

Towards an implicit treatment of periodically-repeated medical data

Bela Stantic^a, Paolo Terenziani^b, Abdul Sattar^a, Alessio Bottrighi^b, Guido Governatori^a

^a*Institute for Integrated and Intelligent Systems, Griffith Univ., Queensland, Australia*

^b*Department of Computer Science, Univ. Piemonte Orientale "A. Avogadro", Alessandria, Italy*

Abstract

Temporal information plays a crucial role in medicine, so that in Medical Informatics there is an increasing awareness that suitable database approaches are needed to store and support it. Specifically, a great amount of clinical data (e.g., therapeutic data) are periodically repeated. Although an explicit treatment is possible in most cases, it causes severe storage and disk I/O problems. In this paper, we propose an innovative approach to cope with periodic medical data in an implicit way. We propose a new data model, representing periodic data in a compact (implicit) way, which is a consistent extension of TSQL2 consensus approach. Then, we present query answering algorithms operating on it. Finally, we show experimentally that our approach outperforms current explicit approaches.

Keywords: Databases, Temporal information, periodic data

Introduction

Most clinical data (e.g., patients' clinical records) are naturally temporal. In order to be meaningfully interpreted, patients' symptoms, laboratory test results, and, in general, all clinical data, must be paired with the time in which they hold (called *valid time* henceforth). In many cases, medical data concerns events that have to be repeated at periodic time. Such events include, e.g., routine activities that nurses have to perform daily on hospitalized patients, as well as intrinsically repeated activities such chemotherapy cycles, or dialysis (which is usually an *open-ended* activity, since it has to be performed for all the life of certain diabetic patients). An explicit representation of all the repetitions to be performed might be important, e.g., for scheduling purposes and resource allocation. Nevertheless, it is very costly, both in terms of storage allocation, and of disk I/O when data have to be retrieved.

Periodic data in databases

Unfortunately, the research about temporal data has widely demonstrated that the simple addition of some timestamped attributes (e.g., the START and END times for the valid time of a tuple) is not enough, since many complex problems need to be tackled. For instance, Das and Musen have identified

several types of mismatches between the temporal support of standard databases and the richness of clinical data [Das and Musen, 97]; analogously, James and Goble have pointed out the requirements that medical records impose on a temporal model [James and Goble, 95]. Designing, querying and modifying time-varying tables requires a different set of techniques. Such techniques have been studied in more than 20 years of research by the temporal database (TDB henceforth) community (some general overviews are, e.g., [Gultekin et al., 95]). Although TDB is still an open area of research, many researchers have already consolidated a "basic core" of results, by defining the TSQL2 consensus approach [Snodgrass, 95].

In the medical area, several temporal database approaches have been devised. For instance, Chronus [Das and Musen, 94] and Chronus II [O'Connor et al., 02] have provided an implementation of a subset of TSQL2 [Snodgrass, 95], with specific focus on valid time. On the other hand, although actions repeated at periodic time are quite frequent in the medical context, no approach has been developed in order to cope with such data in an efficient way. For instance, since periodic actions are an intrinsic constituent of clinical guidelines, several approaches in the area have devised expressive language to represent complex periodic patterns (such as, e.g., those in chemotherapy treatments). Among the others, Asbru's [Duftschmid et al., 02] and GLARE's [Anselma et al., 06] temporal languages have been devised to model complex cases of periodically repeated actions. However, while in Asbru's and GLARE's languages repetition patterns in the guidelines can be represented, to the best of our knowledge no medical database approach has been devised to store in a (relational) database the actual data modelling the effective execution of repeated actions (e.g., dialysis) on each specific patients on which it has to be physically executed.

Explici vs. implicit approaches

The trivial way to store a repeated action in a database is to explicitly store all the repetitions of that action. E.g., consider the following therapy for multiple myeloma (such a therapy has been used as one of the example of application of GLARE's temporal representation language [Anselma et al., 06]).

(Ex.1) The therapy for multiple myeloma is made by six cycles of 5-day treatment, each one followed by a delay of 23 days (for a total time of 24 weeks). Within each cycle of 5 days, 2 inner cycles can be distinguished: the melphalan treatment, to

be provided twice a day, for each of the 5 days, and the prednisone treatment, to be provided once a day, for each of the 5 days. These two treatments must be performed in parallel.

While GLARE's representation language provides an high-level language to represent such a periodic pattern, a separate problem is to provide a proper support to store the time of execution the actions on specific patients affected by multiple mieloma. An *explicit* storage of all the actions (and the time when they have to be executed), although possible, is quite storage expensive. For instance, in a standard relational database approach it would consist, for each patient, of at least 90 tuples, modelling 60 melphalan applications, and 30 prednisone applications. While it is important that all such actions are recorded in some way (e.g., for scheduling purposes, and resource allocation), it is worth noticing that the main drawback of such an explicit approach is not just the waste of memory, but the increase of time devoted to physical disk I/O whenever such data need to be accessed. Additionally, from the logical point of view, an explicit storage of all the actions is not even possible in the case of open-ended repetitions, in which the end of repetitions is unknown (consider, e.g., the dialysis example). For such reasons, in the area of temporal databases, some initial approach has been devised in order to provide an implicit representation of periodically repeated data (consider, e.g., [Kabanza et al., 95; Bettini and De Sibi, 00; Terenziani, 03]). In such approaches, periodically repeated data are not explicitly elicited: on the other hand, the pattern of repetition is directly stored in the database, so that a compact representation is achieved.

However, to the best of our knowledge, no "implicit" approach to periodic data in the literature has explicitly focused on issues related to the efficient representation and management of periodical data. In this paper, we describe an approach overcoming such a limitation, with specific focus on medical data.

Methodology

In this paper, we propose an "implicit" approach to cope with periodical data, which is based on the "consensus" definition of granularity taken from the TDB glossary [Bettini et al., 98], and on its extensions to cover periodical data [Bettini and De Sibi, 00]. The generality of our approach is also granted by the fact that our representation model is a "consistent extension" of TSQL2 [Snodgrass, 95], the most famous "consensus" approach to temporal relational databases.

Our approach articulates as follows:

- (i) We identify a (relational) data model to store periodic data in an implicit way;
- (ii) we consider a "prototypical" class of queries (i.e., temporal range queries), and we address the problem of identifying a suitable query answering approach (with specific attention to the query answering algorithm);
- (iii) we extend the approach (at the algebraic level) to cope with other kinds of queries
- (iii) finally, we have developed an extensive experimentation of our model and methodology, showing that our "implicit" approach overcomes the performance of tradi-

tional "explicit" approaches both in terms of space and disk I/O's, and in terms of answer response time.

Temporal data model

In our approach, a periodic activities are *implicitly* represented through a new type of relation (that we term *periodical* relation), plus an additional relation, PERIODICITY, that we use to define periodicities.

Definition 1 (periodical relation). Given any schema $R=(A_1, \dots, A_n)$ (where A_1, \dots, A_n are standard non-temporal attributes), a periodical relation r is a relation defined over the schema $RP = (A_1, \dots, A_n \mid VT_s, VT_e, Per, Per_{id})$ where VT_s, VT_e , are timestamps representing the starting and the ending point of the interval of time containing all the repetitions (called "frame time" henceforth), Per is an interval, representing the duration of the repetition pattern, and Per_{id} is an identifier, denoting a periodical pattern in the PERIODICITY relation.

Definition 2. (PERIODICITY relation). The PERIODICITY relation is a relation over the schema $(Per_{id} \text{ Start, End})$, in which Per_{id} is a periodicity identifier, and $Start$ and End are temporal attributes (timestamps) denoting the starting and the ending points of the periods in the periodical pattern.

Example (Ex.2) As a simple example, let us suppose that an activity A1 has to be executed on a patient P1 each Monday, Wednesday and Friday for 10 weeks, starting from day 100, which is a Monday (for the sake of simplicity, here we use natural numbers instead of dates, and we assume that the base temporal granularity of the database is 'day'). Such an information is implicitly represented in our approach as shown in Tables 1 and 2 in the following.

ACTIONS

Action	Patient	VT _s	VT _e	Per	Per _{id}
A1	P1	100	169	7	Id1

Table 1. A periodical relation.

PERIODICITY

Per _{id}	Start	End
Id1	100	100
Id1	102	102
Id1	104	104

Table 2. PERIODICITY relation, concerning the periodicity in the example only.

In the ACTIONS relation, VT_s and VT_e states that the *frame time* is 70 days, from day 100 to day 169. The repetition period is 7 days (attribute Per). Per_{id} provides a link to the table PERIODICITY, in which the repetition pattern for the first week is stored. Notice that, although not explicitly stated, all the days in which action A1 has to be executed (on patient P1) can be inferred from the above implicit representation, looking at the pattern in the relation PERIODICITY as a pattern to be repeated each 7 days (Per attribute of the relation ACTIVITY), stopping repetitions after day 169 (see the explicit representation in Table 3) <end example>

It is important to notice that the temporal attributes of our periodical relations, in conjunction with the PERIODICITY relation, allows us to capture the implicit definitions of periodical granularities, as defined in the temporal database literature:

Property 1 (Expressiveness). Our extended data model can represent periodical granularities, as defined in [10].

Moreover, it is worth noticing that non-periodical temporal data could be easily represented as a degenerate case of the periodical one, using tuples in which VT_S and VT_E model the start and the end of the valid time, and the Per and Per_{id} attributes are set to NULL¹. Therefore, our approach can be seen as an extension of the “consense” TSQL2 approach [Snodgrass, 95], to cope also with periodic data.

Property 2 (consistent extension) Our data model is a “consistent extension” of TSQL2 data model.

Query answering: range queries

Here we take into account *range* queries since, according to the temporal database literature, they are particularly relevant. Specifically, the type of query we deal with is the following: given a set of periodical data (e.g., activities in the ACTIVITY table) and an interval denoting the span of time one is interested in the query (e.g., from day 120 to day 124), one wants to know which data holds during such a time period. In particular, in the context of periodical data, we identify two different types or range queries, depending on whether:

- (i) one is interested in the non-temporal part of the tuples only (e.g., What activities have to be performed from 120 to 124?)
 - (ii) one is interested in the tuples and in their explicit time (e.g., what activities have to be performed from 120 to 124?)
- For each of them, list all the times when they have to be performed, between 120 and 124).

For the sake of brevity, however, we will focus only on the type (i) of queries in the rest of the paper.

Given our (implicit) temporal data model, the process of answering such basic types of queries is quite complex, since we only have an implicit representation of data. Given a periodical relation r (e.g., ACTIVITY) and a query interval I_Q (e.g., [120,124]), in the following we sketch the algorithm we propose for efficiently answering queries of type (i):

- (1) For each tuple $t \in r$
- (2) Let P_t be the intersection between I_Q and the frame time of t
- (3) IF the duration of P_t is greater or equal than the period of t (attribute Per of t) THEN return t
- (4) ELSE
- (4.1) get in PERIODICITY the intervals constituting the repetition pattern of t
- (4.2) Using the ‘module’ function, “project” I_Q and the intervals retrieved at step (4.1) onto the same span of time, and check intersection

¹ Although such a representation is theoretically possible, for the sake of efficiency we store non-periodic data into standard TSQL2-like temporal relations, to avoid the use of unnecessary NULL values.

- (4.3) IF there is intersection, then return t

Notice that step (3) above is simply an optimization: in case the interval of interest (i.e., $P_t \cap I_Q$) is longer than the period of t , than for sure some of the intervals in the repetition must intersect the interval $P_t \cap I_Q$, so that the tuple can be directly provided in output, avoiding other checks. In (4.2), the module function is used to check intersection between the pattern and the interval of interest in an efficient way, avoiding an explicit generation of all the intervals of repetitions.

Query answering: temporal algebra

Besides temporal range queries, all kinds of relational queries must be possible on our new data model. Codd designated as complete any query language that is as expressive as his set of five relational algebraic operators: relational union (\cup), relational difference ($-$), selection (σ), projection (π), and Cartesian product (\times) [Codd, 71]. We propose an extension of Codd’s algebraic operators to query our data model.

Several temporal extensions have been provided to Codd’s operators in the temporal database literature [McKenzie & Snodgrass, 91; Snodgrass et al., 95]. In many cases, the extended temporal operators behave as standard non-temporal operators on the non-temporal attributes, and involve the application of set operators on the temporal parts. This approach ensures that the temporal algebras are a consistent extensions of Codd’s operators and are reducible to them when the temporal dimension is removed. For instance, in BCDM [Snodgrass et al., 95], which provides a uniform semantics underlying several temporal database approaches, including TSQL2, temporal Cartesian product involves pairwise concatenation of the values for non-temporal attributes of tuples and pairwise intersection of their temporal values. Analogously, in BCDM, relational union, projection and difference behave in a standard way on non-temporal attributes, and perform union (for relational union and projection) and difference on the temporal part of value-equivalent tuples. We ground our approach on such a “consensus” background, extending it to cope with periodic data. For the sake of brevity, we sketch only our definition of temporal Cartesian product. The other operators are defined in a similar way, according to the above-mentioned discussion. In the definition below, we denote by $t[X_j, \dots, X_k]$ the value of the attributes X_j, \dots, X_k in the tuple t .

Definition 3 (Temporal Cartesian product \times^T) Given two periodic relations r and s defined over the schemas $R1^P = (A_1, \dots, A_n \mid VT_S, VT_E, Per, Per_{id})$ and $R2^P = (B_1, \dots, B_k \mid VT_S, VT_E, Per, Per_{id})$ respectively, the temporal Cartesian product $r \times^T s$ is a periodic relation q defined over the schema $R3^P = (A_1, \dots, A_n, B_1, \dots, B_k \mid VT_S, VT_E, Per, Per_{id})$ containing, for each pair of tuples ($t_r \in r, t_s \in s$), a new tuple t' which is the concatenation of the non-temporal attributes of t_r and t_s (i.e., such that $t'[A_1, \dots, A_n] = t_r[A_1, \dots, A_n]$, and $t'[B_1, \dots, B_k] = t_s[B_1, \dots, B_k]$), whose frame time is the intersection of the frame times of t_r and t_s (i.e., $t'[VT_S] = \max(t_r[VT_S], t_s[VT_S])$ and $t'[VT_E] = \min(t_r[VT_E], t_s[VT_E])$), with $t'[VT_S] < t'[VT_E]$, whose periodicity $t'[Per]$ is the least common multiple of $t_r[Per]$ and $t_s[Per]$, and whose periodicity identifier $t'[Per_{id}]$ is a new system-generated identifier. The periodic pattern of $t'[Per_{id}]$ in the table

PERIODICITY is defined as the intersection of the periodic patterns associated with the identifiers $t_r[Per_{id}]$ and $t_s[Per_{id}]$, evaluated over a period of time which starts at $t'[VT_s]$, and whose duration is $t'[Per]$. Of course, only tuples such that frame times and periodic patterns have a non-empty intersection are retained in q .

The definition of temporal Cartesian product given above can be extended to temporal definitions of theta join, natural join, outer joins, and outer Cartesian products, in a way similar that done in [Gao et al., 05]. It is worth stressing that the consistent extension property also holds for our extended algebra:

Property 3 (consistent extension) Our temporal relational algebra is a “consistent extension” of the BCDM (and TSQL2) algebra.

Experimental results

In order to show the practical relevance of our implicit approach to efficiently manage periodic data, we have performed an extensive experimental evaluation. In particular, we have compared the performance of our approach with respect to the one of the standard explicit one. We remark here that, with the term “explicit” approach, we mean the approach in which periodic data are explicitly stored. For instance, the relation ACTIONS_Expl contains an explicit representation of the actions in example Ex.2.

ACTIONS_Expl			
Action	Patient	VT _s	VT _e
A1	P1	100	100
A1	P1	102	102
A1	P1	104	104
A1	P1	107	107
...
A1	P1	167	167

Table 3. Explicit representation of the periodic data in Ex.2. The relation contains 30 tuples.

Our results are computed on a four 450MHZ CPU - SUN UltraSparc II processor machine, running Oracle 10.2.0 RDBMS, with a database block size of 8K and SGA size of 100MB. At the times of testing the database server did not have any other significant load.

The RI-Tree [Kriegel et al., 00] has been used to index both time intervals both in the implicit and in the explicit approach, since this indexing methodology has been proved to have has the best performance regarding interval data.

We compare our results considering space usage, CPU usage, query response time, and physical I/O, which is usually considered to be the most important parameter while evaluating efficiency of accessing data [Hellerstein et al., 97].

In absence of real data, based on our experience, we have generated periodic data to simulate real medical scenarios. The following parameters have been considered (we used hour as the basic granularity):

- (1) Number of Patients: 16,824;
- (2) Average number of periodic activities per patient: 8.30;

- (3) Average number of periods in a periodical pattern: 4.86;
- (4) Average duration of period of periodical patterns: 87.56;
- (5) Average duration of the frame time: 1169;
- (6) Distribution of the duration of periodical pattern: we have provided different durations, with a prevalence of actions to be repeated daily (about 40%), and weekly (about 30%).

In order to carry on the experiments, the same periodical activities concerning hospital patients have been represented both in the implicit and explicit model. In the implicit model, the representation of data required 353,367 records in the ACTIONS table and about 2 million records in the PERIODICITY table. In order to represent the same activities in the explicit model, more than 194 million records are required in the ACTIONS_Expl table, so that, globally, the space requirement of the explicit approach is more than 100 times greater (see Table 4).

Table name	Number of records	Table Size (M Bytes)	Approach
ACTIONS	353,367	16.25	implicit
PERIODICITY	2,108,495	43.08	implicit
ACTIONS_Expl	194,671,463	7,331.82	explicit

Table 4. Comparing implicit vs explicit approach: space requirement.

Physical disk I/O’s, CPU time and response time for range queries of type (i) for different query duration and different answer sizes are show in Tables 5, 6, 7, and 8. Different range queries duration are considered to investigate effect of the optimization in step (3) of our query answering algorithm. Different answer size is considered to see the effect of clustering the data, aging of database buffers and effect of trade between Physical disk I/O and CPU usage.

Answer Size	Disk I/O	CPU	Response Time
2,068	6,049	477	5
7,471	10,831	1,431	12
17,738	12,364	3,186	32

Table 5. Evaluation of the implicit approach, for range queries lasting 1 hour.

Answer Size	Disk I/O	CPU	Response Time
2,068	3,031	73	3
7,471	12,872	315	30
17,738	31,904	930	154

Table 6. Evaluation of the explicit approach, for range queries lasting 1 hour.

Answer Size	Disk I/O	CPU	Response Time
2,887	1,838	200	3
4,620	2,114	260	5
29,455	9,490	1,375	26

Table 7. Evaluation of the implicit approach, for range queries lasting 1 week.

Answer Size	Disk I/O	CPU	Response Time
2,887	7,970	601	52
4,620	12,913	1,232	116
29,455	183,073	26,460	944

Table 8. Evaluation of the explicit approach, for range queries lasting 1 week.

As regards disk I/O (which is usually regarded to be the most important parameter in the database context [Hellerstein et al., 97]), our implicit approach is increasingly advantageous with respect to the explicit one. This is particularly true when the query range is bigger. This is because our approach can exploit the optimization at step (3) of the query answering algorithm. Also, with increased answer size our implicit approach outperforms explicit method particularly in number of physical disk I/O's, which results in significant shorter query duration to the extent of more than 36 times shorter query response time in case of 168 hour range query and answer size of 29,455 as it can be seen comparing the last raw of Tables 7 and 8.

An extensive experimental evaluation of our algebraic operators is still ongoing. Preliminary results confirm the advantages of our approach with respect to the explicit one, similar to the advantages above as regards temporal range queries.

Conclusions and future work

Temporal data play a fundamental role in medicine. Specifically, periodic data are frequent and important, so that their efficient treatment is a core issue in the area. We propose a new methodology based on an implicit representation, and on efficient query answering algorithms. We have experimentally shown that our approach outperforms the "traditional" explicit approach as regards disk I/O, CPU usage, and response time. The advantages of our approach increase with the increase of the answer size, and of the temporal range of the queries. As regards future work, we want to integrate our approach in GLARE (GuideLine Acquisition, Representation and Execution), a manager of clinical guidelines which strictly interacts with different databases, and devotes specific attention to the treatment of temporal data [Terenziani et al., 08].

References

- [Bettini and de Sibi, 00] C. Bettini and R. De Sibi. Symbolic representation of user-defined time granularities. *Annals of Mathematics and AI*, 30(1-4):53–92, 2000.
- [Bettini et al., 98] C. Bettini, C. Dyreson, W. Evans, R. Snodgrass, and X. Wang. A glossary of time granularity concepts. In *Temporal Databases: Research and Practice*, LNCS 1399, Springer-Verlag, 406–413, 1998.
- [Codd, 71] E. F. Codd, "Relational completeness of data base sublanguages, *Courant Computer Science Symposia 6*, Data Base Systems, Prentice Hall, 1971.
- [Das and Musen, 94] A.K. Das, and M.A. Musen: A temporal query system for protocol-directed decision support, *Methods Inf. Med.* 33(4) (1994) 358-70.
- [Das and Musen, 97] A.K. Das, M.A. Musen: A foundational model of time for heterogeneous clinical databases, *Proc. AMIA'97*, 106-110.
- [Duftschmid et al., 02] G. Duftschmid, S. Miksch, W. Gall: Verification of temporal scheduling constraints in clinical practice guidelines. *Artif. Intelligence in Medicine* 25(2) (2002), 93-121.
- [Gao et al., 05] D. Gao, C. S. Jensen, R. T. Snodgrass, and M. D. Soo, Join Operations in Temporal Databases, *VLDBJ* 14:2–29 (2005).
- [Gultekin et al., 95] Gultekin, Ozsoyoglu and R.T. Snodgrass: Temporal and Real-Time Databases: A Survey, *IEEE Transactions on Knowledge and Data Engineering* 7(4) (1995), 513–532.
- [Hellerstein et al., 97] J. Hellerstein, E. Koutsupias, and C. Papadimitriou. On the Analysis of Indexing Schemes. 16th ACM SIGACT-SIGMOD-SIGART Symposium on PoDS, 1997.
- [James and Goble, 95] R. James, C. Goble: Survey and critique of time and medical records, *Proc. Medinfo'95* (1995), 271-275.
- [Kabanza, 95] F. Kabanza, J.-M. Stevenne, and P. Wolper. Handling infinite temporal data. *J. of Computer and System Sciences*, 51:3–17, 1995.
- [Kriegel et al., 00] H.-P. Kriegel, M. Ptker, and T. Seidl. Managing intervals efficiently in object-relational databases. *Proceedings of the 26th VLDB Conf.*, 407–418, 2000.
- [McKenzie & Snodgrass, 91] E. McKenzie, R.T. Snodgrass: "Evaluation of Relational Algebras Incorporating the Time Dimension in Databases", *ACM Computing Surveys* 23(4), 501-543, (1991).
- [O'Connor et al., 02] M.J. O' Connor, S. Tu, M.A. Musen: The Chronus II Temporal Database Mediator, *Proc. AMIA'02*, (2002), 567-571.
- [Snodgrass, 95] R. T. Snodgrass (Editor): *The TSQL2 Temporal Query Language*, Kluwer Academic Publishers (1995), 674+xxiv pages.
- [Terenziani, 03] P. Terenziani. Symbolic user-defined periodicity in temporal relational databases. *IEEE TKDE*, 15(2):489–509, 2003.
- [Terenziani et al., 08] Terenziani P., Montani S., Bottrighi A., Molino G., Torchio M. Applying Artificial Intelligence to Clinical Guidelines: the GLARE Approach, in *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*, in *Studies in Health Technology and Informatics* 139, 273-282, 2008.

Address for correspondence

Prof. Paolo Terenziani, Dipartimento di Informatica, Università del Piemonte Orientale "Amedeo Avogadro", Via Teresa Michel 11, 15121 Alessandria, Italy. Tel. +39 0131 360174, email: terenz@di.unito.it – terenz@mf.unipmn.it