Discovering Sentinel Rules for Business Intelligence

Morten Middelfart and Torben Bach Pedersen

March 5, 2009

TR-24

A DB Technical Report

Abstract

This paper proposes the concept of *sentinel rules* for multi-dimensional data that warns users when measure data concerning the external environment changes. For instance, a surge in negative blogging about a company could trigger a sentinel rule warning that revenue will decrease within two months, so a new course of action can be taken. Hereby, we expand the window of opportunity for organizations and facilitate successful navigation even though the world behaves chaotically. Since sentinel rules are at the schema level as opposed to the data level, and operate on data *changes* as opposed to absolute data values, we are able to discover strong and useful sentinel rules that would otherwise be hidden when using sequential pattern mining or correlation techniques. We present a method for sentinel rule discovery and an implementation of this method that scales linearly on large data volumes.

1 Introduction

The Computer Aided Leadership and Management (CALM) concept copes with the challenges facing managers that operate in a world of chaos due to the globalization of commerce and connectivity [17]; in this chaotic world, the ability to continuously act is far more crucial for success than the ability to long-term forecast. The idea in CALM is to take the Observation-Orientation-Decision-Action (OODA) loop (originally pioneered by Top Gun fighter pilot John Boyd in the 1950s [15]), and integrate business intelligence (BI) technologies to drastically increase the speed with which a user in an organization cycles through the OODA loop. Using CALM, any organization can be described as a set of OODA loops that are continuously cycled to fulfil one or more Key Performance Indicators (KPI's). One way to improve the speed from observation to action is to expand the "horizon" by allowing the user to see data from the external environment, and not only for the internal performance of the organization. Another way is to give early warnings when factors change that might influence the user's KPI's, e.g., revenue. Placing "sentinels" at the outskirts of the data available seeks to harness both ways of improving reaction time and thus organizational competitiveness.

A sentinel rule is a relationship between two measures, A and B, in an OLAP database where we know, that a change in measure A at one point in time affects measure B within a certain *warning period*, with a certain confidence. If such a relationship exists, we call measure A the *source measure*, and measure B the *target measure*. Usually, the target measure is, or contributes to, a KPI. The source measure ideally represents the external environment, or is as close to the external environment as possible. Examples of source measures for an organization could be: the number of negative blog entries (external), the number of positive articles in papers (external), or the number of complaints from customers (internal, yet as close to external as possible). Examples of target measures could be: revenue or contribution margin. Imagine a company selling a product globally where we discovered the sentinel rule: "IF negative blogs go up THEN revenue goes down within two months AND IF negative blogs go down THEN revenue goes up within two months". Assume that Google is searched daily for negative blogs, and the number of negative blogs is stored in the company's OLAP database. Also, the company's BI solution can notify users based on sentinel rules. When a user receives notification that the number of negative blogs goes up, he will know that revenue will go down in two months with a certain confidence. Depending on the situation, the user might have a number of evasive actions such as: post positive blogs to sway the mood, or reduce cost to cope with a reduction in revenue. Regardless of the action, the sentinel rule has raised awareness of a problem and reduced the time from observation to action. Metaphorically, sentinels seek to do for organizations what radars do for navigation of ships.

The novel contributions in this paper include the sentinel rule concept, and an algorithm that discover sentinel rules on multi-dimensional data. We give a formal definition of sentinel rules, and we define the indication concept for rules and for source and target measures. In this context, we provide a contradiction elimination process that allows us to generate more general rules that are easy to interpret. We also provide a useful notation for sentinel rules. We conduct several experiments to validate that our algorithm scales linearly on large volumes of synthetic and real-world data and on databases with high complexity in terms of the number of measures. Since sentinel rules operate on data changes as opposed to absolute data values, and are at the *schema level* as opposed to the *data level* (such as association rules and sequential patterns), we can find strong rules that neither association rules nor sequential pattern mining would find. In addition, we found sentinel rules to be complementary to correlation techniques, since our solution finds "micro-predictions" that hold for a smaller subset within a dataset; using correlation techniques alone such rules would be "hidden in the average". We believe that we are the first to propose the concept of sentinel rules, and to provide an algorithm and implementation for discovering them.

The next section presents the formal definition, Section 3 presents an algorithm including an assessment of its complexity. Section 4 presents a scalability and a qualitative study. Section 5 presents the work related to the discovery of sentinel rules.

2 Problem Definition

Running Example: Imagine a company that sells products world-wide, and that we, in addition to the traditional financial figures such as revenue, *Rev*, have been monitoring the environment outside our organization and collected that information in three measures. The measure *NBlgs* represents the number of times an entry is written on a blog where a user is venting a negative opinion about our company or products. The measure *CstPrb* represents the number of times a customer contacts our company with a problem related to our products. The measure *WHts* represents the number of hits on our website, and this figure has been cleansed in order to represent human contact exclusively, eliminating traffic by robots etc.

Table 1. Example dataset

T: Time	D_2: Region	M_1: NBlgs	M_2: CstPrb	M_3: WHts	M_4: Rev
2007-Q1	Asia	20	50	1.000	10.000
2007-Q2	Asia	21	45	1.500	9.000
2007-Q3	Asia	17	33	2.000	11.000
2007-Q4	Asia	15	34	2.500	13.000
2007-Q1	EU	30	41	3.000	20.000
2007-Q2	EU	25	36	3.500	25.000
2007-Q3	EU	22	46	4.000	28.000
2007-Q4	EU	19	37	4.500	35.000
2007-Q1	USA	29	60	5.000	50.000
2007-Q2	USA	35	70	5.500	55.000
2007-Q3	USA	40	72	6.500	45.000
2007-Q4	USA	39	73	7.500	40.000

In Table 1 we see a subset from our database, representing each quarter in year 2007 across three geographical regions. It should be noted that a subset like Table 1 can easily be extracted from a multi-dimensional database, i.e., if the desired data are the base level of the database no processing is needed, if the desired levels are higher than the base level, the data might or might not be preaggregated. However, both extraction and aggregation are typically basic built in functions of any multi-dimensional database. The three measures: NBlgs, CstPrb and WHts, representing the external environment around our company, have been presented along with the internal measure, Rev, representing our Revenue. The variable names: $T, D_2, M_1...M_4$ have been assigned to the dimensions and measures in order to create transparency to the formal definition in Section 2.

We are interested in discovering whether we can use any of the external measures to predict a future impact on the internal Revenue measure; in other words we are looking for sentinel rules where one of the measures $M_1...M_3$ can give us an early warning about changes to M_4. To distinguish between which measures are "causing" the other, we call the measures $M_1...M_3$ *source measures* and the measure M_4 is called the *target measure*.

Formal Definition: Let C be a multi-dimensional data cube containing a set of dimensions: $D = \{D_1, D_2...D_n\}$ and a set of measures: $M = \{M_1, M_2...M_p\}$. We denote the members of the dimensions in D by $d_1, d_2...d_n$ and we denote the corresponding *measure values* for any combination of dimension members by $m_1, m_2...m_p$. A measure value is a function, M_i, that returns the value of a given measure corresponding to the dimension members it is presented with. We will now provide a series of definitions that define a source measure, A, is a *sentinel* for a target measure, B, i.e., a guarded watchtower from which we monitor A in order to know about changes ahead of time to B. The sentinel rule between A and B is denoted $A \rightsquigarrow B$. We assume, without loss of generality, that there is only one time dimension, T, in C, and that $T = D_1$, and subsequently $t = d_1$. A fact, f, in C is then defined as:

$$f = (t, d_2, d_3...d_n, m_1, m_2...m_p) \tag{1}$$

Given a fact f, the measure M_i is a function $M_i(t, d_2, d_3...d_n) = m_i$. The "dimension" part of f, $(t, d_2, d_3...d_n)$, is called a cell. The *shifting* of a fact f, f', is a fact with the same non-time dimension values $(d_2...d_n)$ as f, but for time period $t+o$, if it exists in C, i.e., a period of o members later on the time dimension. We denote the *offset*, o, and define the function as:

$$\mathit{Shift}(C, f, o) = f' = (t + o, d_2, d_3...d_n, m'_1, m'_2...m'_p) \text{ if } f' \in C \tag{2}$$

Since we are interested in the change in data, we introduce the *measure difference function*, *Diff*. With *Diff*, we find the relative changes to each of the measures during the time period specified by the offset. *Diff* is defined as follows:

$$\begin{aligned} \mathit{Diff}(C, f, o) = \ & (t, d_2, d_3...d_n, \frac{m'_1 - m_1}{m_1}, \frac{m'_2 - m_2}{m_2} ... \frac{m'_p - m_p}{m_p}) \\ & \text{where } f = (t, d_2, d_3...d_n, m_1, m_2...m_p) \ \wedge \ f \in C \ \wedge \\ & f' = \mathit{Shift}(C, f, o) = (t + o, d_2, d_3...d_n, m'_1, m'_2...m'_p) \wedge f' \in C \end{aligned} \tag{3}$$

Given a threshold, α, we say that $x \in \mathit{Diff}(C, f, o)$ is an *indication* on a measure, M_i, if:

$$x = (t, d_2, d_3...d_n, \frac{m'_1 - m_1}{m_1}, ..., \frac{m'_i - m_i}{m_i}, ..., \frac{m'_p - m_p}{m_p}) \wedge |\frac{m'_i - m_i}{m_i}| \geqq \alpha \tag{4}$$

We say that an indication on M_i, x, is *positive*, denoted $M_i\blacktriangle$, when $\frac{m'_i - m_i}{m_i} > 0$ and consequently that an indication, x, is *negative*, denoted $M_i\blacktriangledown$, when $\frac{m'_i - m_i}{m_i} < 0$. We define a wildcard, $*$, meaning that M_i* can be either $M_i\blacktriangle$ or $M_i\blacktriangledown$.

Table 2. Indications between quarters.

T: Time	D_2: Region	M_1: NBlgs	M_2: CstPrb	M_3: WHts	M_4: Rev
'07:Q1→Q2	Asia	neutral	$M_2\blacktriangledown$	$M_3\blacktriangle$	$M_4\blacktriangledown$
'07:Q2→Q3	Asia	$M_1\blacktriangledown$	$M_2\blacktriangledown$	$M_3\blacktriangle$	$M_4\blacktriangle$
'07:Q3→Q4	Asia	$M_1\blacktriangledown$	neutral	$M_3\blacktriangle$	$M_4\blacktriangle$
'07:Q1→Q2	EU	$M_1\blacktriangledown$	$M_2\blacktriangledown$	$M_3\blacktriangle$	$M_4\blacktriangle$
'07:Q2→Q3	EU	$M_1\blacktriangledown$	$M_2\blacktriangle$	$M_3\blacktriangle$	$M_4\blacktriangle$
'07:Q3→Q4	EU	$M_1\blacktriangledown$	$M_2\blacktriangledown$	$M_3\blacktriangle$	$M_4\blacktriangle$
'07:Q1→Q2	USA	$M_1\blacktriangle$	$M_2\blacktriangle$	$M_3\blacktriangle$	$M_4\blacktriangle$
'07:Q2→Q3	USA	$M_1\blacktriangle$	neutral	$M_3\blacktriangle$	$M_4\blacktriangledown$
'07:Q3→Q4	USA	neutral	neutral	$M_3\blacktriangle$	$M_4\blacktriangledown$

In our running example, when assessing whether a relationship exists, we are not concerned with minor fluctuations, so we define a threshold of 10%, meaning that a measure has to change at least 10% up or down in order to be of interest. Furthermore, given the dataset we have, we are interested in seeing the changes that occur over *quarters* as presented in Table 1. This means that we set the threshold $\alpha = 10\%$

and then the offset o = *1 Quarter*. In Table 2, we have calculated the changes from each quarter to the next and subjected each change to an evaluation against the threshold of 10% change. We denote positive indications by ▲ and subsequently negative by ▼, if a change is less than 10% in either direction it is deemed "neutral". Please note that since we are dealing with *changes* between periods, we naturally get one less row for each region.

$$ST(C,o,w) = \{(\mathit{Diff}(C,f,o), \mathit{Diff}(C, \mathit{Shift}(C,f,w),o)) | f \in C\} \tag{5}$$

A Source-Target Set, ST, is defined as paired indications of changes over time, where the source and target measures have been shifted with the offset, o. The target measures have additionally been shifted with a *warning period*, w, which is the timeframe after which we should expect a change on a target measure, after an indication on a source measure has occurred. We say that $(x,x') \in ST(C,o,w)$ *supports* the *indication rule* $A\blacktriangle \rightarrow B\blacktriangle$ if x is an indication of $A\blacktriangle$ and x' is an indication of $B\blacktriangle$. In this case, we also say that x supports $A\blacktriangle$ and x' supports $B\blacktriangle$. The *support* of an indication rule is the number of $(x,x') \in ST(C,o,w)$ which supports the rule. The support of indication rules $A\blacktriangledown \rightarrow B\blacktriangledown$, $A\blacktriangle \rightarrow B\blacktriangledown$ and $A\blacktriangledown \rightarrow B\blacktriangle$ as well as the support for indications $A\blacktriangledown$ and $B\blacktriangledown$ are defined similarly. We denote the support of an indication and an indication rule by *IndSupp* followed by the name of the indication or indication rule, respectively, e.g., $\mathit{IndSupp}_{A\blacktriangle}$ and $\mathit{IndSupp}_{A\blacktriangle \rightarrow B\blacktriangle}$.

A sentinel rule is an *unambiguous* relationship between A and B, thus we must first eliminate contradicting indication rules, if such exist, before we have a sentinel rule. We refer to this process as the *contradiction elimination process*, and we use it to remove indication rules with the same premise, but a different consequent, and vice versa, e.g., if both $A\blacktriangle \rightarrow B\blacktriangle$ and $A\blacktriangle \rightarrow B\blacktriangledown$ or if both $A\blacktriangle \rightarrow B\blacktriangle$ and $A\blacktriangledown \rightarrow B\blacktriangle$ are supported. To eliminate such contradictions, we pair the indication rules in two sets that do not contradict each other, and we denote these sets by $A \rightarrow B$ and $A \rightarrow \mathit{inv}(B)$, as follows: $A \rightarrow B = \{A\blacktriangle \rightarrow B\blacktriangle, A\blacktriangledown \rightarrow B\blacktriangledown\}$ and $A \rightarrow \mathit{inv}(B) = \{A\blacktriangle \rightarrow B\blacktriangledown, A\blacktriangledown \rightarrow B\blacktriangle\}$. Here *inv* indicates an inverted relationship between the indications on A and B, e.g. if $A\blacktriangle$ then $B\blacktriangledown$, and vice versa.

For the purpose of being able to deduct the support of the indication rule(s) we eliminate, we define functions for returning the premise and the consequent indication, *IndPrem* and *IndCons*, from an indication rule $A\blacktriangle \rightarrow B\blacktriangle$ as follows: $\mathit{IndPrem}(A\blacktriangle \rightarrow B\blacktriangle) = A\blacktriangle$ and $\mathit{IndCons}(A\blacktriangle \rightarrow B\blacktriangle) = B\blacktriangle$. Furthermore, we define the complement of an indication as follows: $\overline{A\blacktriangle} = A\blacktriangledown$ and $\overline{A\blacktriangledown} = A\blacktriangle$. We can now define a contradicting indication rule as a function, *ContraRule*, for an indication rule, *IndRule*, as follows:

$$\mathit{ContraRule}(\mathit{IndRule}) = \mathit{IndPrem}(\mathit{IndRule}) \rightarrow \overline{\mathit{IndCons}(\mathit{IndRule})} \tag{6}$$

$$\mathit{ElimSupp}(\mathit{IndRule}) = \mathit{IndSupp}_{\mathit{IndRule}} - \mathit{IndSupp}_{\mathit{ContraRule}(\mathit{IndRule})} \tag{7}$$

The support after elimination, *ElimSupp*, of an indication rule, *IndRule*, where the support of the contradicting indication rule, *ContraRule(IndRule)*, has been eliminated can be calculated as shown in Formula 7.

$$\mathit{MaxRule} = \begin{cases} \{\mathit{IndRule}_i \mid \mathit{IndRule}_i \in A \rightarrow B \ \wedge \ \mathit{ElimSupp}(\mathit{IndRule}_i) > 0\} \\ \qquad\qquad \text{if } \mathit{IndSupp}_{A \rightarrow B} >= \mathit{IndSupp}_{A \rightarrow \mathit{inv}(B)}, \\ \\ \{\mathit{IndRule}_i \mid \mathit{IndRule}_i \in A \rightarrow \mathit{inv}(B) \ \wedge \ \mathit{ElimSupp}(\mathit{IndRule}_i) > 0\} \\ \qquad\qquad \text{if } \mathit{IndSupp}_{A \rightarrow B} < \mathit{IndSupp}_{A \rightarrow \mathit{inv}(B)}. \end{cases} \tag{8}$$

MaxRule is the set of indication rule(s), $\mathit{IndRule}_i$, in the set ($A \rightarrow B$ or $A \rightarrow \mathit{inv}(B)$) with the highest *IndSupp* and where $\mathit{ElimSupp}(\mathit{IndRule}_i) > 0$. With *MaxRule*, we have identified the best indication rule(s) for a sentinel rule that represents an unambiguous relationship between A and B, i.e., the non-contradicting indication rules with the highest *ElimSupp*. In other words, we have eliminated the *contradicting* indication rules where the premise contradicts the consequent, as well as the *orthogonal* indication rules where different premises have the same consequent. If the *MaxRule* set consists of only one indication rule, we refer to the sentinel rule based on this as a *uni-directional* rule.

We denote the support of a sentinel rule by *SentSupp*, followed by the name of the sentinel rule, e.g., $SentSupp_{A \rightsquigarrow B}$. For a potential sentinel rule, $A \rightsquigarrow B$, we define *SentSupp* as the sum of the support of source measure indications for the indication rule(s) contained in the sentinel rule:

$$SentSupp_{A \rightsquigarrow B} = \begin{cases} IndSupp_{A\blacktriangle} & \text{if } A\blacktriangledown \rightarrow B* \notin MaxRule, \\ IndSupp_{A\blacktriangledown} & \text{if } A\blacktriangle \rightarrow B* \notin MaxRule, \\ IndSupp_{A\blacktriangle} + IndSupp_{A\blacktriangledown} & \text{otherwise.} \end{cases} \tag{9}$$

In Formula (9) we note the difference between the support of an indication rule, *IndSupp*, and a sentinel rule, *SentSupp*. Specifically, when calculating the support of a sentinel rule, $SentSupp_{A \rightsquigarrow B}$, we only consider the support of indications on the source measure (the premise), $A\blacktriangle$ and $A\blacktriangledown$. With indication rules, both indications on the source and target measure needs to occur. The reason is, that the consequential support of indications on the target measure, $B\blacktriangle$ or $B\blacktriangledown$, is taken into consideration when calculating the confidence of the sentinel rule in Formula (10). In the case of a uni-directional rule (the two first cases) we only consider the support of indications on the source measure that have the same direction as the one indication rule in *MaxRule*; this is done in order not to penalize otherwise good uni-directional rules in terms of confidence. We denote confidence by *Conf*, and define the confidence for a sentinel rule, $A \rightsquigarrow B$, as follows:

$$Conf_{A \rightsquigarrow B} = \frac{\sum_{IndRule_i \in MaxRule} ElimSupp(IndRule_i)}{SentSupp_{A \rightsquigarrow B}} \tag{10}$$

The minimum threshold for *SentSupp* is denoted β, and the minimum threshold for *Conf* is denoted γ. With these definitions, we say that a sentinel rule, $A \rightsquigarrow B$, with an offset, o, and a warning period, w, exists in C when $SentSupp_{A \rightsquigarrow B} \geqq \beta$ and $Conf_{A \rightsquigarrow B} \geqq \gamma$.

Sentinel rule notation: To express sentinel rules with easy readability, we use $\rightsquigarrow$ to show that there is a sentinel rule between a source measure, A, and a target measure, B. In the case, where a bi-directional rule represents an inverted relationship between the source and the target measure, we add *inv* to the target measure. In the case where the rule is uni-directional, we add ▲ or ▼ to both the source and the target measure to express the direction of the sentinel rule.

Table 3. Target and source measure comparison

T: Time	D_2: Region	M_1: NBlgs	M_2: CstPrb	M_3: WHts	M_4': Rev
'07:Q1→Q2	Asia	neutral	M_2▼	M_3▲	M_4'▲
'07:Q2→Q3	Asia	M_1▼	M_2▼	M_3▲	M_4'▲
'07:Q1→Q2	EU	M_1▼	M_2▼	M_3▲	M_4'▲
'07:Q2→Q3	EU	M_1▼	M_2▲	M_3▲	M_4'▲
'07:Q1→Q2	USA	M_1▲	M_2▲	M_3▲	M_4'▼
'07:Q2→Q3	USA	M_1▲	neutral	M_3▲	M_4'▼

In our running example, we limit ourselves to investigating whether sentinel rules exist between any of the source measures $M_1...M_3$ and the target measure M_4. We now need to compare the changes in $M_1...M_3$ to changes in M_4 at a later time. In this case, we choose the timeframe of 1 quarter again, meaning that warning period w = *1 Quarter*.

In Table 3, we show the comparison between the source measure indications and the target measure indication one quarter later. The measure M_4 is basically moved one line up -or as shown in Table 3; one quarter back. This means that all source measures for Asia changing 2007: Q2→Q3 as shown in the left column are now compared on the same line, within the same row, to the change on the target measure, M_4, for Asia changing 2007: Q3→Q4 and so on. The shift of M_4 shown in the row with data for the period one quarter earlier is denoted M'_4. Please note that since we are looking at changes between the periods selected on the time dimension, as noted earlier, we naturally get one less row for each geographical region, when we make the comparison across 1 quarter.

Based on Table 3, we count the support for each combination of indication changes, the indication rules, for each potential sentinel rule; in addition, we can count the support of the relationship overall, basically the support means counting all rows that do not have a "neutral" change on the source measure since we define indications as being either positive or negative. For example, we see summarized in Table 4, that the indication rule $M_1\blacktriangledown \rightarrow M'_4\blacktriangle$ is supported 3 times in the dataset shown in Table 3; we say that the indication rule $M_1\blacktriangledown \rightarrow M'_4\blacktriangle$ has a support of 3, and the sentinel rule $M_1 \rightsquigarrow M_4$ has a support of all indication rule combinations which in this case is 5. Table 4 through 6 lists the indication rules for each potential sentinel rule with their respective support (Formula 9).

As mentioned earlier, the ideal sentinel rule describes changes bi-directionally so that it can "predict" both positive and negative changes on the target measure. However, the relationship also needs to be non-contradictory in order to be useful as a sentinel rule. To do this, we eliminate the indications that contradict each other as described in Formulae 6 and 7. In Table 5 we find the a uni-directional rule where the two contradicting indication rules have equal support, thus we disregard these indications completely (Formula 9) and therefore $SentSupp_{M_2 \rightsquigarrow M_4}$=3. In Table 6 the contradiction elimination process does not eliminate both indication rules, it reduces the two indication rules to one and decreases *ElimSupp* (Formula 7) in the calculation of confidence.

In order to identify the best sentinel rules, we set the thresholds $\beta = 3$ and $\gamma = 60\%$. Table 7 through 9 show the sentinel rules from our running example and their respective conformance to the thresholds we have set. As seen in Table 8 and 9, we end up having uni-directional sentinel rules, since the indication rules $M_2\blacktriangle \rightarrow M'_4\blacktriangle$ and $M_2\blacktriangle \rightarrow M'_4\blacktriangledown$, as shown in Table 5, contradict each other and have equal support. In addition, the indication rules $M_3\blacktriangle \rightarrow M'_4\blacktriangle$ and $M_3\blacktriangle \rightarrow M'_4\blacktriangledown$ contradict each other in Table 6. Of these, $M_3\blacktriangle \rightarrow M'_4\blacktriangle$ is strongest and "wins" the elimination process (Formula 8) as seen in Table 9.

Based on this example, we have now found that there are two sentinel rules that can provide our company with an early warning. If we monitor the changes to M_1, the number of negative blog entries, we will know one quarter in advance whether to expect an increase or a decrease in

Indication rule and sentinel rule support

Table 4. $M_1 \rightsquigarrow M_4$

M_1	M'_4	*IndSupp*
$M_1\blacktriangle$	$M'_4\blacktriangle$	0
$M_1\blacktriangledown$	$M'_4\blacktriangledown$	0
$M_1\blacktriangle$	$M'_4\blacktriangledown$	2
$M_1\blacktriangledown$	$M'_4\blacktriangle$	3
$SentSupp_{M_1 \rightsquigarrow M_4} = 5$		

Table 5. $M_2 \rightsquigarrow M_4$

M_2	M'_4	*IndSupp*
$M_2\blacktriangle$	$M'_4\blacktriangle$	1
$M_2\blacktriangledown$	$M'_4\blacktriangledown$	0
$M_2\blacktriangle$	$M'_4\blacktriangledown$	1
$M_2\blacktriangledown$	$M'_4\blacktriangle$	3
$SentSupp_{M_2 \rightsquigarrow M_4} = 3$		

Table 6. $M_3 \rightsquigarrow M_4$

M_3	M'_4	*IndSupp*
$M_3\blacktriangle$	$M'_4\blacktriangle$	4
$M_3\blacktriangledown$	$M'_4\blacktriangledown$	0
$M_3\blacktriangle$	$M'_4\blacktriangledown$	2
$M_3\blacktriangledown$	$M'_4\blacktriangle$	0
$SentSupp_{M_3 \rightsquigarrow M_4} = 6$		

Table 7. $M_1 \rightsquigarrow M_4$

M_1	M'_4	*ElimSupp*
$M_1\blacktriangle$	$M'_4\blacktriangledown$	2
$M_1\blacktriangledown$	$M'_4\blacktriangle$	3
$SentSupp_{M_1 \rightsquigarrow M_4} = 5$		
$Conf_{M_1 \rightsquigarrow M_4} = \frac{5}{5} = 100\%$		
Conformance: ok		

Table 8. $M_2 \rightsquigarrow M_4$

M_2	M'_4	*ElimSupp*
$M_2\blacktriangledown$	$M'_4\blacktriangle$	3
$SentSupp_{M_2 \rightsquigarrow M_4} = 3$		
$Conf_{M_2 \rightsquigarrow M_4} = \frac{3}{5} = 100\%$		
Conformance: ok		

Table 9. $M_3 \rightsquigarrow M_4$

M_3	M'_4	*ElimSupp*
$M_3\blacktriangle$	$M'_4\blacktriangle$	2
$SentSupp_{M_3 \rightsquigarrow M_4} = 6$		
$Conf_{M_3 \rightsquigarrow M_4} = \frac{2}{6} = 33\%$		
Conformance: failed		

M_4 Revenue. If we monitor the number of times a customer contacts our company with a problem related to our products, M_2, we will know one quarter ahead whether to expect an increase in Revenue. This example demonstrates the usefulness of the sentinel concept, and the idea is that we attempt to place our sentinels as close to the external environment as possible and with as high reliability as possible. Using the notation defined earlier in this section, we can express the rules found in our running example as follows: $\textit{NBlgs} \rightsquigarrow \textit{inv}(\textit{Rev})$ and $\textit{CstPrb}\blacktriangledown \rightsquigarrow \textit{Rev}\blacktriangle$

3 The FindSentinels Algorithm

The following algorithm has been implemented in SQL on a Microsoft SQL Server 2005 as explained in Section 4. The actual SQL code can be found in Appendix A. We assume without loss of generality that of the p measures in the dataset, C, $M_1 ... M_{p-1}$ are the source measures and M_p is the target measure.

Step 1 creates a temporary table where each unique value of (time dimension value) *t*, is sorted in ascending order and assigned an integer, *Id*, growing by 1 for each *t*. This temporary table will allow us to select values of t for comparison with a given distance in periods, regardless of the format of the period field, *t*, in the database. To optimize performance, we create an index on the period table. By joining 4 copies of each of the original dataset and the period table (one for each of the periods: $t, t+o, t+w$, and $t+w+o$), we create a Source-Target set (Formula (5)) and calculate indications (Formulae (3) and (4)) for our selected *p-1* source measures and one target measure. We calculate these indications for each cell (dimension combination) in the dataset, and return -1, 0, or 1 depending on whether the indication is negative, neutral or positive against the threshold α.

Step 2 counts the number of positive and negative indications on the source measure, and for each of these source measure indications, it summarizes the indications on the target measure. Since the indications are expressed as -1, 0 or 1, our contradiction elimination process can be carried out using sum.

Step 3 retrieves the potential rules from previous output, meaning that a source measure needs to have at least one indication with a consequential indication on the target measure, i.e., *ElimSupp*<> 0. For each of these rules, we calculate the sum of the support of source measure indications, *SentSupp*, the sum of absolute indications on the target measure, *AbsElimSupp*, as well as *MaxElimSupp* which is max(*ElimSupp*). In addition, we calculate the *Direction* of the relationship between source and target measure where 1 is straightforward and -1 is inverted. The nature of *Direction* also helps us eliminate orthogonal rules since these will always have *Direction=0.* This is true because an orthogonal relationship means that both positive and negative indications on the source measure leads to only one type of indication on the target measure. Finally, we calculate the number of indication rules, *IndRuleCount*, in the potential sentinel rule. This information is used to distinguish between bi- and uni-directional rules. Using this information, we can now identify the sentinel rules that comply with the the criteria of *SentSupp* $>= \beta$ and *Conf* $>= \gamma$. In addition, we can use the values of *IndRuleCount*, *Direction*, and *MaxElimSupp* to describe the sentinel rule in accordance with our notation. We store the output in a table called *FinalResult.*

Algorithm FindSentinels
Input: A dataset, C, an offset, o, a warning period, w, a threshold for indications, α, a minimum *SentSupp* threshold, β, and a minimum *Conf* threshold, γ.
Output: Sentinel rules with their respective *SentSupp* and *Conf*.
Method: Sentinel rules are discovered as follows:

```
1. Scan the dataset $C$ once and retrieve unique values of $t$ into an indexed subset. Use the subset to reference
   each cell $(t, d_2, \ldots, d_n) \in C$ with the corresponding cells for $\{t+o, t+w, t+w+o\} \in C$. Output a Source-
   Target set (Formula (5)) for each cell, $(t, d_2, \ldots, d_n)$, where the indications (Formulae (3) and (4)) on source
   measures, $M_1 \ldots M_{p-1}$, are calculated using $\{t, t+o\}$ and the indications on target measure, $M_p$, is calculated
   using $\{t+w, t+w+o\}$.
2. For each positive and negative source measure indication, $M_i Ind$, in the output from Step 1, count the number
   of source measure indications as $IndSupp_i$ and sum the target measure indications as $ElimSupp_i$.
3. Retrieve from the output from Step 2, each source measure, $M_i \in M_1 \ldots M_{p-1}$, where $ElimSupp <> 0$.
   For each of these source measures, calculate: $SentSupp$=sum($IndSupp$),
   $AbsElimSupp$=sum|$ElimSupp$|, $MaxElimSupp$=max($ElimSupp$),
   $Direction$=avg(sign($M_i Ind$)*sign($ElimSupp$)), and $IndRuleCount$ as the number of different indications (posi-
   tive, negative). Output the rules where $SentSupp >= \beta$ and $Conf = \frac{AbsElimSupp}{SentSupp} >= \gamma$, use $IndRuleCount$=2 to
   identify bi-directional rules and $Direction$ to describe whether the relationship is straight-forward or inverted.
   For uni-directional rules ($IndRuleCount$= 1) use the four combinations of $Direction$ and sign($MaxElimSupp$)
   to describe the relationship.
```

Upon execution of the algorithm, FindSentinels, with the dataset from our running example as C, we get the output table named *FinalResult* as seen in Table 10. We note that the result is similar to that of Tables 7 & 8, and we can easily recognize the sentinel rules: $NBlgs \rightsquigarrow inv(Rev)$ and $CstPrb\blacktriangledown \rightsquigarrow Rev\blacktriangle$

Table 10. The *FinalResult* table

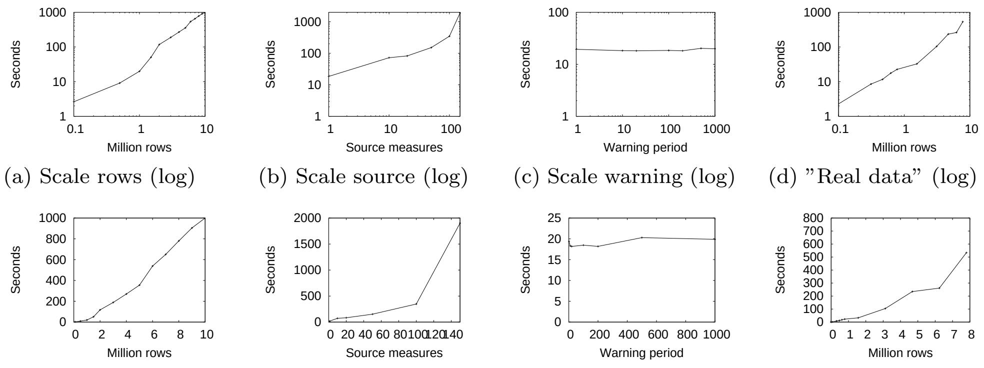
SentinelRule	SentSupp	Conf
NBlgs->inv(Rev)	5	100
CstPrb_dec->Rev_inc	3	100

Computational Complexity: When we examine the individual statements the algorithm, FindSentinels, we notice that the size of output for each step is at most as large as the input, thus the computational complexity will be dominated by the size of the input. The size of the input for Step 3 is much smaller than for previous statements and can thus be disregarded. The retrieve of unique t in Step 1 can be performed in $\mathcal{O}(n)$ using a hash-based algorithm [9], where n is the size of the dataset, C. The indexing can be done in time $\mathcal{O}(p\ log\ p)$ where p is the number of periods, and since $p << n$ this cost can be disregarded. The major cost in Step 1 is the 8-way join, and since $p << n$ this cost will be dominated by the cost of the join of 4 copies of C, and even this can be performed in time $\mathcal{O}(n)$ using hash-joins [9]. Since the number of source measures are a small constant, Step 2 can also be done in $\mathcal{O}(n)$ using hash-aggregation [9]. In summary, the whole algorithm can be done in time $\mathcal{O}(n)$, where n is the size of C, and the algorithm thus scales linearly.

4 Experiments

Setup: The experimental evaluation was conducted on an Intel Core2 Quad CPU (Q6600) 2.40GHz PC server with 4GB RAM and 1 500GB disk (7,200 RPM) running a 32Bit version of Microsoft SQL Server 2005 (MS SQL) on Windows Vista Business, Service Pack 1. The recovery model on MS SQL was set to "simple". In addition, MS SQL is restarted and the database and logfile space are shrunk before each run.

We use two ranges of datasets for the experiments, a range of synthetic datasets and a range of real-world datasets. The synthetic datasets closely resemble our running example, i.e., there are three regions and one product that are iterated over a variable number of periods and variable number of source measures to produce a dataset of the desired size in rows and source measures. The synthetic datasets produce the same sentinel rules as output as our running example when



(a) Scale rows (log) (b) Scale source (log) (c) Scale warning (log) (d) "Real data" (log)
(e) Scale rows (f) Scale source (g) Scale warning (h) "Real data"

Figure 1: Performance results

subjected to our SQL implementation. The synthetic datasets with three source measures and one target measure, similar to our running example, ranges from 100,000 to 10,000,000 rows with 1,000,000 row intervals from 2,000,000 to 10,000,000 rows, and includes datasets with 500,000 and 1,500,000 rows in order to monitor the behavior of the SQL implementation more closely at low data volumes. In addition, we generate datasets with 1,000,000 rows and with 1, 10, 20, 50, 100 and 150 source measures and one target measure.

The real-world datasets are produced based on a dataset from a medium-sized Danish company with one location and one product, the original dataset contains 78 months (6.5 years) of monthly aggregated measures of *Website hits*, *Support cases*, *New customers* and *Revenue*. Descendants of this dataset are produced by adding the original dataset to itself the number of times needed to get the desired number of rows, but each time the dataset is added, it is done with a new product number. By doing this, we end up having a significant amount of data with real-world patterns. Using this approach, all real-world datasets have a number of rows in multiples of 78, namely: 78; 780; 7,800; 78,000; 312,000; 468,000; 624,000; 780,000; 1,560,000; 3,120,000; 4,680,000; 6,240,000 and 7,800,000 rows. The results presented represent the average time for 5 runs.

Synthetic data - scaling number of rows: We run the SQL implementation against all the synthetic datasets with three source and one target measure. We have plotted the results in both logarithmic scale, Figure 1(a), and linear scale, Figure 1(e), in order to get a better impression of the scaling at both the low end and the high end. We notice that the SQL implementation scales linearly as expected based on our assessment of the $\mathcal{O}(n)$ computational complexity. We attribute the jump in process time that occurs between the experiments on 5,000,000 and 6,000,000 rows, to the DBMS' inability to handle everything in memory thus requiring more disk I/O. A similar factor is seen between the experiments on 1,500,000 and 2,000,000 rows.
Synthetic data - scaling source measures: The SQL implementation is run against the synthetic datasets with a varying number of source measures and 1,000,000 rows. The results are plotted in Figures 1(b) and 1(f). We observe a linear behavior in computation time until we reach more than 100 source measures where process time increases drastically. However, 100 measures is a very high number in a normal OLAP cube, since real-world implementations in our experience typically have a sum of source and target measures less than 50.
Synthetic data - scaling warning period: In this experiment, we vary the warning period, w, with the values: 1, 10, 20, 100, 200, 500, 1000 on a dataset with 1,000,000 rows. Results are plotted in Figures 1(c) and 1(g). As expected, the SQL implementation is almost unaffected by a variation in warning period, the process time remains close to constant because the intermediate results of the SQL solution's steps are about the same size.

Real-world data - scaling number of rows: Changing now to real-world datasets, we run the SQL implementation on the real-world datasets with varying number of rows. The results are plotted in Figures 1(d) and 1(h). In general, we observe a behavior close to linear, but the jump in process time due to a change from memory to disk, appear different from what we observed in the experiments on synthetic data. We attribute the variance to the fact that the joins behave slightly different since the period field has lower cardinality and the product field has higher cardinality in the real-world datasets compared to the synthetic datasets. We acknowledge, however, that overall the SQL implementation scales linearly on real-data; additionally, we notice that the performance on real-world data is faster than the performance on the synthetic data with a factor of 1.4 at 7,800,000 rows.
Qualitative Experiment: Other than a performance assessment of our SQL implementation, we conduct an experiment by allowing the o and w values to vary on the original real-world dataset. Our inspiration is a business case in which we want to find the best sentinel rules in terms of *SentSupp* and *Conf.* We vary the granularity of the offset period, o, to be either month, quarter or year. We vary the warning period, w, to be of same granularity as the offset, and to start anywhere in between the period defined by the offset and 12 months prior to the period defined by the offset. When running this experiment, we found an interesting sentinel rule with *SentSupp* of 33 out of 78 input rows and 79.5% *Conf* that told us that if the number of Website hits goes up one year, the revenue will go up the following year. Although this rule is not a surprise, since we are dealing with a company in growth, we notice that our solution has discovered a sentinel rule by autonomously selecting o and w on real data, and the rule is exactly of the type and on the level of detail that we are looking for.
Experiment Summary: From the experiments we validate that our SQL implementation does indeed have $\mathcal{O}(n)$ computational complexity, and that it scales linearly to 10 million rows of aggregate data. We recall that a real-world dataset can easily be as small as 78 rows. A real-world scenario can, however, increase in complexity when the cardinality increases on the dimensions involved. From these experiments, we would expect that good performance can be gained for a real-world organization, e.g., on 6.5 years of data aggregated monthly for a company with 100 geographical locations and 1,000 products gives 7,800,000 rows. Furthermore, we found that even though a brute-force approach was applied in fitting o and w to get the best sentinel rules, such an approach can realistically be used on a real dataset to provide relevant results.

5 Related Work

For decades, focus has been on immediate data warehousing challenges of obtaining and storing data for business purposes, and making it accessible with high performance [24]. OLAP applications have emerged, but with the exception of Kimball's "Analytical Cycle" [14], there has been little theory such as CALM [17] about how to combine specific BI technologies to meet the management challenge of turning data into valuable decisions.

The idea that some actions or incidents are interlinked has been well explored in *association rules* [1]. The traditional case study of association rules has been basket-type association rules, and significant effort has been put into optimizing the initially proposed Apriori algorithm [4], including its extension to exploit the performance of a parallel shared-nothing multiprocessor system [3]. Improvements in performance as well as lower memory usage compared to the Apriori algorithm has been gained by growing a compressed frequent pattern tree in memory [13], and improvements in the selection process of thresholds that determine the interestingness of rules has also been introduced [22].

Sequential pattern mining introduces a sequence in which actions or incidents take place, with the intention of predicting one action or incident based on knowing another one. This adds to the complexity of association rules which makes the Apriori approach even more costly [5], thus new approaches to improving the performance of mining sequential patterns have emerged [12, 18, 21, 19]. Another aspect of sequential pattern mining has been the various approaches to handling the period of the sequence, e.g., involving multiple time granularities [6] or allowing for partial periodicity of patterns [10]. The focus on user control of the sequential pattern mining process in terms of applying constraints [8], and in a querying-type approach [7] has also been explored. The introduction of multi-dimensional databases has given rise to *multi-dimensional pattern mining* [20] which applies the same techniques to more dimensions than just one.

In general, association rule mining seeks to find co-occurrence patterns within *absolute data values*, whereas our solution works on the *relative changes in data*. In addition, association rule mining typically works on *categorical data*, i.e., dimension values, whereas our solution works on *numerical data* such as measure values. Sequential pattern mining allows a time period to pass between the premise and the consequent in the rule, but it remains focused on co-occurrence patterns within absolute data values for categorical data. Furthermore, our solution generates rules at the *schema level*, as opposed to the *data level*, using a contradiction elimination process. The combination of schema-level rules based on relative changes in data allows us to generate fewer, more general, rules that cannot be found with neither association rules nor sequential pattern mining. In Appendix B we demonstrate why sequential pattern mining does not find any meaningful rules in our running example presented in Section 2.

Other approaches to interpreting the behavior of data sequences are various regression [2] and correlation [11, 23] techniques which attempt to describe a functional relationship between one measure and another. In comparison, we can say that sentinel rules are a set of *"micro-predictions"* that are complementary to regression and correlation techniques. Sentinel rules are useful for discovering strong relationships between a smaller subset within a dataset, and thus they are useful for detecting warnings whenever changes (that would otherwise go unnoticed) in a relevant source measure occur. In addition, regression and correlation techniques do not support uni-directional relationships such as our solution. Regression and correlation based techniques, on the other hand, are useful for describing the overall trends within a dataset. In Appendix C, we specifically demonstrate using a concrete, realistic example how correlation tend to *blind the user by the average.*

Finally, the work on similarity search in timeseries databases [2] attempt to describe periods in which one measure behaves similar to another. This work is different from sentinel rules since it does not generate schema-level rules (or rules at all), furthermore it does not allow the description of a uni-directional relationships.

6 Conclusion and Future Work

We have proposed a novel approach for discovering so-called sentinel rules in a multi-dimensional database for business intelligence. The sentinel rules were generated at schema level, which means that they are more general and cleansed for contradictions, and thus easy to interpret. These sentinel rules can be used to expand the window of opportunity for an organization to act based on changes in the environment in which it operates. We demonstrated how an implementation in SQL could be done, and we showed that it scales linearly on large volumes of both synthetic and real-world data. We also demonstrated that sentinel rules with relevance for decision making can be extracted from real-world data. In this context, we proved the possibility of automatic fitting of both warning and observation periods. With regards to novelty, we specifically demonstrated that sentinel rules are different from sequential pattern mining, since sentinel rules operate at the schema level and use a contradiction elimination process to generate fewer, more general rules. Furthermore, we found sentinel rules to be complementary to correlation techniques, in that they could discover strong relationships between a smaller subset within a dataset; a relationship that would otherwise be "hidden in the average" using correlation techniques alone.

There are several perspectives for future work. First, an idea would be to use the SQL implementation described in this paper as a baseline for new improved algorithms in terms of performance. Secondly, the ability to automatically fit o and w on large volumes of data and different granularities should be explored. Third, it would make sense to seek for rules where multiple source measures are combined into a sentinel rule. Fourth, a natural development would also be to seek sentinel rules for multiple target measures at the same time to improve overall performance. Fifth, an idea could be to improve the solution in the multi-dimensional environment by allowing the sentinel rule mining to fit the aggregation level on dimensions automatically as well as automatically select the location and shape of the data area where the sentinel rules best apply.

From a business and decision-making stand point, more effort should be put into automatically pruning the sentinel rules found e.g., by assessing their interrelated relevance. Additionally, the degree of balance between the positive and the negative indications behind the sentinel rule, or to which degree rules are irrelevant based on orthogonal relationships between the source and the target measure, should be further explored in order for it to be automated.

7 Acknowledgments

This work was supported by TARGIT A/S. The SQL implementation and server setup was assisted by lead-programmers Jan Krogsgaard and Jakob Andersen.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *In Proc. of ACM SIGMOD*, pp. 207–216, 1993.

[2] R. Agrawal, K.I. Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in timeseries databases. *In Proc. of VLDB*, pp. 490–501, 1995.

[3] R. Agrawal and J.C. Shafer. Parallel mining of association rules. *IEEE TKDE 8(6)*: 962–969, 1996.

[4] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *In Proc. of VLDB*, pp. 487–499, 1994.

[5] R. Agrawal and R. Srikant. Mining Sequential Patterns. *In Proc. of ICDE*, pp. 3–14, 1995.

[6] C. Bettini, X.S. Wang, and S. Jajodia. Mining temporal relationships with multiple granularities in time sequences. *IEEE DEBU 21(1)*: 32–38, 1998.

[7] L. Feng, J.X. Yu, H. Lu, and J. Han. A template model for multidimensional inter-transactional association rules. *VLDB J. 11(2)*: 153–175, 2002.

[8] M. Garofalakis, R. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. *In Proc. of VLDB*, pp. 223–234, 1999.

[9] G. Graefe, R. Bunker, and S. Cooper. Hash Joins and Hash Teams in Microsoft SQL Server. *In Proc. of VLDB*, pp. 86–97, 1998.

[10] J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. *In Proc. of ICDE*, pp. 106–115, 1999.

[11] J. Han and M. Kamber. *Data Mining Concepts and Techniques.* (2nd Ed.) Morgan Kaufmann Publishers, 2006.

[12] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu. FreeSpan: frequent pattern-projected sequential pattern mining. *In Proc. of KDD*, pp. 355–359, 2000.

[13] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. *In Proc. of ACM SIGMOD*, pp. 1–12, 2000.

[14] R. Kimball. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses.* Wiley, 1998.

[15] W.S. Lind. *Maneuver Warfare Handbook.* Westview Press, 1985.

[16] Microsoft Corporation. *Microsoft Time Series Algorithm Technical Reference (Analysis Services - Data Mining).* <technet.microsoft.com/en-us/library/bb677216(SQL.100).aspx> current as of June 29th 2008.

[17] M. Middelfart. *CALM: Computer Aided Leadership & Management - How Computers can Unleash the Full Potential of Individuals and Organizations in a World of Chaos and Confusion.* iUniverse, 2005.

[18] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.C. Hsu. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. *In Proc. of ICDE*, pp. 215–224, 2001.

[19] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE TKDE 16(11)*: 1424–1440, 2004.

[20] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-Dimensional Sequential Pattern Mining. *In Proc. of CIKM*, pp. 81–88, 2001.

[21] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. *In Proc. of EDBT*, pp. 3–17, 1996.

[22] T. Wu, Y. Chen, and J. Han. Association Mining in Large Databases: A Re-examination of Its Measures. *In Proc. of PKDD*, pp. 621–628, 2007.

[23] Y. Zhu and D. Shasha. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. *In Proc. of VLDB*, pp. 358–369, 2002.

[24] R. Zurawski (ed.), C.S. Jensen, and T.B. Pedersen. *The Industrial Information Technology Handbook*, Multidimensional Databases and OLAP. CRC Press, 2005.

Appendix

A SQL-Based Implementation

The following SQL is written for Microsoft SQL Server 2005 as explained in Section 3. We assume without loss of generality that of the p measures in the dataset, C, $M_1...M_{p-1}$ are the source measures and M_p is the target measure.

Statement 1 creates a temporary table called *Period* where each unique value of (time dimension value) t, is sorted in ascending order and assigned an integer, *Id*, growing by 1 for each t. This table will allow us to select values of t for comparison with a given distance in periods, regardless of the format of the period field, t, in the database. In Statement 2, we create an index on the period table to optimize performance of the following operations.

By joining 4 copies of each of the original dataset and the period table (one for each of the periods: $t, t+o, t+w$, and $t+w+o$) in Statement 3, we create a table called *Result3* which contains the indications on the source measures as a result of the differences in measure values from period *id* to period *id+o*. In addition, we calculate the indications on the target measure from period *id+w* to period *id+w+o*. We calculate these indications for each cell (dimension combination) in the dataset. The custom function, *Ind*, handles the Formulae (3) and (4) and returns -1, 0, or 1 depending on whether the indication is negative, neutral or positive against the threshold α. This statement handles the shifting of facts, Formulae (1) and (2), and produces a Source-Target set (Formula (5) for our selected *p-1* source measures and one target measure. After executing Statement 3, we have a table which contains the same information as Table 3 in our running example with indications expressed as -1, 0 or 1.

Statement 4 counts the number of positive and negative indications on the source measure, and for each of these source measure indications, it summarizes the indications on the target measure. The output is stored in *Result4*. Since the indications are expressed as -1, 0 or 1, our contradiction elimination process can be carried out simply by this summation.

Statement 5 retrieves the potential rules from *Result4*, meaning that a source measure needs to have at least one indication with a consequential indication on the target measure, i.e., *ElimSupp*<> 0. For each of these rules, we calculate the sum of the support of source measure indications, *SentSupp*, the numeric sum of indications on the target measure, *AbsElimSupp*, as well as *MaxElimSupp* which is max(*ElimSupp*). In addition, Statement 5 calculates the *Direction* of the relationship between source and target measure where 1 is straightforward and -1 is inverted. The nature of *Direction* also helps us eliminate orthogonal rules since these will always have *Direction=0.* This is true because an orthogonal relationship means that both positive and negative indications on the source measure leads to only one type of indication on the target measure. Finally, Statement 5 calculates the number of indication rules, *IndRuleCount*, in the potential sentinel rule. This information is used to distinguish between bi- and uni-directional rules. The output of Statement 5 is stored in *Result5.*

Using *Result5* in Statement 6, we identify the sentinel rules that comply with the criteria of *SentSupp* $>= \beta$ and *Conf* $>= \gamma$. We use the values of *IndRuleCount*, *Direction*, and *MaxElimSupp* to describe the sentinel rule in accordance with our notation. The output is stored in *FinalResult.*

SQL Statements for FindSentinels($C, o, w, \alpha, \beta, \gamma$)

```
(1) SELECT $T$, ROW_NUMBER() OVER (ORDER BY $T$) AS Id
     INTO Period FROM (SELECT DISTINCT $T$ FROM $C$) AS subselect

(2) CREATE INDEX PeriodIndex ON Period($T$)

(3) SELECT a.$T$, a.$D_2$, ...  , a.$D_n$,
           Ind((b.$M_1$-a.$M_1$)/a.$M_1$,$\alpha$) AS $M_1$ind,
           Ind((b.$M_2$-a.$M_2$)/a.$M_1$,$\alpha$) AS $M_2$ind,
           ...
           Ind((d.$M_p$-c.$M_p$)/c.$M_p$,$\alpha$) AS TMind
     INTO Result3
     FROM $C$ a, $C$ b, $C$ c, $C$ d,
          Period pa, Period pb, Period pc, Period pd,
     WHERE a.t=pa.t AND b.t=pb.t AND c.t=pc.t AND d.t=pd.t AND
           a.$D_2$=b.$D_2$ AND a.$D_2$=c.$D_2$ AND a.$D_2$=d.$D_2$ AND
           ...
           a.$D_n$=b.$D_n$ AND a.$D_n$=c.$D_n$ AND a.$D_n$=d.$D_n$ AND
           pb.id=pa.id+$o$ AND pc.id=pa.id+$w$ pd.id=pa.id+$w$+$o$

Custom function Ind is implemented as follows for source and target measure:
    SIGN(ROUND(((b.$M_i$-a.$M_i$)/a.$M_i$)*100/$\alpha$, 0, 1)) AS $M_i$ind
    SIGN(ROUND(((d.$M_p$-c.$M_p$)/c.$M_p$)*100/$\alpha$, 0, 1)) AS TMind

(4) SELECT sourcemeasure, Ind, COUNT(*) AS IndSupp, SUM(TMind) AS ElimSupp
     INTO Result4
     FROM (SELECT "$M_1$" AS SourceMeasure, $M_1$ind AS Ind, TMind FROM Result3
     UNION ALL SELECT "$M_2$" AS SourceMeasure, $M_2$ind AS Ind, TMind FROM Result3
     ...
     UNION ALL SELECT "$M_{p-1}$" AS SourceMeasure, $M_{p-1}$ind AS Ind, TMind
     FROM Result3) AS subselect
     WHERE Ind<>0
     GROUP BY SourceMeasure, Ind

(5) SELECT SourceMeasure, SUM(IndSupp) AS SentSupp,
            SUM(ABS(ElimSupp)) AS ElimSupp, MAX(ElimSupp) AS MaxElimSupp,
            AVG(SIGN(Ind)*SIGN(ElimSupp)) AS Direction,
            COUNT(*) AS IndRuleCount
     INTO Result5
     FROM Result4
     WHERE ElimSupp<>0
     GROUP BY SourceMeasure
```

```
(6) SELECT SentinelRule, SentSupp, Conf INTO FinalResult FROM
        (SELECT SourceMeasure+'->inv(target)' AS SentinelRule,
            SentSupp, ElimSupp*100/SentSupp AS Conf FROM Result5
            WHERE Direction<0 and IndRuleCount=2
        UNION ALL SELECT SourceMeasure+'->target' AS SentinelRule,
            SentSupp, ElimSupp*100/SentSupp AS Conf FROM Result5
            WHERE Direction>0 and IndRuleCount=2
        UNION ALL SELECT SourceMeasure+'_inc->target_dec' AS SentinelRule,
            SentSupp, ElimSupp*100/SentSupp AS Conf FROM Result5
            WHERE Direction<0 and MaxElimSupp<0 and IndRuleCount=1
        UNION ALL SELECT SourceMeasure+'_dec->target_inc' AS SentinelRule,
            SentSupp, ElimSupp*100/SentSupp AS Conf FROM Result5
            WHERE Direction<0 and MaxElimSupp>0 and IndRuleCount=1
        UNION ALL SELECT SourceMeasure+'_dec->target_dec' AS SentinelRule,
            SentSupp, ElimSupp*100/SentSupp AS Conf FROM Result5
            WHERE Direction>0 and MaxElimSupp<0 and IndRuleCount=1
        UNION ALL SELECT SourceMeasure+'_inc->target_inc' AS SentinelRule,
            SentSupp, ElimSupp*100/SentSupp AS Conf FROM Result5
            WHERE Direction>0 and MaxElimSupp>0 and IndRuleCount=1
        ) as subselect
    WHERE SentSupp>=β and Conf>=γ
```

B Sentinels Rules vs. Sequential Pattern Mining

Sequential pattern mining identifies patterns of items that occur with a certain frequency in a dataset of transactions grouped in sequences, i.e., a sequence could look like this: $\{(ab)c\}$, where *(ab)* and c are transactions. The support of patterns such as "item a leads to item c" are found by counting the number of times a transaction containing a is followed by a transaction containing c.

To exemplify the difference between our solution and sequential pattern mining; specifically the difference between working on absolute data *values* and relative data *changes*, we can simply subject our running example dataset in Table 1 to sequential pattern mining. Since *all* values in this dataset are *unique*, we will only find patterns with an absolute support of only 1 and confidence of 100%, as follows:

"NBlgs=20 leads to Rev=9000"
"CstPrb=50 leads to Rev=9000"
"WHts=1000 leads to Rev=9000"
"NBlgs=20 and CstPrb=50 leads to Rev=9000"
"NBlgs=20 and WHts=1000 leads to Rev=9000"
"NBlgs=20 and CstPrb=50 and WHts=1000 leads to Rev=9000"
...

The rules above above have been generated using only the first line in the dataset in Table 1, meaning that the complete set would generate 12x6=72 rules. If we think of a 1,000 record dataset with the same properties as Table 1, such a dataset would result in 6,000 rules. In other words, since the data-level rules generated by sequential pattern mining are *value specific*, as opposed to relative change "events" at the schema level, we get a large output in which we do not find any significant patterns. Thus no meaningful, high-level rules can be found by sequential pattern mining directly.

C Sentinels Rules vs. Correlation Techniques

If we compare sentinel rules with correlation techniques, one could intuitively think that correlations would be able to detect exactly the same relationships as sentinel rules, at least when comparing correlations to bi-directional sentinel rules. However, when subjecting data to correlation techniques [11, 23], we do not find the same explicit relationships as we do with sentinel rules.

The reason that correlation techniques gives different results compared to our sentinel rules, lies in the fact, that correlation techniques are focused on identifying relationships with the minimal Euclidean distance between pairs, this means that there is a tendency to favor "smooth" data series with few outliers. On the other hand, the intention of sentinel rules is to give a user an early warning, and from an operational perspective we can even say that we are particularly interested in the source measure changes (perhaps unusual/"outlier") that appear to have consequential change effect on the target measure. This difference means (from a sentinel perspective) that correlation techniques "blind us by the average" when our goal is to get an early warning.

Tables 11-13 exemplify these differences by describing the relationship between a source measure, A, and a target measure, B, with both sentinel rules and correlation. The data series in Tables 11-13 have been pre-processed so that each line represents a combination of A for time period, t, and B for time period, $t + w$ ($w = 1$). The distance in time between the lines is o. We operate with the same values for α, β, and γ as in our running example, specifically: $\alpha = 10\%$, $\beta = 3$, and $\gamma = 60\%$. For correlation, we say that it needs to account for more than 50% of the variation (correlation-coefficient$^2 > 0.5$) between the series in order to be relevant. In fact, it does not even make sense to consider correlations that account for 50% or less, since such correlations would be less precise than simply flipping a coin to decide whether target measure, B, will go up or down. We can now read the individual indications directly, and by correlating the two data series, we get a lagged correlation with the same warning period as the sentinel rules. Table 11 shows a bi-directional sentinel rule, $A \rightsquigarrow inv(B)$, that is **not** discovered using correlation techniques, Table 12 shows a uni-directional sentinel rule, $A\blacktriangledown \rightsquigarrow B\blacktriangle$, that is **not** discovered using correlation techniques. In Table 13, correlation and sentinel rules both find a uni-directional rule, $A\blacktriangledown \rightsquigarrow B\blacktriangle$.

The data from Table 11 is plotted in Fig. 2(a), note again that the top line (B) is shifted $w = 1$ period(s) backwards. We note the visible changes for A and B, where A has been scaled by a factor of 6 for better visibility. The changes that leads to sentinel rules have been highlighted by an arrow from A to B. When looking at Fig. 2(a) it seems clear that there is a relationship

Table 11. $A \rightsquigarrow inv(B)$

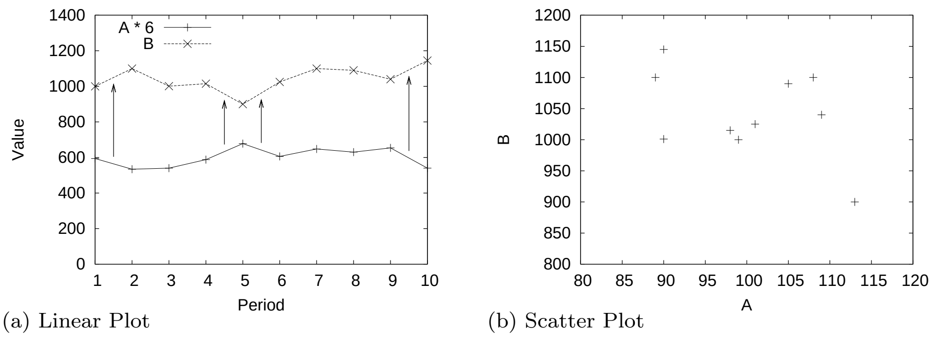
A	B	Indication
99	1000	
89	1100	$A\blacktriangledown \rightarrow B\blacktriangle$
90	1001	
98	1015	
113	900	$A\blacktriangle \rightarrow B\blacktriangledown$
101	1025	$A\blacktriangledown \rightarrow B\blacktriangle$
108	1100	
105	1090	
109	1040	
90	1145	$A\blacktriangle \rightarrow B\blacktriangledown$
Sentinel rule: $SentSupp_{A \rightsquigarrow inv(B)} = 4$ $Conf_{A \rightsquigarrow inv(B)} = 100\%$		
Correlation: *Coefficient* = -0.4307 *Accounts for* = 18.55%		

Table 12. $A\blacktriangledown \rightsquigarrow B\blacktriangle$

A	B	Indication
500	1000	
400	1200	$A\blacktriangledown \rightarrow B\blacktriangle$
363	1095	
399	1200	
350	1400	$A\blacktriangledown \rightarrow B\blacktriangle$
316	1265	
323	1285	
355	1410	
300	1600	$A\blacktriangledown \rightarrow B\blacktriangle$
329	1755	
Sentinel rule: $SentSupp_{A\blacktriangledown \rightsquigarrow B\blacktriangle} = 3$ $Conf_{A\blacktriangledown \rightsquigarrow B\blacktriangle} = 100\%$		
Correlation: *Coefficient* = -0.6919 *Accounts for* = 47.87%		

Table 13. $A\blacktriangledown \rightsquigarrow B\blacktriangle$

A	B	Indication
100	1000	
90	1100	$A\blacktriangledown \rightarrow B\blacktriangle$
81	1210	$A\blacktriangledown \rightarrow B\blacktriangle$
73	1331	$A\blacktriangledown \rightarrow B\blacktriangle$
66	1464	$A\blacktriangledown \rightarrow B\blacktriangle$
59	1611	$A\blacktriangledown \rightarrow B\blacktriangle$
53	1772	$A\blacktriangledown \rightarrow B\blacktriangle$
48	1949	$A\blacktriangledown \rightarrow B\blacktriangle$
43	2144	$A\blacktriangledown \rightarrow B\blacktriangle$
39	2358	$A\blacktriangledown \rightarrow B\blacktriangle$
Sentinel rule: $SentSupp_{A\blacktriangledown \rightsquigarrow B\blacktriangle} = 9$ $Conf_{A\blacktriangledown \rightsquigarrow B\blacktriangle} = 100\%$		
Correlation: *Coefficient* = -0.9688 *Accounts for* = 93.85%		



(a) Linear Plot (b) Scatter Plot

Figure 2: Graphs based on data from Table 11.

between the changes. However, when displaying the same data in a scatter plot, Fig. 2(b), to test the correlation visually, we find that the relationship between A and B seems to be rather chaotic or at least non-linear. This is the reason that we get a very poor correlation, whereas on the other hand, we find a strong sentinel rule based on four indications.

Following these examples, we confirm the differences between sentinel rules and correlation thecniques by applying correlation techniques [11, 23] to our running example in Table 1. Correlation techniques find lagged correlations between all source measures and the target measure, but the only correlation that accounts for more than 50% of the variation, is identified between *WHts* and *Rev* which is in contrast to the bi- and uni-directional sentinel rules between *NBlgs* and *Rev*, and between *CstPrb* and *Rev*, which we found with our solution.

In summary, we could say that sentinel rules are a set of *"micro-predictions"* that are complementary to correlation techniques. Sentinel rules are useful for discovering strong relationships between a smaller subset within a dataset, and thus they are useful for detecting warnings whenever changes (that would otherwise go unnoticed) in a relevant source measure occur. Correlation techniques, on the other hand, are useful for describing the overall trends within a dataset.