

# Abstraction-based Temporal Data Retrieval for a Clinical Data Repository

Andrew R. Post, MD, PhD, Ana N. Sovarel, MCS, and James H. Harrison, Jr., MD, PhD  
University of Virginia, Charlottesville, VA

## Abstract

*Disease and patient care processes often create characteristic states, trends, and temporal patterns in clinical events and observations, called temporal abstractions. Identifying patient populations who share similar abstractions may be useful for clinical research, outcomes studies, and quality assurance. In these settings, abstractions may be specific to a query, and thus allowing the specification of abstractions directly in the query would be desirable. We propose a query language for specifying and retrieving clinical data sets that allows specifying abstractions directly, and automatically selects data for retrieval based on the presence of abstractions inferred from the data. We describe the language and a prototype implementation, demonstrate its features with two queries constructed in response to clinical researcher-initiated data requests submitted to our institution's Clinical Data Repository, and report preliminary results from an evaluation of the implementation's performance.*

## Background

Health care institutions store large volumes of clinical data that are useful for understanding processes associated with disease, therapeutic response and patient care.<sup>1</sup> These processes may be reflected as sequences of time-stamped laboratory test results, medical observations, clinical orders, coded diagnoses and other data that are temporally related.<sup>2</sup> Common clinical data retrieval systems for standard relational databases do not provide tools for retrieving groups of patients whose data sequences have similar features.<sup>3</sup> As a result, patient characteristics that are represented as temporal relationships between data elements, and are not explicitly coded, are difficult to include in clinical, quality assurance or outcomes research studies.

The need for an improved ability to represent and query temporal relationships in databases has been recognized for many years.<sup>4</sup> Standard temporal extensions to Structured Query Language (SQL) have been proposed for specifying temporal relationships between time-stamped data elements,<sup>5</sup> but this effort has not been successful and the current SQL standard supports only limited temporal features.<sup>4</sup> In the medical domain, specialized query architectures and database schemata<sup>6,7</sup> have been developed that provide some features of Temporal SQL.

A separate temporal data analysis method has been developed for specifying and identifying temporal characteristics of clinical data, called temporal abstraction.<sup>8</sup> Temporal abstraction infers clinically meaningful states or processes in time-stamped data sequences based on value thresholds, frequency relationships, trends or other shared characteristics, and represents them as abstractions having time intervals over which the state or process exists. The method also supports specifying and identifying temporal patterns consisting of combinations of abstractions with specified temporal relationships.

Temporal abstraction software has been embedded into temporal query systems either to make abstractions available as temporal SQL data elements,<sup>6,7</sup> or to use temporal patterns directly as queries.<sup>9,10</sup> Those systems have required all state, trend, and other abstractions of raw data to be defined separately from the query, and their performance has been demonstrated only in settings where data pattern identification is applied to individual patients for monitoring tasks. While temporal abstraction supports storing links between found abstractions and the time-stamped data from which they were computed,<sup>8</sup> and existing systems may display raw data related to selected abstractions for specified patients,<sup>10</sup> the use of similar links for population-based clinical data retrieval has not been explored.

To address the need for flexible temporal query of clinical databases that allows the specification of abstractions within a data retrieval query, we have constructed a temporal query language based on temporal abstraction. We describe the language and a prototype implementation, illustrate the language's features with two real-world examples, and report preliminary data on the prototype's performance.

## Methods

### *Query Language*

The proposed language is supported by a query execution system that defines a data model comprising parameters, events, and constants. *Parameters* are measurable or observable time-stamped data values in a patient, such as clinical signs and symptoms, and laboratory test results. Values may be numerical or textual. *Events* are external acts upon a patient, such as procedures, diagnoses, and drug administrations. Events may be

time-stamped, or may occur over a time interval. Parameters and events may have any temporal granularity (e.g., minutes, hours). *Constants* are atemporal measurements or observations about a patient that are not generally assigned timestamps in clinical databases, such as birthdate, gender, and race. The query system provides for translating a database's local schema into these data types (see below) so that they may be referred to in queries.

A set of parameters, events and constants are pre-defined in the system for use in queries. For example, all laboratory tests used at our institution are parameters. Events include the International Classification of Disease (ICD-9) codes and Current Procedural Terminology (CPT) codes. A *VISIT* event specifies the time and duration of a patient visit. Constants include Birthdate, Gender, and Race.

Queries take the form of a **when** statement specifying a list of time intervals in patient data, and a set of abstraction definitions to be searched for within those intervals:

```
let abstraction1 := definition1
let abstraction2 := definition2
...
when interval_list
  let abstraction3 := definition3
  let abstraction4 := definition4
  ...
  retrieve abstraction3, abstraction4, ...
```

**Let** statements allow defining three kinds of abstractions: 1) event; 2) state, trend, and other “low-level” mathematical patterns in time-stamped data; and 3) temporal patterns.

1. *Event abstraction definition* statements have the syntax:

```
let eventAbstraction := event_list
```

where *event\_list* is a comma-separated list of events. Event abstractions categorize events (or other event abstractions), and are useful for specifying groups of procedure and diagnosis codes.

2. *Low-level abstraction definition* statements have the syntax:

```
let lowLevelAbstraction :=
  primitive(parameter constraint_list)
  [gap_constraint]
```

where *primitive* is the name of an abstraction algorithm (e.g., state, trend, rate, frequency), *parameter* is the name of the raw data type to process, and *constraint\_list* is a list of constraints on the values of *parameter* that are specific to the named *primitive* (see Results for examples of primitives). The available primitives are pre-specified by the query system, but are parameterized

with the constraints in *constraint\_list* within query statements. The syntax is designed not to constrain the set of primitives that could be provided by a particular implementation of the query system. An optional clause, *gap\_constraint*, specifies that adjacent abstractions should be concatenated if their nearest endpoints are within a specified time distance.

3. *Temporal pattern abstraction definition* statements have the syntax:

```
let temporalPatternAbstraction :=
  abst_param_evt [dur_constraint]
  [temp_rel abst_param_evt [dur_constraint]]
  and ... [gap_constraint]
```

and specify one or more temporal relationships (*temp\_rel*) between pairs of abstractions, parameters, or events (*abst\_param\_evt*) chained together by **and**. Allen's qualitative relationships<sup>11</sup> are supported, as are quantitative relationships such as 2 days to 2 weeks before. The optional *dur\_constraint* clause specifies constraints on the minimum and maximum time duration of an abstraction or event (parameters all have zero duration). The optional *gap\_constraint* clause works identically to that in low-level abstraction statements. Specified parameters, events, and abstractions may have different temporal granularities, which are automatically resolved and propagated to output abstractions.<sup>12</sup>

The **retrieve** statement retrieves data related to an abstraction, and has the syntax:

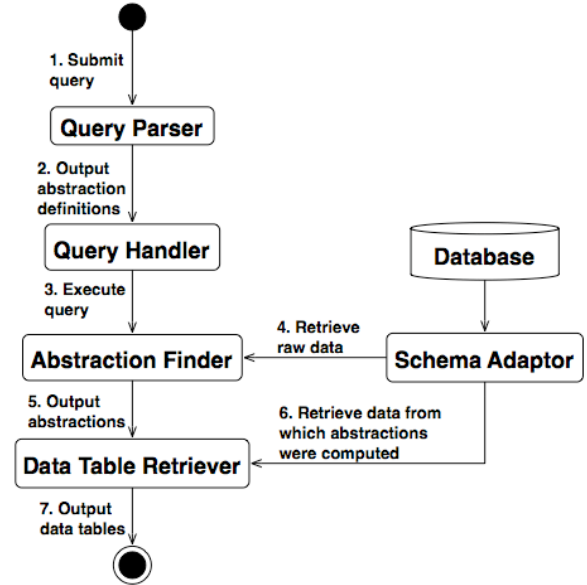
```
retrieve abstraction1 [limit
  data_type_or_abstraction_list1] [and
  abstraction2 ...]
```

where *abstraction* is an abstraction definition, and the optional *limit* clause specifies events, constants, parameters, or abstractions from which an abstraction was inferred, if retrieving only a limited subset of the data types supporting an abstraction is needed.

The **when** statement comprises an initial clause specifying the interval list, and a block consisting of one or more **let** statements and a final **retrieve** statement. The initial clause specifies events, abstractions and thresholds on constants chained together by **and**. The union of the temporal extent of those events and abstractions is used to generate intervals for each patient. Constants function only to exclude patients from processing if the patient does not satisfy the specified threshold of the constant's value, and do not factor into calculating the intervals' temporal extents. **Let** statements may precede the **when** statement to help specify the *interval\_list*. A **let** statement that precedes the **when** statement applies to the entire database.

## System Architecture

The query system’s architecture is illustrated in Figure 1. A query is submitted to the Query Parser (step 1 in Figure 1) and parsed into abstraction definitions. Those definitions are passed to the Query Handler (step 2). The Query Handler executes the query (step 3). The Abstraction Finder computes what parameters, events, and constants are needed in order to answer the query, and retrieves those data from the database (step 4) using the Schema Adaptor to translate between the database’s local schema and the data model. The Abstraction Finder scans the data for abstractions, propagating temporal granularities, and outputs those abstractions to the Data Table Retriever (step 5). The Retriever identifies the parameters, events and constants that contributed to an abstraction by following the links down the abstraction hierarchy (see Figure 2 for an example), retrieves those data over the union of the time spans of all identified parameters and events (step 6), and outputs three data tables for parameters, events, and constants (step 7, see Table 1 for an example).



**Figure 1:** Diagram of the query system. Numbered arrows indicate execution flow (see text).

## Results

A prototype of the architecture has been implemented in the Java programming language (java.sun.com), and has been used to retrieve data for two clinical investigators who requested data from our institution’s Clinical Data Repository.

### Query 1: HELLP syndrome

The first data request was to identify patient visits since 2000 with laboratory signs of HELLP (Hemolysis, Elevated Liver enzymes, and Low Platelets) syndrome, and to identify specific features of their platelet count profiles. HELLP is a dangerous complication of pregnancy that appears during the latter part of the third trimester or after childbirth.<sup>13</sup> There is no standard diagnosis code (ICD-9) for HELLP. Diagnosis and management are based on monitoring clinical symptoms and three clinical laboratory tests: platelet count (PLT), lactate dehydrogenase (LDH), and aspartate aminotransferase (AST). HELLP syndrome has been defined as pre-eclampsia with  $PLT < 100,000/\mu L$ ,  $LDH > 600 U/L$ , and  $AST > 70 U/L$ . Rising PLT indicates recovery. In consultation with the requesting physician, the data set of interest was specified as those patients’ PLT, LDH, and AST results, and their pregnancy-related ICD-9 codes. The following query retrieved the requested data:

```

let Pregnancy_ICD9 := ICD9_630, ICD9_631, ...
when Pregnancy_ICD9 during VISIT with
  startTime=2000
  let PLT_Low := state(PLT < 100)
  let AST_High := state(AST > 70)
  
```

```

let LDH_High := state(LDH > 600)
let PLT_Increasing := trend(PLT > 9 per
  day) with maxGap = 12 hours
let Lab_HELPP := AST_High within at most
  7 days of LDH_High and PLT_Low within
  at most 7 days of AST_High and
  PLT_Low within at most 7 d of LDH_High
  with maxGap=12 hours
let Recovering := PLT_Increasing starts
  at least 0 d after start of Lab_HELPP
  minDuration=12 h with maxGap=12 h
retrieve Recovering
  
```

The first **let** statement defines a `Pregnancy_ICD9` event abstraction as one of a list of ICD-9 codes (630, 631, etc.) used at our institution to code visits in which the primary reason for the visit was pregnancy.

The **when** statement’s initial clause specifies the intervals of interest, namely visits by patients since the year 2000 with a pregnancy ICD-9 code. Specifying the `VISIT` event constrains searching to occur only within each of the patient’s visits.

In the **when** statement block, the first three **let** statements define `state` abstractions specifying thresholds in PLT, AST, and LDH. The fourth statement defines a `trend` abstraction in PLT, and specifies that found abstractions occurring within 12 hours of each other should be concatenated. The fifth **let** statement defines a temporal pattern specifying that the laboratory criteria for HELPP occur within 7 days of each other, to reflect expert judgment that, if those results were more than 7 days apart, they were likely to have been caused by separate patient processes. The last **let** statement defines `Recovering` as a period of increasing platelets that starts after the laboratory criteria for HELPP have been met.

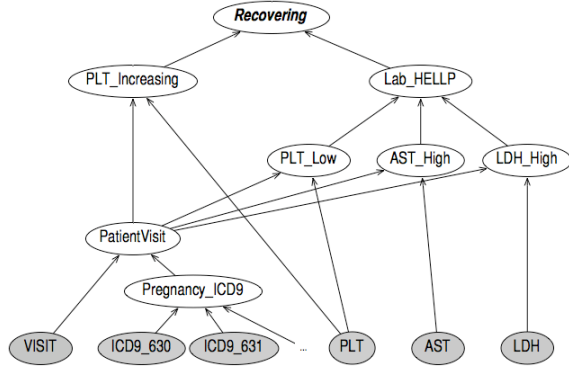


Figure 2. Abstraction hierarchy of the *Recovering* abstraction, showing which data types are retrieved (retrieved data types highlighted in gray). The *PatientVisit* node represents the visit intervals specified in the **when** statement.

Finally, the **retrieve** statement specifies the *Recovering* abstraction. As shown in Figure 2, *Recovering* is inferred from the *Lab\_HELPP* and *PLT\_Increasing* abstractions. *Lab\_HELPP* is in turn inferred from the *LDH\_High*, *AST\_High*, and *PLT\_Low* abstractions. The Data Retriever thus outputs the pregnancy ICD-9 codes, visit times, and PLT, AST, and LDH results over the union of their temporal extents. The PLT, AST, and LSH results populate the parameters table. The pregnancy ICD-9 codes populate the events table. The constants table is empty, as no constants were specified in the query.

### Query 2: Contrast reactions

The second data request came from radiologists who needed a retrospective dataset of creatinine (Creat) results in patients age 18 to 70 who had Computed Tomography (CT) scans with iodinated contrast material administration. In order to evaluate the extent to which the creatinine level changed after the CT scan, they requested only patients with at least three creatinine results: a first within 30 days before the scan to establish a baseline value; a second within 5 days after the scan to see the effect of the scan on the result, if any; and a third between 6 and 30 days after the scan to determine the resolution of any elevated creatinine levels. The following query retrieved the requested data:

```

let CTContrast := CPT_71275, CPT_72191, ...;
when CTContrast and Birthdate <= 1988-1-1
  and Birthdate >= 1936-1-1
  let CTContrastIval := CTContrast
    with maxGap = 5 days
  let Criterial := Creat 0 d to 5 d after
    start of CTContrastIval
  let Criteria2 := Creat 0 d to 30 d before
    start of CTContrastIval
  let Criteria3 := Creat 6 d to 30 d after

```

**Table 1.** Example data tables for a *Contrast* query. Val=Value; PatId=Patient id; Param=Parameter id, Tstamp=Timestamp, Event = Event id, Constant = Constant id, # = abstraction instance number.

### Parameters

PatId	Query	#	Param	Val	Tstamp
1111	Contrast	1	Creat	40	2001-1-1 1:30
1112	Contrast	1	Creat	1.1	2001-1-1 1:50

### Events

PatId	Query	#	Event	Tstamp
2222	Contrast	1	CPT_71275	2001-1-1
2222	Contrast	2	CPT_72191	2001-2-1

### Constants

PatId	Constant	Val
1111	Birthdate	"1970-8-5"
2222	Birthdate	"1980-2-1"

```

start of CTContrastIval
let Contrast := Criteria2 meets
  Criterial and Criterial starts
  Criteria3
retrieve Contrast

```

The **when** statement's initial clause specifies patients with one of the *CTContrast* CPT codes and the specified birthdate thresholds. We do not specify the *VISIT* event, so data is scanned across visits.

In the **when** statement block, the first **let** statement defines a temporal pattern specifying that CT scans occurring within 5 days of each other should be concatenated into intervals. This pattern reflects expert judgment that changes in Creat would reflect the first of a group of CT scans occurring close together in time. The next four **let** statements specify the requested time constraints on Creat relative to the first CT scan of a *CTContrastIval*.

The **retrieve** statement specifies the *Contrast* abstraction, which causes creatinine results and CT with contrast CPT codes within the specified 60-day window, and the patients' birthdates, to be retrieved. The creatinine results are reported in the parameters table, the CPT codes in the events table, and the birthdates in the constants table (see Table 1).

### Preliminary Evaluation

The HELLP syndrome query has been evaluated. In a database of 761 cases from 2000 to 2005 with pregnancy ICD-9 codes and elevated LDH (> 300 U/L), 81 cases met the laboratory criteria for HELLP, and all were confirmed as true positives by manual review. A 190-case subset with the ICD-9 code for severe pre-eclampsia was manually reviewed, and all cases within that subset were correctly categorized as HELLP or not HELLP by laboratory criteria.

## Discussion

We have constructed a query language based on temporal abstraction that allows specification of abstraction definitions directly in queries, and retrieves data supporting those abstractions. The approach avoids the need to construct all abstractions outside of the query language, as has been the case with previous temporal abstraction-based methods, and enhances flexibility in creating abstractions that are specific to individual queries. The need for externally specified primitive algorithms remains, although the language is not expected to significantly constrain the primitives that could be implemented.

Two real-world queries illustrate the approach by defining low-level abstractions with thresholds that are specific to a disease process, event abstractions representing query-specific groups of diagnosis and procedure codes, and temporal patterns expressed in terms of query-specific abstractions with defined qualitative and quantitative temporal relationships.

The proposed language is a prototype, and does not support many advanced features of temporally extended SQL such as statistical aggregators, nested queries, and grouping. Usability studies and further evaluation of the system's performance are warranted to evaluate areas for enhancement and to ensure that researchers' needs are met.

The language is suitable for use by clinical data repository research support personnel, and may also be appropriate for direct use by computer-savvy clinical researchers and physicians with training. In order to be useful to a broader range of clinical workers, a complete query system would likely need to include a graphical user interface supporting query creation and results review.

## Conclusion

We present an application of temporal abstraction to the retrieval of groups of patients and data appropriate for clinical research. This application demonstrates that the temporal abstraction method supports the creation of a temporal query language for clinical databases allowing specification of temporal abstraction definitions directly in queries, and data retrieval based on found abstractions. This approach may significantly enhance the value of clinical data repositories for obtaining data sets reflecting clinical states and processes that are not easily coded, or are expressed as changes in data over time or temporal relationships between data elements.

## Acknowledgements

This research was supported in part by National Library of Medicine grant R01 LM008192.

## References

1. Safran C, Chute CG. Exploration and exploitation of clinical databases. *Int J Biomed Comput.* 1995 Apr;39(1):151-6.
2. Shahar Y. Dimensions of time in illness: an objective view. *Ann Intern Med.* 2000 Jan 4;132(1):45-53.
3. Nigrin DJ, Kohane IS. Temporal expressiveness in querying a time-stamp--based clinical database. *J Am Med Inform Assoc.* 2000 Mar-Apr;7(2):152-63.
4. Adlassnig KP, Combi C, Das AK, Keravnou ET, Pozzi G. Temporal representation and reasoning in medicine: research directions and challenges. *Artif Intell Med.* 2006 Oct;38(2):101-13.
5. Snodgrass R, Bohlen MH, Jensen CS, Steiner A. Transitioning temporal support in TSQL2 to SQL3. In: Etzion O, Jajodia S, Sripada S, editors. *Temporal Databases: Research and Practice.* Berlin, NY: Springer; 1998. p. 150-94.
6. Nguyen JH, Shahar Y, Tu SW, Das AK, Musen MA. A temporal database mediator for protocol-based decision support. *Proc AMIA Annu Fall Symp.* 1997:298-302.
7. O'Connor MJ, Shankar RD, Das AK. An ontology-driven mediator for querying time-oriented biomedical data. 19th IEEE Intl Symp on Computer Based Medical Systems; 2006; Salt Lake City, UT; 2006. p. 264-9.
8. Shahar Y. A framework for knowledge-based temporal abstraction. *Artif Intell.* 1997;90:79-133.
9. Spokoiny A, Shahar Y. A knowledge-based time-oriented active database approach for intelligent abstraction, querying and continuous monitoring of clinical data. *Medinfo.* 2004;2004:84-8.
10. Shahar Y, Goren-Bar D, Boaz D, Tahan G. Distributed, intelligent, interactive visualization and exploration of time-oriented clinical data and their abstractions. *Artif Intell Med.* 2006 Oct;38(2):115-35.
11. Allen JF. Maintaining Knowledge about Temporal Intervals. *Commun ACM.* 1983;26(11):832-43.
12. Combi C, Pinciroli F, Pozzi G. Managing different time granularities of clinical information by an interval-based temporal data model. *Methods Inf Med.* 1995 Dec;34(5):458-74.
13. Sibai BM. The HELLP syndrome (hemolysis, elevated liver enzymes, and low platelets): much ado about nothing? *Am J Obstet Gynecol.* 1990 Feb;162(2):311-6.