions, namely the selection of a node and the drawing of an edge, are at least as expressive as the relational algebra. In this paper the Graph Model has been extended to model the temporal aspects of classes and relationships, and the original set of query operators has been enriched by adding two specific temporal operators: *Snapshot* and *Slice* for temporal selection and temporal projection [19]. This extension allows the user to specify even complex

0-8186-7937-9/97 $10.00

temporal queries in an easy-to-use and higly interactive visual environment.

The rest of this paper is organized as follows. Section 2 presents an example of use of the visual environment, with particular emphasis on the temporal aspects. Section 3 outlines some basic aspects of the formalization. Finally, Section 4 deals with some related work and draws the conclusions.

A preliminary version of this work has been published in [27].

2. Visually Expressing Temporal Queries

Most of the visual query interfaces for databases specify the query using direct manipulation [26] of the visual components representing classes and relationships in the conceptual schema. We follow the same approach and in this section we show some examples of query formulation through TVQOs.

2.1. Temporal Visual Query Operators

A taxonomy for temporal queries is proposed in [19]. That document contains a survey of the most common teminology used in temporal databases as result of a several debates on temporal databases. According to [19], a *temporal query* has two ortogonal components: *temporal selection* and *temporal projection*. Temporal selection is a logical condition, based on a predicate that involves the time associated with the facts. The temporal projection returns the time values associated to the data derived from temporal selection.

In [10] the possible combinations between temporal selection/projection over time and data were analysed, resulting into: *data selection/data projection*, where conditions and results apply to data values only; *temporal selection/temporal projection*, where conditions and results apply to temporal values; and *mixed selection/mixed projection*, where conditions and results apply to both data and temporal values.

In our approach, we use the *snapshot* operator for the extraction of an instantaneous fact (valid at a single time instant) and the *slice* operator for the extraction of a historical fact (with a longer lifetime). In this case, the domain of temporal queries is constituted by historical databases according to *valid time* measure [28].

Combining the classification proposed in [10] with our approach results in fourteen types of primitive queries, as can be seen from Table 1. According to [10] we do not consider the combinations *data selection/data projection* and *temporal selection/temporal projection* (the former represents a conventional query and the latter is impossible to represent in a temporal query since conditions and results do not apply to temporal values only). It is worth noting that this classification does not include queries referring to incomplete temporal information.

	Snapshot Query	*Slice Query*
Data Selection/Temporal Projection	*ex. 3*	
Data Selection/Mixed Projection		*ex. 4*
Temporal Selection/Data Projection	*ex. 1*	
Temporal Selection/Mixed Projection		*ex. 6*
Mixed Selection/Data Projection	*ex. 2*	
Mixed Selection/Temporal Projection		*ex. 7*
Mixed Selection/Mixed Projection	*ex. 5*	

Table 1. Different Query Types

In the following examples we ilustrate seven distinct cases in Table 1:

1. *What were the salaries of employees at 10/01/95?*
2. *What was the salary of 'John' when he changed his status?*
3. *Since when does 'John' own more than 5,000?*
4. *What is the history of the salaries of the database group?*
5. *What was the last salary of employees who started working for more than 1,000? And since when?*
6. *What is the history of the salaries of the employees during the period 01/01/95-31/12/95?*
7. *Which is the period that 'John' worked at personal department during 1990-1995?*

Note that examples 1, 2, 3 e 5 retrieve information concerning a single time instant, whereas the other examples refer to a time slice. TVQOs may be applied to both temporal classes and temporal relationships. For instance, Figure 2.1 shows a symbolical representation of the application of both operators to a temporal relationship *EMPLOYEEhasSALARY* (salary history). Note that the *snapshot* operator returns the salary of only one time interval whereas the *slice* operator returns the salaries of one or more time intervals.

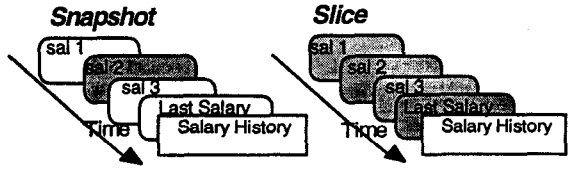

Figure 2.1. TVQOs on a Salary History

2.2. Examples of Interaction

The user starts interacting with a diagrammatic representation of a GMDB[1] containing information on

[1]Since both the Graph Model and the associated Graphical Primitives consist of elementary graphical elements, it is possible to use them as

employees and employement agencies (see Fig. 2.2, where, for the sake of simplicity, not all relationships are visualised).

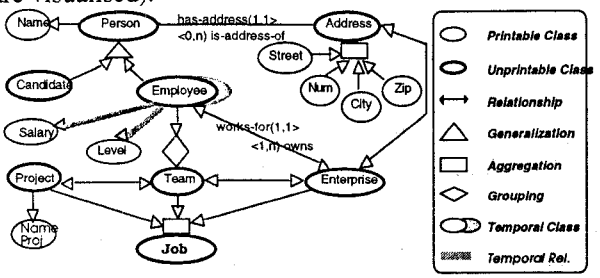

Figure 2.2. A Visual Schema

The meaning of the different graphical symbols is explained in the attached legenda. Moreover, notations such as *works-for(1,1>*, indicates that the relationship ***works-for*** between classes *EMPLOYEE* and *ENTERPRISE* (also denoted as *EMPLOYEE**works-for**ENTERPRISE*) has minimum and maximum cardinalities equal to one. The notation *<1,n) owns* represents the cardinality of the inverse relationship.

We show the interface "look and feel" by a simple example. A typical screen displays three windows: a *Schema Window* that shows the visual schema to which the query operators are applied; a *Temporal Window* that shows more details when a user selects a temporal class/relationship in order to formulate a temporal query; and a *Result Window* that shows the result of the query. Furthermore, *a tool bar* is provided that contains some *pull-down* menus for editing the schema drawing, plus a menu containing the *Conventional Query* and *Temporal Query* options. In this paper, we only concentrate on temporal queries.

Initially, the user chooses one database schema, then s/he selects some classes and relationships of interest, in order to create the sub-schema that contains all information s/he needs to formulate her/his query. Assuming that the user is interested in knowing: *"On what projects and with which salary, each employee worked on 10/01/95?"*, s/he will select the nodes: *EMPLOYEE, NAME, SALARY, PROJECT, PROJNAME,* generating a new sub-schema, where the *Temporal Query* option is also selected. As a consequence, the same diagram appears in the temporal window (Figure 2.3). As a next step, the user selects the temporal relationship *EMPLOYEE**works-on**PROJECT* in the temporal window, representing the projects related to the employees (if the user would like to select the employees related to the projects, the inverse relationship should be selected).

basic constituents of more complex existing visual representations (i.e., E-R diagrams, object networks, etc.) and visual query languages, so giving them a formal semantics independent of the underlying data model.

In the temporal window, there are six ***process icons*** [8], as shown in Figure 2.3: *slice* operator; *snapshot* operator; *temporal condition* (when), *data condition* (where); *data display* and *time display*. Beside these six specific icons for the construction of a temporal query, there is an icon for selecting classes and/or relationships and one help icon. Furthermore, there are two slide bars used in snapshot and slice queries respectively. Inside this visual environment, the user can specify any temporal query within the domain of temporal queries presented in Table 1.

The category of our example is *snapshot, temporal selection/data projection.* So, the user specifies this query selecting the snapshot, temporal condition and data display icons. For instance, Figure 2.3 shows the effects of the user's selection of the snapshot icon applied to the temporal relationship *EMPLOYEE* ***works-on*** *PROJECT.*

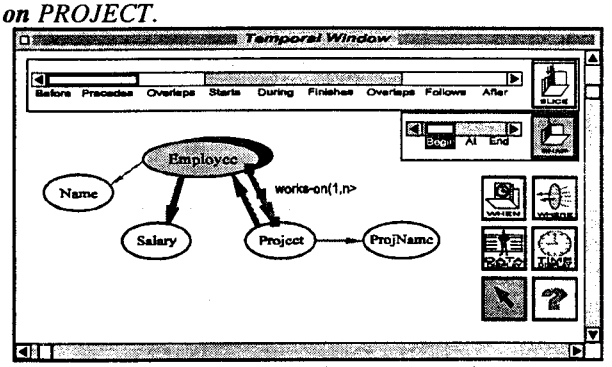

Figure 2.3. Await Condition of the Snapshot

In order to specify the temporal condition *"on 10/01/95* ", the user must select the *when* icon. As a consequence, a menu pop-up with the *constant, now, temporal reference, period* and *all* options appears (*period* and *all* options are disabled because they cannot be used in snapshot query such as the one in the example). *Temporal reference* is a time reference to another data (e.g. *"What was the salary of 'John' when he changed his status?"*), that is, the time during which the data condition (*"when he changed his status"*) occurs (such temporal reference is also used in OQL/T[30] and in the visual query editor for the TEER model[20]). The *all* option should be used for recovering all history of a class or a relationship. In the example, the user selects the *constant* option. Then, s/he selects the constant 10/01/95 after s/he has chosen the *date* (*MMDDYY*) granularity[2]. The constant appears inside the slide bar that contains the *Begin, End,* and *At* operators as shown in Figure 2.4 (the figure only shows the fragment of Fig. 2.3 corresponding to the slide bar).

[2] What has been called *date* in fact means *day* granularity (YYMMDD). Other granularities are year, mounth, hour, etc.

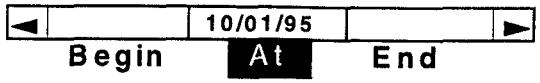

Figure 2.4. Slide Bar used in a Snapshot Query

Let *t* be a time instant. The *Begin* operator retrieves only the elements whose lifetime starts at *t*. With *End* the lifetime must finish at *t*, and with *At* the time *t* must be in between the lifetime of the element. The user moves the constant to the *At* option.

In order to specify the data projection, the user must select the data display icon and select the nodes *NAME*, *SALARY* and *PROJNAME*. At this point, the temporal query has been completely specified, and the system will process the query.

The user formulates *slice* queries similarly to snapshot ones. The only difference is that, when specifying the temporal condition, a different slide bar needs to be used, shown in the upper part of Fig. 2.3 (this slide bar is not used within the *all* condition), containing the predefined temporal comparison operators between time intervals, called primitive temporal relationships, described by Allen[2]. We use Before, After, Precedes, Follows, *Overlap (cross), Start, Finishes, During, Equivalent* as temporal relationships.

Assuming that the query is: *"Give the history of employees' projects before 1995"*, for completing the slice operation, the user selects the *period* option (since the query is a slice, the *constant* and *now* options are disabled). Then, s/he selects the period *"from 01/31/95 until 12/31/95"*, after s/he has chosen the date granularity, and moves the slide on the *Before* value on the bar, as shown in the first bar of Fig. 2.5. Note that the grey part of the bar represents the specified time, and its spatial position with respect to the bar values evokes the temporal relationship among them (e.g., when selecting the *Overlaps* value, the slide partially overlaps the grey area).

The order of the temporal relationships in a slide bar is based on the *neighbors* temporal primitives concept[3], introduced by Freksa in [13] and further discussed in [17].

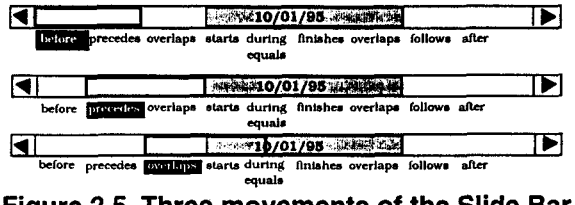

Figure 2.5. Three movements of the Slide Bar

[3] Two temporal relationships are *neighbors* if a continuous change of the events transforms a relation into another without passing through an additional temporal relationship [17].

By acting on the *time display* icon, the user has the possibility of visualizing time intervals. First, s/he may display time intervals or instants in different granularities, as both query result and operands of a temporal reference. A temporal reference can be used in combination with both the snapshot and slice operators. For instance, in order to construct the query *"Which salaries did the employees earn when they changed their level for the first time"*, the slice operator on the temporal relationship *EMPLOYEE has LEVEL* is used for requesting the time when the level changed, and the snapshot operator for retrieving the corresponding salary. Since the result of a slice operation may contain several time intervals, the user can select one or more of them, and specify whether the selection concerns the starting point, the ending point or the duration of each interval. In the example, the endpoint of the first interval associated to the level of employees has been chosen (Figure 2.6). If the *duration* is selected, the system offers a menu of aggregate functions, *min, max, count, avg* and *sum*, which applies to the set of temporal elements in the relationship or class.

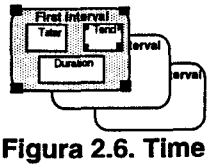

Figura 2.6. Time Interval Display

3. Formalization

In this section we present a formal specification of both the data model and the TVQOs. The formalism to be used is an extension of the work of Tiziana Catarci & al. [3]. They proposed a graph-based formalism (the Graph Model) and a minimal set of model-independent visual primitives (the Graphical Primitives) for visually representing and querying databases.

3.1. The Graph Model

A ***Graph Model Database (GMDB)*** is a triple $<g, c, m>$, where g is a Typed Graph, c is a set of Constraints, and m is an Interpretation. The schema of a database, i.e. its intensional part, is represented in the Graph Model by the Typed Graph and the set of Constraints. The instances of a database, i.e. its extensional part, are represented by the notion of Interpretation.

The **Typed Graph** is a tuple $<N, E, L, f_l, f_s>$, where:

$N = N_c \cup N_r$ is the set of nodes; N_c is the set of so-called class-nodes, and N_r is the set of the so-called role-nodes representing the relationships[4]. Moreover, N_c is partitioned into N_{cp}, the set of printable nodes, N_{cu}, the set of unprintable nodes, and N_{ct}, the set of temporal

[4] Note that the role-nodes are not visualized in the schema drawing, where edges appear to connect class-nodes.

nodes, corresponding to temporal classes. N_r is partitioned into N_{r_n}, the set of non-temporal relationships, and N_{r_t} the set of temporal relationships;

$E \subseteq N_c \times N_r \cup N_r \times N_c$ is the set of edges;

L is the set of labels;

f_l is a total biunivocal function from N to L, associating a label to each node;

f_s is a total function which characterizes the selection state of the elements of the typed graph, mapping each node to a value in {***unselected, selected, displayed***};

The following notations will be used:

$AD\{n_1,...,n_k\}$ is the set of nodes adjacents to a given set of nodes $\{n_1,....,n_k\}$ minus $\{n_1,....,n_k\}$. If n is a role-node corresponding to a binary relationship then $AD\{n\}=\{n_1, n_2\}$ with $n_1, n_2 \in N_c$.

$T = \{t_1, t_2,...\} \cup \{now\}$ is the ordered set of time plus the special value *now*, where *now* is the continuously changing current time.

$LS = \{ls \mid ls = \{I_1,...,I_n\}\}$ is the set of lifespans of type *temporal element* (see also [14], [20], [31]), that is the set of disjoint time intervals, where each I_k is represented as $<t_{k1}, t_{k2}>$.

$O = O_p \cup O_u \cup O_t$ is the set of all objects. O_p are the printable objects, O_u are the unprintable objects and O_t are the temporal objects. Note that $O_p \cap O_u \cap O_t = \emptyset$.

U is a universe, that is a set of structured objects, defined as the smallest set containing O, and all the possible labeled tuples (of any arity) $<l_1:o_1,..... l_k:o_k, l>$, where $l_1,....., l_k$ are labels of class-nodes; $o_1,....., o_k$ are elements of O and l is the label of a role-node, i.e.: $\exists n \in N_r \mid f_l(n) = l$. Objects related by binary relationships are also denoted as $o_1\ l\ o_2$.

$U_t \subseteq U$ is the so-called temporal universe, that is the subset of U constituted by the temporal objects O_t and all the possible labeled tuples (of any arity) $<l_1: o_1,..... l_k:o_k, t>$ where $l_1,....., l_k$ are labels of class-nodes; $o_1,....., o_k$ are elements of O_t and l is the label of a temporal role-node, i.e.: $\exists n \in N_{r_t} \mid f_l(n) = l$.

$\Theta: U_t \rightarrow LS$ is a total function that associates to each element x of U_t a lifespan, denoted as $\Theta(x) = ls$. For each $I_k \in \Theta(x)$, we denote $I_k = <begin_{I_k}(x), end_{I_k}(x)>$, with $x \in U_t$. If $t \in I_k$, then x_t denotes the state of x at time t.

An **interpretation** of a Typed Graph is a function $m: N \rightarrow 2^U$ mapping each node $n \in N$ to a subset of U, as follows:

If $n \in N_{c_p}$, then $m(n) \subseteq O_p$;

If $n \in N_{c_u}$, then $m(n) \subseteq O_u$;

If $n \in N_{c_t}$, then $m(n) \subseteq \{<o, \Theta(o)> \mid o \in O_t\}$, the interpretation of n at a instant t is denoted as $m(n_t) = \{o_t \mid o \in m(n)\}$.

If $n \in N_{r_n}$ and $\{n_1, n_2,..., n_k\} = AD\{n\}$, then $m(n)$ is a set of tuples of the form $<f_1(n_1): o_1,....., f_1(n_k):o_k, f_l(n)>$, where $f_1(n_1),....., f_1(n_k)$ and $f_l(n) \in L$ and $o_1,....., o_k \in m(n_1) \times m(n_2) \times ..... \times m(n_k)$.

If $n \in N_{r_t}$ and $\{n_1, n_2,..., n_k\} = AD\{n\}$, then $m(n)$ is a set of tuples, each one having associated a time interval, i.e.: $m(n) \subseteq \{x, \Theta(x)\}$, where x is the tuple $<f_1(n_1):o_1,....., f_1(n_k):o_k, f_l(n)>$; $f_1(n_1),....., f_1(n_k)$ and $f_l(n) \in L$; $o_1,....., o_k \in m(n_1) \times m(n_2) \times ..... \times m(n_k)$ and the interpretation of n at instant t is denoted as $m(n_t) = \{x_t \mid x \in m(n)\}$.

The set of **constraints** C is specified by the designer using a suitable language[5]. Those constraints which are mostly relevant for the purpose of this paper are the following:

n_1 ***is-a*** $n_2 \Leftrightarrow n_1, n_2 \in N_c \wedge m(n_1) \subseteq m(n_2)$;

$\{n_1,...,n_k\}$ ***part-of*** $n \Leftrightarrow n_1,...,n_k\ ,\ n \in N_c \wedge m(n) \subseteq m(n_1) \times ..... \times m(n_k)$;

The constraints described above represent the hierarchies of generalization and aggregation respectively, as usually defined in semantic and conceptual data models [18].

As for the temporal relationship, we define the following constraint:

For each $n \in N_{r_t}$ and $m(n) \subseteq \{x, \Theta(x)\}$, where x is the tuple $<f_1(n_1):o_1,....., f_1(n_k):o_k, f_1(n)>$; $f_1(n_1),....., f_1(n_k)$ and $f_1(n) \in L$; $o_1,....., o_k \in m(n_1) \times m(n_2) \times ..... \times m(n_k)$; the lifespan of n must be a subset of the intersection of the lifespans of the related objects, i.e.: $\Theta(<f_1(n_1):o_1,....., f_1(n_k):o_k>) \subseteq \Theta(o_1) \cap ..... \cap \Theta(o_k)$.

3.2. The Temporal Visual Query Operators

The formal specification of the TVQOs *Snapshot* and *Slice* is based on the idea of expressing any query-oriented user interaction with a database in terms of a simple set of fundamental Graphical Primitives (GPs) [3]. In [3] two primitives are presented: *selection of a node* and *drawing of an edge*, and it is demonstrated that all first order queries can be expressed by composing the two primitives. In this paper we extend the GP set with the two TVQOs (for the sake of simplicity, we consider only binary relationships). Let $D = <g,c,m>$ be a GMDB. At the beginning of the interaction, the user selects from the schema the classes and relationships that are of interest for his/her query. This operation

[5] for more details on the constraint language see [4].

corresponds to the GP *selection of a node*, through which the state of a node is switched from the value *unselected* to either the value *selected*, which means included in the schema of interest, or *displayed*, which means included in the query result (see [3] for more details). Let $D' = <g',c',m'>$ be the GMDB resulting from the node selections. The application of a TVQO on D' results in a new GMDB $D^r = <g^r,c^r,m^r>$.

TVQO Snapshot

The TVQO *snapshot* corresponds to a sequence of three selections: of either a class-node or role-node, of the snapshot icon, of a temporal predicate $p(t)$ of the form *begin(t)*, *end(t)* or *at(t)*, where t is either an instant of T or a time reference to another data. More formally:

Let $n \in N_{c_t}$, $t \in$ T, $p(t)$ a temporal predicate, a ***snapshot* over a temporal class-node n with $p(t)$** in D' is a function: $\phi_n\ (D', n, p(t)) = D^r$ such that $D^r = D'$, except that:

$N_c{}^r = N_c{}' \cup \{s\}$ (s is a new class node);

$L^r = L' \cup \{l_s = l \circ p(t) \mid l = f'_l(n)\}$;

$f_l^r = f'_l \cup \{f_l^r(s) = l_s\}$;

$c^r = c' \cup \{s$ ***is-a*** $n\}$;

m^r is equal to m' except for $m^r(s)$:

if $p(t) = at(t)$ $m^r(s) = m'(n_t)$.

if $p(t) = begin(t)$ $m^r(s) = \{o^r \mid \exists o, I_k\ (o \in m'(n) \wedge I_k \in \Theta(o) \wedge begin_{I_k}(o) = t \wedge o^r = o_{begin_{I_k}(o)})\}$;

The *end(t)* operator is analogously defined as *begin(t)*.

Let $n \in N_{r_t}$, $t \in$ T, $p(t)$ a temporal predicate, a ***snapshot* over a temporal role-node n with $p(t)$** in D' is a function: $\phi_n\ (D', n, p(t)) = D^r$ such that $D^r = D'$, except that:

$N_{r_n}{}^r = N_r{}'_n \cup \{s\}$ (s is a new role-node);

$N_{r_t}{}^r = N_r{}'_t - \{n\}$;

$L^r = L' \cup \{l_s = l \circ p(t) \mid l = f'_l(n)\}$;

$f_l^r = f'_l \cup \{f_l^r(s) = l_s\}$;

$c^r = c' \cup \{AD\{s\} = AD\{n\}\}$;

m^r is equal to m' except for $m^r(s)$:

If $p(t) = at(t)$, $m^r(s) = m'(n_t)$.

If $p(t) = begin(t)$, $m^r(s) = \{o_1\ l_s\ o_2 \mid \exists\ o_1, n, o_2, I_k\ (o_1\ l\ o_2 \in m'(n) \wedge I_k \in \Theta(o_1\ l\ o_2) \wedge begin_{I_k}(o_1\ l\ o_2) = t \wedge o_1\ l\ o_2 = (o_1\ l\ o_2)_{begin_{I_k}(o_1\ l\ o_2)})\}$;

The *end(t)* operator is analogously defined as *begin(t)*.

TVQO Slice

The TVQO *slice* corresponds to a sequence of three selections: of either a class-node or role-node n, of the slice icon, of a temporal predicate $p(I)$ of the form *before(I)*, *after(I)*, *during(I)*,*overlap(I)*, *equivalent(I)*, *follows(I)*, *precedes(I)*, *start(I)*, *finishes(I)*, where I is an interval of *Int*. More formally:

Let $n \in N_{c_t}$ and $p(I)$ a temporal predicate, a ***slice* over a temporal class-node n with $p(I)$** in D' is a function: $\omega_n\ (D', n, p(I)) = D^r$ such that $D^r = D'$, except that:

$N_c{}^r = N_c{}' \cup \{s,i\}$ (s and i are new class nodes);

$L^r = L' \cup \{l_s = l \circ p(I) \mid l = f'_l(n)\}$;

$f_l^r = f'_l \cup \{f_l^r(s) = l_s, f_l^r(i) = p(I)\}$;

$c^r = c' \cup \{\{n, i\}$ ***part-of*** $s\}$;

m^r is equal to m' except for $m^r(i)$ and $m^r(s)$ ($m^r(s)$ is immediately derivable from $m^r(i)$ and $m^r(n)$):

If $p(I) = before(I)$, $m^r(i\ ) = \{I_k \in \Theta(o) \mid o \in m'(n) \wedge end_{I_k}(o) < begin(I)\}$;

The other cases are analogously defined.

Let $n \in N_{r_t}$, $AD\{n\} = \{n_1, n_2\}$, and $p(I)$ a temporal predicate, a ***slice* over a temporal role-node n with $p(I)$** in D' is a function $\omega_n\ (D', n, p(I)) = D^r$ such that $D^r = D'$, except that:

$N_c{}^r = N_c{}' \cup \{s,i\}$ (s and i are new class nodes);

$N_{r_t}{}^r = N_r{}'_t - \{n\}$;

$L^r = L' \cup \{l_s = l \circ p(I) \mid l = f'_l(n)\}$;

$f_l^r = f'_l \cup \{f_l^r(s) = l_s, f_l^r(i) = p(I)\}$;

$c^r = c' \cup \{\{n_1, n_2, i\}$ ***part-of*** $s\}$;

m^r is equal to m' except for $m^r(i)$ and $m^r(s)$ ($m^r(s)$ is immediately derivable from $m^r(i)$, $m^r(n_1)$ *and* $m^r(n_2)$):

If $p(I) = during(I)$, $m^r(i) = \{I_k \in \Theta(<o_1\ l\ o_2>) \mid \exists\ o_1, n, o_2\ (o_1\ l\ o_2 \in m'(n) \wedge (begin_{I_k}(o_1\ l\ o_2) \geq begin(I)) \wedge (end_{I_k}(o_1\ l\ o_2) \leq end(I))\}$;

The other cases are analogously defined.

Besides the TVQOs, there is a further operator, called *Time-display*(T_d). It takes out temporal information generated by the slice operator. Its application was presented in the last paragraph of the section 2.2. Due to space limitations, we do not explain it in more details.

3.3. Examples

In this subsection, we show the formal counterpart of the examples presented in Section 2.

1) Temporal query with *snapshot: "On what projects and with which salary, each employee worked on 10/01/95?"*

The nodes *EMPLOYEE* and *PROJECT* are selected, while the nodes *NAME-PROJ* and *SALARY* are displayed.

Snapshot (10.01.95):

$D^r = \phi e\ (D',\ EMPLOYEE\textbf{works-on}PROJECT,\ at(10.01.95))$

2) Temporal query with *slice:"Give the history of employees' projects before 10/01/95"*

The nodes *EMPLOYEE* and *PROJECT* are selected, while the nodes *NAME-PROJ* and *NAME* are displayed.

Slice(before 10.01.95):

$D^r = \omega_e\ (D', EMPLOYEE\textbf{works-on}PROJECT,\ before(10.01.95))$.

3) Query combining *slice* with *snapshot*: *"Which salaries did the employees earn when they changed their level for the first time"*

The nodes *EMPLOYEE* and *LEVEL* are selected, while the nodes *NAME* and *SALARY* are displayed.

Slice(*all*):

$D^r = \omega_e\ (D', EMPLOYEE\textbf{has}LEVEL,\ all)$.

Time-Display:

$t = T_d\ (D^r,\ i,\ last\text{-}instant(first\text{-}interval(I)))$

Snapshot(t):

$D^{r'} = \phi_e\ (D^r,\ EMPLOYEE\textbf{has}SALARY,\ at(t))$

4. Related Work

There is a considerable number of proposals for extending existing data models in order to take into account the temporal dimension. Most of them are extensions of the relational model, as HRDM[9], TRM[22], HDBMS[23] and TSQL2[29]. Also, extensions of the Entity-Relationship Model have been developed: TEER[12], ERT[21] and TEMPORA[32]. In the object-oriented approaches, we have OSAM*/T[30], OODAPLEX[35] and TOM[25]. Analogously, the corresponding query languages have been extended in order to manipulate temporal data. There are SQL extensions, such as TSQL[22], HSQL[23], ERT-SQL[21], TSQL2[29] and [15], QUEL extensions, such as TQUEL[28] and HTQUEL[14], an extension of the object language OQL[1], called OQL/T[30], another object-oriented approach to a temporal query language is in [11], while TBE[31] is an extension of QBE[36] to deal with historical databases. To the best of our knowledge, the only proposals of visual interfaces for temporal databases are ERT/vql [33], GEST[16], TVQL[17] and [20].

ERT/vql is a visual language for the ERT model. It contains an interesting feature: a result visualisation phase where advanced interactive visualisation techniques are applied over the query result. However, during the query construction phase, the user needs to adopt a textual syntax in spite of the visual schema representation offered by the system. GEST[16] is a visual interface for TSQL2. It is based on the concept of event, and the basic temporal constructs in GEST are relations between events. As such, it is not suitable to represent and query hystorical (discrete) databases, which need constructs such as class and relationship to be modeled. Whereas, GEST is suitable in applications such as monitoring or planning systems, where the data change their states at fixed time intervals. The visual language TVQL[17] shares with GEST the inadequacy to represent and query historical databases. TVQL is mainly used for video data, more specifically for identifying temporal trends in video data. However, the visual approach the authors adopt for query specification is similar to ours in using slide bars to represent primitive temporal relationships. Another approach is presented in [20]. It consists of a visual query editor for the TEER data model, a temporal extension of the Extended Entity-Relationship (EER) Model [12]. This work is similar to ours, the difference being in the visual query formulation. Indeed, in [20] the user has to follow different procedures in order to express semantically similar queries, while in our approach all temporal queries can be simply expressed by combining the snapshot and slice operators.

Conclusion

This paper is an attempt to put in a easy-to-use visual form the difficult task to formulate queries on temporal databases. The different kinds of temporal queries (snapshot and slice) are shown in a homogeneous way, inside a global visual environment. This approach facilitates the user in expressing the query, since s/he has not to learn any complex syntax and can incrementally formulate the query, also receiving immediate graphical feedback.

The system is actually under implementation, and the next step will be to carry out usability tests in real settings. Moreover, future work will aim to extend the language expressive power. In particular, we plan to: a) allow the explicit specification of quantifiers. Currently, the existential quantifier is assumed implicitly, while universal selection must be done by using set operations; b) permit the visual specification of recursive queries.

References

[1] Alashqur A.M et al., "OQL: A Query Language for Manipulating Object Oriented Databases", *Proc. of the 15th VLDB*, Amsterdam, 433-441, August, 1989.

[2] Allen, J.F., "Maintaining Knowledge about Temporal Intervals", *Comm. of ACM*, 26(1), 832-843, 1983.

[3] Catarci T., Santucci G., Angelaccio M., "Fundamental Graphical Primitives for Visual Query Languages", *Information Systems,* 18(2), 75-98, 1993.

[4] Catarci T., Santucci G., Cardiff J., "Knowledge-based Schema Integration in a Heterogeneous Environment", *Proc. of the 2nd NGITS Conf.,* Naharia, Israel, 1995.

[5] Catarci T., Chang S.K., Costabile M.F., Levialdi S., Santucci G., "A Graph-based Framework for Multiparadigmatic Visual Access to Databases", *IEEE TKDE,* 8(3), 455-475, 1996.

[6] Catarci T., Costabile M.F, Levialdi S., Batini C., "Visual Query Systems: Analysis and Comparison", *Journal of Visual Languages and Computing, to* appear, March 1997.

[7] Catarci T., Santucci G., Cardiff J., "Graphical Interaction with Heterogeneous Databases", *VLDB Journal,* to appear, 1997.

[8] Chang S.K, "Principles of Pictorial Information Systems Design", Prentice-Hall, Englewood Cliffs, NJ, 1989.

[9] Clifford J., Crocker A., "The Historical Relational Data Model (HDRM) and Algebra Based on Lifespans", *Proc. Int. Conf. on Data Engineering,* Los Angeles, CA, 528-537, 1987.

[10] Edelweiss N., Oliveira J., "Modelagem de Aspectos Temporais de Sistemas de Informação", *IX Escola de Computação,* Recife, 1994.

[11] Edelweiss N., Oliveira J., Pernici B., "An Object-Oriented Approach to a Temporal Query language", *Proc. of 5th DEXA Conf.,* 225-235, Athens, Greece. LCNS, No. 856, Springer-Verlag.

[12] Elmasri R. et al. "A Temporal Model and Query Language for EER Databases", *Temporal Databases,* Tansel A. et al (Ed.), Series Benjamin/Cummings, Redwood City, CA, 212-229, 1993.

[13] Freksa C., "Temporal reasoning based on semi-intervals", *Artificial Intelligence,* 54, 199-227, 1992.

[14] Gadia S.K., "A Homogeneous Relational and Query Language for Temporal Databases" *ACM TODS,* 13(4), 1988.

[15] Grandi F., Scalas M., Tiberio P., "A History-oriented Temporal SQL Extension", in *Proc. of 2nd NGITS Conf.,* Naharia, Israel, 1995.

[16] Harris, W., Gray A., "Using Temporal Constructs in Temporal Databases", *Recent Advances in Temporal Databases,* Clifford J. (Ed.), Series Workshop in Computing, Springer-Verlag, Berlin, 133-152, 1993.

[17] Hibino S., Rundensteiner E., "A Visual Query Language for Identifying Temporal Trends in Video Data", *Int. Work. MDBMS,* New York, 74-81, 1995.

[18] Hull R., King R., "Semantic Database Modeling: Survey, Applications and Research Issues", *ACM Computing Surveys,* 19(3), 201- 260, 1987.

[19] Jensen C.S. et al, "A Consensus Glossary of Temporal Database Concepts", *SIGMOD-RECORD* 23(1), 1994.

[20] Kouramajian V., Gertz M., "A Visual Query Editor for Temporal Databases", *Proc. of the 14th Int. Conf. on OO and E-R Modeling,* 388-399, 1995.

[21] Loucopoulos P., Theodoulidis B., Pantazis D., "Business Rules Modelling: Conceptual Modelling and Object-oriented Specifications", *IFIP TC8/WG8.1 Working Conference,* Canada, 323-342, 1991.

[22] Navathe S.B., Ahmed R., "Temporal extensions to the Relational Model and SQL", *Temporal Databases,* Tanzel A. et al. (Ed.), Series Benjamin/Cummings, Redwood City, CA, 92-109, 1993.

[23] Sarda N.L., "Extensions to SQL for Historical Databases", *IEEE TKDE,* 2(2), 220-230, 1990.

[24] Sawier P. et al, "Object-Oriented Database Systems: A Framework for User Interface Development", *IDS Glasgow 1992,* Cooper R. (Ed.), Series Workshop in Computing, Springer-Verlag, London, 25-38, 1993.

[25] Schiel U., "An Open Environment for Objects with Time and Versioning", *Proc. EastEurOOpe,* Bratislava, 116-125, 1991.

[26] Shneiderman B., "Direct Manipulation, a Step Beyond Programming Languages", *IEEE Computer,* 16(8), 57-69, 1983.

[27] Fernandes S., "Um Ambiente Gráfico de Consultas a um Banco de Dados Temporal Orientado a Objetos", *IX Simpósio Brasileiro de Banco de Dados,* São Carlos, 1-15, 1994.

[28] Snodgrass R.T., "The Temporal Query Language Tquel" in *ACM TODS 12(2),* 247-298, 1987.

[29] Snodgrass R.T. et al, "A TSQL2 Tutorial", in *SIGMOD Record,* 23(3), September, 1994.

[30] Su S.Y.W., Hsin-Hsing Chen, M., "A Temporal Knowledge Representation Model OSAM*/T and Its Query Language OQL/T", *Proc. of the 17th VLDB,* Barcelona, 431-442, September 1991.

[31] Tansel A.U., Arkun M.E., Ozsoyoglu G., "Time-by-Example Query Language for Historical Databases", *IEEE TSE,* 15(4), 464-478, April 1989.

[32] Theodoulidi B., Loucopoulos P., Wangler B., "A Conceptual Modeling Formalism for Temporal Database Applications", *Information Systems,* 16(4), 401-416, 1991.

[33] Theodoulidis B. et al., "Interactive Querying and Visualisation in Temporal Databases", in *Temporal Reasoning Workshop of the 4th DOOD Conf.,* Singapore, 1995.

[34] Tsotras V.J., Kumar A., "Temporal Database Bibliography Update", *ACM SIGMOD,* 25(1), 41-51, March 1996,

[35] Wuu G.T.J., Dayal U., "A Uniform Model for Temporal and Versioned Object-oriented Databases", *Temporal Databases,* Tansel A. et al. (Ed.), Series Benjamin/Cummings, Redwood City, CA, 230-247, 1993.

[36] Zloof M., "Query-by-Example:A Database Language", *IBM Systems Journal,* 21(3), 324-343, 1977.