시간지원 데이타베이스의 영상화를 위한 접속 시스템의 설계 및 구현

이 언 배† 류 근 호††

요 약

본 연구는 시간지원 데이타베이스 관리시스템과 영상화 시스템간의 사용자 접속 시스템을 설계하고 구현하였다. 이 접속 시스템은 시간지원 데이타베이스 관리시스템과 영상화 시스템 및 접속 제어 시스템으로 구성된다. 시간지원 데이타베이스 관리시스템은 시간을 지원하고 영상화 시스템은 아이콘을 사용하여 영상을 만든다. 또한 접속 시스템은 영상 지식베이스와 시간요소를 이용하여 영상 애니메이션을 보여준다. 본 논문은 시간지원 데이타베이스 및 영상화 시스템의 구조와 상호 접속 방법을 설명하였다.

Design and Implementation of Interfacing System for Graphical Display of Temporal Databases

Eun Bae Lee† and Keun Ho Ryu††

ABSTRACT

This paper describes a design and implementation of a user interface system between a temporal database management system (DBMS) and a graphical display system. This interfacing system consists of a temporal DBMS, a graphical display system, and an interface control system. The temporal DBMS supports time and the graphical display system draws a graphic using an icon. The interfacing system shows the graphical animation with time factors by using a graphical knowledgebase. We describe how to interface them as well as the structure of temporal DBMS and graphical display system.

1. Introduction

In this paper, we describe a graphical display by using interfacing technique between a temporal DBMS and a graphical display system. The user interface system consists of an interface control system, a temporal DBMS, and a graphical display system.

The simple temporal database management system supports time. The temporal DBMS performs retrieve, append, delete, and replace commands to manipulate a query. The temporal DBMS retrieves tuples from a temporal database by using the temporal query language in [Sno87]. The graphical display system is a graphical system which displays the temporal information as described by a graphical schema with a data specification language IDL(Interface Description Language)[Sno89]. We implemented an interface control system that the result of temporal query in the temporal DBMS can not only be displayed or saved, but also sent to the display system as input instances for the graphi-

• This work was supported in part by the Korea SANHAK Foundation and in part by NSF grant IRI -8902707.

† 종신회원 : 한국방송통신대학교 전자계산학과 부교수
†† 종신회원 : 충북대학교 컴퓨터과학과 부교수

논문접수 : 1994년 2월 21일, 심사완료 : 1994년 7월 14일

cal display system. The graphical display system uses a graphical knowledge, while the temporal DBMS manipulates a temporal database.

The tuples retrieved from the temporal DBMS are displayed according to a user description, which specifies how to represent time and attributes of tuples. The graphical display system uses the Sun Core graphic package to draw graphical objects described by graphical schema. Finally, the user interface system draws graphical animation with time concept from temporal database.

For graphical animation, when a user command is issued, the user interface control system activates two processes: *temporal DBMS*(the temporal DBMS) and *display*(the graphical display system). According to the user command, it may generate a new command. The user interface control system sends the user command or new generated command to the temporal DBMS and/or the graphical display system. Tuples produced by the temporal DBMS are sent to the graphical display system for the graphical display.

For example, let us assume that a user issues a *represent* and a *display* command. The *interface* control system reads the commands and interprets them. The represent command is sent to the display system in order to create a graphical schema and the temporal DBMS marks the temporal tuples to be displayed. Also, the display system defines the graphical schema from the contents of the *represent* command. Following the *represent* command, the user gives a *display* command from the user to the interface control system. It is sent to the temporal DBMS. The temporal DBMS creates a tuple file from relations marked and then sends the file to the graphical display system. While the temporal DBMS produces the tuple file, the display system will wait for the tuple file produced by the temporal DBMS after getting the *display* command from the interface control system.

The temporal DBMS and the graphical display system works at the system software level. The temporal database and the graphical knowledgebase are stored in secondary storage. The interface control system handles the temporal DBMS and the graphical display system for the user interface. Ryu[Ryu91a] has implemented a temporal DBMS which was supported by a valid time and a transaction time. The dispay of temporal information was studied in [Sha86]. He implemented the display system to display temporal information. Also, Han[han88] and Ryu[Ryu91B, LeR93] described how to display the temporal information with examples.

The organization of the rest of the paper is as follows. In chapter 2, we describe temporal relations and temporal queries. Then , in chapter 3, the structures of the temporal DBMS, the display system, and interface control system are covered. Finally, chapter 4 summarizes this paper.

2. Temporal Database, Temporal Query, and Graphical Information

2.1 Temporal database

A temporal information is the recording of past activities. A temporal database has temporal information instances. A temporal query can be used to extract temporal information out of the temporal database.

The temporal database supports two types of time: transaction and valid. The valid time is a time at which an event happens in the real world, while the transaction time is a time at which it is recorded in the database.

Depending on supported time, four different types of database relations may result: snapshot, rollback, historical, and temporal. Each one of these can be associated with a class of time. We similarly have an example [Ryu91B,LeR93,Sha86] for a temporal database relation in Figure 1.

airplanestatus(plane, model, status)

plane	*model*	*status*	*from*	*to*
CessI	414A	inrepair	8:00	10:30
CessII	414A	demoflight	8:00	10:00
PiperI	ArcherII	onground	8:00	9:00
PiperII	ArcherII	trainingflight	8:00	9:00
CessI	414A	onground	10:30	12:00
CessII	414A	onground	10:00	11:00
CessII	414A	trainingflight	11:00	12:00
PiperI	ArcherII	freeflight	9:00	12:00
PiperII	ArcherII	inrepair	9:00	12:00

(Fig. 1) Example relation.

The example of database relation, *airplanestatus* has three explicit attributes (plane, model, status) and two implicit attributes (from, to). The implicit attribute *from* and to represent a valid time for supporting an interval of time. The attribute *from* is the starting point of the valid time and the attribute *to* is the ending point of the valid time.

A snapshot database relation only supports current time. A rollback database relation supports transaction time and historical database relation supports valid time. Lastly, a temporal relation supports all of the above, transaction time and valid time.

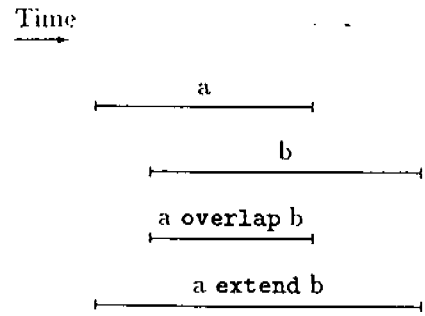
2.2 Temporal query and graphical information

We review a temporal query language TQuel[Sno87] which will be used in the interfacing system. The temporal query may contain valid and when clause.

The valid clause specifies the value of the implicit valid time attribute. The valid at clause indicates a single time in the temporal attribute while the valid interval clause defines an interval of time as valid **from ... to ...** in the temporal attribute.

The when clause is the temporal predicate analogous to Quel's where clause. The temporal predicate that follows the keyword defines a boolean value. As the where clause defines constraints in terms of the contents of the database relation, the when clause defines time constraints that should be met.

The temporal operators, such as overlap, precede, extend, and equal, can determine two time values defining an interval while the operators, begin of and end of, determine a single time. For example, the predicates **(a overlap b)** and **(a extend b)** are shown in Figure 2.



(Fig. 2) Example of the overlap and the extend.

The query, "list all status of the plane, CessI" may be expressed as follows.

```
range of f is airplanestatus
retrieve (f.all)
  where f.plane='CessI'
  valid from begin of f to end of f
```

This query doesn't have a rollback predicate, but all of the tuples that satisfy the predicate of the where clause will be shown for the above temporal relation, airplanestatus in Figure 1, as follows:

plane	*model*	*status*	*from*	*to*
CessI	414A	inrepair	8:00	10:30
CessI	414A	onground	10:30	12:00

(Fig. 3) Result for executing the above query.

Information in temporal database relations can be represented with the graphical characteristics. The mapping of each attribute in the tuples onto characteristics of graphical objects is described in terms of a graphical schema. Each graphical object corresponds to a certain tuple in a temporal database relation. The graphical object is defined to a certain shape and is modified by modifiers depending on the value of attributes in each tuple.

We use the example in Figure 1 to show the graphical schema. The relation airplanestatus shows the status of the planes between **8:00 am** and **12:00 pm** on a certain day. This information can be given from the temporal query.

The temporal query statement "create" is used for graphical schema. This statement has to be defined prior to being used in any represent statement. For example,

```
create interval airplanestatus (
  plane is char,
  model is char,
  status is char) key plane
```

In this example, the interval relation *airplanestatus* has three attributes which are interpreted as *char* when reading an instance file. The key of the relation is an attribute plane.

While describing graphical characteristics for each tuple, you can assign arbitrary variables to a tuple of a relation. Range statements assign a variable to arbitrary tuple of the relation. The range statement in temporal query language may be used.

```
range of f is airplanestatus
```

The variable f represents an arbitrary tuple in the relation airplanestatus. Tuples in a relation have a basic shape, which can be defined by a represent statement. For example,

```
represent f with icon plane
  position (10.00,10.00)
  xsize 80 ysize 80
  text f.plane at (37.00,50.00)
  xsize 4 ysize 4
```

In this represent statement the tuples in the relation *airplanestatus* are represented with the icon *plane*, which is defined in the separate file icon, and text, which is the same characters of attribute plane of the tuple.

The basic shape of graphical objects can be modified with a modifier. Modifiers are specified with represent statements, but with a slightly different syntax. The basic idea is that a certain value of an attribute in a tuple can modify the basic shape. Also the represent statement can be modified such as shape that icons are overlapped like position, color, and size. For example,

```
represent f.status="inrepair"
  with yposition 1.00
  color blue
  icon garage position(5.00,5.00)
  xsize 90 ysize 90
```

Sometimes the domain of an attribute may not be finite. The modification depending on an infinite domain can be specified. The graphical display system counts the number of instances and assigns a value between specified minimum and maximum.

Graphical objects on the screen can be positioned as follow. The graphical screen is divided into grid of squares. The screen coordinates define the coordinates of the lower left and upper right grids. By default, the screen coordinates are assigned to (0,0) (80,60), it means that upper right corner of the screen has the x coordinate(horizontal) value 80 and y coordinate(vertical) value 60. Each graphical object usually occupies one grid square, and its position on screen is represented by the coordinate(x position, y position) of their lower left corner. The size of objects can be replaced by changing the characteristic x size and y size of objects.

The time domain of the relation *airplanestatus* can be represented in several different ways, such as animation, clock face, position, size, and intensity [Han88, Sha86]. In this example, animation and clock face are used to show time. By animation, the following example shows the status of the relation at a certain time on the screen, and advances the time by updating the screen.

```
represent time with animation
  clockface icon position (6.00,5.00)
  step by 30 mignutes
  speed 0
  mode stop
  ordering forward
```

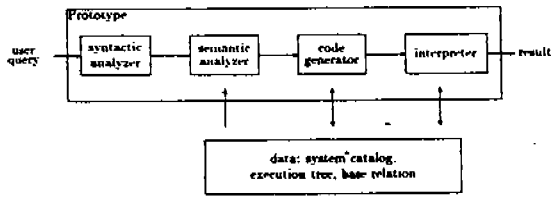
The above shows the status of the temporal relation for each 30 minute interval and asks user to advance the time and update the screen according to the temporal relation and modifier.

3. Interfacing System Architecture

The interfacing system consists of a temporal DBMS, a graphical display system, and an interface control system.

3.1 Temporal DBMS

The temporal DBMS is a temporal database management system which executes the temporal query. The temporal DBMS consists of a *syntatic analyzer*, a *semantic analyzer*, a *code generator*, an *interpreter*[Rryu91A, Ryu93] as shown in Figure 4. A query processing is executed by the four phases of the temporal DBMS.



(Fig. 4) The temporal DBMS architecture.

The temporal query for the temporal DBMS may be read from the keyboard. A parse tree of the temporal query which describes the query syntax is analyzed by the syntactic analyzer. The syntactic analyzer builds a parse tree for the input query, and the parse tree is then checked for correctness against the system catalog by the semantic

analyzer, as in conventional query processing. Its instance is composed by the input query. The query in the parse tree is classified by a *range*, *create*, *query*, and *quit* statement in the parse tree. The range statement describes a tuple variable. A parse tree may have one or more range statements. The create statement contains the type of database relation to be created such as *snapshot*, *rollback*, *historical*, and *temporal*. The historical relation can be divided into an interval or event historical relation. Similarly, the temporal relation can be also classified as an interval or event temporal relation.

A query of the user may have four different commands: *retrieve*, *append*, *delete* and *replace*. They can include two kinds of temporal clauses and a conventional predicate, i.e, *where*, *valid*, and *when*. If the output data from the semantic analyzer is semantically correct, the parse tree is then passed to the code generator.

The code generator builds an execution tree like an update network depending on the query command. The execution tree consists of nodes: *cartesian product*, *selection*, *derivation*, *projection*, and *store* or *display*. The terminal node(s) of inverted trees which is the execution tree generated by the code generator, which may be shared among several trees, represent the existing relations. Each root node of these inverted trees represents a derived relation, the result of a specific query.

Finally, the interpreter of the temporal DBMS evaluates the query execution plans in batch fashion. The interpreter should activate each node of the execution tree whenever a query is issued. The interpreter processes tuples from one or more relations for a restricted subset of TQuel queries.

A forest of execution trees for representating the TQuel query is an instance of an IDL data structure built by the code generator. Input to the interpreter is the set of tuples to be processed and the instances representing one or more queries. The set of tuples can be referred to the data structure for the execution tree.

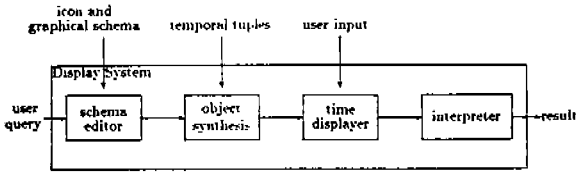
3.2 Graphical display system

The graphical display system is a system which shows temporal tuples in graphical form as described by a graphical schema. The graphical display system consists of a *schema editor*, an *object synthesis*, a *time displayer*, and an *interpreter*[Han88,Ryu91B, Sha86] as shown in Figure 5.

The graphical schema editor reads an icon file and a description of graphical schema, and then translates them to a knowledgebase of one or more graphical objects. Another part is the generator of graphical object. The generator of graphical objects is composed of three phases, which are object synthesis, time displayer, and interpreter.

The input of the graphical display system can be classified into four categories: temporal database instances, the graphical schema, icon description, and user input in terms of time displyer. Temporal database instances are ASCII files in the IDL external representation which are generated by the temporal DBMS with temporal query input as described in the previous section. The descriptions of graphical schema specifies how to translate values of attributes into graphical features of a graphical object. The icon de-

scriptions specifies what the icons look like. Icons that are used in the graphical schema are specified separately from the graphical schema, because icon descriptions have a different format of graphical schema. The graphical schema may be read from the schema editor or entered by keyboard.



(Fig. 5) The graphical display system architecture.

The schema editor reads descriptions and translates them into the internal form called the graphics knowledgebase. The graphics knowledgebase is a template used to convert tuples into graphical objects. The object synthesis builds a data structure called the object frame for the graphical object. Graphical objects are synthesized from tuples selected by the temporal query in the temporal DBMS and the template by the object synthesis of the graphical display system. Users can modify the graphics knowledgebase by using schema editor commands through a schema editor. The graphics knowledgebase made by users can be saved in a file for later use. The time displayer manages the display of time. It displays the object depending on the order of the specified representation for time. Finally, the interpreter interprets the objects in order to draw the picture on the frame buffer of a sun workstation.

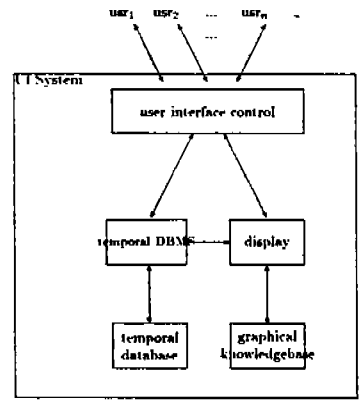
3.3 Interface control system

The user interfacing technique enables the user to communicate with a system conveniently through a terminal. Through interfacing a user can employ several systems as if they were one. The interface control system is the user interface system which integrates the temporal DBMS and the display system. The interfacing system should control the temporal DBMS and the graphical display system according to a request of the user, as shown in Figure 6.

After the temporal DBMS and graphical display system is activated, the command is sent to them. In order to achieve interfacing, we have the following scenario;

(1) activate the temporal DBMS and the graphical display system,
(2) read user command, and convert it to proper form,
(3) then send it to the temporal DBMS and/or the graphical display system.

Since the temporal DBMS is different from the display system, user commands or statements may be changed to proper form. For example, *represent* statements in the graphical display system specify a shape of graphical object or *modifier* to modify the basic shape



(Fig. 6) The interface control system structure.

of graphical object. However, *represent* statements in the temporal DBMS prepare to generate tuples for the display system. Therefore, the interfacing system may convert user command or statements to proper form and send it to the temporal DBMS and/or the graphical display system. The user interface system covers several commands for the temporal query of the temporal DBMS and the schema editor of the graphical display system.

(Fig. 6) The interface control system structure.

The statements, create, retrieve, append, and delete, apply to temporal queries. When composing temporal queries except for create statement, the query must contain a where clause and a valid and/or when clause. The relation, represent, and set, commands are schema editor commands[Han88,Ryu91B, Sha86]. The represent statement defines that temporal tuples have a basic shape, and allow modification of their shape at once. Several represent statements can be defined.

4. Conclusion

We have not only designed user interface system, but also implemented them. They consisted of an interface control system, a temporal DBMS, and a graphical display system.

The user interface control system activates the temporal DBMS and the graphical display system. The temporal DBMS is a temporal database management system which consists of a syntactic analyzer, a semantic analyzer, a code generator, and an interpretor. The temporal DBMS can support the times such as transaction time and valid time. The display system which is a graphical display system consists of a schema editor, an object synthesis, a time displayer, and an interpreter, which shows the graphical shape according to time varying.

The user interface system prompts the user for commands. When the user enters a command, it sends the command to the system software, according to the scenario of Chapter 3. Internally, the user interface control system activates the temporal DBMS, and then the temporal DBMS produces the temporal tuple instances. The temporal tuples are either displayed as the result of the user query, or sent to the graphical display system to draw the picture with a graphical knowledgebase.

Acknowledgements

The author, Keun Ryu, wishes to thank Richard Snodgrass for his valuable suggestions. Dr. Richard Snodgrass is a leader of TempIS Project. The part of this work was performed while Keun Ryu worked as a TempIS member at the University of Arizona.

References

[Han 88] T. Han. Display Users Manual and Tutorial. Technical Report TempIS Doc. 20, University of North Carolina at Chapel Hill, 1988.

[KeP 84] Brian Kernighan and Rob Pike. The UNIX Programming Environment. Prentice-Hall Inc., 1984.

[LeR 93] E. Lee and K. Ryu. TempIS: Interfacing System for Graphical Display of Temporal Databases. IEEE Tencon 93 Oct. 1993 pp 319-322.

[McK 88] E. McKenzie. An Algebraic Language for Query and Update of Temporal Database. Technical Report 88-050, University of North Carolina at Chapel

Hill, 1988.

[Ryu 91A] K. Ryu. A Temporal Database Management Main Memory Prototype. TempIS Technical Report No. 26, The University of Arizona, 1991.

[Ryu 91B] K. Ryu. Users Manual and Tutorial for TempIS User Interface. TempIS Tecnical Report No.27, The University of Arizona, 1991.

[Ryu 93] K. Ryu. Execution Tree for Incremental View Materialization of Temporal Database. J. of KISS Vol.20, No.8, Aug. 1993 pp 1167－1178.

[Sha 86] K. Shannon. The Display of Temporal Information. Technical Report 86－109, University of North Carolina at Chapel Hill, 1986.

[Sno 87] R. Snodgrass. The Temporal Query Language TQuel. ACM TODS, 12(2): 247－298, June 1987.

[Sno 89] R. Snodgrass. The Interface Description Language: Definition and Use. Computer Science Press, 1989.

이 언 배

1974년 서울시립대학교 졸업(학사)
1982년 동국대학교 경영대학원 전산전공(석사)
1992년－충북대학교 컴퓨터과학과(박사과정)
1974년～1983년 총무처 정부전자계산소 근무
1983년～현재 한국방송통신대학교 전자계산학과 부교수로 재직중
관심분야 : 데이터베이스, 소프트웨어공학 등.

류 근 호

1976년 숭실대 전산학과 졸업
1980년 연세대 산업대학원 전산전공(공학석사)
1988년 연세대 대학원 전산전공(공학박사)
1976년～1986년 육군 군수 지원사 전산실(ROTC장교), 한국 전자 통신연구소(연구원), 한국방송통신대 전산학과(조교수) 근무
1989년～1991년 Univ, of Arizona 연구원
1986년～현재 충북대학교 컴퓨터과학과 부교수
관심분야 : 시간지원 데이타베이스, 시공간 데이타베이스, DBMS 및 OS, 객체 및 지식베이스 시스템 등임