Application of Information Technology ■

Temporal Expressiveness in Querying a Time-stamp–based Clinical Database

DANIEL J. NIGRIN, MD, MS, ISAAC S. KOHANE, MD, PHD

Abstract Most health care databases include time-stamped instant data as the only temporal representation of patient information. Many previous efforts have attempted to provide frameworks in which medical databases could be queried in relation to time. These, however, have required either a sophisticated database representation of time, including time intervals, or a time-stamp–based database coupled with a nonstandard temporal query language. In this work, the authors demonstrate how their previously described data retrieval application, DXtractor, can be used as a database querying application with expressive power close to that of temporal databases and temporal query languages, using only standard SQL and existing time-stamp–based repositories. DXtractor provides the ability to compose temporal queries through an interface that is understood by nonprogramming medical personnel. Not all temporal constructs are easily implemented using this framework; nonetheless, DXtractor's temporal capabilities provide a significant improvement in the temporal expressivity accessible to clinicians using standard time-stamped clinical databases.

■ **JAMIA.** 2000;7:152–163.

As medical data are increasingly stored in a computerized, structured way, the ability to analyze large volumes of patient data has been greatly enhanced. Critical to these analyses is the ability to investigate time-based relationships between facts. Many previous efforts have attempted to provide frameworks in which medical databases could be queried in relation to time. Of these, two broad categories of approaches have emerged: creation of a sophisticated internal database representation of time (including time intervals), with an equally sophisticated temporal query language and database management system[1,2]; and use of a simple, time-stamp–based database architecture coupled with a sophisticated temporal query language.[3]

Affiliation of the authors: Children's Hospital, Boston, Massachusetts.

This work was supported by research fellowship award 1-F38-LM00055 from the National Library of Medicine and by equipment grants from Hewlett-Packard.

Correspondence and reprints: Daniel J. Nigrin, MD, MS, Division of Endocrinology, Children's Hospital, 300 Longwood Avenue, Boston, MA 02115; e-mail: ⟨nigrin_d@a1.tch.harvard.edu⟩.

Received for publication: 4/13/99; accepted for publication: 10/26/99.

The fundamental starting point of the work described in this paper is the following assumption: At present and for the foreseeable future, the temporal representation of most health care databases consists of *time-stamped events*, such as time of admission, time a laboratory specimen was received, time a laboratory result was reported, and time medication was dispensed. Far more rare are databases in which *intervals* are represented, such as the start and end of the administration of a medication, the duration of validity of a problem on the problem list, or the duration of a symptom. Similarly, very few clinical databases in production use any of the proposed or designed temporal extensions to the standard relational model. Yet the importance of rich temporal representations and query languages for reflecting effectively and accurately the course of a patient's health care has been repeatedly described. Consequently, the goal of the work described here has been to achieve a significant breadth of temporal expressiveness using existing time-stamped databases.

In working toward this goal, we developed a pragmatic hybrid of the two approaches mentioned in the first paragraph. Since the vast majority of existing medical databases contain time information that is represented as time-stamped instants, we felt compelled to use only this form of time information. In addition, we wanted to use standard SQL queries, so that the techniques could be easily implemented using existing off-the-shelf Web and database interface technologies and could be ported to other databases and institutions. We then constructed an intermediate representation of the results of queries made against the standard time-stamped clinical databases. We designed this intermediate representation (of sets of time-stamped instants) so that it could be easily queried by means of operators borrowed from a well-tested point-based temporal logic, the Temporal Utility Package (TUP),[4] which has been used in several medical applications.[5,6] This provided us with the full expressivity of TUP's point-based logic and, in addition, enabled us to express the entire set of interval queries described by Allen.[7,8] We were able to do this because it was previously shown that these interval relations could be expressed in a point-based logic, and we illustrate this again in this paper. Furthermore, by allowing for the iterative use of a series of simple time-based and Boolean operations, we were able to achieve much of the temporal expressiveness achieved by more complex temporal data models or databases and by more sophisticated query techniques. This is illustrated below through examples of queries that have been identified by other investigators as difficult to express in standard relational query languages.

In the sections that follow, we describe the underlying database that we used as well as the clinical application that populates and maintains it. We then describe the operation of the clinician-oriented data retrieval and mining tool, DXtractor, that we have developed. We explain its population query procedures as well as its first-class set operations and temporal querying capabilities. We describe how DXtractor's temporal operators map to those of the TUP and, by extension, to the interval representations of Allen. We then describe how DXtractor's temporal functions can be combined sequentially to achieve rich expression in the time-based domain. This is illustrated by clinical examples. Finally, we describe the general limitations of DXtractor's temporal operations.

The primary motivation for developing DXtractor was not the exploration of temporal expressivity in database query languages but, rather, a service mission. Clinicians and clinical researchers have many potential uses for the information stored in medical data repositories. Many of these uses, however, are not met by the traditional "results reporting" functionality that most electronic medical record systems provide. Broad population-based exploration with these systems is difficult and costly, in the use of both computational and programmer resources, since the systems center on single patient-focused retrieval applications. In addition, exploration based on temporal constraints is not commonly supported in current systems. It is important to note that population-wide "exploration" requests do not necessarily represent esoteric information needs. Many physicians, nurse practitioners, and administrative personnel routinely require this type of information in their everyday work.

Background

The historical relational data model (HRDM) provides an example of a highly sophisticated representation of the temporal validity of assertions to a database.[9] It allows a user or applications programmer to specify the temporal interval over which an assertion is valid at the table level, the tuple level, or the attribute level. This internal temporal representation provides the user with a detailed and accurate view of the chronologic history of the database. Unfortunately, there are no widely distributed industrial implementations of HRDM or similar temporally oriented database management systems.

Most medical databases use the temporal model proposed for the time-oriented database (TOD).[10] This model assigns each patient attribute a time-stamped instant, which corresponds to the time of the attribute's occurrence (e.g., the time of a clinic visit or the time a laboratory value was obtained). This triad of parameters (patient identifier, attribute, and times of occurrence) creates a three-dimensional, or cubic, view of the stored data. Many temporal operations can be performed on this form of data; temporal intervals, however, cannot be elicited from such a schema without the use of application-level processing.

Das and Musen[2] have introduced temporal database systems (TimeLine SQL and the Chronus System) that add the ability to support interval semantics. These systems provide a framework for both the storage and the manipulation of time-based clinical data, which includes specification of intervals of time. This approach calls for the addition of a temporal dimension at the level of the database tuple, with an associated temporal extension to SQL to allow queries of complex temporal features. Das and Musen demonstrate several classes of queries, which are either very difficult or impossible to express in SQL, that are enabled by these interval-based databases.

Apart from the literature on temporal databases, there have been at least two decades of research in temporal logics. This research includes the temporal logics of McDermott,[11] the interval logics of Allen,[7,8] and the point-based logics of Dean[12] and Kohane.[4] Kohane was able to show that most temporal relationships in medicine, could be expressed by simply specifying partial orders of point relations, including elaborate interval relationships and alternative possible chronologies.

Design and Implementation

CWS Database

The Clinician's Workstation System (CWS)[13] was introduced at Boston Children's Hospital in July 1991 and is still actively used. It allows for formatted data-entry of clinical notes into a relational database. These notes result from all patient visits to four ambulatory clinics at the hospital—the diabetes, general endocrine, general renal, and renal transplant clinics. Clinicians use a printed sheet to record notes during a patient visit; these sheets have several clearly defined areas for recording physical examination findings obtained during the visit. Administrative assistants then use CWS to transfer these findings, with a variety of other demographic elements, into the database. Dictated portions of the clinic notes are segmented by the assistants into sections (e.g., past medical history, assessment, and plan) and are entered into the database as free-text fields. Laboratory data are automatically transferred from the hospital's central laboratory repository into the CWS database daily. All data elements have an associated simple time stamp; no temporal intervals or other sophisticated temporal representations are coded into the database. In this regard, the CWS database utilizes the TOD model mentioned above.

Currently, the CWS database contains data for more than 10,000 distinct patients, representing more than 34,000 visits, 180,000 distinct clinical measurements, and 2 million laboratory findings. More than 12,000 problem list entries have been coded for this group of patients. This large database has already been successfully used in several clinical research projects.[14–16]

DXtractor

We have previously described DXtractor.[17] To summarize briefly, DXtractor allows for the effective retrieval of patient subpopulations from the CWS database. It queries for patient populations on the basis of specific clinical, demographic, and temporal constraints. Its functionality was conceptualized by the development, first, of a series of prototypical clinical queries that we, as clinicians, wanted to make of the database. Several of these queries, in clinical endocrinology, involve analyses that demonstrate significant temporal complexity:

- Generate a list of type I diabetic patients who have each had a glycohemoglobin (hemoglobin A1c, or HbA1c) value greater than 10 percent, excluding the first such value.
- Generate a list of all girls in the pediatric endocrinology program who developed Tanner stage II pubic hair at least one year before developing Tanner stage II breasts.
- Generate a list of all patients in the pediatric endocrinology program who are currently less than 18 years of age and whose last visit to a physician was between one and two years ago.

"Atomic" Queries

DXtractor's fundamental, or "atomic," query is a population query. Each patient group retrieved by a query is characterized by one or more clinical or demographic features. Each patient identified in a subpopulation has an associated list of one or more time stamps representing the transactional time of the queried event (Figure 1). This permits the user to execute temporal interval and point operations within and between retrieved patient population sets. This is described in greater detail below.

More formally, set S contains all unique pairs [Patient, Timestamp], where each time stamp represents the time of the event that satisfies the query criteria. Often, there are many time-stamped instances for a patient in a set. The set S may, therefore, be represented as:

[Patient$_A$, Timestamp$_{A1}$(, Timestamp$_{A2}$, . . . , Timestamp $_{An}$)]
[Patient$_B$, Timestamp$_{B1}$(, Timestamp$_{B2}$, . . . , Timestamp$_{Bn}$)]
.
.
.
[Patient$_N$, Timestamp$_{N1}$(, Timestamp$_{N2}$, . . . , Timestamp$_{Nn}$)]

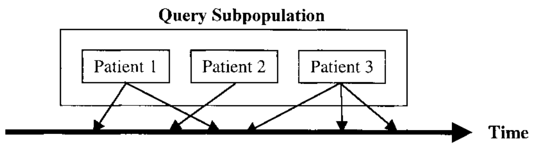
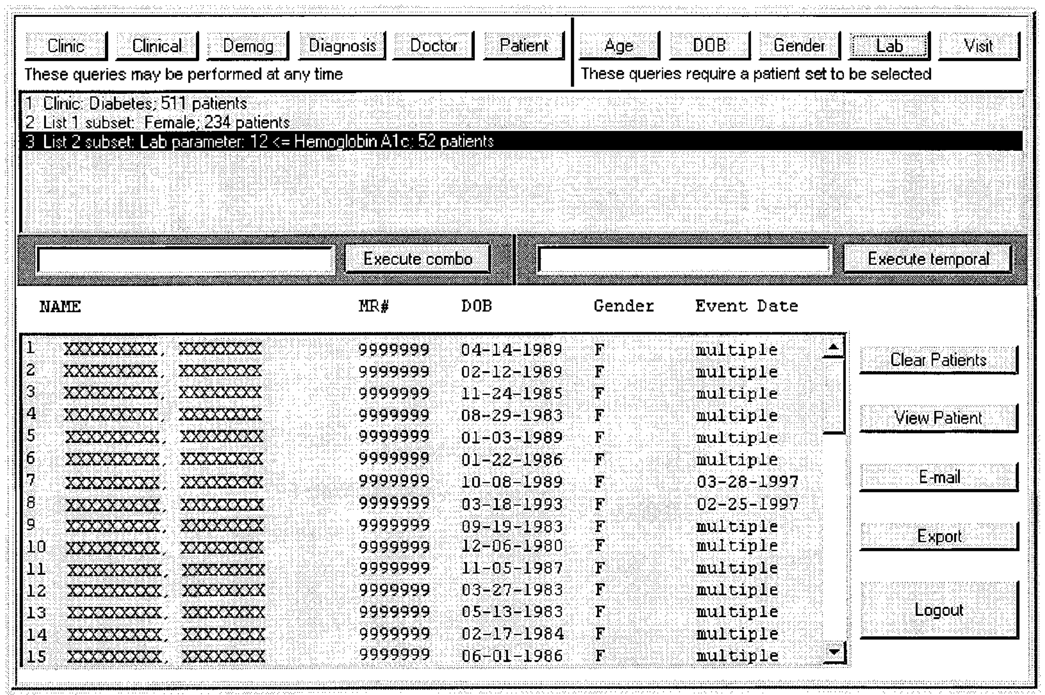


Figure 1 Diagram of a patient subpopulation generated from a Dxtractor query. Each patient may have one or more event dates, which represent the time stamps of the queried-for events.



These queries may be performed at any time

These queries require a patient set to be selected

1 Clinic: Diabetes; 511 patients
2 List 1 subset: Female; 234 patients
3 List 2 subset: Lab parameter: 12 <= Hemoglobin A1c; 52 patients

	NAME	MR#	DOB	Gender	Event Date
1	XXXXXXXXX, XXXXXXXX	9999999	04-14-1989	F	multiple
2	XXXXXXXXX, XXXXXXXX	9999999	02-12-1989	F	multiple
3	XXXXXXXXX, XXXXXXXX	9999999	11-24-1985	F	multiple
4	XXXXXXXXX, XXXXXXXX	9999999	08-29-1983	F	multiple
5	XXXXXXXXX, XXXXXXXX	9999999	01-03-1989	F	multiple
6	XXXXXXXXX, XXXXXXXX	9999999	01-22-1986	F	multiple
7	XXXXXXXXX, XXXXXXXX	9999999	10-08-1989	F	03-28-1997
8	XXXXXXXXX, XXXXXXXX	9999999	03-18-1993	F	02-25-1997
9	XXXXXXXXX, XXXXXXXX	9999999	09-19-1983	F	multiple
10	XXXXXXXXX, XXXXXXXX	9999999	12-06-1980	F	multiple
11	XXXXXXXXX, XXXXXXXX	9999999	11-05-1987	F	multiple
12	XXXXXXXXX, XXXXXXXX	9999999	03-27-1983	F	multiple
13	XXXXXXXXX, XXXXXXXX	9999999	05-13-1983	F	multiple
14	XXXXXXXXX, XXXXXXXX	9999999	02-17-1984	F	multiple
15	XXXXXXXXX, XXXXXXXX	9999999	06-01-1986	F	multiple

Figure 2 The Dxtractor main screen. Population queries are performed by selecting one of the buttons that appear across the top of the screen. The buttons have self-explanatory names like "Doctor," "Clinical," "Diagnosis," and "Lab," which select a group of patients based on, respectively, which doctor cares for them, a specific clinical finding on physical examination, a diagnosis, or a specific laboratory finding. The results of several queries are visible in the list of sets. Set 3 represents girls in the diabetes clinic who had a history of elevated glycohemoglobin (HbA1c) values—that is, values greater than or equal to 12 percent. (Names and medical record numbers have been masked to preserve confidentiality.) The Event Date column shows that many patients have had multiple occurrences of the elevated value.

where N is the number of patients that satisfy the query criteria and n is the number of time stamps for each patient that represent events that satisfy the query criteria. The value of n varies for each patient and must be at least 1.

DXtractor's main screen is shown in Figure 2. Results of queries are displayed as sequentially numbered one-line summaries in the top panel, the query summary area. When a query summary is highlighted in this area, the full list of patients is displayed in the lower panel, the query detail area, with a field called "Event Date" for each patient. The event date corresponds to the time stamp at which the queried-for event occurred. For example, when a query has been made for a specific diagnosis, the event date corresponds to the date the diagnosis was assigned to the patient; for laboratory queries, the event date corresponds to the date the specified laboratory test and range of values were obtained for the patient. If the patient had multiple occurrences of the queried-for event (i.e., $n > 1$), then the word "multiple" appears in the Event Date column.

Boolean Combinations

Although useful in themselves, the "atomic" queries become even more powerful when their results are combined. Logical set combinations may be formed in DXtractor using standard Boolean set operators (AND, OR, and NOT, including parentheses); these are entered in the field to the left of the "Execute combo" button shown in Figure 2. Query subpopulations are referenced by their query number, which is found in the query summary area. The logical operations are based on the patients contained in the query subpopulations. The time stamps of patients in the resulting set are always those that appeared in *both* the original sets (i.e., the union of the patient's time stamps). So, for example, if two query subpopulations are listed in the query summary area, then the Boolean expression "1 AND 2" will generate a new set, Set 3, which is the *intersection* of patients in the two original sets. The set of time stamps associated with each patient in this new Set 3 is the *union* of all time stamps for that patient in Sets 1 and 2. Similarly, "1 NOT 2" will generate a new set, which contains patients in Set 1 who are *not* in Set 2. These logical expressions may be of unlimited complexity (e.g., nested parentheses are allowed, as in "1 AND (2 NOT (3 OR 4))").

The formal relationships are defined below:

if Set A = {[Patient_A, Timestamp_{A1}(, . . . , Timestamp_{Ai})], . . .}
and all Set A's distinct patients are represented as Patients_A,

and Set B = {[Patient_B, Timestamp_{B1}(, . . . , Timestamp_{Bj}], . . .}
and all Set B's distinct patients are represented as Patients_B,

AND case:
then A AND B = {Patient_C, Timestamp_{C1}(, . . . , Timestamp_{Ck})], . . .}
where each Patient_C is in Patients_A and Patients_B,
and where $\text{Timestamp}_{C1,\ldots,Ck}$ is the set of all time stamps associated with Patient_C that appear in either Set A or Set B

OR case:
then A OR B = {Patient_C, Timestamp_{C1}(, . . . , Timestamp_{Ck})], . . .}
where each Patient_C is in Patients_A or Patients_B,
and where $\text{Timestamp}_{C1,\ldots,Ck}$ is the set of all time stamps associated with Patient_C that appear in either Set A or Set B

NOT case:
then A NOT B = {Patient_C, Timestamp_{C1}(, . . . , Timestamp_{Ck})], . . .}
where each Patient_C is in Patients_A and not in Patients_B,
and where $\text{Timestamp}_{C1,\ldots,Ck}$ is the set of all time stamps associated with Patient_C that appear in Set A

A simple example—that of finding all patients with diabetes and histories of elevated glycohemoglobin values—illustrates the use of a Boolean set combination. This becomes a simple task of first finding all patients with diabetes, then all patients with elevated glycohemoglobin values, and then performing a Boolean AND operation between them. This is illustrated in Figure 3.

A drawback of these Boolean set operations is that the temporal information contained in the combined set may no longer be meaningful, as shown in the example above. While the time stamps stored in Set 1 represent the date of a patient's diagnosis of diabetes, and those in Set 2 the date(s) of their elevated laboratory value(s), the combined temporal values have no easily definable meaning. This limits the use of Boolean-combined subpopulations in the temporal operations described next.

Temporal Combinations

Temporal combinations are a powerful adjunct to DXtractor's atomic queries and Boolean combinations. They are based on the fact that each patient identified in a query subpopulation has an associated list of one or more time stamps representing the time of the queried event. In providing temporal combinations, we permit the user to execute temporal interval and point operations within and between retrieved patient sets. These temporal expressions are entered into the field to the left of the "Execute temporal" button on Figure 2. The temporal operators BEFORE, AFTER, EQUALS, EARLIEST, and LATEST are available for these temporal operations.

Let us formally define these operators. First, we will concentrate on the EARLIEST and LATEST operators:

if Set A = {[Patient_{A1}, Timestamp_{A1a}(, Timestamp_{A1b}, . . . , Timestamp_{A1n})], . . .}

then EARLIEST/LATEST (Set A) = {[Patient_{A1}, $\text{Timestamp}_{A\ \text{Earliest/Latest1}}$], ...}
where $\text{Timestamp}_{A\,\text{Earliest/Latest1}}$ = EARLIEST/LATEST (Timestamp_{A1a}(, Timestamp_{A1b}, . . . , Timestamp_{A1n}))

For example, entering "EARLIEST 1" will create a new Set 2, containing the earliest event time stamp associated with each patient in Set 1 (Figure 4). Similarly, "LATEST 1" will retrieve the latest time for each patient in Set 1. Defining the starting and ending time stamps associated with a particular patient attribute makes the specification of event intervals possible. This expressiveness is further explored below.

The relative temporal operators (BEFORE, AFTER, and EQUALS) support time-based relationships between sets. Their formal definitions are:

if
Set A = {[Patient_{A1}, Timestamp_{A1a}(, Timestamp_{A1b}, . . . , Timestamp_{A1n})], . . .}
and whose distinct patient members are represented as Patients_A,

and
Set B = {[Patient_{B1}, Timestamp_{B1}], . . .},
and whose distinct patient members are represented as Patients_B,

then
Set C contains all [Patient, Timestamp(, . . .)] pairs

such that

Patients_C are members of (Patients_A AND Patients_B)

and
Timestamps_{Ci} are members of
$(\text{Timestamp}_{Aia}(, \text{Timestamp}_{Aib}, \ldots, \text{Timestamp}_{Ain}))$BEFORE/AFTER/EQUALS Timestamp_{Bi}
when Patient Ai = Patient Bi

where Set C's distinct patient members are represented as Patients_C,
and each of Set C's patients, *i*, has associated Timestamp_{Ci}

Entering "1 BEFORE 2" (and similarly "1 AFTER 2") will create a new Set 3, containing only patients for whom the event time stamp(s) in Set 1 comes before the event time stamp for that patient in Set 2 (Figure 5). Entering "1 EQUALS 2" retrieves those members of each set that have matching time stamps. In these examples, DXtractor requires that the time stamps in Set 2 be distinct for each patient; that is, no multiple time stamps are permitted in the second set of the expression. It is important to note that only patients in *both* sets are "eligible" to appear in the result set; this is equivalent to performing a Boolean AND operation between the sets prior to the temporal operation. Also, DXtractor allows for an absolute date to be specified as the second field in these relationships, as in "1 AFTER 07/28/1997."

The relative temporal operators can be further augmented by specifying the amount of time one event occurred before or after another, using the BY or WITHIN descriptors. For example, "1 BEFORE 2 BY 6 MONTHS" selects patients from Set 1 who have at least one event time stamp that precedes that patient's time stamp in Set 2 by at least six months. That is, the BY operator specifies a lower bound on the temporal relationship between the two point events. Similarly, "1 EQUALS 2 WITHIN 3 DAYS" selects for patients whose time stamps in the two sets differ by up to three days. That is, the WITHIN operator specifies an upper bound on the temporal relationship between two point events.

Combined Use of Techniques

Combined use of the Boolean and temporal operators allows for the specification of arbitrarily complex combinations of temporal interval and point patterns. Before proceeding to motivating clinical examples, we illustrate how DXtractor can be used to express the following point relation in the TUP:

(RREL ⟨p1⟩ ⟨p2⟩ ⟨lb⟩ ⟨ub⟩)

This is TUP's syntax, which specifies that there is a range relation (RREL) between point p1 and point p2

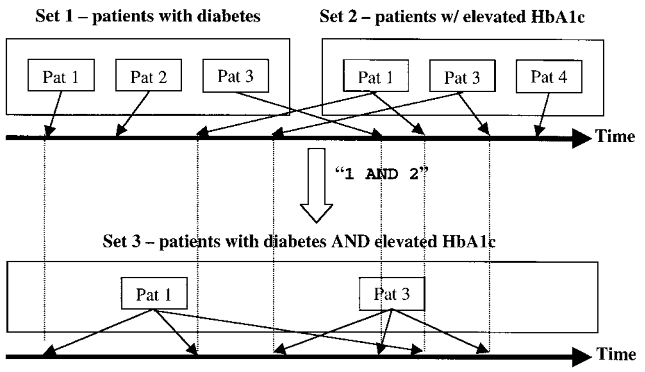

Set 1 – patients with diabetes
Set 2 – patients w/ elevated HbA1c
Pat 1
Pat 2
Pat 3
Pat 1
Pat 3
Pat 4
Time
"1 AND 2"
Set 3 – patients with diabetes AND elevated HbA1c
Pat 1
Pat 3
Time

Figure 3 Diagram of a Boolean combination of two retrieved patient subpopulations. Only patients present in both sets are present in the AND combination of the two. All time stamps for patients in both sets are passed to the output set, Set 3. After they are combined, these "event" time stamps may, however, no longer have a meaningful interpretation. HbA1c indicates glycohemoglobin, or hemoglobin A1c.

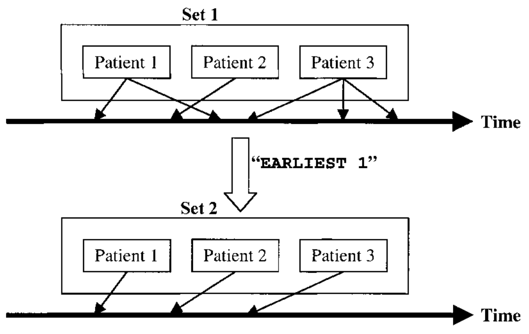


Figure 4 Diagram of the temporal set operation "EARLIEST 1." Only the earliest time stamp for each patient is passed to the output set. If a patient has only one associated time stamp in the original set, then that time stamp is passed to the output set.

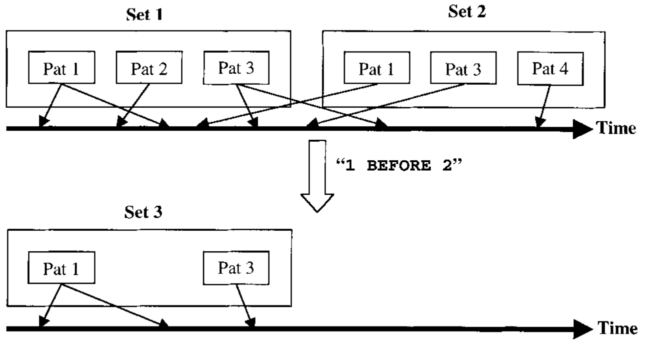


Figure 5 Diagram of the temporal set operation "1 BEFORE 2." Only patients in both sets are in the result set. For these patients, only the event dates that satisfy the temporal condition are passed to the result set.

that has a lower bound of lb and an upper bound of ub. The lower bound is always less than the upper bound. Possible values for bounds range from negative to positive infinity with all the real numbers in between. So, for example,

```
(RREL onset–headache Temperature–of–99F
   (−5 minutes) (2 hours))
```

specifies that the onset of headache occurred from two hours before the temperature of 99°F was recorded to five minutes after the temperature of 99°F was recorded, and

```
(RREL serum–sodium–measurement serum–
   potassium–measurement (0 seconds)
   (0 seconds))
```

specifies that the time of the two measurements was identical.

Consequently, if we perform the following two DXtractor queries,

```
1 BEFORE 2 BY 5 days
1 BEFORE 2 WITHIN 10 days
```

and combine their results with the AND operator, DXtractor will return all those cases in which 1 occurred 5 to 10 days prior to 2. This is equivalent to TUP's

```
(RREL 1 2 (5 days) (10 days))
```

In general, by combining results of the BY and WITHIN operators, we can express all bound pair values for the lower and upper bounds between two time points. This provides the expressivity of a point-based system such as the TUP.

Now that we have shown that DXtractor temporal operators can express the point relations of the TUP, we demonstrate the expressive power available through the use of DXtractor's simple combination mechanisms. We present several clinical examples that illustrate the following complex operations—ordinal selection, time windows, and two interval relations. Some of these operations have been identified by other researchers as being either awkward or impossible to implement using standard (e.g., not temporally enhanced) SQL.[18]

Example I: Ordinal Selection. How can the group of patients with type I diabetes who have had at least two glycohemoglobin (HbA1c) values greater than 10 percent be retrieved from the database? Such a query would benefit health care providers who may be managing a group of patients with diabetes, since it readily identifies those patients who have histories of poor glycemic control. The set of all patients who have each had only one glycohemoglobin value greater than 10 percent is less helpful, since many newly diagnosed diabetics will have had such a value at diagnosis. Therefore, the exclusion of the first such value would yield a more useful and meaningful group of patients. To create this list of patients, we could generate the following hypothetic subpopulations in DXtractor:

```
Set 1: Diagnosis: Type I diabetes mellitus;
       604 patients
Set 2: Lab: HbA1c > 10%; 516 patients
Set 3: Temporal: EARLIEST 2; 516 patients
Set 4: Temporal: 2 AFTER 3; 198 patients
Set 5: Boolean: 1 AND 4; 165 patients
```

Set 4 represents only those patients who have had at least two occurrences of glycohemoglobin values greater than 10 percent; if a patient only had one occurrence of the elevated laboratory value, they would not have a temporal value AFTER the EARLIEST one isolated in Set 3, and so would not be included in Set 4. Set 5 represents the goal group—only diabetic patients with at least two occurrences of glycohemoglobin values greater than 10 percent. This set could be further pruned to be even more clinically useful, perhaps by limiting it to those patients in whom the glycohemoglobin values occurred only within the last year, for example.

Ordinal selection of time points is also possible with one additional procedure; in the example above, the second occurrence of a glycohemoglobin value greater than 10 percent could be isolated by inserting the following step:

```
Set 5: Temporal: EARLIEST 4; 198 patients
Set 6: Boolean: 1 AND 5; 165 patients
```

The time stamp for each patient in Set 5 represents the date and time of the second elevated glycohemoglobin value, since Set 4 included only those patients who had two or more occurrences, and the EARLIEST of those must be the second. To limit this group to the diabetic patients only, we must again use AND on our final set with the group of diabetic patients, Set 1.

It is important to note that the temporal information stored in the final set (Set 6) is *not* the time of the second occurrence of the elevated glycohemoglobin value. This is because the Boolean combination combines all temporal information between the two sets and renders it uninterpretable.

If the *n*th occurrence of an elevated glycohemoglobin value is desired, this is just a recursive generalization of steps 4 and 5 in the last example. In many ways, this procedure is conceptually similar to the technique in the Lisp or Scheme programming language of "CDRing down a list" of temporal events until we arrive at the *n*th term in the sequence. We repeatedly strip away the first term in the sequence (the EARLIEST one)—the CAR—and leave the remaining terms to work with (those AFTER the EARLIEST)—the CDR. Although an *n*th operator is not itself implemented in DXtractor, repeated iterations of the procedure above effectively allow for retrieval of *n*th instances, if *n* is not especially large.

The previous example, however, contains an intentional flaw to allow for a more detailed analysis of DXtractor's capabilities. The retrieved sequence of temporal occurrences of elevated glycohemoglobin values does not guarantee that the first, excluded value occurred around the time of the patient's initial diagnosis of diabetes, and it is only these initial values that should be excluded. The query sequence must be refined, therefore, to exclude only those patients whose first elevated glycohemoglobin value occurred around the time (say, within one month) of their diagnosis. This can be done with the following hypothetic DXtractor query sequence:

Set 1: Diagnosis: Type I diabetes mellitus; 604 patients
Set 2: Lab: HbA1c > 10%; 516 patients
Set 3: Temporal: 2 AFTER 1; 405 patients
Set 4: Temporal: EARLIEST 3; 405 patients
Set 5: Temporal: 4 EQUALS 1 WITHIN 1 MONTH; 315 patients
Set 6: Temporal: 3 AFTER 5; 252 patients
Set 7: Boolean: 3 NOT 5; 90 patients
Set 8: Boolean: 6 AND 7; 342 patients

Set 3 represents all patients with diabetes who have had an elevated glycohemoglobin value at any time after their diagnosis date. Set 5 represents those patients for whom the earliest recorded high glycohemoglobin value was around the time of their diagnosis. Set 6 represents those patients who have had at least one *other* high glycohemoglobin value after the time of their diagnosis; this is part of our goal population.

The remaining patients for whom we are looking are those with one or more high values, the first of which did *not* come around the time of their diagnosis. This group is represented by Set 7. Finally, we join these last two groups in the final step and generate our goal population, Set 8. Restated, this patient set represents the group of all diabetic patients who have had at least one elevated glycohemoglobin value that is *not* coincident with the time of their diagnosis.

It is important to note that Sets 1 and 3 were used several times in this sequence of steps. This helps illustrate one of the advantages of performing complex queries in a stepwise fashion—that is, the intermediary query subpopulations may be used several times in the process.

Example II: Time Windows. The ability to determine the temporal relationship between two events is another important aspect of medicine. Often, it is desirable to know that two events occurred within a certain amount of time of each other or that they were separated by at least a certain amount of time. These "time windows" are completely specifiable by DXtractor. For instance, to find those patients with congenital hypothyroidism who were not seen in the endocrine clinic after their most recent elevated thyrotropin (TSH) laboratory result (i.e., a value greater than or equal to 10 μU/ml), the following hypothetic query sequence could be performed:

Set 1: Diagnosis: congenital hypothyroidism; 68 patients
Set 2: Lab: TSH ≥ 10 μU/mL; 262 patients
Set 3: Temporal: LATEST 2; 262 patients
Set 4: Visit: Visit ≤ {Current Date}; 3962 patients
Set 5: Temporal: 4 AFTER 3; 205 patients
Set 6: Boolean: 2 NOT 5; 57 patients
Set 7: Boolean: 1 AND 6; 6 patients

Set 3 represents patients with a history of elevated thyrotropin values and, for each, the associated time point that represents the most recent occurrence of an elevated value. Set 5 represents the patients who visited the clinic after their most recent thyrotropin elevation (the opposite of what we are looking for). Set 6 *excludes* the patients who were seen after the most recent elevation from all those who had elevated values. In our goal set, Set 7, we find the intersection of Set 6 and Set 1, those who have the diagnosis of congenital hypothyroidism.

This example demonstrates a qualitative temporal relationship. If we want to examine a more quantitative relationship, such as the group of patients who were seen within one month after an elevated thyrotropin level, we can simply modify Set 5 above to:

Set 5: Temporal: 4 AFTER 3 WITHIN 1 MONTH; 140 patients

Example III: Time Intervals MEETS and MET-BY. Finding the interval of time that a certain characteristic holds true for a patient is possible in the DXtractor environment. An example of this draws on the CWS clinical data described earlier. Let us determine the interval of time that a group of patients in a pediatric endocrinology clinic population was noted to have a minimal amount of breast development, described as "Tanner Stage 2 breasts." The following hypothetic query sequence in DXtractor generates the desired information:

Set 1: Clinical: Tanner breast = 2; 819 patients
Set 2: Temporal: EARLIEST 1; 819 patients
Set 3: Temporal: LATEST 1; 819 patients

For each patient in this subpopulation, the interval of time during which they had Tanner Stage 2 breasts is delimited by the starting date (the Event Date for that patient in Set 2) and the ending date (the Event Date for that patient in Set 3).

We may want to know whether this interval is punctuated by periods during which the patient did *not* have Tanner Stage 2 breasts. Clinically, this scenario occurs with girls who present initially to a pediatric endocrinologist with the chief complaint of premature breast development (Tanner Stage 2). If this problem presents in infancy, it usually resolves with no intervention and the child regresses to a prepubertal state (Tanner Stage 1). During a normal puberty, these children will eventually develop Tanner Stage 2 breasts again. They will then have been followed through the sequence of Tanner breast Stages 2, 1, and 2. In other words, the total interval of time represented by the earliest to the latest dates of Tanner Stage 2 really represents a concatenated series of two shorter intervals. To generate the list of patients in whom a Tanner Stage 2 breast characterization has been uninterrupted by a Tanner Stage 1 characterization (as described above), we can perform the following hypothetic query sequence:

Set 1: Clinical: Tanner Breast = 2; 819 patients
Set 2: Temporal: EARLIEST 1; 819 patients
Set 3: Temporal: LATEST 1, 819 patients
Set 4: Clinical: Tanner Breast = 1; 1427 patients
Set 5: Temporal: EARLIEST 4; 1427 patients
Set 6: Temporal: LATEST 4; 1427 patients
Set 7: Temporal: 6 BEFORE 2; 515 patients
Set 8: Temporal: 5 AFTER 3; 50 patients
Set 9: Boolean: 7 OR 8; 565 patients

Set 7 contains those patients for whom all episodes of "Tanner Breast = 1" occur before the first occurrence of "Tanner Breast = 2"; Set 8 contains patients for whom all episodes of "Tanner Breast = 1" occur after the last occurrence of "Tanner Breast = 2." The union of these groups (in Set 9) represents the goal population, those patients in whom the Tanner Stage 2 characterization has been uninterrupted by a Tanner Stage 1 characterization.

This example illustrates some of the 13 temporal relations described in Allen's interval algebra[7]—specifically, the MEETS and MET-BY relations. We could also have looked for those patients in whom the Tanner 2 characterization *was* interrupted by a Tanner 1 characterization. This would have illustrated Allen's DURING and CONTAINS relations. In a similar manner, judicious use of our temporal operators allows us to express the remaining nine relations—EQUALS, OVERLAPS, OVERLAPPED-BY, STARTS, STARTED-BY, ENDS, ENDED-BY, BEFORE, and AFTER. Previously, we have shown how all 13 interval relations specified by Allen can be expressed as conjunctions of upper and lower bounds on pairs of points (in this case, interval endpoints) using the range relation of the TUP. Since we have already shown that we can test for all values of upper and lower bounds between two time points (i.e., a TUP range relation), we can also query for all 13 interval relations. The next example illustrates the OVERLAPS and OVERLAPPED-BY relations.

Example IV: Time Intervals OVERLAPS and OVERLAPPED-BY. Some patients with insulin-dependent diabetes mellitus and a long history of very high glycohemoglobin values are at risk of developing renal complications, including proteinuria. To monitor for this condition, endocrinologists routinely test patients' urine for microalbuminuria. This finding may often persist even after patients have improved their glycemic control and glycohemoglobin values. The following query sequence finds patients with these findings—elevated glycohemoglobin value (greater than 12 percent), followed by onset of microalbuminuria, followed by a glycohemoglobin value less than 12 percent, but with persistence of microalbuminuria.

Set 1:	Lab: HbA1c > 12%; 112 patients
Set 2:	Temporal: EARLIEST 1; 112 patients
Set 3:	Temporal: LATEST 1; 112 patients
Set 4:	Lab: Urine microalbumin > 20 mg/g; 21 patients
Set 5:	Temporal: EARLIEST 4; 21 patients
Set 6:	Temporal: LATEST 4; 21 patients
Set 7:	Temporal: 2 BEFORE 5; 20 patients
Set 8:	Temporal: 3 BEFORE 6; 13 patients
Set 9:	Temporal: 5 BEFORE 3; 19 patients
Set 10:	Boolean: 7 AND 8 AND 9; 12 patients

This form of interval querying is possible in the DXtractor environment, even though neither the underlying database nor our query language directly supports interval representations. Even so, it is often not immediately obvious how to generate such results in DXtractor, primarily because of the many steps required. In the Discussion section we address how the complexity that may arise from the multistep composition of temporal operations can be simplified for users by storing complex query sequences for reuse.

Implementation

DXtractor is implemented in the Java computer language and makes extensive use of Java database connectivity (JDBC) methods for database operations. It is a password-protected applet and is accessed via a Web page using any Java-capable Web browser. The Web page lies within the Children's Hospital firewall, adding another level of security. The CWS data are stored in an Oracle 7 relational database. Execution times for queries range from approximately two seconds to ten minutes, depending on the complexity of the query and the number of patients retrieved. None of the individual queries contained in the examples above took longer than one minute. DXtractor employs multi-threading techniques to allow many queries to execute concurrently, preventing ''downtime'' while complex retrievals are being processed.

DXtractor is currently in its evaluation phase; members of both the endocrinology and nephrology divisions at Children's Hospital are using the application for a variety of clinical and basic science research efforts as well as clinical management and administrative tasks. DXtractor logs all user activity, and each user completes a brief exit questionnaire when they have completed their queries. This questionnaire asks whether users were able to retrieve the data they were seeking and whether they could have obtained these same data expeditiously using another retrieval technique (e.g., by manual record review or with the help of research assistants).

Discussion

We have described DXtractor, an application we developed to allow for population-based data mining by clinicians of a large clinical database. We then outlined DXtractor's basic operations and described how the temporal operators of DXtractor can be combined to create the expressivity of a point-based reasoner such as the TUP. Furthermore, we demonstrated how this expressivity allows DXtractor to test for the 13 temporal interval relations specified by Allen. We then gave detailed examples of how DXtractor can be used to address real clinical needs and questions, using existing time-stamped clinical databases.

The four examples also show how our application can generate subpopulations of patients based on complex temporal relationships. None of these temporal queries are unique to DXtractor; what is unique, however, is the way in which they are created. Most other approaches for temporal querying of databases use methodologies that perform the search using a single statement (possibly involving one or more subqueries). DXtractor's stepwise method allows for the same overall queries to be performed, but with multiple cascading queries. An advantage of this method is that the subpopulations generated ''along the way'' remain clinically meaningful and can themselves be useful to the clinician. Subpopulations may also be used several times in the course of a complex overall query.

DXtractor works with straightforward database implementations of Wiederhold's classic TOD model using standard SQL with no temporal extensions. Since many (if not most) medical institutions now maintain TOD-based databases, DXtractor's operations and architecture are available to them. This is in contrast to approaches that require at least a re-engineering of part of the database to permit extensions to its temporal representations. Although the SQL queries necessary to implement each of the ''atomic'' queries we have defined are specific to our institution, it would not be difficult to adapt them to other locations.

Initial use by nonprogramming clinicians has shown that the concept of set-based manipulation of patient populations is easily grasped. We speculate that this comes from a broad familiarity with the overall approach, since it is similar to that used in PaperChase,[19] Web search engines, and other routinely used computer applications.

We informally evaluated this by asking a group of four clinicians in the Division of Endocrinology at our institution to perform a group of clinical queries using DXtractor. The physicians were chosen on the basis of their perceived familiarity with computers. One has a strong computing background, including a degree in computer science, whereas the other three have only a standard working knowledge of commonly used computer applications (such as word processors and spreadsheets) and modest experience using the Internet. Each clinician was present at a one-hour seminar in which DXtractor was described and demonstrated, but no further help in executing the queries was given. Each clinician was asked to generate the following lists of patients:

- Your patients (patients primarily followed by you, not ones you may have supervised in clinic)
- Your female patients
- Your female patients with insulin-dependent diabetes mellitus who have a history of at least one glycohemoglobin value greater than or equal to 12 percent
- The patients who have both growth hormone deficiency and hypothyroidism and were 8 years old or older on Jan 1, 1998
- Of the patients found on the last query, those who have never had a thyrotropin value greater than or equal to 20 μU/mL

All four clinicians were able to successfully complete all the queries. Although clearly not a rigorous or comprehensive study, it does lend support to DXtractor's fundamental querying framework and indicates that other personnel could feasibly exploit the program's features, including both Boolean and temporal set combinations.

More sophisticated temporal queries may be awkward to run routinely because of their length and complexity. In this light, we are currently augmenting DXtractor to allow users to store commonly performed query operations. This will allow a series of linked steps to be performed with a single command. For instance, a diabetes nurse educator who routinely wants to analyze the group of diabetic patients with elevated glycohemoglobin levels will be able to store the set of queries included in Example I and retrieve this list of patients in one step, hiding the underlying complexity. Essentially, these linked steps will be implemented as macros for which parameters can be specified, for which users need to enter only the specific values that interest them. (For example, the diabetes nurse educator would simply input the worrisome glycohemoglobin value.) This macro language will also enable us to provide higher-level abstractions of the more complex temporal operations described in the examples above.

Finally, a programmatic interface to identified subpopulations will also be created, to provide standardized ways of delivering streams of subpopulation-specific data. This will allow other developers to create applications that exploit these data sources, including graphing, report generation, and machine learning tools.

There is a large class of temporal operations that DXtractor does not perform and has not been designed to perform. For example, it can generate only the most basic abstractions across intervals (e.g., an interval in which no glycohemoglobin value was less than 11 percent). It does not have, for example, the abstraction mechanisms of the Resume[20] or TrenDx[5,6] systems that would enable it to describe multiphasic trends over noisy data sets. Another temporal function for which it cannot test is periodicity of repetition of a particular set of events. There are almost certainly many more temporal queries that are difficult or impossible to make in the DXtractor framework. Nonetheless, DXtractor's temporal capabilities provide a significant improvement in the temporal expressivity accessible to clinicians using standard time-stamped clinical databases, capabilities that address many of the most routinely requested temporal data analyses in medical environments.

References ■

1. Combi C, Pinciroli F, Cavallaro M, Cucchi G. Querying temporal clinical databases with different time granularities: the GCH-OSQL language. Proc Annu Symp Comput Appl Med Care. 1995:326–30.
2. Das AK, Musen MA. A temporal query system for protocol-directed decision support. Methods Inf Med. 1994;33(4): 358–70.
3. Rucker DW, Maron DJ, Shortliffe EH. Temporal representation of clinical algorithms using expert-system and database tools. Comput Biomed Res. 1990;23(3):222–39.
4. Kohane IS. Temporal reasoning in medical expert systems. In: Salomon R, Blum B, Jørgensen M (eds). Medinfo. 1986: 170–4.
5. Haimowitz IJ, Kohane IS. Managing temporal worlds for medical trend diagnosis. Artif Intell Med. 1996;8(3):299–321.
6. Haimowitz IJ, Kohane IS. Automated trend detection with alternate temporal hypotheses. Proceedings of the 13th International Joint Conference on Artificial Intelligence; Chambery, France; Aug 28–Sep 3, 1993; pp 146–51.
7. Allen J. An interval-based representation of temporal knowledge. Proceedings of the International Joint Conference on Artificial Intelligence; Vancouver, Canada; Aug 24–28, 1981; pp 221–6.
8. Allen JF. Maintaining knowledge about temporal intervals. Commun ACM. 1983;26(11):832–43.
9. Tansel AU, Clifford J, Gadia SK, Segev A, Snodgrass RT (eds). Temporal Databases: Theory, Design, and Implementation. Redwood City, Calif: Benjamin/Cummings, 1993.
10. Wiederhold G, Fries JF, Weyl S. Structured organization of clinical databases. Proceedings of the American Federation of Information Processing Societies National Computer Conference; Anaheim, California; May 19–22, 1975; pp 479–85.
11. McDermott D. A temporal logic for reasoning about processes and plans. Cognit Sci. 1982;6:101–55.
12. Dean T. Temporal reasoning involving counterfactuals and disjunctions. Proceedings of the 5th International Joint Conference on Artificial Intelligence; Los Angeles, Calif; Aug 18–23, 1985; pp 1060–2.

13. Kohane IS. Getting the data in: three-year experience with a pediatric electronic medical record system. Proc Annu Symp Comput Appl Med Care. 1994:457–61.
14. Cohen LE, Wondisford FE, Salvatoni A, et al. A ''hot spot'' in the Pit-1 gene responsible for combined pituitary hormone deficiency: clinical and molecular correlates. J Clin Endocrinol Metab. 1995;80(2):679–84.
15. Kohane IS, Faizan K, Adjanee N, Najjar SS. Can cost effectiveness of growth hormone be improved? Pediatr Res. 1993;33(suppl):S51.
16. Gordon CM, Rowitch DH, Mitchell ML, Kohane IS. Topical iodine and neonatal hypothyroidism. Arch Pediatr Adolesc Med. 1995;149(12):1336–9.
17. Nigrin DJ, Kohane IS. Data mining by clinicians. Proc AMIA Symp. 1998:957–61.
18. Das AK, Musen MA. A comparison of the temporal expressiveness of three database query methods. Proc Annu Symp Comput Appl Med Care. 1995:331–7.
19. Beckley RF, Bleich HL. PaperChase: a computer-based reprint storage and retrieval system. Comput Biomed Res. 1977;10(4):423–30.
20. Shahar Y, Tu S, Musen M. Knowledge acquisition for temporal abstraction mechanisms. Knowledge Acquisition. 1992;1(4):217–36.