

FTSQL2: Fuzzy Time in Relational Databases*

José Galindo¹

Juan M. Medina²

¹Dpto. Lenguajes y Ciencias de la Computación, Universidad de Málaga, Spain (ppgg@lcc.uma.es).

²Dpto. Ciencias de la Computación e I.A., Universidad de Granada, Spain.

Abstract

FSQL language is an extension of the *SQL* language which permits us to write flexible (or fuzzy) conditions in our queries to a fuzzy or traditional database. In this work we present some new fuzzy data types based on time concepts. We extend the existing temporal data types in *SQL2* and we give a brief overview of basic definitions in fuzzy temporal databases. In particular, we explain how query operations in *TSQL2* language need to be extended for fuzzy temporal querying.

Keywords: Fuzzy Time, Temporal Databases, Information Retrieval, Flexible Queries, Fuzzy *SQL*, Fuzzy Query Languages, Fuzzy Relational Databases.

1 Introduction

Many times databases require some aspect of time when organizing their information. Temporal databases, in the broadest sense, encompass all databases of this kind [2, 7]. On the other hand, fuzzy databases have been developed in the last years, rising up different models [5, 9], among which they highlight the Umano-Fukami model, and the GEFRED model by Medina-Pons-Vila. This last model represents an eclectic synthesis of the different models for fuzzy RDB (Relational Databases).

In this article we present some new fuzzy data types based on time concepts. We extend the existing data types in *SQL2*. Time in databases is very useful, because it allows to store the date/time in which the fact was considered to be true in the real world. But sometimes, this time is not exactly known or it is a vague time period. It is studied here and we give a brief overview of basic definitions in fuzzy temporal RDB.

Besides, we explain how query operations in *TSQL2* language [11] need to be extended for fuzzy temporal querying. *TSQL2* language extends *SQL* for querying valid time, transaction time and bitemporal RDB.

All these definitions are made based using the Fuzzy *SQL* (*FSQL*) language concepts [3, 4, 5]. In [6], authors extend *FSQL* to express fuzzy division in fuzzy RDB.

First, we include a brief explanation of the main advantages of the *FSQL* *SELECT* sentence, in order to express fuzzy queries (see [4, 5] for details about this and other *FSQL* sentences). Next, we will expose briefly the information that is stored in the database (fuzzy attributes types...). Finally, we will study how fuzzy time may be included and treated in databases.

*Partially supported by CICYT project TIC99-0558.

2 Fuzzy Database with *FSQL*

The *FSQL* language [3, 4, 5] extends the *SQL* language to allow flexible queries, taking into account the characteristics of imprecise information. We have extended the *SELECT* command to express flexible queries and, due to its complex format, we only show here an abstract with the main extensions added to this command:

- **Linguistic Labels:** If an attribute is capable of fuzzy treatment then linguistic labels can be defined on it. Every label has an associated trapezoidal possibility distribution (four values like figure in Table 1) or there is a similarity relationship defined between them.

- **Fuzzy Constants:** In *FSQL* we can use all the fuzzy constants which appear in Table 1.

- **Fuzzy Comparators:** In addition to the common comparators (=, >, etc), *FSQL* includes the fuzzy comparators of Table 2, where U is the underlying domain or Universe, $A = [\alpha_A, \beta_A, \gamma_A, \delta_A]$, $B = [\alpha_B, \beta_B, \gamma_B, \delta_B]$ and *CDEG* function indicates the compatibility degree in the fuzzy comparison. Of course, the specified definition may be changed. In the same way as in *SQL*, fuzzy comparators can compare one column value with one constant or two column values of the same type. Necessity comparators are more restrictive than possibility comparators are, i.e. necessity degree is always lower than possibility degree.

- **Fulfillment Thresholds (γ):** For each simple condition a fulfillment threshold may be established (default is 1) with the format: <condition> THOLD γ , indicating that the condition must be satisfied with a minimum degree $\gamma \in [0, 1]$. The reserved word THOLD is optional and it can be substituted by a traditional crisp comparator (=, \leq , etc), modifying the meaning of the query. The word THOLD is equivalent to \geq .

- ***CDEG*(<attribute>) function:** This function, used in the select list, shows a column with the fulfillment degree of the query condition for the specified attribute. *CDEG*(*) computes the fulfillment degree of each tuple in the condition (considering all its attributes, not just one of them). If logic operators appear, the calculation of the compatibility degree is carried out, by default, using the minimum T-norm and the maximum T-conorm.

We have a painstaking ***FSQL* Server** [4, 5] to obtain the answers to *FSQL* queries. It is programmed in PL/*SQL* language for Oracle[©] DBMS. The *FSQL* Server has been programmed to work with crisp or fuzzy databases (based on the GEFRED model).

Example 1: *SELECT * FROM Persons WHERE Hair*

Table 1: Fuzzy values that may be used in the fuzzy database or in FSQL sentences.

F. Constant	Significance
UNKNOWN	Unknown value but the attribute is applicable.
UNDEFINED	The attribute is not applicable or it is meaningless.
NULL	Total ignorance: We know nothing about it.
$\$[a, b, c, d]$	Fuzzy trapezium: $a \leq b \leq c \leq d$ (See figure on the right).
$\$label$	Linguistic label (a trapezium or a scalar defined in FMB).
$[n, m]$	Interval "Between n and m" ($a=b=n$ and $c=d=m$).
$\#n$	"Approximately n" (Triangle: $b=c=n$ and $n-a=d-n=margin$).

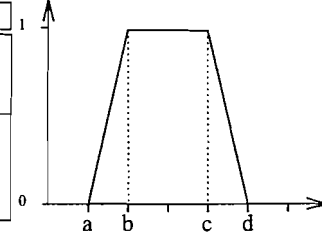


Table 2: Fuzzy comparators definitions for fuzzy attributes Type 1 and 2, and their (temporal) meaning.

Meaning	Possibility Comparators	Necessity Comparators
Fuzzy Equal	$CDEG(A \text{ FEQ } B) = \sup_{u \in U} \{ \min(A(u), B(u)) \}$	$CDEG(A \text{ NFEQ } B) = \inf_{u \in U} \max(1 - A(u), B(u))$
Fuzzy Greater (Fuzzy After)	$CDEG(A \text{ FGT } B) = \begin{cases} 1 & \text{if } \gamma_A \geq \delta_B \\ \frac{\delta_A - \gamma_B}{(\delta_B - \gamma_B) - (\gamma_A - \delta_A)} & \text{if } \gamma_A < \delta_B \text{ y } \delta_A > \gamma_B \\ 0 & \text{in other case } (\delta_A \leq \gamma_B) \end{cases}$	$CDEG(A \text{ NFGT } B) = \begin{cases} 1 & \text{if } \alpha_A \geq \delta_B \\ \frac{\beta_A - \gamma_B}{(\delta_B - \gamma_B) - (\alpha_A - \beta_A)} & \text{if } \alpha_A < \delta_B \text{ y } \beta_A > \gamma_B \\ 0 & \text{in other case } (\beta_A \leq \gamma_B) \end{cases}$
Fuzzy Greater or Equal (Fuzzy After or at a Time)	$CDEG(A \text{ FGEQ } B) = \begin{cases} 1 & \text{if } \gamma_A \geq \beta_B \\ \frac{\delta_A - \alpha_B}{(\beta_B - \alpha_B) - (\gamma_A - \delta_A)} & \text{if } \gamma_A < \beta_B \text{ y } \delta_A > \alpha_B \\ 0 & \text{in other case } (\delta_A \leq \alpha_B) \end{cases}$	$CDEG(A \text{ NFGEQ } B) = \begin{cases} 1 & \text{if } \alpha_A \geq \beta_B \\ \frac{\beta_A - \alpha_B}{(\beta_B - \alpha_B) - (\alpha_A - \beta_A)} & \text{if } \alpha_A < \beta_B \text{ y } \beta_A > \alpha_B \\ 0 & \text{in other case } (\beta_A \leq \alpha_B) \end{cases}$
Fuzzy Less (Fuzzy Before)	$CDEG(A \text{ FLT } B) = \begin{cases} 1 & \text{if } \beta_A \leq \alpha_B \\ \frac{\alpha_A - \beta_B}{(\alpha_B - \beta_B) - (\beta_A - \alpha_A)} & \text{if } \beta_A > \alpha_B \text{ y } \alpha_A < \beta_B \\ 0 & \text{in other case } (\alpha_A \geq \beta_B) \end{cases}$	$CDEG(A \text{ NFLT } B) = \begin{cases} 1 & \text{if } \delta_A \leq \alpha_B \\ \frac{\gamma_A - \beta_B}{(\alpha_B - \beta_B) - (\delta_A - \gamma_A)} & \text{if } \delta_A > \alpha_B \text{ y } \gamma_A < \beta_B \\ 0 & \text{in other case } (\gamma_A \geq \beta_B) \end{cases}$
Fuzzy Less or Equal (Fuzzy Before or at a Time)	$CDEG(A \text{ FLEQ } B) = \begin{cases} 1 & \text{if } \beta_A \leq \gamma_B \\ \frac{\delta_B - \alpha_A}{(\beta_A - \alpha_A) - (\gamma_B - \delta_B)} & \text{if } \beta_A > \gamma_B \text{ y } \alpha_A < \delta_B \\ 0 & \text{in other case } (\alpha_A \geq \delta_B) \end{cases}$	$CDEG(A \text{ NFLEQ } B) = \begin{cases} 1 & \text{if } \delta_A \leq \gamma_B \\ \frac{\gamma_A - \delta_B}{(\gamma_B - \delta_B) - (\delta_A - \gamma_A)} & \text{if } \delta_A > \gamma_B \text{ y } \gamma_A < \delta_B \\ 0 & \text{in other case } (\gamma_A \geq \delta_B) \end{cases}$
Much Greater Than (Much After)	$CDEG(A \text{ MGT } B) = \begin{cases} 1 & \text{if } \gamma_A \geq \delta_B + \mathcal{M} \\ \frac{\gamma_B + \mathcal{M} - \delta_A}{(\gamma_A - \delta_A) - (\delta_B - \gamma_B)} & \text{if } \gamma_A < \delta_B + \mathcal{M} \text{ y } \\ & \delta_A > \gamma_B + \mathcal{M} \\ 0 & \text{o. c. } (\delta_A \leq \gamma_B + \mathcal{M}) \end{cases}$	$CDEG(A \text{ NMGT } B) = \begin{cases} 1 & \text{if } \alpha_A \geq \delta_B + \mathcal{M} \\ \frac{\gamma_B + \mathcal{M} - \beta_A}{(\alpha_A - \beta_A) - (\delta_B - \gamma_B)} & \text{if } \alpha_A < \delta_B + \mathcal{M} \text{ y } \\ & \beta_A > \gamma_B + \mathcal{M} \\ 0 & \text{o. c. } (\beta_A \leq \gamma_B + \mathcal{M}) \end{cases}$
Much Less Than (Much Before)	$CDEG(A \text{ MLT } B) = \begin{cases} 1 & \text{if } \beta_A \leq \alpha_B - \mathcal{M} \\ \frac{\beta_B - \mathcal{M} - \alpha_A}{(\beta_A - \alpha_A) - (\alpha_B - \beta_B)} & \text{if } \beta_A > \alpha_B - \mathcal{M} \text{ y } \\ & \alpha_A < \beta_B - \mathcal{M} \\ 0 & \text{o. c. } (\alpha_A \geq \beta_B - \mathcal{M}) \end{cases}$	$CDEG(A \text{ NMLT } B) = \begin{cases} 1 & \text{if } \delta_A \leq \alpha_B - \mathcal{M} \\ \frac{\beta_B - \mathcal{M} - \gamma_A}{(\delta_A - \gamma_A) - (\alpha_B - \beta_B)} & \text{if } \delta_A > \alpha_B - \mathcal{M} \text{ y } \\ & \gamma_A < \beta_B - \mathcal{M} \\ 0 & \text{o. c. } (\gamma_A \geq \beta_B - \mathcal{M}) \end{cases}$

FEQ \$Fair 0.6 AND Age FGT \$Young 0.8; □

Two kind of information are stored in the database:

a) **Traditional Database:** It consists of all the data stored in the relations but with a special format in order to represent fuzzy attribute values. *Fuzzy attributes* are classified by the system in 3 types:

- **Type 1:** These attributes are totally crisp (traditional), but they have some linguistic trapezoidal labels defined on them. With these fuzzy attributes, we can use all constants (Table 1) in the query conditions.
- **Type 2:** These attributes admit crisp data as well as possibility distributions over an ordered underlying domain or Universe U . With these attributes we can store and use all the constants shown in Table 1.
- **Type 3:** These attributes are defined on a not ordered underlying domain, for instance the hair colour. On these attributes some labels are defined and on these labels a similarity relation has yet to be defined. With these attributes we can only use the fuzzy comparator FEQ, as they have no relation of order.

b) **Fuzzy Meta-knowledge Base (FMB):** It stores data about the FRDB: attributes which are capable to fuzzy treatment, linguistic labels, margin and \mathcal{M} (see Tables 1 and 2), the similarity relation (Type 3)...

3 Fuzzy Time

In [1], a framework for modeling temporal knowledge pervaded with imprecision or uncertainty is introduced in terms of possibility distributions: Ill-known dates, time intervals with fuzzy boundaries, fuzzy durations and uncertain precedence relations between events. In [10], fuzzy time is used for problem solving in industrial dynamic systems. In [12], time dependent fuzzy set $A(t)$ is defined as a fuzzy set whose membership function may change with time. Besides, an inference processing is defined in order to obtain the fuzzy set $A(t)$ when t is a time fuzzy set (based on [1]). Authors apply those definitions in time dependent fuzzy rules.

Many authors [1, 8] have pointed out that problems

of processing space or time information in reasoning are similar. Thus, time may be considered like a fuzzy attribute Type 2, where its underlying domain (X axis in figure in Table 1) is the absolute time. This absolute time is an ordered sequence of points in the time. Thus, it is necessary to set a *granularity* in this absolute time, i.e., time is considered to be an ordered sequence of points with a minimum and fixed distance between every two consecutive points. Of course, granularity is determined by the application. This special underlying domain makes necessary to define new fuzzy types.

There are two types of temporal information according to the event duration:

1. Point events: These are events or facts without duration. They are typically associate with a single time point in some granularity. In SQL2, temporal data types including this information are: DATE (specifying year, month and day), TIME (specifying hour, minute and second), and TIMESTAMP (specifying a DATE/TIME combination). Fuzzy point events allow to use fuzzy values in this domain. We define new fuzzy time types named F_DATE, F_TIME and F_TIMESTAMP (including its corresponding crisp values). Fuzzy point events are represented using approximately values in Table 1 (triangular functions) with a predetermined margin (according to the context). For example, in healthcare application it is interesting to store that a patient recovered approximately on January 6, 1994. Besides, using fuzzy comparators (Table 2) we can realize fuzzy queries. For example: "Retrieve all patients who recovered approximately on December 31, 2000".

2. Duration events: These events are associate with a specific time period in the database. In SQL2, temporal data types including this information are: INTERVAL (a relative time duration, such as 2 days, 5 hours or 30 seconds), and PERIOD (an anchored time duration with a fixed starting point, such as from August 3, 1970 to October 29, 1970). A new fuzzy INTERVAL type is not necessary, because the underlying domain is the real numbers (just like fuzzy attributes Type 2). However, measurement unit (years, hours, seconds...) must be stored with every value or with every attribute, according to the application. Of course, measurement unit must be used in all computations.

A new fuzzy PERIOD type is not necessary either, because there are two alternative representation ways: The first one is using fuzzy point events, i.e. the fuzzy types defined above but with others values, specially the trapezoidal shape (Table 1). For example, value "on May but specially near or between days 11 and 27" is represented by $a=(\text{May } 1, 2001)$, $b=(\text{May } 11, 2001)$, $c=(\text{May } 27, 2001)$ and $d=(\text{May } 31, 2001)$. The second way is using two fuzzy point events, representing its start and end time points. For example, we can store that the most grave stage of a patient disease was from approximately April 14, 2001 until approximately July 25, 2001.

4 Fuzzy Temporal RDB

In this section we study how to incorporate fuzzy time in RDB (tuple versioning). The extension for incorporating fuzzy time in Object-Oriented Databases (attribute versioning) is easy [2].

Basically, time may be interpreted to mean two different things (time dimensions). **Valid time** is the most natural interpretation and it is that the associated time is the time that the event occurred, or the period during which the fact was considered to be true in the real world. **Transaction time** mean that the associated time refers to the time when the information was actually stored in the database, i.e. the value of the system time clock when the information was changed in the system. Some applications need only one of the dimensions (valid time databases or transaction time databases), others need both time dimensions (bitemporal databases) and others need an user-defined interpretation (user-defined time databases).

We study only valid time databases, because transaction time databases need the exact system time. However, valid time databases need the time in which the fact was considered to be true in the real world. Sometimes, this time is not exactly known or it is a vague time period, as we saw in previous examples.

Just like usual temporal RDB, valid time relations have two additional attributes whose data type is one of the previously defined fuzzy time types: VST (Valid Start Time) and VET (Valid End Time). These attributes in tuple t represent that its information is valid in the real world only during the time period $[t.VST, t.VET]$. A special value "now" is included in the domain of VET attribute. Value "now" in a tuple represents that this tuple is valid from VST till now, i.e., this tuple represents the current values of the entity represented in the relation.

Example 2: Let us suppose a healthcare database storing the most grave stage of some diseases. Then, some attributes of this relation would be patient name, disease, treatment... and attributes VST and VET. So, two valid tuples are (Dominique, asthma, ... #(June 4), \$[July 6, July 6, July 6, July 15]) and (Dominique, asthma, ... #(September 1), now).

First tuple represents that Dominique suffered an asthma attack approximately on June 4. The same symptoms continued until July 6. Then, patient began slowly to get better until July 15. Note that VET attribute is a trapezoidal value with four values (like figure in Table 1). The second tuple represents that Dominique suffered a new asthma attack approximately on September 1 and it is the current state. \square

Note that several tuples for the same entity may exist (like Dominique in the previous example). In temporal databases this is solved including VST (or VET) in the relation key. With fuzzy VST attributes, we must suppose that two tuples with different values in VST are different, even though they are very similar.

Fuzzy valid time relations keep track of the history of

changes as they become effective in the real world, even if these changes occurred gradually or in an ill-known time. However, because updates, insertions, and deletions may be applied proactively (before current time) or retroactively (after current time), there is no record of the actual database state at any point in time. If it is needed, then we must use bitemporal relations. They include TST (Transaction Start Time) and TET (Transaction End Time) attributes, whose data type is typically crisp `TIMESTAMP`. In this case we must include TST attribute in the relation key, instead of VST.

5 Extending TSQL2 Language

So far, we have discussed how data models may be extended with fuzzy time using temporal constructs. We now give a brief overview of how query operations need to be extended for fuzzy temporal querying. We briefly discuss an extension of TSQL2 language [2, 11].

Typical conditions for valid time databases are, for example, to select all tuple versions that were valid on a certain time point T or that were valid during a certain time period $[T1, T2]$. The specified time point or time period may be fuzzy or crisp values and they are compared with the valid time period of each tuple version t : $[t.VST, t.VET]$. In Table 3 we extend the more common operations with the prefix `F_` and we define each one using existing fuzzy possibility comparators (Table 2). The last four comparators are totally new and the last two need a value \mathcal{M} (according to the context) in order to consider two time values as very separated. The extension using necessity comparators (with prefix `FN_`) is trivial.

Table 3: Extending TSQL2 comparators (possibility versions) and four new ones (X means eXclusively).

Expression	Equivalence
$[t.VST, t.VET]$ <code>F_INCLUDES</code> $[T1, T2]$	<code>T1 FGEQ t.VST AND T2 FLEQ t.VET</code>
$[t.VST, t.VET]$ <code>F_INCLUDED_IN</code> $[T1, T2]$	<code>T1 FLEQ t.VST AND T2 FGEQ t.VET</code>
$[t.VST, t.VET]$ <code>F_OVERLAPS</code> $[T1, T2]$	<code>T1 FLEQ t.VET AND T2 FGEQ t.VST</code>
$[t.VST, t.VET]$ <code>F_BEFORE</code> $[T1, T2]$	<code>T1 FGEQ t.VET</code>
$[t.VST, t.VET]$ <code>F_AFTER</code> $[T1, T2]$	<code>T2 FLEQ t.VST</code>
$[t.VST, t.VET]$ <code>F_XBEFORE</code> $[T1, T2]$	<code>T1 FGT t.VET</code>
$[t.VST, t.VET]$ <code>F_XAFTER</code> $[T1, T2]$	<code>T2 FLT t.VST</code>
$[t.VST, t.VET]$ <code>F_MUCH_BEFORE</code> $[T1, T2]$	<code>T1 MGT t.VET</code>
$[t.VST, t.VET]$ <code>F_MUCH_AFTER</code> $[T1, T2]$	<code>T2 MLT t.VST</code>

This new fuzzy temporal comparisons may be completed with the threshold clause, in order to retrieve only values with a certain minimum threshold. The logic operator `AND` may be translated by the minimum T-norm. For example, to select all tuple versions that were valid (at any point) during around 1994 (in minimum degree 0.6), the condition is: $[t.VST, t.VET]$ `F_OVERLAPS` $\#(1994, 1, 1), \#(1994, 12, 31)]$ `THOLD` 0.6 (`OVERLAPS` has also been called `INTERSECTS_WITH`).

6 Conclusions and Future Lines

We have presented a definition for fuzzy time, extending the SQL2 data types, and we have extended temporal comparators in TSQL2 language in order to use fuzzy time values. These extensions are useful because many times the database user do not know the exactly time when an event occurs or this exactly time does not exist, because the event occurs in a fuzzy time period (gradual events). All of this may be used for fuzzy queries in existing temporal databases (crisp or fuzzy).

In these definitions we extend the TSQL2 language using the FSQL comparators, in what may be called FTSQL2 language, for fuzzy temporal databases. Next, we plan to study how to store in fuzzy databases values on the precedence relations between two fuzzy dates in the database. Besides, some others functions may be useful, such as length of a fuzzy time value.

References

- [1] D. Dubois, H. Prade, "Processing Fuzzy Temporal Knowledge". IEEE Trans. Systems Man Cybernet, 19, pp. 729-744, 1989.
- [2] R. Elmasri, S.B. Navathe, "Fundamentals of Database Systems", Third Edition. Addison-Wesley, 2000.
- [3] J. Galindo, J.M. Medina, A. Vila, J.C. Cubero, "Fuzzy Comparators for Flexible Queries to Databases". Iberoamerican Conference on Artificial Intelligence, IBERAMIA'98, Lisbon (Portugal), 1998.
- [4] J. Galindo, J.M. Medina, O. Pons, J.C. Cubero, "A Server for Fuzzy SQL Queries", in "Flexible Query Answering Systems", eds. T. Andreasen, H. Christiansen and H.L. Larsen, Lecture Notes in Artificial Intelligence (LNAI) 1495, pp. 164-174. Ed. Springer, 1998.
- [5] J. Galindo, "Tratamiento de la Imprecisión en Bases de Datos Relacionales: Extensión del Modelo y Adaptación de los SGBD Actuales". Ph. Doctoral Thesis, University of Granada (Spain), 1999 (www.lcc.uma.es).
- [6] J. Galindo, J.M. Medina, J.C. Cubero, M.T. García, "Relaxing the Universal Quantifier of the Division in Fuzzy Relational Databases". International Journal of Intelligent Systems 16(6), pp. 713-742, 2001.
- [7] C. Jensen et al., "A Glossary of Temporal Database Concepts". ACM SIGMOD Record, Vol. 23, 1, 1994.
- [8] J. Malik, T.O. Binford, "Reasoning in Time and Space". Proc. 8th Int. Joint Conf. Artificial Intell., Karlsruhe, pp. 343-345, 1983.
- [9] F.E. Petry, "Fuzzy Databases: Principles and Applications" (with chapter contribution by Patrick Bosc). International Series in Intelligent Technologies. Ed. H.-J. Zimmermann. Kluwer Academic Publ. (KAP), 1996.
- [10] Da-qun Quian, "Representation and Use of Imprecise Temporal Knowledge in Dynamic Systems". Fuzzy Sets and Systems, 50, pp. 59-77, 1992.
- [11] R. Snodgrass, "The TSQL2 Temporal Query Language". Kluwer, 1995.
- [12] J. Virant and N. Zimic, "Attention to time in fuzzy logic". Fuzzy Sets and Systems (82)1, pp. 39-49, 1996.