Bipolar Fuzzy Querying of Temporal Databases

Christophe Billiet[2], Jose Enrique Pons[1], Tom Matthé[2], Guy De Tré[2], and Olga Pons Capote[1]

[1] Department of Computer Science and Artificial Intelligence
University of Granada
C/Periodista Daniel Saucedo Aranda s/n E-18071 (Granada-Spain)
jpons,opc@decsai.ugr.es

[2] Department of Telecommunications and Information Processing,
Ghent University,
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
Christophe.Billiet,Tom.Matthe,Guy.DeTre@UGent.be

Abstract. Temporal databases handle temporal aspects of the objects they describe with an eye to maintaining consistency regarding these temporal aspects. Several techniques have allowed these temporal aspects, along with the regular aspects of the objects, to be defined and queried in an imprecise way. In this paper, a new technique is proposed, which allows using both positive and negative -possibly imprecise- information in querying relational temporal databases. The technique is discussed and the issues which arise are dealt with in a consistent way.

1 Introduction

Temporal databases are databases that handle certain temporal aspects of the data they contain [9]. In temporal databases, time-related attributes are generally not treated like the other attributes, though both describe properties of the same objects. This is because the time-related attributes are considered to have an impact on the consistency of the object set modelled by the database. The database will thus enforce a consistent behaviour towards time.

In the presented paper, the focus is on relational temporal databases. The necessity of the mentioned consistency is explained in the following example. Consider a car rental service with a database at it's disposal, which contains the properties of every car the service owns, including usual properties like car color, but also time-related properties like the starting and ending day of a car rental. If time is not handled specifically by the database, the rental office employees could insert several different records for the same car, each containing a different starting day and the same ending day. This would suggest that that same car was rented multiple times to possibly different clients at the same time, which is physically impossible, resulting in loss of consistency. A temporal database system would implicitly prevent this situation from occurring.

Now consider clients expressing their preferences for cars in both positive and negative statements and describing these preferences in an imperfect way.

E.g. a client could request a car which is 'about' 2 years old, preferably black, but certainly not yellow, and will be available during approximately all of next month.

In the previous example, as in almost every real-life application, human preferences are at the basis of every query. However, humans express their preferences in both positive and negative statements. Positive preferences express what is desired, acceptable or satisfactory, while negative statements express what is undesired, unacceptable or unsatisfactory. Depending on the situation, both can be used. This introduces the need for bipolar querying, a querying technique which allows introducing both positive and negative user preferences in a database query.

The combination of bipolar querying and the use of imprecise query preferences is well discussed in existing literature [5], [8], but not in the context of temporal databases. This paper will present an approach to query temporal databases using both positive and negative imprecise -and possibly temporal-preferences. In existing bipolar querying techniques, query results are ranked before presenting to the user, to provide the user with the best fitting results. In some existing techniques for querying temporal databases, records are either accepted or rejected as query results based on whether the queried time indication is totally covered by the record's time specification or not. The approach presented here will integrate a bipolar ranking technique with a technique to determine the degree of satisfaction of records to the query's temporal preference, to determine a record ranking best fitting for the user.

The rest of the paper is structured as follows. In section 2, some general concepts and issues of both temporal databases and bipolar querying are explained, both in the context of crisp and imprecise databases. In section 3, the presented approach is described and discussed and finally illustrated with an example. Section 4 contains the conclusions.

2 Preliminaries

2.1 Bipolar Satisfaction Degrees

As stated above, humans sometimes express their preferences using both positive and negative statements. Moreover, the positive and negative statements do not necessarily have to be each others inverse. E.g., when a user states that he doesn't want a blue car, it is not necessarily the case that he/she will be equally satisfied with any other color. This is what is called *heterogeneous* bipolarity [7], [8].

In fuzzy querying of regular databases, query satisfaction modeling is a matter of degree. Usually, criteria satisfaction is modeled by means of a satisfaction degree $s \in [0, 1]$, expressing the degree to which a particular database record is satisfactory according to a given criterion. It is implicitly assumed that the degree on which the database record is unsatisfactory according to the given criterion, is exactly the inverse of the satisfaction degree, i.e., $d = 1 - s$ with $d \in [0, 1]$ the dissatisfaction degree. This assumption leads to the concept of ‘symmetric bipolarity’. However, as explained above, this assumption does not always hold, leading to the concept of ‘heterogeneous bipolarity’ where satisfaction and dissatisfaction are not necessarily each others inverses. This can happen in two ways. First, there can be some indifference about whether some attribute values are satisfactory or not. In the extreme case the attribute value will be neither satisfactory nor unsatisfactory. E.g., in searching for a new car, a user might be totally indifferent about the satisfaction of a car regarding the color attribute when the car has the color ‘green’ (i.e., a green car will be neither satisfactory, nor unsatisfactory). In that case the overall satisfaction depends on the other attributes. Secondly, there might be some conflict in the user’s query specifications. In the extreme case, the attribute value will be satisfactory as well as unsatisfactory according to the user’s query specifications. Of course, these conflicting specifications are not desirable in query handling, but they can never be ruled out when trying to model human reasoning. To explicitly handle and model this ‘heterogeneous bipolarity’, the concept of a bipolar satisfaction degree has been introduced [15].

Definition A *bipolar satisfaction degree* (BSD) is a pair

$$(s, d),\ \ s, d \in [0, 1]$$

with s the *satisfaction degree* and d the *dissatisfaction degree*. Both s and d take their values in the unit interval $[0, 1]$ and are independent of each other. They reflect to what extent the bipolar representation corresponds to the concepts ‘satisfied’, respectively ‘dissatisfied’. The extreme values are 0 (‘not at all’) and 1 (‘fully’).

This definition is very closely related to Atanassov intuitionistic fuzzy sets (AFS) [2], except that the consistency condition ($0 \leq s + d \leq 1$) is missing. Because s and d are considered to be completely independent of each other, it is allowed that $s + d > 1$. The reason for this is that BSDs try to model heterogeneous bipolarity in human reasoning, and that human reasoning by definition isn’t always consistent.

Dealing with BSD’s in bipolar ‘fuzzy’ querying In this paper, we only study one specific kind of ‘fuzzy’ queries, namely those that are specified by two composed query conditions Q^{pos} and Q^{neg}. The case where one (bipolar) query condition can specify both positive and negative preferences at the same time, falls outside the scope of this paper. The condition Q^{pos} reflects all the positive (possibly ‘fuzzy’) preferences of the user, whereas the condition Q^{neg} expresses all the negative (possibly ‘fuzzy’) preferences of the user. As such, the composed conditions Q^{pos} and Q^{neg} can respectively be considered as poles of positive and negative conditions, where the conditions themselves can possibly be fuzzy.

A bipolar ‘fuzzy’ query $\tilde{Q}$ with positive and negative criteria is then formally specified by

$$\tilde{Q} = (Q^{pos}, Q^{neg}).$$

where Q^{pos} represents the logical expression of positive query conditions and Q^{neg} represents the logical expression of negative query conditions.

Ranking Processing a bipolar 'fuzzy' query as described above will, in the presented framework, result in an associated BSD for each database tuple. In order to rank these tuples in accordance with their global query satisfaction, a ranking function for BSDs is required. Different ranking functions can be used [15]. One of them, which gives equal importance to the satisfaction degree and the dissatisfaction degree, and which will be used in this paper, is

$$Rank_{BSD} = s - d \ \in [-1, 1]$$

Three special cases can be distinguished:

- $s - d = 1$: in this case it must be that $s = 1$ and $d = 0$, so this is the case of *full satisfaction* (without any indifference or conflict).
- $s - d = -1$: in this case it must be that $s = 0$ and $d = 1$, so this is the case of *full dissatisfaction* (without any indifference or conflict).
- $s - d = 0$: in this case the ranking is *neutral.* The criterion is as satisfied as it is dissatisfied.

2.2 Time in databases

The concept of time has been studied in databases for a long time. A true standard for adding temporal aspects to relational databases does not exist, but there is a consensus in the literature [9] on what is called a *temporal database*: a temporal database is a database dealing with some aspects of time in its schema. In a temporal DBMS, a **chronon** is the shortest duration of time supported by the system. In temporal databases, some temporal attributes can be managed without treating the attribute differently from non-temporal attributes. The time described by such an attribute is called **user defined time** (UDT). In addition to UDT, the following types of time can be discerned in a temporal database, all of which are handled exceptionally by the DBMS:

– **Transaction time** (TT) [20],[13] denotes the time when the fact (object) is stored in the database. It is usually append-only: as the past can not be changed, TT can not be changed neither. Furthermore, at the moment of insertion, a TT can be neither in the past nor in the future.
– **Valid time** (VT) [14],[21] denotes the time when the fact (object) is true in the modelled reality. A fuzzy extension has been proposed by [11].
– **Decision time** (DT, proposed in [17]) denotes the time when an event was decided to happen.

E.g., consider a database containing employee contract descriptions. The time when the employee's contract is valid, represented by an interval, is VT. The time when the employee's contract is stored in the database is the TT. The time when the decision for hiring this employee was made is the DT.

When working with these time concepts, the Data Manipulation Language (*DML*, which is part of the standard database querying language SQL) is extended to deal with possible temporal inconsistencies within the data and to handle more complex (temporal) queries. Depending on the time managed, a database is classified as either a **Valid Time Database** (*VTDB*), a **Transaction Time Database** (*TTDB*), a **bi-temporal database** (both valid and transaction time are managed) or a **tri-temporal database** (valid time, transaction time and decision time are managed).

Imperfection and time Representing imprecision and its semantics when dealing with time has been studied for a long time. Several proposals for representing and computing imprecise time indications can be found in [3] and [4]. Also, the changes between several granularities can be seen as a source of imprecision [6].

In the proposal section we will consider two kinds of imprecision:

- **Uncertainty in the database** denotes the uncertainty that arises when the knowledge about the temporal data in the database is uncertain. E.g., a database record shows that *'The car is in the garage around April.'*
- **Imprecision in the query specification** denotes the imprecision in the specification of temporal criteria by the user, when querying. E.g., *'The user wants to obtain a car which is red and which is in the garage around April.'*

Representation Several proposals for managing uncertain time in a database exist. Some proposals work with rough sets [19], other proposals rely on possibility distributions for representing uncertainty in time [11], [10]. In order to compare temporal possibility distributions, extensions of the classical Allen's operators [1] are defined in [18] and [16]. In the proposal section, we will follow the representation by means of possibility distributions, in order to work with both satisfaction and dissatisfaction degrees. Also, in order to work properly with fuzzy operators, the underlying domain should be numeric. In this paper, the representation for the dates will follow the Julian Day Number (JDN) representation [12].

If the starting point and/or the end point of the interval representing the time are not known precisely, it is easy to fuzzify them, using, e.g., two triangular membership functions. A **Fuzzy Validity Period** (*FVP*) is defined as a fuzzy time interval specifying when an object is valid. A fuzzy time interval is then the fuzzification of a crisp time interval. Several options to transform fuzzy time intervals corresponding to a fuzzy starting point and a fuzzy end point into one consistent fuzzy time interval exist [11], e.g (Fig. 1):

- The **convex hull** approach is the most intuitive approach. The resulting FVP is the convex hull of the union of both fuzzy sets.
- The **information preserving** approach is less intuitive but more realistic. The total amount of information is maintained at the edges of the membership function of the fuzzy time interval [11].

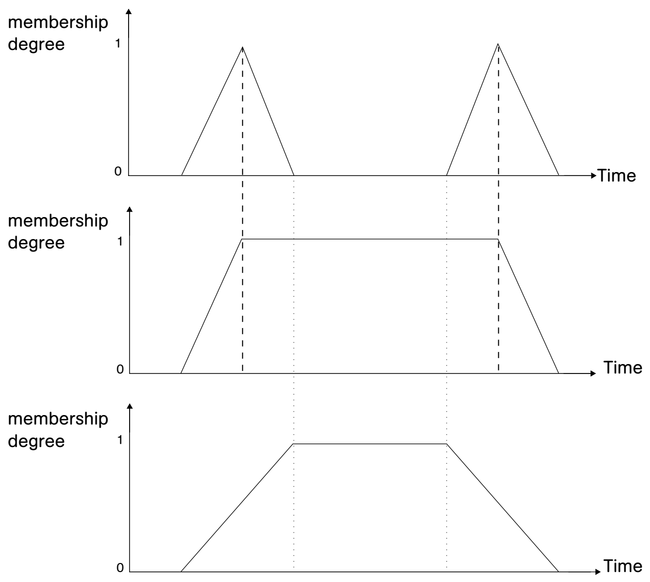

Fig. 1. Transformation to obtain the FVP. The top graph shows the two triangular membership functions. The middle graph shows the convex hull validity period, the bottom one shows the result of the second transformation, which maintains the information.

3 Proposal

In subsection 3.1, the context of this paper's approach is described, and the general ideas behind this approach are discussed. In subsection 3.2, the approach itself is presented and in 3.3, an example database and query are presented, which are used to illustrate the proposed approach.

3.1 Context

In this paper, relational VTDB's are considered, as the intention is to deal with cases in which the validity of an object is important knowledge, but the value of the transaction time is not.

Furthermore, a VTDB is queried by means of an extension of the bipolar 'fuzzy' querying approach introduced in [5] and briefly described in section 2.1. In accordance with [5], every non valid time attribute can contain imprecise data. To support valid time querying, the queries are extended to allow users to indicate a time period in which a result record is preferred to be valid. This introduces a consideration.

When departing from a query as specified in [5], a preference towards valid time can be inserted at two levels:

- **Local level.** Here, Q^{pos}, as well as Q^{neg}, are seen as logical aggregations of elementary query conditions and each of these elementary conditions is extended with a time demand. This means that every elementary condition is given an elementary temporal constraint which is evaluated and aggregated with the evaluation result of the condition. The introduced demands are defined by the user when the query is constructed and each one specifies a time constraint related with the fulfillment of the corresponding non-temporal condition.
- **Global level.** Here, the entire query is extended with a single time demand. This means that the entire query is given one elementary temporal constraint. The introduced temporal demand is defined by the user when the query is constructed and specifies a condition or constraint on the valid time period of a record.

In the proposed approach, only the last option is chosen. This is further described in section 3.2.

It should be clear that it is still possible to query user defined time, without modifying the model: the query specification remains the same and the temporal constraint is aggregated in the Q^{pos}- or Q^{neg}-expression, depending on the positive or negative nature of the time constraint.

3.2 Approach

The Query Structure The approach followed here introduces a global time demand: the user can define one time demand for the entire query. A query now has the following structure:

$$(Q^{time}, (Q^{pos}, Q^{neg}))$$

As in [5], Q^{pos} and Q^{neg} represent the positive and negative preference criteria, respectively, and Q^{time} represents the global temporal condition.

The user can specify the time demand using a time interval or a starting and end time. Both cases can contain imprecision concerning the starting and end times.

Time Imperfection It should be noted that in the presented approach, time will be represented as an interval with a starting point and an end point and a time point will be seen as a time interval in which the starting point and the end point coincide. Only the starting point and the end point are allowed to contain some imperfection. Even though the time domain can be discrete, a time interval will be seen as a continuous interval to easily allow the transition to FVP's.

The time specification in the query will instantly be translated into one trapezoidal FVP. The presented approach stays indifferent to the way in which this query FVP is calculated based upon a possible given crisp time period, but the interpretation of the query FVP should always be that the user prefers an object that is valid during at least the entire query FVP (the interpretation is conjunctive) and the certainty with which the user needs the object to be valid is expressed by the membership degree of the FVP.

The valid time indications in the database can take the form of one point, an interval or a FVP. All three options will be treated as FVP's. The interpretation is that the object described by the record in the database is valid during its entire FVP, but there can be uncertainty about the exact starting point and the exact end point of the valid time, which is modelled by the membership function of the FVP of the record (the interpretation is, again, conjunctive).

The Evaluation of the Query In the presented approach, every record r contains a trapezoidal FVP $\tilde{V}_r$ to express the record's valid time. The approach then is the following:

For every record r in the database, the bipolar query criteria Q^{pos} and Q^{neg} are evaluated, resulting in a BSD of the form (s_r, d_r), where s_r denotes the degree of satisfaction and d_r denotes the degree of dissatisfaction of the record. Separately, the temporal condition Q^{time} is evaluated in an attempt to define the degree to which r satisfies the query's temporal demand. Q^{time} contains a trapezoidal FVP $\tilde{V}_q$. For every record in the database, a validity satisfaction degree (VSD) $deg_{vs}(\tilde{V}_q,\ \tilde{V}_r)$ is computed using the fuzzy set gradual inclusion operator, with the minimum-function as t-norm.

$$\begin{aligned} deg_{vs}(\tilde{V}_q,\ \tilde{V}_r) &= deg(\tilde{V}_q\ \subseteq\ \tilde{V}_r) \\ &= \frac{card(\tilde{V}_q \cap \tilde{V}_r)}{card(\tilde{V}_q)} \end{aligned}$$

As time intervals are seen as continuous intervals, this formula is interpreted as follows.

$$deg_{vs}(\tilde{V}_q,\ \tilde{V}_r) = \frac{\int_{x \in U} min(\mu_{\tilde{V}_q}(x),\ \mu_{\tilde{V}_r}(x))dx}{\int_{x \in U} \mu_{\tilde{V}_q}(x)dx}$$

In this equation, U is the time domain and $\mu_{\tilde{V}_q}$ and $\mu_{\tilde{V}_r}$ are the membership functions of $\tilde{V}_q$ and $\tilde{V}_r$ respectively.

The interpretation is the following: as the interest here lies with the degree to which it is possible for the record to satisfy the query's temporal requirement, this method will compare the overlapping surface of the record's FVP and the query's FVP to the entire surface of the query's FVP. This results in a degree (VSD) measuring how much of the query's temporal requirement can be resolved by the record and thus how well the record fits the query's temporal demand. This VSD is now the requested degree to which r satisfies the query's temporal demand.

In the approach presented here, every trapezoidal FVP $\tilde{V}$ is represented using four values ($[\alpha_{\tilde{V}}, \beta_{\tilde{V}}, \gamma_{\tilde{V}}, \delta_{\tilde{V}}]$), where $\alpha_{\tilde{V}} \leq \beta_{\tilde{V}} \leq \gamma_{\tilde{V}} \leq \delta_{\tilde{V}}$. $\alpha_{\tilde{V}}$ and $\delta_{\tilde{V}}$ denote the beginning, resp. end of the support of $\tilde{V}$ and $\beta_{\tilde{V}}$ and $\gamma_{\tilde{V}}$ denote the beginning, resp. end of it's core. The calculation of the VSD for a record can then be reduced to a case study on the relative positions of the α, β, γ and δ values of the query FVP and the record FVP.

Presenting the Results to the User This approach was designed to present the user both the resulting records and the degrees to which these records satisfy his or her query. To discover the records which are the most useful to the user, the results have to be ranked. As the Q^{time} is evaluated independently from the (Q^{pos}, Q^{neg}) for every record, the ranking for the VSD will be determined independently from the ranking for the BSD for every record. The BSD's are ranked as presented in section 2.1. The VSD's are values in [0, 1] and thus have a natural ranking.

To achieve the final ranking, the rank of the total BSD ($Rank_{BSD}$) and the VSD of a record are combined using a convex combination of both ranks:

$$Rank_{Total} \;=\; \omega * Rank_{BSD} \;+\; (1-\omega) * VSD$$

This somewhat unusual approach allows the final ranking to show the effects of the BSD and the VSD in chosen proportions. Thus, by increasing the parameter ω, the non-temporal demands of the user can be given more importance and by lowering the ω, the time demand can be emphasized.

3.3 Example

The database. Consider a car rental service which uses the VTDB given in Table 1. Next to some general attributes (Color, Fuel consumption (noted F.C., in l/100 km), Age (in years)), the relation shows the FVP of each car in the $[\alpha,\ \beta,\ \gamma,\ \delta]$ - format explained above, where a null value in γ and δ means that the car is still valid right now. The ID field uniquely identifies a physical car, but the properties of a car can be modified: some engine optimizations could allow less fuel consumption, changes in the color could be made, etc. E.g. the car described in records 1 and 5 is the same (both records have the same ID value), but their colors differ. Therefor, a record in this VTDB will be uniquely defined by the combination of the ID field and the Instance ID (IID) field, which contains unique values for records with the same ID field value. A little remark: As the age of the car has nothing to do with when the car is available for rent, the 'Age'-attribute is considered UDT. Also, in this example, for the sake of simplicity, the chronons will be days.

The query. Consider the following query:

The user does not want to rent a black car, and prefers a red car which is either younger than 6 years or has an average fuel consumption that is around 5 or 6 litres/100 km and the car should be available around February 2011.

Using the structure (Q^{time}, (Q^{pos}, Q^{neg})) presented above, the query translates to:

$$(c^{time}, (c^{pos}_{Color} \wedge (c^{pos}_{Age} \vee c^{pos}_{Fuel}), c^{neg}_{Color}))$$

with

- $c^{pos}_{Color} = \{(Red,\ 1)\}$
- $c^{neg}_{Color} = \{(Black,\ 1)\}$
- $c^{pos}_{Fuel} = \{(5,\ 1),\ (6,\ 1),\ (7,\ 0.5)\}$
- $c^{pos}_{Age} = \{(0,\ 1),\ (1,\ 1),\ (2,\ 1),\ (3,\ 1),\ (4,\ 0.7),\ (5,\ 0.5),\ (6,\ 0.3)\}$

The temporal condition is then represented by the next trapezoidal FVP.

$$c^{time} = [\alpha_t,\ \beta_t,\ \gamma_t,\ \delta_t] = [25/01/2011, 1/02/2011, 28/02/2011, 10/03/2011].$$

Results and discussion. The example data set is shown in Table 1. The result set of the query, using the presented approach, is given in Table 2. Here, $Rank_{BSD}$ is the ranking of the BSD originating from the non-temporal bipolar constraints and $Rank_{Total}$ gives the total ranking (between -1 and 1) of the record with respect to the query, based on the chosen values for ω.

With $\omega = 0$, only the temporal constraint is taken into account. Because of the graduality of the VSD's, a richer ranking can be discerned than the simple acceptance or rejection which would occur when the query time interval would be requested to be fully contained in the record time intervals.

With growing values of ω, the non-temporal constraints grow in importance and the third record is ranked increasingly better than the fourth record, though the fourth record has a much higher VSD. This phenomenon can be desirable, as the fourth record has a much lower BSD rank.

With $\omega = 0.5$, only the record that scores high in both constraints gets a very high ranking (the fifth and the second records). Records with a low ranking for one of both constraints, are punished for this in the overall ranking, with the degree of punishment relative to the lowness of the ranking. Vice versa, low scores for either the temporal or the non-temporal constraints can be compensated for with high scores for the other constraints. This introduces a very natural ordering, which ranks the records with respect to both the temporal and the non-temporal preferences of the user.

4 Conclusions

In the presented work we have introduced the imprecise querying of valid time as a new criterion in a bipolar query specification. This criterion is independent of the positive and negative non-temporal preferences. Thus, an independent way of evaluating and ranking the temporal constraint is suggested, which takes the user's preferences into account. It is reasoned that a poor satisfaction of an imprecise time constraint can be compensated with a good satisfaction of other constraints. In view of this, a general ranking system is proposed and discussed.

Table 1. The valid time car relation with fuzzy validity periods (FVP)

ID	IID	Color	F.C.	Age	FVP
001	1	Black	6	4	[01/12/2010,10/12/2010,10/01/2011,19/01/2011]
002	1	Red	5	6	[20/12/2010,25/12/2010,25/02/2011,28/02/2011]
003	1	Blue	4	2	[27/02/2011,01/03/2011, - ,-]
004	1	Black	6	7	[28/12/2010,01/01/2011,05/03/2011,10/03/2011]
001	2	Red	6	4	[1/01/2011,11/01/2011, - , -]
002	2	Red	5	6	[16/02/2011,26/02/2011, - , -]
004	2	Black	6	7	[25/03/2011,05/04/2011, - , -]

Table 2. Result set of the query

ID	IID	$Rank_{BSD}$	VSD	$Rank_{Total}$				
				$\omega = 0$	$\omega = 0.25$	$\omega = 0.5$	$\omega = 0.75$	$\omega = 1$
001	1	-1	0	0	-0.25	-0.5	-0.75	-1
002	1	1	0.817	0.817	0.863	0.9085	0.954	1
003	1	0	0.142	0.142	0.107	0.071	0.036	0
004	1	-1	1	1	0.5	0	-0.5	-1
001	2	1	1	1	1	1	1	1
002	2	1	0.338	0.338	0.504	0.669	0.835	1
004	2	-1	0	0	-0.25	-0.5	-0.75	-1

Further research work includes introducing bipolarity in the temporal query specification. E.g. the user may request one time period but reject another one, when specifying the valid time constraint in the query. More complex relations may be modelled using Allen's [1] operators.

Representing valid time in a bipolar way is of particular interest in historical databases. In these databases, time is not always precisely known. Sometimes the information about the validity period is expressed in an imprecise way. Applying these techniques to historical temporal databases will improve the research in this field and is also a topic for further research.

Acknowledgements Part of the research is supported by the grant BES-2009-013805 within the research project TIN2008-02066: *Fuzzy Temporal Information treatment in relational DBMS.*

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM 26, 832–843 (1983)
2. Atanassov, K.T.: Intuitionistic fuzzy sets. Fuzzy Sets and Systems 20, 87–96 (1986)
3. De Caluwe, R., Van der Cruyssen, B., De Tré, G., Devos, F., Maesfranckx, P.: Fuzzy time indications in natural languages interfaces, pp. 163–185. Kluwer Academic Publishers, Norwell, MA, USA (1997)
4. De Tré, G., De Caluwe, R., Van der Cruyssen, B.: Dealing with time in fuzzy and uncertain object-oriented database models. EUFIT'97 pp. 1157–1161 (Sep 1997)

5. De Tré, G., Zadrozny, e.a.: Dealing with Positive and Negative Query Criteria in Fuzzy Database Querying Bipolar Satisfaction Degrees. In: Proceedings of 8th Int. Conf. FQAS. pp. 593–604. Springer Verlag Berlin, Denmark (2009)
6. Devos, F., Maesfranckx, P., De Tré, G.: Granularity in the interpretation of around in approximative lexical time indications. Journal of Quantitative Linguistics 5, 167–173 (1998)
7. Dubois, D., Prade, H.: Rough Sets and Current Trends in Computing, Lecture Notes in Computer Science, vol. 4259, chap. Bipolar Representations in Reasoning, Knowledge Extraction and Decision Processes, pp. 15–26. Springer, Heidelberg, Germany (2006)
8. Dubois, D., Prade, H.: Handbook of Research on Fuzzy Information Processing in Databases, chap. Handling bipolar queries in Fuzzy Information Processing, pp. 97–114. Information Science Reference, New York, USA (2008)
9. Dyreson, C., Grandi, F.e.a.: A consensus glossary of temporal database concepts. SIGMOD Rec. 23, 52–64 (1994)
10. Galindo, J., Medina, J.M.: Ftsql2: Fuzzy time in relational databases. In: EUSFLAT Conf. '01. pp. 47–50 (2001)
11. Garrido, C., Marin, N., Pons, O.: Fuzzy intervals to represent fuzzy valid time in a temporal relational database. Int. J. Uncertainty Fuzziness Knowlege-Based Syst. 17 (2009)
12. Husfeld, D., Kronberg, C.: Astronomical time keeping. http://www.maa.mhn.de/Scholar/times.html (1996)
13. Jensen, C.S., Mark, L., Roussopoulos, N.: Incremental implementation model for relational databases with transaction time. IEEE Trans. Knowl. Data Eng. 3, 461–473 (1991)
14. Jensen, C.S., Snodgrass, R.T., Soo, M.D.: The tsql2 data model. In: The TSQL2 Temporal Query Language, pp. 153–238 (1995)
15. Matthé, T., De Tré, G.: Bipolar query satisfaction using satisfaction and dissatisfaction degrees: Bipolar satisfaction degrees. In: Proc. of the ACM SAC'09 Conference. pp. 1699–1703. Honolulu, Hawaii, USA (2009)
16. Nagypál, G., Motik, B.: A fuzzy model for representing uncertain, subjective, and vague temporal knowledge in ontologies. In: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, LNCS, vol. 2888, pp. 906–923. Springer, Heidelberg (2003)
17. Nascimento, M.A., Eich, M.H.: Decision time in temporal databases. In: Proceedings of the Second International Workshop on Temporal Representation and Reasoning. pp. 157–162 (1995)
18. Ohlbach, H.J.: Relations between fuzzy time intervals. International Symposium on Temporal Representation and Reasoning 0, 44–51 (2004)
19. Qiang, Y., Asmussen, K., Delafontaine, M., De Tré, G., Stichelbaut, B., De Maeyer, P., Van de Weghe, N.: Visualising rough time intervals in a two-dimensional space. In: 2009 IFSA World Congress / EUSFLAT Conference, Proceedings (Jul 2001)
20. Rowe, L.A., Stonebraker, M.: The Postgres Papers. University of California at Berkeley, Berkeley, CA, USA (1987)
21. Sarda, N.L.: Extensions to sql for historical databases. IEEE Trans. Knowl. Data Eng. 2, 220–230 (1990)