Fuzzy Intervals to Represent Fuzzy Valid Time in a Temporal Relational Database

Carmen Garrido
Junta de Andalucia, Spain
cglupi@gmail.com

Nicolás Marín, Olga Pons
IdBIS Group, Dept. Computer Science and A.I.
University of Granada, Granada, Spain.
{nicm, opc}@decsai.ugr.es

Received (received date)
Revised (revised date)

Temporal databases offer a common framework to those database applications that need to store or handle different types of temporal data from a variety of sources. They allow the concept of time to be handled from the point of view of meaning, representation, and manipulation. Although at first sight the incorporation of time into a database might appear to be a direct and simple task, it is, however, quite complex: not only must new structures and specific operators be included, but the semantics of conventional DML sentences (insert, update, or delete) and queries must be appropriately changed. In addition, temporal information is not always as precise as desired since it might be affected by imprecision due to the use of natural language or to the nature of the information source. In this paper, we deal with the problem of the update (and, implicitly, insert and delete) and query operations when time is expressed by means of a fuzzy interval of dates.

Keywords: Fuzzy Data, Temporal Database, Fuzzy Interval, Data Manipulation

1. Introduction

Temporal databases (TDB), in the broadest sense, offer a common framework for all database (DB) applications that involve temporal aspects when organizing data. These databases allow the time concept to be unified in terms of representation, semantics, and manipulation.

Although there is nothing new about database applications involving temporal data and they have in fact been developed since relational databases were first used, application programmers were responsible for designing, representing, programming, and managing the necessary temporal concepts.

One of the first formalizations of the concept of granularity in the time domain can be seen in \cite{6}, but the approach to this problem from the DB and the temporal reasoning points of view came later and was first studied in depth in the nineties. In
this sense, the papers by Duncan, Goralwalla and Bettini are very interesting because the concept of calendar is introduced and the definition of granularity is extended.

Up until now, most works (both theoretical and implementations) carried out on this topic have used the relational DB model as a starting point (since it is the most complete and consolidated model) with the results based on extension of the table schemata, the range of operators to be used, and on the addition of specific integrity constraints related to the new data types.

There are other interesting proposals about the architecture of a system that supports the time concept, as can be seen in. In line with the theoretical results obtained, much effort has therefore been devoted to achieving a temporal language either as an extension of SQL or of Quel.

Other approaches to temporal databases deal with the problem of imperfect temporal data in the sense that we do not know exactly when an event happened but we have approximate information about it. In this problem is tackled from the probability theory point of view by assigning a probability distribution to the set of possible instants. The authors call this interpretation valid-time indeterminacy. The main inconvenience of this proposal is that in many cases it is very difficult (or even impossible) for the user to give the underlying mass function that associates a probability to every time point. More over, if the user does not provide the mass function, the method proposed is useless and some important information provided by the user is wasted. This problem is also recently dealt with by attaching a probability distribution to every point.

On the other hand, certain authors working in the area of soft computing have opted to study temporal data affected by imprecision from the possibility theory point of view, which is more flexible and better represents users’ appreciations. In the case of relational databases (as in our case), Kurutach’s paper presents a preliminary study of this topic with a formal representation of fuzzy-temporal data and various query-related operators. Nevertheless, crucial operations such as insert, update, and delete (which have strong constraints) are not addressed.

A significant survey of papers relating to spatio-temporal databases can be found in. Time has also been dealt with in the scope of datawarehouses, where it is considered a dimension. More specifically in and, the fuzzy sets theory is used to perform temporal interpolation in a GIS object-oriented database. In the problem of fuzzy time is also addressed in an object-oriented environment, which is beyond the scope of this paper.

In summary, there is a wide range of interesting proposals on this topic but there are still many important problems to be solved, such as the ones being tackled in this paper.

Additionally, it is not always possible for the user to give an exact but an imprecise starting/ending point for a fact validity period. In this case, fuzzy set theory is a very suitable tool for not missing such information since fuzzy time
values can be represented and managed.

In this sense, this paper proposes an approach to represent and interpret fuzzy valid time in temporal databases and describes how to properly adapt the UPDATE and the SELECT operations when the time is expressed in fuzzy terms.

The paper is organized as follows. The second section explores the preliminary concepts and includes a brief summary of temporal databases, fuzzy sets, and fuzzy database operators. Section 3 introduces two particular ways of representing and interpreting fuzzy time intervals from two fuzzy dates. Section 4 examines the manipulation of fuzzy temporal data focusing on the update operator since delete and insert are particular cases of it. In order to complete the DML, Section 5 describes different types of queries on a fuzzy TDB and explains how to compute a final fulfillment degree for the selected tuples. Finally, Section 6 presents some of our conclusions and indicates some open points for future lines of research.

2. Preliminaries

In this section we will introduce some previous concepts on classical temporal databases, give a brief explanation of fuzzy numbers, and summarize the main operators of the Fuzzy SQL language, since all this background form the basis of this paper.

2.1. Temporal Databases

TDB provide suitable data types and operators for handling time. In terms of the TDB, time is an application-based ordered sequence of points of the same granularity; in other words, one of several measurement units will be used depending on requirements (i.e. a second, a day, a week, etc.).

From the point of view of the real world, there are basically two ways to associate temporal concepts to a fact:

(1) Punctual facts: a fact is related to a single time mark that depends on the granularity and provides information about the time when it happened (e.g. instances, birthdays, the date of purchase, an academic year, etc.).

(2) Time periods: a fact is related to a period represented by a starting and an ending instant and so the duration (or valid time) of the fact is implicit (e.g. [admission date, discharge date], [contract start date, contract end date], etc.).

The time dimension may appear with many semantics according to the problem to be represented. In many situations, time periods are used to express the validity of the data representing a fact. This way of interpreting time is called valid time.

However, in a TDB, apart from the valid time, another interesting use of time dimension is to reflect the instant when the fact was stored that may be calculated using the system time, and it is called transaction time. Some applications only use one of the two dimensions, while others use both of them. When this is the
case, the TDB is called a bitemporal database. In order to take into account one interpretation, or both of them, table schemata in the TDB must be extended with specific attributes, as shown in next figure for table EMP (employees).

When only using the valid time interpretation, the schema must be extended in order to include the attributes VST (Valid Start Time) and VET (Valid End Time) and a valid time relation (VTR) is obtained.

When only using the transaction time interpretation, the schema must be extended in order to include the attributes TST (Transaction Start Time) and TET (Transaction End Time), and a transactional time relation (TTR) is obtained. If both interpretations are necessary at the same time, then all attributes previously mentioned have to be included into the schema. In the rest of the paper we will focus on the valid time approach, since we are dealing with imprecise data and system timestamps are always precise.

Let us consider the following example of the time relation EMP, where each tuple represents a given version of the available information about an employee, and every version is valid only when used in its time interval [VST,VET]. The current version (also called the valid tuple) takes the special null value in the attribute VET since the interval is still open. In the Table 1 an instance of EMP is shown.

<table>
<thead>
<tr>
<th>EMPNAME</th>
<th>EMPID</th>
<th>SALARY</th>
<th>BOSS</th>
<th>EXPERTISE</th>
<th>VST</th>
<th>VET</th>
</tr>
</thead>
<tbody>
<tr>
<td>REDFORD</td>
<td>9877</td>
<td>1200</td>
<td>4588</td>
<td>TRAINER</td>
<td>20-08-1994</td>
<td>31-01-1996</td>
</tr>
<tr>
<td>REDFORD</td>
<td>9877</td>
<td>2200</td>
<td>9877</td>
<td>JUNIOR</td>
<td>01-02-1996</td>
<td>31-03-1997</td>
</tr>
<tr>
<td>BROWN</td>
<td>1278</td>
<td>2800</td>
<td>9877</td>
<td>JUNIOR</td>
<td>01-05-2005</td>
<td>10-09-2008</td>
</tr>
<tr>
<td>STREEP</td>
<td>6312</td>
<td>4000</td>
<td>9877</td>
<td>TRAINER</td>
<td>10-06-1994</td>
<td>Null</td>
</tr>
</tbody>
</table>

Table 1. Instance of EMP valid-time relation

Some consequences are derived from this extension of schemata with new attributes:

- In a VTR, the old primary key ceases to be unique (in general, this happens with any candidate key). The new primary key is the result of combining the previous value for the key and the valid start time (VST) since it is never set to null. In the case of Table EMP, the primary key is not EMPID (employee’s code) but EMPID+VST.
- There is only one valid tuple for each entity at a given time. Every operation must be strictly controlled so that the valid time periods of the same entity do not overlap.
- Internal implementation of common operations is completely different from those implemented in a non-temporal database. For instance, when an UPDATE is required, the current version of the tuple is closed and a new version of it is created with the modified data. In this case, the user is responsible for giving the valid time; when closing the active version, the value of the attribute VET is updated with the previous grain to the value of the VST of the new
version. Similarly, the DELETE operation is carried out by closing the active version (i.e., updating the value of the attribute VET) whereas the INSERT operation involves creating a new tuple, the valid one.

Additionally, it is not always possible for the user to give an exact but an imprecise starting/ending point for a fact validity period. In this case, the fuzzy set theory is a very suitable tool for not missing such information since fuzzy time values can be represented and managed. In possibility theory is used as a general framework for modeling temporal knowledge pervaded with imprecision or uncertainty from a theoretical point of view. Also, in an extension of the SQL language for managing time is presented (FTSQL2).

This paper explores the representation and interpretation of fuzzy valid time together with the UPDATE and the SELECT operations when the time is expressed in fuzzy terms. It should be noted that the study of the UPDATE operation also includes the DELETE and the INSERT operations as particular cases.

2.2. Fuzzy Numbers

A fuzzy value is a fuzzy representation of the real value of a property (attribute) when it is not precisely known.

In this paper, following Goguen’s Fuzzification Principle, we will call every fuzzy set of the real line a fuzzy quantity. A fuzzy number is a particular case of a fuzzy quantity with the following properties:

Definition 1.-
The fuzzy quantity \(A\) with membership function \(\mu_A(x)\) is a fuzzy number if:

1. \(\forall \alpha \in [0, 1], \; A_\alpha = \{x \in R \mid \mu_A(x) \geq \alpha\}\) is a convex set.
2. \(\mu_A(x)\) is an upper-semicontinuous function.
3. The support set of \(A\), defined as \(\text{Supp}(A) = \{x \in R \mid \mu_A(x) > 0\}\), is a bounded set of \(R\), where \(R\) is the set of real numbers.

We will use \(\tilde{R}\) to denote the set of fuzzy numbers. For the sake of simplicity, we will use capital letters at the beginning of the alphabet to represent fuzzy numbers.

The interval \([a_\alpha, b_\alpha]\) (see figure 1) is called the \(\alpha\)-cut of \(A\). So then, fuzzy numbers are fuzzy quantities whose \(\alpha\)-cuts are closed and bounded intervals: \(A_\alpha = [a_\alpha, b_\alpha]\) with \(\alpha \in (0, 1]\).

If there is, at least, one point \(x\) verifying \(\mu_A(x) = 1\) we say that \(A\) is a normalized fuzzy number. We use \(h(A)\) to denote the height of the fuzzy number \(A\), that is, \(\text{Max}_x\{\mu_A(x)\}\).

Sometimes, a trapezoidal shape is used to represent fuzzy values. This representation is very useful as the fuzzy number is completely characterized by four parameters \((m_1, m_2, a, b)\) as shows figure 2 and the height \(h(A)\) when the fuzzy
value is not normalized. We will call modal set all values in the interval \([m_1, m_2]\), i.e, the set \(\{x \in \text{Supp}(A) \mid \forall y \in \mathbb{R}, \mu_A(x) \geq \mu_A(y)\}\). The values \(a\) and \(b\) are called left and right spreads, respectively.

In our approach, we will use trapezoidal and normalized fuzzy values.

### 2.3. FSQL (Fuzzy SQL)

The FSQL language\(^{15}\)\(^{17}\) extends the SQL language in order to express flexible queries. Due to its complex format, we only show here an abstract with the main extensions to the select command that we will use later in this work.

- Fuzzy values, represented by trapezoidal functions, that can be considered from the disjunctive and conjunctive points of view.
- **Fuzzy Comparators**: Besides the typical comparators (\(=\), \(>\)..., FSQL includes the fuzzy comparators shown in table 2. Like in SQL, fuzzy comparators compare one column with one constant or two columns of the same type.
- **Fulfilment thresholds** (\(\gamma\)): In queries, for each simple condition a fulfilment threshold may be established (default is 1) with the format:

\[
\text{<condition> THOLD } \gamma
\]

indicating that the condition must be satisfied with minimum degree \(\gamma \in [0, 1]\)
Table 2. Fuzzy Comparators for FSQL (Fuzzy SQL).

<table>
<thead>
<tr>
<th>Comparator for:</th>
<th>Possibility</th>
<th>Necessity</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEQ</td>
<td>NFEQ</td>
<td>Fuzzy EQual (Possibly/Necessarily Equal)</td>
<td></td>
</tr>
<tr>
<td>FGT</td>
<td>NFGT</td>
<td>Fuzzy Greater Than</td>
<td></td>
</tr>
<tr>
<td>FGEQ</td>
<td>NFGEQ</td>
<td>Fuzzy Greater or Equal</td>
<td></td>
</tr>
<tr>
<td>FLT</td>
<td>NFLT</td>
<td>Fuzzy Less Than</td>
<td></td>
</tr>
<tr>
<td>FLEQ</td>
<td>NFLEQ</td>
<td>Fuzzy Less or Equal</td>
<td></td>
</tr>
<tr>
<td>MGT</td>
<td>NMGT</td>
<td>Much Greater Than</td>
<td></td>
</tr>
<tr>
<td>MLT</td>
<td>NMLT</td>
<td>Much Less Than</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Default computation for function CDEG with logic operators in FSQL.

<table>
<thead>
<tr>
<th>&lt;Condition&gt;</th>
<th>CDEG(&lt;Condition&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;cond1&gt; AND &lt;cond2&gt;</td>
<td>min(CDEG(&lt;cond1&gt;),CDEG(&lt;cond2&gt;))</td>
</tr>
<tr>
<td>&lt;cond1&gt; OR &lt;cond2&gt;</td>
<td>max(CDEG(&lt;cond1&gt;),CDEG(&lt;cond2&gt;))</td>
</tr>
<tr>
<td>NOT &lt;cond1&gt;</td>
<td>1 - CDEG(&lt;cond1&gt;)</td>
</tr>
</tbody>
</table>

Table 4. Fuzzy constants that may be used in FSQL queries.

<table>
<thead>
<tr>
<th>F. Constant</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNKNOWN</td>
<td>Unknown value but the attribute is applicable.</td>
</tr>
<tr>
<td>UNDEFINED</td>
<td>The attribute is not applicable or it is meaningless.</td>
</tr>
<tr>
<td>NULL</td>
<td>Total ignorance: We know nothing about it.</td>
</tr>
<tr>
<td>$[a,b,c,d]$</td>
<td>Fuzzy trapezoid (a \leq b \leq c \leq d).</td>
</tr>
<tr>
<td>$\textit{label}$</td>
<td>Linguistic Label: It may be a trapezoid or a scalar.</td>
</tr>
<tr>
<td>$[n,m]$</td>
<td>Interval “Between (n) and (m)” ((a=b=n) and (c=d=m)).</td>
</tr>
<tr>
<td>$#n$</td>
<td>Fuzzy value “Approximately (n)” ((b=c=n) and (n-a=d-n=\text{margin})).</td>
</tr>
</tbody>
</table>

to be considered. The reserved word THOLD is optional and may be substituted by a traditional crisp comparator (=, ≤, ...).

- **Function CDEG(<attribute>):** It shows a column with the fulfilment degree of the condition imposed on a specific attribute. If logic operators appear, the calculation of this compatibility degree is carried out as table 3 shows.

- **Fuzzy Constants:** In FSQL we can use the fuzzy constants detailed in table 4.

3. Fuzzy Time Representation

The main goal of this paper is to propose a unified way to represent imprecise temporal information by means of fuzzy values that can be used as the basis for the
implementation of fuzzy temporal capabilities in a conventional database system.

3.1. Imprecision Measure on Fuzzy Values

In order to find a unified way to represent fuzzy temporal information, it is possible that we need to transform the original data provided into a different format. When doing this transformation, it is very important that there is not a loss of information, that is, the amount of information (and therefore imprecision) should remain equal before and after the changes are made.

Therefore, the first step is to define an information function for fuzzy numbers that quantifies the amount of imprecision contained in a fuzzy number (in our case, a fuzzy date).

In 20 we propose an axiomatic definition of information, partially inspired by the theory of generalized information given by Kampé de Fériet 24 and that can be related to the precision indexes 10 and the specificity concept, introduced by Yager in 40.

Definition 1.-

Let \( \mathcal{D} \subseteq \sim \mathbb{R} \mid \mathbb{R} \subseteq \mathcal{D} \); we say that the mapping \( I \) defined as:

\[
I : \mathcal{D} \longrightarrow [0, 1]
\]

is an information measure on \( \mathcal{D} \) if it verifies:

1. \( I(A) = 1, \forall A \in \mathbb{R} \)
2. \( \forall A, B \in \mathcal{D} \mid h(A) = h(B) \) and \( A \subseteq B \Rightarrow I(B) \leq I(A) \).

The information about fuzzy numbers may depend on different factors, in particular, on imprecision and certainty. We focus on general types of information related only to these two factors.

To compute a measure of the imprecision contained in a fuzzy number, we will consider a measure of the imprecision of its \( \alpha \)-cuts, which are closed intervals on which the following function is defined:

\[
\forall A \in \sim \mathbb{R}, f_A(\alpha) = \begin{cases} b_\alpha - a_\alpha & \text{if } \alpha \leq h(A) \\ 0, \text{ otherwise} \end{cases}
\]  

(1)

From this imprecision function on the \( \alpha \)-cuts (see Figure 7), we define the total imprecision of a fuzzy value as a combination of the imprecision in every level \( \alpha \).

When \( \alpha = 0 \), we will consider that \( f_A(0) \) is the length of the support set.

Definition 2.-

The imprecision of a fuzzy number is defined as follows:

\[
f : \sim \mathbb{R} \longrightarrow \mathbb{R}_0^+
\]
∀ A ∈ \tilde{R}, f(A) = \int_{0}^{h(A)} f_A(\alpha) \, d\alpha \quad (2)

That is, the imprecision function f coincides with the area below the membership function of the fuzzy value. Since we are considering that fuzzy time values are always normalized, then h(A) = 1.

### 3.2. A summarized domain for temporal data

In the introduction we have seen that, in classical TDB, the valid time is managed thanks to the extension of the tables schemata by adding two new attributes, the valid start time -VST- and the valid end time -VET- to determine the period of validity of the fact expressed by a tuple.

In this paper we are going to consider that the information provided by the VST and VET for the classical TDB is fuzzy, in the sense that we are not completely sure about when the current values of the tuple become to be valid.

The more immediate solution to this problem is to soften the VST and the VET in such a way that they may contain fuzzy dates represented by means of a fuzzy number. This means that, if we use the parametrical representation for fuzzy numbers, we need to store four values for the VST and four values for the VET, as shown in Fig. 3.

Since the meaning of the attributes VST and VET is the period of time during which the values of a tuple are valid, it is more convenient to summarize the information given by the two fuzzy possibility distributions in a single fuzzy interval (from now on FVP or fuzzy validity period). The use of just one fuzzy value (which has now conjunctive semantics) for representing the period of validity reduces the complexity of storage and data manipulation in queries. As previously mentioned, it is quite easy to represent a fuzzy interval with these characteristics since only four parameters need to be stored in order to specify it. In a generalized model...
of fuzzy DB that supports this representation for fuzzy data and the corresponding implementation in a classical relational DB system (Oracle) is presented. Some notes about this model have been summarized in the previous section.

3.2.1. **Transformation that preserves imprecision**

Thus, a validity period can be represented by the trapezoidal fuzzy set shown in Figure 4 which incorporates the semantics of our problem.

As can be seen in such figure, the left and right sides of the interval are the parts that reflect the imprecision about the starting and ending time point of the validity time of the facts associated.
The problem now is that the imprecision provided by the two original fuzzy
dates must be translated to the interval that summarizes the considered period of
time. That is, all the imprecision of the starting date must be converted in the
imprecision of the left side of the interval and, in the same way, all the imprecision
of the ending date must be converted in the imprecision of the right side of the
interval.

If we consider that a way to measure the imprecision of a fuzzy set is to compute
its area, the problem we have in hands is a matter of geometrical computation.
The posed problem is shown in a graphical way in Figure 5.

The resulting fuzzy interval is obtained by means of the equality

\[ S_1 = S_2 \]

that obliges to maintain the same amount of imprecision after the transformation is
performed.

\[ (d_s + b_s) - (d_s - a_s) = d_1 - (d_1 - a) \]  

(3)

If we assume that the data associated to this time specification are precisely
known from \((d_s + b_s)\) to \((d_e - a_e)\), then \(d_1 = d_s + b_s\) and both terms become equal
and \(d_1 - a = d_s - a_s\), as shown in Figure 5. The same substitution should be made
to obtain the right part of the interval.

3.2.2. Transformation based on the convex hull

Another way to obtain the period of time between two fuzzy dates is to compute
the convex hull of the union of both fuzzy values, that is, the smallest convex
set including both dates; in this case, the imprecision before the transformation is
larger than the imprecision after it, since only the left spread of the initial date and
the right spread of the ending date are considered, as shown in Figure 6.

This interpretation, in spite of not preserving the original amount of imprecision,
is more natural and close to human beings intuition, since it joins the "most possible"
starting point of the period with the "most possible" ending point of it to construct
the validity period.

4. Operations with Fuzzy Time Intervals

Once the fuzzy interval is adopted as the main data type to store valid time, we
can define a suitable set of operators on it.

4.1. Operators on Fuzzy Intervals

Let \(P_1\) and \(P_2\) be two fuzzy periods of time. We can define on them the following
operators.

- **CONTAINS**\((P_1,P_2)\): This operator extends the classical one for intervals but,
in this case, it computes to what degree \(P_2\) is included in \(P_1\). This situation can
be modeled by means of the implication:
If we assume that the implication function $I(P_2(d), P_1(d))$ used is the material implication, then the fulfillment degree of this fuzzy inclusion will be:

$$\text{CONTAINS}(P_1, P_2) = \min_{x \in D} I(\mu_{P_2}(x), \mu_{P_1}(x))$$

$$\text{CONTAINS}(P_1, P_2) = \min_{x \in D} \{(1 - \mu_{P_2}(x)) \oplus \mu_{P_1}(x)\}$$  \hspace{1cm} (4)

If the t-conorm considered is the maximum, the resulting measure is a necessity. Note that this measure includes the classical sets inclusion as a particular case.

- BEFORE($P_1, P_2$): This operator computes the matching degree of $P_1$ with the complement of $P_2$ on the left side (the period before $P_2$) as shown in Figure 7. The possibility-based expression to obtain this result is:

$$\text{BEFORE}(P_1, P_2) = \begin{cases} 
1 & \text{if } \beta_{P_1} \leq \alpha_{P_2} \\
\frac{\alpha_{P_1} - \beta_{P_2}}{(\alpha_{P_2} - \beta_{P_2}) - (\beta_{P_1} - \alpha_{P_1})} & \text{if } \beta_{P_1} > \alpha_{P_2} \text{ and } \alpha_{P_1} < \beta_{P_2} \\
0 & \text{otherwise } (\alpha_{P_1} \geq \beta_{P_2})
\end{cases}$$  \hspace{1cm} (5)

- AFTER($P_1, P_2$): The computation of the AFTER operator is analogous to the BEFORE one, but building the period after $P_2$ and comparing it with $P_1$. The possibility-based expression to obtain this result is:
Fig. 7. Computation of BEFORE(P1,P2).

\[
AFTER(P_1, P_2) = \begin{cases} 
1 & \text{si } \gamma_{P_1} \geq \delta_{P_2} \\
\frac{\delta_{P_1} - \gamma_{P_2}}{(\delta_{P_2} - \gamma_{P_2}) - (\gamma_{P_1} - \delta_{P_1})} & \text{si } \gamma_{P_1} < \delta_{P_2} \text{ and } \delta_{P_1} > \gamma_{P_2} \\
0 & \text{en otro caso } (\delta_{P_1} \leq \gamma_{P_2})
\end{cases}
\]  

(6)

Note that both operators (AFTER and BEFORE) can also be used to compare a period with a fuzzy date or two fuzzy dates since the operation can also be applied on disjunctive values. This capability is very useful for many queries as, for example, Find the employees that earned more than 1000 euros before the beginning of January.

- OVERLAPS(P1, P2): The overlaps operator on non-fuzzy periods returns the boolean value true when there is at least a point (date) that belongs to both periods and false otherwise. In a fuzzy context, the returned value will belong to [0,1] and will be computed as:

\[
OVERLAPS(P_1, P_2) = \text{Sup}_x \{\text{Min}\{\mu_{P_1}(x), \mu_{P_2}(x)\}\} 
\]  

(7)

which includes the classical operator on intervals as a particular case.

- EQUALS(P1, P2): The problem now is to compute to what degree two events are simultaneous. This operation is very useful since joins are based on it. The simultaneity degree or temporal equality between two periods of time can be carried out computing the degree to which both fuzzy/crisp sets -P1 and P2- are mutually included one in the other, that is, we should compute:

\[
EQUALS(P_1, P_2) = \ominus\{\text{CONTAINS}(P_1/P_2), \text{CONTAINS}(P_2/P_1)\} 
\]  

(8)

as the final value for the fuzzy equality degree between the two fuzzy sets where \( \ominus \) stands for a T-norm (minimum in our case).

Of course, other definitions for EQUAL operator are possible and the best one depends on the context (for some applications even the strict equality of fuzzy sets might be useful).
• START(P): This operator computes the starting fuzzy date of the interval. Depending on the way the interval is built (preserving imprecision or by the convex hull) the result of this operator will be different. Let us suppose that \( P = (\alpha, \beta, \gamma, \delta) \) where \([\alpha, \beta]\) is the modal interval and \(\gamma\) and \(\delta\) are the left and right spreads, respectively.

1. If the FVP was built preserving the original imprecision, (and under the symmetry assumption for fuzzy dates) the resulting parameters for the starting date are:

\[
\text{START}(P) = (\alpha - (\gamma/2), \alpha - (\gamma/2), (\gamma/2), (\gamma/2)) \tag{9}
\]

Note that without the symmetry hypothesis infinite results could be found. Anyway, if no information is provided, this result is the most coherent from the semantical point of view because it corresponds to the scaled up (normalized) intersection of \( P \) and its complement on the left side (before \( P \)) like shown in Figure 8.

![Fig. 8. Computation of the STARTS operator when imprecision is preserved.](image)

(2) If the FVP is the convex hull of the original two fuzzy dates, the resulting parameters for the starting date are:

\[
\text{START}(P) = (\alpha, \alpha, \gamma, \gamma) \tag{10}
\]

as shown in Figure 9.

• END(FVP): The computation of the ending date of a fuzzy interval is analogous to the START case.

Note that, whatever the transformation is, both operators return the crisp start (conversely the crisp end) when the period considered is not fuzzy, since the spreads are 0.
Finally, there are many interesting queries that involve the computation of the cardinality of a FVP as, in our example, the number of days an employee has been working. In this case, we will need to take into account the scalar cardinality of a fuzzy set $A$ defined as:

$$|A| = \sum_{x \in X} (\mu_A(x))$$

Of course, this sum makes sense for a concrete granularity (day, in our paper). Let us call $\text{LENGTH}(P)$ the function that computes the FVP cardinality (that is, FVP lasting).

5. Updating the Valid Time Relation

As we explained in the introduction, information is never deleted in a TDB when an update operation is carried out. The process is to leave the old version of the data in the DB and to add the new version with some suitable modifications made, closing the old one and setting as the valid end time the immediately previous granule to the valid start time of the inserted tuple. This idea has to be generalized to the fuzzy case.

From the semantical point of view it makes sense that, as the new period begins, the old period ends. That is, as the membership to the new period increases, the membership to the old one must decrease. This concept can be formalized as follows.

**Definition 2.**

Let us use $\mu_O(x)$ to denote the membership function associated to the fuzzy interval of the old version of the tuple to be updated and $\mu_N(x)$ the membership function associated to the new fuzzy interval. Then, the membership function of the fuzzy interval ($\mu'_O(x)$) that serves to close the validity time of the old one is:
\[ \mu'_O(x) = \begin{cases} \mu_O(x) & \forall x \mid \mu_N(x) = 0 \\ 1 - \mu_N(x) & \forall x \mid \mu_N(x) > 0 \end{cases} \]  

(11)

Fig. 10. Membership function of the fuzzy value O' that closes a version of a tuple with new valid time N

This result can be seen graphically in Figure 10.

Let us suppose that the parametrical definition of the old and new periods are respectively: \( FVP_O = (\alpha_O, \beta_O, \gamma_O, 0) \) and \( FVP_N = (\alpha_N, \beta_N, \gamma_N, 0) \). In this situation, the parametrical definition of \( FVP'_O \) is:

\[ FVP'_O = \begin{cases} (\alpha_O, \alpha_N - \gamma_N, \gamma_O, \gamma_N) & \text{when } \gamma_N > 0 \\ (\alpha_O, \alpha_N - 1, \gamma_O, 0) & \text{otherwise} \end{cases} \]  

(12)

Notice that, thanks to the fact that the underlying domain is discrete, we can easily implement the update operation with periods for both the fuzzy and crisp cases.

We will use closing interval to denote all the values \( x \in D \) such that \( 0 < \mu_N(x) < 1 \) and \( 0 < \mu'_O(x) < 1 \).

It is obvious that the non-overlapping condition required for the crisp TDB is generalized in the sense that the function used to close the old version guarantees that for any time point \( t \) in closing interval, \( \mu'_O(t) + \mu_N(t) = 1 \).

Finally, this way of performing the update operation involves the following consequences:

- In the case that the period comes from the convex hull, \( \text{END}(FVP'_O) \) equals \( \text{START}(FVP_N) \) at 0.5 degree, except for the crisp case, where the degree is 0.
- In the case that the period preserves imprecision, \( \text{END}(FVP'_O) \) equals \( \text{START}(FVP_N) \) at degree 1. The crisp case remains the same.
This situation, which might be considered a drawback, is assumed for the sake of interpretability of the ending and starting dates of the periods.

6. Querying the Fuzzy TDB

Once we are able to represent fuzzy periods of valid time, queries about a crisp/fuzzy date or period to the valid time relations can be solved computing the corresponding fulfillment degree between the time we are querying about (QT) and the database valid time (FVP). In this section we show some representative types of queries.

In order to illustrate all the cases, let us consider the following fuzzy TDB instance.

<table>
<thead>
<tr>
<th>EMPNAM</th>
<th>EMPID</th>
<th>SALARY</th>
<th>BOSS</th>
<th>EXPERTISE</th>
<th>FVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRANT</td>
<td>1245</td>
<td>1500</td>
<td>9877</td>
<td>TRAINEE</td>
<td>(15/06/1997, 31/05/1998, 2,2)</td>
</tr>
<tr>
<td>GRANT</td>
<td>1245</td>
<td>1800</td>
<td>9877</td>
<td>JUNIOR</td>
<td>(02/06/1994, 31/12/2050,2,0)</td>
</tr>
<tr>
<td>REDFORD</td>
<td>9877</td>
<td>1200</td>
<td>4588</td>
<td>TRAINEE</td>
<td>(20/08/1994, 31/01/1996, 2,3)</td>
</tr>
<tr>
<td>REDFORD</td>
<td>9877</td>
<td>1300</td>
<td>4588</td>
<td>JUNIOR</td>
<td>(04/02/1996, 31/01/1997, 4,1)</td>
</tr>
<tr>
<td>REDFORD</td>
<td>9877</td>
<td>2200</td>
<td>9989</td>
<td>SENIOR</td>
<td>(04/04/1997, 31/12/2050,4,0)</td>
</tr>
<tr>
<td>BROWN</td>
<td>1245</td>
<td>2500</td>
<td>4588</td>
<td>JUNIOR</td>
<td>(01/05/1994, 31/12/2050,0,0)</td>
</tr>
<tr>
<td>STREEP</td>
<td>6579</td>
<td>4000</td>
<td>9877</td>
<td>TRAINEE</td>
<td>(15/06/1997, 31/12/2050,0,0)</td>
</tr>
<tr>
<td>NEWMAN</td>
<td>5546</td>
<td>2300</td>
<td>9877</td>
<td>SENIOR</td>
<td>(18/06/1997, 29/04/1998,8,10)</td>
</tr>
</tbody>
</table>

Table 5.

Example 1. Find the boss of employee number 9877 during the beginning of April 1997 (01/04/97,04/04/97,0,2). The formal expression of this query using FSQL syntax is:

```
fsql> SELECT e.empnam, e.expertise, e.boss, CDEG(e.fvp) FROM employees WHERE e.empid=9877 AND CONTAINS(fvp,$['01/04/1997','05/04/1997',0,2])>0 TTHOLD 0.0;
```

<table>
<thead>
<tr>
<th>EMPNAM</th>
<th>EXPERTISE</th>
<th>BOSS</th>
<th>CDEG(FVP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>REDFORD</td>
<td>SENIOR</td>
<td>9989</td>
<td>0.25</td>
</tr>
<tr>
<td>REDFORD</td>
<td>JUNIOR</td>
<td>4588</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Example 2. Find employees with boss 9877 during the same period of time. The formal expression of this query using FSQL syntax is:

```
fsql> SELECT e.empnam,f.empnam,CDEG(*) FROM employees e, employees f WHERE e.boss=9877 AND f.boss=9877 AND e.empid<>f.empid AND EQUALS(f.fvp,e.fvp)>0;
```

E.EMPNAM F.EMPNAM CDEG(*)
Example 3. Find employees who have changed their status after more than 6 years in the company. The formal expression of this query using FSQL syntax is:

```
fsql> SELECT e1.empid
FROM employees e1, employees e2
WHERE FGEQ(END(e2.fvp),TODAY)>0
AND e1.empid=e2.empid
AND e1.EXPERTISE<>e2.EXPERTISE
GROUP BY e1.empid
HAVING sum(LENGTH(fvp))>=365*6;
```

E.EMPNAM
-------
REDFORD

Example 4. Find employees whose expertise was JUNIOR before the first of May, 1997. The formal expression of this query using FSQL syntax is:

```
SELECT e.empnam
FROM employees e
WHERE BEFORE(fvp, ['01/05/1997', '01/05/1997', 0, 0])>0
AND e.EXPERTISE='JUNIOR';
```

E.EMPNAM
--------
BROWN

7. Conclusions and Future Work

In this paper we have shown the advantages of representing fuzzy temporal data with a parametrical representation and two ways to do it. On this temporal data, we have proposed a variety of operators and explained how an update operation can be carried out by taking into account that no deletion is possible when temporal information is stored. As a result of this operation, a modification to the old version of the tuple is needed by changing some of the parameters that define it. As a consequence of our approach, the queries return a set of tuples together with a fulfillment degree when a query is made on these fuzzy temporal data. The paper also analyzes the different types of queries that can be made on these data. We are currently analyzing the behavior of other operators and considering a wider range
of temporal data. We are also studying the problem of primary keys in the presence of fuzzy intervals instead of VST and VET attributes and trying to find out new indexing techniques that take the new primary keys into account.

8. Acknowledgements

This work has been partially supported by Research Projects Representación y Manipulación de Objetos Imperfectos en Problemas de Integración de Datos: Una Apli-
cación a los Almacenes de Objetos de Aprendizaje, Junta de Andalucía (TIC03175) and Tratamiento de Información Difusa Temporal en un SGBD Relacional: Formu-
lación Teórica, Implementación y Aplicaciones, Estado Español (TIN2008-02066).

References