Characterization of Temporal Sequences in Geophysical Databases*

Arie Shoshani[1], Preston Holland[2], Janet Jacobsen[2], Debasis Mitra[3]
Lawrence Berkeley National Laboratory
Berkeley, California 94720

Abstract

We examine here a specific scientific application that demonstrates the need for temporal concepts familiar to the scientists. By concentrating on a particular (geophysical data) application, we are able to identify in some detail realistic and useful temporal concepts. We propose a framework for the specification of time sequence properties and define operators classes. We then work out a case study in detail, and demonstrate the usefulness of these properties and operator classes. We conclude with several examples of queries over time sequences which are not typically supported by existing query languages and commercial database systems.

1. Introduction

There has been a lot of work in the last several years on temporal databases, see many references in [1]. There have even been proposals to extend SQL with temporal features [2]. However, much of this work is complex in nature and incorporates concepts that are foreign to scientists (such as "valid time" and "transaction time"). We explore in this paper temporal concepts as viewed by scientists. We choose a particular discipline, geophysical databases, as a way to concentrate on specific and realistic needs. However, as will become evident, many of the concepts are not unique to this application, and can be naturally found in other scientific applications.

Temporal sequences and spatial structures are common in geophysical databases. They often arise from experiments where some agent is injected into one well or borehole (such as water or radioactive solution) and measurements of the effect of that are taken at other boreholes (e.g. measure moisture, radioactive levels). Similarly, such measurements can arise from other stimulus, such as explosions, or simply collecting observations of the lithology (rock structure) of the sub-surface. Measurements are usually taken at regular intervals. Examples of temporal sequences generated from such experiments are shown in figure 1.

The first temporal sequence may represent an injection sequence, such as certain quantities of water injected for various length of time. Such a sequence can be thought of as "interval data", where a value is associated with each interval. The times that the water is injected are "irregular" in the sense that water can be injected at any time, not only at equally spaced time units. The value for each interval is "piece-wise constant" in this case. In general the value can be a function over time.

The second temporal sequence shown is an example of a measurement sequence that resulted from the injection of water. After some delay from the beginning of injection, the measured phenomena (e.g. moisture) starts rising. After the end of injection, the level falls, until the next injection. The measurements are taken at regular times, and the sequence is considered continuous in the following sense. A sequence is considered "continuous" if values exist for time points where no measurements were taken, but their values can be interpolated from neighboring measured values. Typically, all physical phenomena are naturally continuous. As an example of a discrete (non-continuous) sequence, consider a temporal sequence of the number of books sold per day. It is generally not possible to interpolate a value for a particular day from other days.

The third example is a temporal sequence of the drift in the measuring device usually collected for calibration at a later time. A drift in measurements is common depending on temperature (e.g. day measurements vs. night measurements) or other conditions. Drift data is usually collected at a lower rate, but is also typically regular and continuous.

The fourth example shows a sequence that represents the state of a device. It may be discovered at a later time that a device was malfunctioning, and this fact needs to be recorded. Similarly, there may be periods when the device is replaced, during which time any recorded measurements should be ignored (usually low "noise" values). Rather than storing this information as textual annotations in the

* This work was supported by the Department of Energy Under contract DE-AC03-76SF00098

1 Information and Computing Science Division
2 Earth Sciences Division
3 Jackson State University, Mississippi (visited LBNL during summer 1995)

database or recorded by the experimenter in a notebook, one can store it as a sequence with binary values (0, 1 representing "true" or "false"). The advantage is that such a sequence can be used by the database to mask the undesired portions of the measured values. Such a data sequence is considered an event binary data, where the values associated are "true" or "false".

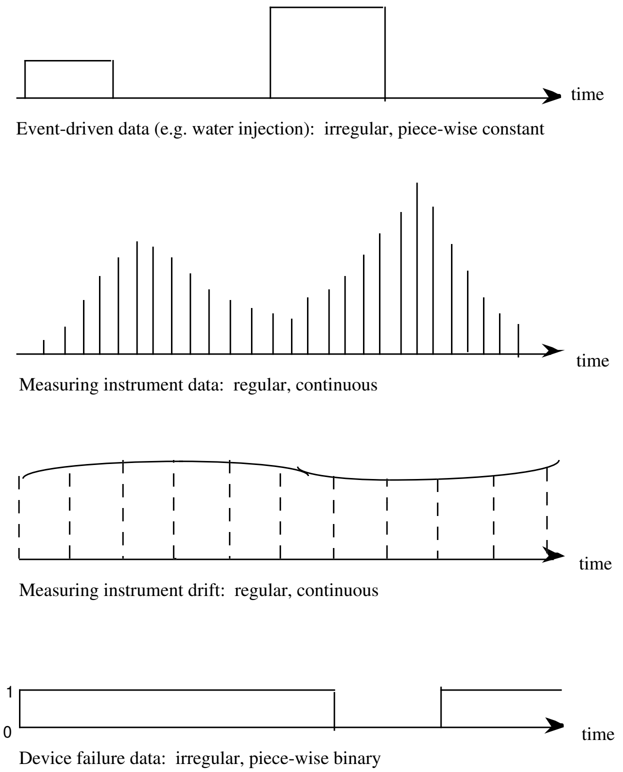

Figure 1: typical temporal sequences

Our purpose here is to characterize the properties and operations that can meaningfully be applied to such temporal sequences. Such operators can apply to a single sequence (such as computing a running average), or between sequences (such as multiplying two sequences for values in each time point). The properties described here are based on an earlier paper [3], but were refined in light of having to deal with a specific scientific application. Specifically, while in the previous model we defined some general purpose operators, we define here a larger class of operator types that are meaningful to scientific applications in general, and to geophysical temporal sequences in particular. In addition, we found it necessary to distinguish between granularity points and interpolation points of the model, and that the sequence types need to accommodate piece-wise-continuous sequences. These are explained in detail in the next sections.

2. Temporal features

As can be seen from the above examples, there are several features that we wish to have that characterize such temporal sequences. A temporal sequence is defined as a sequence of two-tuples $\{(e_1, d_1), (e_2, d_2), \ldots (e_n, d_n)\}$, where e_i's are *existing* time values and d_i's are corresponding data values. We define below the properties of time sequences.

2.1 Granularity. Granularity is the smallest unit where values can exist. Thus, if a measuring device is set to collect values every minute, then the granularity is 1 minute, whether or not values are actually collected every minute or every several minutes. The granularity of a sequence has a value g, and a unit of measurement u, associated with it. The set of all granularity points is referred to as g-points, and the values associated with these points g-values. The granularity points are the potential points where data values may be collected. Only the g-points that have actual values collected for them are called "existing" time points, or e-points.

2.2 Span. The span of a sequence specifies the extent of valid granularity points. The extent is specified as the g-values of the beginning and end g-points of the sequence. For example, a span of a measurement sequence that has a granularity of 10 minutes can be 8:00 - 20:30 (hours), meaning that measurements were taken every 10 minutes between 8:00 - 20:30.

2.3 Regularity. Regularity indicates whether every g-point has a data value associated with it. A sequence that is "regular" has no missing data values (we also refer to this as a "dense" sequence). An "irregular" sequence implies that data values are missing for some of the g-points. Note that irregular sequences must support the concept of a null value in the sense of "a value does not exist for a particular g-point". No distinction is made for the various reasons as to why a missing value does not exist (such as "unknown", "cannot exist for this point", etc.)

2.4 Density factor. The density factor is a measure that specifies an approximate percentage of data values that exist in a sequence. Obviously a regular sequence is 100% dense, and irregular sequences will have <100%. The importance of this factor is for physical organization, where different methods may be used to store and access the sequences depending on their density. It is an optional factor that may be left unspecified.

2.5 Type. The type of a sequence characterizes the behavior of the data points of the sequence. We find it useful to distinguish between interpolable types and non-interpolable type. The interpolable types we found useful are: continuous, piece-wise continuous, and piece-wise constant. The non-interpolable types we found useful are discrete, discrete-binary, and piece-wise binary.

The continuous type implies that the data values represent a differentiable function. The piece-wise continuous type permits the description of a sequence that is continuous in regions of the sequence (i.e. the function includes discontinuity points). The piece-wise constant is a special case where the function is constant over each region. A discrete sequence indicates that data values can not be interpolated because it does not make sense to do so for the given application. For example, if the sequence represent times when a device was replaced, then the sequence is considered discrete. Finally, a binary sequence has only two possible data values (0,1). There are two kinds of binary sequences: the piece-wise binary sequence is useful to represent interval events (such as periods that a device did not function properly), and the binary-discrete is useful to represent point events (such as days that the stock market was open or closed).

2.6 Interpolation factor. The interpolation factor is a fraction of the granularity size which specifies the sequence points for which we wish to interpolate data values. It is specified as an integer i, where g/i is the size of the fraction. For example, if g=30 minutes, and we wish to interpolate to the level of 5 minutes, then i=6. We refer to the set of all interpolation points as the i-points. Note that g-points are time points for which values can exist (measured), while i-points can only be generated by interpolation. This distinction is very important in scientific applications, because the g-points are dictated by the properties of the measuring devices.

2.7 Interpolation rule. For an interpolable sequence type, there might be a default interpolation rule available. Such rules can be incorporated into the system that support the temporal sequence data type. In addition, it should be possible for a user to attach a user-defined interpolation rule to a sequence. The interpolation rule can be used to allow operations over multiple sequences that have different granularity.

2.8 Compound sequences. A sequence is compound if it is made of multiple sequences, each with it's own properties. The need for this feature is that some sequences are collected in different segments at different granularity rates, and may have different types, interpolation factors, etc. for each segment. The implication of this type is that operators over compound sequences will operate piece-wise on each segment as if it was an independent sequence.

3. Operations over the time domain

It is convenient to categorize operations over the time domain into 4 classes, as described below. As will become evident, the operators are closed in the sense that they have input time sequences and they produce output time sequences. Thus, it is possible to apply these operators repeatedly over the results of previous operators. This feature is essential for any powerful query language.

3.1 Granularity invariant operations. These are operations over a time sequence which results in another time sequence of the same granularity and life span. Note that we permit the other temporal properties, regularity and type, to change as a result of the operation.

In general, we include in this category all operations that act as a "filtering" function over the sequence. For example, a "differentiation function" (i.e. di $=(x_{i+1}-x_i)/dt$) can be thought of as a filtering function. Another example is applying an interpolation function to an irregular sequence and producing a regular sequence with the same granularity. Globally useful filtering functions should be supported by the system. However, it should be possible for a user to apply a user defined filtering operator as well.

3.2 Granularity altering operators. These operators also generate a time sequence as an output, but alter the granularity of the input time sequence.

This category includes statistical operations over sequences of data values. For example, suppose that we have a sequence of measurements taken hourly, and we generate another sequence of the daily average value over the hourly values per day. In this case the operation generated a sequence whose granularity is different (day) than that of the input sequence (hour). Note that in this case the life span has not changed, but it is expressed in terms of the new granularity (i.e. days).

A special case of such operators is the generation of a single value for the entire sequence. Rather than considering the result as a scalar, we consider the result as a sequence with a single point. The reason is that we preserve the semantic properties of the result.

3.3 Life span altering operations. Operators that fall into this category are operators that select a part of the sequence by some criterion.

The criterion may be based on time (between t_1 and t_2), or on value (e.g. value is over x), or on patterns of values (e.g. monotonically increasing sequence of values). The criteria can in general be quite complicated. Here again, some common criteria types can be chosen to be supported. In addition, it should be possible to invoke user defined criteria. We provide several examples of such operations in Section 5.

3.4 Operations over two or more sequences. The categories of operations described above all deal with a single input sequence. In addition to these operations, it is necessary to define operations between two or more input sequences which generate a single output sequence. We consider only those operations which do not alter the granularity of the resulting sequence. The life span may change if it is not identical for the two input sequences. In case that an operation is needed over sequences of different granularity, then these sequences need to be transformed into the same granularity first.

An example of an operator over two sequences is the subtraction of one sequence from another for corresponding time points (e.g. to remove drift of the measuring device). Other examples may involve arithmetic operations or expressions. Here again, it should be possible to invoke a user defined function.

We claim that all the operations that we observed as useful in this application can be expressed as a sequence of the above operator types. We demonstrate this with examples in Section 5. Therefore, it is only necessary to provide these 4 basic operator types and their corresponding syntax structures. Furthermore, we believe that this basic operator types are sufficient for many other scientific applications.

4. A case study

In this section we describe an actual example of measurement patterns that are found in geophysical experiments. Water was injected into a borehole, and measurements of water moisture content were taken over time in a several adjacent boreholes.. We describe first the patterns in such measurements, and then the implications to the needed sequence properties and operators to process such patterns.

In figure 2a we show a typical pattern of a measurement sequence taken over many days in one of these boreholes. We make several observations about such patterns:

1) There are discontinuities in the sequence. This comes about because the measuring devices are battery operated, and need to be replaced periodically. During these discontinuity periods the recorded measurements are background noise. Sometimes, the researcher may be late in replacing a device, in which case the measurements are considered unreliable. In figure 2a the period shown between replacement is 12 days, but in practice it may be longer.

2) While each section of measurement corresponding to a particular device has a falling trend, in actuality it has a cyclical pattern to it per day. This is because the measuring device characteristic (and perhaps the flow of water) changes with temperature over a 24 hour period. This can be seen more clearly in Fig 2b which shows the details over a hourly measurements. The cyclical pattern repeats over 24 hour period.

3) The measurements may have "outliers" or missing values, as can be seen in Fig 2b for the point marked hour 20. This can occur for several reasons, such as a researcher tripping over a wire. Such outliers need to be found and removed.

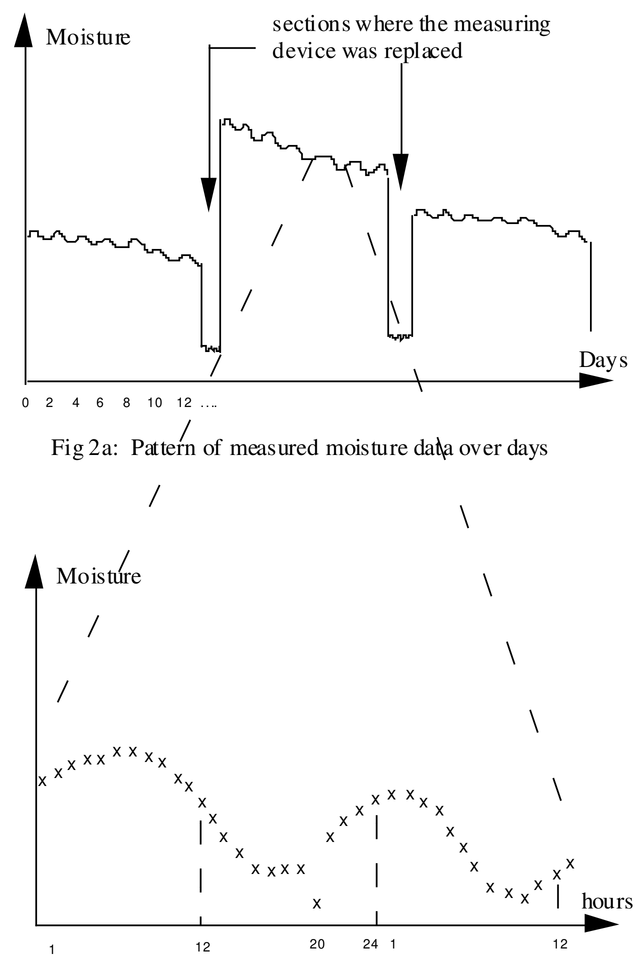

Fig 2a: Pattern of measured moisture data over days

Fig 2b: Details of pattern of moisture data over hours

4) In Figure 2a each section of measurements (which correspond to a particular device) may have a different base calibration. This is recorded separately as part of the description of each device properties. In processing a given sequence it is necessary to normalize the base calibration.

In order to process such temporal sequences for further analysis, several tasks need to be performed. We delineate these tasks below, which we label T1, T2, etc., and emphasize the time sequence properties and operators needed.

Task 1) Identifying the valid sections of the temporal sequences.

In order to achieve that, we need to have a time sequence of device replacements corresponding to each borehole. This is the information entered by the researcher at the time of device replacement. The sequence simply consists of a device identifier and the time of replacement. This is an example of an irregular, piece-wise constant sequence, with granularity of minutes (time of replacement).

This sequence is then operated on to produce another sequence with an hourly granularity, where the hour chosen is always the next integral hour. This is an example of a granularity changing operator, achieved with a simple temporal operator.

Since each device can be relied on for a certain period of time (12 days in this example) it is necessary to invalidate any measurements taken after 12 days and before the next device was installed. Thus, the above sequence needs to be turned into an hourly piece-wise binary sequence, where each segment is 12*24 hours. A gap exists between sequences if the installation of a device was not done on time (i.e. late by an hour or two). In some cases, a replacement of a device may have been forgotten until the next cycle, perhaps a day later. To achieve this, a "granularity invariant" operator that takes a piece-wise constant sequence and turns it into a piece-wise binary sequence, subject to the condition of 12 days valid periods, is needed.

Finally, an operator between the original sequence and the piece-wise binary sequence needs to be applied to generate a sequence of measurement of valid values. Note that the resulting sequence will end up with valid and invalid regions, a feature that needs to be supported by the sequence properties.

Task 2) Remove outliers

One method to identify outliers is by interpolating each value from its neighbors, and comparing it with the actual value. Such a function can be even provided as part of the operations supported by the system. In this case it was found that a more reliable method to identify outliers is by comparing values at the same time of day over successive days. This is because of the fluctuations due to temperature variations over a 24 hour period. This is an example where a user-defined interpolation function is needed. Because of the special nature of such sequences, this "cyclical" interpolation function may be written once and added to the library of callable interpolation functions.

Task 3) Correct for base calibrations

In order to correct for the base calibration of each device, we need to produce a time sequence of the base values of the device. This sequence can then be subtracted from the actual measurement sequence.

The sequence of base values of a device can be generated from the "time of replacement" time sequence (adjusted to hourly values) mentioned above. Recall that the values of this sequence were the identifier numbers of those devices. A separate table, usually produced at the laboratory, contains the information on base calibration per device before it is deployed (as well as other information, such as who has done the calibration, and when). It is only necessary to replace the identifier numbers with the base values to generate the time sequence of base values. This is an example of a utility that generates time sequences from ordinary tables, such as relations.

The operation of subtracting the base value time sequence from the measured time sequence is an example of an operator between two different time sequences. By capturing the semantics of valid and invalid sections, the details of the operations are taken care of automatically.

Task 4) Generate a smooth sequence

Finally, we may want to generate a smooth function for the behavior of the measurements in each borehole. First, one can use one of many interpolation functions (provided in the temporal sequence library) to produce missing values. Then, it is possible to apply a smoothing function (one of many filtering functions provided in the library) to remove the fluctuations that repeat every 24 hours.

These are examples of the ability to reuse fairly complex operators that are provided by a library. While such operators are common in scientific applications they are not usually provided as part of a database system library. We make the case here that such libraries are very useful when associated with specific data types such as time sequences. Furthermore, such libraries can be specialized for specific applications, and should be made flexible so that new operators can be added easily. Because of performance reasons it is usually better to support such operators and data types as files systems outside the database systems. However, some new database system technology are now designed to incorporate new data types and operators. The efficiency of such integration will determine their success for scientific applications.

We conclude this section with a discussion of the value of time sequence collections (TSCs). Virtually, all the operations described in the tasks above can be done on the entire collections of time sequences for all the boreholes. Thus, if the boreholes' time sequences were defined as belonging to a collection, TSC1, then all operations such as smoothing and interpolation can be applied to the entire collection. Further, if all the validity sequences per borehole (mentioned in task 1) were to be defined as belonging to a collection, TSC2, then the operations between members of TSC1 and TSC2 can be applied pairwise, borehole by borehole, in a single specification. Similarly, the sequences of base values of devices (mentioned in task 3), can also be defined as belonging to a collection, TSC3, and the subtraction of sequences in TSC3 from sequences in TSC1 can be made pairwise, borehole by borehole, in a single specification. This demonstrates the value of supporting operations over entire time sequence collections.

5. Example queries for geophysical databases

Once the temporal sequences were processed, various operations over such sequences can be useful as part of a query and analysis capability. We present here several examples that are useful in the context of geophysical applications, in order to demonstrate how those differ from conventional operators provided by most query languages.

As mentioned above all the example operators discussed below can be applied to a time sequence collection (TSC) or operate pairwise between the corresponding members of two or more TSCs.

1. Find time regions where the (absolute) value is greater than x.

This is a very simple and common predicate, but when it is applied to a time sequence the result is another time sequence that is piece-wise binary. The "true" value of the sequence is for the regions where values were over x. Another variation is to produce a time sequence that preserves the values that are above x, in which case the result is a step-wise continuous sequence, where the sections below x are marked as "invalid". An example for the use of this function is in order to find obvious outliers.

2. Find the time points where the value is y% higher (or lower) from that of a given time point.

This is similar to the previous predicate except that the value for the comparison is taken from a specific time point. This function can be used to identify an unusual phenomenon we are looking for. When applied over a TSC, one can check if this phenomenon is consistent over multiple boreholes.

3. Find the time points where value is y% higher (or lower) from its preceding point.

This example can be used to identify quick rises (or falls) in the measurements, such as discontinuities in sequences. The significance of this example is that it represents a "running" operation; that is, it is applied to all points of the sequence one at a time. Variation on this type of operation can be to find points that are y% higher than the average of the last 3 points. It can also be used to generate "running averages" (or other "aggregate" functions), such as producing the average of the last n values for each time point.

4. Find an extent of length x where the last value is at least y% higher (or lower) than the first value.

This is an example that is common in geophysical applications. The phenomenon we wish to identify is a significant rise (or fall) of values over certain period of time.

5. Generate average value over a set of time sequences.

This is an example where an operation is applied over a set of time sequences, but the operation is applied to values of the same time points in different sequences. Such operations are usually "aggregate function". For example, generating the average for the same time point over the entire collection of borehole time sequences may also identify outliers that cannot be identified by looking at a single sequence at a time.

6. Find patterns

Identification of certain patterns are very important in scientific applications in general, and in geophysical applications in particular. Common needs for certain pattern identification include, finding rising (falling) patterns, finding peaks or valleys, and finding a rise and a fall within an interval. Such functions should be provided by the system.

6. Conclusions and Future Work

Our purpose in this paper is to identify the specific needs of scientific application in the modeling of properties and the unique operations over temporal sequences of measurements. Temporal sequences are very common is scientific applications, but the semantics of these sequences and the specialized operators over them are not well understood, and certainly not supported in commercial systems available today. A notable exception is the Illustra system [4] that has introduced the "time series data blade". However, the emphasis is on business time series applications (such as stock market), and the operations discussed here are only partially covered by the functionality of this data blade.

We have proposed a framework for the specification of time sequence properties and defined operator classes. We then worked out a case study in detail, and demonstrated the usefulness of these properties and operator classes. We have also shown that that there is a need to have an open system approach where new (user-defined) operators can be easily added and integrated into the system.

Our purpose in this paper was to identify the need in this area and to propose a framework. We plan, within this framework, to develop a syntax and a language that can support such functionality.

References

[1] Tansel, A.U. et al, *Temporal Databases*, The Benjamin/Cummings Publishing Company, Redwood City, CA., 1993.

[2] Snodgrass, R. el al, TSQL2 Language Specification, ACM SIGMOD Record, 32(1), 65-86, March 1994.

[3} Segev A., and Shoshani, A., Logical Modeling of Temporal Databases, in Proceedings of ACM SIGMOD International Conference on Management of Data, May 1987.

[4] Illustra user's guide, Illustra Information Technologies, Oakland, California, 1995.