OLAP-Like Analysis of Time Point-Based Sequential Data*

Bartosz Bębel, Mikołaj Morzy, Tadeusz Morzy,
Zbyszko Królikowski, and Robert Wrembel

Poznań University of Technology, Institute of Computing Science, Poznań, Poland
{Bartosz.Bebel,Mikolaj.Morzy,Tadeusz.Morzy,
Zbyszko.Krolikowski,Robert.Wrembel}@put.poznan.pl

Abstract. Nowadays business intelligence technologies allow to analyze mainly set oriented data, without considering order dependencies between data. Few approaches to analyzing data of sequential order have been proposed so far. Nonetheless, for storing and manipulating sequential data the approaches use either the relational data model or its extensions. We argue that in order to be able to fully support the analysis of sequential data, a dedicated new data model is needed. In this paper, we propose a formal model for time point-based sequential data with operations that allow to construct sequences of events, organize them in an OLAP-like manner, and analyze them. To the best of our knowledge, this is the first formal model and query language for this class of data.

1 Introduction

Multiple applications generate huge sets of ordered data. Some typical examples include: workflow systems, user navigation through web pages, diseases curing, RFID-based goods transportation systems (e.g., [10]), public transportation infrastructures (e.g., [2,1,15]), and remote media consumption measurement installations (e.g., [12]). Some of the data have the character of events that last an instant - a chronon, whereas some of them last for a given time period - an interval. In this regard, sequential data can be categorized either as *time point-based* or *interval-based* [16], but for all of them the order in which they were generated is important.

Since over 20 years, data analysis has been performed by means of business intelligence (BI) technologies [5] that include a data warehouse (DW) system architecture and the set of tools for advanced data analysis – the on-line analytical processing (OLAP) applications (e.g., sales trend analysis, trend prediction, data mining, social network analysis). Traditional DW system architectures have been developed in order to efficiently analyze data that originally are coming from heterogeneous and distributed data sources, maintained within an enterprise. OLAP applications, although very advanced ones, allow to analyze mainly set oriented data, but they are not capable of exploiting existing order among data. For this reason, a natural extension to traditional OLAP tools

* This work was supported from the Polish National Science Center (NCN), grant No. 2011/01/B/ST6/05169.

has been proposed in the research literature as the set of techniques and algorithms allowing to analyze data that have sequential nature, e.g., [7,8,10,11,14,15]. For storing and manipulating sequential data, the approaches use either the relational data model or its extension. We argue that in order to be able to fully support the analysis of sequential data, a dedicated new data model is needed.

Paper contribution. In this paper, we extend the draft of a formal model for time point-based sequential data [3] with the definitions of a fact, measure, dimension, and a dimension hierarchy. Thus, the model allows to analyze sequential data in an OLAP-like manner. To the best of our knowledge, this is the first comprehensive model and query language for this class of data.

2 Leading Example

As an illustration of sequential data and their analysis, let us consider patient treatment data, as shown in Table 1. A patient, identified by a SSN, is diagnosed by a doctor. A patient obtains prescriptions, each of which is described by: a unique identifier, date of drawing, patient SSN and his/her age, doctor identifier, medicine name, a dose, a package capacity, and a discount the patient is eligible for.

Table 1. Example data on patient medicine prescription

prescriptionNo.	date	patientSSN	age	doctorID	medicine	dose	package	discount
1198/2011	26.04.2011	74031898333	37	3424	zinnat	0.25 g/5 ml	5 0ml	70%
1199/2011	26.04.2011	98111443657	13	3424	pulmeo	5 ml	150 ml	96.5%
3023/2011	27.04.2011	98111443657	13	9843	pulmicort	0.125 mg/ml	20 ml	70%
3024/2011	27.04.2011	98111443657	13	9843	ventolin	1.5 ml	100 ml	70%
3024/2011	27.04.2011	74031898333	37	5644	augmentin	0,6 g/5 ml	100 ml	70%
3026/2011	27.04.2011	98111443657	13	9843	ventolin	0.1 mg/ml	10 a 2 ml	70%
3027/2011	28.04.2011	34122224334	77	9843	zyrtec	1 mg/ml	75 ml	100%
3031/2011	30.04.2011	56090565958	66	9843	pulmicort	0.125 mg/ml	40 ml	100%

Typical OLAP analyses could include: (1) finding the average number of prescriptions filled by a single doctor monthly during one year period, or (2) finding the total amount of medicines prescribed by doctors working at hospital X.

However, from exploiting the sequential dependencies between data we could mine a valuable knowledge. Examples of such analyses could include: (1) finding the number of patients that were treated with medicine A after they were treated with medicine B, (2) finding the number of patients that within one year were treated at least two times with medicine A, but these treatments were separated with at least one treatment with medicine B. We argue that these and many other analyses require a new data model and query language.

3 Data Model for Sequential Time Point-Based Data

The foundation of our model includes an event and a sequence. A sequence is created from events by clustering and ordering them. Sequences and events have distinguished attributes - measures that can be analyzed in an OLAP-like manner in contexts set up by dimensions.

3.1 Building Elements of the Model

Event. We assume that an elementary data item is an *event*, whose duration is a chronon. Formally, event $e_i \in \mathbb{E}$, where $\mathbb{E}$ is the set of events, is a n-tuple $(a_{i1}, a_{i2}, ..., a_{in})$, where $a_{ij} \in dom(A_j)$. $dom(A_j)$ is the domain of attribute A_j and $dom(A_j) = \mathbb{V}$, where $\mathbb{V}$ is the set of atomic values (character string, date, number) $\cup$ null value. $A_j \in \mathbb{A}$, where $\mathbb{A}$ is the set of event attributes.

Attribute hierarchy. Similarly like in traditional OLAP, event attributes may have some hierarchies associated. Let $\mathbb{L} = \{L_1, L_2, ..., L_k\}$ be the set of levels in the **hierarchies** of the event attributes. Pair $\langle \mathbb{L}_{A_i}, \rhd_{A_i} \rangle$ describes a hierarchy of attribute $A_i \in \mathbb{A}$, where $\mathbb{L}_{A_i} \subseteq \mathbb{L}$ and $\rhd_{A_i}$ is a partial order on set $\mathbb{L}_{A_i}$.

Example 1. In the example from Section 2, an event represents drawing a prescription for a patient. Thus, one event is represented by one row in Table 1, i.e., $\mathbb{E} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$, where:

– e_1=(1198/2011, 26.04.2011, 74031898333, 37, 3424, zinnat, 0.25 g/5ml, 50 ml, 70%),
– e_2=(1199/2011, 26.04.2011, 98111443657, 13, 3424, pulmeo, 5 ml, 150 ml, 96.5%),
– ...
– e_8=(3031/2011, 30.04.2011, 56090565958, 66, 9843, pulmicort, 0.125 mg/ml, 40 ml, 100%).

$\mathbb{A} = \{prescriptionNo, date, patientSSN, age, doctorID, medicine, dose, package, discount\}$. Two attributes have the following hierarchies ($\mathbb{L} = \mathbb{L}_{A_2} \cup \mathbb{L}_{A_3}$):

– $\mathbb{L}_{A_2} = \{date, month, quarter, year\}$ and $\rhd_{A_2}$: *date* $\rightarrow$ *month* $\rightarrow$ *quarter* $\rightarrow$ *year* (with values: *26.04.2011* $\rightarrow$ *April 2011* $\rightarrow$ *Q2 2011* $\rightarrow$ *2011*),
– $\mathbb{L}_{A_3} = \{person, patient\ type\}$ and $\rhd_{A_3}$*person* $\rightarrow$ *patient type* (with values: *98111443657* $\rightarrow$ *child*, *34122224334* $\rightarrow$ *retired*).

Sequence. An ordered list of events that fulfill a given condition is called a *sequence*. The order of events in a sequence is defined by values of selected event attributes. Such attributes will further be called *ordering attributes*. A sequence is composed of the events that have the same value of another selected attribute (or attributes). Such attributes will further be called *forming attributes*. If a forming attribute has a hierarchy associated, then a selected level in the hierarchy can also be used as a forming attribute. Formally, $S_i \in \mathbb{S}$, where $\mathbb{S}$ is the set of sequences, is pair $\langle \mathbb{E}_i, \rhd \rangle$, where $\mathbb{E}_i \subseteq \mathbb{E}$ and $\rhd$ is a partial order on $\mathbb{E}$.

Creating a sequence. For the purpose of creating a sequence from events, we define operator *CreateSequence*: $\mathbb{E} \rightarrow \mathbb{S}$, with the syntax $CreateSequence(\mathbb{E}, \mathbb{F}, \mathbb{A}_o, p)$, where:

– $\mathbb{E}$ is th set of elementary events,
– $\mathbb{F}$ is the set of pairs $\langle A_i, L_j \rangle$, where $A_i \in \mathbb{A}$ is a forming attribute and $L_j \in \mathbb{L}_{A_i}$ is the level of the forming attribute A_i, or $\langle A_i, \phi \rangle$ if attribute A_i does not have a hierarchy,
– $\mathbb{A}_o$ is the set of ordering attributes, $\mathbb{A}_o \subseteq \mathbb{A}$,
– $p \in \mathbb{P}$ is a logical predicate which selects events to form sequences.

Notice that sequences are not defined statically, but their structure is dynamically constructed based on the features of analyses, for which the sequences are created.

Example 2. For the purpose of analyzing patients' treatments, all events describing prescriptions given to the same patient are included into one sequence. They are further ordered by a drawing date.

$CreateSequence(\mathbb{E}, \{\langle A_3, patient\rangle\}, \{A_2\}, null) = \{S_1, S_2, S_3, S_4\}$ where: $S_1 = \langle e_1, e_5\rangle$, $S_2 = \langle e_2, e_3, e_4, e_6\rangle$, $S_3 = \langle e_7\rangle$, $S_4 = \langle e_8\rangle$.

For the purpose of analyzing prescriptions drawn by a doctor during a year, with discounts equal to or greater than 80%, all events describing prescriptions given by the same doctor are included into a sequence. They are further ordered by a drawing date.

$CreateSequence(\mathbb{E}, \{\langle A_5, \phi\rangle \langle A_2, year\rangle\}, \{A_2\}, A_9 \geq 80\%) = \{S_1, S_2\}$ where: $S_1 = \langle e_2\rangle$, $S_2 = \langle e_7, e_8\rangle$.

Fact and measure. Any sequence which is the subject of analysis is the *fact* of analysis. Similarly as in traditional OLAP, sequences are characterized by the values of their *measure* attributes. Measure $m_i \in \mathbb{M}$, where $\mathbb{M}$ is the set of measures. $dom(m_i) = \mathbb{V}$. A measure can be either the attribute of an event or the property of the whole sequence. In order to treat measures uniformly, a measure is defined as function *ComputeMeasure* that associates an atomic value with a sequence, i.e., $ComputeMeasure : \mathbb{S} \times \mathbb{M} \rightarrow \mathbb{V}$. The syntax of the function is as follows: $ComputeMeasure(S_i, name_j, p_j)$, where:

- $S_i \in \mathbb{S}$ is a sequence, for which the values of the measure are computed,
- $name_j$ is the name of the measure,
- $p_j \in \mathbb{P}$ is an expression that computes the values of the measure for a given sequence.

Example 3. Examples of measures being event's attributes include: patient's age, medicine's doze and discount rate. Examples of measures being properties of a sequence include: duration of patient's treatment (a number of days between events: the first and last ones in sequence which describes patient's treatment) or the number of prescriptions drawn by a doctor within a day.

Dimension. A dimension sets up the context of an analysis and defines aggregation paths of facts. Let D_i denote a dimension and $\mathbb{D}$ denote the set of dimensions, thus $D_i \in \mathbb{D}$. A dimension can be either an event attribute or the property of the whole sequence. The *CreateContext* operator associates a dimension with either an event attribute or the whole sequence. It also defines a dimension hierarchy, namely the set of levels and a partial order on this set. The syntax of the operator is as follows: $CreateContext(name_{D_i}, A_{D_i}, p_{D_i}, \mathbb{H}_{D_i}) = D_i$, where:

- $name_{D_i}$ is the name of dimension D_i,
- A_{D_i} equals to $A_j \in \mathbb{A}$ if the dimension is an event attribute A_j or $A_{D_i} = \phi$ if the dimension is the property of the whole sequence,
- p_{D_i} equals to predicate $p \in \mathbb{P}$ if the dimension is the property of the whole sequence (p is an expression that computes the values of the dimension) or $p_{D_i} = \phi$ if the dimension is an event attribute,
- $\mathbb{H}_{D_i}$ is the set of hierarchies of dimension D_i, composed of pairs $\langle \mathbb{L}_{D_i}, \rhd_{D_i}\rangle$, where $\mathbb{L}_{D_i} \subseteq \mathbb{L}$ is the set of levels in the dimension hierarchy and $\rhd_{D_i}$ is a partial order on set $\mathbb{L}_{D_i}$; $\mathbb{H}_{D_i} = \phi$ if dimension D_i does not have a hierarchy.

Example 4. An example of a dimension defined by means of attributes include *patient* with hierarchy: *person* $\rightarrow$ *patient type*; the dimension is set up by an operator

$CreateContext(patient, A_3, \phi, \{\langle\{person, patient\ type\}, person \rightarrow patient\ type\rangle\})$. However, if we would like to analyze the distribution of treatment length, the dimension should be defined as a function which calculates the length of a sequence describing treatment of a single patient. In this case the dimension is defined as follows: $CreateContext(treatment\ length, \phi, for\ all\ S_i \in \mathbb{S}\ find\ Tail(S_i).A_2 - Head(S_i).A_2, \phi)$.

3.2 Operations of the Model

The operations defined in our model are classified into: (1) operations on sequences, (2) general operations, (3) operations on dimensions, and (4) analytical functions. Due to space limitations, we briefly describe the operations in this section.

Operations On Sequences – transform the structure of a single sequence and their result is another sequence. The operations include:

1. $Head(S_i)$ – removes from sequence $S_i \in \mathbb{S}$ all elements except the first one, e.g., $Head(\langle e_2, e_3, e_4, e_6\rangle) = \langle e_2\rangle$.
2. $Tail(S_i)$ – removes from sequence $S_i \in \mathbb{S}$ all elements except the last one, e.g., $Tail(\langle e_2, e_3, e_4, e_6\rangle) = \langle e_6\rangle$.
3. $Subsequence(S_i, m, n)$ – removes from sequence $S_i \in \mathbb{S}$ all elements prior to the element at position m and all elements following element at position n, e.g., $Subsequence(\langle e_2, e_3, e_4, e_6\rangle, 2, 3) = \langle e_3, e_4\rangle$.
4. $Split(S_i, expression)$ – splits sequence $S_i \in \mathbb{S}$ into the set of new sequences based on *expression*; each element of the original sequence belongs to only one of the resulting sequences, e.g., $Split(\langle e_2, e_3, e_4, e_6\rangle$,"the same values of A_5") = $\{\langle e_2\rangle, \langle e_3, e_4, e_6\rangle\}$ (the original sequence describes the whole treatment of a patient regardless of doctors, whereas the resulting set of sequences represents the treatments of the same patient but now each treatment is conducted by one doctor).
5. $Combine(\mathrm{S})$ – creates a new sequence from elements of all sequences in $\mathrm{S} \subseteq \mathbb{S}$ given as parameters; the elements in a new sequence are ordered by the values of ordering attributes of the original sequences, e.g., $Combine(\{\langle e_2\rangle, \langle e_3, e_4, e_6\rangle\}) = \langle e_2, e_3, e_4, e_6\rangle$ (the original sequences describe treatments of the same patient but by two different doctors and the resulting sequence represents the whole treatment of the patient).

General Operations – allow to manipulate sets of sequences.

1. $Pin(\mathrm{S}, expression)$ – filters sequences $\mathrm{S} \subseteq \mathbb{S}$ that fulfill a given expression, e.g., $Pin(\mathrm{S}, length(S_i \in \mathrm{S}) > 3)$ (rejects all sequences that consist of less than four events).
2. $Select(\mathrm{S}, expression)$ – removes from $\mathrm{S} \subseteq \mathbb{S}$ events that do not fulfill a given expression, e.g., $Select(\mathrm{S}, A_6 = zinnat)$ (removes from sequences all events that concern prescriptions other than "zinnat").
3. $GroupBy(\mathrm{S}, expression \mid D_i)$ – assigns sequences from $\mathrm{S} \subseteq \mathbb{S}$ to groups according to the results of a given grouping *expression* (case A) or to dimensions in $D_i \in \mathbb{D}$ (case B). Sequences with the same value of the grouping expression or dimension value belong to one group. The result of the operation is set $\mathbb{G}$ of pairs $\langle value, \mathrm{S}_i\rangle$,

where *value* is a given value of grouping expression and $\mathrm{S}_i \subseteq \mathbb{S}$ is the set of sequences with the same value of a grouping expression (case A), or set of pairs $\langle value(D_i), \mathrm{S}_i \rangle$, where $value(D_i)$ is the value of dimension D_i (case B).

4. Set operations: union $\cup$, difference $\setminus$, and intersection $\cap$ – they are standard set operations that produce a new set of sequences, e.g., $\mathrm{S}_1 \cup \mathrm{S}_2$.

Operations on Dimensions – allow to navigate in the hierarchy of a given dimension. The operations include:

1. $LevelUp(\mathrm{D}, \mathrm{S})$ navigates one level up in the hierarchy of dimensions in D (where $\mathrm{D} \subseteq \mathbb{D}$) for all sequences in $\mathrm{S} \subseteq \mathbb{S}$.
   An example of this operation main include changing the levels of attribute A_2 - from *date* to *month* and A_3 - from *person* to *patient type*: $LevelUp(\{date, patient\}, \{S_1\}) = \{S'_1\}$, where
   - $S_1 = \langle e_1, e_5 \rangle$ (e_1=(1198/2011, 26.04.2011, 74031898333, ..., 50 ml, 70%), e_5=(3024/2011, 27.04.2011, 74031898333, ..., 100 ml, 70%)),
   - $S'_1 = \langle e'_1, e'_5 \rangle$ (e'_1=(1198/2011, April 2011, regular, ..., 50 ml, 70%), e'_5=(3024/2011, April 2011, regular, ..., 100 ml, 70%)).
2. $LevelDown(\mathrm{D}, \mathrm{S})$ navigates one level down in the hierarchy of dimensions in D (where $\mathrm{D} \subseteq \mathbb{D}$) for all sequences in $\mathrm{S} \subseteq \mathbb{S}$.
   An example of this operation may include changing the level of attribute A_3 from *patient type* to *person*: $LevelDown(\{patient\}, \{S'_1\}) = \{S''_1\}$, where
   - $S''_1 = \langle e''_1, e''_5 \rangle$ (e''_1=(1198/2011, April 2011, 74031898333, ..., 50 ml, 70%), e''_5=(3024/2011, April 2011, 74031898333, ..., 100 ml, 70%)).

Analytical Functions – compute aggregates of measures. They include OLAP-like functions *Count*, *Sum*, *Avg*, *Min*, and *Max*. For example, in order to compute the number of sequences formed with attribute A_3 (*patientSSN*) equal to 98111443657 the following expression is used: $Count(Pin(\mathbb{S}, A_3 = 98111443657))$.

Example 5. In order to illustrate the application of our model, let us consider two simple analyses.

Find the number of patients who were treated at least two times with the same medicine in 2000, and in between they were prescribed a different medicine. The implementation of this query using the presented model is as follows:

1. $\mathbb{S} = CreateSequence(\mathbb{E}, \{\langle A_2, year \rangle, \langle A_3, person \rangle\}, \{A_2\},$
   A_2 *between* 1.1.2010 *and* 31.12.2010)
2. $\mathbb{S}' = Pin(\mathbb{S}, e_i.A_6 = e_{i+2}.A_6 \; and \; e_i.A_6 != e_{i+1}.A_6)$
3. $Count(\mathbb{S}')$.

Find distribution of treatment lengths. The implementation of this query is as follows:

1. $\mathbb{S} = CreateSequence(\mathbb{E}, \{\langle A_3, person \rangle\}, \{A_2\}, null)$
2. $D_1 = CreateContext(treatment\ length, \phi, for\ all\ S_i \in \mathbb{S}\ find\ Tail(S_i).A_2 - Head(S_i).A_2, \phi)$
3. $\mathbb{G} = GroupBy(\mathbb{S}, D_1)$
4. $Count(\mathbb{G})$.

4 Related Work

The research and technological areas related to processing sequential data include: (1) complex event processing (CEP) over data streams and (2) OLAP. The CEP technology has been developed for the purpose of continuous analysis of data streams for the purpose of detecting patterns, outliers, and generating alerts, e.g., [4,9]. On the contrary, the OLAP technology [5] has been developed for the purpose of analyzing huge amounts of data organized in relations but it is unable to exploit the sequential nature of data. In this respect, *Stream Cube* [13] and *E-Cube* [14] implement OLAP on data streams. Their main focus is on providing tools for OLAP analysis within a given time window of constantly arriving streams of data.

Another research problem is storage and analysis of data whose inherent feature is an order. These problems have been researched since several years with respect to storage, e.g., [20,17] and query processing, e.g., [19,18]. In [20] sequences are modeled by an enhanced abstract data type, in an object-relational model, whereas in [17] sequences are modeled as sorted relations. The query languages proposed in [19,18] allow typical OLTP selects on sequences but do not support OLAP analyzes. Further extension towards sequence storage and analysis have been made in [6] that proposes a general concept of a RFID warehouse. Unfortunately, no in-dept discussion on RFID data storage and analysis has been provided.

[7,8,15] focus on storage and analysis of time point-based sequential data. [15] propose the set of operators for a query language for the purpose of analyzing patterns. [7,8] focus on an algorithm for supporting ranking pattern-based aggregate queries and on a graphical user interface. The drawback of these approaches is that they are based on relational data model and storage for sequential data.

[10,11] address interval-based sequential data, generated by RFID devices. The authors focus on reducing the size of such data. They propose techniques for constructing RFID cuboids and computing higher level cuboids from lower level ones. For storing RFID data and their sequential orders the authors propose to use three tables, called Info, Stay, and Map.

Our approach differs from the related ones as follows. First, unlike [13,14], we focus on analyzing sequential data stored in a data warehouse. Second, unlike [20,17,18], we concentrate on an OLAP-like analysis of sequential data. Third, unlike [10,11], we focus on analyzing time point-based sequential data. Finally, unlike [7,8,15], we propose a new formal model for sequential data, since in our opinion a relational-like data model is not sufficient for the support of fully functional analysis.

5 Conclusions and Future Work

In this paper we proposed a formal model for representing and analyzing time point-based sequential data, based on the notion of an *event* and a *sequence*, where sequences are dynamically created from events. In order to support OLAP-like analyses of sequences, we associate with events and sequences attributes representing *measures*. Next, we analyze sequences in contexts defined by *dimensions*. Dimensions can have hierarchies, like in traditional OLAP. Sequence analysis is performed by four classes of operations, i.e., operations on sequences, on dimensions, general operations, and analytical functions. To the best of our knowledge, this is the first comprehensive formal model and query language for this class of data. Based on the model, we are currently developing a query processor and results visualizer. Future work will focus on internal mechanisms, like query optimization and data structures.

OLAP-like analysis of interval-based data requires even more advanced data model and operators, e.g., computing aggregates on intervals requires additional semantics of aggregate functions; intervals sort criteria may include begin time, end time, or mid-point time, resulting in different interval sequences; creating and comparing sequences of intervals requires well defined operators. For this reason, we start our research with a simpler problem and in the future we plan to extend our findings towards the interval-based model.

References

1. Octopus card, `http://hong-kong-travel.org/Octopus/` (retrieved March 30, 2012)
2. Smart card alliance, `http://www.smartcardalliance.org` (retrieved March 30, 2012)
3. Bębel, B., Krzyżagórski, P., Kujawa, M., Morzy, M., Morzy, T.: Formal model for sequential olap. In: Information Technology and its Applications. NAKOM (2011) ISBN 978-83-89529-82-4
4. Buchmann, A.P., Koldehofe, B.: Complex event processing. IT - Information Technology 51(5), 241–242 (2009)
5. Chaudhuri, S., Dayal, U., Narasayya, V.: An overview of business intelligence technology. Communications of the ACM 54(8), 88–98 (2011)
6. Chawathe, S.S., Krishnamurthy, V., Ramachandran, S., Sarma, S.: Managing rfid data. In: Proc. of Int. Conf. on Very Large Data Bases (VLDB), pp. 1189–1195 (2004)
7. Chui, C.K., Kao, B., Lo, E., Cheung, D.: S-olap: an olap system for analyzing sequence data. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 1131–1134. ACM (2010)
8. Chui, C.K., Lo, E., Kao, B., Ho, W.-S.: Supporting ranking pattern-based aggregate queries in sequence data cubes. In: Proc. of ACM Conf. on Information and Knowledge Management (CIKM), pp. 997–1006. ACM (2009)
9. Demers, A.J., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., White, W.M.: Cayuga: A general purpose event monitoring system. In: CIDR, pp. 412–422 (2007)
10. Gonzalez, H., Han, J., Li, X.: FlowCube: constructing rfid flowcubes for multi-dimensional analysis of commodity flows. In: Proc. of Int. Conf. on Very Large Data Bases (VLDB), pp. 834–845 (2006)
11. Gonzalez, H., Han, J., Li, X., Klabjan, D.: Warehousing and analyzing massive rfid data sets. In: Proc. of Int. Conf. on Data Engineering (ICDE), p. 83. IEEE (2006)
12. Gorawski, M.: Multiversion Spatio-temporal Telemetric Data Warehouse. In: Grundspenkis, J., Kirikova, M., Manolopoulos, Y., Novickis, L. (eds.) ADBIS 2009. LNCS, vol. 5968, pp. 63–70. Springer, Heidelberg (2010)
13. Han, J., Chen, Y., Dong, G., Pei, J., Wah, B.W., Wang, J., Cai, Y.D.: Stream cube: An architecture for multi-dimensional analysis of data streams. Distributed and Parallel Databases 18(2), 173–197 (2005)
14. Liu, M., Rundensteiner, E., Greenfield, K., Gupta, C., Wang, S., Ari, I., Mehta, A.: E-cube: multi-dimensional event sequence analysis using hierarchical pattern query sharing. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 889–900. ACM (2011)

15. Lo, E., Kao, B., Ho, W.-S., Lee, S.D., Chui, C.K., Cheung, D.W.: Olap on sequence data. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data, pp. 649–660. ACM (2008)
16. Mörchen, F.: Unsupervised pattern mining from symbolic temporal data. SIGKDD Explor. Newsl. 9(1), 41–55 (2007)
17. Ramakrishnan, R., Donjerkovic, D., Ranganathan, A., Beyer, K.S., Krishnaprasad, M.: Srql: Sorted relational query language. In: Proc. of Int. Conf. on Scientific and Statistical Database Management (SSDBM), pp. 84–95. IEEE (1998)
18. Sadri, R., Zaniolo, C., Zarkesh, A., Adibi, J.: Optimization of sequence queries in database systems. In: Proc. of ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pp. 71–81. ACM (2001)
19. Seshadri, P., Livny, M., Ramakrishnan, R.: Sequence query processing. SIGMOD Record 23(2) (1994)
20. Seshadri, P., Livny, M., Ramakrishnan, R.: The design and implementation of a sequence database system. In: Proc. of Int. Conf. on Very Large Data Bases (VLDB), pp. 99–110. Morgan Kaufmann Publishers Inc. (1996)