A Systematic Approach for Dynamic Targeted Monitoring of KPIs

Alejandro Maté[1] Kostas Zoumpatianos[1] Themis Palpanas[3] Juan Trujillo[2]
John Mylopoulos[1] Elvis Koci[1]

[1] Department of Computer Science, University of Trento, Italy
alejandro.matemorga@unitn.it {zoumpatianos,jm}@disi.unitn.it
elvis.koci@studenti.unitn.it
[2]Software and Computing Systems, University of Alicante, Spain
jtrujillo@dlsi.ua.es
[3] Paris Descartes University, France
themis@mi.parisdescartes.fr

Abstract

There has been growing interest for more than a decade in Business Analytics as a means for improving business performance. One of the most popular Business Analytics technique involves monitoring performance by means of Key Performance Indicators (KPIs). A KPI is a powerful tool that relates enterprise data to business goals, thereby enabling managers to guide the analytic process and identify deviations in their strategic plan. Nevertheless, monitoring KPIs requires that they are evaluated at multiple levels of detail, in order to identify potential problems earlier instead of being noted after the fact. Unfortunately, there are obstacles to the generation and enactment of such monitoring processes. In particular, there is no systematic, tool-supported process for defining what is to be monitored given a strategic plan, nor are there tools for automatically generating monitoring queries. As a result, monitoring consists of a manual process whereby queries are generated for high level indicators across a few scorecards and dashboards. In this paper we present a systematic semi-automatic approach that covers the entire monitoring process. Our approach performs a partial search guided by the KPIs of the company, generating queries required during the monitoring process. Thanks to our approach, users become aware of the existence of problems and where they are located, without requiring a priori information about the nature of the problem being searched. Moreover, we have implemented our approach in our CASE tool HERMES and evaluated the results on a case study using real data.

Keywords: Monitoring, Business Intelligence, intelligent search, KPI, business analytics

1 Introduction

Business monitoring is a critical activity that looks for problems in business performance and alerts of their existence together with its source. It is one of the key tasks for companies because it allows decision makers to take corrective actions sooner rather than later, but it is also a challenging task because of the large volumes and high velocity of data that need to be processed.

Business monitoring is typically supported by an information system that provides information about several Key Performance Indicators (KPIs) [18], spread across one or more Balanced Scorecards [13] and multiple dashboards. Traditionally, business monitoring has been based on the evaluation of the aggregated values of KPIs by users

who regularly check the scorecard in order to ensure that everything is in order. For example, the KPI "Customer retention increased by 3%" considers the customers retained throughout all our stores in the current year. In order to provide a more complete view for KPI monitoring, dashboards provide detailed information. For example, if the abovementioned KPI is failing, what has been its trend for the past months?

However, dashboards are not flexible enough to quickly and accurately identify potential problems that may arise. What if we could not expect to achieve our target by the end of the year because our shops do not retain customers in certain regions? On the one hand, adopting a bird's eye view of the data we are monitoring over makes it difficult to pinpoint a problem. On the other hand, adding all available information or subKPIs would render the dashboards useless due to their complexity. As there are no clear criteria on what information should be provided and how, such approach is currently based on the experience of the business analyst, and does not allow users to easily discover potential deviations in order to apply corrections. Therefore, the main objective of this paper is to propose a systematic process for monitoring KPIs from multiple perspectives by automatically generating monitoring queries.

Previous research has tackled this problem by either (i) providing the user a cube that contains all the discovered deviations in the data identified with respect to current trends [21, 19, 20], or (ii) by specifying a certain set of targets to be monitored within a subset of the data [23, 24]. Nevertheless, the first approach does not take into account the goals and expectations of the users, thus potentially providing information to the user that is irrelevant. For example, during an economic downturn we expect our sales trends to change. Additionally, the second approach requires the user to specify a subset of the data to be monitored. While it takes into account the expectations of the user, it also requires a priori knowledge of the problem being monitored and does not provide detailed information about the source of the problem, since it does not perform an in-depth analysis. For example, using the abovementioned KPI, we would need to know that we have a significant problem at stores levels instead of in a certain region or country.

In this paper we propose a systematic targeted monitoring approach based on user KPIs, that provides insights for the user about existing deviations in subKPIs. First, our approach takes each KPI defined and performs a guided search looking for the most significant subKPI segments contributing to the deviation of the KPI. Then, once these segments have been identified, it provides the user a result tree that can be navigated by the user in order to visualize how the deviation is spread across instances. As a result, our approach helps the user to pinpoint the root of the problem. The advantages of our approach are that it (i) dynamically locates the most relevant contributors to the deviation, whether they are general problems or very specific instances, by drilling deeper only when it is relevant, (ii) requires no a priori knowledge of the problem, (iii) shows the user the most important information and allows him to navigate the results in order to understand the structure of the problem, and (iv) allows the user to visualize the results for the overall business plan, allowing him to decide what problems should be prioritized and what other targets will be affected by the corrections.

The remainder of this paper is structured as follows. Section 2, presents an overview of techniques proposed for finding deviations and monitor datasets. Section 3 describes the main components involved in our approach and how they interrelate to each other. Section 4 provides details about the implementation of our CASE tool for supporting our approach. Section 5 presents the results of applying our approach to a case study using synthetic data. Finally. Section 6 presents the conclusions and sketches the future work to be done.

2 Related Work

2.1 Strategy modeling

Initial works to represent business strategies and monitor them were proposed during the 90's and early 2000 [13, 12]. The Balanced Scorecard was proposed in [13], which quickly became one of the most widespread monitoring techniques for enterprises. A scorecard presents a high level list of the most important KPIs for monitoring business performance from different perspectives. However, the simplicity of the Balanced Scorecard made it impossible to perform detailed analysis on KPIs nor reason about the suitability of the business strategy and business initiatives. Dashboards [8] and Strategy maps were proposed [12] to tackle

these problem. Dashboards provide context information on one or more KPIs, thereby enabling more detailed analysis, while Strategy Maps capture the relationships between business initiatives and goals. However, dashboards are not flexible, they offer a pre-set view of the data and are not meant for exhaustive analysis. Furthermore, Strategy maps are not completely formal and do not integrate a view of the status of the business. thus offering limited reasoning support.

More recently, the Business Motivation Model (BMM) [16] and the Business Intelligence Model (BIM) [22, 11] have appeared in order to formalize business strategies. The former proposes a metamodel that captures concepts used in textual business plans, whereas the second proposes a domain specific language for modeling business strategies using a goal oriented approach.

2.2 Data cube mining techniques

In the last twenty years there has been extensive research [5, 9, 21, 19, 20, 10], as well as a great amount of applications developed from business vendors (SSAS, Hyperion, Cognos, etc.) on OLAP technology. An OLAP system [5] allows a user to quickly acquire all the interesting information in a large data warehouse by querying on aggregates at different levels of abstraction. Additionally, he is given the ability to drill down on specific areas of the database, in order to get more detailed information about the parts that look most interesting. In such systems it is expected to have a high number of dimensions and measures, one for each attribute of the original tuples. Each dimension is used by the analyst for grouping and filtering and it is common that they are part of hierarchies for a top down data analysis. An analyst can query an OLAP system for aggregates of different types (SUM, AVG, COUNT) in respect to a grouping he has defined and a specified measure. All the possible groupings that an analyst can query to the system form a multi-dimensional data-cube. This structure contains a set of cells, one for each most detailed grouping, which are contained by parent cubes at multiple different levels. Each level summarizes the data aggregates in the levels below it.

Various techniques have been proposed for exploring a data-cube efficiently [17]. But as Sarawagi et al. [21] have noted, by simply observing the aggregates it becomes very tedious to identify on which regions one has to drill down in order to get information of interest. To solve this problem they have proposed a method that guides the user to the "most anomalous" areas by presenting some indicators of exceptions based on deviations from a statistical model. Additionally, using a DIFF operator [19] the user can explore the reasons that explain these anomalous segments.

When having to deal with large amounts of dynamic data coming into streams though, simple aggregations are not enough. As Chen et al. [4] have noted, it is important to be able to do regression and trend analysis and while traditional data cube technology is good for static aggregates it fails to explain trends in the multi-dimensional space. For this reason they have introduced the stream cube architecture [10] which allows for the multi-dimensional monitoring of regression lines. Subsequent work [1] enabled the monitoring of trend exceptions, by capturing regression lines with slopes above an exception threshold.

While both previous methods were able to identify exceptional behaviors in a data warehouse, they were tailored for untargeted analytics. This means that deviations were calculated in regards to the rest of the data warehouse. While this is important when the analyst looks for unexpected results, it is not helpful when a formal strategy has to be evaluated for success or failure. In such cases, aggregations in OLAP cubes are used as input to KPI calculations, which comprise not only of a value but also a target that has to be met at a specific time point. This target can be different for every subKPI and also vary across time. Our target is to guide the analyst to the most failing - in regards to a target value - subKPIs that mostly explain why a formally specified goal is going to be successful or not. That said the target of our algorithm is to find the subKPIs with that contribute most to a global failure based always on their local target values.

2.3 Forecasting

Normally, KPIs are defined together with strategic plans in order to help analysts measure the success of their goals. While it is important to know why a past goal has failed or succeeded, it is even more important to monitor what the trend is for a current goal. Is the goal going to be successful or failed eventually? Therefore, formally defined goals refer always to the future. This means that target values

for KPIs associated with the measurement of goal, are always defined for the future. There has been a lot of work done in this direction for forecasting sequential data. This work is mostly within the domain of statistics. Models such as ARIMA, simple moving averages or even trend estimation can be used to predict a future value. The interested reader can study in detail the variety of methods in [3].

2.4 Data Series Mining

KPIs are inherently sequential data, most of the times involving time. A lot of methods have been developed for finding similarities in sets of data-series [25, 2]. Such methods can be used complimentary to our approach for performing outlier analysis and to assist analysts search for clusters or data that are deviating from similar past behavior. These techniques can be extended in the case of uncertain data-series [6, 7].

3 A Systematic Approach for Targeted Monitoring

Our approach is based on an architecture with four clearly differentiated components. We first present the architecture and then discuss each component.

3.1 System Architecture

Our architecture for targeted monitoring is based on the idea of exploiting existing domain knowledge captured in the business strategy and presenting the results in the same terms. To this aim, the first component in the architecture, shown in Figure 1, is the Business Strategy Model. The Business Strategy Model is responsible for capturing business goals and their corresponding KPIs. These elements contain all the relevant information for the monitoring process such as target values that the users wish to achieve for each KPI, the worst value expected, time to target available for each KPI, etc.

The business strategy provides information to the monitoring module, that is responsible for the monitoring process according to parameters specified by the user. First, the monitoring module preprocesses the data in order to calculate all the values required for the search. Then, the monitoring process starts for each KPI. First, with the aid of the query generator, it poses an initial query to the data warehouse. Afterwards, the module starts the monitoring process, interacting on each step with the query generator in order to retrieve all the necessary values from the information system. The initial query created by the query generator is based on the definition of each KPI. Then, as further detailed in Sections 3.4 and 3.5, during each step it generates a new query by constraining the previous one to a subset of the instances and to a certain dimension level. These queries allow the monitoring module to store the most relevant deviations found during each iteration.

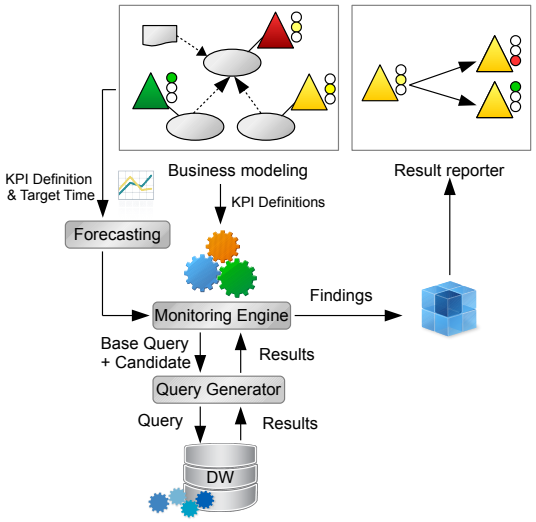

Figure 1: Architecture of our monitoring solution

Finally, the result set obtained during the monitoring process is sent to the result reporter in order to be analyzed by the user. The result reporter focuses on highlighting the main sources of deviation for each KPI found in the data. To this aim, the result reporter shows the elements found on each iteration by the monitoring module in a hierarchical manner, as detailed in Section 3.6.

3.2 Business Strategy Modeling and Process Setup

Business Strategy Modeling plays a key role in the targeted monitoring process. It constitutes the starting point of the process, where users define business goals, the relationships between them, and specify KPIs that will monitor these goals, among other elements. The combination of these elements serves a double purpose: (i) it allows users to iden-

tify what goals and KPIs are more important for them and how often they should be monitored, and (ii) it allows users to understand, in their own domain, the results of the monitoring process.

The most important construct in the business strategy for our approach is the KPI. KPIs provide information about how to calculate the values to be monitored, which is the basis of the process. However, they also provide more important data, such as the target value to be met by the KPI, the time to target, which determines when the KPI should meet its mark, the threshold to separate acceptable performance from failure, and a worst case threshold to identify severely failing KPIs.

In order to support the business strategy modeling step we make use of the Business Intelligence Model (BIM) framework [22, 11], which was extended using the Semantic Vocabulary and Business Rules (SBVR) proposal for supporting KPI definition [15]. BIM provides the basic constructs to capture business goals, indicators that monitor them and represent KPIs, and situations, that represent external and internal factors affecting business goals and help users understand why their expectations may not be met or be exceeded. Additionally, the SBVR extension allows users to formalize KPIs and derive data warehouse queries from their definition. These definitions are supported by a Business Dictionary that represents a business ontology, and allows users to specify specialized instances of each KPI, for example to specify customized target values for certain subKPIs. For a more detailed explanation we refer the reader to [15].

Once we have modeled the business strategy and have a clear view of the important KPIs to monitor we need to prepare the monitoring process. Before we can start, we need to perform two tasks. First, we have to decide if we are going to use forecasted values. Usually, when KPIs are used for monitoring, it is often the case that they are failing until the time to target is close to 0, because they actually represent *lag* information [14]. For example, a KPI "Increase sales by 3%" will be failing during most part of the fiscal year until we are near its conclusion. In these cases, we need to either forecast the actual values of the KPIs to the point in time when they will be evaluated or we need to interpolate their target values to the current period of time. Second, we need to calculate the target values for the different subKPIs according to the dimension levels available in the data warehouse. This task can be performed in different ways, as mentioned in [24]. First, users can manually define a hierarchy of target values for each KPI. This is the case of Europe 2020, where multiple dimensions and levels, such as country, province, population age, and so on have their own manually defined target values for each KPI. Second, we can apply interpolation techniques like proportional projection, using a size attribute for each instance (e.g. population), or direct projection, where we get the values of the previous period for each subKPI and add the corresponding increment. For example, "Increase sales by 3%" would imply a 3% increase across all instances contributing to the KPI. Given that this can be a time consuming process, the calculus of target values is performed during the setup step, before the monitoring process starts. Then, the same values can be reused during the whole period that we are actively monitoring. Due to paper constraints we do not discuss these techniques any further.

After the process setup has been completed, we need to proceed to the monitoring step, where we require an approach to perform an intelligent search of the data space.

3.3 KPI Monitoring: Performing an Intelligent Search

Ideally, we would like to know all relevant deviations in the data so that appropriate actions can be taken, and, if possible, targeted specifically towards problematic instances. In order to perform a meaningful identification of instances, we can take advantage of the fact that most Business Intelligence (BI) systems are supported by a data warehouse, structuring the data into facts (center of analysis) and dimensions (context of analysis). Dimensions allow us to aggregate the results at different granularity using multiple axis depending on the perspective we wish to adopt.

Therefore, we can reformulate our problem so that, for each KPI K_i that has an aggregated value, we want to know if any instance j, defined by a set of dimension levels $D = l_1, ..., l_n$, and contributing to the value of K_i, presents an anomaly, such that K_{ij} is deviating. A formal definition of this deviation is given below.

Definition 1 (KPI-Deviation) *A KPI-Deviation is the difference between the target and the actual value of the KPI at a given time point. It can be*

mathematically formalized as:

$$\mathcal{D}_K = Target_K - Actual_K \tag{1}$$

As data warehouses typically present multiple levels of aggregation for each dimension, the problem becomes recursive thus requiring to repeat the same steps for K_{ij} until the finest level of detail has been reached in order to ensure that no deviation goes unnoticed. Unfortunately, such exhaustive approach has three important drawbacks. First, it is time consuming, as we always drill into all the potential combinations of dimensions and we need to calculate the value of the subKPI K_{ij} for all instances at each point. Second, it extends the search space through all dimension levels without guaranteeing that they will be useful for pinpointing the problem, increasing the number of operations performed as the number of individual instances increases. Third, it overloads the user with information, making him unable to understand the results, mixing important findings with irrelevant data, thereby rendering the process pointless.

An alternative to this problem is to select a certain set of dimensions and levels to monitor, and constantly monitor them for each indicator [24]. This way, the most relevant set of instances for the user can be monitored automatically while the number of queries generated is controlled.

However, a more interesting approach is to direct the search across several dimensions and levels, by providing criteria that drills into the dimensions containing instances that are contributing most to the deviation, and report only the most significant findings. This approach allows the user to better understand the problems arising, by focusing on the main contributors to the deviation and gaining insight about how the deviation is spread across instances and levels.

In order to apply this approach, we adapt the concept of *Information Gain* (IG) used in decision trees. IG measures the change in entropy between the initial set of instances available and the set classified according to a candidate attribute a. Similarly, we can consider the deviation in a KPI as a classification problem, where we wish to classify the instances according to their contribution, whether they are responsible for the failure or for an outstanding performance. Then, the set of available dimensions D act as attributes that allow us to separate the data into meaningful instances which are failing or succeeding.

Initially, we could consider using KPI values (current or forecasted) to classify each KPI into succeeding, struggling, or failing. However, not all subKPIs have the same weight with regards to the parent KPI. For example, North America and Europe can both be failing by 2% for a KPI "Increase sales by 5%". Nevertheless, North America may be accounting for 70% of the sales while Europe only makes up for 30%. While both subKPIs are failing, North America is a much more relevant instance than Europe in this case. In order to avoid this pitfall, we redefine the concept of IG as follows:

$$\mathcal{C}_{Kj} = \omega_j * \mathcal{D}_{Kj} \tag{2}$$

$$\mathcal{IG}_l = \frac{(max(\mathcal{C}_l) - min(\mathcal{C}_l))}{\sum_{j=1}^{|l|} \mathcal{C}_{Kj}} \tag{3}$$

Where $\mathcal{D}_K$ represents the deviation of a KPI K, $\mathcal{C}_{Kj}$ represents the weighted (ω) contribution of instance j to the deviation $\mathcal{D}_K$, where ω is calculated according to the size of j compared to other instances at the same level for weighted functions such as avg, and $\mathcal{IG}_l$ denotes the IG provided by a candidate dimension level. This definition of IG prioritizes dimension levels with instances that present more extreme deviation values with regards to the target as opposed to those that spread the deviation evenly across all instances. Therefore, this allows us to direct the search through the data space drilling always into the dimension with the biggest discrepancy. The denominator will be constant for all dimensions whose members aggregate the value of the parent KPI. In other words, $\sum_{j=1}^{|l|} \mathcal{C}_{Kj} = \mathcal{D}_K$. Thus, a higher IG in one dimension implies that the deviation is more concentrated into a single instance compared to other dimensions.

However, although it is interesting to drill into dimension levels with more extreme values, we may be interested in slicing the data in such a way that we find not one instance, but rather sets such as the top-10 contributors to the deviation. It is easy to see that these are specific cases of a more general approach, IG provided by top-K vs bottom-K contributors, which redefines Equation 3 as follows:

Data: K-value : N, KPIs: $\mathcal{K}$, Target Value Projections: P, Dimension Set: D, IL-threshold : $ILTh$
Result: Set of deviations found

$R = \emptyset$;
foreach $k \in \mathcal{K}$ **do**
  append(R, k); $K_p = k$; $K_s = k$; $IG = 0$; $IL = 0$;
  while $IL < ILTh$ **do**
    $K_p = K_s$;
    foreach d *in* D **do**
      $l = GetCandidateDimensionLevel(d)$;
      $K_{d,l} = RestrictToDimensionLevel(K_p, l)$;
      $IG_d = CalculateIG(K_{d,l}, P, N)$;
    end
    $d = GetMaxIGDimension(IG_d)$;
    $DrillDownDimension(d)$;
    $l = GetCurrentDimensionLevel(d)$;
    $K_s = selectedInstances(K_s, l)$;
    $IL = IL + CalculateIL(K_s, K_p)$;
    append(R, K_s);
  end
end
return R;

Algorithm 1: Top-K search algorithm

$$\mathcal{IG}_l = \frac{(\sum_{i=1}^{k} asc(\mathcal{C}_l)) - (\sum_{i=1}^{k} des(\mathcal{C}_l))}{\sum_{j=1}^{|l|} \mathcal{C}_{Kj}} \quad (4)$$

3.4 An Algorithm for KPI Monitoring

So far we have defined criteria to guide the search. Nevertheless, this criteria does not provide a clear stopping condition. Indeed, although at every step we choose the most relevant perspective of the data according to the available dimensions, we would potentially keep drilling down until we reached the most detailed view possible. Such case would only be interesting for the user if there were few detailed instances accounting for most of the total deviation, but completely irrelevant otherwise. In order to avoid this situation we could set a threshold for IG below which we would stop the drilling process. However it is worth noting that the IG of each level varies depending on what other levels have been previously selected. Therefore, a threshold cannot be set and dimensions cannot be discarded from the search space just because they initially present a low IG.

As we wish to drill down only as long as the size of the problem we are looking at is still significant for the user, we introduce the concept of *Information Loss* (IL). IL represents the intuitive notion that we are setting aside part of the problem, when we drill down into the most relevant descendants of a set of instances. IL captures this value by calculating the difference between the total deviation of the parent KPI K and the contribution to the deviation (C_{Kj}) of the candidate instances selected for further search at level l. IL is aggregated as the process drills in through multiple levels, eventually going above a threshold specified by the user and marking the process as finished for the current KPI.

$$\mathcal{IL}_{Kl} = \mathcal{D}_K - \sum_{i=1}^{|l|} C_{Kj} * isSelected(i) \quad (5)$$

Now that we have described our approach for guiding and stopping the drilling process, we only need to define clear criteria as to what instances should be selected for further search. The first option is to maintain the same criteria we used for selecting a dimension level, and drill into the top-K candidates. However, as we do not have a priori knowledge of how the deviation is spread across in-

stances, it may be the case that less (or more) than K instances should be analyzed. An alternative is to specify a minimum coverage threshold Th instead of the parameter K. Using this criterion, the user specifies the minimum deviation threshold Th with regards to the parent KPI that he wants to cover on each iteration:

$$\frac{\sum_{i=1}^{n} C_{Kj}}{\mathcal{D}_K} > Th \tag{6}$$

As a result, applying this criterion, the algorithm selects the top-N instances required to meet the threshold stated by the user, allowing for a flexible selection on each iteration.

Finally, after defining all the necessary elements for our search algorithm, we proceed to describe the process followed, as shown in Algorithm 1. The algorithm takes as input the following parameters: number N representing K candidates, the KPIs to be monitored, the set of target value projections, the set of dimensions available, and the IL threshold. Then, for each KPI, it starts the search process. First, it retrieves the candidate level of each dimension, one at a time (line 7). The candidate level is used to further restrict the current KPI (line 8). The restricted KPI $K_{d,l}$ is used to calculate the IG provided by the candidate level l using Equation 4 (line 9). Then, the algorithm retrieves the dimension whose candidate provided the highest IG and restricts the KPI according to the level specified (lines 11, 12). Afterwards, we select the most relevant instances according to their contribution and append them to the relevant results (lines 13, 14, 16). Finally, we calculate the IL by comparing the deviation within the selected subset of instance and the parent KPI (line 15). After we have finished the current iteration, we start the cycle again with the new subset of instances obtained until we finish with the current KPI and move to the next one. We can see a graphical representation of the process in Figure 2 [1].

In this figure, we are drilling through the top-1 instances. First, we drill into the Time dimension to analyze the IG provided by the level Quarter (step 1). As the employment is relatively stable across quarters, the IG provided by the Time dimension is very low. Next, we drill into the Geographic dimension to analyze the IG provided

[1]Example is based on the Europe 2020 dataset, further described in Section 5.

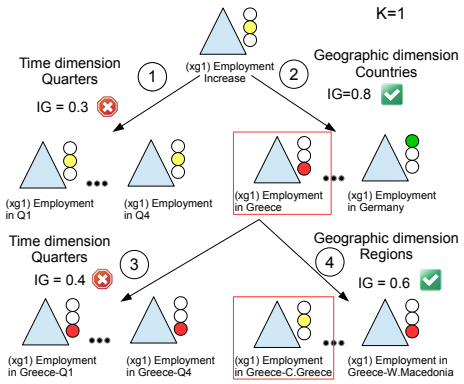

Figure 2: Example of the multidimensional search process performed by the algorithm

by the level Country (step 2). As certain countries present higher absolute deviations than others, the Country level provides significantly higher IG. Therefore, we select the Country level as our choice to drill in, and drill into the most significant instance, Greece in our example. After drilling into Greece we repeat the process again. We drill into the Time dimension to analyze if Greece presents any anomaly across quarters (step 3). Since the employment rate is still stable (even if it is failing), the IG is still low. Finally, we analyze the Geographical dimension, this time drilling into the level Region (step 4). As some regions are bigger than others, the algorithm ranks Central Greece as the region to drill in, even if West Macedonia presents a higher deviation from its goal.

After presenting the algorithm, we highlight two final considerations that affect its implementation. First, it can be the case that a dimension presents multiple aggregation paths, i.e. multiple hierarchies. In this case, the algorithm can be easily adapted simply by calculating the IG of each candidate level from a each hierarchy as if they were two separate dimensions. Then, once a hierarchy has been selected by the algorithm, the remaining hierarchies within the same dimension will be ignored, as it is not possible to restrict a KPI using multiple hierarchies from the same dimension simultaneously. Second, in order to reduce the number of queries posed to the engine, the queries can be optimized and the results of each restriction can be cached. As we will see in Section **??**, we can create a single query restricted to the subset of instances selected on each iteration, thus allowing us

to drill in parallel across all the relevant instances. As a result, we reduce the number of a queries to a single query per level analyzed.

A directed search approach using these criteria has several advantages. First, it generates a limited number of queries, $\Omega(d * l)$. Second it does not require a priori knowledge of the relevant dimensions and levels. Third, it allows the user to regulate the search, choosing whether the algorithm should stop earlier or drill deeper. Fourth, it allows us to report the most relevant instances found during the search in a hierarchic manner, thus, avoids flooding the user with information, as we will discuss in the next section.

3.5 Query Generator

In order to apply the previous algorithm, we require to generate a set of queries that retrieve the necessary data on each iteration. Query generation is the responsibility of the query generator module, starting from the query generated from the definition of the KPI and every subsequent query required during the search process. Due to implementation details, our approach generates the initial query in the MDX (MultiDimensional eXpressions) language (standard input for OLAP engines), whereas the subsequent queries are generated in SQL. However, this aspect does not alter the logic behind the process and it could be performed using either of the query languages on its own.

The initial query is generated from the KPI definition, following the approach presented in [15]. For example, given the KPI K defined as sum employed divided by sum active population representing employment rate, the following MDX query is generated (the set of unrestricted dimensions is not outputted unless requested):

```
WITH MEMBER [Measures].[result1] AS
SUM([Measures].[EmployedPopulation])
MEMBER [Measures].[result2] AS
SUM([Measures].[TotalPopulation])
MEMBER [Measures].[result3] AS
[Measures].[result1] /
[Measures].[result2]
SELECT [Measures].[result3] ON 0
FROM [Europe2020]
```

This initial MDX query is translated into SQL by the OLAP engine, and sliced according to the time period selected by the user for the monitoring process. As a result we obtain the initial SQL query for our monitoring process $Q_{K0} = \{M, D, C\}$, where M specifies a set of measures to be projected into cube cells, D contains the set of dimension levels that define the axis of the cube, and C specifies a set of constraints over the dimensions. Q_{K0} acts as the baseline query for the rest of the queries during the search process and, thus, all the following queries will only add or specialize levels in D and add additional constraints to C.

At each step i, we need to choose which dimension $d_j \in D$ we will drill in next. In order to make this choice, use given Q_{Ki-1}, we generate one IG evaluation query Q_{Kid} for each candidate dimension level l_{dj}, resulting in a query with $D_i = D_{i-1} \cup l_{dj}$. If D_{i-1} already includes another, more general level from d_j, then this level is substituted by the current more specific candidate l_{dj}. For example, given the previous query, the IG evaluation query generated for the region level within the country dimension is:

```
SELECT country.Region,
(SUM(EmployedPopulation) * 100 /
SUM(TotalPopulation)),
year.Year_cod
FROM growth,country,year WHERE
 country.Country_ID=growth.Country
  AND year.Year_ID=growth.Year
  AND year.Year_cod in (2012)
  GROUP BY country.Region,
  year.Year_cod
```

Then, Q_{Kid} is used to calculate the IG of the dimension d_j. Once we have selected the dimension providing the highest IG, a query Q_{Ki} is generated with $C_i = C_{i-1} \cup c$, where c constraints further search to the top-K instances contributing to the deviation at level l_{dj} (or the set of instances required for the specified coverage threshold).

Once we have obtained Q_i we start the process again until the stopping criteria is met. Following these steps, we have generated all the necessary queries needed for our search algorithm. Therefore, the only issue left to tackle is how to present the results obtained by the process in an understandable way.

3.6 Result Reporter

As a result of the process described so far, there are multiple findings that are relevant to the user.

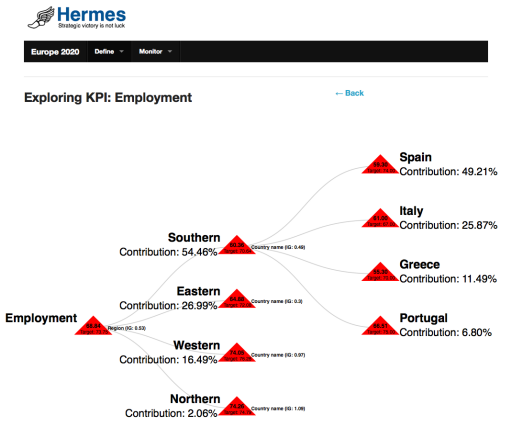

Figure 3: Most failing drill-down exploration

However, we need to visualize them in order to hel the user understand the nature of the problem.

During each step, the first set of elements that we identify are the top-K instances that contribute most to the deviation. It is noteworthy that these instances are the result of combining current and previous constraints on each iteration of the algorithm. For example, if we first drill into the Quarter level and then we drill into the country level for the first quarter, then, the instances will be Q1-Canada, Q1-Italy, and so on. Therefore, in order to provide detailed information about the nature of the deviation, we organize the instances in a hierarchical way. At each level we have to provide the name of the dimension d and the level l_d selected by the algorithm for drilling in, along with with the KPIs of the top-K instances, showing the relationships with their corresponding parent KPIs in the previous level. For each instance, we show the % of contribution to the global deviation as well as how much it is deviating from its particular target.

Additionally, we can also highlight the top-K instances outperforming their expected targets. These instances are represented by the deviating instances that present negative deviation values, as $Target_k < Actual_k$. This can help users to identify success factors that contribute to the achievement of their goals. A guided search that drills into these elements can easily be derived from the proposed algorithm, simply by selecting the tail-K contributors on each step.

Finally, some users may consider relevant to visualize the top-K instances that are deviating most from their own targets (even if they are not the ones

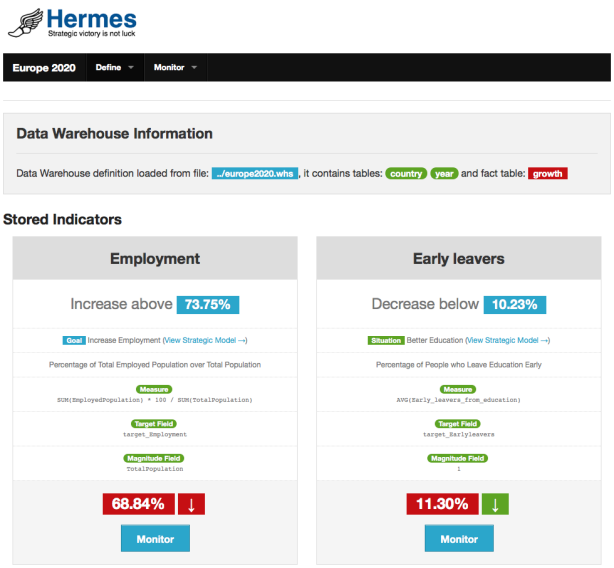

Figure 4: KPI definitions within the system

contributing most to the deviation). As in the previous case, this data can again be easily obtained simply by ordering the instances according to their normalized deviation instead of their absolute contribution during the drill down process. Extracting this information does not require introducing additional modifications to the algorithm.

The final result is an interactive tree that can be set to visualize different aspects of the data found during the search. Therefore, it allows users to analyze the most relevant findings according to the expected performance of the different instances stored in the data. Furthermore, it helps them understand where most of the deviation is located and, with the aid of the dimension levels and instance values, aids them in finding why. We can see an example of this visualization technique in figure 3.

Now that we have described the different parts involved in our approach, we proceed to show the details of our implementation and our case study in the following sections.

4 Implementation

We have implemented a web based prototype for monitoring KPIs as part of a Business Strategy.

It allows for the definition of KPIs which comprise of a measure definition (the measured quantity), a target column definition (the target values), and a market magnitude column definition (how the size of each sub-market is calculated). The magni-

tude definition allows us to calculate the weights for our algorithm, and when it is not defined, a count of the instances in each dimension is used as a measure of magnitude. Additionally, these KPIs can be attached to Goals or Situations in order to measure the degree to which they are happening. All the target values are inserted in the database at the lowest level of aggregation, and are then aggregated as averages for capturing the target of larger sub-markets. For example if we have defined per country targets, the target for a group of countries will be the weighted among all targets. Different types of aggregations for the targets can be defined by the user if they fit the KPI definition.

All of these definitions are given by the user as aggregation functions that are evaluated above a ROLAP schema. An example can be seen in the bottom part of Figure 4, where two KPIs are defined, one that measures the employment in Europe and another one that measures the early leavers from education. They contain a measure defined as a function evaluated above a snowflake ROLAP schema, a target column, and a magnitude column which are columns of the fact table. This schema is also defined as input in a setup step, where dimensions can be clustered by the user in sequential and non-sequential ones. Sequential dimensions, such as time, are used to measure how a KPI value fluctuates as they increase, for example trends can be measured along the time dimension, and forecasting models can applied in the resulting time-series. The system continuously generates such sequences at the lowest level of aggregation, and uses them to perform forecasting. The resulting forecasted values are then stored again in the database, only to be overwritten when the real values are available. Non-sequential dimensions on the other hand, can be used to drill in the most interesting sub-markets. The system automatically proposes a set of instances and dimensions to drill in based on our KPI monitoring algorithm.

The user can browse the data warehouse information, and monitor a set of KPIs (using the Monitor buttons shown in Figure 4). The KPI value in regards to the target is depicted in this screen, along with the current trend that it is following. By clicking the Monitor button, the user gets a detailed graphical break-down (Figure 3) of the results of the algorithm. Each sub-tree represents a dimension along which the analyst can drill in. The sub-trees are ordered based on their information gain, and the information gain of each dimension is depicted next to each node. In Figure 3, the first dimension that the algorithm chooses to explore (highest information gain) is the region dimension. It then expands the tree and presents the top-4 failing regions (Southern, Eastern, Western, Northern), in the next level of the tree for the Southern dimension for example, the country itself is chosen as a next drill operation. This gives us a next sub-tree with the countries Spain, Italy, Greece, Portugal, ranked by the degree to which they are failing to meet their corresponding targets (either aggregate targets, or individual targets).

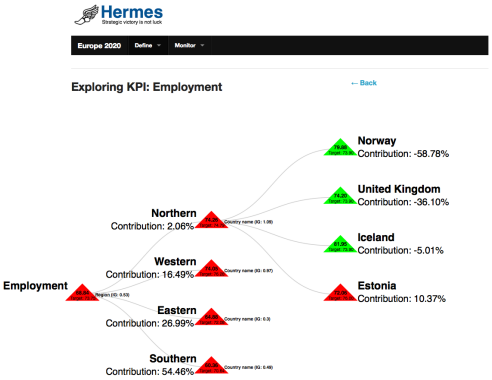

Figure 5: Most successful drill-down exploration

In the same way the user can also view the top most succeeding sub-markets and explore them intuitively (Figure 5), with green color representing success and red color representing failure to meet target. In this case the top-4 regions are Northern, Western, Eastern, Southern, while for the Northern region (best performing region) the top-4 performers are Norway, UK, Iceland and Estonia.

Finally, if the user is further interested to examine the trend of a sub-market she can get more information through a screen like the one in Figure 6. The overall trend of this sub-market is displayed along with the target values for every year.

4.1 Implementation Details

The backend of the system has been implemented in Python, while the user interface is making extensive use of JavaScript, HTML5 and AJAX. Moreover, a custom made KPI definition API has been developed in order to support our application. This API allows for the intuitive definition of complex KPIs and the automated generation of SQL queries, which can be directly evaluated on a Relational

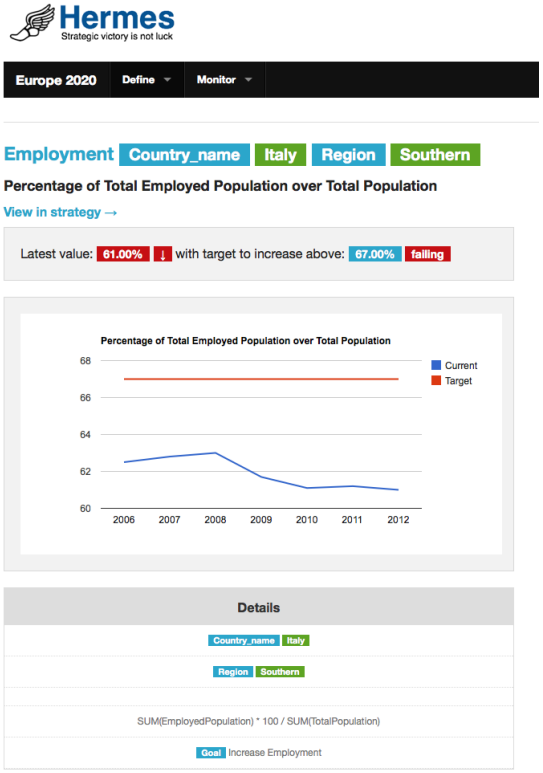

Figure 6: Trend examination

Database. For our experiments, we have used the MySQL database for storing the data, and the KPI monitoring API has been optimized to generate compatible queries.

5 Evaluation

For our experiments, we have implemented a data warehouse in MySQL with information about indicators for the European 2020 strategic planning framework, extracted from Eurostat. This scenario is an ideal candidate since it involves real data and we are aware of the conclusions reached by analysts after analyzing the data.

The data warehouse uses a snowflake schema and has two dimensions, Country and Year. First, Country dimension presents two hierarchies that combine three different levels: Country name → Region → All, and Country name → Euro → All. Second, Year dimension contains the actual year, specified as being a sequential dimension, and chosen for calculating all the trends.

Our fact table stores information including total population, employed population, GDP, and more, as well as target values for employment.

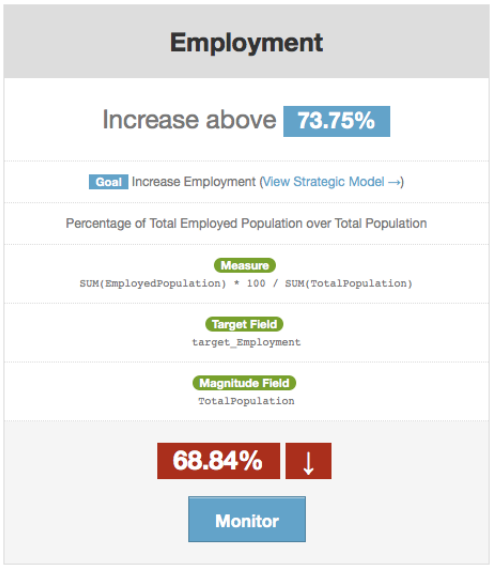

Figure 7: The employment KPI definition

According to this schema we have defined a KPI called Employment, as the ratio of the sum of the employed population over the sum of the total population. For the target value of this KPI, we defined an aggregation function that calculates the average per sub-market target values. Our raw target values were defined per year, per country. For calculating the aggregate target values, we summed the weighted sub-targets. In our case the query that was generated for projecting targets to higher-level sub-markets, weighted each country towards the total population of Europe and used this weight to calculate the target of each Region. In a recursive way all the targets were calculated from the root level to the overall Europe target. We chose the last year available in the database, 2012, as pivot for performing the analysis. This latest year can either come from forecasting, or using the latest values existing in the system.

In the general case any time point can be used for restricting the values in time and running the algorithm, resulting in the algorithm showing to the user why a target at the selected time point is failing. If a forecasted year is used, the user can identify why something "will" fail, if a past year is used a user can identify why something "has failed" and if the present time is used, the user can identify why something "is failing". The same analysis can also be done for successful KPIs for identifying the top-k most succeeding sub-markets. Finally, for calculating the weights we used the sum of the total

population of each sub-market for calculating the ratio over its immediate parent-market.

Considering this setup, and according to analysts, the algorithm should pinpoint the Region hierarchy as providing more interesting information than the Euro/non-Euro hierarchy, and rank the countries according to their contribution to the deviation of Europe 2020 employment.

5.1 Results

Our current tool implementation uses the recursive non-optimised version of the algorithm, and therefore, it generated 1 query for each instance to get its target value, 1 query to get its current value, and 1 query for getting the magnitude of each instance. However, the quick query processing speed allowed us to perform the process iteratively, as the user decided to drill down more. We display the instances of every dimension that is chosen for a drill in, along with the weighted contribution of each instance to the higher level failure.

We set the algorithm to recommend the top-4 reasons behind a failure or success. Indeed, our algorithm chose to split over the Region hierarchy as the most informative one, and subsequently ranked the most failing regions.

These regions were: Southern, Eastern, Western and Northern. The first region (Southern), was additionally split to its next hierarchy level (country name) and returned the top failing countries as follows: Spain, Italy, Greece and Portugal (shown in Figure 3. The same split was performed showing the corresponding countries in other regions.

Finally, we also instructed the algorithm to return the top-4 most successful sub-markets. The algorithm again chose the Region dimension (as we expected since it maximized the information gain) and ranked the regions in the reverse order: Northern, Western, Eastern and Southern. For the Northern region (highest performing), the top countries were: Norway, UK, Iceland and Estonia.

As we can see, positive contributions to the deviation mean a failure, while negative contributions mean that the target is succeeding, and as a result is compensating for the total loss. This is a very relevant aspect for the user, because there can exist instances that are deviating more than the total deviation, but are being compensated by other instances. The output screen for the top-4 worst performers can be seen in Figure 3 and the screen for the top-4 best performers can be seen in Figure 5.

6 Conclusions and Future work

Current monitoring techniques lack the detail and simplicity required by users to be able to pinpoint problems in their business strategies and take corrective actions. In this paper we have presented a systematic approach that takes into account users' goals and targets to be met, finds the main elements threatening or contributing to them, and presents this information back to the user in a simple way. Our approach performs a dynamic, guided search of the data space, drilling deeper into the levels that provide more information in order to pinpoint the problem and stopping when the size of the problem becomes irrelevant for the user. Additionally, we have discussed how this criteria can provide different information to the user. Furthermore, we have shown how query generation can be automated in order to perform the search without user intervention. Finally, we have presented how our approach is implemented in our CASE tool and have applied it to a case study using real data.

The lack of a priori knowledge required in our approach, as well as the possibility of automating query generation, allow the user to identify significant findings with few mouse clicks. Compared to our approach, traditional monitoring would require tedious OLAP navigation, chart elaboration, and analysis, without ensuring that the effort is well-directed. Furthermore, as we focus on the biggest contributors to the deviation of each target, this allows the user to get a better understanding of the reasons behind the deviation and how corrective actions could be taken.

Nevertheless there is still much work to be done in the area. Although in this paper we have explored an information measure to successfully guide the search process, other information measures could be defined in order to look for different aspects in the data. Additionally, Targeted monitoring can only be applied as long as we have available a clear target criteria. In other cases, such as when monitoring changes in trends, we require untargeted monitoring techniques. Furthermore, business strategy models capture factors that affect the achievement of business goals and could be used to improve the monitoring process. Finally, all the efficiency of these monitoring techniques needs to be tested on very large datasets, such as those stored

in the data warehouses of international companies. To this aim, our future work is focused on (i) evaluating and improving the effectiveness of the algorithm presented, and (ii) defining and evaluating the applicability of a monitoring process that empowers current SWOT (Strengths, Weaknesses, Opportunities, and Threats) analysis making use of the data available.

Acknowledgements

This research was partially funded by the FP7 EU ERC Advanced Investigator project Lucretius (grant agreement no. 267856), and developed in the context of the project GEODAS-BI (TIN2012-37493-C03-03) funded by the Ministry of Education and the Ministry Economy and Competitiveness of Spain (MINECO). Alejandro Maté is funded by the Generalitat Valenciana under the postdoctoral grant APOSTD/2014/064.

About the Author

Alejandro Maté is postdoc researcher at the University of Trento, Italy. He received a BS and a MSc Degree in Computer Science from the University of Alicante. He earned his PhD degree in Computer Science from the University of Alicante in 2013. He has published several papers in international conferences and journals such as CAiSE, ER, RE, JSS, and IS. His research involves conceptual modeling, data warehouses, model driven development, and requirements engineering.

Kostas Zoumpatianos is a PhD student at the dbTrento group, University of Trento, Italy. His research involves data warehouses, business intelligence and data series management. He holds a MSc Degree in Information Management and a BSc degree in Information and Communication Systems Engineering, both from the University of the Aegean, Greece.

Themis Palpanas Themis Palpanas is a professor of computer science at the Paris Descartes University (France). He received a BS degree from the National Technical University of Athens (Greece), and the MSc and PhD degrees from the University of Toronto (Canada). He has worked at the University of California at Riverside (USA), the IBM T.J. Watson Research Center (USA), and the University of Trento (Italy). His solutions have been implemented in world leading data management products and he is the author of eight US patents. He is the recipient of three Best Paper awards, and founding member of the Event Processing Technical Society. He has been General Chair for VLDB 2013.

Juan Trujillo is a Full-time Professor at the University of Alicante (Spain) and leader of the Lucentia Research Group. His main research topics include BI applications, Big Data, KPIs, data warehouses, UML, and MDA. He has also registered several tools related to data warehouse modelling. He has advised 11 PhD students and published more than 150 papers in highly impact conferences such as ER, UML or CAiSE, and 38 papers in JCR indexed jounals such as the DKE, DSS, ISOFT, IS, or JDM. He has also been co-editor of nine special issues in different JCR journals (e.g. DKE). He has also been PC member of different events and journals such as ER, DOLAP, DSS, JDM, or DKE, and PC Chair of DOLAP'05, DAWAK'05-'06, FP-UML'05-'09, MoDIC'12 and ER'13.

John Mylopoulos holds a distinguished professor position (chiara fama) at the University of Trento, and a professor emeritus position at the University of Toronto. He earned a PhD degree from Princeton University in 1970 and joined the Department of Computer Science at the University of Toronto that year. His research interests include conceptual modeling, requirements engineering, data semantics and knowledge management. Mylopoulos is a fellow of the Association for the Advancement of Artificial Intelligence (AAAI) and the Royal Society of Canada (Academy of Sciences). He has served as general chair of international conferences in Artificial Intelligence, Databases and Software Engineering.

References

[1] Y Dora Cai, David Clutter, Greg Pape, Jiawei Han, Michael Welge, and Loretta Auvil. MAIDS: mining alarming incidents from data streams. In *SIGMOD*. ACM, 2004.

[2] Alessandro Camerra, Jin Shieh, Themis Palpanas, Thanawin Rakthanmanon, and Eamonn J. Keogh. Beyond one billion time series: indexing and mining very large time series collections with i sax2+. *Knowl. Inf. Syst.*, 39(1):123–151, 2014.

[3] Chris Chatfield. *Time-series forecasting*. CRC Press, 2002.

[4] Yixin Chen, Guozhu Dong, Jiawei Han, Benjamin W Wah, and Jianyong Wang. Multi-dimensional regression analysis of time-series data streams. *VLDB*, 2002.

[5] Edgar F Codd, Sharon B Codd, and Clynch T Salley. *Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate*, volume 32. E. F. Codd & Associates, 1993.

[6] Michele Dallachiesa, Besmira Nushi, Katsiaryna Mirylenka, and Themis Palpanas. Uncertain time-series similarity: Return to the basics. *PVLDB*, 5(11):1662–1673, 2012.

[7] Michele Dallachiesa, Themis Palpanas, and Ihab F. Ilyas. Top-k nearest neighbor search in uncertain data series. *PVLDB*, 8(1), 2015.

[8] Wayne W Eckerson. *Performance dashboards: measuring, monitoring, and managing your business*. Wiley, 2010.

[9] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov.*, 1(1):29–53, January 1997.

[10] Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W Wah, Jianyong Wang, and Y Dora Cai. Stream Cube: An Architecture for Multi-Dimensional Analysis of Data Streams. *Distributed and Parallel Databases*, 18(2):173–197, 2005.

[11] Jennifer Horkoff, Daniele Barone, Lei Jiang, Eric Yu, Daniel Amyot, Alex Borgida, and John Mylopoulos. Strategic business modeling: representation and reasoning. *Software & Systems Modeling*, pages 1–27, 2012.

[12] Robert S. Kaplan and David P. Norton. *Strategy maps: Converting intangible assets into tangible outcomes*. Harvard Business Press, 2004.

[13] Robert S. Kaplan, David P. Norton, RC Dorf, and M Raitanen. *The balanced scorecard: translating strategy into action*, volume 4. Harvard Business school press Boston, 1996.

[14] Gert HN Laursen and Jesper Thorlund. *Business analytics for managers: Taking business intelligence beyond reporting*, volume 40. John Wiley & Sons, 2010.

[15] Alejandro Maté, Juan Trujillo, and John Mylopoulos. Conceptualizing and specifying key performance indicators in business strategy models. In *CASCON*. IBM Corp., 2012.

[16] Object Management Group (OMG). Business motivation model 1.2, 2014.

[17] Themistoklis Palpanas and Nick Koudas. Entropy based approximate querying and exploration of datacubes. *SSDBM*, 2001.

[18] David Parmenter. *Key performance indicators (KPI): developing, implementing, and using winning KPIs*. Wiley, 2010.

[19] Sunita Sarawagi. Explaining differences in multidimensional aggregates. *VLDB*, 1999.

[20] Sunita Sarawagi. User-adaptive exploration of multidimensional data. *VLDB*, 2000.

[21] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-Driven Exploration of OLAP Data Cubes. *Advances in Database Technology*, 1377, 1998.

[22] Eric Yu, Jennifer Horkoff, Daniel Amyot, Greg Richards, and John Mylopoulos. Business modeling for business intelligence. *Synthesis Lectures on Data Management*, 5(1):19–32, 2013.

[23] Konstantinos Zoumpatianos, Themis Palpanas, and John Mylopoulos. Strategic management for real-time business intelligence. In *BIRTE*, 2012.

[24] Konstantinos Zoumpatianos, Themis Palpanas, John Mylopoulos, Alejandro Maté, and Juan Trujillo. Monitoring and diagnosing indicators for business analytics. In *CASCON*. IBM Corp., 2013.

[25] Kostas Zoumpatianos, Stratos Idreos, and Themis Palpanas. Indexing for interactive exploration of big data series. In *SIGMOD*, 2014.