MANIPULATION OF TEMPORAL OBJECTS WITH IMPRECISE ATTRIBUTES AND TIME PARAMETERS

Ed Porto Bezerra*, Ulrich Schiel+ e Edilson Ferneda+

*Departamento de Informática
Universidade Federal da Paraíba
João Pessoa - Brazil

+Departamento de Sistemas e Computação
Universidade Federal da Paraíba
Campina Grande - Brazil
e-mail: {ed, ulrich, edilson}@dsc.ufpb.br

1 - INTRODUCTION

The notion of time is present in most advanced activities. Facts occur dynamically over time, which becomes a basic reference for objects stored in a database. In spite of this, actual database systems and information systems do not treat the time dimension adequately [Vi94].

In several applications the information disposable is incomplete or imprecise. If the system includes temporal information, by means of a temporal database, also the temporal information may be incomplete. We present here a prototype which stores and retrieves incomplete data and temporal information. We show how the system treats several types of queries and how monotonic and non-monotonic updates are performed.

1 - INTRODUCTION

The notion of time is present in most advanced activities. Facts occur dynamically over time, which becomes a basic reference for objects stored in a database. In spite of this, actual database systems and information systems do not treat the time dimension adequately [Vi94].

In several applications the information disposable is incomplete or imprecise. According to Moles [Mo95] it is better to know something instead of nothing. It is fundamental, in particular for database systems, the treatment of incompleteness, supporting queries and updates. Dyreson [Dy96] states that treatment of imprecision means the representation and query support for information that can be unknown, fuzzy, partially known, vague, uncertain, probabilistic, indefinite, disjunctive, possible, incomplete, approximate, erroneous or imprecise.

Research about the treatment of incompleteness continues in evidence [Li79, IL84, LS90, Mo90, Be91, GNP92, AHV95, Dy96], but little has been done about incompleteness on the temporal information [Ko94, GT96].

There are mainly two possible interpretations of a database. The closed world assumption - CWA means that all relevant information is stored in the database, i.e. if a fact can not be derived from the database, it is assumed to be false. By the open world assumption-OWA a fact is false only if its negation can be derived from the database. Therefore facts not stored in the database and not derivable from the existing data are considered false in the CWA and unknown or possible in the OWA. The CWA is not adequate for systems considering some kind of imprecision. On the other hand, with the OWA things that are not true must explicitly declared or stored as false. This kind of information is called negative information.
We distinguish three types of incomplete information: unknown values, partial or imprecise information, and negative information. A similar distinction apply to the temporal information.

This paper represents a continuation of the work done in [SC86, Or88]. The original prototype has been re-implemented and several improvements has been done, both in theoretical as in implementation concerns.

The most significant improvements are a complete revision on the processing of incomplete temporal intervals a the adding of an update component, including consistency checking, not present in the original version.

2 - REPRESENTATION OF INCOMPLETE OBJECTS

An object is something that makes sense in the context of an application and is distinguishable from other objects [RBP+94].

Incomplete objects are those who have some attribute with incomplete value [Zi90]. Temporal objects have a time associated describing the lifetime of the object and, finally, incomplete temporal objects are incomplete objects whose time attribute is also incomplete.

We show, below, or guiding example with several imprecisions which will become clear during the text. The example shows a set of objects of type employment relating employees with companies and the corresponding salary.

\[
\begin{align*}
\text{employment(1)} & : \text{employee} = \text{Antonio OR João}, \\
& \quad \text{company} = \text{UFRN OR UFPB}, \\
& \quad \text{salary} = 1600, \\
& \quad \text{time} = (1979, 1982). \\
\text{employment(4)} & : \text{employee} = \text{João}, \\
& \quad \text{company} = \text{UFPB}, \\
& \quad \text{salary} = >1500, \\
& \quad \text{time} = (1994, \text{now}). \\
\text{employment(2)} & : \text{employee} = \text{João}, \\
& \quad \text{company} = \text{UFPE}, \\
& \quad \text{salary} = \text{interval}(2001, 2999), \\
& \quad \text{time} = (1983, \text{now}). \\
\text{employment(5)} & : \text{employee} = \ast 1, \\
& \quad \text{company} = \text{UFPB}, \\
& \quad \text{salary} = \text{1500}, \\
& \quad \text{time} = (1970, 1990). \\
\text{employment(3)} & : \text{employee} = \ast 1, \\
& \quad \text{company} = \text{not UFPB}, \\
& \quad \text{salary} = \text{interval}(2000, 2999), \\
& \quad \text{time} = (11/1982, \text{after}(1994)). \\
\text{employment(6)} & : \text{employee} = \text{José}, \\
& \quad \text{company} = \text{UFRN}, \\
& \quad \text{salary} = \ast, \\
& \quad \text{time} = \text{not}(1970, 1981).
\end{align*}
\]

The three types of incomplete information are the following.

UNKNOWN VALUE: an unknown value (also called null value [Co79]) is here represented by an asterisk and means that the corresponding attribute holds for the object, but we do not know anything about his value. The asterisk can be marked (e.g. *1) [KW85, IL84] or not (*) [Mo90]. If a marked asterisk appears in more than one object, it represents the same (unknown) value. For instance \text{employment( joão, \ast 1, _ \)
and employment(josé, *I, _ ) means that joão and josé are working for the same unknown company.

IMPRECISE VALUE: in this case we have some information about the attribute but not a precise value. The value may be given as a disjunctive OR-list of possible values, or a numeric domain of the possible values. For instance, in our example, employment(1) has two OR-imprecisions, employment(4) states that João’s salary at UFPB is greater than 1,500, and employment(2) represents a salary of João between 2,000 and 3,000.

NEGATIVE VALUE: the negation is necessary in an OWA approach, in order to state what information is false. We can negate specific attribute values (e.g. company=not(UFPB)), the existence of the attribute for an object (e.g. salary=not means that this employment does not evolve a salary), the time interval (e.g. employment(6) states that José worked for UFRN but not during (1970-1981), or negate the whole tuple (not treated in the actual version of the prototype).

3 - REPRESENTATION OF TEMPORAL INFORMATION

A pioneering work about temporal modeling and representation is that of J. Allen [Al83, Al84]. He has defined an interval calculus based on 13 relations that correspond to mutually exclusive possible relations between two time intervals. [Al83]. In [Al84] he identified three kinds of entities in a temporal system: Facts, events and processes. As should be seen, we use several elements of the interval calculus and model Allen’s facts and processes. In a later extension, also the (long-duration) events of Allen will also be included.

There are three ways in defining a logic of time [Vi94]: a first order logic with time parameters, modal temporal logic, and a materialized temporal logic. The analysis of this forms is beyond the scope of this paper, but we show how formulas of the modal temporal logic of Prior [Pr55] can be represented in our system. Prior has defined a logic with future and past with the following operators:

\[ F\phi \equiv \phi \text{ is true sometimes in the future} \]
\[ P\phi \equiv \phi \text{ is true sometimes in the past} \]
\[ G\phi \equiv \phi \text{ is true always in the future} \]
\[ H\phi \equiv \phi \text{ is true always in the past} \]

where \(\phi\) represents a generic formula. For instance \( F \text{ dance}(João, Maria, valtz)\) means that in the future João will dance valtz with Maria, and \( F(P \text{ dance}(João, Maria, valtz))\) means that the valtz between João and Maria should be happen.

In our approach temporal objects have, beyond their common attributes, a special time attribute, which specifies the lifetime of the object. This time attribute can be given by a pair of values (given by constants or time operators) specifying the beginning and end of the time interval I of the object, or a single time value. This value can be a constant in a higher granularity as the basic one and consequently determines an interval or a temporal operator defining an imprecision domain. The temporal operators and their meaning is given in the following Table 1. We distinguish between time instants t, which are times in the lowest granularity of the system and time intervals I.
OPERATOR | MEANING
--- | ---
constant | the fact is valid during the time of the constant
before(t) | the fact has been true sometimes before t
after(t) | the fact will be true sometimes after t
during(I) | the fact will be true sometimes during I
* | the time is unknown

<table>
<thead>
<tr>
<th>OPERATOR</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>the fact is valid during the time of the constant</td>
</tr>
<tr>
<td>before(t)</td>
<td>the fact has been true sometimes before t</td>
</tr>
<tr>
<td>after(t)</td>
<td>the fact will be true sometimes after t</td>
</tr>
<tr>
<td>during(I)</td>
<td>the fact will be true sometimes during I</td>
</tr>
<tr>
<td>*</td>
<td>the time is unknown</td>
</tr>
</tbody>
</table>

**Table 1 - temporal operators**

Therefore, the dance between João and Maria is modeled in our system as

<table>
<thead>
<tr>
<th>Dance(1)</th>
<th>men = joao,</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman = maria,</td>
<td></td>
</tr>
<tr>
<td>rhythm = valets</td>
<td></td>
</tr>
<tr>
<td>time = after(now).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dance(2)</th>
<th>men = joao,</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman = maria,</td>
<td></td>
</tr>
<tr>
<td>rhythm = valtz,</td>
<td></td>
</tr>
<tr>
<td>time = before(after(now)).</td>
<td></td>
</tr>
</tbody>
</table>

The use of indefinite time operators at the endpoints of the time attribute of an object gives rise to several truth values on the time axis. The predicates of Table 2 are used by the system.

**Table 2 - predicates over the time line**

This predicates apply to intervals <X,Y>, where X is the starting point of the interval and Y the ending point. The intervals can be:

- [X,Y] closed interval
- (X,Y) open interval
- (X,Y] semi-open interval left
- [X,Y) semi open interval at right

Now we show the table used in the construction truth value assignments on the time axis. It was originally created by Oresotu [Or88], and was reformulated and corrected here. X and Y are the endpoints of the time interval.

<table>
<thead>
<tr>
<th>X\Y</th>
<th>constant</th>
<th>before(Y)</th>
<th>after(Y)</th>
<th>during(Y)</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>S(X,Y)</td>
<td>P(Y+1, +∞)</td>
<td>S[X,Y]</td>
<td>P(X, +∞)</td>
<td></td>
</tr>
</tbody>
</table>
Table 4 - values of the time axis for indefinite time intervals

For a better understanding of the table, suppose a time attribute 'time= before(1990), during(1992)'. The validity of the object has started sometimes before 1990 and ended in 1992. Graphically, this can be shown as

-∞ ________________ | 1990 | _______ | 1992 | __________________ 

and the resulting list of truth values is

[P(01/01/0001, 30/12/1989), A(31/12/1989, 31/12/1991), S(01/01/1992, 31/12/1992)]

The part of the time axis not appearing in the list is considered unknown.

4 - OPERATIONS OVER THE DATA BASE

In general, we distinguish query operations and update operations.

4.1 - QUERY OPERATIONS

With a query the user of a system can retrieve data or get information about the existence or not of some specific data. A query is called **exact matching** [KB96] if the required data is in the database as it was specified by the query. We call a query **partial matching** if it is not exact matching but the domain space of the query is a subset of the domain space in the database.

The general form of a query is:

`object-name"("value,\{[,value]\}\")"[";"temporal-value["-"temporal-value]]\""`

Where `object-name` is the name of the corresponding object;

`value` can be one of the following:

- a constant
- an asterisk, an imprecise value or a reference to another value;
- the symbol (?) meaning that the value of this attribute should be retrieved;
• the symbol (_) meaning that the value if this attribute is not involved in the query
  temporal value corresponds to the temporal part of the query. It can be time value, time interval, both with some imprecision. A missing time interval means the value “now”.

Queries can be classified as **Boolean queries**, which return a truth value or retrieve queries which return some information stored in the database.

### 4.1.1 - BOOLEAN QUERIES

A Boolean query is characterize syntactically by the fact that it does not contain any “?” . The text of the query is matched against the database and this process returns one or more of the truth values: yes, no, possible or unknown. Each answer is connected to a time interval. Since there can be several objects in the database which matches partially with the query, independent of time, in a first step, several answers can be obtained which are the integrated.

Suppose the following query, which asks if João worked for UFPB with a salary greater than 1,500 during 1970 and 1995:

\[ \text{employment (João, UFPB, >1500);1970-1995.} \]

The objects employment(1) and employment(4) matches with the query, and return

**Possible**

\[
[U(01/01/1970, 31/12/1978), A(01/01/1979, 31/12/1982), U(01/01/1983, 31/12/1995)]
\]

**Yes**

\[
[U(01/01/1970, 31/12/1993), A(01/01/1994, 31/12/1995)]
\]

These answers are integrated to

**Unknown (01/01/1970 - 31/12/1978)**

**Possible (01/01/1979 - 31/12/1982)**

**Unknown (01/01/1983 - 31/12/1993)**

**Yes (01/01/1994 - 31/12/1995)**

### 4.2.1 - RETRIEVAL QUERIES

A retrieval query returns some information stored in the database. For instance, if we want to know who has worked for UFPB we state employment (?, UFPB, _). The object employment(1) returns Possible(João or Antonio) , whereas employment(4) and employment(5) retrieves Yes(João) and Yes(*1), respectively. Integrating this two answers we obtain Possible(Antonio) and Yes(João, *1).
4.2 - DATA MANIPULATION

We distinguish two kind of manipulations: the non-monotonic alteration, which does not respect the actual state of the database and the monotonic append, which considers the stored information and reduces incompleteness.

The difference appears in the strategy of integration of the new information in the database. We consider mainly update and insert operations since a delete is the same in the two cases. If an alteration generates a contradiction, the new information remains and the old is removed. If there are domains of imprecision, the alteration maintains only the new domain whereas the append maintains the intersection of the domains. If the value of the an append contradicts a stored value, the user can chose between canceling the update or consider the union of the two domains.

In the following table we show some examples of manipulation operations over a attribute salary:

<table>
<thead>
<tr>
<th>DATABASE</th>
<th>NEW VALUE</th>
<th>result with alteration</th>
<th>result with append</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 1000</td>
<td>≥ 500</td>
<td>≥ 500</td>
<td>interval(500, 1000)</td>
</tr>
<tr>
<td>≤ 1000</td>
<td>≥ 1500</td>
<td>≥ 1500</td>
<td>≤ 1000 OR not(1000, 1500)</td>
</tr>
</tbody>
</table>

Table 5 - examples alterations and appends

5 - IMPLEMENTATION ASPECTS

In this chapter we show the main procedures in processing temporal and non-temporal queries with incomplete information. We also comment the implementation environment.

Imprecise numeric and time values are represented as intervals while alphanumeric values are sets. Table 6 below shows some examples.

<table>
<thead>
<tr>
<th>ATTRIBUTE VALUE</th>
<th>REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 600</td>
<td>(-∞, 600)</td>
</tr>
<tr>
<td>before(1990)</td>
<td>(01/01/0001 31/12/1989]</td>
</tr>
<tr>
<td>interval(100-200)</td>
<td>[100,200]</td>
</tr>
<tr>
<td>1990</td>
<td>[01/01/1990, 31/12/1990]</td>
</tr>
<tr>
<td>* (numeric)</td>
<td>(-∞, +∞)</td>
</tr>
<tr>
<td>UFPB or UFPE</td>
<td>{UFPB, UFPE}</td>
</tr>
<tr>
<td>* (alfanumérico)</td>
<td>U (universe)</td>
</tr>
<tr>
<td>not (UFPB or UFPE)</td>
<td>U - {UFPB, UFPE}</td>
</tr>
</tbody>
</table>

Table 6 - examples of attribute representations

5.1 - PROCESSING OF THE TEMPORAL PART OF A QUERY

Several procedures where implemented for the processing of temporal information. We emphasize here the procedures TIME and VERIFY-INTERVAL.
TIME implements Table 4 above and uses special calendar procedures to compute the days after or before the limits, in order to determine the resulting time intervals.

VERIFY-INTERVAL creates a list of time intervals of the query and another list of time intervals of the matching objects from the database. Each time interval has associated one of the truth values always, sometimes, possible, never or unknown. Comparing the two time axis, from the query and from the database, the result is obtained from the logic of Table 7.

<table>
<thead>
<tr>
<th>DB \ Query</th>
<th>Always</th>
<th>Sometimes</th>
<th>Possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always</td>
<td>Always</td>
<td>Sometimes</td>
<td>Possible</td>
</tr>
<tr>
<td>Sometimes</td>
<td>Sometimes</td>
<td>Sometimes</td>
<td>Possible</td>
</tr>
<tr>
<td>Possible</td>
<td>Possible</td>
<td>Possible</td>
<td>Possible</td>
</tr>
<tr>
<td>Never</td>
<td>Never</td>
<td>Never</td>
<td>Never</td>
</tr>
<tr>
<td>Unknown</td>
<td>Unknown</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

*Table 7 - penta-valued logic*

Take, for instance, the Boolean query: `employment(João, UFPB, >1500); after(1992)-1995`. The object `employment(4)` matches with the query, and we have the following time values:

**Query:**

\[
\begin{array}{cccccccc}
-\infty & U & U & P & P & A & U & U +\infty \\
\end{array}
\]

**Object of the database:**

\[
\begin{array}{cccccccc}
& 1994 & & now & & & \\
-\infty & U & A & A & A & U & +\infty \\
\end{array}
\]

**Result:**

\[
\begin{array}{cccccccc}
-\infty & U & U & P & A & U & U +\infty \\
\end{array}
\]

resulting list: `[P(01/01/1994, 31/12/1994), A(01/01/1995, 31/12/1995)]`

The intervals before(1994) and after(now) are evaluated as Unknown and removed from the list.

For the query `employment(José, UFRN, _);1970-1986` the resulting list becomes: `[N(01/01/1970, 31/12/1981), U(01/01/1982, 31/12/1986)]` after evaluation against `employment(6)`. The value never is associated to the negated time domain not(1970,1986).
5.2 - PROCESSAMENTO DA PARTE NÃO-TEMPORAL DE UMA CONSULTA

Cada valor incompleto de um atributo estabelece um domínio de possíveis valores. Processar uma consulta com valores incompletos significa comparar o domínio do atributo da consulta (QAD) com o domínio do atributo de um objeto do banco de dados (OAD). A resposta a tal consulta é computada pelo algoritmo:

\[
\text{Algorithm} \quad \text{If} \quad \text{DAC} \cap \text{DAO} \neq \emptyset \\
\text{Then} \\
\text{If} \quad \text{DAC} \subset \text{DAO} \\
\text{Then} \quad \text{Return} \quad \text{“Possible” \{result 1\}} \\
\text{Else} \\
\text{If} \quad \text{DAC} \supseteq \text{DAO} \\
\text{Then} \quad \text{Return} \quad \text{“Yes” \{result 2\}} \\
\text{Else} \quad \text{Return} \quad \text{“Possible” \{result 3\}} \\
\text{Else} \quad \text{Return} \quad \text{“No” \{result 4\}}
\]

No seção subsequente exibimos como se obtêm esses domínios.

5.2.1 - NUMERIC ATTRIBUTES

Os valores incompletos numéricos podem ser intervalos, valores negados ou valores desconhecidos (*). Intervalos podem ser delimitados como interval(1600,2000) ou semi-abertos como (≤1500).

Tabela 7 mostra a comparação de uma consulta com salario=(1600,2000) e alguns valores do banco de dados.

<table>
<thead>
<tr>
<th>RESULT</th>
<th>SALARY VALUES OF THE OBJECT IN THE DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (possible)</td>
<td>&gt;1500, ≥1600, interval(1500,2100), ...</td>
</tr>
<tr>
<td>2 (yes)</td>
<td>1700, interval(1600,2000), interval(1700,1900), 1600 ou 1700,...</td>
</tr>
<tr>
<td>3 (possible)</td>
<td>&gt;1600, &gt;1700, interval(1700,2100), not(2000),...</td>
</tr>
<tr>
<td>4 (no)</td>
<td>&gt;2000, &gt;2200, &lt;1600, ...</td>
</tr>
</tbody>
</table>

Tabela 7 - exemplos de consultas sobre valores numéricos incertos

5.2.2 - NON-NUMERIC ATTRIBUTES

Nos valores incompletos não-numéricos têm implicações como disjunciões or’s, negações e valores desconhecidos (*). Suponha a consulta employment(João, UFPIB or UFPE, _). Tabela 8 mostra os resultados da aplicação do algoritmo com vários domínios de objetos do banco de dados.

<table>
<thead>
<tr>
<th>RESULT</th>
<th>OBJECT DOMAINS OF THE DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (possible)</td>
<td>UFPIB or UFPE or UFRN, *,...</td>
</tr>
<tr>
<td>2 (yes)</td>
<td>UFPIB or UFPE, UFPIB, UFPE, UFPE</td>
</tr>
<tr>
<td>3 (possible)</td>
<td>UFRN or UFPIB, not(UFPIB),...</td>
</tr>
<tr>
<td>4 (no)</td>
<td>UFRN, UFCE, not(UFPIB or UFPE),...</td>
</tr>
</tbody>
</table>

Tabela 8 - exemplos de consultas sobre valores não-numéricos incertos
6 - RELATED WORKS

There are some similarities between our work and that of Morrisey [Mo90]. He distinguishes two possible incomplete values. First when the disposable values are given by a list of possible values (p-domain). In the second case the imprecision is given by a variation between given limits (p-variation). For instance the salary is something between 1700 and 3500. Morrisey do not approaches negative values and temporal information.

There exist also a proposal of an extension of SQL, called SQL-i [GT96]. This proposal includes several kinds of imprecision and have an operation, called append, which convert a conventional update in monotonic actualization of the database.

7 - CONCLUSION AND FUTURE WORK

We are convinced about the great importance of the processing of incomplete information in information systems. By the growing importance of temporal databases this research must be extended to tread incomplete information adequately.

A prototype has been implemented in Prolog in order to validate our ideas. It processes two basic kinds of queries (Boolean and retrieval queries) and distinguishes between monotonic and non-monotonic updates. Boolean queries generates, temporally dependent, answers from the set of truth values {yes, no, possible, unknown}.

Special predicates, such as, before and during allows a correct treatment of temporal information.

We conclude with a partial list of extensions planned for the next future:

a) consequences of incorporation of incomplete information in complex objects
b) creation of conditional objects, by the inclusion of a condition attribute. This condition may be NOT, IF <object>, or OR<oid>. For instance, the object

```
employment(7) : employee = João,
company = UFPE,
salary = 1,000,
time = (1985, now),
condition = IF employment(Pedro, UFPE, _); 1980-1984
```

means that João works for UFPE since 1985 if Pedro has worked for the same university during 1980-1984. A NOT condition negates the whole tuple.

c) a more detailed study of the logical implications of our formalisms, in particular the combination of not and unknown values.

8 - REFERENCES


