Esteban Zimányi (Ed.)

Tutorial
LNBIP 172

Business Intelligence

Third European Summer School, eBISS 2013
Dagstuhl Castle, Germany, July 7–12, 2013
Tutorial Lectures

Springer

Lecture Notes in Business Information Processing 172

For further volumes:
http://www.springer.com/series/7911

Esteban Zimányi (Ed.)

Business Intelligence

Third European Summer School, eBISS 2013 Dagstuhl Castle, Germany, July 7–12, 2013 Tutorial Lectures

Springer

Editor
Esteban Zimányi
Universite Libre de Bruxelles
Brussels
Belgium

ISSN 1865-1348 ISSN 1865-1356 (electronic)
ISBN 978-3-319-05460-5 ISBN 978-3-319-05461-2 (eBook)
DOI 10.1007/978-3-319-05461-2
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014934656

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The Third European Business Intelligence Summer School (eBISS 2013) took place in the Dagstuhl Schloss, Wadern, Germany, during July 2013. Tutorials were given by renowned experts and covered several recent topics in business intelligence. This volume contains the lecture notes of the summer school.

The first chapter presents an overview of pattern mining techniques for extracting knowledge from large databases. Two main strategies for mining frequent itemsets are discussed, namely, the Apriori and the FP Growth algorithms. The chapter then delves into the pattern explosion problem and presents some recent techniques to reduce the redundancy in pattern collections. These techniques use, on the one hand, statistical methods to model user expectations given background knowledge, and on the other, the minimal description length principle.

The second chapter introduces process mining, a new scientific discipline on the interface between process models and event data. Process mining aims at bridging the gap between business process management and data mining. The challenge is to turn huge amounts of event data into valuable insights related to process performance and compliance. The chapter introduces basic process mining techniques that can be used for process discovery and conformance checking. Then, the chapter discusses decomposition techniques, which enable process mining in the large.

The third chapter presents an ontology-driven business intelligence approach for comparative data analysis. This approach has been developed in a joint research project that involves academia, industry, and prospective users from public health insurers. This approach employs techniques from knowledge-based systems, ontology engineering, and data warehousing in order to support business analysts in their analysis tasks.

The fourth chapter explores how to integrate traditional business intelligence systems with the Linked Open Data paradigm. This paradigm enables the sharing of freely available data on the Web through the use of open standards and formalisms, such as RDF and ontology languages. Business intelligence systems must meet new requirements for integrating the Linked Open Data paradigm. This includes, in particular, to provide on-demand analysis tasks over any relevant data source in right-time. This chapter discusses the technical challenges behind such requirements, and describes a new kind of business intelligence system to support this scenario.

The fifth chapter presents an overview of forecasting techniques in database management systems. Time series forecasting estimates future, not yet available, data of a time series. After discussing possible application areas for time series forecasting, the chapter outlines various general strategies of integrating time series forecasting inside a database and discusses some individual techniques from the database community. The chapter concludes by introducing a novel forecasting-enabled database management architecture that natively and transparently integrates forecast models.

The sixth chapter addresses the issue of optimizing analytical queries by means of indexes. The chapter starts with an overview of the basic index structures for data warehouses, namely, bitmap indexes, join indexes, and bitmap join indexes. Then, the chapter presents a new index, called Time-HOBI, which can be used for optimizing queries that address the time dimension and compute aggregates along dimension hierarchies. Furthermore, the paper shows how the index can be used for answering queries, and presents experimental results about the performance of the proposed index.

Finally, the seventh chapter presents a novel extension to TARGIT's patented meta-morphing called "The Intelligent Wizard". After presenting the relevant state-of-the-art, the chapter describes the Intelligent Wizard as implemented in a real-world industrial Business Intelligence (BI) application. The paper shows how the Intelligent Wizard allows a user to navigate a real-world data warehouse using only human language and knowledge of business terms, thus significantly simplifying the generation of analytics and reports.

We would like to thank the attendees of the summer school for their active participation, as well as the speakers and their co-authors for the high quality of their contribution in a constant evolving and highly competitive domain. Finally, the lectures in this volume greatly benefit from the comments of the external reviewers.

January 2014 Esteban Zimányi

Organization

The Third European Business Intelligence Summer School (eBISS 2013) was organized by the Department of Computer and Decision Engineering (CoDE) of the Université Libre de Bruxelles.

Program Commitee

Alberto Abelló	Universitat Politécnica de Catalunya, BarcelonaTech, Spain
Marie-Aude Aufaure	Ecole Centrale de Paris, France
Ralf-Detlef Kutsche	Technische Universität Berlin, Germany
Patrick Marcel	Université François Rabelais de Tours, France
Esteban Zimányi	Université Libre de Bruxelles, Belgique

External Referees

Rafael Berlanga	Universitat Jaume I, Castellón, Spain
Stefan Conrad	Heinrich-Heine-Universität, Germany
Benoit Depaire	Hasselt University, Belgium
Cristina Dutra de Aguiar	University of São Paulo at São Carlos, Brazil
Matteo Golfarelli	University of Bologna, Italy
Katja Hose	Aalborg University, Denmark
Mykola Pechenizkiy	Technical University Eindhoven, The Netherlands
Kai-Uwe Sattler	Technische Universität Ilmenau, Germany
Jim Warren	The University of Auckland, New Zealand
Hans Weigand	Tilburg University, The Netherlands

Contents

Introduction to Pattern Mining

Toon Calders(✉)

Université Libre de Bruxelles, Bruxelles, Belgium
tcalders@ulb.ac.be

Abstract. We present an overview of data mining techniques for extracting knowledge from large databases with a special emphasis on the unsupervised technique pattern mining. Pattern mining is often defined as the automatic search for interesting patterns and regularities in large databases. In practise this definition most often comes down to listing all patterns that exceed a user-defined threshold for a fixed interestingness measure. The simplest such problem is that of listing all frequent itemsets: given a database of sets, called transactions, list all sets of items that are subset of at least a given number of the transactions. We revisit the two main strategies for mining all frequent itemsets: the breadth-first Apriori algorithm and the depth-first FPGrowth, after which we show what are the main issues when extending to more complex patterns such as listing all frequent subsequences or subgraphs. In the second part of the paper we then look into the pattern explosion problem. Due to redundancy among patterns, most often the list of all patterns satisfying the frequency thresholds is so large that post-processing is required to extract useful information from them. We give an overview of some recent techniques to reduce the redundancy in pattern collections using statistical methods to model the expectation of a user given background knowledge on the one hand, and the minimal description length principle on the other.

1 Introduction

In the last decades the amount of information that is generated and stored has increased exponentially. Illustrative for the magnitude of the overload problem is the quote of Eric Schmidt, CEO of Google inc. on the data explosion problem: "There was 5 exabytes of information created between the dawn of civilization through 2003, but that much information is now created every 2 days, and the pace is increasing." His claim is supported by other researchers: "As computational resources, sensor networks and other large-scale instruments and experiments grow, the quantity of data generated from these sources is also growing. A 2010 study by IDC (sponsored by data storage company EMC) estimates that the world generated 800,000 petabytes of digital information in 2009, and that we are on track to generate 1.2 million petabytes (or 1.2 zettabytes) in 2010." [26]. Companies all over the world are becoming more and more aware that the huge amount of data they collect about their operations, customers, products,

E. Zimányi (Ed.): eBISS 2013, LNBIP 172, pp. 1–32, 2014.
DOI: 10.1007/978-3-319-05461-2_1,

suppliers, and so on, can help them better understand their business. The ability to analyze and make sense out of the data can be of strategic importance and give a competitive advantage.

For instance, insurance companies can increase their market share by offering more competitive rates and banks decrease their losses due to loan defaulting if they are able to better understand the risk profiles of their customers, telephone companies can decrease customer churn and service providers could offer more personalized services if they better understand their customers' needs. Also in the public domain there are numerous examples of benefits that data can bring: based on nation-wide statistics we may be able to identify factors that are correlated to kids dropping out of schools, sensory data of thousands of patients may uncover hidden correlations that enable less costly or earlier recognition of certain diseases, and the systematic analysis of police reports may lead to the detection of criminality patterns that can be exploited to fight crime. These are just a few promises held by the shear amount of data gathered nowadays.

In the whole process of making sense out of the available data, we identify several steps: (1) data needs to be collected. (2) Collected data needs to be stored, for instance in data warehouses, in a format that allows for efficient analysis. (3) For the analysis itself efficient techniques such as Online Analytical Processing (OLAP), and data mining need to be present in order to turn the enormous amounts of data into predictive models, trends, regularities. (4) Finally, based on the models generated by data mining, business decisions need to be taken.

For instance, in the example of the insurance company intending to improve the risk assessment, data gathering implies that the company needs to carefully record the multiple factors that can be used to assess the risk of a customer. Furthermore, for every existing customer, the insurance claims and the awarded insurance amounts need to be recorded. All this information needs to be centralized and stored in a data warehouse. For this purpose it is important that the company uses uniform processes and records all relevant business data. This data needs to be cleaned and stored in a way that allows analysis to be performed on the data, such as in a data warehouse. Then, based on the collected business data, models can be built correlating the features of the customers with the actual risk they presented as computed from the insurance claims. These models could then be used to assess the risk of new clients that come to the company. Based on the company strategy, the risk models could for instance be used to target and attract specific groups with low risk profiles by offering more competitive rates, or to deny insurance to high-risk customers. The ability to collect and analyze huge amounts of data hence helps the company to better understand and recognize risk factors leading to a competitive advantage.

In this article we will concentrate on a specific set of techniques for analyzing data, called *data mining*. Han and Kamber [39] define data mining as " . . . the use of sophisticated data analysis tools to discover previously unknown, valid patterns and relationships in large data sets." Most data mining methods can be divided into one of the following three categories.

1. *Classification and regression* [39, Ch. 6] [68, Ch. 4]: Predicting respectively a categorical or a numerical target based on some input attributes. Models are inferred from labelled training data.
2. *Clustering* [39, Ch. 7] [68, Ch. 8]: Dividing a given dataset into logically coherent groups. In contrast to classification, no training set with labels is given, but the algorithm is supposed to find a logical division of the data into clusters.
3. *Pattern Mining* [39, Ch. 5] [68, Ch. 6]: Discovering regularities, trends, or patterns in large datasets. This includes, among others, finding associations in transactional data [2] and sequences [55], frequent subgraphs in graph data [1], and spatio-temporal patterns describing for instance trends in mobility data [30].

Next to these three main techniques we also identify *outlier detection* [68, Chap. 10]; i.e., finding data instances that are "different" as compared to the other instances, *semi-supervised clustering* [81]; i.e., clustering in which some instances have already be divided into clusters, and *active learning* [64], where we need to learn classes with only few labeled instances, but with an interactive user that can be asked to label some instances. Outlier detection can be seen as a by-product of clustering, although some fully-fledged solutions exits that do not require building a model for the complete dataset. Semi-supervised clustering is at the intersection of clustering and classification, whereas active learning can be considered as a particular approach towards classification.

Clustering and pattern mining are often referred to as *unsupervised* methods as they require only data, whereas supervised methods, such as classification, require annotated or labeled data. In this paper we will mainly concentrate on the unsupervised pattern mining techniques. Pattern mining has several interesting applications, including:

1. The patterns themselves can provide interesting information for the data miner; similarly as visualization tools, the output itself of the data mining operation might reveal interesting patterns to the user that would otherwise remain unnoticed due to the high dimensionality of the dataset. For instance, using pattern mining it may be observed that a certain combination of genes are always over-expressed together under certain conditions. Clearly, iterating manually over all possible combinations is impossible and therefore pattern mining techniques can be very useful to highlight strong patterns.
2. Patterns are often used as input for other data mining operations. For instance, frequent patterns could be used as features in classification tasks. Often classification algorithms require tabular data as input. It may, however, not always be straightforward to transform graphs, sequences, strings, sets of items, etc. into a tabular format. Pattern mining can be particularly helpful in such applications to identify potentially interesting features. For instance, in graph data, a frequently occurring subgraph pattern may be a good candidate to generate a feature that represents presence or absence of this particular structure.
3. Also for clustering frequent patterns may be useful as input. Frequent patterns represent common behavior encountered in the dataset and as such define

Pattern	Support	Pattern	Support
algorithm algorithm	0.376	method method	0.250
learn learn	0.362	algorithm result	0.247
learn algorithm	0.356	Data set	0.244
algorithm learn	0.288	learn learn learn	0.241
data data	0.284	learn problem	0.239
learn data	0.263	learn method	0.229
model model	0.260	algorithm data	0.229
problem problem	0.258	learn set	0.228
learn result	0.255	problem learn	0.227
problem algorithm	0.251	algorithm algorithm algorithm	0.222

Fig. 1. The subsequences with the highest frequency in the JLMR dataset. The patterns have already been filtered to only include the so-called closed sequences.

a subgroup of the data. By combining the information given by multiple frequent patterns, a similarity measure between instances can be constructed based on how many patterns they share.

Most of the earlier work on pattern mining focussed on algorithmic techniques for quickly finding all patterns that satisfy a certain frequency criterium. Therefore, we will first give an overview of the main algorithmic techniques for mining frequent itemsets, the simplest of all patterns. We present the well-known breadth-first Apriori algorithm [3] and the main characteristics of depth-first algorithms such as FPGrowth [40] and Eclat [80]. All algorithms rely heavily on the monotonicity principle of support; that is, the observation that enlarging a pattern cannot make it more frequent. This important property of a good interestingness measure is the cornerstone of all frequent pattern mining algorithms.

The methods aimed at finding all frequent patterns, however, suffer greatly from the so-called *pattern explosion problem*: when the threshold on the interestingness measure is set too low, this may result in an enormous amount of often uninteresting and redundant patterns, while setting the threshold too high may cause important patterns to be missed. Therefore, in the second part of the paper we will concentrate on the pattern explosion problem and propose some solutions to reduce or even avoid the problem. The first solution is the so-called *Condensed Representation* [18]. A condensed representation of the collection of patterns is a subset of all patterns that still contains enough information to reproduce the other patterns and their frequency information. As such they represent a lossless subset of the complete pattern set.

In many cases, however, even the number of patterns in the condensed representation is still too large. For instance, consider Fig. 1 from [49]; in this figure the most frequent *closed* subsequences in the JLMR dataset are shown. The JLMR dataset is a modestly-sized dataset containing 787 abstracts of papers that appeared in the Journal of Machine Learning and Research. These patterns have already been filtered to only include the so-called *closed patterns*, one of the best known condensed representations. Even though the collection of terms

is already reduced with this filtering step, we still see an extremely high redundancy in the collection: many patterns with very similar meaning are shown to the user. Beside redundancy issues, the set of frequent patterns also contains trivial and meaningless patterns. For example, the set of frequent closed sequential patterns in Fig. 1 contains random combinations or repetitions of frequent terms in the JMLR abstracts such as *algorithm, result, learn, data* and *problem*.

Because the condensed representation approach does not completely remove all redundancy, the second solution will be based on surprisingness of patterns given *statistical models of expectation* and the *Minimal Description Length*. These methods have as property that they discourage overlapping and uninformative patterns. For the statistical model, redundant patterns are avoided by removing "non-surprising" patterns, where non-surprising depends on a statistical model built for the expectation of the patterns. The minimal-description length principle, on the other hand, avoids overlapping patterns by selecting them based on their ability to compress the database. Highly overlapping patterns will be poor at compressing the database as they tend to target the same parts.

The structure of the paper is as follows: in the first part of the paper (Sect. 2 till 5) we will concentrate on the traditional frequent pattern mining problem; that is, techniques for listing all patterns that satisfy a certain minimal support threshold: Sect. 2 defines the frequent itemset mining problem, and Sect. 3 sketches the main algorithmic techniques. Section 4 shows several extensions of the frequent itemset mining problem to more complex pattern types and Sect. 5 presents applications. The second part of the paper, Sect. 6 will then concentrate on the pattern explosion problem and solutions for it based on statistical methods and the minimal description length. Section 7 concludes the paper.

2 Pattern Mining: Definition

We will start with an illustrative example of frequent pattern mining that will serve as a running example to illustrate the concepts throughout the paper. Figure 2 contains a part of a much larger dataset of 3.5 million sets of tags assigned by users of the photo website www.flickr.com to their pictures[1]. When confronted with this dataset, one may wonder if there are certain subsets of tags that frequently co-occur in the dataset. Such frequent groups may represent important concepts among the pictures. Unfortunately, the enormous amount of pictures and tags makes it very difficult to manually identify the frequent sets of tags in the dataset. This task is exactly what frequent pattern mining tackles.

In frequent itemset mining [2], we assume that a database of sets, called *transactions*, is given. The task is to find all sets of items that are subset of at least a user-given number of transactions. This number of transactions of which the itemset is a subset is called the *support*, and the user-given threshold, the *minimal support*. That is, let $\mathcal{I}$ be a set of items. A transaction T is a pair

[1] A dataset with 3.5 million tag sets is made available by Xirong Li and can be accessed on his website http://staff.science.uva.nl/~xirong/index.php?n=DataSet.Flickr3m

```
1000376015  7928333@N03      SHIP FERRY SEA BOAT BLUE
1007199160  56445078@N00     Black and White Australia
1011677092  10730206@N02     Emergency Vehicle Show Boat
1028308843  80642895@N00     hawaii tender boat ship ocean cruise
1031991872  44124297297@N01  Greece Mykonos cruise ship blue boat lens
                             flare Louis
120120111   25203555@N00     2006 police sheriff communications
                             dispatch vehicle truck emergency
1212917111  21651009@N00     sydney harbour harbour bridge opera house
                             long exposure circular quay train station
                             night ferry
1229907742  79038663@N00     Thomas Aylett Australia Sydney Harbour
                             Bridge Canon 30D 70 200 themoulinrouge
                             Canon EF 70 200mm f 2 8L IS USM 70 200L
1263045627  47054297@N00     Police COP Emergency Vehicle PD
1306918395  10615477@N04     Military War Heavy Duty Police Truck
                             Emergency Response Auto Vehicle
```

Fig. 2. Tag sets assigned by users of the Flickr website. The first and second column represents respectively the picture and user identifier, followed by the tags assigned by the user. Every line represents a tag set of a real photo, but for educational purposes a convenient subset of photos was selected and slightly modified.

(TID, J) with TID the transaction *identifier* and J a subset of $\mathcal{I}$. A transaction database is a set of transactions. The support of an itemset $I \subseteq \mathcal{I}$ in a transaction database $\mathcal{D}$ is defined as:

$$support(I, \mathcal{D}) \ := \ |\{(TID, J) \in \mathcal{D} \mid I \subseteq J\}| \ .$$

Given a minimal threshold *minsup*, the outcome of the frequent itemset mining problem is the following set of *frequent patterns*:

$$\mathcal{F}(\mathcal{D}, minsup) \ := \ \{I \subseteq \mathcal{I} \mid support(I, \mathcal{D}) \geq minsup\}$$

Hence, if we want to apply itemset mining to the data in Fig. 2, we first need to transform the data into transactions. We can do that by grouping the tags by photo, or by user. If we group by photo, we get the transaction database depicted in Fig. 3. To reduce the total number of items we removed all tags that appear in less than 2 photos in the example. Every line represents one transaction and consists of the set of words that appear on that line.

For a threshold of 3, the frequent itemsets have been listed in Fig. 4. For example, the support of {emergency,vehicle} is 4 as there are 4 transactions (3, 6, 9, 10) in which both items of this itemset appear. As the support of the itemset {cruise, ship} is 2 (there are exactly two transactions—the 4th and 5th—in which both items appear), this itemset is not in the list of frequent itemsets.

In its original formulation [2], the frequent itemset mining problem was only part of the larger association rule mining problem. In the association rule mining problem, itemsets are combined into rules of the form $X \Rightarrow Y$, indicating that

TID	Set of items
1	{blue, boat, ferry, ship}
2	{australia}
3	{boat, emergency, vehicle}
4	{boat, cruise, ship}
5	{blue, boat, cruise, ship}
6	{emergency, police, truck, vehicle}
7	{bridge, ferry, harbour, sydney}
8	{australia, bridge, harbour, sydney}
9	{emergency, police, vehicle}
10	{emergency, police, truck, vehicle}

Fig. 3. Transaction database constructed for the data in Fig. 2. Every photo became a transaction and the de-duplicated tags the items. Tags were transformed to lower case and for readability of our running example only tags that appear in 2 or more photos were kept.

Frequent Itemset	support
{}	10
{ship}	3
{boat}	4
{vehicle}	4
{emergency}	4
{police}	3
{boat, ship}	3
{emergency, vehicle}	4
{police, vehicle}	3
{emergency, police}	3
{emergency, police, vehicle}	3

Fig. 4. Frequent itemsets in the database given in Fig. 3 for a minimal frequency threshold of 3

X in a transaction implies the presence of Y as well. The quality of such rules is measured using *support* and *confidence* of the rule. These measures are defined as follows:

$$\begin{aligned} \mathit{support}(X \Rightarrow Y, \mathcal{D}) &:= \mathit{support}(X \cup Y, \mathcal{D}) \\ \mathit{confidence}(X \Rightarrow Y, \mathcal{D}) &:= \frac{\mathit{support}(X \cup Y, \mathcal{D})}{\mathit{support}(X, \mathcal{D})} \end{aligned}$$

$\mathcal{D}$ is omitted when the database is clear from the context. Hence, the support of the rule corresponds to the number of transactions that support the rule in the sense that both antecedent and consequent, while the confidence estimates the conditional probability of observing the consequent in a transaction given that the antecedent is present. Another popular rule quality measure is the *lift* of a

rule. The lift of rule $X \Rightarrow Y$ in a database $\mathcal{D}$ is defined as follows:

$$lift(X \Rightarrow Y, \mathcal{D}) := |\mathcal{D}| \frac{confidence(X \Rightarrow Y, \mathcal{D})}{support(Y, \mathcal{D})}$$

Lift hence expresses how much more probably it is that a transaction containing itemset X contains itemset Y; indeed: $\frac{support(Y,\mathcal{D})}{|\mathcal{D}|}$ is the probability that a randomly selected transaction from $\mathcal{D}$ contains itemset Y, whereas $confidence(X \Rightarrow Y, \mathcal{D})$ is the probability that a randomly selected itemset that contains itemset X, also contains itemset Y. Lift is especially interesting when comparing rules with different consequents. For instance, even though the rule *drinker* $\Rightarrow$ *brown hear* may have a higher confidence than the rule *drinking* $\Rightarrow$ *liver problems*, the latter one is more interesting as there are far less people with liver problems than there are people with brown hear.

From a computational perspective, however, the step from itemsets to association rules is not significant; almost all association rule mining algorithms start by mining frequent itemsets which, in a second phase, are split in every possible way into antecedent and consequent to form rules. The main computational bottleneck in this combined operation is heavily skewed towards the computation of all frequent itemsets. Furthermore, association rules are often considered to be misleading since they seem to imply causal dependencies between the head and the consequent of the rule. It is however impossible to derive such causal relationships from an observational dataset. Therefore, in this chapter we concentrate on the frequent itemset mining problem.

3 Algorithms for Mining Frequent Itemsets

All algorithms for mining frequent itemsets rely heavily on the following *monotonicity principle*:

if $X \subseteq Y$, then $support(X) \geq support(Y)$

That is, adding items to a set will never increase its support. The monotonicity principle can be used to prune large parts of the search space; whenever we encounter an infrequent itemset X, there is no need to explore any of its supersets, as they have to be infrequent as well due to the monotonicity principle. This principle plays a key role both in breadth-first and depth-first algorithms.

3.1 Breadth-First Algorithm Apriori

The Apriori algorithm [3], depicted in Algorithm 1, maximizes the pruning capability of the monotonicity principle by traversing the search-space in a breadth-first fashion. It first starts with counting the singleton itemsets in a single scan over the dataset, then it counts all candidates of length 2 in one scan, and so on. When generating the candidates of length k to be counted in the kth step, only those itemsets are considered of which all subsets were observed to be frequent

Algorithm 1. Apriori

Input: Database $\mathcal{D}$, *minsup*
Output: Set $\mathcal{F}$ of frequent itemsets

$k \leftarrow 1$
$\mathcal{C}_1 \leftarrow \{\{i\} \mid i \text{ item occurring in } \mathcal{D}\}$
while $\mathcal{C}_k \neq \{\}$ **do**
  $\mathcal{F}_k \leftarrow \{I \in \mathcal{C}_k \mid I \text{ is frequent}\}$; ▷ Count itemsets in $\mathcal{C}_k$ in 1 scan over $\mathcal{D}$;
  $\mathcal{C}_{k+1} \leftarrow \{I \mid |I| = k+1 \text{ and all } J \subset I \text{ with } |J| = k \text{ are in } \mathcal{F}_k\}$;
  ▷ Generate new candidates
  $k \leftarrow k+1$
end while
$\mathcal{F} \leftarrow \bigcup_{i=1\ldots k-1} \mathcal{F}_i$;
return $\mathcal{F}$

Step $k = 1$

Itemset	Support
{boat}	4
{emergency}	4
{police}	3
{ship}	3
{vehicle}	4
{australia}	2
{blue}	2
{bridge}	2
{cruise}	2
{ferry}	2
{harbour}	2
{sydney}	2
{truck}	2

$\longrightarrow$

Step $k = 2$

Itemset	Support
{boat, ship}	3
{emergency, police}	3
{emergency, vehicle}	4
{police, vehicle}	3
{boat, emergency}	1
{boat, vehicle}	1
{boat, police}	0
{emergency, ship}	0
{ship, vehicle}	0
{police, ship}	0

$\longrightarrow$

Step $k = 3$

Itemset	Support
{emergency, police, vehicle}	3

Fig. 5. Candidate itemsets generated by the Apriori-algorithm together with their support. The minimal support threshold is 3

in the previous iterations. In Fig. 5 the sets that are tested with their support for the dataset of Fig. 3 is given. In the first step all singletons are counted. In the second step only those pairs that are composed of frequent items are counted. In the third and last step, only one itemset of length 3 is counted: {emergency, vehicle, police}, since it is the only set of length 3 of which all subsets are frequent. For instance, the set {boat, emergency, vehicle} is not counted since the subset {boat,emergency} was found to be infrequent in the second step.

3.2 Depth-First Algorithms

Algorithm 2. Depth-First Itemset Mining

Input: Database $\mathcal{D}$, *minsup*
Output: Set $\mathcal{F}$ of frequent itemsets

```
function MineFrequent(D)
    F ← {a | {a} is frequent in D}
    F ← {{a} | a ∈ F}
    if |F| < 2 then
        return F                                              ▷ Base Case
    end if
    D ← {(TID, J ∩ F) | (TID, J) ∈ D and J ∩ F ≠ ∅}
            ▷ Remove infrequent items from D; keep only nonempty transactions
    while |F| > 1 do
        Pick an item a from F
        F ← F \ {a}
        D[a] ← {(TID, J \ {a}) | (TID, J) ∈ D and a ∈ J}
                                          ▷ Construct conditional Database
        F[a] ← MineFrequent(D[a])                             ▷ Recursion
        F ← {I ∪ {a} | I ∈ F[a]}
        D ← {(T, J \ {a}) | (TID, J) ∈ D}                     ▷ Remove a from D
    end while
    return F
end function

F ← MineFrequent(D)
return F
```

Although the breadth-first algorithms are provable optimal with respect to the exploitation of the monotonicity principle [54], they have the disadvantage that due to the simultaneous counting of all candidates of the same length, the counting procedure cannot exploit the fact that all transactions that support a certain itemset $\{a, b, c\}$, also contain its subsets, for instance $\{a, b\}$. Hence, instead of scanning the whole database in order to find and count those transactions that support $\{a, b, c\}$ as we have to do in a breadth-first algorithm, we could count $\{a, b, c\}$ directly after counting $\{a, b\}$. If we "remember" one way or another which transactions contained $\{a, b\}$, we could hence restrict our scan to only those transactions containing $\{a, b\}$. In a depth-first algorithm, this observation is key for improving the counting procedure. We illustrate the improved counting procedure with an example. Before counting the support of the supersets of the frequent itemset {emergency}, we copy all transactions containing {emergency} to form the *conditional database* for the itemset {emergency}. If there are only few transactions containing the itemset {emergency}, this can significantly reduce the time required for counting, since we do no longer need to scan the complete database, but only the smaller conditional database. In Algorithm 2, this principle is systematically and recursively exploited to form

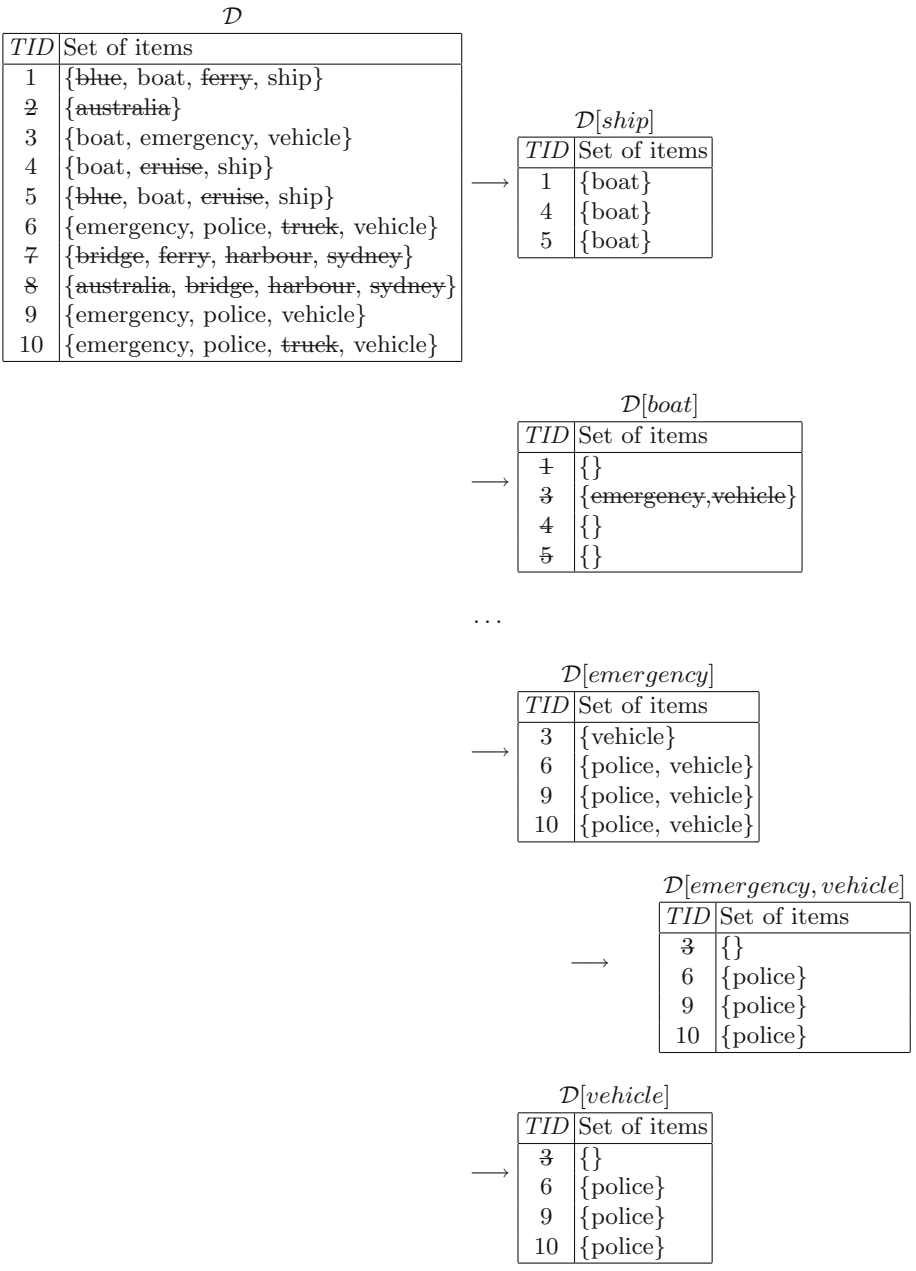


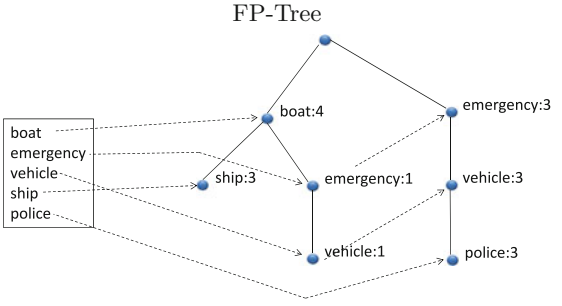
Fig. 6. Depth-First algorithm applied to the dataset depicted in Fig. 3

the framework of most depth-first frequent itemset mining algorithms, including FPGrowth [40] and Eclat [80]. These algorithms follow a divide-and-conquer strategy:

1. Select a frequent item a;
2. Make the conditional database $\mathcal{D}[a]$ containing only those transactions that contain item a and generate all frequent itemsets containing item a from $\mathcal{D}[a]$;
3. Remove item a from $\mathcal{D}$ and repeat.

Figure 6 illustrates the depth-first algorithm on the dataset of Fig. 3. In every recursive step we generate the associated conditional database. Initially we start with the database $\mathcal{D}[]$ containing only the frequent items.

The main source of variation between the depth-first pattern mining algorithms lies in the internal data structure they use. As can be derived from the list of operations to be performed by the depth-first algorithms, the data structure should allow to quickly (a) identify the frequent items, (b) construct the conditional database consisting of all transactions that contain a particular item, and (c) remove (or ignore) infrequent and already processed items. FPGrowth uses for this purpose a so-called *FPTree*, which is based on a double-linked prefix-tree of the transactions extended with a header table connecting the same item appearing in transactions with different prefixes. Eclat instead employs *TID-lists*; for every item, the list of transaction-identifiers containing that item is stored. In Fig. 7 both the FPTree and the Eclat representation of the database in Fig. 3 have been depicted. The exact details of both algorithms are beyond the scope of this tutorial. We refer the interested reader to [32] for further details.



Eclat representation

Item	*TID*-list
ship	1, 4, 5
boat	1, 3, 4, 5
emergency	3, 6, 9, 10
vehicle	3, 6, 9, 10
police	6, 9, 10

Fig. 7. FPTree and Eclat-representation of the database in Fig. 3.

4 Alternative Pattern Types and Interestingness Measures

There exist many extensions and variations to the original frequent itemset mining problem. In general, the pattern mining problem is defined as follows.

Given a class of patterns $\mathcal{C}$, and a predicate q expressing if a pattern p is interesting ($q(p)$), find the set of all interesting patterns: $\{p \in \mathcal{C} \mid q(p)\}$. For example, for frequent itemset mining, the class of patterns $\mathcal{C}$ consists of all itemsets, and the predicate q is defined as follows:

$$q(I) \;\Leftrightarrow\; support(I, \mathcal{D}) \geq minsup.$$

For the purpose of the overview we divide them according to two orthogonal dimensions: on the one hand, more complex pattern types by adding structural or temporal dimensions and on the other by introducing alternative, often application-oriented interestingness measures. Nevertheless, despite the great variety of measures it is fair to say that most commercial data mining suites usually support only the most basic types, being: frequent itemset, association rules, and sequence mining based on support and confidence.

4.1 More Complex Pattern Types

In the literature, among others, the following classes of patterns have been studied:

1. Sequences [55]. Either a database of sequences or one large sequence is given. A support measure can be defined as the number of sequences in the database of which the pattern sequence is a subsequence. In the case of one large sequence the largest number of non-overlapping embeddings of the sequence can be chosen as support measure. For example, under the non-overlapping embedding semantics, the support of *abc* in

   $$a\underline{a}d\underline{bc}bcade\underline{ba}d\underline{be}\underline{cc}\underline{a}dd\underline{be}\underline{ca}e\underline{bc}$$

   would be 4. Many extensions, for instance to include control over the gap length in an embedding, have been considered in the literature [78].
2. Graphs [1], Networks [62, Ch. 10] [67], and even hypergraphs [42]. Similar as for the sequences, there are two main paradigms: either a database of graphs is considered and the support represents in how many of the "transaction" graphs in the database the pattern graph occurs [44,74], or the data consists of one large graph and the support measure represents how common the pattern is in the large graph [13,41,71]. Examples for the first setting would be a database of molecules represented as graphs; a frequent pattern would then correspond to a fragment common to many of the molecules. For the latter case, a social web graph, such as for instance a friendship graph between participants in a social network, would be a good example. A frequent pattern would express a common structure in the social network. Special cases of graphs have been studied as well, including trees [79], partial orders [19,60], connected subgraphs in a graph with bounded tree-width [43], cliques [73], bicliques [63], and quasi-cliques [70].
3. By combining the temporal aspect with graphs we arrive at *dynamic or evolving graphs* [7]. In many networks the temporal aspect is of great importance,

for instance if we want to study the evolution of the network. Therefore, recently different research groups have started studying dynamic graph patterns [7]. Such patterns may express common evolution patterns in a dynamic graph.

4. Multi-relational patterns. A graph can be considered as an extension of transaction in which the set of items is replaced by a set of edges which are pairs of items. In a similar fashion, we can extend sets of pairs of items to arbitrary relations over our universe of items to arrive at relational or even multi-relational datasets in which we can define multi-relational patterns [28,33,65] or subgroups [75].
5. Another popular class of patterns are the rules. A rule typically is an implication $X \Rightarrow Y$ in which X and Y belong to one of the pattern classes mentioned above. The semantics of such a rule is that an occurrence of a pattern X implies the occurrence of pattern Y. The rules themselves can be considered to be patterns themselves. Important classes of rules are association rules [2], and quantitative association rules [66], functional and inclusion dependencies [57], and embedded functional dependencies [24]. For rules, next to the support which expresses the prevalence of the rule, usually also the strength of the rule is expressed by a measure of confidence.

There is an incredibly wealth of different variations and extensions of association rules. To illustrate the enormity of the domain, consider the fact that the original papers [2,3] have been cited *over 13 000 times, each*[2]. For an overview of this enormous domain, we refer the interested reader to the following survey papers [32,38].

4.2 Alternative Measures of Interestingness

Depending on the need in different applications, also different interestingness measures have been defined. In [29], an extensive overview of different interestingness measures for the standard transaction database setting has been given. The overview considers 39 common measures for rules and simple patterns, including support, confidence, precision, conviction, lift, coverage, leverage, odds ratio, prevalence, specificity, information gain and many others. In [29], the measures are divided into three large classes: the objective measures, that measure a certain statistic of the data such as the number of occurrences, subjective measures that try to quantify the interestingness of a pattern, and semantic measures that capture for instance the utility or actionability of a pattern [52,76]. All 39 measures concern itemsets and association rules in normal transactional data. Furthermore, depending on the type and properties of the data, support measures have been adapted. We list a couple of popular alternative formulations that depend on the type and properties of data at hand:

1. Often the data can be split into two or more groups, for instance in a classification task. In such a context it could be interesting to find patterns that are

[2] Source: Google scholar http://scholar.google.be/, January 8th, 2014.

much more prevalent in one dataset/class than in the other. Emerging patterns [22] are an example of pattern mined under such context, in which the interestingness is defined as the support in one dataset divided by the support in the other. Similar in nature are the so-called contrast sets [5,6]. Needless to say, several other existing measures of correlation between class/dataset and pattern could be used as alternative measure for contrast/emerging patterns as well. Much similar in spirit is the field of subgroup discovery [4,34,75]. Although the type of patterns studied in subgroup discovery is in essence the same as for emerging patterns and contrast sets, the communities studying these patterns adhere different search strategies. Whereas the search for emerging and contrast sets is usually exhaustive and Apriori-style, in the subgroup discovery community the most popular search techniques include branch-and-bound, and heuristic search strategies such as beam search.
For an unified overview of the emerging patterns, contrast sets, and subgroup discovery, see [56].

2. Alternative measures have been defined for the case when the input data is assumed to contain errors, hence requiring a support measure that accounts for the influence of noise on the rigid exact measures that require every single item in an itemset to be present in a transaction in order for the transaction to support the itemset. Examples of the pattern types resulting from a more relaxed support definition are fault-tolerant (FT) frequent itemsets [46,59], weak and strong Error Tolerant Itemsets (ETIs) [77], Approximate Frequent Itemsets (AFIs) [51], and extensions of these [20,61].
3. Sometimes it is assumed that we do not only know that there might be noise present in the data, but we also have some level of certainty of correctness of our data. This leads to a probability distribution over the possible data values in the database entries. For such data, we can define probabilistic measures such as the expected support [21], or the frequentness probability [8]. For the case where independence between the uncertain entries in the database is assumed, is has been shown that good approximations can be obtained using simple techniques from statistics, such as sampling and the normal approximation of the binomial distribution [14,15].
4. Mining numerical data: if the database contains numerical attributes it is no longer possible to directly apply frequent itemset mining. The most popular and prevalent solution in such case is to discretize the numerical values in order to reduce the numerical data to the required itemset data; a record would then be transformed to the set of attribute-bucket pairs listing in which bucket a specific attribute can be classified. Nevertheless, there have been a few extensions studied that directly work on the numerical data removing the need for discretization [25,45].

4.3 Main Challenges for Mining Alternative Pattern Types

Mining more complex patterns comes at a price. There are two main challenges when mining more complex pattern types which we will illustrate both with the mining of frequent subgraphs in one large datagraph.

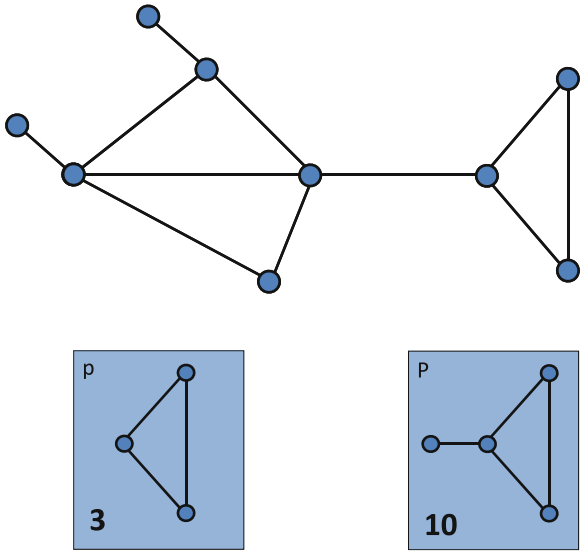


Fig. 8. Illustration of the fact that counting the number of embeddings of a pattern in a single datagraph is not a monotonic measure. The triangle has 3 embeddings in the given datagraph, whereas its extension has 10 unique embeddings.

Monotone Support Measures. It is important to have a monotone support measure expressing the frequency of a pattern, because all algorithms rely heavily on this property for efficient enumeration of all frequent patterns. Furthermore, a non-monotone measure is counter-intuitive as intuitively it should not be possible that a specialization of a pattern is more frequent than the pattern itself, which is part of this specialization and hence occurs every time the specialization occurs. In many cases, however, finding such a monotone measure is highly non-trivial. Consider for instance Fig. 8. This figure shows one database graph together with two patterns, one triangle, and its specialization consisting of one triangle with a dangling edge attached to one of its corners. The most straightforward choice for a support measure would be to count the number of embeddings of the pattern. As we can see in the figure, however, this measure is not monotone as it assigns a support of 3 to the triangle, and a support of 10 to the specialization with the dangling edge. The reason is that the specialized pattern can have many embeddings that correspond to one embedding for the triangle; i.e., the embeddings only differ in the embedding of the dangling edge. Therefore multiple other measures for mining subgraphs of a single large graph have been proposed, most of which are based on the notion of an overlap graph [17,71].

Efficient Search Space Traversal. Most of the more complex pattern mining algorithms apply a depth-first enumeration of all patterns, comparable to the depth-first enumeration for the frequent itemsets described in the previous

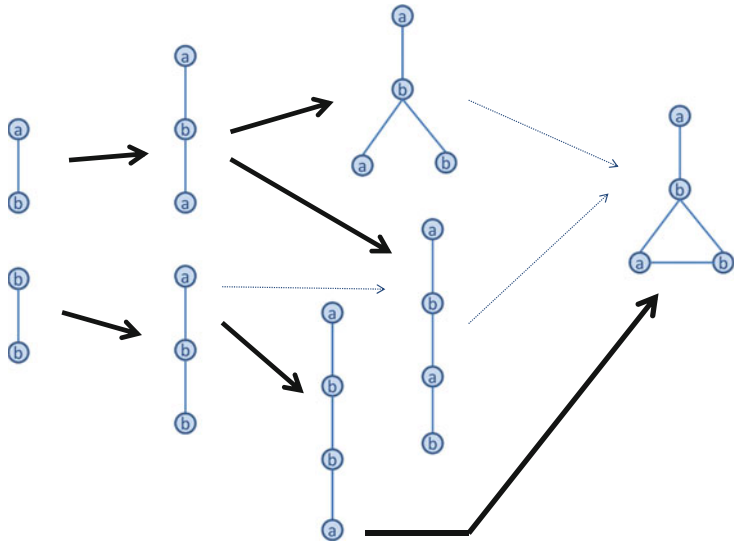


Fig. 9. Part of the search space of all graph patterns. The arrows represent the specialization relation.

section. The reason for the preference of depth-first over breadth-first algorithms is because for most of the more complex pattern types, the number of patterns at a certain level is much higher than for the frequent itemset mining problem. For instance, there are $\binom{n}{k} \approx \frac{n^k}{k!}$ itemsets of length k for a set of n items, whereas there are $\binom{n}{k} 2^{k(k-1)} \approx \frac{n^k}{k!} 2^{k(k-1)}$ directed graphs with k nodes from a set of n nodes. As the breadth-first algorithm implies dealing with a complete level at once, this incurs a great memory footprint. Even more important is that for more complex patterns counting support is an expensive operation. For graphs it usually involves some sort of subgraph-isomorphism test, a notoriously hard problem [27]. Therefore, depth-first enumeration is preferable as it more easily allows reusing the computations of the support of one of the pattern's parents. For instance, finding all embedded subgraphs of a pattern can be done more efficiently if all embeddings of one of its generalizations with one less edge have been cached.

An important component of a depth-first algorithm is a procedure to generate the children of a pattern, that is, the specializations of a pattern that are recursively explored after it. Enumerating all candidate patterns to be counted against the datagraph without duplicates, however, is not straightforward. Figure 9 shows part of the search space of all subgraph patterns. The edges indicate the specialization relation between the different patterns. In order to avoid duplicates, every pattern should be generated as a specialization of one single parent pattern only. For itemsets such is easy to achieve: fix an order between the items and let

every itemset $i_1 \dots i_n$ only be generated by its prefix $i_1 \dots i_{n-1}$. For graphs this problem is much more difficult to solve. Indeed, an efficient way to generate all graphs without duplicates would immediately lead to an efficient way to determine a canonical representation of a graph by tracking the path from the graph to its ancestors and storing the edges of the graph in the order they are added on this path. Obtaining a canonical form of a graph, however, is assumed not to be in polynomial time, as it would allow to solve the graph homomorphism problem efficiently. Graph homomorphism, however, is a well-known problem of which the complexity is still open and expected not to be possible polynomial [27].

5 Pattern Mining Applied on Real Data

For an overview of different applications of pattern mining and more specifically sequence mining, we refer to [36]. In that book chapter several examples of applications of pattern mining have been given in healthcare, education, web usage mining, text mining, bioinformatics, telecommunication, and intrusion detection. In this section we will concentrate of a few examples from our own work which we divide into two categories.

5.1 Patterns as Input for Other Algorithms

Patterns can be used as input for other applications. Frequent itemsets could be used as features in classification, where each frequent itemset is transformed into a feature for a transaction expressing whether or not the itemset is present in the transaction. In combination with feature selection techniques to reduce the total number of features and retain the most promising features only, this can lead to performant classifiers [37,50]. Similarly, frequent itemsets can be used as an alternative description of the data and changes in the patterns can be monitored to detect changes in continuous processes.

Figure 10 represents two experiments in which patterns are used to find anomalies in streams of events. In both cases it is monitored how well the patterns can

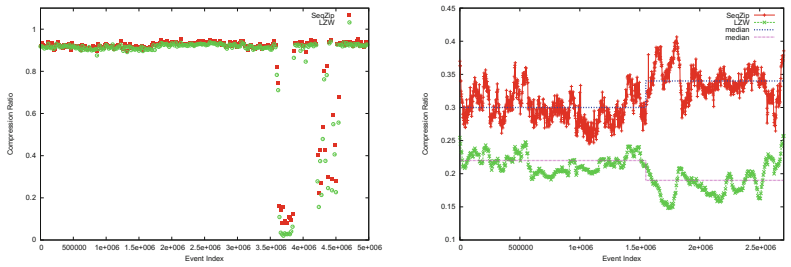


Fig. 10. Example of patterns being used to detect changes in a stream of events. By monitoring how well the patterns can cover (compress) the data stream, exceptional changes in the distribution of the stream can be detected. Figures are taken from [69]

compress the data streams. This number expresses the regularity of the stream. The more regular it is, the better the patterns will compress it. Figure 10(left) represents an experiment with weblog data from the FIFA World Cup 1998 website. The weblog contains 5 million accesses to the websites objects, being html pages and images, by football fans before the opening date of the event. We used two compression algorithms based on patterns, to compress sliding windows of size 50 000 over the stream of events. The results are plotted in Fig. 10 (left). The interesting and suspicious points in the data are those where the compression ratio suddenly drops. In order to figure out what happened at these suspicious points we extracted the compressing patterns for the corresponding windows. The obtained patterns always contain repetitions of an access to the frontpage of the website. By looking further at the IP addresses of the users who have accessed the frontpage at that time, we discovered that there are only two IP addresses continuously sending requests for the websites frontpage. These suspicious points might correspond to the website being under attack. This technique could be used in order to improve data security by automatic detection of diverging behavior, leading to a more sensitive intrusion detection system than the more common systems based on recognizing known attack patterns.

Figure 10(right) represents the results of similar experiments with log data from a photolithography system for the semiconductor industry. The machine log contains 2.7 million messages collected in a period of six months from May till November 2011. We used again the same algorithms to compress sliding windows of size 50 000 over the event stream and plotted the results in Fig. 10(right). The obtained results show that two compression ratio curves change their medians as the stream approaches 1.6 million events. We extracted the most compressing pattern for the window with the lowest compression ratio right after the change point (window number 1 750 000). We found a very long pattern consisting of sixteen different warning messages:

```
fER-0FFF RY-2078 RY-5607 RY-5608 RY-5609 RY-560A RY-560B RY-560C
RY-560E RY-560F RY-5610 RY-5611 RY-5612 RY-5613 RY-5614g
```

All the warning messages are regarding out-of-range movements of a specific component of the machine, i.e., the RY component. For example the warning message RY-5607 is associated with *X Force peak values during move*, while the warning message RY-560E is associated with *First Touch during move*. This information may help experts to further analyze the machine operation changes. By systematically analyzing the systems behavior, we may be able to recognize patterns correlated to future breakdowns of the machine. Such early detection of potential future malfunctioning of the machine allows for a more efficient scheduling of repairs or maintenance and therefore result in a reduction of downtime.

5.2 Patterns for Summarization

Patterns can also be used for exploratory analysis by providing a summary of the most important sets of items. For instance, in Fig. 11 a visualization of a twitter stream is given that is based upon a carefully selected subset of the frequent

Fig. 11. Visualization of a twitter stream with the help of frequent patterns. Figure is taken from [69]

patterns. Such visualizations could help in exploratory analysis by providing a quick overview of the content of a dataset.

The ability to quickly analyze and visualize the most important topics in social media has several applications. Many companies are very interested in what the public opinion is about their products. For this they harvest social media such as specialized blog, online reviews, and twitter. The textual data, however, is hard to analyze using traditional methods that assume a structured format of the data. Here pattern mining techniques can help to automatically identify important and orthogonal topics from the textual messages. In this way the semi-structured textual messages are transformed to subsets of topics.

5.3 Implementations of Pattern Mining

There are several implementations of pattern mining available. Virtually every commercial data warehousing product contains data mining extensions for frequent pattern and association rule mining, including MS SQL Server Analysis Services, Oracle, SAP BW, Teradata Warehouse Miner, as well as virtually every data mining and statistics package, including IBM SPSS, SAS enterprise miner, MATLAB, etc. Furthermore, there are many free implementations available, such as for instance the starter version of Rapid Miner[3], Weka[4], the R-extension Rattle[5], and KNime[6]. Many easy-to-use research prototypes can be found in

[3] http://rapidminer.com/products/rapidminer-studio/
[4] http://www.cs.waikato.ac.nz/ml/weka/
[5] http://rattle.togaware.com/
[6] http://www.knime.org/

the "Freqeunt Itemset Mining Implementations Repository"[7]. For a more comprehensive list of available tools, consult the software listing of the kdnuggets community page.[8]

6 The Pattern Explosion Problem

One of the biggest problems in pattern mining is the so-called *pattern explosion problem*. This means that listing all patterns that satisfy a certain minimal support threshold often results in a huge amount of patterns. These patterns are often very similar and contain a lot of redundancy. For example, due to the monotonicity principle, if a pattern of size 10 is frequent, it means that all its 1024 subsets are frequent as well, and will be part of the output, too. In Fig. 12, the longest frequent patterns for a subset of the tags dataset for a minimal support threshold of 300 have been listed. As can be seen, this set of patterns contains a lot of redundant patterns; many patterns are combinations of the words aeroplane, airport, aircraft, aviation, flight. Once a few of these combinations are known to be frequent, the other patterns that are subset of this list of words add little to the understanding of the popular topics in the tags dataset. In this section we will sketch some techniques to remove redundancy from the set of all frequent itemsets. First we will introduce condensed representations [18], which were studied mainly in the 90s and the 2000s. After that we describe more recent techniques based on statistics on the one hand and the minimal description length principle on the other.

6.1 Condensed Representations for Frequent Patterns

A *condensed representation* of the set of frequent itemsets is a subset of the frequent itemsets that contains the same information. There are some exceptions in which the condensed representation may also contain a few infrequent itemsets as well, such as the free sets representation. But the general principle remains: the goal of the condensed representation is to represent the same information without the redundancies that normally occur in the collection of frequent itemsets. Therefore, condensed representations are based upon reasoning about supports of itemsets. For instance: if the support of $\{a, b, c\}$ is 3, and the support of $\{a\}$ is 3, there is no need to store the supports of $\{a, b\}$ and $\{a, c\}$, since they must be 3 as well, due to the monotonicity principle. The closed itemset representation [58] which we will detail in the next paragraphs, will in this particular case only store $\{a, b, c\}$. Other well-known representations include the *free sets representation* [12] which is closely connected to the closed itemsets representation, and the *non-derivable itemsets representation* [16] that is based upon a complete set of deduction rules for support. For a survey about condensed representations, we refer the interested reader to [18].

[7] http://fimi.ua.ac.be/
[8] http://www.kdnuggets.com/software/index.html

Itemset	Support
{flight, aeroplane, travel, aircraft, plane}	304
{flight, aeroplane, aircraft, plane}	319
{aeroplane, travel, aircraft, plane }	308
{flight, travel, aircraft, plane }	305
{flight, aeroplane, travel, plane }	304
{flight, aeroplane, travel, aircraft}	304
{airport, aircraft, plane}	639
{and, black, white}	630
{war, protest, demonstration}	511
{aeroplane, aircraft, plane}	489
{ussmidway, sandiego, aircraftcarrier}	458
{aviation, aircraft, plane}	449
{protest, demonstration, of }	416
{boat, ship, water}	404
{aeroplane, airport, aircraft}	395
{flight, aeroplane, plane}	393
{aviation, airport, aircraft}	392
{flight, aircraft, plane}	388
{airplane, aircraft, plane}	372
{aviation, airport, plane}	371

Fig. 12. List of frequent patterns for a sample of the tags dataset. Only the 20 longest frequent patterns have been depicted

Closed Itemsets. The closed itemset representation is based on the notion of closed set used in formal concept analysis, a branch of lattice theory dedicated to the study of the lattice structure induced by a binary relation (structure called Galois lattice or concept lattice). The application of this theory to frequent itemset mining has been proposed by Pasquier et al. in [58]. An itemset I is said to be *closed* in $\mathcal{D}$ if and only if no proper superset of I has the same support than I in $\mathcal{D}$. Consider, for instance the example in Fig. 13. The itemset {police, vehicle} is not closed because it has the same support as its proper superset {emergency, police, vehicle}.

The *closure* of an itemset I in $\mathcal{D}$, denoted $cl(I)$, is the unique maximal superset of I having the same support than I. A closed itemset is hence equal to its own closure. The closure of both {police} and {police, vehicle} is {emergency, police, vehicle}. Notice that these three sets all appear in exactly the same set of transactions, namely transactions 6, 9, and 10 (cfr. the database in Fig. 3). This is not a coincidence, but actually forms the basis of an elegant alternative definition: two itemsets are equivalent if they appear in exactly the same transactions. The closed itemsets then correspond to the unique maximal elements of these equivalence classes.

For a given support threshold, it is sufficient to know the collection of all frequent closed itemsets and their supports, to be able to generate all the frequent itemsets and their supports. For example, consider an itemset X. If X is frequent,

Frequent Itemset	support
{}	10
~~{ship}~~	~~3~~
{boat}	4
~~{vehicle}~~	~~4~~
~~{emergency}~~	~~4~~
~~{police}~~	~~3~~
{boat, ship}	3
{emergency, vehicle}	4
~~{police, vehicle}~~	~~3~~
~~{emergency, police}~~	~~3~~
{emergency, police, vehicle}	3

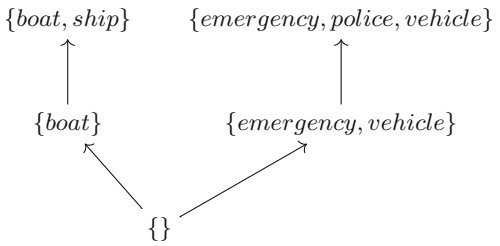


Fig. 13. Closed frequent itemsets in the database given in Fig. 3 for a frequency threshold of 3. The stricken itemsets indicate frequent itemsets that are not closed. On the right-hand side the Hasse-diagram of the frequent part of the concept lattice is given.

then either X is closed and we can derive its support from the closed itemset representation, or X is not closed. In that case, the closure of X, $cl(X)$ must be in the collection of frequent closed itemsets, since its support is the same as the support of X. Hence, for every itemset we can quickly derive if it is frequent or not, based only upon the collection of frequent closed itemsets: only if a superset of X is in the representation, X is frequent. Finally, the frequency of X itself we can derive using the monotonicity principle. The closure of X must be in the collection, and hence the support of X must be equal to the support of one of its supersets. If there are multiple candidates, it has to be the maximal support over all its supersets because otherwise the monotonicity principle is violated. For instance, from the collection of frequent itemsets in Fig. 13, we can derive that {police} is frequent, because its superset {emergency, police, vehicle} is frequent and closed. Since it is the only superset of {police} in the representation, their supports have to be the same.

The closed itemset representation, however, does not remove all redundancies either. In Fig. 14, the list of longest frequent patterns in Fig. 12 has been restricted to the closed itemsets only. The set still contains many redundancies. The main reason is the strict requirement that the support of a frequent itemset must be *exactly* equal to the support of one of its supersets before it is removed from the output. Furthermore, the closed itemset representation does not remove the problem of frequent but unrelated items that together form sets of unrelated items that meet the support threshold. Consider for instance an imaginary case of an item with a support of 99 % that is added to the database. Adding only one such item would result in almost doubling the number of frequent (closed) itemsets! Therefore in the next subsections we will look at more advanced techniques based upon statistics and data encoding.

Itemset	Support
{flight, aeroplane, travel, aircraft, plane}	304
{flight, aeroplane, aircraft, plane }	319
{aeroplane, travel, aircraft, plane }	308
{flight, travel, aircraft, plane }	305
{airport, aircraft, plane }	639
{and, black, white }	630
{war, protest, demonstration }	511
{aeroplane, aircraft, plane }	489
{ussmidway, sandiego, aircraftcarrier }	458
{aviation, aircraft, plane }	449
{protest, demonstration, of }	416
{boat, ship, water }	404
{aeroplane, airport, aircraft }	395
{flight, aeroplane, plane }	393
{aviation, airport, aircraft }	392
{flight, aircraft, plane }	388
{airplane, aircraft, plane }	372
{aviation, airport, plane }	371
{travel, aircraft, plane }	358
{flight, travel, plane }	346

Fig. 14. List of frequent closed patterns for a sample of the tags dataset. Only the 20 longest frequent closed patterns have been listed.

6.2 Statistical Methods for Modelling Expectation

The statistical method for mining non-redundant patterns is depicted in Fig. 15. This method is based upon the notions of expectation and surprisingness. If a user knows the support of certain patterns, this raises expectations about the supports of other patterns. This expectation is modelled as a distribution over the possible support values of the pattern. For instance, if in a dataset 50 % of the transactions contains item a, and 50 % contains item b, without further knowledge, one would expect that 25 % of the transactions contains the itemset $\{a, b\}$. If the support of $\{a, b\}$ in the dataset does not divert too much from 25 % it is hence not necessary to report this support as it is according to expectation and thus not surprising. The different approaches in this area [9,10,31,47,53] vary mainly in the following aspects:

- How is the "expectation" modelled? Given information about the supports of some itemsets, how do we derive distributions that express our expectations regarding the supports of the others? Due to its favorable mathematical properties, often distributions based upon maximal entropy are chosen.
- Given a distribution over the supports of the itemsets, how do we quantify the interestingness of an itemset not yet in the collection? There are approaches based on information theory that assess how much information a certain support carries, and others that use p-values that express how extreme/unlikely a

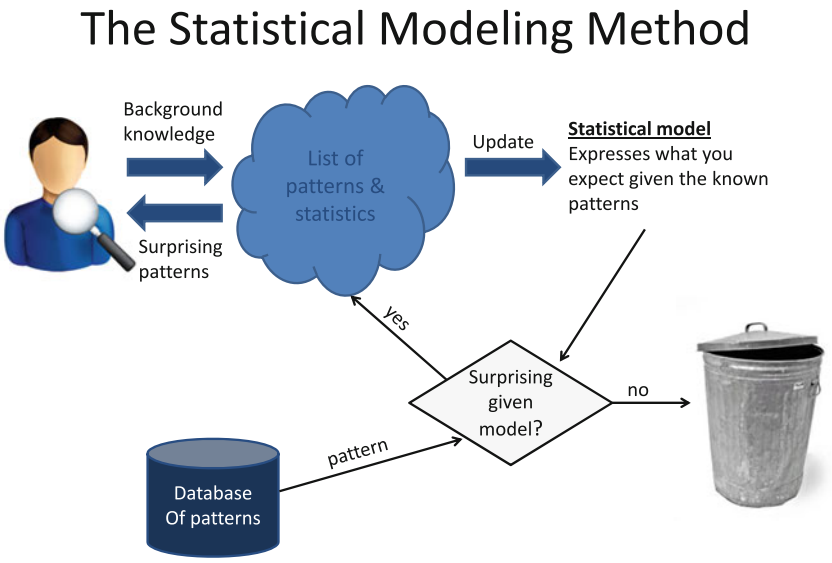


Fig. 15. Illustration of the statistical model to filter out redundant patterns

certain observed support is, given the knowledge that we already have about the other patterns.

All methods then roughly work according to the following iterative procedure: start with an empty knowledge base. In every step the most surprising/informative itemset is selected given the current knowledge base, and added to it. After adding the support information of the itemset, the model of the expectation changes and patterns that may have been surprising before the latest addition may suddenly become much less interesting or surprising. In the end only those patterns that have been selected into the knowledge base are reported to the user. In Fig. 16 the result of the MTV-algorithm [53] which is based upon these principles has been depicted for the same dataset that was used to produce the frequent itemsets of Fig. 12 and the closed frequent itemsets of Fig. 14. Clearly the set of patterns produced by this method contains far less redundancy and is much more informative with respect to summarizing the actual content of the Flickr dataset.

6.3 Minimal Description Length Based Methods for Removing Redundancy

Closely related to the statistical method is the method based on the minimal description length (MDL). For more information on MDL and its many applications we refer the interested reader to [23,35]. In the MDL-based method not statistical expectation is used to assess the interestingness of a pattern given a set of background patterns, but rather how well the pattern can be used to

Itemset
{geo, geotagged, lat, lon}
{airplane, plane, flying, aircraft}
{boat, ship}
{city, nyc, new, york}
{two, people}
{and, white, black}
{night, exposure, long}
{b, w}
{protest, demonstration}
{airplane, flying, aviation}
{san, francisco}
{diamondclassphotographer, flickrdiamond}

Fig. 16. Output of the MTV algorithm based on the statistical approach

compress the database [48,49,72]. The relation between the interestingness of a pattern and its ability to compress the dataset can best be illustrated with an example. Suppose for instance that we know that the pattern $\{a, b\}$ occurs very frequently. In that case we could consider introducing a special code for the itemset $\{a, b\}$, which is much shorter than the items a and b separately. In that way, the database could be compressed considerably by replacing every occurrence of $\{a, b\}$ with the code. As such, the usefulness of the pattern is quantified by its ability to compress. The rationale of this measure can be seen as follows: consider again the list of patterns in Fig. 12. Clearly any of the long patterns in that figure would help reducing the size of the database enormously. However, once we select one of these patterns, the benefit of the other patterns would drop. For instance, if we select pattern {aircraft, plane}, almost all of the patterns in Fig. 12 would see their benefit reduced. For example, the benefit of the pattern {airport, aircraft, plane} per transaction that contains it is no longer the cost of representing the three individual items (cost without any pattern) minus the cost of one code for the pattern (cost with the pattern), but instead the difference between the reduced cost of the item airport and the code for {aircraft, plane} (cost with the pattern {aircraft, plane} but without {airport, aircraft, plane}) minus the cost for one code for the itemset {airport, aircraft, plane}. As such, patterns that overlap with patterns that are added to the output will drop in the ranking of most useful for compression.

Several new approaches use this minimal description length inspired model for reducing the redundancy in many pattern mining problems. The advantage of the MDL method over the statistical method is that overall it is easier to come up with a good encoding for more complicated pattern types than it is to come up with a good statistical model expressing the surprisingness of patterns. For instance, think about the complexity of coming up with a statistical model expressing the frequencies of certain subgraphs given the frequencies of some other subgraphs. Another advantage is that usually the complexity of assessing

the usefulness of a pattern is lower in the MDL case, although it comes at the price of having less well-founded techniques; most of the MDL-based techniques are highly heuristic in nature.

7 Conclusion

In this tutorial paper we started with an overview of the frequent itemset mining problem, the main algorithmic techniques, extensions, and applications. Despite its simple definition, the frequent itemset mining problem has a high computational complexity. We described the two main techniques for mining frequent itemsets, namely the breadth-first and depth-first algorithms. There exist many extensions to the frequent itemset mining problem and we have reviewed the main difficulties when extending to the more complex pattern types.

Then, in the second part of the paper we discussed the pattern explosion problem and some ways to deal with it. The first attempts to deal with this problem, condensed representations, originate from the late 90s. Condensed representations aim at reducing the redundancy by providing a lossless subset of the complete collection of frequent itemsets. This combinatorial approach, based upon exact reasoning with itemset supports, however, soon turned out to be insufficient and new techniques based on statistical modeling and the minimal description length emerged. These methods are based upon quantifying the interestingness of patterns by measuring their surprisingness and/or their usefulness in describing or compressing the database.

We expect that in the future we will see increasingly more techniques to deal with the redundancy problem, and we expect advances in the following directions:

- Theory behind the new statistical and MDL based methods. The new techniques rely heavily on mathematically well-founded notions such as maximal-entropy distributions and the MDL principle. Algorithmically, however, there is a lot of ad-hocness in order to improve performance. These ad-hoc steps in the algorithms form a threat for the validity of the techniques. More theoretical work is needed in order to be able to assess the impact of the ad-hoc steps in the algorithms.
- Despite the appeal of automatic techniques for selecting and ranking non-redundant patterns, it is inevitable that at a certain point the user needs to be involved in the process of identifying the interesting patterns. What is interesting to someone is highly subjective and depends on what this person already knows. We can, however, not expect from the user to provide us with a complete list of patterns describing in detail his or her background knowledge. Therefore there will be a need for more interactive schemes involving the user in the pattern selection process. A nice preliminary approach for automatic incorporation of implicit user feedback in pattern mining is given in [11].
- Many of the techniques are aimed at mining frequent itemsets. Extending these techniques to more complex pattern domains will be a challenging venue for future work.

References

1. Aggarwal, C.C., Wang, H.: Managing and Mining Graph Data. Springer, New york (2010)
2. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. ACM SIGMOD Rec. **22**(2), 207–216 (1993)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, VLDB, vol. 1215, pp. 487–499 (1994)
4. Atzmüller, M., Puppe, F.: SD-Map–a fast algorithm for exhaustive subgroup discovery. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - ECML PKDD, pp. 6–17. Springer (2006)
5. Bay, S.D., Pazzani, M.J.: Detecting change in categorical data: Mining contrast sets. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 302–306. ACM (1999)
6. Bay, S.D., Pazzani, M.J.: Detecting group differences: mining contrast sets. Data Min. Knowl. Disc. **5**(3), 213–246 (2001)
7. Berlingerio, M., Bonchi, F., Bringmann, B., Gionis, A.: Mining graph evolution rules. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part I. LNCS, vol. 5781, pp. 115–130. Springer, Heidelberg (2009)
8. Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., Zuefle, A.: Probabilistic frequent itemset mining in uncertain databases. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 119–128. ACM, (2009)
9. De Bie, T.: Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Min. Knowl. Disc. **23**(3), 407–446 (2011)
10. De Bie, T., Spyropoulou, E.: A theoretical framework for exploratory data mining: recent insights and challenges ahead. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part III. LNCS, vol. 8190, pp. 612–616. Springer, Heidelberg (2013)
11. Boley, M., Mampaey, M., Kang, B., Tokmakov, P., Wrobel, S.: One click mining–interactive local pattern discovery through implicit preference and performance learning. In: KDD 2013 Workshop on Interactive Data Exploration and Analytics (IDEA) (2013)
12. Boulicaut, J.-F., Bykowski, A., Rigotti, C.: Free-sets: a condensed representation of boolean data for the approximation of frequency queries. Data Min. Knowl. Disc. **7**(1), 5–22 (2003)
13. Bringmann, B., Nijssen, S.: What is frequent in a single graph? In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 858–863. Springer, Heidelberg (2008)
14. Calders, T., Garboni, C., Goethals, B.: Approximation of frequentness probability of itemsets in uncertain data. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 749–754. IEEE (2010)
15. Calders, T., Garboni, C., Goethals, B.: Efficient pattern mining of uncertain data with sampling. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part I. LNCS, vol. 6118, pp. 480–487. Springer, Heidelberg (2010)
16. Calders, T., Goethals, B.: Non-derivable itemset mining. Data Min. Knowl. Disc. **14**(1), 171–206 (2007)
17. Calders, T., Ramon, J., Van Dyck, D.: All normalized anti-monotonic overlap graph measures are bounded. Data Min. Knowl. Disc. **23**(3), 503–548 (2011)

18. Calders, T., Rigotti, v., Boulicaut, J.-F.: A survey on condensed representations for frequent sets. In: Constraint-Based Mining and Inductive Databases, pp. 64–80. Springer (2006)
19. Casas-Garriga, G.: Summarizing sequential data with closed partial orders. In: Proceedings of the SIAM International Conference on Data Mining (SDM), pp. 380–391 (2005)
20. Cheng, H., Yu, P.S., Han, J.: AC-Close: efficiently mining approximate closed itemsets by core pattern recovery. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 839–844 (2006)
21. Chui, C.-K., Kao, B., Hung, E.: A decremental approach for mining frequent itemsets from uncertain data. In: Washio, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 64–75. Springer, Heidelberg (2008)
22. Dong, G., Li, J.: Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 43–52. ACM (1999)
23. Faloutsos, Ch., Megalooikonomou, V.: On data mining, compression, and kolmogorov complexity. Data Min. Knowl. Disc. **15**(1), 3–20 (2007)
24. Fan, W., Geerts, F., Li, J., Xiong, M.: Discovering conditional functional dependencies. IEEE Trans. Knowl. Data Eng. **23**(5), 683–698 (2011)
25. Fukuda, T., Morimoto, Y., Morishita, S., Tokuyama, T.: Mining optimized association rules for numeric attributes. In: Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp. 182–191. ACM (1996)
26. Gantz, J., Reinsel, D.: The digital universe decade—are you ready? IDC White Paper, May 2010. http://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf
27. Garey, M.R., Johnson, D.S.: Computers and Intractability, vol. 174. Freeman, New York (1979)
28. Garriga, G.C., Khardon, R., De Raedt, L.: On mining closed sets in multi-relational data. In: Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI), vol. 7, pp. 804–809 (2007)
29. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: a survey. ACM Comput. Surv. (CSUR) **38**(3):9 (2006)
30. Gianni, G., Fosca, G., Pedreschi, D.: Mobility, data mining and privacy: geographic knowledge discovery. Springer, Heidelberg (2008)
31. Gionis, A., Mannila, H., Mielikäinen, T., Tsaparas, P.: Assessing data mining results via swap randomization. ACM Trans. Knowl. Disc. Data (TKDD) **1**(3), 14 (2007)
32. Goethals, B.: Survey on Frequent Pattern Mining. University of Helsinki, Finland (2003)
33. Goethals, B., Le Page, W., Mampaey, M.: Mining interesting sets and rules in relational databases. In: Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 997–1001. ACM (2010)
34. Grosskreutz, H., Rüping, S., Wrobel, S.: Tight optimistic estimates for fast subgroup discovery. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 440–456. Springer, Heidelberg (2008)
35. Grünwald, P.D.: The Minimum Description Length Principle. The MIT Press, Cambridge (2007)

36. Gupta, M., Han, J.: Applications of pattern discovery using sequential data mining. In: Kumar, P., Radha Krishna, P., Bapi Raju, S. (eds.) Pattern Discovery Using Sequence Data Mining: Applications and Studies, chapter 1, pp. 1–23. IGI Global (2012)
37. Han, J.: CPAR: Classification based on predictive association rules. Proceedings of the Third SIAM International Conference on Data Mining, pp. 331–335. SIAM, Philadelphia (2003)
38. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Disc. **15**(1), 55–86 (2007)
39. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2006)
40. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. ACM SIGMOD Rec. **29**(2), 1–12 (2000)
41. Holder, L.B., Cook, D.J., Djoko, S.: Substucture discovery in the subdue system. In: KDD Workshop AAAI, pp. 169–180 (1994)
42. Horváth, T., Bringmann, B., De Raedt, L.: Frequent hypergraph mining. In: Muggleton, S.H., Otero, R., Tamaddoni-Nezhad, A. (eds.) ILP 2006. LNCS (LNAI), vol. 4455, pp. 244–259. Springer, Heidelberg (2007)
43. Horváth, T., Otaki, K., Ramon, J.: Efficient frequent connected induced subgraph mining in graphs of bounded tree-width. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part I. LNCS, vol. 8188, pp. 622–637. Springer, Heidelberg (2013)
44. Huan, J., Wang, W., Prins, J., Yang, J.: Spin: mining maximal frequent subgraphs from graph databases. In: Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 581–586. ACM (2004)
45. Jaroszewicz, S.: Polynomial association rules with applications to logistic regression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 586–591. ACM (2006)
46. Koh, J.-L., Yo, P.-W.: An efficient approach for mining fault-tolerant frequent patterns based on bit vector representations. In: Zhou, L., Ooi, B.-C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 568–575. Springer, Heidelberg (2005)
47. Kontonasios, K.-N., Vreeken, J., De Bie, T.: Maximum entropy modelling for assessing results on real-valued data. In: Proceedings IEEE International Conference on Data Mining (ICDM), pp. 350–359 (2011)
48. Lam, H.T., Mörchen, F., Fradkin, D., Calders, T.: Mining compressing sequential patterns. In: Proceedings of the SIAM International Conference on Data Mining (SDM) (2012)
49. Lam, H.T., Mörchen, F., Fradkin, D., Calders, T.: Mining compressing sequential patterns. Stat. Anal. Data Min. (2013) doi:10.1002/sam.11192
50. Li, W., Han, J., Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In: Proceedings IEEE International Conference on Data Mining (ICDM), pp. 369–376. IEEE (2001)
51. Liu, J., Paulsen, S., Sun, X., Wang, W., Nobel, A.B., Prins, J.: Mining approximate frequent itemsets in the presence of noise: algorithm and analysis. In: Proceedings of the SIAM International Conference on Data Mining (SDM), pp. 405–416 (2006)
52. Liu, J., Wang, K., Fung, B.C.M.: Direct discovery of high utility itemsets without candidate generation. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 984–989. IEEE Computer Society (2012)
53. Mampaey, M., Vreeken, J., Tatti, N.: Summarizing data succinctly with the most informative itemsets. ACM Trans. Knowl. Discovery Data (TKDD) **6**(4), 16 (2012)

54. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Min. Knowl. Disc. **1**(3), 241–258 (1997)
55. Mooney, C.H., Roddick, J.F.: Sequential pattern mining-approaches and algorithms. ACM Comput. Surv. (CSUR), 45(2):19 (2013)
56. Novak, P.K., Lavrač, N., Webb, G.I.: Supervised descriptive rule discovery: a unifying survey of contrast set, emerging pattern and subgroup mining. J. Mach. Learn. Res. **10**, 377–403 (2009)
57. Novelli, N., Cicchetti, R.: Fun: an efficient algorithm for mining functional and embedded dependencies. In: Proceedings of the International Conference on Database Theory (ICDT), pp. 189–203. Springer (2001)
58. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Proceedings of the International Conference on Database Theory (ICDT), pp. 398–416 (1999)
59. Kosch, H., Sampaio, P.R.F., Hameurlain, A., Brunie, L.: Topic 05 parallel and distributed databases, data mining and knowledge discovery. In: Sakellariou, R., Keane, J.A., Gurd, J.R., Freeman, L. (eds.) Euro-Par 2001. LNCS, vol. 2150, p. 278. Springer, Heidelberg (2001)
60. Pei, J., Wang, H., Liu, J., Wang, K., Wang, J., Yu, P.S.: Discovering frequent closed partial orders from strings. IEEE Trans. Knowl. Data Eng. **18**(11), 1467–1481 (2006)
61. Poernomo, A.K., Gopalkrishnan, V.: Mining statistical information of frequent fault-tolerant patterns in transactional databases. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 272–281 (2007)
62. Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, Cambridge (2012)
63. Ramon, J., Miettinen, P., Vreeken, J.: Detecting bicliques in GF[q]. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part I. LNCS, vol. 8188, pp. 509–524. Springer, Heidelberg (2013)
64. Settles, B.: Active Learning Literature Survey. University of Wisconsin, Madison (2010)
65. Spyropoulou, E., De Bie, T.: Interesting multi-relational patterns. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 675–684. IEEE (2011)
66. Srikant, R., Agrawal, R.: Mining quantitative association rules in large relational tables. ACM SIGMOD Rec. **25**(2), 1–12 (1996)
67. Sun, Y., Han, J.: Mining heterogeneous information networks: principles and methodologies. Synth. Lect. Data Min. Knowl. Disc. **3**(2), 1–159 (2012)
68. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Pearson Education India, Philipine (2007)
69. Lam, H.T.: Pattern mining in data streams. Ph.D. thesis, Eindhoven University of Technology (2013)
70. Tsourakakis, C.E., Bonchi, F., Gionis, A., Gullo, F., Tsiarli, M.A.: Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, pp. 104–112. ACM (2013)
71. Vanetik, N., Shimony, S.E., Gudes, E.: Support measures for graph data. Data Min. Knowl. Disc. **13**(2), 243–260 (2006)
72. Vreeken, J., van Leeuwen, M., Siebes, A.: Krimp: mining itemsets that compress. Data Min. Knowl. Disc. **23**(1), 169–214 (2011)

73. Wang, J., Cheng, J., Fu, A.W.-C.: Redundancy-aware maximal cliques. In: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, pp. 122–130. ACM (2013)
74. Washio, T., Motoda, H.: State of the art of graph-based data mining. ACM SIGKDD Explor. Newsl. **5**(1), 59–68 (2003)
75. Wrobel, S.: An algorithm for multi-relational discovery of subgroups. In: Principles of Data Mining and Knowledge Discovery, pp. 78–87. Springer (1997)
76. Wu, C.-W., Lin, Y.-F., Yu, P.S., Tseng, V.S.: Mining high utility episodes in complex event sequences. In: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, pp. 536–544. ACM (2013)
77. Yang, C., Fayyad, U.M., Bradley, P.S.: Efficient discovery of error-tolerant frequent itemsets in high dimensions. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 194–203 (2001)
78. Zaki, M.J.: Sequence mining in categorical domains: incorporating constraints. In: Proceedings of the International Conference on Information and Knowledge Management (CIKM), pp. 422–429. ACM (2000)
79. Zaki, M.J.: Efficiently mining frequent trees in a forest. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80. ACM (2002)
80. Zaki, M.J.: Scalable algorithms for association mining. IEEE Trans. Knowl. Data Eng. **12**(3), 372–390 (2000)
81. Zhu, X.: Semi-supervised learning literature survey. Technical report 1530, Computer Science. University of Wisconsin-Madison (2005)

Process Mining in the Large: A Tutorial

Wil M.P. van der Aalst[1,2,3](✉)

[1] Department of Mathematics and Computer Science,
Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Business Process Management Discipline, Queensland University of Technology,
Brisbane, Australia
[3] International Laboratory of Process-Aware Information Systems,
National Research University Higher School of Economics, Moscow, Russia
w.m.p.v.d.aalst@tue.nl

Abstract. Recently, process mining emerged as a new scientific discipline on the interface between process models and event data. On the one hand, conventional Business Process Management (BPM) and Workflow Management (WfM) approaches and tools are mostly model-driven with little consideration for event data. On the other hand, Data Mining (DM), Business Intelligence (BI), and Machine Learning (ML) focus on data without considering end-to-end process models. Process mining aims to bridge the gap between BPM and WfM on the one hand and DM, BI, and ML on the other hand. Here, the challenge is to turn torrents of event data ("Big Data") into valuable insights related to process performance and compliance. Fortunately, process mining results can be used to identify and understand bottlenecks, inefficiencies, deviations, and risks. This tutorial paper introduces basic process mining techniques that can be used for process discovery and conformance checking. Moreover, some very general decomposition results are discussed. These allow for the decomposition and distribution of process discovery and conformance checking problems, thus enabling process mining in the large.

Keywords: Process mining · Big Data · Process discovery · Conformance checking

1 Introduction

Like most IT-related phenomena, also the growth of event data complies with Moore's Law. Similar to the number of transistors on chips, the capacity of hard disks, and the computing power of computers, the digital universe is growing exponentially and roughly doubling every 2 years [55,64]. Although this is not a new phenomenon, suddenly many organizations realize that increasing amounts of "Big Data" (in the broadest sense of the word) need to be used intelligently in order to compete with other organizations in terms of efficiency, speed and service. However, the goal is not to collect as much data as possible. The real challenge is to turn event data into valuable insights. Only *process mining* techniques directly relate event data to end-to-end business processes [2]. Existing

E. Zimányi (Ed.): eBISS 2013, LNBIP 172, pp. 33–76, 2014.
DOI: 10.1007/978-3-319-05461-2_2,

business process modeling approaches generating piles of process models are typically disconnected from the real processes and information systems. Data-oriented analysis techniques (e.g., data mining and machines learning) typically focus on simple classification, clustering, regression, or rule-learning problems (Fig. 1).

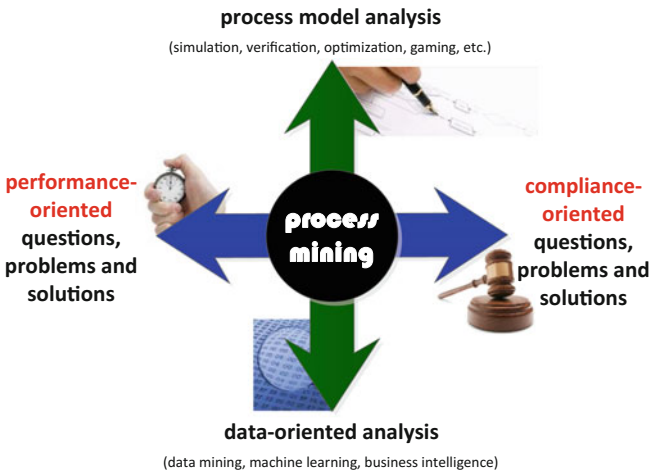


Fig. 1. Process mining provides the missing link between on the one hand process model analysis and data-oriented analysis and on the other hand performance and conformance.

Process mining aims to *discover, monitor and improve real processes by extracting knowledge from event logs* readily available in today's information systems [2]. Starting point for any process mining task is an *event log*. Each event in such a log refers to an *activity* (i.e., a well-defined step in some process) and is related to a particular *case* (i.e., a *process instance*). The events belonging to a case are *ordered* and can be seen as one "run" of the process. The sequence of activities executed for a case is called a *trace*. Hence, an event log can be viewed as a multiset of *traces* (multiple cases may have the same trace). It is important to note that an event log contains only *example behavior*, i.e., we cannot assume that all possible traces have been observed. In fact, an event log often contains only a fraction of the possible behavior [2].

The growing interest in process mining is illustrated by the *Process Mining Manifesto* [57] recently released by the *IEEE Task Force on Process Mining*. This manifesto is supported by 53 organizations and 77 process mining experts contributed to it.

The process mining spectrum is quite broad and includes techniques like process discovery, conformance checking, model repair, role discovery, bottleneck analysis, predicting the remaining flow time, and recommending next steps. In this paper, we focus on the following two main process mining problems:

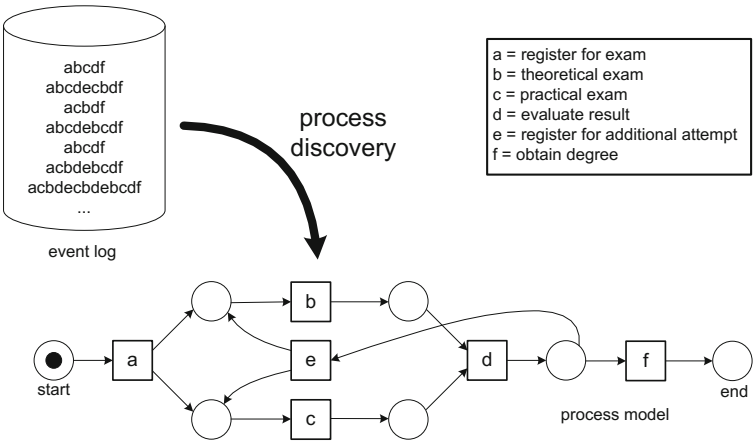


Fig. 2. Starting point for *process discovery* is an event log consisting of traces. In this example each trace describes activities related to an exam candidate. Based on the observed behavior a process model is inferred that is able to reproduce the event log. For example, both the traces in the event log and the runs of the process model always start with a (register for exam) and end with f (obtain degree). Moreover, a is always directly followed by b or c, b and c always happen together (in any order), d can only occur after both b and c have happened, d is always directly followed by e or f, etc. There are various process discovery techniques to automatically learn a process model from raw event data.

- *Process discovery problem*: Given an event log consisting of a collection of traces (i.e., sequences of events), construct a Petri net that "adequately" describes the observed behavior (see Fig. 2).[1]
- *Conformance checking problem*: Given an event log and a Petri net, diagnose the differences between the observed behavior (i.e., traces in the event log) and the modeled behavior (i.e., firing sequences of the Petri net). Figure 3 shows examples of deviations discovered through conformance checking.

Both problems are formulated in terms of Petri nets. However, any other process notation can be used, e.g., BPMN models, BPEL specifications, UML activity diagrams, Statecharts, C-nets, and heuristic nets.

The incredible growth of event data is also posing new challenges [84]. As event logs grow, process mining techniques need to become more efficient and highly scalable. Dozens of process discovery [2,19,21,26,28,32–34,50,52,65,85,92,93] and conformance checking [9,22,23,25,31,35,52,68,69,79,90] approaches have been proposed in literature. Despite the growing maturity of these approaches, the quality and efficiency of existing techniques leave much to be desired. State-of-the-art techniques still have problems dealing with large and/or

[1] As will be shown later, there are different ways of measuring the quality of a process discovery result. The term "adequately" is just an informal notion that will be detailed later.

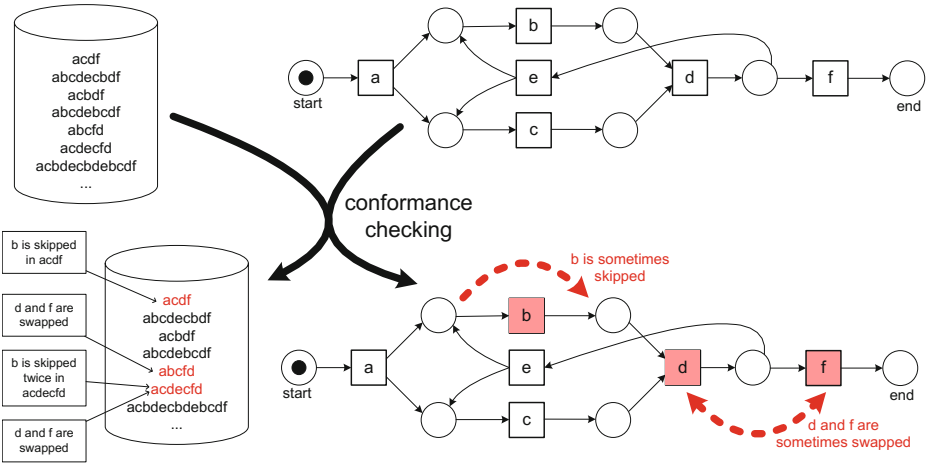


Fig. 3. *Conformance checking* starts with an event log and a process model. Ideally, events in the event log correspond to occurrences of activities in the model. By replaying the traces on the model one can find differences between log and model. The first trace shown cannot be replayed because activity b is missing (although no theoretical exam was made the result was evaluated). The fifth trace cannot be replayed because d and f are swapped, i.e., the candidate obtained a degree before the formal decision was made). The sixth trace has both problems. Conformance checking results can be diagnosed using a log-based view (bottom left) or a model-based view (bottom right).

complex event logs and process models. Consider for example Philips Healthcare, a provider of medical systems that are often connected to the Internet to enable logging, maintenance, and remote diagnostics. More than 1500 Cardio Vascular (CV) systems (i.e., X-ray machines) are remotely monitored by Philips. On average each CV system produces 15,000 events per day, resulting in 22.5 million events per day for just their CV systems. The events are stored for many years and have many attributes. The error logs of ASML's lithography systems have similar characteristics and also contain about 15,000 events per machine per day. These numbers illustrate the fact that today's organizations are already storing terabytes of event data. Process mining techniques aiming at very precise results (e.g., guarantees with respect to the accuracy of the model or diagnostics), quickly become intractable when dealing with such real-life event logs. Earlier applications of process mining in organizations such as Philips and ASML, show that there are various challenges with respect to performance (response times), capacity (storage space), and interpretation (discovered process models may be composed of thousands of activities) [29]. Therefore, we also describe the generic divide and conquer approach presented in [7]:

– For *conformance checking*, we decompose the process model into smaller partly overlapping model fragments. If the decomposition is done properly, then any trace that fits into the overall model also fits all of the smaller model fragments

and vice versa. Hence, metrics such as the fraction of fitting cases can be computed by only analyzing the smaller model fragments.
– To decompose *process discovery*, we split the set of activities into a collection of partly overlapping activity sets. For each activity set, we project the log onto the relevant events and discover a model fragment. The different fragments are glued together to create an overall process model. Again it is guaranteed that all traces in the event log that fit into the overall model also fit into the model fragments and vice versa.

This explains the title of this tutorial: "Process Mining in the Large".

The remainder of this paper is organized as follows. Section 2 provides an overview of the process mining spectrum. Some basic notions are introduced in Sect. 3. Section 4 presents two process discovery algorithms: the α-algorithm (Sect. 4.1) and region-based process discovery (Sect. 4.2). Section 5 introduces two conformance checking techniques. Moreover, the different quality dimensions are discussed and the importance of *aligning* observed and modeled behavior is explained. Section 6 presents a very generic decomposition result showing that most process discovery and conformance checking can be split into many smaller problems. Section 7 concludes the paper.

2 Process Mining Spectrum

Figure 4 shows the *process mining framework* described in [2]. The top of the diagram shows an external "world" consisting of business processes, people, and organizations supported by some information system. The information system records information about this "world" in such a way that events logs can be extracted. The term *provenance* used in Fig. 4 emphasizes the systematic, reliable, and trustworthy recording of events. The term provenance originates from scientific computing, where it refers to the data that is needed to be able to reproduce an experiment [39,61]. *Business process provenance* aims to systematically collect the information needed to reconstruct what has actually happened in a process or organization [37]. When organizations base their decisions on event data it is essential to make sure that these describe history well. Moreover, from an auditing point of view it is necessary to ensure that event logs cannot be tampered with. Business process provenance refers to the set of activities needed to ensure that history, as captured in event logs, "cannot be rewritten or obscured" such that it can serve as a reliable basis for process improvement and auditing.

As shown in Fig. 4, event data can be partitioned into *"pre mortem"* and *"post mortem"* event logs. "Post mortem" event data refer to information about cases that have completed, i.e., these data can be used for process improvement and auditing, but not for influencing the cases they refer to. "Pre mortem" event data refer to cases that have not yet completed. If a case is still running, i.e., the case is still "alive" (pre mortem), then it may be possible that information in the event log about this case (i.e., current data) can be exploited to ensure the correct or efficient handling of this case.

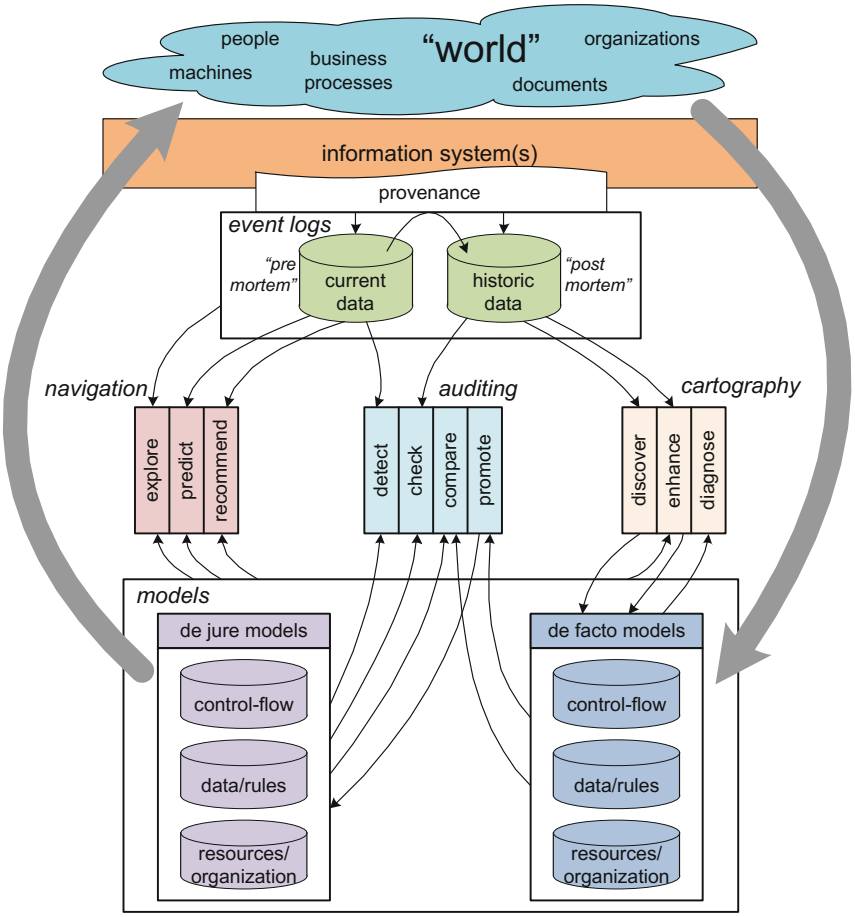


Fig. 4. Overview of the process mining spectrum [2].

"Post mortem" event data are most relevant for *off-line process mining*, e.g., discovering the control-flow of a process based on one year of event data. For *online process mining* mixtures of "pre mortem" (current) and "post mortem" (historic) data are needed. For example, historic information can be used to learn a predictive model. Subsequently, information about a running case is combined with the predictive model to provide an estimate for the remaining flow time of the case.

The process mining framework described in [2] also distinguishes between two types of models: *"de jure models"* and *"de facto models"*. *A de jure model is normative*, i.e., it specifies how things should be done or handled. For example, a process model used to configure a BPM system is normative and forces people to work in a particular way. *A de facto model is descriptive* and its goal is not to steer or control reality. Instead, de facto models aim to capture reality.

As shown in Fig. 4 both de jure and de facto models may cover multiple perspectives including the *control-flow perspective* ("How?"), the *organizational perspective* ("Who?"), and the *case perspective* ("What?"). The control-flow perspective describes the ordering of activities. The organizational perspective describes resources (worker, machines, customers, services, etc.) and organizational entities (roles, departments, positions, etc.). The case perspective describes data and rules.

In the middle of Fig. 4 ten process mining related activities are depicted. These ten activities are grouped into three categories: *cartography*, *auditing*, and *navigation*. The activities in the cartography category aim at making "process maps". The activities in the auditing category all involve a de jure model that is confronted with reality in the form of event data or a de facto model. The activities in the navigation category aim at improving a process while it is running.

Activity *discover* in Fig. 4 aims to learn a process model from examples stored in some event log. Lion's share of process mining research has been devoted to this activity [2,47]. A discovery technique takes an event log and produces a model without using any additional a-priori information. An example is the α-algorithm [21] that takes an event log and produces a Petri net explaining the behavior recorded in the log. If the event log contains information about resources, one can also discover resource-related models, e.g., a social network showing how people work together in an organization.

Since the mid-nineties several groups have been working on techniques for process discovery [18,21,26,34,38,44,45,92]. In [12] an overview is given of the early work in this domain. The idea to apply process mining in the context of workflow management systems was introduced in [26]. In parallel, Datta [38] looked at the discovery of business process models. Cook et al. investigated similar issues in the context of software engineering processes [34]. Herbst [54] was one of the first to tackle more complicated processes, e.g., processes containing duplicate tasks.

Most of the classical approaches have problems dealing with concurrency. The α-algorithm [21] is an example of a simple technique that takes concurrency as a starting point. However, this simple algorithm has problems dealing with complicated routing constructs and noise (like most of the other approaches described in literature). Process discovery is very challenging because techniques need to balance four criteria: *fitness* (the discovered model should allow for the behavior seen in the event log), *precision* (the discovered model should not allow for behavior completely unrelated to what was seen in the event log), *generalization* (the discovered model should generalize the example behavior seen in the event log), and *simplicity* (the discovered model should be as simple as possible). This makes process discovery a challenging and highly relevant research topic.

Activity *enhance* in Fig. 4 corresponds any effort where event data are used to repair a model (e.g., to better reflect reality) or to extend a model (e.g., to show bottlenecks). When existing process models (either discovered or hand-made) can be related to events logs, it is possible to enhance these models. The connection can be used to repair models [49] or to extend them [78,80–82].

Activity *diagnose* in Fig. 4 does not directly use event logs and focuses on classical model-based process analysis, e.g., simulation or verification.

Activity *detect* compares de jure models with current "pre mortem" data (events of running process instances) with the goal to detect deviations at runtime. The moment a predefined rule is violated, an alert is generated [17,62,63].

Activity *check* in Fig. 4 analyzes conformance-related questions using event data. Historic "post mortem" data can be cross-checked with de jure models. The goal of this activity is to pinpoint deviations and quantify the level of compliance. Various conformance checking techniques have been proposed in literature [9,22, 23,31,35,52,68,69,79,90]. For example, in [79] the fitness of a model is computed by comparing the number of missing and remaining tokens with the number of consumed and produced tokens during replay. The more sophisticated technique described in [9,22,23] creates as so-called *alignment* which relates a trace in the event log to an execution sequence of the model that is as similar as possible. Ideally, the alignment consists of steps where log and model agree on the activity to be executed. Steps where just the model "makes a move" or just the log "makes a move" have a predefined penalty. This way the computation of fitness can be turned into an optimization problem: for each trace in the event log an alignment with the lowest costs is selected. The resulting alignments can be used for all kinds of analysis since any trace in the event log is related to an execution sequence of the model. For example, timestamps in the model can be used to compute bottlenecks and extend the model with performance information (see activity *enhance* in Fig. 4).

Activity *compare* highlights differences and commonalities between a de jure model and a de facto model. Traditional equivalence notions such as trace equivalence, bisimilarity, and branching bisimilarity [51,67] can only be used to determine equivalence using a predefined equivalence notion, e.g., these techniques cannot be used to distinguish between very similar and highly dissimilar processes. Other notions such a graph-edit distance tend to focus on the syntax rather than the behavior of models. Therefore, recent BPM research explored various alternative similarity notions [42,43,58,59,66,91]. Also note the *Greatest Common Divisor* (GCD) and *Least Common Multiple* (LCM) notions defined for process models in [11]. The GCD captures the common parts of two or more models. The LCM embeds all input models. We refer to [42] for a survey and empirical evaluation of some similarity notions.

Activity *promote* takes (parts of) de facto models and converts these into (parts of) de jure models, i.e., models used to control or support processes are improved based on models learned from event data. By promoting proven "best practices" to de jure models, existing processes can be improved.

The activities in the cartography and auditing categories in Fig. 4 can be viewed as "backward-looking". The last three activities forming the navigation category are "forward-looking" and are sometimes referred to as *operational support* [2]. For example, process mining techniques can be used to make predictions about the future of a particular case and guide the user in selecting suitable actions. When comparing this with a car navigation system from TomTom or

Garmin, this corresponds to functionalities such as predicting the arrival time and guiding the driver using spoken instructions.

Activity *explore* in Fig. 4 visualizes running cases and compares these cases with similar cases that were handled earlier. The combination of event data and models can be used to explore business processes at run-time and, if needed, trigger appropriate actions.

By combining information about running cases with models (discovered or hand-made), it is possible to make predictions about the future, e.g., predicting the remaining flow time or the probability of success. Figure 4 shows that activity *predict* uses current data and models (often learned over historic data). Various techniques have been proposed in BPM literature [20,46,76]. Note that already a decade ago Staffware provided a so-called "prediction engine" using simulation [86].

Activity *recommend* in Fig. 4 aims to provide functionality similar to the guidance given by car navigation systems. The information used for predicting the future can also be used to recommend suitable actions (e.g. to minimize costs or time) [17,83]. Given a set of possible next steps, the most promising step is recommended. For each possible step, simply assume that the step is made and predict the resulting performance (e.g., remaining flow time). The resulting predictions can be compared and used to rank the possible next steps.

The ten activities in Fig. 4 illustrate that process mining extends far beyond process discovery. The increasing availability and growing volume of event data suggest that the importance of process mining will continue to grow in the coming years.

It is impossible to cover the whole process mining spectrum in this tutorial paper. The reader is referred to [2,6] for a more complete overview.

3 Preliminaries

This section introduces basic concepts related to Petri nets and event logs.

3.1 Multisets, Functions, and Sequences

Multisets are used to represent the state of a Petri net and to describe event logs where the same trace may appear multiple times.

$\mathcal{B}(A)$ is the set of all multisets over some set A. For some multiset $B \in \mathcal{B}(A)$, $B(a)$ denotes the number of times element $a \in A$ appears in B. Some examples: $B_1 = [\,]$, $B_2 = [x, x, y]$, $B_3 = [x, y, z]$, $B_4 = [x, x, y, x, y, z]$, $B_5 = [x^3, y^2, z]$ are multisets over $A = \{x, y, z\}$. B_1 is the empty multiset, B_2 and B_3 both consist of three elements, and $B_4 = B_5$, i.e., the ordering of elements is irrelevant and a more compact notation may be used for repeating elements.

The standard set operators can be extended to multisets, e.g., $x \in B_2$, $B_2 \uplus B_3 = B_4$, $B_5 \setminus B_2 = B_3$, $|B_5| = 6$, etc. $\{a \in B\}$ denotes the set with all elements a for which $B(a) \geq 1$. $[f(a) \mid a \in B]$ denotes the multiset where element $f(a)$ appears $\sum_{x \in B | f(x) = f(a)} B(x)$ times.

A relation $R \subseteq X \times Y$ is a set of pairs. $\pi_1(R) = \{x \mid (x,y) \in R\}$ is the domain of R, $\pi_2(R) = \{y \mid (x,y) \in R\}$ is the range of R, and $\omega(R) = \pi_1(R) \cup \pi_2(R)$ are the elements of R. For example, $\omega(\{(a,b),(b,c)\}) = \{a,b,c\}$.

To the (total) function $f \in X \rightarrow Y$ maps elements from the set X onto elements of the set Y, i.e., $f(x) \in Y$ for any $x \in X$. $f \in X \not\rightarrow Y$ is a partial function with domain $dom(f) \subseteq X$ and range $rng(f) = \{f(x) \mid x \in X\} \subseteq Y$. $f \in X \rightarrow Y$ is a total function, i.e., $dom(f) = X$. A partial function $f \in X \not\rightarrow Y$ is injective if $f(x_1) = f(x_2)$ implies $x_1 = x_2$ for all $x_1, x_2 \in dom(f)$.

Definition 1 (Function Projection). *Let $f \in X \not\rightarrow Y$ be a (partial) function and $Q \subseteq X$. $f \restriction_Q$ is the function projected on Q: $dom(f \restriction_Q) = dom(f) \cap Q$ and $f \restriction_Q (x) = f(x)$ for $x \in dom(f \restriction_Q)$.*

The projection can also be used for multisets, e.g., $[x^3, y, z^2] \restriction_{\{x,y\}} = [x^3, y]$.

$\sigma = \langle a_1, a_2, \ldots, a_n \rangle \in X^*$ denotes a sequence over X of length n. $\langle\,\rangle$ is the empty sequence. $mults^k(\sigma) = [a_1, a_2, \ldots, a_k]$ is the multiset composed of the first k elements of σ. $mults(\sigma) = mults^{|\sigma|}(\sigma)$ converts a sequence into a multiset. $mults^2(\langle a,a,b,a,b \rangle) = [a^2]$ and $mults(\langle a,a,b,a,b \rangle) = [a^3, b^2]$.

Sequences are used to represent paths in a graph and traces in an event log. $\sigma_1 \cdot \sigma_2$ is the concatenation of two sequences and $\sigma \restriction_Q$ is the projection of σ on Q.

Definition 2 (Sequence Projection). *Let X be a set and $Q \subseteq X$ one of its subsets. $\restriction_Q \in X^* \rightarrow Q^*$ is a projection function and is defined recursively: (1) $\langle\,\rangle \restriction_Q = \langle\,\rangle$ and (2) for $\sigma \in X^*$ and $x \in X$:*

$$(\langle x \rangle \cdot \sigma) \restriction_Q = \begin{cases} \sigma \restriction_Q & \text{if } x \notin Q \\ \langle x \rangle \cdot \sigma \restriction_Q & \text{if } x \in Q \end{cases}$$

So $\langle y,z,y \rangle \restriction_{\{x,y\}} = \langle y,y \rangle$. Functions can also be applied to sequences: if $dom(f) = \{x,y\}$, then $f(\langle y,z,y \rangle) = \langle f(y), f(y) \rangle$.

Definition 3 (Applying Functions to Sequences). *Let $f \in X \not\rightarrow Y$ be a partial function. f can be applied to sequences of X using the following recursive definition (1) $f(\langle\,\rangle) = \langle\,\rangle$ and (2) for $\sigma \in X^*$ and $x \in X$:*

$$f(\langle x \rangle \cdot \sigma) = \begin{cases} f(\sigma) & \text{if } x \notin dom(f) \\ \langle f(x) \rangle \cdot f(\sigma) & \text{if } x \in dom(f) \end{cases}$$

Summation is defined over multisets and sequences, e.g., $\sum_{x \in \langle a,a,b,a,b \rangle} f(x) = \sum_{x \in [a^3,b^2]} f(x) = 3f(a) + 2f(b)$.

3.2 Petri Nets

We use Petri nets as the process modeling language used to introduce process mining (in the large). However, as mentioned in Sect. 1 the results presented in the paper can be adapted for various other process modeling notations (BPMN

models, BPEL specifications, UML activity diagrams, Statecharts, C-nets, heuristic nets, etc.). This does not imply that these notations are equivalent. There are differences in *expensiveness* (e.g., classical Petri nets are not Turing complete, but most extension of Petri nets are) and *suitability* (cf. research on the workflow patterns [15]). Translations are often "lossy", i.e., the model after translation may allow for more or less behavior. However, in practice this is not a problem as the basic concepts are often the same. There is also a trade-off between accuracy and simplicity. For example, inclusive OR-joins are not directly supported by Petri nets, because an OR-join may need to synchronize a variable (at design-time unknown) number of inputs. Using a rather involved translation it is possible to model this in terms of classical Petri nets using so-called "true" and "false" tokens [15]. This only works if there are no arbitrary unstructured loops. See for example the many translations proposed for the mapping from BPEL to Petri nets [56,60,72]. There also exists a naïve much simpler translation that includes the original behavior (but also more) [1,10]. Using Single-Entry Single-Exit (SESE) components and the refined process structure tree (RPST) [74,87] it is often possible to convert aan unstructured graph-based model into a structured model. Also see the approaches to convert Petri nets and BPMN models into BPEL [16,73].

The above examples illustrate that many conversions are possible depending on the desired outcome (accuracy versus simplicity). It is also important to stress that the representational bias used during process discovery may be different from the representational bias used to present the result to end-users. For example, one may use Petri nets during discovery and convert the final result to BPMN.

In this paper we would like to steer away from notational issues and conversions and restrict ourselves to Petri nets as a representation for process models. By using Petri nets we minimize the notational overhead allowing us the focus on the key ideas.

Definition 4 (Petri Net). *A Petri net is a tuple $N = (P, T, F)$ with P the set of places, T the set of transitions, $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ the flow relation.*

Figure 5 shows a Petri net $N = (P, T, F)$ with $P = \{start, c1, \ldots, c9, end\}$, $T = \{t1, t2, \ldots, t11\}$, and $F = \{(start, t1), (t1, c1), (t1, c2), \ldots, (t11, end)\}$. The state of a Petri net, called *marking*, is a multiset of places indicating how many *tokens* each place contains. $[start]$ is the initial marking shown in Fig. 5. Another potential marking is $[c1^{10}, c2^5, c4^5]$. This is the state with ten tokens in $c1$, five tokens in $c2$, and five tokens in $c4$.

Definition 5 (Marking). *Let $N = (P, T, F)$ be a Petri net. A marking M is a multiset of places, i.e., $M \in \mathcal{B}(P)$.*

A Petri net $N = (P, T, F)$ defines a directed graph with nodes $P \cup T$ and edges F. For any $x \in P \cup T$, $\overset{N}{\bullet} x = \{y \mid (y, x) \in F\}$ denotes the set of input nodes and

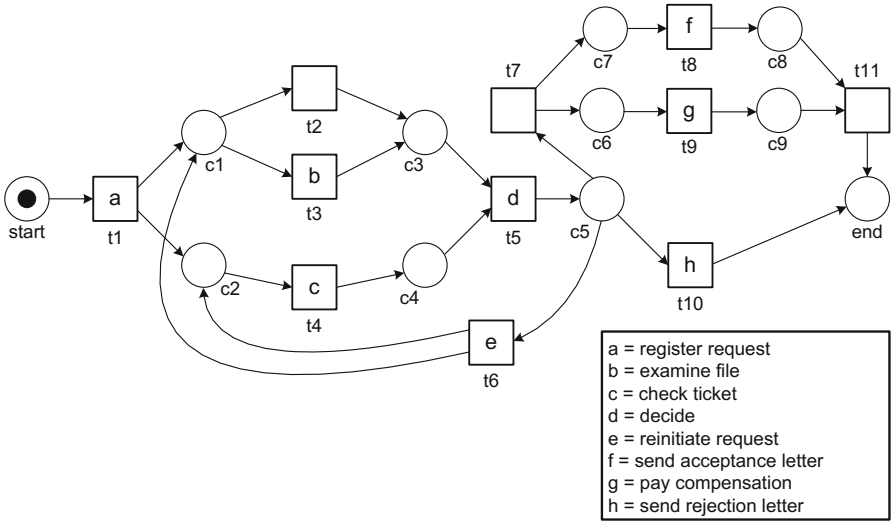


Fig. 5. A labeled Petri net.

$x \overset{N}{\bullet} = \{y \mid (x,y) \in F\}$ denotes the set of output nodes We drop the superscript N if it is clear from the context.

A transition $t \in T$ is *enabled* in marking M of net N, denoted as $(N,M)[t\rangle$, if each of its input places $\bullet t$ contains at least one token. Consider the Petri net N in Fig. 5 with $M = [c3, c4]$: $(N,M)[t5\rangle$ because both input places of $t5$ are marked.

An enabled transition t may *fire*, i.e., one token is removed from each of the input places $\bullet t$ and one token is produced for each of the output places $t\bullet$. Formally: $M' = (M \setminus \bullet t) \uplus t\bullet$ is the marking resulting from firing enabled transition t in marking M of Petri net N. $(N,M)[t\rangle(N,M')$ denotes that t is enabled in M and firing t results in marking M'. For example, $(N,[start])[t1\rangle(N,[c1,c2])$ and $(N,[c3,c4])[t5\rangle(N,[c5])$ for the net in Fig. 5.

Let $\sigma = \langle t_1, t_2, \ldots, t_n\rangle \in T^*$ be a sequence of transitions. $(N,M)[\sigma\rangle(N,M')$ denotes that there is a set of markings $M_0, M_1, \ldots, M_n$ such that $M_0 = M$, $M_n = M'$, and $(N,M_i)[t_{i+1}\rangle(N,M_{i+1})$ for $0 \leq i < n$. A marking M' is *reachable* from M if there exists a σ such that $(N,M)[\sigma\rangle(N,M')$. For example, $(N,[start])[\sigma\rangle(N,[end])$ with $\sigma = \langle t1, t3, t4, t5, t10\rangle$ for the net in Fig. 5.

Definition 6 (Labeled Petri Net). *A labeled Petri net $N = (P,T,F,l)$ is a Petri net (P,T,F) with labeling function $l \in T \not\to \mathcal{U}_A$ where $\mathcal{U}_A$ is some universe of activity labels. Let $\sigma_v = \langle a_1, a_2, \ldots, a_n\rangle \in {\mathcal{U}_A}^*$ be a sequence of activities. $(N,M)[\sigma_v \rhd (N,M')$ if and only if there is a sequence $\sigma \in T^*$ such that $(N,M)[\sigma\rangle(N,M')$ and $l(\sigma) = \sigma_v$ (cf. Definition 3).*

If $t \notin dom(l)$, it is called invisible. An occurrence of visible transition $t \in dom(l)$ corresponds to observable activity $l(t)$. The Petri net in Fig. 5 is labeled. The

labeling function is defined as follows: $dom(l) = \{t1, t3, t4, t5, t6, t8, t9, t10\}$, $l(t1) = a$ (a is a shorthand for "register request"), $l(t3) = b$ ("examine file"), $l(t4) = c$ ("check ticket"), $l(t5) = d$ ("decide"), $l(t6) = e$ ("reinitiate request"), $l(t8) = f$ ("send acceptance letter"), $l(t9) = g$ ("pay compensation"), and $l(t10) = h$ ("send rejection letter"). Unlabeled transitions correspond to so-called "silent actions", i.e., transitions $t2$, $t7$, and $t11$ are unobservable.

Given the Petri net N in Fig. 5: $(N, [start])[\sigma_v \rhd (N, [end])$ for $\sigma_v = \langle a, c, d, f, g \rangle$ because $(N, [start])[\sigma\rangle(N, [end])$ with $\sigma = \langle t1, t2, t4, t5, t7, t8, t9, t11 \rangle$ and $l(\sigma) = \sigma_v$.

In the context of process mining, we always consider processes that start in an initial state and end in a well-defined end state. For example, given the net in Fig. 5 we are interested in so-called *complete* firing sequences starting in $M_{init} = [start]$ and ending in $M_{final} = [end]$. Therefore, we define the notion of a *system net*.

Definition 7 (System Net). *A system net is a triplet $SN = (N, M_{init}, M_{final})$ where $N = (P, T, F, l)$ is a labeled Petri net, $M_{init} \in \mathcal{B}(P)$ is the initial marking, and $M_{final} \in \mathcal{B}(P)$ is the final marking. $\mathcal{U}_{SN}$ is the* universe of system nets.

Definition 8 (System Net Notations). *Let $SN = (N, M_{init}, M_{final}) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$.*

- $T_v(SN) = dom(l)$ *is the set of* visible transitions *in SN,*
- $A_v(SN) = rng(l)$ *is the set of corresponding* observable activities *in SN,*
- $T_v^u(SN) = \{t \in T_v(SN) \mid \forall_{t' \in T_v(SN)}\ l(t) = l(t') \Rightarrow t = t'\}$ *is the set of* unique *visible transitions in SN (i.e., there are no other transitions having the same visible label), and*
- $A_v^u(SN) = \{l(t) \mid t \in T_v^u(SN)\}$ *is the set of corresponding* unique *observable activities in SN.*

Given a system net, $\phi(SN)$ is the set of all possible *visible* traces, i.e., complete firing sequences starting in M_{init} and ending in M_{final} projected onto the set of observable activities.

Definition 9 (Traces). *Let $SN = (N, M_{init}, M_{final}) \in \mathcal{U}_{SN}$ be a system net. $\phi(SN) = \{\sigma_v \mid (N, M_{init})[\sigma_v \rhd (N, M_{final})\}$ is the set of* visible *traces starting in M_{init} and ending in M_{final}. $\phi_f(SN) = \{\sigma \mid (N, M_{init})[\sigma\rangle(N, M_{final})\}$ is the corresponding set of complete firing sequences.*

For Fig. 5: $\phi(SN) = \{\langle a, c, d, f, g \rangle, \langle a, c, b, d, f, g \rangle, \langle a, c, d, h \rangle, \langle a, b, c, d, e, c, d, h \rangle, \ldots\}$ and $\phi_f(SN) = \{\langle t1, t2, t4, t5, t7, t8, t9, t11 \rangle, \langle t1, t3, t4, t5, t10 \rangle, \ldots\}$. Because of the loop involving transition $t6$ there are infinitely many visible traces and complete firing sequences.

Traditionally, the bulk of Petri net research focused on *model-based analysis*. Moreover, the largest proportion of model-based analysis techniques is limited to *functional properties*. Generic techniques such as model checking can be used to check whether a Petri net has particular properties, e.g., free of deadlocks. Petri-net-specific notions such as traps, siphons, place invariants, transition invariants,

and coverability graphs are often used to verify desired functional properties, e.g., liveness or safety properties [77]. Consider for example the notion of *soundness* defined for *WorkFlow nets* (WF-nets) [13]. The Petri net shown in Fig. 5 is a WF-net because there is a unique source place *start*, a unique sink place *end*, and all nodes are on a path from *start* to *end*. A WF-net is sound if and only if the following three requirements are satisfied: (1) *option to complete*: it is always still possible (i.e., from any reachable marking) to reach the state which just marks place *end*, (2) *proper completion*: if place *end* is marked all other places are empty, and (3) *no dead transitions*: it should be possible to execute an arbitrary transition by following the appropriate route through the WF-net. The WF-net in Fig. 5 is sound and as a result cases cannot get stuck before reaching the end (termination is always possible) and all parts of the process can be activated (no dead segments). Obviously, soundness is important in the context of business processes and process mining. Fortunately, there exist nice theorems connecting soundness to classical Petri-net properties. For example, a WF-net is sound if and only if the corresponding short-circuited Petri net is live and bounded. Hence, proven techniques and tools can be used to verify soundness.

Although the results in this paper are more general and not limited of WF-nets, all examples in this paper use indeed WF-nets. As indicated most of Petri net literature and tools focuses on model-based analysis thereby ignoring actual observed process behavior. Yet, the confrontation between modeled and observed behavior is essential for understanding and improving real-life processes and systems.

3.3 Event Log

As indicated earlier, *event logs* serve as the starting point for process mining. An event log is a multiset of *traces*. Each trace describes the life-cycle of a particular *case* (i.e., a *process instance*) in terms of the *activities* executed.

Definition 10 (Trace, Event Log). *Let $A \subseteq \mathcal{U}_A$ be a set of activities. A trace $\sigma \in A^*$ is a sequence of activities. $L \in \mathcal{B}(A^*)$ is an event log, i.e., a multiset of traces.*

An event log is a *multiset* of traces because there can be multiple cases having the same trace. In this simple definition of an event log, an event refers to just an *activity*. Often event logs store additional information about events. For example, many process mining techniques use extra information such as the *resource* (i.e., person or device) executing or initiating the activity, the *timestamp* of the event, or *data elements* recorded with the event (e.g., the size of an order). In this paper, we abstract from such information. However, the results presented can easily be extended to event logs containing additional information.

An example log is $L_1 = [\langle a, c, d, f, g\rangle^{10}, \langle a, c, d, h\rangle^5, \langle a, b, c, d, e, c, d, g, f\rangle^5]$. L_1 contains information about 20 cases, e.g., 10 cases followed trace $\langle a, c, d, f, g\rangle$. There are $10 \times 5 + 5 \times 4 + 5 \times 9 = 115$ events in total.

The projection function $\restriction_X$ (cf. Definition 2) is generalized to event logs, i.e., for some event log $L \in \mathcal{B}(A^*)$ and set $X \subseteq A$: $L \restriction_X = [\sigma \restriction_X | \ \sigma \in L]$. For example, $L_1 \restriction_{\{a,g,h\}} = [\langle a, g\rangle^{15}, \langle a, h\rangle^5]$. We will refer to these projected event logs as *sublogs*.

4 Process Discovery

Process discovery is one of the most challenging process mining tasks. In this paper we consider the basic setting where we want to learn a system net $SN = (N, M_{init}, M_{final}) \in \mathcal{U}_{SN}$ from an event log $L \in \mathcal{B}(A^*)$. We will present two process discovery techniques: the α-algorithm and an approach based on language-based regions. These techniques have many limitations (e.g., unable to deal with noise), but they serve as a good starting point for better understanding this challenging topic.

4.1 Alpha Algorithm

First we describe the α-algorithm [21]. This was the first process discovery technique able to discover concurrency. Moreover, unlike most other techniques, the α-algorithm was proven to be correct for a clearly defined class of processes [21]. Nevertheless, we would like to stress that the basic algorithm has many limitations including the inability to deal with noise, particular loops, and non-free-choice behavior. Yet, it provides a good introduction into the topic. The α-algorithm is simple and many of its ideas have been embedded in more complex and robust techniques. We will use the algorithm as a baseline for discussing the challenges related to process discovery.

The α-algorithm *scans the event log for particular patterns*. For example, if activity a is followed by b but b is never followed by a, then it is assumed that there is a causal dependency between a and b.

Definition 11 (Log-based ordering relations). *Let $L \in \mathcal{B}(A^*)$ be an event log over A, i.e., $L \in \mathcal{B}(A^*)$. Let $a, b \in A$:*

- *$a >_L b$ if and only if there is a trace $\sigma = \langle t_1, t_2, t_3, \ldots t_n\rangle$ and $i \in \{1, \ldots, n-1\}$ such that $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$;*
- *$a \rightarrow_L b$ if and only if $a >_L b$ and $b \not>_L a$;*
- *$a \#_L b$ if and only if $a \not>_L b$ and $b \not>_L a$; and*
- *$a \|_L b$ if and only if $a >_L b$ and $b >_L a$.*

Consider for instance event log $L_2 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$. For this event log the following log-based ordering relations can be found.

$$>_{L_2} = \{(a, b), (a, c), (a, e), (b, c), (c, b), (b, d), (c, d), (e, d)\}$$
$$\rightarrow_{L_2} = \{(a, b), (a, c), (a, e), (b, d), (c, d), (e, d)\}$$
$$\#_{L_2} = \{(a, a), (a, d), (b, b), (b, e), (c, c), (c, e), (d, a), (d, d), (e, b), (e, c), (e, e)\}$$
$$\|_{L_2} = \{(b, c), (c, b)\}$$

Relation $>_{L_2}$ contains all pairs of activities in a "directly follows" relation. $c >_{L_2} d$ because d directly follows c in trace $\langle a, b, c, d \rangle$. However, $d \not>_{L_2} c$ because c never directly follows d in any trace in the log. $\rightarrow_{L_2}$ contains all pairs of activities in a "causality" relation, e.g., $c \rightarrow_{L_2} d$ because sometimes d directly follows c and never the other way around ($c >_{L_2} d$ and $d \not>_{L_2} c$). $b \|_{L_2} c$ because $b >_{L_2} c$ and $c >_{L_2} b$, i.e., sometimes c follows b and sometimes the other way around. $b \#_{L_2} e$ because $b \not>_{L_2} e$ and $e \not>_{L_2} b$.

For any log L over A and $x, y \in A$: $x \rightarrow_L y$, $y \rightarrow_L x$, $x \#_L y$, or $x \|_L y$, i.e., precisely one of these relations holds for any pair of activities. The log-based ordering relations can be used to *discover patterns* in the corresponding process model as is illustrated in Fig. 6. If a and b are in sequence, the log will show $a \rightarrow_L b$. If after a there is a choice between b and c, the log will show $a \rightarrow_L b$, $a \rightarrow_L c$, and $b \#_L c$ because a can be followed by b and c, but b will not be followed by c and vice versa. The logical counterpart of this so-called XOR-split pattern is the XOR-join pattern as shown in Fig. 6(b-c). If $a \rightarrow_L c$, $b \rightarrow_L c$, and $a \#_L b$, then this suggests that after the occurrence of either a or b, c should happen. Figure 6(d-e) shows the so-called AND-split and AND-join patterns. If $a \rightarrow_L b$, $a \rightarrow_L c$, and $b \|_L c$, then it appears that after a both b and c can be executed in parallel (AND-split pattern). If $a \rightarrow_L c$, $b \rightarrow_L c$, and $a \|_L b$, then the log suggests that c needs to synchronize a and b (AND-join pattern).

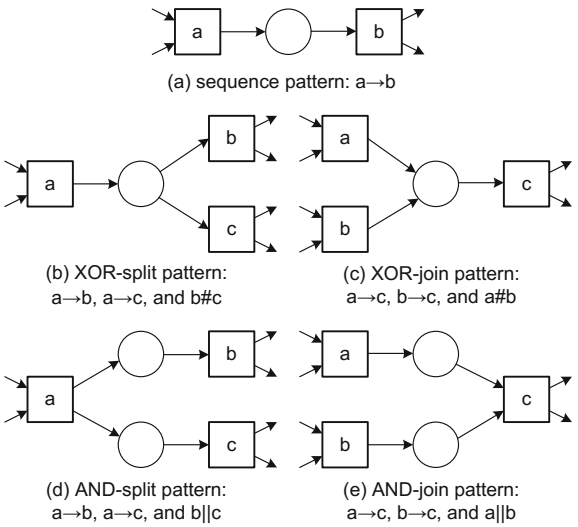


Fig. 6. Typical process patterns and the footprints they leave in the event log

Figure 6 only shows simple patterns and does not present the additional conditions needed to extract the patterns. However, it provides some initial insights useful when reading the formal definition of the α-algorithm [21].

Definition 12. (α-algorithm). *Let $L \in \mathcal{B}(A^*)$ be an event log over A. $\alpha(L)$ produces a system net and is defined as follows:*

1. $T_L = \{t \in A \mid \exists_{\sigma \in L}\ t \in \sigma\}$,
2. $T_I = \{t \in A \mid \exists_{\sigma \in L}\ t = first(\sigma)\}$,
3. $T_O = \{t \in A \mid \exists_{\sigma \in L}\ t = last(\sigma)\}$,
4. $X_L = \{(A, B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B}\ a \rightarrow_L b \wedge \forall_{a_1, a_2 \in A}\ a_1 \#_L a_2 \wedge \forall_{b_1, b_2 \in B}\ b_1 \#_L b_2\}$,
5. $Y_L = \{(A, B) \in X_L \mid \forall_{(A', B') \in X_L} A \subseteq A' \wedge B \subseteq B' \Longrightarrow (A, B) = (A', B')\}$,
6. $P_L = \{p_{(A,B)} \mid (A, B) \in Y_L\} \cup \{i_L, o_L\}$,
7. $F_L = \{(a, p_{(A,B)}) \mid (A, B) \in Y_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A, B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) \mid t \in T_I\} \cup \{(t, o_L) \mid t \in T_O\}$,
8. $l_L \in T_L \rightarrow A$ *with* $l(t) = t$ *for* $t \in T_L$, *and*
9. $\alpha(L) = (N, M_{init}, M_{final})$ *with* $N = (P_L, T_L, F_L, l_L)$, $M_{init} = [i_L]$, $M_{final} = [o_L]$.

In Step 1 it is checked which activities do appear in the log (T_L). These are the observed activities and correspond to the transitions of the generated system net. T_I is the set of start activities, i.e., all activities that appear first in some trace (Step 2). T_O is the set of end activities, i.e., all activities that appear last in some trace (Step 3). Steps 4 and 5 form the core of the α-algorithm. The challenge is to determine the places of the Petri net and their connections. We aim at constructing places named $p_{(A,B)}$ such that A is the set of input transitions ($\bullet p_{(A,B)} = A$) and B is the set of output transitions ($p_{(A,B)}\bullet = B$) of $p_{(A,B)}$.

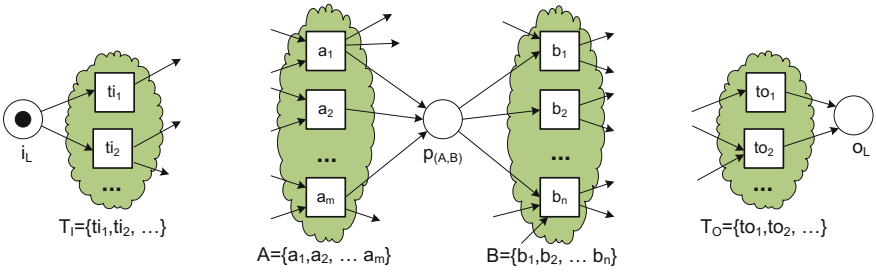


Fig. 7. Place $p_{(A,B)}$ connects the transitions in set A to the transitions in set B, i_L is the input place of all start transition T_I, and o_L is the output place of all end transition T_O.

The basic motivation for finding $p_{(A,B)}$ is illustrated by Fig. 7. All elements of A should have causal dependencies with all elements of B, i.e., for all $(a, b) \in A \times B$: $a \rightarrow_L b$. Moreover, the elements of A should never follow one another, i.e., for all $a_1, a_2 \in A$: $a_1 \#_L a_2$. A similar requirement holds for B. Let us consider $L_2 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$ again. Clearly, $A = \{a\}$ and $B = \{b, e\}$ meet the requirements stated in Step 4. Also $A' = \{a\}$ and $B' = \{b\}$ meet the

same requirements. X_L is the set of all such pairs that meet the requirements just mentioned. In this case:

$$X_{L_2} = \{(\{a\},\{b\}),(\{a\},\{c\}),(\{a\},\{e\}),(\{a\},\{b,e\}),(\{a\},\{c,e\}),\\ (\{b\},\{d\}),(\{c\},\{d\}),(\{e\},\{d\}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$$

If one would insert a place for any element in X_{L_2}, there would be too many places. Therefore, only the "maximal pairs" (A, B) should be included. Note that for any pair $(A, B) \in X_L$, non-empty set $A' \subseteq A$, and non-empty set $B' \subseteq B$, it is implied that $(A', B') \in X_L$. In Step 5, all non-maximal pairs are removed, thus yielding:

$$Y_{L_2} = \{(\{a\},\{b,e\}),(\{a\},\{c,e\}),(\{b,e\},\{d\}),(\{c,e\},\{d\})\}$$

Every element of $(A, B) \in Y_L$ corresponds to a place $p_{(A,B)}$ connecting transitions A to transitions B. In addition P_L also contains a unique source place i_L and a unique sink place o_L (cf. Step 6). In Step 7 the arcs of the Petri net are generated. All start transitions in T_I have i_L as an input place and all end transitions T_O have o_L as output place. All places $p_{(A,B)}$ have A as input nodes and B as output nodes. Figure 8 shows the resulting system net. Since transition identifiers and labels coincide ($l(t) = t$ for $t \in T_L$) we only show the labels. For any event log L, $\alpha(L) = (N, M_{init}, M_{final})$ with $N = (P_L, T_L, F_L, l_L)$, $M_{init} = [i_L]$, $M_{final} = [o_L]$ aims to describe the behavior recorded in L.

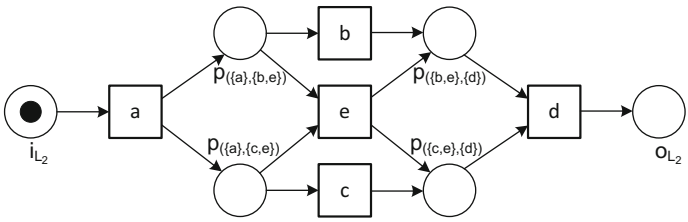


Fig. 8. System net $\alpha(L_2) = (N, [i_{L_2}], [o_{L_2}])$ for event log $L_2 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^2, \langle a, e, d\rangle]$.

Next, we consider the following three events logs $L_{3a} = [\langle a, c, d\rangle^{88}, \langle a, c, e\rangle^{82}, \langle b, c, d\rangle^{83}, \langle b, c, e\rangle^{87}]$, $L_{3b} = [\langle a, c, d\rangle^{88}, \langle b, c, e\rangle^{87}]$, $L_{3c} = [\langle a, c, d\rangle^{88}, \langle a, c, e\rangle^2, \langle b, c, d\rangle^3, \langle b, c, e\rangle^{87}]$. $\alpha(L_{3a}) = SN_{3a}$, i.e., the system net depicted in Fig. 9 without places p_3 and p_4 (modulo renaming of places). It is easy to check that all traces in L_{3a} are allowed by the discovered model SN_{3a} and that all firing sequences of the SN_{3a} appear in the event log. Now consider L_{3b}. Surprisingly, $\alpha(L_{3b}) = \alpha(L_{3a}) = SN_{3a}$ (modulo renaming of places). Note that event logs L_{3a} and L_{3b} are identical with respect to the "directly follows" relation, i.e., $>_{L_{3a}} = >_{L_{3b}}$. The α-algorithm is unable to discover SN_{3b} because the dependencies between on the one hand a and d and on the other hand c and e are

non-local: a, d, c and e never directly follow one another. Still, $\alpha(L_{3a})$ allows for all behavior in L_{3b} (and more). Sometimes it is not so clear which model is preferable. Consider for example L_{3c} where two traces are infrequent. SN_{3a} allows for all behavior in L_{3c}, including the infrequent ones. However, SN_{3b} is more precise as it only shows the "highways" in L_{3c}. Often people are interested in the "80/20 model", i.e., the process model that can describe 80 % of the behavior seen in the log. This model is typically relatively simple because the remaining 20 % of the log often account for 80 % of the variability in the process. Hence, people may prefer SN_{3b} over SN_{3a} for L_{3c}.

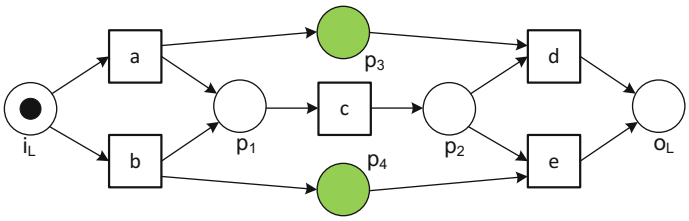


Fig. 9. $SN_{3a} = (N_{3a}, [i_L], [o_L])$ is the system net depicted *without* places p_3 and p_4. $SN_{3b} = (N_{3b}, [i_L], [o_L])$ is the same net but now *including* places p_3 and p_4. (Only the transition labels are shown.)

If we assume that all transitions in Fig. 5 have a visible label and we have an event log L that is complete with respect to the "directly follows" relation (i.e., $x >_L y$ if and only if y can be directly followed by x in the model), then the α-algorithm is able to rediscover the original model. If $t7$ is invisible (not recorded in event log), then a more compact, but correct, model is derived by the α-algorithm. If $t2$ or $t11$ is invisible, the α-algorithm fails to discover a correct model, e.g., skipping activity b ($t2$) does not leave a trail in the event log and requires a more sophisticated discovery technique.

4.2 Region-Based Process Discovery

In the context of Petri nets, researchers have been looking at the so-called *synthesis problem*, i.e., constructing a system model its desired behavior. State-based regions can be used to construct a Petri net from a transition system [36,48]. Language-based regions can be used to construct a Petri net from a prefix-closed language. Synthesis approaches using language-based regions can be applied directly to event logs. To apply state-based regions, one first needs to create a transition system as shown in [19]. Here, we restrict ourselves to an informal introduction to language-based regions.

Suppose, we have an event log $L \in \mathcal{B}(A^*)$. For this log one could construct a system net SN without any places and just transitions being continuously enabled. Given a set of transitions with labels A this system net is able to reproduce any event log $L \in \mathcal{B}(A^*)$. Such a Petri net is called the "flower model"

and adding places to this model can only limit the behavior. Language-based regions aim at finding places such that behavior is restricted properly, i.e., allow for the observed and likely behavior [27,28,32,93].

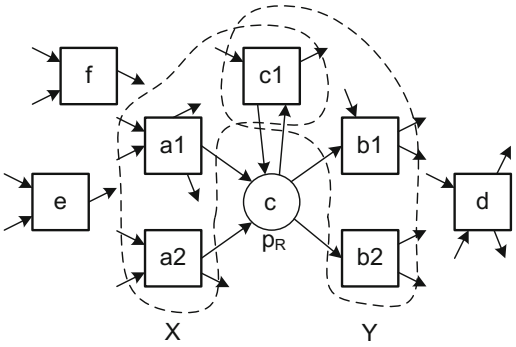


Fig. 10. Region $R = (X, Y, c)$ corresponding to place p_R: $X = \{a1, a2, c1\} = {\bullet}p_R$, $Y = \{b1, b2, c1\} = p_R{\bullet}$, and c is the initial marking of p_R

Consider for example place p_R in Fig. 10. Removing place p_R will not remove any behavior. However, adding p_R may remove behavior possible in the Petri net without this place. The behavior gets restricted when a place is empty while one of its output transitions wants to consume a token from it. For example, $b1$ is blocked if p_R is unmarked while all other input places of $b1$ are marked. Suppose now that we have a multiset of traces L. If these traces are possible in the net with place p_R, then they are also possible in the net without p_R. The reverse does not always hold. This triggers the question whether p_R can be added without disabling any of the traces in L. This is what regions are all about.

Definition 13 (Language-Based Region). *Let $L \in \mathcal{B}(A^*)$ be an event log. $R = (X, Y, c)$ is a* region *of L if and only if:*

- *$X \subseteq A$ is the set of input transitions of R;*
- *$Y \subseteq A$ is the set of output transitions of R;*
- *$c \in \{0, 1\}$ is the initial marking of R; and*
- *for any $\sigma \in L$, $k \in \{1, \ldots, |\sigma|\}$:*

$$c + \sum_{t \in X} mults^{k-1}(\sigma)(t) - \sum_{t \in Y} mults^{k}(\sigma)(t) \geq 0.$$

$R = (X, Y, c)$ is a region of L if and only if inserting a place p_R with ${\bullet}p_R = X$, $p_R{\bullet} = Y$, and initially c tokens does not disable the execution of any of the traces in L. To check this, Definition 13 inspects all events in the event log. Let $\sigma \in L$ be a trace in the log. $a = \sigma(k)$ is the k-th event in this trace. This event should

not be disabled by place p_R. Therefore, we calculate the number of tokens $M(p_R)$ that are in this place just before the occurrence of the k-th event.

$$M(p_R) = c + \sum_{t \in X} mults^{k-1}(\sigma)(t) - \sum_{t \in Y} mults^{k-1}(\sigma)(t)$$

$mults^{k-1}(\sigma)$ is the multiset of events that occurred before the occurrence of the k-th event. $\sum_{t \in X} mults^{k-1}(\sigma)(t)$ counts the number of tokens produced for place p_R, $\sum_{t \in Y} mults^{k-1}(\sigma)(t)$ counts the number of tokens consumed from this place, and c is the initial number of tokens in p_R. Therefore, $M(p_R)$ is indeed the number of tokens in p_R just before the occurrence of the k-th event. This number should be positive. In fact, there should be at least one token in p_R if $a \in Y$. In other words, $M(p_R)$ minus the number of tokens consumed from p_R by the k-th event should be non-negative. Hence:

$$M(p_R) - \sum_{t \in Y} [a](t) = c + \sum_{t \in X} mults^{k-1}(\sigma)(t) - \sum_{t \in Y} mults^{k}(\sigma)(t) \geq 0.$$

This shows that a region R, according to Definition 13, indeed corresponds to a so-called *feasible place* p_R, i.e., a place that can be added without disabling any of the traces in the event log.

The requirement stated in Definition 13 can also be formulated in terms of an inequation system. To illustrate this we use the example log $L_{3b} = [\langle a, c, d \rangle^{88}, \langle b, c, e \rangle^{87}]$ for which the α-algorithm was unable to find a suitable model. There are five activities. For each activity t we introduce two variables: x_t and y_t. $x_t = 1$ if transition t produces a token for p_R and $x_t = 0$ if not. $y_t = 1$ if transition t consumes a token from p_R and $y_t = 0$ if not. A potential region $R = (X, Y, c)$ corresponds to an assignment for all of these variables: $x_t = 1$ if $t \in X$, $x_t = 0$ if $t \notin X$, $y_t = 1$ if $t \in Y$, $y_t = 0$ if $t \notin Y$. The requirement stated in Definition 13 can now be reformulated in terms of the variables x_a, x_b, x_c, x_d, x_e, y_a, y_b, y_c, y_d, y_e, and c for event log L_{3b}:

$$\begin{aligned}
c - y_a &\geq 0 \\
c + x_a - (y_a + y_c) &\geq 0 \\
c + x_a + x_c - (y_a + y_c + y_d) &\geq 0 \\
c - y_b &\geq 0 \\
c + x_b - (y_b + y_c) &\geq 0 \\
c + x_b + x_c - (y_b + y_c + y_e) &\geq 0 \\
c, x_a, \ldots, x_e, y_a, \ldots, y_e &\in \{0, 1\}
\end{aligned}$$

Note that these inequations are based on all non-empty prefixes of $\langle a, c, d \rangle$ and $\langle b, c, e \rangle$. Any solution of this linear inequation system corresponds to a region. Some example solutions are:

$$\begin{aligned}
R_1 &= (\emptyset, \{a, b\}, 1) \\
&\quad c = y_a = y_b = 1, \quad x_a = x_b = x_c = x_d = x_e = y_c = y_d = y_e = 0
\end{aligned}$$

$$R_2 = (\{a,b\},\{c\},0)$$
$$x_a = x_b = y_c = 1, \quad c = x_c = x_d = x_e = y_a = y_b = y_d = y_e = 0$$
$$R_3 = (\{c\},\{d,e\},0)$$
$$x_c = y_d = y_e = 1, \quad c = x_a = x_b = x_d = x_e = y_a = y_b = y_c = 0$$
$$R_4 = (\{d,e\},\emptyset,0)$$
$$x_d = x_e = 1, \quad c = x_a = x_b = x_c = y_a = y_b = y_c = y_d = y_e = 0$$
$$R_5 = (\{a\},\{d\},0)$$
$$x_a = y_d = 1, \quad c = x_b = x_c = x_d = x_e = y_a = y_b = y_c = y_e = 0$$
$$R_6 = (\{b\},\{e\},0)$$
$$x_b = y_e = 1, \quad c = x_a = x_c = x_d = x_e = y_a = y_b = y_c = y_d = 0$$

Consider for example $R_6 = (\{b\},\{e\},0)$. This corresponds to the solution $x_b = y_e = 1$ and $c = x_a = x_c = x_d = x_e = y_a = y_b = y_c = y_d = 0$. If we fill out the values in the inequation system, we can see that this is indeed a solution. If we construct a Petri net based on these six regions, we obtain SN_{3b}, i.e., the system net depicted in Fig. 9 including places p_3 and p_4 (modulo renaming of places).

Suppose that the trace $\langle a,c,e\rangle$ is added to event log L_{3b}. This results in three additional inequations:

$$c - y_a \geq 0$$
$$c + x_a - (y_a + y_c) \geq 0$$
$$c + x_a + x_c - (y_a + y_c + y_e) \geq 0$$

Only the last inequation is new. Because of this inequation, $x_b = y_e = 1$ and $c = x_a = x_c = x_d = x_e = y_a = y_b = y_c = y_d = 0$ is no longer a solution. Hence, $R_6 = (\{b\},\{e\},0)$ is not a region anymore and place $p4$ needs to be removed from the system net shown in Fig. 9. After removing this place, the resulting system net indeed allows for $\langle a,c,e\rangle$.

One of the problems of directly applying language-based regions is that the linear inequation system has many solutions. Few of these solutions correspond to sensible places. For example, $x_a = x_b = y_d = y_e = 1$ and $c = x_c = x_d = x_e = y_a = y_b = y_c = 0$ also defines a region: $R_7 = (\{a,b\},\{d,e\},0)$. However, adding this place to Fig. 9 would only clutter the diagram. Another example is $c = x_a = x_b = y_c = 1$ and $x_c = x_d = x_e = y_a = y_b = y_d = y_e = 0$, i.e., region $R_8 = (\{a,b\},\{c\},1)$. This region is a weaker variant of R_2 as the place is initially marked.

Another problem is that classical techniques for language-based regions aim at a Petri net that does not allow for any behavior not seen in the log [28]. This means that the log is considered to be complete. This is very unrealistic and results in models that are complex and overfitting. To address these problems dedicated techniques have been proposed. For instance, in [93] it is shown how to avoid overfitting and how to ensure that the resulting model has

desirable properties (WF-net, free-choice, etc.). Nevertheless, pure region-based techniques tend to have problems handling noise and incompleteness.

4.3 Other Process Discovery Approaches

The α-algorithm and the region-based approach just presented have many limitations. However, there are dozens of more advanced process discovery approaches. For example, consider *genetic process mining* techniques [30,65]. The idea of genetic process mining is to use evolution ("survival of the fittest") when searching for a process model. Like in any genetic algorithm there are four main steps: (a) initialization, (b) selection, (c) reproduction, and (d) termination. In the *initialization step* the initial population is created. This is the first generation of individuals to be used. Here an individual is a process model (e.g., a Petri net, transition system, Markov chain or process tree). Using the activity names appearing in the log, process models are created randomly. In a generation there may be hundreds or thousands of individuals (e.g., candidate Petri nets). In the *selection step*, the fitness of each individual is computed. A fitness function determines the quality of the individual in relation to the log.[2] Tournaments among individuals and elitism are used to ensure that genetic material of the best process models has the highest probability of being used for the next generation: *survival of the fittest*. In the *reproduction phase* the selected parent individuals are used to create new offspring. Here two genetic operators are used: *crossover* (creating child models that share parts of the genetic material of their parents) and *mutation* (e.g., randomly adding or deleting causal dependencies). Through reproduction and elitism a new generation is created. For the models in the new generation fitness is computed. Again the best individuals move on to the next round (elitism) or are used to produce new offspring. This is repeated and the expectation is that the "quality" of each generation gets better and better. The evolution process terminates when a satisfactory solution is found, i.e., a model having at least the desired fitness.

Next to genetic process mining techniques [30,65] there are many other discovery techniques. For example, *heuristic* [92] and *fuzzy* [53] mining techniques are particularly suitable for practical applications, but are outside the scope of this tutorial paper (see [2] for a more comprehensive overview).

5 Conformance Checking

Conformance checking techniques investigate how well an event log $L \in \mathcal{B}(A^*)$ and a system net $SN = (N, M_{init}, M_{final})$ fit together. Note that SN may have been discovered through process mining or may have been made by hand. In any case, it is interesting to compare the observed example behavior in L with the potential behavior of SN.

[2] Note that "fitness" in genetic mining has a different meaning than the (replay) fitness at other places in this paper. Genetic fitness corresponds to the more general notion of conformance including replay fitness, simplicity, precision, and generalization.

5.1 Quality Dimensions

Conformance checking can be done for various reasons. First of all, it may be used to audit processes to see whether reality conforms to some normative or descriptive model [14,41]. Deviations may point to:

- *fraud* (deliberate non-conforming behavior),
- *inefficiencies* (carelessness or sloppiness causing unnecessary delays or costs),
- *exceptions* (selected cases are handled in an ad-hoc manner because of special circumstances not covered by the model),
- *poorly designed procedures* (to get the work done people need to deviate from the model continuously), or
- *outdated procedures* (the process description does not match reality anymore because the process evolved over time).

Second, conformance checking can be used to evaluate the performance of a process discovery technique. In fact, genetic process mining algorithms use conformance checking to select the candidate models used to create the next generation of models [30,65].

There are four quality dimensions for comparing model and log: (1) *replay fitness*, (2) *simplicity*, (3) *precision*, and (4) *generalization* [2]. A model with good *replay fitness* allows for most of the behavior seen in the event log. A model has a perfect fitness if all traces in the log can be replayed by the model from beginning to end. If there are two models explaining the behavior seen in the log, we generally prefer the *simplest* model. This principle is known as Occam's Razor. Fitness and simplicity alone are not sufficient to judge the quality of a discovered process model. For example, it is very easy to construct an extremely simple Petri net ("flower model") that is able to replay all traces in an event log (but also any other event log referring to the same set of activities). Similarly, it is undesirable to have a model that only allows for the exact behavior seen in the event log. Remember that the log contains only example behavior and that many traces that are possible may not have been seen yet. A model is *precise* if it does not allow for "too much" behavior. Clearly, the "flower model" lacks precision. A model that is not precise is "underfitting". Underfitting is the problem that the model over-generalizes the example behavior in the log (i.e., the model allows for behaviors very different from what was seen in the log). At the same time, the model should generalize and not restrict behavior to just the examples seen in the log. A model that does not *generalize* is "overfitting" [8,9]. Overfitting is the problem that a very specific model is generated whereas it is obvious that the log only holds example behavior (i.e., the model explains the particular sample log, but there is a high probability that the model is unable to explain the next batch of cases). Process discovery techniques typically have problems finding the appropriate balance between precision and generalization because the event log only contains "positive examples", i.e., the event log does not indicate what could *not* happen.

In the remainder, we will focus on fitness. However, replay fitness is the starting point to the other quality dimensions [8,9,30].

5.2 Token-Based Replay

A simple fitness metric is the fraction of perfectly fitting traces. For example, the system net shown in Fig. 8 has a fitness of 0.8 for event log $L_4 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^3, \langle a,e,d\rangle^2, \langle a,d\rangle, \langle a,e,e,d\rangle]$ because 8 of the 10 traces fit perfectly. Such a naïve fitness metric is less suitable for more realistic processes because it cannot distinguish between "almost fitting" traces and traces that are completely unrelated to the model. Therefore, we also need a more refined fitness notion defined *at the level of events* rather than full traces. Rather than aborting the replay of a trace once we encounter a problem we can also continue replaying the trace on the model and record all situations where a transition is forced to fire without being enabled, i.e., we count all missing tokens. Moreover, we record the tokens that remain at the end.

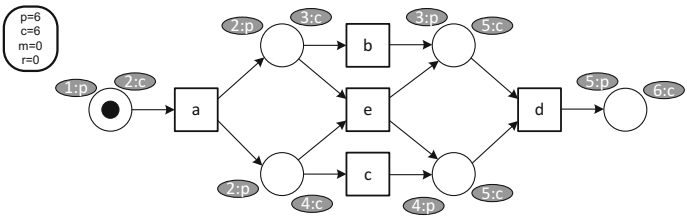


Fig. 11. Replaying trace $\sigma_1 = \langle a,b,c,d\rangle$ on the system net shown in Fig. 8: $\mathit{fitness}(\sigma_1) = \frac{1}{2}(1-\frac{0}{6}) + \frac{1}{2}(1-\frac{0}{6}) = 1$. (Place and transition identifiers are not shown, only the transition labels are depicted.)

To explain the idea, we first replay $\sigma_1 = \langle a,b,c,d\rangle$ on the system net shown in Fig. 8. We use four counters: p (produced tokens), c (consumed tokens), m (missing tokens), and r (remaining tokens). Initially, $p = c = 0$ and all places are empty. Then the environment produces a token to create the initial marking. Therefore, the p counter is incremented: $p = 1$ (Step 1 in Fig. 11). Now we need to fire transition a first. This is possible. Since a consumes one token and produces two tokens, the c counter is incremented by 1 and the p counter is incremented by 2 (Step 2 in Fig. 11). Therefore, $p = 3$ and $c = 1$ after firing transition a. Then we replay the second event (b). Firing transition b results in $p = 4$ and $c = 2$ (Step 3 in Fig. 11). After replaying the third event (i.e. c) $p = 5$ and $c = 3$. They we replay d. Since d consumes two tokens and produces one, the result is $p = 6$ and $c = 5$ (Step 5 in Fig. 11). At the end, the environment consumes a token from the sink place (Step 6 in Fig. 11). Hence the final result is $p = c = 6$ and $m = r = 0$. Clearly, there are no problems when replaying the σ_1, i.e., there are no missing or remaining tokens ($m = r = 0$).

The fitness of trace σ is defined as follows:

$$\mathit{fitness}(\sigma) = \frac{1}{2}\left(1-\frac{m}{c}\right) + \frac{1}{2}\left(1-\frac{r}{p}\right)$$

The first parts computes the fraction of missing tokens relative to the number of consumed tokens. $1-\frac{m}{c}=1$ if there are no missing tokens ($m=0$) and $1-\frac{m}{c}=0$ if all tokens to be consumed were missing ($m=c$). Similarly, $1-\frac{r}{p}=1$ if there are no remaining tokens and $1-\frac{r}{p}=0$ if none of the produced tokens was actually consumed. We use an equal penalty for missing and remaining tokens. By definition: $0 \leq \mathit{fitness}(\sigma) \leq 1$. In our example, $\mathit{fitness}(\sigma_1) = \frac{1}{2}(1-\frac{0}{6}) + \frac{1}{2}(1-\frac{0}{6}) = 1$ because there are no missing or remaining tokens.

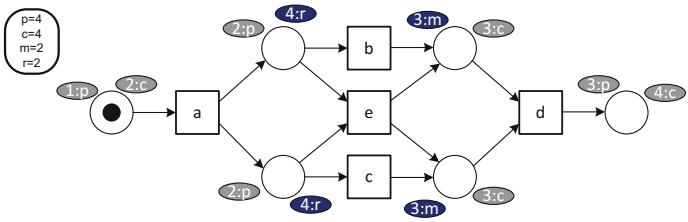


Fig. 12. Replaying trace $\sigma_2 = \langle a, d \rangle$ on the system net shown in Fig. 8: $\mathit{fitness}(\sigma_2) = \frac{1}{2}(1-\frac{2}{4}) + \frac{1}{2}(1-\frac{2}{4}) = 0.5$.

Let us now consider a trace that cannot be replayed properly. Figure 12 shows the process of replaying $\sigma_2 = \langle a, d \rangle$. Initially, $p = c = 0$ and all places are empty. Then the environment produces a token for the initial marking and the p counter is updated: $p = 1$. The first event (a) can be replayed (Step 2 in Fig. 12). After firing a, we have $p = 3$, $c = 1$, $m = 0$, and $r = 0$. Now we try to replay the second event. This is not possible, because transition d is not enabled. To fire d, we need to add a token to each of the input places of d and record the two missing tokens (Step 3 in Fig. 12) The m counter is incremented. The p and c counter are updated as usual. Therefore, after firing d, we have $p = 4$, $c = 3$, $m = 2$, and $r = 0$. At the end, the environment consumes a token from the sink place (Step 4 in Fig. 12). Moreover, we note the two remaining tokens on the output places of a. Hence the final result is $p = c = 4$ and $m = r = 2$. Figure 12 shows diagnostic information that helps to understand the nature of non-conformance. There was a situation in which d occurred but could not happen according to the model (m-tags) and there was a situation in which b and c or e were supposed to happen but did not occur according to the log (r-tags). Moreover, we can compute the fitness of trace σ_2 based on the values of p, c, m, and r: $\mathit{fitness}(\sigma_2) = \frac{1}{2}\left(1-\frac{2}{4}\right) + \frac{1}{2}\left(1-\frac{2}{4}\right) = 0.5$.

Figures 11 and 12 illustrate how to analyze the fitness of a single case. The same approach can be used to analyze the fitness of a log consisting of many cases. Simply take the sums of all produced, consumed, missing, and remaining tokens, and apply the same formula. Let p_σ denote the number of produced tokens when replaying σ on N. c_σ, m_σ, r_σ are defined in a similar fashion, e.g., m_σ is the number of missing tokens when replaying σ. Now we can define the

fitness of an event log L on a given system net:

$$\mathit{fitness}(L) = \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times m_\sigma}{\sum_{\sigma \in L} L(\sigma) \times c_\sigma}\right) + \frac{1}{2}\left(1 - \frac{\sum_{\sigma \in L} L(\sigma) \times r_\sigma}{\sum_{\sigma \in L} L(\sigma) \times p_\sigma}\right)$$

By replaying the entire event log, we can now compute the fitness of event log $L_4 = [\langle a, b, c, d\rangle^3, \langle a, c, b, d\rangle^3, \langle a, e, d\rangle^2, \langle a, d\rangle, \langle a, e, e, d\rangle]$ for the system net shown in Fig. 8. The total number of produced tokens is $p = 3{\cdot}6{+}3{\cdot}6{+}2{\cdot}6{+}1{\cdot}4{+}1{\cdot}8 = 60$. There are also $c = 60$ consumed tokens. The number of missing tokens is $m = 3 \cdot 0 + 3 \cdot 0 + 2 \cdot 0 + 1 \cdot 2 + 1 \cdot 2 = 4$. There are also $r = 4$ remaining tokens. Hence, $\mathit{fitness}(L_4) = \frac{1}{2}\left(1 - \frac{4}{60}\right) + \frac{1}{2}\left(1 - \frac{4}{60}\right) = 0.933$.

Typically, the event-based fitness is higher than the naïve case-based fitness. This is also the case here. The system net in Fig. 8 can only replay 80 % of the cases from start to end. However, about 93 % of the individual events can be replayed. For more information on token-based replay we refer to [2,79].

An event log can be split into two sublogs: *one event log containing only fitting cases and one event log containing only non-fitting cases*. Each of the event logs can be used for further analysis. For example, one could construct a process model for the event log containing only deviating cases. Also other data and process mining techniques can be used, e.g., one can use classification techniques to further investigate non-conformance.

5.3 Aligning Observed and Modeled Behavior

There are various ways to quantify fitness [2,9,22,52,65,68,69,79]. The simple procedure of counting missing, remaining, produced, and consumed tokens has several limitations. For example, in case of multiple transitions with the same label or transitions that are invisible, there are all kinds of complications. Which path to take if multiple transitions with the same label are enabled? Moreover, in case of poor fitness the Petri net is flooded with tokens thus resulting in optimistic estimates (many transitions are enabled). The notion of *cost-based alignments* [9, 22] provides a more robust and flexible approach for conformance checking.

To measure fitness, we *align* traces in the event log to traces of the process model. Consider the following three alignments for the traces in $L_1 = [\langle a, c, d, f, g\rangle^{10}, \langle a, c, d, h\rangle^5, \langle a, b, c, d, e, c, d, g, f\rangle^5]$ and the system net in Fig. 5:

$$\gamma_1 = \begin{array}{|c|c|c|c|c|c|c|c|}\hline a & c & \gg & d & \gg & f & g & \gg \\ \hline a & c & \tau & d & \tau & f & g & \tau \\ t1 & t4 & t2 & t5 & t7 & t8 & t9 & t11 \\ \hline \end{array} \qquad \gamma_2 = \begin{array}{|c|c|c|c|c|}\hline a & c & \gg & d & h \\ \hline a & c & \tau & d & h \\ t1 & t4 & t2 & t5 & t10 \\ \hline \end{array}$$

$$\gamma_3 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline a & b & c & d & e & c & \gg & d & \gg & g & f & \gg \\ \hline a & b & c & d & e & c & \tau & d & \tau & g & f & \tau \\ t1 & t3 & t4 & t5 & t6 & t4 & t2 & t5 & t7 & t9 & t8 & t11 \\ \hline \end{array}$$

The top row of each alignment corresponds to "moves in the log" and the bottom two rows correspond to "moves in the model". Moves in the model are represented by the transition and its label. This is needed because there could be

multiple transitions with the same label. In alignment γ_1 the first column refers to a "move in both", i.e., both the event log and the process model make an a move. If a move in the model cannot be mimicked by a move in the log, then a "$\gg$" ("no move") appears in the top row. This situation is referred to as a "move in model". For example, in the third position of γ_1 the log cannot mimic the invisible transition $t2$. The τ above $t2$ indicates that $t2 \notin dom(l)$. In the remainder, we write $l(t) = \tau$ if $t \notin dom(l)$. Note that all "no moves" (i.e., the seven $\gg$ symbols) in $\gamma_1 - \gamma_3$ are "caused" by invisible transitions.

Let us now consider some example alignments for the deviating event log $L'_1 = [\langle a, c, d, f\rangle^{10}, \langle a, c, d, c, h\rangle^5, \langle a, b, d, e, c, d, g, f, h\rangle^5]$ and system net SN in Fig. 5:

$$\gamma_4 = \begin{array}{|c|c|c|c|c|c|c|c|}\hline a & c & \gg & d & \gg & f & \gg & \gg \\ \hline a & c & \tau & d & \tau & f & g & \tau \\ t1 & t4 & t2 & t5 & t7 & t8 & t9 & t11 \\ \hline \end{array} \qquad \gamma_5 = \begin{array}{|c|c|c|c|c|c|}\hline a & c & \gg & d & c & h \\ \hline a & c & \tau & d & \gg & h \\ t1 & t4 & t2 & t5 & & t10 \\ \hline \end{array}$$

$$\gamma_6 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline a & b & \gg & d & e & c & \gg & d & \gg & g & f & \gg & h \\ \hline a & b & c & d & e & c & \tau & d & \tau & g & f & \tau & \gg \\ t1 & t3 & t4 & t5 & t6 & t4 & t2 & t5 & t7 & t9 & t8 & t11 & \\ \hline \end{array}$$

Alignment γ_4 shows a "$\gg$" ("no move") in the top row that does not correspond to an invisible transition. The model makes a g move (occurrence of transition $t9$) that is not in the log. Alignment γ_6 has a similar move in the third position: the model makes a c move (occurrence of transition $t4$) that is not in the log. If a move in the log cannot be mimicked by a move in the model, then a "$\gg$" ("no move") appears in the bottom row. This situation is referred to as a "move in log". For example, in γ_5 the c move in the log is not mimicked by a move in the model and in γ_6 the h move in the log is not mimicked by a move in the model. Note that the "no moves" not corresponding to invisible transitions point to deviations between model and log.

A *move* is a pair $(x, (y, t))$ where the first element refers to the log and the second element refers to the model. For example, $(a, (a, t1))$ means that both log and model make an "a move" and the move in the model is caused by the occurrence of transition $t1$. $(\gg, (g, t9))$ means that the occurrence of transition $t9$ with label g is not mimicked by corresponding move of the log. $(c, \gg)$ means that the log makes an "c move" not followed by the model.

Definition 14 (Legal Moves). *Let $L \in \mathcal{B}(A^*)$ be an event log and let $SN = (N, M_{init}, M_{final}) \in \mathcal{U}_{SN}$ be a system net with $N = (P, T, F, l)$. $A_{LM} = \{(x, (x, t)) \mid x \in A \wedge t \in T \wedge l(t) = x\} \cup \{(\gg, (x, t)) \mid t \in T \wedge l(t) = x\} \cup \{(x, \gg) \mid x \in A\}$ is the set of* legal moves.

An alignment is a sequence of legal moves such that after removing all $\gg$ symbols, the top row corresponds to a trace in the log and the bottom row corresponds to a firing sequence starting in M_{init} and ending M_{final}. Hence, the middle row corresponds to a visible path when ignoring the τ steps.

Definition 15 (Alignment). *Let $\sigma_L \in L$ be a log trace and $\sigma_M \in \phi_f(SN)$ a complete firing sequence of system net SN. An* alignment *of σ_L and σ_M is a sequence $\gamma \in {A_{LM}}^*$ such that the projection on the first element (ignoring $\gg$) yields σ_L and the projection on the last element (ignoring $\gg$ and transition labels) yields σ_M.*

γ_1–γ_3 are examples of alignments for the traces in L_1 and their corresponding firing sequences in the system net of Fig. 5. γ_4–γ_6 are examples of alignments for the traces in L_1' and complete firing sequences of the same system net. The projection of γ_6 on the first element (ignoring $\gg$) yields $\sigma_L = \langle a, b, d, e, c, d, g, f, h\rangle$ which is indeed a trace in L_1'. The projection of γ_6 on the last element (ignoring $\gg$ and transition labels) yields $\sigma_M = \langle t1, t3, t4, t5, t6, t4, t2, t5, t7, t9, t8, t11\rangle$ which is indeed a complete firing sequence. The projection of γ_6 on the middle element (i.e., transition labels while ignoring $\gg$ and τ) yields $\langle a, b, c, d, e, c, d, g, f\rangle$ which is indeed a visible trace of the system net of Fig. 5.

Given a log trace and a process model there may be many (if not infinitely many) alignments. Consider the following two alignments for $\langle a, c, d, f\rangle \in L_1'$:

$$\gamma_4 = \begin{array}{|c|c|c|c|c|c|c|c|}\hline a & c & \gg & d & \gg & f & \gg & \gg \\ \hline a & c & \tau & d & \tau & f & g & \tau \\ t1 & t4 & t2 & t5 & t7 & t8 & t9 & t11 \\ \hline \end{array} \qquad \gamma_4' = \begin{array}{|c|c|c|c|c|c|c|}\hline a & c & \gg & d & \gg & f & \gg \\ \hline a & c & b & d & \tau & \gg & h \\ t1 & t4 & t3 & t5 & t7 & & t10 \\ \hline \end{array}$$

γ_4 seems to be better alignment than γ_4' because it has only one deviation (move in model only; $(\gg, (g, t9))$) whereas γ_4' has three deviations: $(\gg, (b, t3))$, $(f, \gg)$, and $(\gg, (h, t11))$. To select the most appropriate one we associate *costs* to undesirable moves and select an alignment with the lowest total costs. To quantify the costs of misalignments we introduce a cost function δ.

Definition 16 (Cost of Alignment). *Cost function $\delta \in A_{LM} \to \mathbb{N}$ assigns costs to legal moves. The* cost *of an alignment $\gamma \in {A_{LM}}^*$ is the sum of all costs: $\delta(\gamma) = \sum_{(x,y)\in\gamma} \delta(x, y)$.*

Moves where log and model agree have no costs, i.e., $\delta(x, (x, t)) = 0$ for all $x \in A$. Moves in model only have no costs if the transition is invisible, i.e., $\delta(\gg, (\tau, t)) = 0$ if $l(t) = \tau$. $\delta(\gg, (x, t)) > 0$ is the cost when the model makes an "x move" without a corresponding move of the log (assuming $l(t) = x \neq \tau$). $\delta(x, \gg) > 0$ is the cost for an "x move" in just the log. These costs may depend on the nature of the activity, e.g., skipping a payment may be more severe than sending too many letters. However, in this paper we often use a standard cost function δ_S that assigns unit costs: $\delta_S(x, (x, t)) = 0$, $\delta_S(\gg, (\tau, t)) = 0$, and $\delta_S(\gg, (x, t)) = \delta_S(x, \gg) = 1$ for all $x \in A$. For example, $\delta_S(\gamma_1) = \delta_S(\gamma_2) = \delta_S(\gamma_3) = 0$, $\delta_S(\gamma_4) = 1$, $\delta_S(\gamma_5) = 1$, and $\delta_S(\gamma_6) = 2$ (simply count the number of $\gg$ symbols not corresponding to invisible transitions). Now we can compare the two alignments for $\langle a, c, d, f\rangle \in L_1'$: $\delta_S(\gamma_4) = 1$ and $\delta_S(\gamma_4') = 3$. Hence, we conclude that γ_4 is "better" than γ_4'.

Definition 17 (Optimal Alignment). *Let $L \in \mathcal{B}(A^*)$ be an event log with $A \subseteq \mathcal{U}_A$ and let $SN \in \mathcal{U}_{SN}$ be a system net with $\phi(SN) \neq \emptyset$.*

- *For* $\sigma_L \in L$*, we define:* $\Gamma_{\sigma_L,SN} = \{\gamma \in {A_{LM}}^* \mid \exists_{\sigma_M \in \phi_f(SN)} \ \gamma$ *is an aligment of* σ_L *and* $\sigma_M\}$.
- *An alignment* $\gamma \in \Gamma_{\sigma_L,SN}$ *is* optimal *for trace* $\sigma_L \in L$ *and system net SN if for any* $\gamma' \in \Gamma_{\sigma_L,M}$*:* $\delta(\gamma') \geq \delta(\gamma)$.
- $\lambda_{SN} \in A^* \rightarrow {A_{LM}}^*$ *is a deterministic mapping that assigns any log trace* σ_L *to an optimal alignment, i.e.,* $\lambda_{SN}(\sigma_L) \in \Gamma_{\sigma_L,SN}$ *and* $\lambda_{SN}(\sigma_L)$ *is optimal.*
- $costs(L, SN, \delta) = \sum_{\sigma_L \in L} \delta(\lambda_{SN}(\sigma_L))$ *are the* misalignment costs *of the whole event log.*

$\gamma_1 - \gamma_6$ is are optimal alignments for the corresponding six possible traces in event logs L_1 and L_1' and the system net in Fig. 5. γ_4' is not an optimal alignment for $\langle a, c, d, f\rangle$. $costs(L_1, SN, \delta_S) = 10 \times \delta_S(\gamma_1) + 5 \times \delta_S(\gamma_2) + 5 \times \delta_S(\gamma_3) = 10 \times 0 + 5 \times 0 + 5 \times 0 = 0$. Hence, L_1 is perfectly fitting system net SN. $costs(L_1', SN, \delta_S) = 10 \times \delta_S(\gamma_4) + 5 \times \delta_S(\gamma_5) + 5 \times \delta_S(\gamma_6) = 10 \times 1 + 5 \times 1 + 5 \times 2 = 25$.

It is possible to convert misalignment costs into a fitness value between 0 (poor fitness, i.e., maximal costs) and 1 (perfect fitness, zero costs). We refer to [9,22] for details.

Only perfectly fitting traces have costs 0 (assuming $\phi(SN) \neq \emptyset$). Hence, Event log L is perfectly fitting system net SN if and only if $costs(L, SN, \delta) = 0$.

Once an optimal alignment has been established for every trace in the event log, these alignments can also be used as a basis to quantify other conformance notations such as precision and generalization [9]. For example, precision can be computed by counting "escaping edges" as shown in [68,69]. Recent results show that such computations should be based on alignments [24]. The same holds for generalization [9]. Therefore, we focus on alignments when decomposing conformance checking problems in Sect. 6.

5.4 Beyond Conformance Checking

The importance of alignments cannot be overstated. Alignments relate observed behavior with modeled behavior. This is not only important for conformance checking, but also for enriching and repairing models. For example, timestamps in the event log can be used to analyze bottlenecks in the process model. In fact, partial alignments can also be used to predict problems and to recommend appropriate actions. This is illustrated by Fig. 13. See [2,9] for concrete examples.

6 Decomposing Process Mining Problems

The torrents of event data available are an important enabler for process mining. However, the incredible growth of event data also provides computational challenges. For example, conformance checking can be time consuming as potentially many different traces need to be aligned with a model that may allow for an exponential (or even infinite) number of traces. Event logs may contain millions of events. Finding the best alignment may require solving many optimization problems [22] or repeated state-space explorations [79]. In worst case

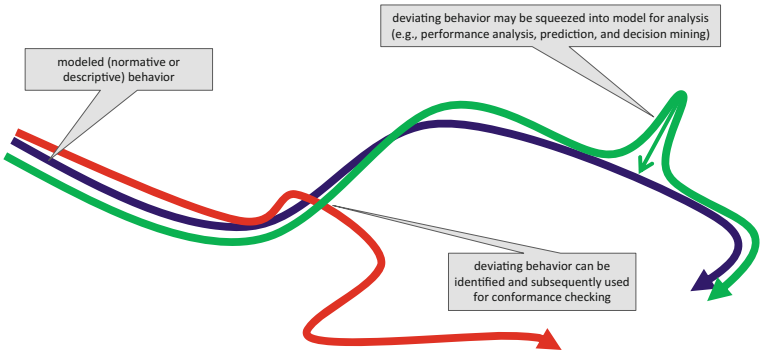


Fig. 13. The essence of process mining: relating modeled and observed behavior.

a state-space exploration of the model is needed per event. When using genetic process mining, one needs to check the fitness of every individual model in every generation [30,65]. As a result, thousands or even millions of conformance checks need to be done. For each conformance check, the whole event log needs to be traversed. Given these challenges, we are interested in reducing the time needed for conformance checking by decomposing the associated Petri net and event log. See [3,4,7] for an overview of various decomposition approaches. For example, in [4] we discuss the *vertical partitioning* and *horizontal partitioning* of event logs.

Event logs are composed of cases. There may be thousands or even millions of cases. In case of vertical partitioning these can be distributed over the nodes in the network, i.e., each case is assigned to one computing node. All nodes work on a subset of the whole log and in the end the results need to be merged.

Cases are composed of multiple events. We can also partition cases, i.e., part of a case is analyzed on one node whereas another part of the same case is analyzed on another node. This corresponds to a horizontal partitioning of the event log. In principle, each node needs to consider all cases. However, the attention of one computing node is limited to a particular subset of events per case.

Even when only one computing node is available, it may still be beneficial to decompose process mining problems. Due to the exponential nature of most conformance checking techniques, the time needed to solve "many smaller problems" is less than the time needed to solve "one big problem". In the remainder, we only consider the so-called horizontal partitioning of the event log.

6.1 Decomposing Conformance Checking

To decompose conformance checking problems we split a process model into model fragments. In terms of Petri nets: the overall system net SN is decomposed into a collection of subnets $\{SN^1, SN^2, \ldots, SN^n\}$ such that the union of these subnets yields the original system net. The union of two system nets is defined as follows.

Definition 18 (Union of Nets). *Let* $SN^1 = (N^1, M^1_{init}, M^1_{final}) \in \mathcal{U}_{SN}$ *with* $N^1 = (P^1, T^1, F^1, l^1)$ *and* $SN^2 = (N^2, M^2_{init}, M^2_{final}) \in \mathcal{U}_{SN}$ *with* $N^2 = (P^2, T^2, F^2, l^2)$ *be two system nets.*

- $l^3 \in (T^1 \cup T^2) \not\to \mathcal{U}_A$ *with* $dom(l^3) = dom(l^1) \cup dom(l^2)$, $l^3(t) = l^1(t)$ *if* $t \in dom(l^1)$, *and* $l^3(t) = l^2(t)$ *if* $t \in dom(l^2) \setminus dom(l^1)$ *is the union of* l_1 *and* l_2,
- $N^1 \cup N^2 = (P^1 \cup P^2, T^1 \cup T^2, F^1 \cup F^2, l^3)$ *is the union of* N^1 *and* N^2, *and*
- $SN^1 \cup SN^2 = (N^1 \cup N^2, M^1_{init} \uplus M^2_{init}, M^1_{final} \uplus M^2_{final})$ *is the union of system nets* SN^1 *and* SN^2.

Using Definition 18, we can check whether the union of a collection of subnets $\{SN^1, SN^2, \ldots, SN^n\}$ indeed corresponds to the overall system net SN. It suffices to check whether $SN = \bigcup_{1 \leq i \leq n} SN^i = SN^1 \cup SN^2 \cup \ldots \cup SN^n$. A decomposition $\{SN^1, SN^2, \ldots, SN^n\}$ is *valid* if the subnets "agree" on the original labeling function (i.e., the same transition always has the same label), each place resides in just one subnet, and also each invisible transition resides in just one subnet. Moreover, if there are multiple transitions with the same label, they should reside in the same subnet. Only unique visible transitions (i.e., $T^u_v(SN)$, cf. Definition 8) can be shared among different subnets.

Definition 19 (Valid Decomposition). *Let* $SN \in \mathcal{U}_{SN}$ *be a system net with labeling function* l. $D = \{SN^1, SN^2, \ldots, SN^n\} \subseteq \mathcal{U}_{SN}$ *is a valid decomposition if and only if*

- $SN^i = (N^i, M^i_{init}, M^i_{final})$ *is a system net with* $N^i = (P^i, T^i, F^i, l^i)$ *for all* $1 \leq i \leq n$,
- $l^i = l \restriction_{T^i}$ *for all* $1 \leq i \leq n$,
- $P^i \cap P^j = \emptyset$ *for* $1 \leq i < j \leq n$,
- $T^i \cap T^j \subseteq T^u_v(SN)$ *for* $1 \leq i < j \leq n$, *and*
- $SN = \bigcup_{1 \leq i \leq n} SN^i$.

$\mathcal{D}(SN)$ *is the set of all valid decompositions of* SN.

Every system net has a trivial decomposition consisting of only one subnet, i.e., $\{SN\} \in \mathcal{D}(SN)$. However, we are often interested in a *maximal decomposition* where the individual subnets are as small as possible. Figure 14 shows the maximal decomposition for the system net shown in Fig. 5.

In [7] it is shown that a unique maximal valid decomposition always exists. Moreover, it is possible to decompose nets based on the notion of *passages* [3] or using Single-Entry Single-Exit (SESE) components [70]. In the remainder, we assume a valid decomposition without making any further assumptions.

Next, we show that conformance checking can be done by locally inspecting the subnets using correspondingly projected event logs. To illustrate this, consider the following alignment for trace $\langle a, b, c, d, e, c, d, g, f \rangle$ and the system net in Fig. 5:

$$\gamma_3 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \hline a & b & c & d & e & c & \gg & d & \gg & g & f & \gg \\ \hline a & b & c & d & e & c & \tau & d & \tau & g & f & \tau \\ t1 & t3 & t4 & t5 & t6 & t4 & t2 & t5 & t7 & t9 & t8 & t11 \\ \hline \end{array}$$

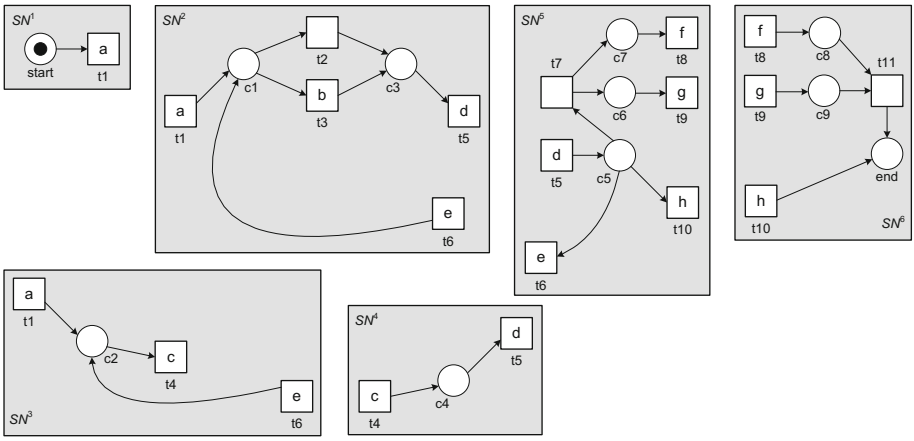


Fig. 14. Maximal decomposition of the system net shown in Fig. 5 with $M_{init} = [start]$ and $M_{final} = [end]$. The initial and final markings are as follows: $M^1_{init} = [start]$ and $M^i_{init} = [\,]$ for $2 \leq i \leq 6$, $M^i_{final} = [\,]$ for $1 \leq i \leq 5$, and $M^6_{final} = [end]$.

For convenience, the moves have been numbered. Now consider the following six alignments:

$$\gamma_3^1 = \begin{array}{|c|}\hline 1 \\ \hline a \\ \hline a \\ \hline t1 \\ \hline \end{array} \quad \gamma_3^2 = \begin{array}{|c|c|c|c|c|c|}\hline 1 & 2 & 4 & 5 & 7 & 8 \\ \hline a & b & d & e & \gg & d \\ \hline a & b & d & e & \tau & d \\ \hline t1 & t3 & t5 & t6 & t2 & t5 \\ \hline \end{array} \quad \gamma_3^3 = \begin{array}{|c|c|c|c|}\hline 1 & 3 & 5 & 6 \\ \hline a & c & e & c \\ \hline a & c & e & c \\ \hline t1 & t4 & t6 & t4 \\ \hline \end{array}$$

$$\gamma_3^4 = \begin{array}{|c|c|c|c|}\hline 3 & 4 & 6 & 8 \\ \hline c & d & c & d \\ \hline c & d & c & d \\ \hline t4 & t5 & t4 & t5 \\ \hline \end{array} \quad \gamma_3^5 = \begin{array}{|c|c|c|c|c|c|}\hline 4 & 5 & 8 & 9 & 10 & 11 \\ \hline d & e & d & \gg & g & f \\ \hline d & e & d & \tau & g & f \\ \hline t5 & t6 & t5 & t7 & t9 & t8 \\ \hline \end{array} \quad \gamma_3^6 = \begin{array}{|c|c|c|}\hline 10 & 11 & 12 \\ \hline g & f & \gg \\ \hline g & f & \tau \\ \hline t9 & t8 & t11 \\ \hline \end{array}$$

Each alignment corresponds to one of the six subnets $SN^1, SN^2, \ldots SN^6$ in Fig. 14. The numbers are used to relate the different alignments. For example γ_3^6 is an alignment for trace $\langle a, b, c, d, e, c, d, g, f \rangle$ and subnets SN^6 in Fig. 14. As the numbers 10, 11 and 12 indicate, γ_3^6 corresponds to the last three moves of γ_3.

To create sublogs for the different model fragments, we use the projection function introduced in Sect. 3. Consider for example the overall log $L_1 = [\langle a, c, d, f, g \rangle^{10}, \langle a, c, d, h \rangle^5, \langle a, b, c, d, e, c, d, g, f \rangle^5]$. $L_1^1 = L_1 \restriction_{\{a\}} = [\langle a \rangle^{20}]$, $L_1^2 = L_1 \restriction_{\{a,b,d,e\}} = [\langle a, d \rangle^{15}, \langle a, b, d, e, d \rangle^5]$, $L_1^3 = L_1 \restriction_{\{a,c,e\}} = [\langle a, c \rangle^{15}, \langle a, c, e, c \rangle^5]$, etc. are the sublogs corresponding to the subnets in Fig. 14.

The following theorem shows that any trace that fits the overall process model can be decomposed into smaller traces that fit the individual model fragments. Moreover, if the smaller traces fit the individual model fragments, then they can be composed into an overall trace that fits into the overall process model. This result is the basis for decomposing a wide range of process mining problems.

Theorem 1 (Conformance Checking Can be Decomposed). *Let $L \in \mathcal{B}(A^*)$ be an event log with $A \subseteq \mathcal{U}_A$ and let $SN \in \mathcal{U}_{SN}$ be a system net. For any valid decomposition $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$: L is perfectly fitting system net SN if and only if for all $1 \leq i \leq n$: $L \upharpoonright_{A_v(SN^i)}$ is perfectly fitting SN^i.*

Proof. See [7]. □

Theorem 1 shows that any trace in the log fits the overall model if and only if it fits each of the subnets.

Let us now consider trace $\langle a, b, d, e, c, d, g, f, h\rangle$ which is not perfectly fitting the system net in Fig. 5. An optimal alignment is:

$$\gamma_6 = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1&2&3&4&5&6&7&8&9&10&11&12&13\\\hline a&b&\gg&d&e&c&\gg&d&\gg&g&f&\gg&h\\\hline a&b&c&d&e&c&\tau&d&\tau&g&f&\tau&\gg\\\hline t1&t3&t4&t5&t6&t4&t2&t5&t7&t9&t8&t11&\\\hline\end{array}$$

The alignment shows the two problems: the model needs to execute c whereas this event is not in the event log (position 3) and the event log contains g, f, and h whereas the model needs to choose between either g and f or h (position 13). The cost of this optimal alignment is 2. Optimal alignment γ_6 for the overall model can be decomposed into alignments $\gamma_6^1 - \gamma_6^6$ for the six subnets:

$$\gamma_6^1 = \begin{array}{|c|}\hline 1\\\hline a\\\hline a\\\hline t1\\\hline\end{array} \quad \gamma_6^2 = \begin{array}{|c|c|c|c|c|c|}\hline 1&2&4&5&7&8\\\hline a&b&d&e&\gg&d\\\hline a&b&d&e&\tau&d\\\hline t1&t3&t5&t6&t2&t5\\\hline\end{array} \quad \gamma_6^3 = \begin{array}{|c|c|c|c|}\hline 1&3&5&6\\\hline a&\gg&e&c\\\hline a&c&e&c\\\hline t1&t4&t6&t4\\\hline\end{array}$$

$$\gamma_6^4 = \begin{array}{|c|c|c|c|}\hline 3&4&6&8\\\hline \gg&d&c&d\\\hline c&d&c&d\\\hline t4&t5&t4&t5\\\hline\end{array} \quad \gamma_6^5 = \begin{array}{|c|c|c|c|c|c|c|}\hline 4&5&8&9&10&11&13\\\hline d&e&d&\gg&g&f&h\\\hline d&e&d&\tau&g&f&\gg\\\hline t5&t6&t5&t7&t9&t8&\\\hline\end{array} \quad \gamma_6^6 = \begin{array}{|c|c|c|c|}\hline 10&11&12&13\\\hline g&f&\gg&h\\\hline g&f&\tau&\gg\\\hline t9&t8&t11&\\\hline\end{array}$$

Alignments γ_6^1 and γ_6^2 have costs 0. Alignments γ_6^3 and γ_6^4 have costs 1 (move in model involving c). Alignments γ_6^5 and γ_6^6 have costs 1 (move in log involving h). If we would add up all costs, we would get costs 4 whereas the costs of optimal alignment γ_6 is 2. However, we would like to compute an upper bound for the degree of fitness in a distributed manner. Therefore, we introduce an adapted cost function δ_Q.

Definition 20 (Adapted Cost Function). *Let $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$ be a valid decomposition of some system net SN and $\delta \in A_{LM} \rightarrow \mathbb{N}$ a cost function (cf. Definition 16). $c_Q(a, (a, t)) = c_Q(\gg, (a, t)) = c_Q(a, \gg) = |\{1 \leq i \leq n \mid a \in A_i\}|$ counts the number of subnets having a as an observable activity. The adapted cost function δ_Q is defined as follows: $\delta_Q(x, y) = \frac{\delta(x,y)}{c_Q(x,y)}$ for $(x, y) \in A_{LM}$ and $c_Q(x, y) \neq 0$.*

An observable activity may appear in multiple subnets. Therefore, we divide its costs by the number of subnets in which it appears: $\delta_Q(x,y) = \frac{\delta(x,y)}{c_Q(x,y)}$. This way we avoid counting misalignments of the same activity multiple times. For our example, $c_Q(\gg,(c,t4)) = |\{3,4\}| = 2$ and $c_Q(h,\gg) = |\{5,6\}| = 2$. Assuming the standard cost function δ_S this implies $\delta_Q(\gg,(c,t4)) = \frac{1}{2}$ and $\delta_Q(h,\gg) = \frac{1}{2}$. Hence the aggregated costs of $\gamma_6^1 - \gamma_6^6$ are 2, i.e., identical to the costs of the overall optimal alignment.

Theorem 2 (Lower Bound for Misalignment Costs). *Let $L \in \mathcal{B}(A^*)$ be an event log with $A \subseteq \mathcal{U}_A$, $SN \in \mathcal{U}_{SN}$ be a system net, and δ a cost function. For any valid decomposition $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$:*

$$costs(L, SN, \delta) \geq \sum_{1 \leq i \leq n} costs(L \restriction_{A_v(SN^i)}, SN^i, \delta_Q)$$

Proof. See [7]. □

The sum of the costs associated to all selected optimal local alignments (using δ_Q) can never exceed the cost of an optimal overall alignment using δ. Hence, it can be used as an optimistic estimate, i.e., computing an upper bound for the overall fitness and a lower bound for the overall costs. More important, the fitness values of the different subnets provide valuable local diagnostics. *The subnets with the highest costs are the most problematic parts of the model. The alignments for these "problem spots" help to understand the main problems without having to look at very long overall alignments.*

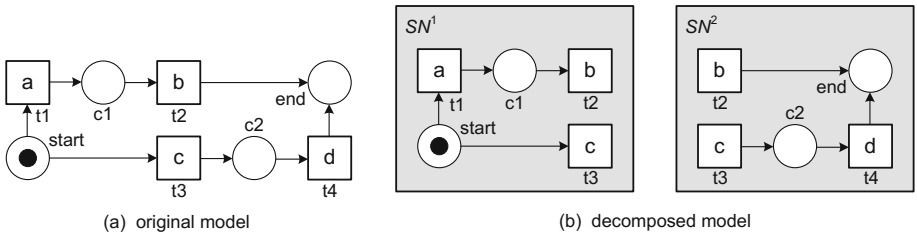


Fig. 15. Example showing that the total misalignment costs may be higher than the costs associated to the two optimal local alignments.

Theorem 2 only provides a lower bound for the misalignment costs. The total misalignment costs may be higher than the costs associated to the optimal local alignments. To understand this, consider the following optimal alignments for trace $\langle a,b,c,d\rangle$ and the (decomposed) process model shown in Fig. 15:

$$\gamma = \begin{array}{|c|c|c|c|}\hline a & b & c & d \\ \hline a & b & \gg & \gg \\ t1 & t2 & & \\ \hline \end{array} \qquad \gamma' = \begin{array}{|c|c|c|c|}\hline a & b & c & d \\ \hline \gg & \gg & c & d \\ & & t3 & t4 \\ \hline \end{array} \qquad \gamma_1 = \begin{array}{|c|c|c|}\hline a & b & c \\ \hline a & b & \gg \\ t1 & t2 & \\ \hline \end{array} \qquad \gamma_2 = \begin{array}{|c|c|c|}\hline b & c & d \\ \hline \gg & c & d \\ & t3 & t4 \\ \hline \end{array}$$

There are two optimal alignments (γ and γ') for $\langle a, b, c, d\rangle$ and the overall system net shown in Fig. 15(a). Both optimal alignments (γ and γ') have two moves in log only (i.e., these events cannot be mimicked by the model). Hence, $\delta_S(\gamma) = \delta_S(\gamma') = 2$. Now consider the decomposition shown in Fig. 15(b). The cost of the optimal alignment γ_1 for subnet SN^1 is $\delta_Q(\gamma_1) = 0 + 0 + \delta_Q(c, \gg) = \frac{\delta_S(c,\gg)}{c_Q(c,\gg)} = \frac{1}{2} = 0.5$. The cost of the optimal alignment γ_2 for subnet SN^2 is $\delta_Q(\gamma_2) = \delta_Q(b, \gg) + 0 + 0 = \frac{\delta_S(b,\gg)}{c_Q(b,\gg)} = \frac{1}{2} = 0.5$. Since $\delta_Q(\gamma_1) + \delta_Q(\gamma_2) = 1$ and $\delta_S(\gamma) = 2$, we can observe the misalignment costs for γ are indeed higher than the costs associated to the two optimal local alignments (γ_1 and γ_2). This is caused by the fact that the two optimal local alignments don't agree on the moves with respect to activities b and c. γ_1 suggests a move in both for activity b and a move in model for c. γ_2 makes a different choice and suggests a move in both for activity c and a move in model for d. Therefore, γ_1 and γ_2 cannot be stitched back into an overall alignment with costs 1. Practical experiences show that the difference between $costs(L, SN, \delta)$ and $sum_{1\leq i\leq n} costs(L \restriction_{A_v(SN^i)}, SN^i, \delta_Q)$ increases when the fragments are getting smaller. Hence, there is tradeoff between the accuracy of the lower bound and the degree of decomposition. If the subnets are chosen large enough, accuracy tends to be quite good.

Theorem 2 uses a rather sophisticated definition of fitness. We can also simply count the *fraction of fitting traces*. In this case the problem can be decomposed easily using the notion of valid decomposition.

Corollary 1 (Fraction of Perfectly Fitting Traces). *Let $L \in \mathcal{B}(A^*)$ be an event log with $A \subseteq \mathcal{U}_A$ and let $SN \in \mathcal{U}_{SN}$ be a system net. For any valid decomposition $D = \{SN^1, SN^2, \ldots, SN^n\} \in \mathcal{D}(SN)$:*

$$\frac{|[\sigma \in L \mid \sigma \in \phi(SN)]|}{|L|} = \frac{|[\sigma \in L \mid \forall_{1\leq i\leq n}\ \sigma \restriction_{A_v(SN^i)} \in \phi(SN^i)]|}{|L|}$$

The corollary follows directly from the construction used in Theorem 1. A trace is fitting the overall model if and only if it fits all subnets. As Corollary 1 suggests, traces in the event log can be marked as fitting or non-fitting *per subnet*. These results can be merged easily and used to compute the *fraction of traces fitting the overall model*. Note that Corollary 1 holds for any decomposition of SN.

6.2 Decomposing Process Discovery

Process discovery, i.e., discovering a process model from event data, is highly related to conformance checking. This can be observed when considering genetic process mining algorithms that basically "guess" models and recombine parts of models that have good fitness to discover even better models [30,65]. The fitness computation in genetic process mining is in essence a conformance check.

Using Theorem 1 we can distribute *any* process discovery algorithm by (1) *decomposing the overall event log into smaller sublogs*, (2) *discovering a model fragment for each sublog*, and (3) *merging the discovered models into one overall process model*.

Given an event log L containing events referring to a set of activities A, we decompose discovery by distributing the activities over multiple sets $A^1, A^2, \ldots, A^n$. The same activity may appear in multiple sets as long as $A = A^1 \cup A^2 \cup \ldots \cup A^n$. For each activity set A_i, we discover a system net $\overline{SN}^i$ by applying a discovery algorithm to sublog $L \restriction_{A^i}$, i.e., the overall event log projected onto a subset of activities. Subsequently, the discovered system nets are massaged to avoid name clashes and to make sure that transitions with a visible label are merged properly. By combining the resulting system nets we obtain an overall system net $SN = SN^1 \cup SN^2 \cup \ldots \cup SN^n$ describing the behavior in the overall event log L [7].

The quality of the system net obtained by merging the process models discovered for the sublogs highly depends on the way the activities are decomposed and the characteristics of the discovery algorithm used per sublog. However, the system nets discovered for the sublogs are always a valid decomposition of the overall model. This implies that we can apply Theorem 1 (Conformance Checking Can be Decomposed), Theorem 2 (Lower Bound for Misalignment Costs), and Corollary 1 (Fraction of Perfectly Fitting Traces). If the discovery algorithm is able to create a perfectly fitting model for each sublog, then the overall model is also perfectly fitting. Moreover, if the discovery algorithm has problems finding a perfectly fitting model for a particular sublog, then the overall model will also exhibit these problems. For example, the fraction of traces fitting the overall model equals the fraction of traces fitting all individual models.

6.3 Decomposition Strategies

The decomposition results for conformance checking and process discovery are not tailored towards a particular type of decomposition. Recently, several papers have been published on different ways of decomposing process mining problems. In [3,88,89] it is shown that so-called "passages" can be used to decompose both process discovery and conformance checking problems. In [70,71] it is shown that so-called SESE (Single-Exit-Single-Entry) components obtained through the Refined Process Structure Tree (RPST) [74,87] can be used to decompose conformance checking problems. These papers use a particular decomposition strategy. However, as shown in [7], there are many ways to decompose process mining problems.

The above papers are all using Petri nets as a representation. However, as shown in [5] the essence of the decomposition results is not limited to Petri nets at al.

Experimental results shows that significant speed-ups are possible through decomposition [70,88]. Process mining algorithms are typically linear in the number of cases and exponential in the average length of traces or the number of unique activities. Through a vertical partitioning [4] many process mining algorithms can be decomposed trivially. Consider for example conformance checking problems. These are solved per case. Hence, by distributing the cases over different computing nodes it is easy to realize a linear speedup. Discovery algorithms

often use some variant of the "directly follows" relation ($>_L$) discussed in the context of the α-algorithm. Obviously a vertical partitioning (e.g. using a Map-Reduce [40,75] programming style) can be used to create such a relation in a distributed fashion.

In this paper we focused on a horizontal partitioning of the event log because process mining algorithms tend to be exponential in the average length of traces or the number of unique activities. Hence, the potential gains of horizontal partitioning are much larger. Just like the state space of a Petri net may grow exponentially in the number of transitions, the search space may grow rapidly as the number of different activities increases. Hence, the time needed to solve "many smaller problems" is often less than the time needed to solve "one big problem", even when this is done sequentially. In fact, horizontal partitioning may lead to super linear speedups. Consider for example conformance checking approaches that use state-space analysis (e.g., in [79] the shortest path enabling a transition is computed) or optimization over all possible alignments (e.g., in [22] the A^* algorithm is used to find the best alignment). These techniques do *not* scale linearly in the number of activities. Therefore, decomposition is often useful even if the checks per subnet are done on a single computer. Moreover, decomposing conformance checking is not just interesting from a performance point of view: decompositions can also be used to pinpoint the most problematic parts of the process and provide localized diagnostics [70]. Decompositions are not just interesting for conformance diagnostics; also performance-related diagnostics (e.g., bottleneck analysis) benefit from a hierarchical structuring of the whole process.

7 Conclusion

The torrents of event data available in most organizations enable *evidence-based Business Process Management* (ebBPM). We predict that there will be a remarkable shift from pure model-driven or questionnaire-driven approaches to data-driven process analysis as we are able to monitor and reconstruct the real business processes using event data. At the same time, we expect that machine learning and data mining approaches will become more *process-centric*. Thus far, the machine learning and data mining communities have not been focusing on end-to-end processes that also exhibit concurrency. Hence, it is time to move beyond decision trees, clustering, and (association) rules.

Process mining can be used to diagnose the actual processes. This is valuable because in many organizations most stakeholders lack a correct, objective, and accurate view on important operational processes. Process mining can subsequently be used to improve such processes. Conformance checking can be used for auditing and compliance. By replaying the event log on a process model it is possible to quantify and visualize deviations. Similar techniques can be used to detect bottlenecks and build predictive models. Given the applicability of process mining, we hope that this tutorial encourages the reader to start using process mining today. The book [2] provides a comprehensive introduction into the process mining field. Moreover, the open source process mining

tool *ProM* can be downloaded from www.processmining.org. Many of the ideas developed in the context of *ProM* have been embedded in commercial tools such as Fluxicon's Disco (www.fluxicon.com), Perceptive Process Mining (www.perceptivesoftware.com), Celonis (www.celonis.de), and QPR ProcessAnalyzer (www.qpr.com). This illustrates the practical relevance of process mining.

In the last part of this tutorial we discussed some very general decomposition results. Clearly, highly scalable analysis approaches are needed to deal with the ever-growing amounts of event data. This requires additional research efforts. Moreover, we refer to the *Process Mining Manifesto* by the IEEE Task Force on Process Mining [57] for additional challenges in this exciting new research field.

Acknowledgements. This work was supported by the Basic Research Program of the National Research University Higher School of Economics (HSE).

References

1. van der Aalst, W.M.P.: Formalization and verification of event-driven process chains. Inf. Softw. Technol. **41**(10), 639–650 (1999)
2. van der Aalst, W.M.P.: Process Mining: Discovery Conformance and Enhancement of Business Processes. Springer, Berlin (2011)
3. van der Aalst, W.M.P.: Decomposing process mining problems using passages. In: Haddad, S., Pomello, L. (eds.) PETRI NETS 2012. LNCS, vol. 7347, pp. 72–91. Springer, Heidelberg (2012)
4. van der Aalst, W.M.P.: Distributed process discovery and conformance checking. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 1–25. Springer, Heidelberg (2012)
5. van der Aalst, W.M.P.: A general divide and conquer approach for process mining. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (eds.) Federated Conference on Computer Science and Information Systems (FedCSIS 2013), pp. 1–10. IEEE Computer Society (2013)
6. van der Aalst, W.M.P.: Business process management: a comprehensive survey. ISRN Softw. Eng. 1–37 (2013). doi:10.1155/2013/507984
7. van der Aalst, W.M.P.: Decomposing Petri nets for process mining: a generic approach. Distrib. Parallel Databases **31**(4), 471–507 (2013)
8. van der Aalst, W.M.P.: Mediating between modeled and observed behavior: the quest for the "Right" process. In: IEEE International Conference on Research Challenges in Information Science (RCIS 2013), pp. 31–43. IEEE Computing Society (2013)
9. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Min. Knowl. Disc. **2**(2), 182–192 (2012)
10. van der Aalst, W., Adriansyah, A., van Dongen, B.: Causal nets: a modeling language tailored towards process discovery. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 28–42. Springer, Heidelberg (2011)
11. van der Aalst, W.M.P., Basten, T.: Identifying commonalities and differences in object life cycles using behavioral inheritance. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 32–52. Springer, Heidelberg (2001)

12. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data Knowl. Eng. **47**(2), 237–267 (2003)
13. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: Soundness of workflow nets: classification, decidability, and analysis. Formal Aspects Comput. **23**(3), 333–363 (2011)
14. van der Aalst, W.M.P., van Hee, K.M., van der Werf, J.M., Verdonk, M.: Auditing 2.0: using process mining to support tomorrow's auditor. IEEE Comput. **43**(3), 90–93 (2010)
15. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distribut. Parallel Databases **14**(1), 5–51 (2003)
16. van der Aalst, W.M.P., Lassen, K.B.: Translating unstructured workflow processes to readable BPEL: theory and implementation. Inf. Softw. Technol. **50**(3), 131–159 (2008)
17. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond process mining: from the past to present and future. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 38–52. Springer, Heidelberg (2010)
18. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A.K., Song, M., Verbeek, H.M.W.: Business process mining: an industrial application. Inf. Syst. **32**(5), 713–732 (2007)
19. van der Aalst, W.M.P., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, E., Günther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting. Softw. Syst. Model. **9**(1), 87–111 (2010)
20. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011)
21. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
22. Adriansyah, A., van Dongen, B., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: Chi, C.H., Johnson, P. (eds.) IEEE International Enterprise Computing Conference (EDOC 2011), pp. 55–64. IEEE Computer Society (2011)
23. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Towards robust conformance checking. In: zur Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 122–133. Springer, Heidelberg (2011)
24. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 137–149. Springer, Heidelberg (2013)
25. Adriansyah, A., Sidorova, N., van Dongen, B.F.: Cost-based fitness in conformance checking. In: International Conference on Application of Concurrency to System Design (ACSD 2011), pp. 57–66. IEEE Computer Society (2011)
26. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Saltorè, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
27. Badouel, E., Darondeau, P.: Theory of regions. In: Reisig, W., Rozenberg, G. (eds.) APN 1998. LNCS, vol. 1491, pp. 529–586. Springer, Heidelberg (1998)
28. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process mining based on regions of languages. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 375–383. Springer, Heidelberg (2007)
29. Chandra Bose, R.P.J.C.: Process mining in the large: preprocessing, discovery, and diagnostics. Ph.D. thesis, Eindhoven University of Technology (2012)

30. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: On the role of fitness, precision, generalization and simplicity in process discovery. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 305–322. Springer, Heidelberg (2012)
31. Calders, T., Guenther, C., Pechenizkiy, M., Rozinat, A.: Using minimum description length for process mining. In: ACM Symposium on Applied Computing (SAC 2009), pp. 1451–1455. ACM Press (2009)
32. Carmona, J., Cortadella, J.: Process mining meets abstract interpretation. In: Balcázar, J., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part I. LNCS, vol. 6321, pp. 184–199. Springer, Heidelberg (2010)
33. Carmona, J., Cortadella, J., Kishinevsky, M.: A region-based algorithm for discovering Petri nets from event logs. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 358–373. Springer, Heidelberg (2008)
34. Cook, J.E., Wolf, A.L.: Discovering models of software processes from event-based data. ACM Trans. Software Eng. Methodol. **7**(3), 215–249 (1998)
35. Cook, J.E., Wolf, A.L.: Software process validation: quantitatively measuring the correspondence of a process to a model. ACM Trans. Software Eng. Methodol. **8**(2), 147–176 (1999)
36. Cortadella, J., Kishinevsky, M., Lavagno, L., Yakovlev, A.: Deriving Petri nets from finite transition systems. IEEE Trans. Comput. **47**(8), 859–882 (1998)
37. Curbera, F., Doganata, Y., Martens, A., Mukhi, N.K., Slominski, A.: Business provenance - a technology to increase traceability of end-to-end operations. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 100–119. Springer, Heidelberg (2008)
38. Datta, A.: Automating the discovery of As-Is business process models: probabilistic and algorithmic approaches. Inf. Syst. Res. **9**(3), 275–301 (1998)
39. Davidson, S., Cohen-Boulakia, S., Eyal, A., Ludaescher, B., McPhillips, T., Bowers, S., Anand, M., Freire, J.: Provenance in scientific workflow systems. Data Eng. Bull. **30**(4), 44–50 (2007)
40. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
41. Depaire, B., Swinnen, J., Jans, M., Vanhoof, K.: A process deviation analysis framework. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 701–706. Springer, Heidelberg (2013)
42. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: metrics and evaluation. Inf. Syst. **36**(2), 498–516 (2011)
43. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
44. van Dongen, B.F., van der Aalst, W.M.P.: Multi-phase process mining: building instance graphs. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 362–376. Springer, Heidelberg (2004)
45. van Dongen, B.F., van der Aalst, W.M.P.: Multi-phase mining: aggregating instances graphs into EPCs and Petri nets. In: Marinescu, D. (ed.) Proceedings of the Second International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management, pp. 35–58. Florida International University, Miami (2005)
46. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle time prediction: When will this case finally be finished? In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 319–336. Springer, Heidelberg (2008)

47. van Dongen, B.F., Alves de Medeiros, A.K., Wen, L.: Process mining: overview and outlook of Petri net discovery algorithms. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 225–242. Springer, Heidelberg (2009)
48. Ehrenfeucht, A., Rozenberg, G.: Partial (Set) 2-Structures - Part 1 and Part 2. Acta Inf. **27**(4), 315–368 (1989)
49. Fahland, D., van der Aalst, W.M.P.: Repairing process models to reflect reality. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 229–245. Springer, Heidelberg (2012)
50. Gaaloul, W., Gaaloul, K., Bhiri, S., Haller, A., Hauswirth, M.: Log-based transactional workflow mining. Distrib. Parallel Databases **25**(3), 193–240 (2009)
51. van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. J. ACM **43**(3), 555–600 (1996)
52. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust process discovery with artificial negative events. J. Mach. Learn. Res. **10**, 1305–1340 (2009)
53. Günther, C.W., van der Aalst, W.M.P.: Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
54. Herbst, J.: A machine learning approach to workflow management. In: Lopez de Mantaras, R., Plaza, E. (eds.) ECML 2000. LNCS (LNAI), vol. 1810, pp. 183–194. Springer, Heidelberg (2000)
55. Hilbert, M., Lopez, P.: The world's technological capacity to store, communicate, and compute information. Science **332**(6025), 60–65 (2011)
56. Hinz, S., Schmidt, K., Stahl, C.: Transforming BPEL to Petri nets. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 220–235. Springer, Heidelberg (2005)
57. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011 Workshops, Part I. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
58. Jin, T., Wang, J., Wen, L.: Efficient retrieval of similar business process models based on structure. In: Meersman, R., et al. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 56–63. Springer, Heidelberg (2011)
59. Jin, T., Wang, J., Wen, L.: Efficient retrieval of similar workflow models based on behavior. In: Sheng, Q.Z., Wang, G., Jensen, C.S., Xu, G. (eds.) APWeb 2012. LNCS, vol. 7235, pp. 677–684. Springer, Heidelberg (2012)
60. Lohmann, N., Massuthe, P., Stahl, C., Weinberg, D.: Analyzing interacting BPEL processes. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 17–32. Springer, Heidelberg (2006)
61. Ludaescher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency Comput. Pract. Experience **18**(10), 1039–1065 (2006)
62. Maggi, F.M., Montali, M., van der Aalst, W.M.P.: An operational decision support framework for monitoring business constraints. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 146–162. Springer, Heidelberg (2012)
63. Maggi, F.M., Westergaard, M., Montali, M., van der Aalst, W.M.P.: Runtime verification of LTL-based declarative process models. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 131–146. Springer, Heidelberg (2012)
64. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Byers, A.: Big data: the next frontier for innovation, competition, and productivity. McKinsey Global Institute (2011)

65. Alves de Medeiros, A.K., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic process mining: an experimental evaluation. Data Min. Knowl. Disc. **14**(2), 245–304 (2007)
66. Mendling, J., van Dongen, B.F., van der Aalst, W.M.P.: On the degree of behavioral similarity between business process models. In: Nuettgens, M., Rump, F.J., Gadatsch, A. (eds.) Proceedings of Sixth Workshop on Event-Driven Process Chains (WI-EPK 2007), St. Augustin, November 2007, pp. 39–58. Gesellschaft für Informatik, Bonn (2007)
67. Milner, R.: Communication and Concurrency. Prentice-Hall Inc., Upper Saddle River (1989)
68. Munoz-Gama, J., Carmona, J.: A fresh look at precision in process conformance. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 211–226. Springer, Heidelberg (2010)
69. Munoz-Gama, J., Carmona, J.: Enhancing precision in process conformance: stability, confidence and severity. In: Chawla, N., King, I., Sperduti, A. (eds.) IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011), Paris, France, April 2011, pp. 184–191. IEEE (2011)
70. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Conformance checking in the large: partitioning and topology. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 130–145. Springer, Heidelberg (2013)
71. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Hierarchical conformance checking of process models based on event logs. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 291–310. Springer, Heidelberg (2013)
72. Ouyang, C., van der Aalst, W.M.P., Breutel, S., Dumas, M., ter Hofstede, A.H.M., Verbeek, H.M.W.: Formal semantics and analysis of control flow in WS-BPEL. Sci. Comput. Program. **67**(2–3), 162–198 (2007)
73. Ouyang, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Pattern-based translation of BPMN process models to BPEL web services. Int. J. Web Serv. Res. **5**(1), 42–62 (2007)
74. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: Bravetti, M., Bultan, T. (eds.) WS-FM 2010. LNCS, vol. 6551, pp. 25–41. Springer, Heidelberg (2011)
75. Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, Cambridge (2011)
76. Reijers, H.A.: Case prediction in BPM systems: a research challenge. J. Korean Inst. Ind. Eng. **33**, 1–10 (2006)
77. Reisig, W.: Petri Nets: Modeling Techniques, Analysis, Methods, Case Studies. Springer, Heidelberg (2013)
78. Rozinat, A., van der Aalst, W.M.P.: Decision mining in ProM. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 420–425. Springer, Heidelberg (2006)
79. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
80. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering colored Petri nets from event logs. Int. J. Softw. Tools Technol. Transfer **10**(1), 57–74 (2008)
81. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering simulation models. Inf. Syst. **34**(3), 305–327 (2009)
82. Rozinat, A., Wynn, M., van der Aalst, W.M.P., ter Hofstede, A.H.M., Fidge, C.: Workflow simulation for operational decision support. Data Knowl. Eng. **68**(9), 834–850 (2009)

83. Schonenberg, H., Weber, B., van Dongen, B.F., van der Aalst, W.M.P.: Supporting flexible processes through recommendations based on history. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
84. Sheth, A.: A new landscape for distributed and parallel data management. Distrib. Parallel Databases **30**(2), 101–103 (2012)
85. Solé, M., Carmona, J.: Process mining from a basis of state regions. In: Lilius, J., Penczek, W. (eds.) PETRI NETS 2010. LNCS, vol. 6128, pp. 226–245. Springer, Heidelberg (2010)
86. Staffware. Staffware Process Suite Version 2 - White Paper. Staffware PLC, Maidenhead, UK (2003)
87. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data Knowl. Eng. **68**(9), 793–818 (2009)
88. Verbeek, H.M.W., van der Aalst, W.M.P.: An experimental evaluation of passage-based process discovery. In: La Rosa, M., Soffer, P. (eds.) BPM 2012 Workshops. LNBIP, vol. 132, pp. 205–210. Springer, Heidelberg (2013)
89. Verbeek, H.M.W., van der Aalst, W.M.P.: Decomposing replay problems: a case study. BPM Center Report BPM-13-09. www.bpmcenter.org (2013)
90. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A robust F-measure for evaluating discovered process models. In: Chawla, N., King, I., Sperduti, A. (eds.) IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011), Paris, France, pp. 148–155. IEEE (2011)
91. Weidlich, M., Dijkman, R.M., Weske, M.: Behavior equivalence and compatibility of business process models with complex correspondences. Comput. J. **55**(11), 1398–1418 (2012)
92. Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering workflow models from event-based data using little thumb. Integr. Comput. Aided Eng. **10**(2), 151–162 (2003)
93. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming. Fundamenta Inf. **94**, 387–412 (2010)

Ontology-Driven Business Intelligence for Comparative Data Analysis

Thomas Neuböck[1], Bernd Neumayr[2], Michael Schrefl[2(✉)], and Christoph Schütz[2]

[1] Solvistas GmbH, Graben 18, 4020 Linz, Austria
thomas.neuboeck@solvistas.at
[2] Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria
{neumayr,schrefl,schuetz}@dke.uni-linz.ac.at

Abstract. In this tutorial, we present an ontology-driven business intelligence approach for comparative data analysis which has been developed in a joint research project, *Semantic Cockpit* (semCockpit), of academia, industry, and prospective users from public health insurers. In order to gain new insights into their businesses, companies perform comparative data analysis by detecting striking differences between different, yet similar, groups of data. These data groups consist of measure values which quantify real-world facts. Scores compare the measure values of different data groups. semCockpit employs techniques from knowledge-based systems, ontology engineering, and data warehousing in order to support business analysts in their analysis tasks. Concept definitions complement dimensions and facts by capturing relevant business terms which are used in the definition of measures and scores. Furthermore, domain ontologies serve as semantic dimensions and judgement rules externalize previous insights. Finally, we sketch a vision of analysis graphs and associated guidance rules to represent analysis processes.

Keywords: Business intelligence · OLAP · Data warehouses · Semantic technologies

1 Introduction

For their analysis tasks, business analysts rely on a data warehouse which organizes data as multi-dimensional facts. Each fact represents a business event that has been recorded in a transactional database. In the data warehouse, facts are identified by dimensions and quantified by measures. For example, the recipient patient, the issuing doctor, and the date of issuance identify a drug prescription. The prescribed quantity and the incurred costs are measures which quantify the drug prescription.

Business intelligence (BI) tools support interactive reporting over the corporate data warehouse through online analytical processing (OLAP). OLAP operations allow for the selection of different data groups and the aggregation

E. Zimányi (Ed.): eBISS 2013, LNBIP 172, pp. 77–120, 2014.
DOI: 10.1007/978-3-319-05461-2_3,

of measures. Business analysts employ OLAP operations for different types of analysis. First, *is-reporting* provides summary or detail information about a current or past business situation. For example, a health insurance manager might be interested in the total costs of drug prescriptions from last month. Second, *is-to-target comparison* contrasts a current or past business situation with a (hypothetical) target situation. For example, a health insurance company sets out a target figure for the monthly costs of drug prescriptions which a business analyst contrasts with the actually incurred costs from last month. Third, *is-to-is comparison* contrasts different, yet similar, business situations in order to gain insights into the analysis area. For example, a business analyst compares the incurred costs of drug prescriptions for different groups of patients in various months. In this tutorial, we focus on is-to-is comparison of data.

Whereas the tasks of is-reporting and is-to-target comparison tend to be simple and structured, is-to-is comparison is fundamentally more complex and often left to human intuition. The business analyst faces the challenge of defining meaningful comparisons. This definition demands knowledge about relevant business terms as well as their semantics. The business analyst must select relevant comparison groups, identify the subsets of facts to consider, and define the relevant measures for the illustration of the differences between the comparison groups. For example, a health insurance manager might be interested whether there are any exceptional differences between any groups of diabetes mellitus patients. The business analyst identifies patients from different insurance companies, provinces, and of different age as meaningful comparison groups. The business analyst considers only the facts that concern oral anti-diabetic drugs, insulin, and metformin. For the illustration of the differences between groups, the business analyst defines the average drug costs per patient and the drug costs that are prescribed by general practitioners for regular patients.

Comparative data analysis is an interactive, exploratory, and iterative process which employs OLAP for is-to-is comparison. Initially, relevant comparison groups and the measures for the illustration of the differences between the comparison groups are unknown. As a first step, the business analyst determines comparison groups and measures. Once comparison groups and measures are determined, the business analyst repeatedly conducts the comparative analysis with varying parameters. By varying the parameters of the analysis, the business analyst discovers dependencies among the data. Thereby comparative data analysis leads to the detection of exceptional differences between selected groups of data, suggests plausible explanations as well as cause-and-effect relationships, and highlights paths for further investigation. In this sense, comparative data analysis is not a replacement for data mining. Rather, comparative data analysis precedes data mining, assisting business analysts in the formulation of appropriate questions to statisticians.

Traditional BI tools fail to support the full process of comparative data analysis. The definition of business terms is left to the business analyst rather than providing a central repository which unambiguously defines the semantics of business terms. The comparison of data is a simple enumeration of selected measures, their interpretation left to the business analyst. In traditional BI tools, e.g.,

Tableau[1] and Oracle BI[2], business analysts conduct analysis in an ad-hoc manner with each analysis task started from scratch. The *Semantic Cockpit* (semCockpit) approach as presented herein fully supports comparative data analysis.

In semCockpit, a multi-dimensional ontology (MDO) provides an unambiguous definition of business terms for the specific needs of OLAP. Business terms are hierarchically ordered and become first-class citizens, which allows analysts to employ business terms in formulating OLAP queries. Furthermore, semantic dimensions allow for the integration of existing domain ontologies in OLAP.

Ontology-based measures and scores use concepts of ontologies to specify the data to be included in the calculation of derived measures and scores. Scores make comparison a first-class citizen in semCockpit. They capture the results of a comparison explicitly and free the business analyst from visual comparison by diagram inspection. A generic definition facilitates the reuse of scores in various analysis situations to avoid vast enumerations of similar measures.

Analysis graphs and rules capture otherwise tacit knowledge about how to proceed in analysis and about possible explanations of analysis results. Analysis graphs explicitly define the process of the analysis. Analysis rules recommend the initiation of a specific analysis process to the business analyst when certain conditions are met. Guidance rules lead the business analyst through the analysis graph. Judgement rules provide explanations for exceptional values. Analysis, guidance, and judgement rules externalize actionable knowledge otherwise tacit to the business analyst.

Existing BI approaches employ ontologies complementary to the semCockpit approach. Ontology-based data warehouse design [20,29,31,38] employs ontologies to automate tasks concerning construction of data warehouses and ETL processes. Ontologies serve as foundation for open access semantic-aware business intelligence [32]. Saggion and colleagues [34] employ ontologies for information extraction for business intelligence. Combining reasoning over ontologies and OLAP, Nebot and colleagues [24,25] build data warehouses for the analysis of semantic web data. The potential of ontology-based querying in business intelligence has been identified by Spahn and colleagues [40] but has not been elaborated for OLAP and multi-dimensional data warehouses.

The remainder of this tutorial is organized as follows. In Sect. 2, we present the semCockpit architecture and introduce a simplified real-world use case. In Sect. 3, we present the fundamentals of an MDO, including semantic dimensions. In Sect. 4, we investigate the definition of measures and scores based on the concepts of the MDO. In Sect. 5, we describe how the MDO can be beneficially applied for ontology-based comparative OLAP, thereby extending the well-known OLAP operations dice, slice, drill-down and roll-up for comparative analysis and making use of the MDO. In Sect. 6, we present a vision of BI analysis graphs for the definition of analysis processes. In Sect. 7, we introduce corresponding judgement and analysis rules for representing knowledge of the analyst.

[1] http://www.tableausoftware.com
[2] http://www.oracle.com

2 The semCockpit Approach

In this section, we describe the data warehouse behind a comparative data analysis project, introduce a case study, and identify the steps in a comparative data analysis project.

2.1 Data Warehouse

A data warehouse (DWH) typically organizes data as multi-dimensional facts, where each fact represents a business event that has been recorded in a transactional database, is identified by a node for each dimension, and described by one or several measures. Each dimension is given by a leveled hierarchy of nodes, whereby each node is described by a set of non-dimensional attributes and all nodes of the same level have the same attributes with different attribute values. The "base facts" of a data warehouse refer in each dimension to a leaf node of the dimension.

More specifically, a *data warehouse* consists of a set of dimensions and a set of fact classes. Each *dimension* has a dimension schema and a dimension instance. A *dimension schema* consists of a set of levels and a set of attributes for each level. Roll-up relationships organize the levels in a lattice. A *dimension instance* consists of a set of nodes where each node belongs to exactly one level and is described by a value for each attribute of the level. The nodes of a dimension are organized in a roll-up relationship that forms a semi-lattice such that each node of some level rolls up to exactly one node of each level that is in roll-up relationship to the level of the former node. Each dimension contains a single *top* level with a single *all* node to which all levels and all nodes of the dimension roll-up to. A *base fact class* consists of a fact schema and a set of facts. A *base fact schema* is given by a set of dimension roles and a set of measures (referred to as base measures). Each dimension role refers to a dimension schema (whereby two different dimension roles can refer to the same dimension schema). A *base fact* of a fact class refers for each dimension role to a leaf node in the respective dimension and is described by a measure value for each measure of the fact schema.

Example 1 (Existing Data Warehouse). Figure 1 illustrates a fragment of a simplified data warehouse schema of Austrian public health insurers, represented in a slight variation (explained later) of the Dimensional Fact Model (DFM) [12]. The dimensions are Insurant, Doctor, DrugATC (drug classification in accordance with ATC), and Time. *Top* levels of dimensions are not depicted. Medical sections (level medSec in dimension Doctor) denote specialisms of doctors, e.g., general practitioner (GP), internist, oculist. ATC is an abbreviation for Anatomical Therapeutic Chemical Classification System, which is an international classification system of drugs with the hierarchy levels for anatomical main group (atcAnatom), therapeutic main group (atcTherap), therapeutic/pharmacological subgroup (atcPharm), chemical/therapeutic/pharmacological subgroup (atcChemSubGr), and chemical substance. We omit the level for chemical substance.

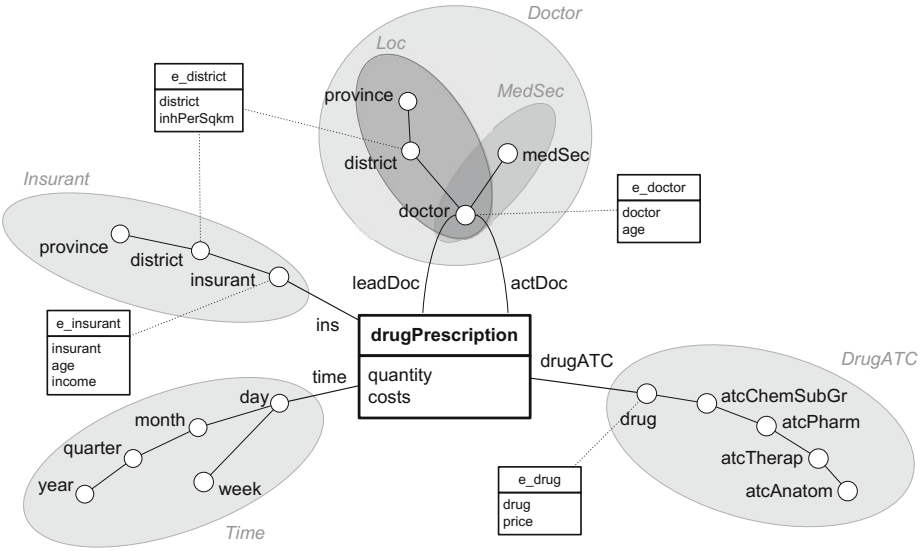


Fig. 1. DWH schema for drug prescriptions

The dimension roles of fact class drugPrescription are ins (insurant), leadDoc (lead doctor), actDoc (acting doctor), drugATC (drug classification according to ATC), and time. Dimension roles leadDoc and actDoc refer to the same dimension (Doctor). The fact class comprises measures quantity and costs of drug prescriptions.

In order to avoid redundant representations of attributes of levels the semDWH data model does not represent attributes with levels but with entity classes, where each level of a dimension schema refers to an entity class (and each node of that level to an entity of the entity class) and two levels of different dimensions (but not of the same dimension) can refer to the same entity class.

Example 2 (Entity Classes). Level district of dimension Doctor and level district of dimension Insurant refer to entity class e_district which specifies attributes district (the name of the district which is used as external identifier) and inhPerSqkm (the population density of the district).

As a means for specifying the domain of measures and multi-dimensional concepts, the semCockpit data model introduces the notion of a *dimension space*. A dimension space is defined by a set of dimension roles. A *point* in a dimension space is identified by a set of coordinates, one coordinate for each dimension role. Each coordinate of a point refers to a node of the dimension referred to by the dimension role. A *fact* is a point described by measure values. A dimension space comprises a point for each tuple in the cross product of the nodes in the dimension roles of the dimension space. A *granularity* in a dimension space is identified by a set of levels one for each dimension role. A lattice of granularities, from finer to coarser, can be derived from the hierarchies of levels of the dimensions referred to by the dimension roles. A dimension space may be restricted to

points at a particular granularity or to points that fall between a "from" granularity and a "to" granularity (both inclusive). Each fact class is defined over a dimension space restricted to a single granularity.

Example 3 (Dimension Space). In Fig. 1, the domain of measures quantity and costs of fact class drugPrescription is given by a dimension space which is defined by dimension roles time, ins, leadDoc, actDoc, and drugATC, restricted to the finest granularity [time:day, ins:insurant, leadDoc:doctor, actDoc:doctor, drugATC:drug]. A point in this dimension space, for example ⟨time:20130708, ins:mrHuber, leadDoc:drMaier, actDoc:drMueller, drugATC:paracetamol500mg⟩, may be described by base measures quantity and costs. Roll-up to coarser granularities together with aggregation of measures (for example SUM(costs)) may be possible to all points in the dimension space which is given by the same dimension roles but not restricted to the finest granularity, for example also to points like ⟨time:2013, ins:linz, leadDoc:GP, actDoc:all, drugATC:all⟩.

Each dimension is organized into one or more *roll-up hierarchies* (also referred to as *roll-up paths* or simply *hierarchies*). Per default, these hierarchies are alternative hierarchies, that is, each point in the dimension space is identified by exactly one coordinate per dimension role. The semDWH data model also allows for parallel roll-up, that is, points may have separate coordinates for the different roll-up paths of a dimension. Dimension spaces may then be defined over *dimension roles* and *hierarchy-specific dimension roles*. A hierarchy-specific dimension role is defined over a named roll-up path of a dimension.

Example 4 (Roll-up Hierarchies). In Fig. 1, dimension Time has two alternative hierarchies, one with a roll-up path along day, month, quarter, and year, and the other one along day and week, which cannot be used simultaneously in one query. Dimension Doctor has named roll-up paths Loc (location) and MedSec (medical section). Hierarchy-specific dimension roles actDocMedSec and actDocLoc as well as leadDocMedSec and leadDocLoc can be used to define dimension spaces that allow for parallel roll-up.

To simplify the later definition of concepts and their use in measure and score definitions and applications, we assume that dimension roles that are used in multiple fact classes carry the same meaning in each of these fact classes. This approach is akin to the unique role assumption for attributes in relational database systems. To support the unique role assumption for dimension roles, the semDWH data model provides for the definition of a *universal dimension space* consisting of all dimension roles of a semDWH, where each dimension role is identified by a unique name and described by the dimension it refers to. The universal dimension space also comprises all hierarchy-specific dimension roles. All other dimension spaces may not contain both, a dimension role and one of its hierarchy-specific dimension roles.

Example 5 (Universal Dimension Space). Fact class ambTreatment (ambulant treatments) has the same dimension roles as fact class drugPrescription but it

possesses dimension role medServItem (medical service items) instead of dimension role drugATC. Examples of medical service items are doctor visits and blood glucose examinations. The universal dimension space comprises the dimension roles of both fact classes, drugPrescription and ambTreatment. The dimension spaces DrugPrescriptionSpace and AmbTreatmentSpace only contain the dimension roles of the fact classes drugPrescription and ambTreatment, respectively.

In general, OLAP operations allow to join facts over different dimension spaces in drill-across operations. The result of such an operation is a fact over a new dimension space, whereby the dimension roles of the drill-across fact are mapped to the dimension roles of the joined facts. Accordingly, a *drill-across* dimension space can be defined based on two other dimension spaces (by mapping dimension roles based on common names like in natural joins for relations, or explicitly, like in equi-joins). It comprises the union of dimension roles.

The semCockpit data warehouse (semDWH) can be defined in two ways. First, the semDWH can be defined from scratch using a simple data definition language (DDL). Each DDL statement of a semDWH construct has a corresponding relational representation (for dimensions and fact classes). Actual data have to be provided thereafter as materialized or virtual views over the enterprise DWH according to this relational representation. We do not describe this schema and instance mapping problem and refer to the relevant literature [4] instead. Second, the semDWH can be defined by immediately generating the appropriate relational representation (i.e., materialized or virtual views) of the semDWH. The former approach is more appropriate for reuse in similar settings (e.g., health insurers in different states of Austria), the latter for single in-house projects.

2.2 Use Case

Effective and efficient medical care is an overall goal of public health insurance companies. Comparative data analysis often focuses on diseases that are responsible for high overall costs. For example, diabetes mellitus of type 2 (DM2) is one specific example of a lifestyle disease with high prevalence that causes high costs. Disease management programs (DMP) have been established to provide effective and cost efficient treatment of DM2 patients.

In this context the managerial accounting department of a public health insurer might recognize an above-average increase of total costs concerning the treatment of DM2 patients. The management asks the business intelligence department to analyze the issue by finding striking differences through comparison. In the subsequent comparative data analysis processes, a business analyst may compare different groups of patients (rural vs. urban districts, young vs. old, DMP patients vs. non-DMP patients, etc.), different groups of doctors, different drugs, different insurers, or different periods.

The analysis process is interactive, exploratory, and iterative. The analyst starts with a vague analysis question and interacts with various domain experts

to discuss what kinds of comparison might be relevant or should be chosen next. The interesting comparisons develop over time. Once successful and relevant sequences of analysis steps have been discovered, they can be described in generalized form for later re-use in analogous situations (e.g., for other years, insurers, or diseases).

2.3 Steps in a Comparative Data Analysis Project

Comparative data analysis focuses on the interactive comparison of various sets of data. Such comparisons are based on measures and scores. A measure describes a multi-dimensional point which consists of nodes from data warehouse dimensions; a point and a measure together give a *fact*. A score describes a relationship between a pair of points, the point of interest and the point of comparison; score, point of interest (PoI) and point of comparison (PoC) together give a *comparative fact* which explicitly expresses the result of a comparison that would otherwise have been left to the human eye. Thus, relating our DWH model to the Entity-Relationship model, points correspond to entities, measures to attributes of entities, and scores to attributes of binary relationships between entities. Since a comparative fact actually relates, via aggregation, sets of facts that are compared, we speak also of group of interest (GoI) and group of comparison (GoC).

The typical steps in applying an ontology-driven business intelligence approach for comparative data analysis are: (1) *Model transformation*, in which a given DWH schema is transformed into a semDWH data warehouse schema as introduced in Subsect. 2.1, (2) *Semantic Enrichment*, in which business terms are expressed as concepts in a multi-dimensional ontology (MDO) or imported from an external domain ontology, (3) *Calculation Definition*, in which measures and scores are defined by calculations over other measures and scores, employing concepts of the MDO to select the data to be included in calculations, (4) *Explorative Measure and Score Application*, in which measures and scores are applied to different points of interest and comparison, (5) *Analysis Design*, in which promising sequences of measure and score applications are modeled as BI analysis graphs for later re-use, (6) *Rule Design*, in which different kinds of rules are designed to complement analysis: (a) guidance rules that provide context-sensitive semantic guidance on which path to follow in an instantiation of an analysis graph, (b) judgement rules that provide background information about possible reasons for a striking score, and (c) analysis rules that express how to react on specific measures and scores detected (action rules) or provide a concise analysis report (reporting rules), (7) *Proper comparative data analysis*, in which BI analysis graphs are instantiated and traversed for particular analysis problems, thereby possibly backtracking to any of the previous steps. – Steps 1 to 4 and 6 (b,c) have been investigated in the semCockpit project and implemented in our semCockpit prototype. Steps 5, 6 (a), and 7 have been identified during the project as beneficial future extensions to capture and exploit – next to "static" knowledge – knowledge about analysis processes.

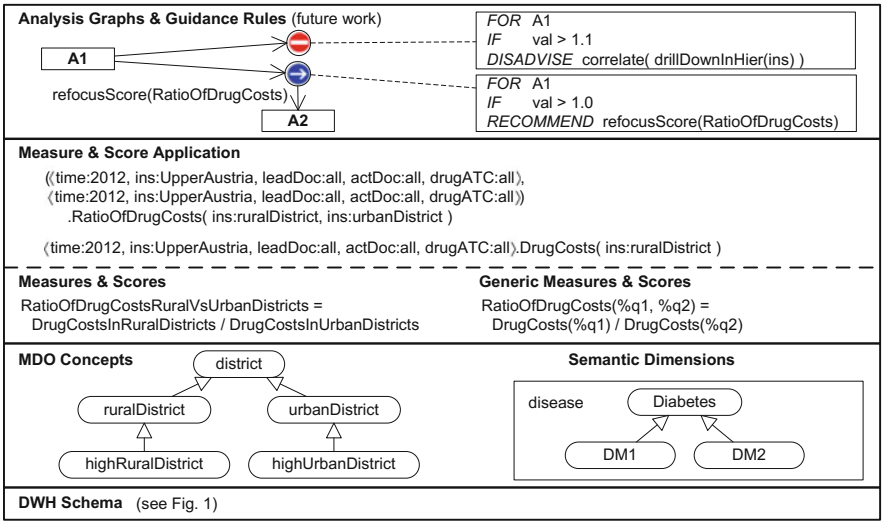


Fig. 2. The semCockpit stack

The semCockpit stack (Fig. 2) reflects the typical steps in the application of an ontology-driven BI approach for comparative data analysis. Its starting point (bottom of Fig. 2) is founded on the DWH schema as described in example 1.

MDO concepts are defined by logical expressions over nodes of a dimension. MDO concepts are organised in subsumption hierarchies through reasoning over concept expressions. Furthermore terms of an external domain ontology such as SNOMED CT[3] can be used as semantic dimensions in the way that leaf concepts of the external ontology classify facts.

Example 6 (MDO concepts and semantic dimensions). The MDO concepts ruralDistrict and urbanDistrict in Fig. 2 are defined over nodes of a location dimension (logical definitions are omitted), organized in a subsumption hierarchy. The semantic dimension disease incorporates an OWL representation of a subset of SNOMED CT.

Measures describe by measurement instructions how a particular measure value is calculated from facts in the DHW for a point. Measures are *ontology-based* in the sense that measurement instructions refer to MDO-concepts to identify facts to be included in (parts of) calculations. Similarly, scoring instructions describe for scores how a score value is calculated for a pair of points.

Example 7 (Measures and Scores). Figure 2 shows two measures, DrugCostsInRuralDistricts and DrugCostsInUrbanDistricts, that calculate the total costs for drugs in rural and urban districts, respectively. The measurement instructions (omitted) refer to the respective ontology concepts, ruralDistrict and urbanDistrict.

[3] Systematized Nomenclature Of Medicine Clinical Terms.

The score RatioOfDrugCostsRuralVsUrbanDistricts compares the total costs for drugs in rural districts against urban districts (by using the appropriate measures in the scoring instructions, not shown).

Generic measures and scores avoid the need of repeated definitions of instructions that are identical apart from a particular MDO-concept. They are defined with parameters for concepts. We refer to these parameters also as (generic) qualifiers as they are used to select facts.

Example 8 (Generic Measures and Scores). Generic measure DrugCosts has a qualifier, which can be instantiated for example by a location concept. If instantiated, e.g., with ins:urbanDistrict, the instantiation gives a non-generic measure, e.g., DrugCostsInUrbanDistricts. Similarly, generic score RatioOfDrugCosts has two qualifiers, one for the group of interest, one for the group of comparison.

Once defined, measure and scores can be applied to multi-dimensional points (shown in the next stage of the semCockpit stack). Additionally, they can be qualified by MDO concepts.

Example 9 (Measure and Score Application). Figure 2 shows an application of generic measure DrugCosts to multi-dimensional point ⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩ with actual qualifier ins:ruralDistrict giving the total drug costs prescribed in rural districts in Upper Austria in the year 2012. The application of generic score RatioOfDrugCosts to group of interest ⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩ with qualifier ins:ruralDistrict and group of comparison ⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drug ATC:all⟩ with qualifier ins:urbanDistrict returns the ratio of drug costs of Upper Austrian patients of rural versus urban districts in year 2012.

An analysis graph has analysis situations as nodes and analysis steps as arcs and describes promising analysis processes. Analysis situations are parameterized cubes, analysis steps are parameterized navigation operations, e.g., drill-down. The instantiation of the parameters leads to a specific BI analysis.

Depending on measure and score values, guidance rules open or close analysis paths. Judgement rules provide background information on striking score values, and analysis rules describe how to react upon (action rules) or which specific comparative facts to report (reporting rules) from a set of comparative facts provided (e.g., those loaded in the last ETL cycles).

Example 10 (Analysis Graphs and Rules). The top compartment of Fig. 2 depicts a simple analysis graph consisting of two analysis situations, A1 and A2, and two guidance rules. For example, A1 may represent the ratio of ambulant treatment costs for some selected comparison. If the result is greater than 1.1, the associated guidance rule disadvises to drill down in the insurant hierarchy. If the result is greater than 1.0, the associated guidance rule suggests to refocus in A2 the analysis on the ratio of drug costs. The figure assumes that for the currently selected comparison A1 the first rule does not apply and the second rule applies.

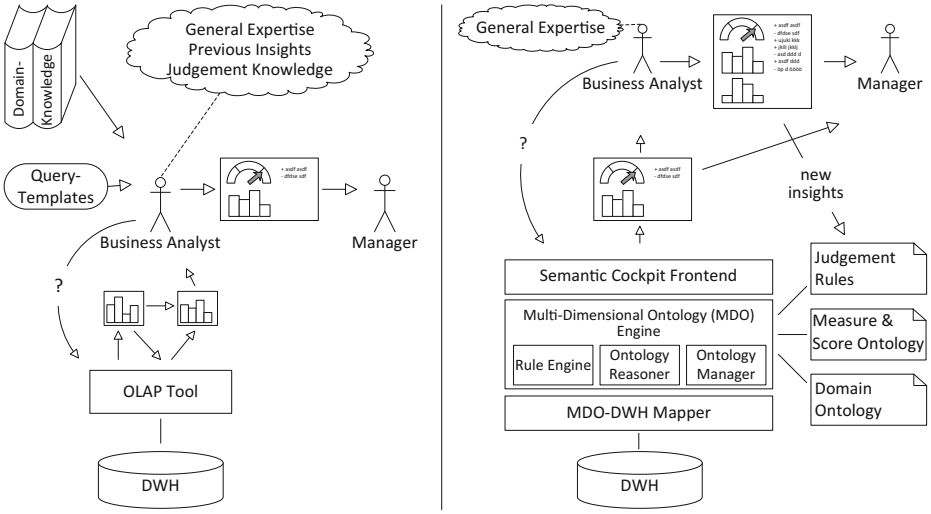


Fig. 3. Conventional Comparative Data Analysis (left) vs. Semantic Cockpit: Setting & Components (right)

Figure 3 gives an overview of the semCockpit architecture. The left-hand side illustrates conventional comparative data analysis in which a business analyst uses OLAP tools on an operative level and domain knowledge is not represented. The right-hand side illustrates the semCockpit approach in which domain knowledge is captured by a multi-dimensional ontology (MDO) that describes relevant business terms in the context of business analysis and which may relate to a domain ontology. Measures and scores are defined based on the ontology. Previous insights and analysis experience are captured by judgement rules. In order to implement these features, semCockpit comprises several components. The MDO-DWH Mapper accesses DWH data that is used by other components. The MDO Engine administers concepts, measures and scores, and organizes them by reasoning. The management and evaluation of judgement rules is carried out by the Rule Engine. Finally the semCockpit Frontend provides an appropriate user interface. – The planned future extensions, analysis graphs and guidance rules, are not shown.

3 Multi-Dimensional Ontology

In this section, we present the representation of business terms as concepts of a multi-dimensional ontology (MDO), the handling of context-specific concepts, the seamless import of a domain ontology into the MDO as *semantic dimension*, and the translation of concepts into SQL for querying and into OWL for determining subsumption hierarchies by OWL reasoners.

In the semCockpit approach, a multi-dimensional ontology is managed as a central repository of business terms that are defined and maintained collaboratively by business analysts. Business terms can be translated automatically

to SQL and simplify the formulation of otherwise complex multi-dimensional queries. Automatically-derived subsumption hierarchies simplify the organization of business terms and the detection of similar or redundant concepts. In analysis sessions, subsumption hierarchies allow to move up or down the hierarchy to 'broaden' or to 'narrow' the selection predicate of a query.

With regard to querying, the semCockpit approach assumes that the data in the underlying data warehouse is complete (closed world assumption). With regard to subsumption reasoning, the approach does not consider the complete data in the data warehouse but only the incomplete knowledge represented in the ontology (open world assumption) because subsumption hierarchies should not change due to changes in the data of the data warehouse.

Using ontologies with defined classes for querying data- or knowledge bases has seen considerable attention in the literature. Staudt et al. [41] discuss the definition of query classes in deductive databases. The partitioning of the terminological box of an ontology into a schema part and a view part, with distinct language constructs for either part, as proposed by Buchheit et. al. [5], is of particular importance for MDOs. The MASTRO system [6] for ontology-based data access uses ontologies for querying incomplete (relational) databases. Lim et al. [22] employ virtual views as a query interface for semantically-enriched relational data.

3.1 Concepts: Signatures and Concept Expressions

The MDO enriches the underlying semDWH by a set of concepts representing business terms and their meaning in the context of data analysis. A concept may be defined over (a) entities of an entity class (*entity concepts*), (b) nodes of a dimension (*dimensional concept*), (c) points of a dimension space (*multi-dimensional concept*, *md-concept*), or (d) pairs of points, referred to as point of interest (PoI) and point of comparison (PoC), (*comparative concept*).

Each concept has a *signature* and a *membership condition*, which may be defined independently of each other. The separation of signature and membership condition is a prerequisite for specializing membership conditions for different contexts (see next subsection).

The *signature* of a concept is given by a name and an interpretation domain for the concept. The interpretation domain is the set of individuals for which the concept is defined. It is given by the sort of individuals over which the concept is defined (entities of an entity class, nodes of a dimension, points of a dimension space, point-pairs of a comparative space consisting of two dimension spaces) and is possibly restricted to some subset of the sort. E.g., the interpretation domain of a dimensional concept may be restricted to nodes of some level or nodes that fall into a level range. A point satisfies a multi-dimensional concept, if the point satisfies the concept for the dimension roles for which it is defined. (Notice that this interpretation is consistent with classifying individuals in ontologies where properties not referred to in a concept expression are ignored.) This holds analogously for pairs of points and comparative concepts. The signature of a

concept does not need be explicitly stated but may also be derived from its membership condition, which is a concept expression.

Example 11 (Signature). Signature ruralDistrict(e_district) describes an entity concept called ruralDistrict with interpretation domain entity class e_district. InOAD(DrugATC[drug .. atcPharm]) represents the signature of a dimensional concept. It indicates that concept InOAD refers to dimension DrugATC and comprises nodes between level drug and atcPharm (both inclusive). Signature PatInRuralDistrLeadDocInUrbanDistr(ins[insurant .. district], leadDoc[doctor .. district]) represents the multi-dimensional concept PatInRuralDistrLeadDocInUrbanDistr that comprises points over insurants (dimension role ins) and lead doctors (dimension role leadDoc) at granularity range from level insurant to district and from level doctor to district, respectively. The signature PatWithRegularDocVisitsInYear(ins[insurant], time[year]) restricts the multi-dimensional concept PatWithRegularDocVisitsInYear to level insurant for dimension role ins and to level year for dimension role time.

The membership condition (also often referred to as necessary & sufficient condition) is given by a concept expression in a simple, high-level MDO language (surveyed below) or by an SQL view over the underlying semDWH. SQL-defined concepts provide for extensibility and are treated as primitive when determining subsumption hierarchies between concepts (see Subsect. 3.4). The MDO language has been designed to support the most important use cases and to provide for a mapping [28] into OWL 2 DL.

Example 12 (Entity concepts on district). The concepts ruralDistrict, urbanDistrict, highRuralDistrict, and highUrbanDistrict presented in Fig. 4 are defined over entity e_district. Membership conditions are denoted next to the box of the concept, e.g., inhPerSqkm <= 400 represents the expression for concept ruralDistrict. As discussed later, the reasoner will detect that concept highUrbanDistrict is subsumed by concept urbanDistrict because expression inhPerSqkm $>$ 1000 implies inhPerSqkm $>$ 400. In Fig. 4 inferred subsumption relationships between concepts are denoted by arrows. Dotted lines link entity concepts to their entity class. For better readability this type of lines are omitted for subsumed concepts like highUrbanDistrict.

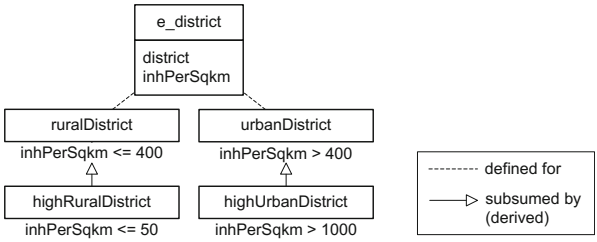


Fig. 4. Entity concepts on district

The membership condition of a dimensional concept is given by one of the following kinds of concepts expressions: (1) by a reference to an entity concept (such that each node that refers to an entity satisfying the entity concept is in the interpretation of the defined concept), (2) by hierarchy expansion of some concept (such that each node of the dimension that is in the interpretation of the concept or some direct or indirect successor node thereof is in the interpretation), (3) by level range restriction of some concept (such that only nodes of that concept that fall in between an indicated top and bottom level, inclusive, are in the interpretation), (4) by intersection, union, or complement (open world interpretation) of concepts defined for the same level (with the usual interpretation), (5) as ⟨node⟩concept[level-or-levelRange]-expression (such that all nodes that satisfy the hierarchical expansion of the indicated concept, are at the indicated level(s), and beneath the indicated node are in the interpretation). Notice that the construct ⟨node⟩concept[level-or-levelRange]-expression does not enhance the expressiveness of MDO, but assists in structuring concept expressions in an OLAP setting along modeling elements of DWH dimensions: hierarchy of nodes, properties of nodes (via entities), and levels.

Example 13 (Dimensional concepts). Figure 5 shows concepts for DM2 specific drugs: InAD (to be read as "in antidiabetic drug group"), InOAD (to be read as "in oral antidiabetic drug group"), InStarterOAD (comprises OAD drugs that should be used first when diagnosis DM2 is determined). The concept expression of InOAD denotes that at level atcPharm ATC code A10B is selected and the asterisk on the right of the expression denotes hierarchical expansion, i.e., all subnodes below node A10B belong to concept InOAD. On the left hand side of Fig. 5 one can see the level hierarchy and on the right hand side a selection of the node hierarchy of the dimension. Items bordered by continuous lines (InAD, InOAD, and InStarterOAD) represent hierarchical concepts, which comprise nodes of multiple

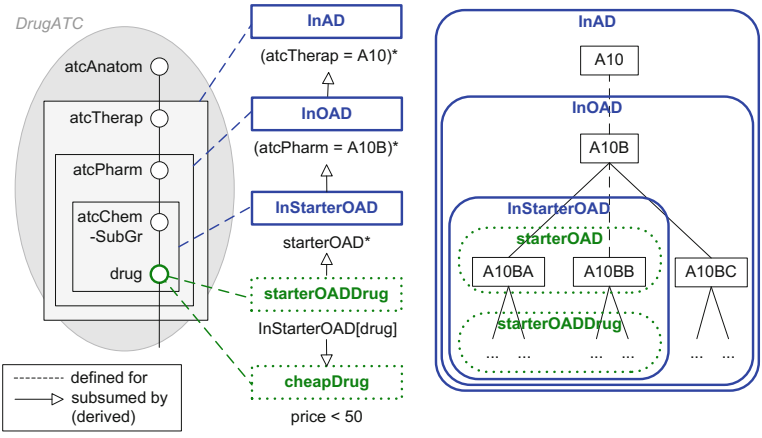


Fig. 5. Dimensional concepts over drugs

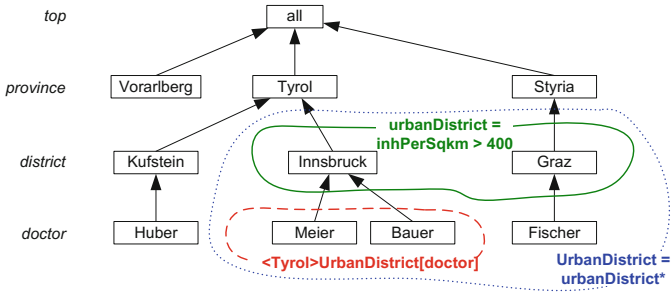


Fig. 6. ⟨point⟩concept[granularity]-expression

levels. Items bordered by dotted lines (starterOAD, starterOADDrug, and cheapOAD) denote flat concepts, which only comprise nodes of one level. Additionally derived subsumption relations are shown, e.g., cheap drugs subsume starter OAD drugs. Figure 6 illustrates the interpretation of a ⟨node⟩concept[level]-expression. Concept ⟨Tyrol⟩UrbanDistrict[doctor] comprises nodes of dimension Doctor at level doctor who live in an urban district in province Tyrol. Concept UrbanDistrict is the hierarchical expansion of concept urbanDistrict, comprising districts with more than 400 inhabitants per square km.

The membership condition of a multi-dimensional concept (md-concept) is given in one of the following ways: (1) by reference to a dimensional concept for some dimension role (in which case all points that satisfy the dimensional concept in the indicated dimension role are in the interpretation), (2) by hierarchy expansion of a md-concept (such that each point that is in the interpretation of the md-concept or a descendent thereof is in the interpretation), (3) by granularity restriction of some md-concept (such that only points of the indicated md-concept that are between a given top and bottom granularity are in the interpretation), (4) by intersection, union, or complement (open world interpretation) of md-concepts defined over the same dimension roles and the same granularity (usual interpretation), (5) as ⟨point⟩concept[granularity-or-granularityRange]-expression (such that all points that satisfy the hierarchial expansion of the indicated concept, are at the indicated granularity, and beneath the indicated point are in the interpretation), or (6) by a boolean expression over measure-value comparisons of measures applied to a point (*fact-based concept*). Further, each dimension space is also a md-concept.

Example 14 (Multi-dimensional concepts). The concept expression of multi-dimensional concept InsInRuralDistrLeadDocInUrbanDistr indicates that points that refer in the dimension role ins to urban districts and in the dimension role leadDoc to urban districts are interpretation as well as points that roll up to such points (Fig. 7). The expression uses concepts ins:ruralDistrict and leadDoc:urbanDistrict. Multi-dimensional concept InsInHighRuralDistrLeadDocInUrbanDistr is defined in a similar way. As discussed later, the subsumption relationship between both concepts can be detected by reasoning.

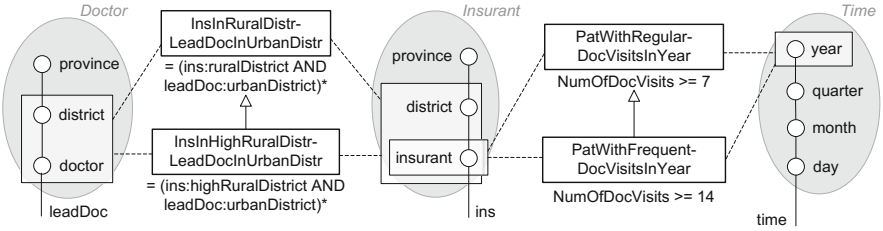


Fig. 7. Multi-dimensional concepts

Example 15 (Fact-based multi-dimensional concepts). PatWithRegularDocVisitsInYear is a fact-based multi-dimensional concept (Fig. 7). It comprises points of patients and years, for (patient, year)-pairs such that the patient had at least seven doctor visits in the year. The restriction to levels insurant and year is necessary because the concept as such is meaningful only for a single patient and a given year. The second fact-based multi-dimensional concept PatWithFrequentDocVisitsInYear comprises patients having at least fourteen doctor visits a year, a corresponding subsumption relationship will be inferred (as mentioned already before). The concept expressions assume that an aggregate measure NumOfDocVisits (number of doctor visits) over patients and years has been defined (Measure definition is described later).

The membership condition of a comparative concept is given like multi-dimensional concepts, with the addition that it may be also given by (a) two md-concepts, (b) a join condition relating nodes of the PoI and the PoC by a conjunction of pre-defined comparison predicates such as equality or predecessor/successor-relationships, and (c) by a boolean expression over score-value comparisons of scores applied to PoI and PoC.

3.2 Context-Specific and Contextualized Concepts

We present now a kind of specialization for concepts that is akin to specialization in object-oriented systems employing the abstract superclass rule [17].

In object orientation, the signature or head of a method may be introduced in an abstract superclass without providing an implementation of the method. The implementation (also referred to as body) of the method is provided by each concrete subclass of the abstract superclass. In an MDO, in analogy to object-orientation, a concept may be regarded as a boolean method of its domain (a concept may be applied on each node or point in its domain and returns either true or false) where the domain of the concept is analogous to the class in which the method is introduced and where the membership condition of the concept is analogous to the implementation of the boolean method. A contextualized concept is analogous to a boolean method introduced at some abstract superclass (with the domain of the contextualized concept playing the role of the abstract

superclass). Context-specific concepts are analogous to boolean methods at subclasses that implement the method introduced at the abstract superclass, with the context of a context-specific concept, which is given by an MDO concept expression, being analogous to the subclass at which the boolean method is implemented.

Context-specific concepts are defined over a selected subset of nodes of a dimension (dimensional concepts) or over a subset of points of a dimension space (md-concepts) by indicating a *context* in the signature. Thereby a *context* is given by ⟨node⟩concept[level-or-levelRange]-expression (for dimensional concepts) or a ⟨point⟩concept[granularity-or-GranularityRange]-expression (for md-concepts).

Contextualized concepts are defined by several context-specific concepts. The contexts of these concepts must cover all points in the signature of the contextualized concepts such that each point belongs to exactly one most specific context (i.e., the point does not belong also to another context subsumed by the former).

Example 16 (Contextualized concepts). Different to concept PatWithRegularDocVisitsInYear in example 15, we now consider regular visits of a patient to a particular doctor in a year. Regularity of visits depends on the medical section of a doctor. We assume, for simplicity, that there are only two medical sections, namely general practitioner and oculists. Suppose a regular patient of a general practitioner (GP) must have at least four GP visits per year whereas for a regular patient of an oculist it is sufficient to have at least two visit per year. Of course, one could define two separate concepts, but we specify one contextualized concept PatWithRegularVisitsToDocInYear([time:year,ins:insurant,actDoc:doctor]) consisting of two context-specific concept definitions: NumOfDocVisits ≥ 4 for context ⟨time:all, ins:all, actDoc:GP⟩[time:year, ins:insurant, actDoc:doctor] and NumOfDocVisits ≥ 2 for context ⟨time:all, ins:all, actDoc:oculist⟩[time:year, ins:insurant, actDoc:doctor]

3.3 Semantic Dimensions

Transaction systems collect records that refer to concepts of domain ontologies in semantic attributes. For example, medical treatment records may refer to a diagnosis code of SNOMED CT. In order to exploit semantic attributes for OLAP-style analysis, we wish to use the existing domain ontology to which semantic attributes refer like a common dimension in data warehousing and call a dimension based on a domain ontology in analogy to semantic attributes, a semantic dimension. A semantic dimension will be usually expressed or mapped to an ontology language such as OWL.

Querying over semantic attributes in relational databases is discussed by Das et al. [8] and implemented in *Oracle Database 11g Semantic Technologies*. In addition to their approach, the semCockpit approach also allows to use concept expressions as selection criteria (post-coordination) and provide for a seamless integration in data warehousing and OLAP. The challenges tackled by this approach are akin to the challenges of heterogeneous dimensions [18,21,27].

Malinowski and Zimányi [23] give an overview of different kinds of dimension hierarchies.

We briefly explain how domain ontologies can be used as semantic dimensions, and refer to [1] for a more elaborate treatment: (1) The existing concepts (usually called pre-coordinated concepts) are mapped to nodes of a dimension hierarchy. (2) Facts may refer to leaf or inner nodes. The meaning of a fact refering to an inner node "c" is "c only", e.g., "DM-2 without further information on the subkind of DM-2". The latter concept ("c only") is not represented in the original domain ontology, but would be a leaf node and a child of the former ("c"). (3) New concepts (usually called post-coordinated concepts) may be defined upon existing ones in the external domain ontology language (e.g., by OWL expressions) and are mapped to an MDO concept. An MDO concept C corresponding to concept c in the external domain ontology comprises all nodes that correspond to a concept subsumed by c. (4) While the domain ontology is un-leveled, levels may be introduced explicitly by identifying the concepts (nodes) that make up the members of a level. To provide for summarizability (i.e., ensuring that the sum over all base facts of an additive measure is the same than the sum of the aggregated measure over all roll-up facts at some granularity), the member concepts (nodes) of such a level must be disjoint (i.e., have non-overlapping interpretations) and be complete with respect to the bottom level (i.e., each leaf node must be a member of some member concept). (5) Levels may be defined context-specific, i.e., local to a node. (6) Built that way, precoordinated concepts can be used like native nodes and post-coordinated concepts like MDO concepts over native dimensions for OLAP operations slice, dice, and roll-up.

The treatment of native and semantic dimension becomes seamless by unifying native dimensions and semantic dimensions. A node of a dimension may relate to either an entity (entity node) or to an entity concept (concept node), which may be given by an entity concept as introduced above or by a domain ontology concept. Levels consist either of entity nodes (entity levels) or concept nodes (concept levels), which may be introduced above an entity level.

Example 17 (Semantic Dimension – SNOMED CT). Figure 8 shows a small part of the SNOMED CT hierarchy. It can be taken to implement a semantic dimension for disease which can be linked to fact classes (e.g., drugPrescription and ambTreatment). In Fig. 8 we have already extended the SNOMED CT concepts with "only"-concepts, i.e., each inner node has as an "only"-node as a subconcept. E.g., the node Diabetes mellitus has the additional subnode Diabetes mellitus only, which does not exist in the original SNOMED CT hierarchy. A fact can refer all leaf nodes presented in our diagram, i.e., all original leaf nodes and all "only"-leaf nodes (original inner nodes). The figure also illustrates how conventional OLAP operations can be applied for semantic dimensions. The nodes bordered by the continuous line represent the result of a DICE operation that selects the subhierarchy under node Diabetes mellitus (analogously to conventional OLAP operation DICE which selects a subcube). Nodes bordered by the coarse dotted line represent the result of a SLICE operation with condition "all nodes

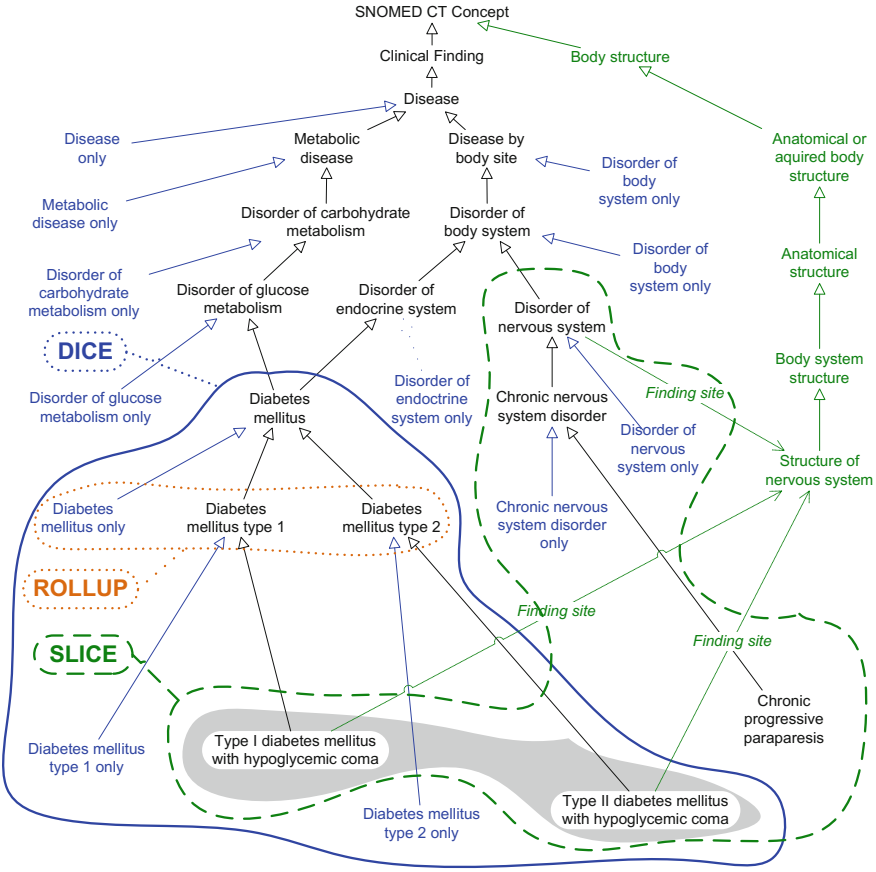


Fig. 8. Semantic Dimension (SNOMED CT)

for which there exists finding site equal to Structure of nervous system". Applied to DICE node Diabetes mellitus the SLICE operation returns the grey filled area that comprises the nodes Type I diabetes mellitus with hypoglycemic coma and Type II diabetes mellitus with hypoglycemic coma. Finally one can ROLLUP to nodes Diabetes mellitus only, Diabetes mellitus type 1, and Diabetes mellitus type 2 (bordered by the fine dotted line) that represent a virtual level local to Diabetes mellitus.

3.4 Relational and OWL Representations

MDO concepts are translated to SQL for querying the underlying closed world data warehouse. MDO concepts are translated to OWL in order to delegate subsumption checking to an off-the-shelf OWL reasoner. The mappings for each kind of concept (apart from comparative concepts) are given in [28].

The relational representation of MDO concepts builds on the relational representation of the semCockpit data warehouse where entity classes, dimensions

and dimension spaces are directly available through SQL tables and views. Entity concepts, dimensional concepts, and multi-dimensional concepts are translated to views over this relational representations of entity classes, dimensions, and dimension spaces as well as over views generated for previously defined concepts.

We now shortly explain the rationale of the representation of MDO concepts in OWL and refer to [28] for a more elaborate treatment. We assume a basic familiarity with OWL [16]. Using the OWL representation, the initially unordered set of business terms represented by the MDO may be automatically organized in subsumption hierarchies.

Example 18 (Concept organization). Figure 9 illustrates for a selection of our use case, how business terms are organized in subsumption hierarchies along the dimensions of a data warehouse.

Individuals of the ontology are MDO entities, nodes, levels, points, and point-pairs. MDO entities are collected into disjoint OWL classes, one OWL class for each MDO entity class. Attributes of MDO entities are represented in OWL as object properties of MDO entities. Nodes are collected into disjoint OWL classes, one OWL class for each dimension. Each node is associated with one level and with one MDO entity, this is represented by object properties atLevel and roleOf, respectively. Roll-up relationships between nodes as well as between levels are represented as transitive and reflexive property rollsUpTo. Dimension roles are represented as object properties of points, where the range of a dimension role is a dimension. Dimension roles of point-pairs are distinguished into PoI- and PoC-dimension roles. Also, an OWL class is defined for each dimension space and each comparison space. Notice that MDO entities and nodes not explicitly referred to in MDO concept expressions as well as points and point-pairs are not represented as named individuals in the OWL ontology.

MDO concepts are translated to OWL classes according to the MDO concept expression. To capture level-restrictions the level range is checked by a property restriction on property atLevel indicating that node's levels must drill-down to the "from-level" and roll-up to the "to-level".

One of the challenges of representing MDO concepts in OWL is to cope with the following restriction of OWL 2 DL[4]: It is not allowed to express that a transitive relationship such as rollsUpTo maps to exactly one object in a given range (e.g., to one node of one level). But, the essential characteristics of roll-up hierarchies of data warehouse dimensions are that (i) the rollsUpTo-relationship between nodes is transitive, and (ii) each node of a level rolls up to exactly one node of a higher level (to which the former level rolls up). Without this semantics of roll-up hierarchies in data warehousing being captured, OWL reasoners will not be able to recognize that certain concepts are disjoint. To cope with this limitation of OWL, for each level X there is a functional object property rollsUpTo_X. For each named node nd at level X there is a subclass axiom stating

[4] http://www.w3.org/TR/owl2-syntax/#Global_Restrictions_on_Axioms_in_OWL_2_DL

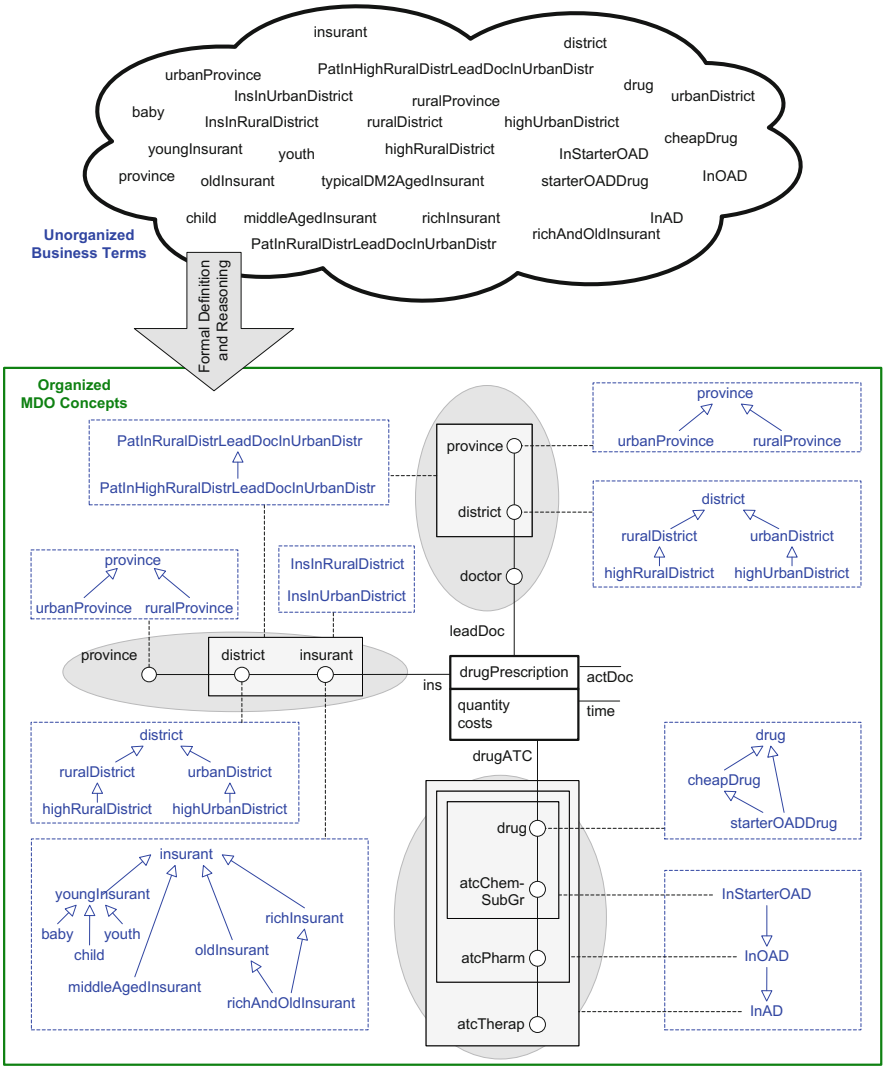


Fig. 9. Concept organization

that every descendant node of *nd* rolls up to *nd* via functional object property rollsUpTo_X.

Example 19 (Translation of MDO concepts to OWL). Multi-dimensional concept InsInRuralDistrLeadDocDoctorInUrbanDistr[ins: insurant..district, leadDoc: doctor] is interpreted by the set of points that each refer via dimension role ins to a node that rolls up to a node which is a role of a rural district and refer via dimension role leadDoc to a node at level doctor that rolls up to a node which is

a role of an urban district. In OWL this is represented (using Description Logics notation) as:

InsInRuralDistrLeadDocDoctorInUrbanDistr ≡
  ∃ins.∃rollsUpTo_district.∃roleOf.ruralDistrict ⊓
  ∃leadDoc.(∃atLevel.{doctor} ⊓∃rollsUpTo_district.∃roleOf.urbanDistrict)

4 Ontology-Based Measures and Scores

A *measure* is defined for points in a dimension space, which is also called the domain of the measure. Measures are distinguished into base and derived. Base measures are given as primitive and relate to a measure of a fact class (Note: To provide for parallel analysis across multiple roll-up hierarchies of a dimension role, several base measures may be defined for a measure of a fact class, each of them defined for a different dimensions space that replaces selected dimension roles of the dimension space of the fact class by one or several hierarchy-specific dimension roles). Derived measures have measurement instructions that describes how a measure value is calculated for each point in the measure domain from other measures.

A *score* is defined for pairs of points (PoI, PoC) in a comparison space. The scoring instruction of a score describes for each pair of points (PoI, PoC) in the score domain how a score value is calculated from measures.

A measure may be *applied* to a point for which it is defined, returning the measure value; a measure applied to a set of points gives a set of measure values. A measure may be applied to a point with more dimension roles for which the measure is defined; in such a case the superfluous dimension roles are ignored. We denote measure application by the '.'-operator.

Example 20 (Measure application). The following measure application returns the drug costs of patients in Upper Austria in the year 2012:

⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩.DrugCosts

Alike measure applications a score may be *applied* to a pair of points for which it is defined, returning the score value; a score applied to a set of pairs of points gives a set of scores. If a score is applied to a pair of points with more dimension roles as for which the score is defined, the superfluous dimension roles are ignored. As for measures we denote score application by the '.'-operator.

Example 21 (Score application). The following score application returns the ratio of drug costs of patients in Upper Austria in year 2012 (as point of interest) to the drug costs of patients in Upper Austria in 2011 (as point of comparison):

(⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩,
 ⟨time:2011, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩)
      .RatioOfDrugCosts

A *fact* is a point of a dimension space together with values for one or several measures; a *comparative fact* is a pair of points together with values for one or several scores.

A *cube* (*comparative cube*) is a set of facts (comparative facts), possibly at different granularities. Special kinds of cubes are (a) fact classes, which are mono-granular and primitive (i.e., its facts are not calculated from other facts in the DWH), (b) measure cubes and score cubes, which are cubes over the domain of a measure or score and whose facts possess only this measure or score, and (c) cuboids, which are mono-granular slices of another cube. Cubes - other than fact classes and measure cubes - are defined by a *cube space*, given by a md-concept, preferably in the form ⟨point⟩concept[granularity-or-granularity-range], and a set of measures that may be applied to points in the cube space to construct facts. This is indicated by applying a measure with the ".."-operator to the cube space and optionally indicating after the measure by the "\"-operator whether a null-value of the measure should be replaced by some other value.

Example 22 (Cube). A cube consisting of drug-costs facts for points at granularity [time:month, ins:district, leadDoc:district, actDoc:top, drugATC:top] that satisfy concept InsInRuralDistrLeadDocInUrbanDistr and roll-up to point ⟨time:2012, ins:Austria, leadDoc:Austria, actDoc:all, drugATC:all⟩, is defined by:

⟨time:2012, ins:Austria, leadDoc:Austria, actDoc:all, drugATC:all⟩
InsInRuralDistrLeadDocInUrbanDistr
[time:month, ins:district, leadDoc:district, actDoc:top, drugATC:top]
..DrugCosts

Likewise, a comparative cube is defined for a comparative cube space, given by a comparative concept, and a set of scores defined for the comparative cube space.

Example 23 (Comparative cube). The following comparative cube lists for each rural district of insurants in Upper Austria the ratio of drug costs in year 2012 to drug costs in year 2011, thereby comparison predicate SameDistrict is used to relate facts where PoI and PoC of the fact refer to the same district in the insurant dimension role.

(⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩
ins:ruralDistrict[time:top, ins:district, leadDoc:top, actDoc:top, drugATC:top]
SameDistrict(PoI.ins,PoC.ins)
⟨time:2011, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩
ins:ruralDistrict[time:top, ins:district, leadDoc:top, actDoc:top, drugATC:top])
..RatioOfDrugCosts

Measure and score instructions are arithmetic or aggregation expressions over measures of selected facts of measure cubes. The scoring instructions are either given in native form by using artihmetic and aggregation operations with MDO-query expressions (using '.' and '..'-operators as described above) as operands, or by SQL-built-ins. The measurement instruction of an *arithmetic measure* is

defined natively in MDO in an object-oriented flavor by an arithmetic expression over measures applied to a point self in the domain of the measure for which the measure value is calculated. The measurement instruction of an *aggregation measure* for point self is given in MDO by some kind of aggregation (such as AVG or SUM) over measure values of selected facts of some cube (which is a fact class for first-step aggregation measures). As the selection of facts is based on using concepts of the MDO as *qualifiers* to selecting facts (using ⟨self⟩(concept)-expressions for fact classes and ⟨self⟩(concept)[granularity]-expressions for other cubes; note: self may be omitted), we speak of ontology-based measures and scores.

Example 24 (Measure with one step aggregation). DrugCosts is a derived measure of type float over dimension space DrugPrescriptionSpace

```
CREATE MEASURE DrugCosts
DATATYPE float FOR DrugPrescriptionSpace AS
   SUM(⟨self⟩..drugPrescription.costs);
```

The measurement instruction indicates that the measure value is calculated for a point self by the sum over base measure costs of all facts in factclass drugPrescription that roll-up to self. If one restricts the dimension space DrugPrescriptionSpace to oral antidiabetic drugs (InOAD), one can define another measure that returns overall costs for oral antidiabetic drugs:

```
CREATE MEASURE OADDrugCosts
DATATYPE float FOR DrugPrescriptionSpace AS
   SUM(⟨self⟩InOAD..drugPrescription.costs);
```

Example 25 (Measure with two step aggregation). AvgDrugCostsPerIns is defined as a measure with two step aggregation that returns the average drug costs per insurant:

```
CREATE MEASURE AvgDrugCostsPerIns
DATATYPE float FOR DrugPrescriptionSpace AS
   AVG( ⟨self⟩[time:top, ins:insurant, leadDoc:top, actDoc:top, drugATC:top]
            ..DrugCosts );
```

The measure calculates, first, the total costs per insurant stated as cube ⟨self⟩ [time:top, ins:insurant, leadDoc:top, actDoc:top, drugATC:top]..DrugCosts (first aggregation step, cf. Ex. 24) and, second, the average of drug costs per insurant (second aggregation step).

Example 26 (Drill across measure). TotalCosts are computed by adding DrugCosts, which has been defined over dimension space DrugPrescriptionSpace, and AmbTreatmentCosts, which has been defined over dimension space AmbTreatmentSpace:

```
CREATE MEASURE TotalCosts
DATATYPE float FOR MedcareSpace AS
      ⟨self⟩.DrugCosts\0 + ⟨self⟩.AmbTreatmentCosts\0;
```

This is an example where two fact classes (drugPrescription and ambTreatment) are used, one providing measure DrugCosts and the other providing measure AmbTreatmentCosts. The dimension space MedcareSpace is defined as drill across space of DrugPrescriptionSpace and AmbTreatmentSpace (not shown). The decoration \0 indicates that the default value 0 is to be used if a measure application returns a null value.

Measurement and scoring instructions frequently have the same structure (or pattern). To avoid the need to define measurement and scoring instructions of similar kind, semCockpit provides a set of predefined measure and score templates, which can be extended. For example, the *first-step aggregation template* defines an aggregate measure based on the template parameters base measure m, slice-concept c, and aggregation function f, with underlying measurement instruction $f\langle\text{self}\rangle(c).m$ The *higher-step aggregation template* defines an aggregate measure based on: roll-up granularity g, derived measure m, and aggregation function f with underling measurement instruction $f\langle\text{self}\rangle[g].m$.

Scores are distinguished into arithmetic and analytic. Arithmetic scores relate two measures of two points (PoI and PoC) of a cube by an arithmetic function such as ratio or percentage difference.

Example 27 (Ratio score). Score RatioOfDrugCosts returns a ratio of drug costs:

```
CREATE SCORE RatioOfDrugCosts
DATATYPE float FOR (DrugPrescriptionSpace, DrugPrescriptionSpace) AS
    RATIO( ⟨PoI⟩.DrugCosts, ⟨PoC⟩.DrugCosts );
```

The score is defined for comparison dimension space (DrugPrescriptionSpace, DrugPrescriptionSpace). The keyword RATIO is used to indicate that drug costs of the group of interest represented as first parameter (PoI) are to be divided by the drug costs of the group of comparison denoted as second parameter (PoC).

Analytic scores use an analytic scoring function (such as average-percentile rank or mean-percentile rank) on two sets of points, GoI and GoC, each identified by a ⟨pnt⟩(concept)[granularity]-expression.

Example 28 (Median percentile rank score). Score MPROfDrugCostsPerPatient has median percentile rank as a scoring function. The score can be applied in the time dimension roles on month or higher levels, and in dimension roles ins, leadDoc, and actDoc from level district to top.

```
CREATE SCORE MPROfDrugCostsPerPatient
DATATYPE float FOR (DrugPrescriptionSpace, DrugPrescriptionSpace)
AT ([time:month..top, ins:district..top, leadDoc:district..top,
        actDoc:district..top, drugATC:drug..top],
    [time:month..top, ins:district..top, leadDoc:district..top,
        actDoc:district..top, drugATC:drug..top]) AS
  MEDIAN_PERCENTILE_RANK(
      ⟨PoI⟩[time:top, ins:insurant, leadDoc:top, actDoc:top, drugATC:top]
          ..DrugCosts,
      ⟨PoC⟩[time:top, ins:insurant, leadDoc:top, actDoc:top, drugATC:top]
          ..DrugCosts );
```

Drug costs are computed per insurant for group of interest as well as group of comparison. Based on both groupings the median percentile rank is calculated.

Generic measures and *generic scores* avoid the need of repeated definition of measure and score instructions with different selection criteria (concepts) in place. They provide for flexibility in measure and score use, and they enable reasoning (about how measures relate) by providing common structures. Generic measures and scores have generic parameters (denoted by %name) for concepts, which we call *qualifiers* as they are used to qualify selection-expressions for facts of some cube. The domain of a generic parameter may be restricted to a listed set of concepts or to the set of concepts subsumed by a concept.

Example 29 (Generic measures and scores). We generalize the definition of measure DrugCosts of Example 24 and add a generic parameter %q. It is restricted to multi-dimensional concepts which are subsumed by DrugPrescriptionSpace (denoted by ↑):

```
CREATE MEASURE DrugCosts( %q ↑ DrugPrescriptionSpace )
DATATYPE float FOR DrugPrescriptionSpace AS
   SUM(⟨self⟩(%q)..costs);
```

Analogously one can define generic scores like RatioOfDrugCosts:

```
CREATE SCORE RatioOfDrugCosts( %qoi ↑ DrugPrescriptionSpace,
                               %qoc ↑ DrugPrescriptionSpace )
DATATYPE float FOR (DrugPrescriptionSpace, DrugPrescriptionSpace) AS
      RATIO( ⟨PoI⟩.DrugCosts(%qoi), ⟨PoC⟩.DrugCosts(%qoc) );
```

Generic parameters can be used in place of concepts of concept expressions (e.g., oldPatient AND %q) in measurement or scoring instructions. A generic measure (score) is instantiated by binding the generic parameter to a concept, giving a non-generic measure (score) by replacing the generic qualifiers accordingly in measurement (scoring) instructions.

Example 30 (Use of generic measures and scores). Instantiation of generic measure DrugCosts with actual qualifier InOAD gives the previously defined measure OADDrugCosts (Ex. 24):

```
⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩
   .DrugCosts( InOAD )
```

The following instantiation of generic score RatioOfDrugCosts binds %qoi to InStarterOAD and %qoc to InOAD. It returns the ratio of starter oral antidiabetic drug costs to oral antidiabetic drug costs:

```
(⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩,
⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩)
   .RatioOfDrugCosts(InStarterOAD, InOAD)
```

Notice that in this special case of comparative data analysis point of interest and point of comparison are equal and the only difference is in the qualifications of the score.

In order to provide for the definition of measures and scores which cannot be directly expressed in the semCockpit language, *built-in measures and scores* provide for an ad-hoc extension facility and for the possibility to define new templates for defining measure or scores. A built-in measure or score is defined by providing an SQL view that defines the function from point to measure value (from pair of point to score value, respectively). This approach is not novel, but common in Oracle where new cubes and measures can be derived from multi-granular cubes of other measures. However, different to Oracle, all MDO-concepts are available as SQL views as well and may be used in measure- and score definitions. Thereby, rather than writing complex selection query predicates for selecting tuples of some cube, these can be easily selected by a simple (natural) join between the SQL view of the concept and the cube. This provides for a simple, natural definition of the measurement or scoring instructions. Platform-dependent optimization (in consideration of the capabilities and limitations of query optimizers) is a separate issue. Built-in generic measures and generic scores are provided as macros with qualifiers (view names) as parameters.

Measures and scores are organized in a measure & score drivers hierarchy. The change of value of a measure may change the value of another measure or score, the change of value of a score may change the value of another score. The influence hierarchy known from definition of measures and scores or from background knowledge (e.g., relationships of base measures) is captured explicitly and used later as background knowledge to guide analysis processes.

5 Ontology-Based Comparative OLAP

In online analytical processing, OLAP operations slice, dice, drill-down and roll-up are applied to a data cube in order to navigate from one to another cuboid of the data cube. We sketch a possible extension of semCockpit for modeling and representing such analysis steps.

In the context of ontology-based comparative data analysis and in the presence of generic scores, a particular comparative cuboid, which we call *comparative analysis situation*, is described in the form ⟨pntGoI⟩conceptGoI[granGoI]joinCond ⟨pntGoC⟩conceptGoC [granGoC]..score(qGoI,qGoC) with variables for nodes of points (pntGoI, pntGoC), levels of granularities (granGoI, granGoC), concepts (conceptGoI, conceptGoC, qGoI, qGoC), join condition (joinCond), and score. For simplicity, we consider here only scores with two qualifiers, one for the group of interest and one for the group of comparison.

The description of a non-comparative analysis situation is based on cube definition of the form ⟨point⟩concept [granularity]..measure (qualifier$_1$, ..., qualifier$_n$).

In the remainder, we speak for simplicity of analysis situations, if we refer to comparative and non-comparative analysis situations.

An OLAP-operation between two analysis situations, which we refer to as *navigation*, reflects the change of the bindings of the variables of the target analysis situation with respect to the source analysis situation. An atomic navigation changes the binding of a single variable. Atomic navigation can be classified in

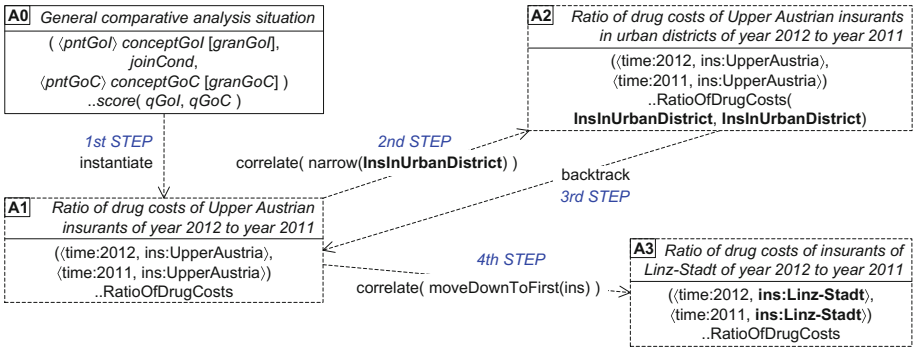


Fig. 10. Ontologoy-based comparative OLAP

an OLAP-setting according to the kind of OLAP-step performed. Such a *navigation step* indicates a movement along some "semantic relationship" between two cubes (such as drill-down one level in hierarchy x, move up to ancestor node in hierarchy x) and is expressed by a *navigation operator* and possibly a *navigation variable*. A variable may be for a node, a level, a concept, a join condition, or a score. The binding may be indicated absolute (i.e., by a new value) or relative to the binding of a source variable (e.g., drill-down one level from current level in hierarchy x).

Example 31 (Ontology-based comparative OLAP). Figure 10 illustrates[5] how a business analyst applies ontology-based comparative OLAP operators. First, A0 shows the general description of an analysis situation. She or he selects the points ⟨time:2012, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩ and ⟨time:2011, ins:UpperAustria, leadDoc:all, actDoc:all, drugATC:all⟩ to compare drug costs of Upper Austria in year 2012 with 2011 by calculating the ratio (resulting in analysis situation A1). Next, the analyst considers that it is of interest to restrict the comparison to patients of urban districts. He or she applies the navigation operator correlate(narrow) with actual parameter InsInUrbanDistrict which narrows the group of interest as well as the group of comparison to urban districts. The navigation results in analysis situation A2 in which the generic score RatioOfDrugCosts is qualified in the GoI and in the GoC by hierarchical concept InsInUrbanDistrict. Afterwards the analyst backtracks to A1 and applies navigation operator correlate(moveDownToFirst) that results in moving down to the first district of Upper Austria in GoI and GoC (analysis situation A3). We assume a descending order by number of inhabitants, thus the user navigates to district Linz-Stadt.

A *composite analysis situation* is a tree of analysis situations and navigation steps. Composite analysis situations provide a global coherent picture of several

[5] In this and subsequent figures we omit for brevity the representation of "all"-nodes of points in dimension space DrugPrescription.

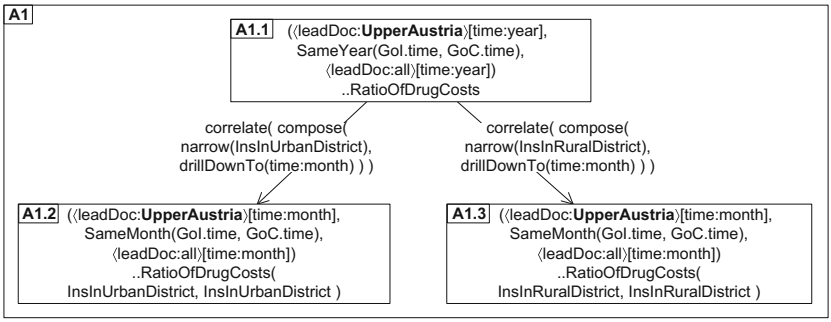


Fig. 11. Composite analysis situation

related measures and scores, aggregated and detailed. Changing the variables of the root analysis situation leads to coherent change of all dependent analysis situations. A generic composite analysis situation with all variables unconstraint and modifiable provides for a general, comprehensive multi-perspective browsing facility similar as it is provided by "surf and save" BI tools like Tableau[6], but with enhanced flexibility and guidance support (as will be discussed in the subsequent section). This mode of use is typical during phases of explorative search for meaningful scores and comparison groups, as well as for developing analysis processes (which are thereafter captured as more elaborated BI analysis graphs).

Example 32 (Composite analysis situation). By a composite analysis situation one can synchronize various semantically related analysis situations. In Fig. 11 root situation A1.1 selects drug costs ratios of lead doctors of province Upper Austria per year[7]. Correspondingly, A1.2 and A1.3 select the drug costs ratios of urban and rural districts, respectively, of Upper Austria per month. Two navigation operations denote these semantic relations. If the user changes lead doctor location from Upper to Lower Austria in A1.1, also the situations A1.2 and A1.3 are adapted automatically to province Lower Austria.

One can capture the history of a particular analysis performed, in the form of a graph, consisting of the analysis situations and navigation steps used to navigate between them.

Example 33 (History of an analysis). The essence of a history of an analysis can be depicted as a graph. Figure 12 presents the static view of the dynamic process of example 31. There are navigation arcs from analysis situation A1 to analysis situation A2 and A3. The graph only shows the semantic dependencies. It omits the dynamic behaviour like the order in which analysis situations were performed, or backtracking paths.

[6] http://www.tableausoftware.com

[7] In this and subsequent figures we omit for brevity the representation of "top"-levels of a granularity in dimension space DrugPrescriptionSpace.

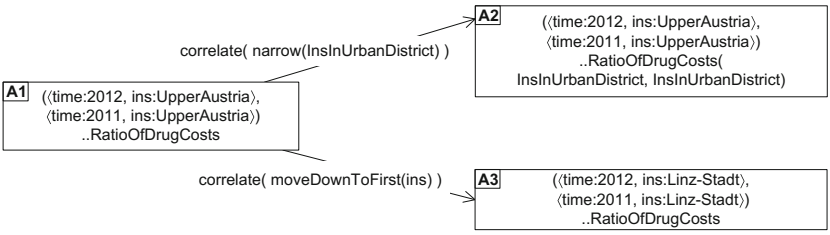


Fig. 12. History of an analysis (without backtracking steps)

6 BI Analysis Graphs

A typical BI analysis session in comparative data analysis is described by the following process: (1) Determine an initial analysis situation (by setting parameters of an OLAP query generating a cube or comparative cube, usually monogranular). (2) Visually inspect the result. (3) Modify parameters (usually based on semantic relationships) to move to a new analysis situation. (4) Inspect the result: (a) stop if satisfied, (b) continue with (3), or (c) backtrack to a previous analysis situation.

We present a vision of BI analysis graphs to capture this "analysis process knowledge" at the schema level for later analysis as reference. BI analysis graphs may be compared to process schemas designed in BPMN [37] which reflect how a particular kind of business process is handled, or to the navigation model of WebML [7] in web engineering, which describes how data and their relationships may be traversed.

BI analysis graphs are inspired by WebML: The navigation model of WebML is a graph of units and links. A unit represents objects or set of objects that are retrieved by a parameterized SQL query associated with the unit. Links describe how objects of source and target units relate. Changing the parameters of a source unit and, thus, the object(s) represented, is propagated to the target unit by information transported along the link (which binds parameters of source to parameters of target units), leading to related changes of the objects represented in the target unit. Such changes may be automatic or dependent on user input. Similarly, in BI analysis graphs, analysis situations are parameterized cube definitions (or MDO queries), and navigation steps between analysis situations bind parameters of the target based on parameters of the source and, optionally, dependent on user input. Different to BPMN and WebML, which come with a clear distinction between schema and instance, BI analysis graphs adhere to a Frame-inspired [9] approach in which generic and individual analysis situations co-exist in one graph. This reflects the very nature of BI analysis in which the BI analysis graph should on the one hand generalize from individual analysis, but on the other hand is never complete and as such is continually extended and refined based on known analysis process knowledge acquired in subsequent analysis.

Further related work concerns navigation modeling and query prediction [35], describing analytical sessions [33], modeling OLAP behavior [44], modeling preferences [13], personalization [2], query recommendations [3,11,19], and annotations [10]. Heer et al. [14] focus on recording and visualizing interaction histories and propose a taxonomy of interactive dynamics for visual analysis [15]. Thollot [43] propose a graph-based approach for context-aware BI recommendations. Unlike [14,15,43], BI analysis graphs are motivated for designing and re-using general (non-personalized) analysis processes, and for modeling navigation knowledge as semantic relationships (*'Navigation is Knowledge'*). Recording the history of an analysis process is a side product of the BI analysis graph approach. BI analysis graphs and the associated envisioned guidance rules (discussed in Sect. 7) draw ideas from active data warehouses [42] and from OLAP querying at a conceptual level [30]. A simple form of BI analysis graphs [26] has been introduced for multi-dimensional navigation modeling.

We now give an overview of BI analysis graphs. A BI analysis graph is a directed graph with generic or individual comparative analysis situations as vertices and generic navigation steps as directed edges (we will extend this definition later by specialization and instantiation edges).

A *generic comparative analysis situation* is a comparative analysis situation in which some or all variables are unbound and in which unbound variables are restricted by domain indications. The domain of a variable is given by a dimensional concept for node variables of points (and, optionally, by a multi-dimensional concept for a point or a comparative concept for the point pair), a set of concepts for concept variables (expressed by enumeration or as $\uparrow c$ for the set consisting of concept c and all subsumed concepts), a set of levels for level variables, a set of join conditions for join-condition variables, and a set of scores for score variables (where $\uparrow s$ denotes score s and any score that directly or indirectly drives s in the score drivers hierarchy). If a domain is not explicitly indicated for a variable, the domain is any value possible for the variable's kind. A variable that is bound to a value or has only one permissive value is said to be *closed*, otherwise it is said to be *open*. The notion of a *generic non-comparative analysis situation* is analogously defined.

Example 34 (Generic analysis situation). The top left corner of Fig. 13 shows a generic comparative analysis situation A0 where all variables for nodes of points (pntGoI, pntGoC), concepts of external slice conditions (conceptGoI, conceptGoC), levels of granularities (granGoI, granGoC), join condition (joinCond), score (score), and qualifiers (qGoI, qGoC) are open.

An *individual (comparative or non-comparative) analysis situation* is an analysis situation in which variables are bound. In a short hand notation, variables may not be used. In such a case, they are bound to default values (e.g., a missing granularity is bound to the granularity of the point). It is an *instance of* any generic (comparative or non-comparative, resp.) analysis situation for which all variable values are from the respective domains defined by the generic analysis situation.

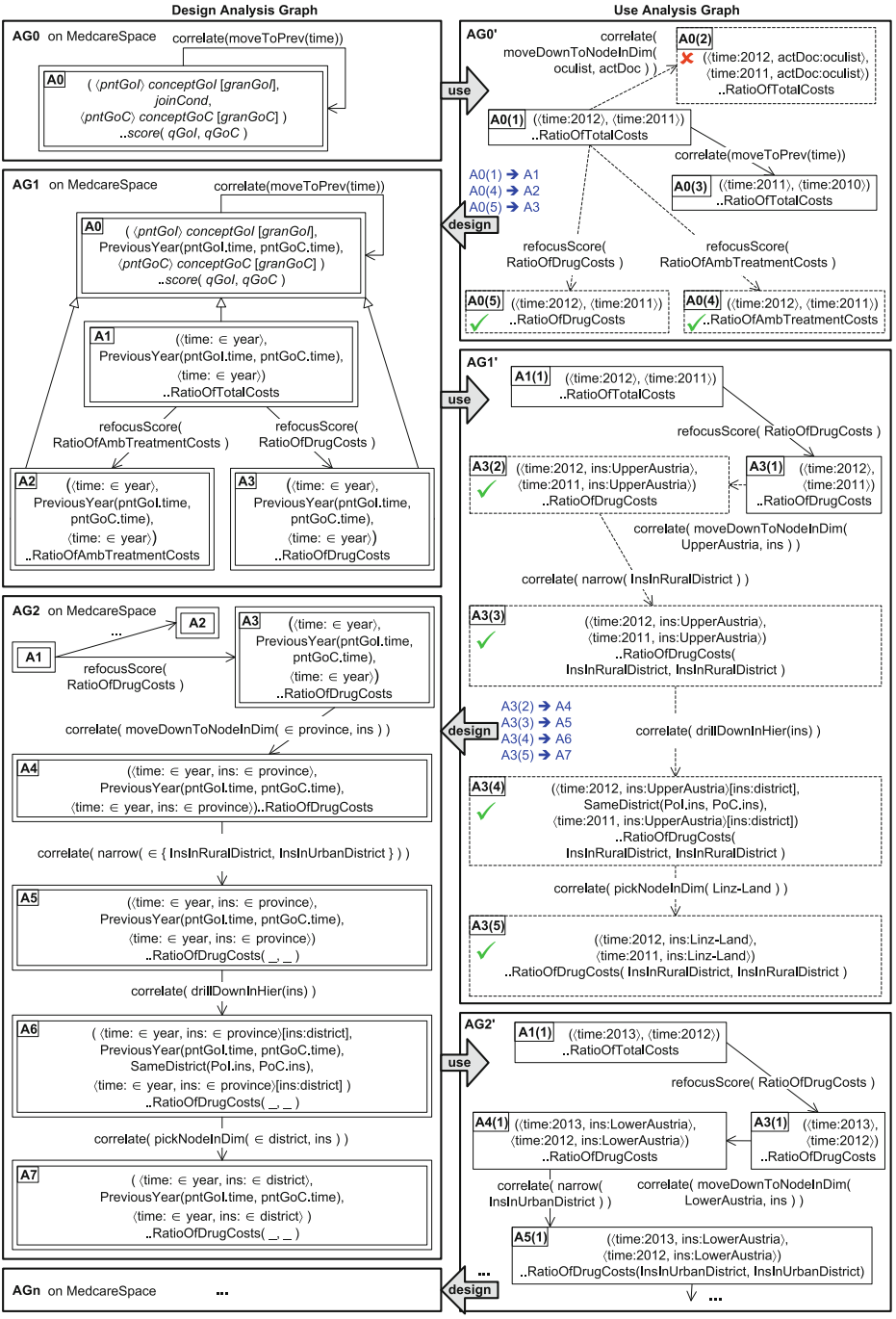


Fig. 13. BI Analysis Graph with use & design steps

In the remainder, we speak for simplicity of analysis situations, if we refer to comparative and non-comparative analysis situations, generic or individual.

Example 35 (Individual analysis situation). In Fig. 13 individual analysis situations are depicted on the right hand side. A business analyst uses the generic situation A0 and instantiates, e.g., the individual analysis situation A0(1).

A *generic navigation step* is defined by a navigation operator and, if it has a navigation variable, a domain for the navigation variable. An *individual navigation step* is given by applying the generic navigation step to an individual analysis situation and by binding any navigation variable of the navigation operator. The application yields an individual target analysis situations that is identical to the individual source analysis situation apart from the changes induced by the navigation operator.

Navigation operators express (a) movements in the nodes of a dimension hierarchy (like moveDownToNodeInDimension, moveUpInHierarchy, moveToPrev(ious sibling)), (b) changes of the granularity of a cube (like drillDownToLevel, drillDownInHierarchy, (c) strengthening (narrow), weakening (broaden), or resetting (qualifyAside) a qualifier of a score, (d) strengthening (filter), weakening (extend), or resetting (shift) the slice condition (i.e., conceptGoI or conceptGoC), or (e) the change of a score (refocus). Navigation operators may either change the group of interest (rerelate), the group of comparison (retarget), or both (correlate). Several navigation operators may also be composed into one (compose)). We omit a complete list for brevity.

Example 36 (Navigation step). Figure 13 shows a general navigation operation correlate(moveToPrev(time)) from A0 to itself. The instantiation of A0 to A0(1) and the application of the navigation step leads to A0(3).

We now revise the definition of BI analysis graphs to include modeled specialization relationships. A *BI analysis graph* is a directed graph whose vertices are analysis situations and whose directed edges are either navigation steps or modeled specialization relationships between analysis situations. Navigation steps may be specialized, too, in that a generic navigation step with a more restricted variable domain connects more specific analysis situations than the specialized navigation step. The specialization hierarchy is not inferred, but explicitly modeled similar to the conceptual modeling of business processes. It acts as constraint for the definition of analysis situations and allows to capture alternative navigation paths that apply to different specializations of a generic analysis situation. An analysis situation that *specializes* another analysis situation has for each variable the same or a more restrictive domain.

Navigation steps may be proper OLAP steps as introduced above or backtracking-steps. A backtracking step indicates that in a particular analysis the analyst moves back to a previously encountered analysis situation but chooses thereafter a different analysis situation to continue.

Example 37 (BI analysis graph). The left side of Fig. 13 shows analysis graphs AG0, AG1, and AG2. Each graph consists of vertices (analysis situations) and

directed edges (navigation steps). Backtracking-steps are not shown. Analysis situations A1, A2, and A3 are specializations of A0. Thus they are linked by inheritance arrows. Figure 13 is explained in more detail later.

A *BI analysis* is an alternate sequence of individual analysis situations and navigation steps that represent a sequential trace of the analysis steps performed by an analyst in a particular analysis. A particular BI analysis can be carried out by traversing an analysis graph and, if necessary, by extending the traversal.

A schema traversal $T = (A_1, S_1, \ldots, S_{k-1}, A_k)$ of a BI analysis graph G is an alternate sequence of analysis situations and navigation steps where for each $i = 1..k-1$ either (a) there exists an analysis situation A'_i such that S_i is an arc in G from A'_i to A_{i+1}, whereby $A'_i = A_i$, or A'_i is directly or indirectly connected to A_i by modelled specialisation relationships (or vice versa), or (b) S_i is a backtracking step to $A_{i+1} = A_j$ with $j < i$.

A BI analysis $t = (a_1, s_1, \ldots, s_{k-1}, a_k)$ is a *traversal* of a given BI analysis graph G if there exists a schema traversal $T = (A_1, S_1, \ldots, S_{k-1}, A_k)$ of G such that a_1 is an instance of A_1 and for $i = 1..k-1$, a navigation step S_i of G such that a_i is an instance of A_i, s_i is an individual navigation step of generic navigation step S_i, and a_{i+1} is an instance of A_{i+1}.

A traversal of a given BI analysis graph is specified by binding the open variables of some analysis situation of the BI analysis graph and by subsequent bindings of all open navigation variables of navigation steps followed. Notice that the definition of a traversal of a BI analysis graph permits to initially jump to any node of the graph and, once some analysis situation is reached, it does not require to continue with the most specific navigation step defined in the BI analysis graph. The graph acts as guidance and is not prescription, it can be incomplete. A traversal of a navigation step that has been already specialized may lead later to an inclusion of a different specialization in the BI analysis graph.

Example 38 (BI analysis). The right side of Fig. 13 demonstrates the use of analysis graphs AG0, AG1, and AG2, leading to BI analyses AG0', AG1', and AG2'. All open variables of analysis situations and navigation steps are bound. The sequence (A1(1), refocusScore (RatioOfDrugCosts), A3(1), ..., A3(5)) of analysis AG1' is a traversal of AG1.

Once an initial analysis graph has been defined, proper repeated analysis proceeds as follows: (1) Jump to a predefined initial analysis situation (2) Set open parameters for this analysis situation. (3) Evaluate the analysis situation and inspect the result. (4) Choose an outgoing arc to move to another analysis situation. (5) Set an open parameter for this arc, if any. (6) Evaluate and inspect, the result; stop if satisfied or continue with 4.

If the business analyst is not satisfied with the options provided by the BI analysis graph, he or she may add new edges and vertices that move to new terrain or specialize given edges or vertices (in that more parameters are bound and thus closed, or choices of bindings of open parameters are restricted). semCockpit provides reasoning support to determine applicable values for open parameters

and to check consistency of BI analysis graphs, especially with respect to node and link specialization. The whole analysis process can be described in alternating use- and design-phases, i.e., an analyst applies an existing analysis graph (use-phase) and, if necessary, extends or modifies it (design-phase), afterwards she uses the new graph, etc.

Example 39 (Analysis process). Figure 13 demonstrates the explorative, iterative, and incremental characteristics of an analysis process. A business analyst alternates between design and use of analysis graphs. When she or he uses the graph the analyst also performs individual explorations:

(1) Initial analysis graph: AG0 comprises a generic analysis situation A0. The navigation step from A0 to itself with navigation operation correlate(moveToPrev(time)) represents the rule of thumb that if some analysis situation is relevant then the similar analysis situation for the previous time period is also relevant.
(2) Use phase: A business analyst instantiates analysis graph AG0 by binding variables pntGoI.time and pntGoC.time to 2012 and 2011, respectively, resulting in analysis graph AG0' with individual analysis situation A0(1). Following the navigation step proposed in AG0, the business analyst moves to analysis situation A0(3). Further, the business analyst makes *exploratory* individual navigation steps, which are not proposed as such in AG0, leading to analysis situations A0(2), A0(4), and A0(5). Some of these comparisons are deemed relevant, A0(4) and A0(5), others are not, A0(2). In Fig. 13 relevant analysis situations are decorated by a hooklet and others are decorated by a cross. Analysis situations instantiated from generic ones, such as A0(1) and A0(3), are depicted as boxes with solid border. Analysis situations reached through exploratory navigation steps, such as A0(2), A0(4), and A0(5), are depicted as boxes with dotted border.
(3) Design phase: The business analysts wants to re-use analysis situations A0(1), A0(4), and A0(5) in later similar analysis sessions and generalizes them, in analysis graph AG1, to A1, A2, and A3, respectively. Variable pntGoI.time with domain year takes the place of constant 2012 and variable pntGoC.time takes the place of constant 2011. The relation between 2012 and 2011 is represented by constraining pntGoI and pntGoC to comparative concept PreviousYear. Analysis situations A1, A2, and A3 are specializations of A0, which is depicted by specialization links.
(4) Continuation of alternating use and design phases: The business analyst incrementally designs a BI analysis graph, from analysis graph AG1 to analysis AG1' to analysis graph AG2, and so forth. Resulting BI analysis graphs can be re-used in various related analyses, for example, to reiterate the analysis in the next year in order to reassess the conclusions made by the analyst or in order to monitor the effects of actions taken in reaction to the analysis.

7 Guidance, Judgement, and Analysis Rules

Guidance, judgement, and analysis rules provide actionable knowledge about comparative analysis that otherwise is tacit knowledge of a business analyst or captured and processed in some other form, usually outside BI analysis tools.

A simple form of actionable knowledge is supported in many BI systems by the possibility of setting alerters that fire if some measure exceeds a certain threshold. Guidance rules are defined over analysis situations and provide guidance on how to proceed best in analysis, by suggesting a generic or individual navigation step to follow or advising that a particular navigation is not deemed relevant. Judgement rules are defined over facts of a comparative cube and represent static knowledge about possible explanations of a striking low or high score. Analysis rules are defined over facts of a comparative cube as well and decide based on the score of a fact, whether a specific action, e.g., starting a specific analysis, should be taken (action rules), or whether a fact should be reported (reporting rules). Analysis rules are evaluated for specific set of point pairs explicitly identified (e.g. after an ETL cycle), and we will see later that they require different evaluation strategies in the context of inheritance and overriding.

A *guidance rule* is given by (a) a name, (b) an analysis situation (generic or individual), (c) a rule condition which is either (c1) a condition over all facts of the analysis situation (set-oriented rule) or (c2) a condition over a fact of the analysis situation (fact-oriented rule), and (d) a recommended or disadvised generic or individual navigation operation, whose variable domain may be restricted or its variable set (d1) absolute or (d2) relative to variables of the individual analysis situation (for set-oriented rules) or also of coordinates of the fact (for fact-oriented rules) for which the rule fires. In case of a composite analysis situation the guidance rule is defined for the root analysis situation and the rule conditions may also consider the component analysis situations and their facts. Rule conditions are expressed as SQL queries (set- or tuple-oriented) over the (comparative) cubes of the analysis situations. A set-oriented rule (FOR analysis situation ONCE) fires once for an individual analysis situation, if the set-oriented query is not null. A tuple-oriented rule (FOR analysis situation) fires for each fact that is retrieved by the SQL query.

Guidance rules of the same name are organized in an inheritance hierarchy based on specialization relationships between generic analysis situations for which they are defined.

A guidance rule *applies* to an analysis situation if it is an instance of the generic analysis situation for which the rule is defined and there is no more specific generic analysis situation with this property. If applicable, a set-oriented rule *fires* for an individual analysis situation if the rule-condition is satisfied; a fact-oriented rule fires for every fact of the individual analysis situation for which the rule-condition is satisfied, and, depending on the rule, the navigation is recommended (RECOMMEND) or disadvised (DISADVISE). Guidance rules are evaluated after every individual navigation step of a BI analysis.

Several guidance rules may apply to a given individual analysis situation. Since guidance rules suggest potentially promising navigation steps to follow in subsequent analysis, a unique choice is not required but rather it may be worthwhile to explore all options suggested to gain further insight about the data at hand. Inheritance as described above can be used to specialize guidance rules by providing more specific suggestions for more specific analysis situations.

More specific guidance rules override more general ones and, thus, exclude the more general suggestions from the suggestions list. Initially, very general guidance rules may be defined that are extended and specialized over time.

Example 40 (Guidance Rules). Figure 14 shows guidance, judgement, and analysis rules (action and reporting rules). The guidance rules refer to the analysis situation in Fig. 13, but guidance rules may be also defined for generic analysis situations independent of analysis graphs (not shown). GR0 for analysis situation A0 recommends navigation operation correlate(moveToPrev(time)), if the score value is less than 0.8 or greater than 1.2. For analysis situation A3, which is a specialization of A0, guidance rule GR0 for A0 is overridden.

A *judgement rule* is specified by (a) a name, (b) a comparative cube with a single score (defining the facts over which the judgement rule is defined), and (c) a score-value comparison over such a fact, and (d) an informative judgement. Judgement rules are evaluated any time a fact over which the judgement rule is defined is retrieved and fires if the condition is met. Judgement rules of the same name are organized in an inheritance hierarchy along the subset relationships of the set of point-pairs of comparative cubes over which they are defined. A generic judgement rule may be defined for a comparative cube with a generic score whereby the score qualifiers may be constrained in the join condition. A generic judgement rule inherits from another judgement rule if the set of point pairs of the former is a subset of the set of point pairs of the latter, both are for the same score with the same qualifiers, and the domains of the qualifiers are in subset relationships. A generic judgement rule is defined for each fact with point pairs in the comparative cube and with instantiations of the generic score whose actual qualifiers are in the qualifier domains.

Example 41 (Judgement Rules). Judgement rule AR1 for cube C1 in Fig. 14 indicates, once a fact of cube C1 is accessed, that one has to take into account an increase of drug costs of about 5 % per year. The rule is overridden for facts of cube C2, which specializes cube C1 (for year 2012), and justifies an exceptional increase for oral antidiabetic drugs in year 2012.

An *analysis rule* is specified like a judgement rule, but different to judgement rules its action is a recommended action or report and analysis rules need to be explicitly triggered (for example after an ETL-cycle has been completed for the set of new facts just loaded into the DWH) and analysis rules have two conditions (explained later). In its basic form, application and triggering, as well as inheritance is defined as for judgement rules. But, rather than guidance and judgement, which concern an individual analysis situation or fact at hand during a BI analysis, actions and reports are compiled for a bulk of facts where some additional form of abstraction and accompanying rule evaluation strategy is required to reduce repetition and information overload. Moreover, analysis rules with the same name will frequently be for the same action or report, and vice versa.

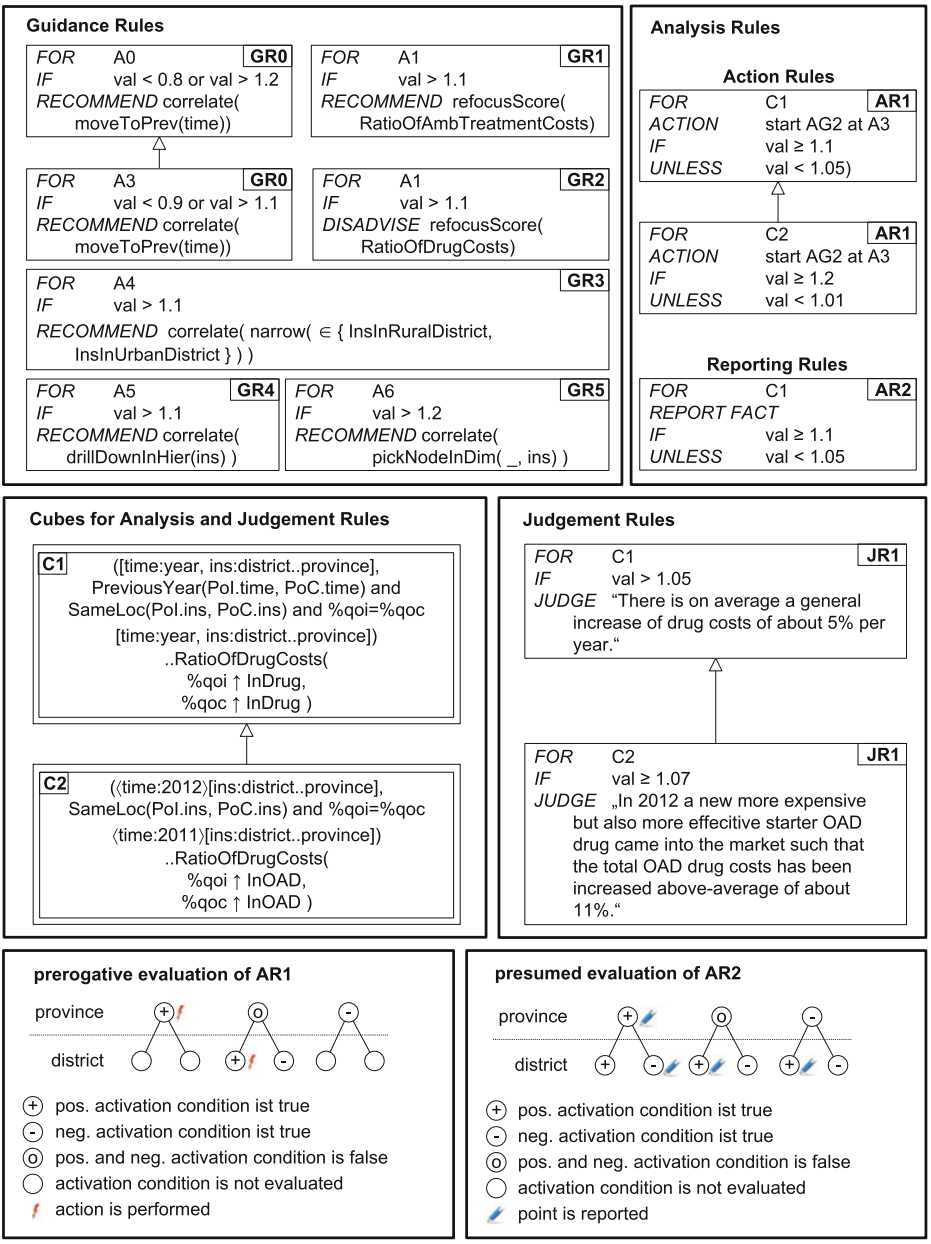


Fig. 14. Guidance, Judgement, and Analysis Rules

It is common in organizational contexts and in law to apply a decision-scope approach [36] to decision making. Higher organization levels set a decision scope within which lower organization levels may operate. In case of conflict the

regulations and rules of a higher level (e.g., European Union) take precedence over those of a lower level (e.g., a member state). Such rules define under which conditions certain actions may, must not, or need to be taken. This approach can also be applied to action and reporting rules with two conditions, one (IF condition or positive activation condition) stating when the rule should fire and the other one (UNLESS condition or negative activation condition) when the rule should not fire. If both conditions are not complementary, a decision scope for more specific rules is left.

Applying the decision scope approach to analysis rules in BI analysis requires to consider two alternative hierarchies: (1) hierarchies of sets of points at the same granularity and (2) the roll-up hierarchy of points in multi-dimensional space.

Specialization along subset relationships between sets of point at the same granularity is governed by the same rules for inheritance and overriding as introduced before. Specialization along a roll-up hierarchy of points in multi-dimensional space actually concerns entities of different kinds, yet connected by some form of part-of relationship. This gives rise to two alternative rule evaluation strategies, both meaningful in practice, but with a different area of application, actioning and reporting in mind.

We first consider the roll-up hierarchy of points (or pairs of points) and we will then discuss the interplay between specialization along subset relationships of points and the roll-up hierarchy of points. We discuss two evaluation strategies. The prerogative strategy which is more appropriate for action rules and the presumed strategy which is more appropriate for reporting rules.

In the *prerogative* strategy, an action triggered for a higher-level point (or pairs of points) implicitly implies the same action for each lower level point (or pairs of points). E.g., if a company decides to abandon a product line (such as mobile phones) this decision is implied for every product (i.e., every phone model in our example) of this product line. In the prerogative evaluation strategy, analysis rules for the same action are evaluated top-down along a user-specified roll-up path of granularities. We assume at first that for points of one granularity at most one analysis rule is defined with a positive and negative activation condition. If one of the two condition applies for a roll-up fact, analysis is completed, whereby the indicated action is triggered, if the positive activation condition is satisfied. Only if both conditions are not satisfied, i.e., for the situation that a fact falls into the open space of the condition scope, the analysis rules for drill-down granularities and drill-down facts at these granularities are considered in further rule evaluation.

Example 42 (Action rules – prerogative evaluation strategy). In Fig. 14 action rule AR1 is defined for facts in cube C1. Remember that action rules need to be explicitly evaluated for a specific set of facts (e.g., those just loaded into the data warehouse), identified by multi-granular cube and for a roll-up path of granularities. This roll-up path needs not to be indicated, if the analysis rule is only over multi-granular cubes with granularities along a single roll-up path, which is the case for AR1. If rule AR1 fires, it starts a traversal of analysis graph

AG2 at situation A3 of Fig. 13. For AR1 we have a prerogative evaluation strategy. If the positive activation condition ($val \geq 1.1$) is true for a province, AG2 is started for that province. If neither the positive ($val \geq 1.1$) nor the negative ($val < 1.05$) activation condition is true, the rule evaluates the conditions for districts to decide whether an analysis graph traversal is triggered.

In the *presumed* strategy, a report triggered for a higher-level point is presumed to cover also lower-level points and is thus not reported again for lower-level points (to avoid unnecessary information overload, the basic motivation behind performing roll-up analysis in data warehousing), unless a lower-level point fulfills the negative activation condition of a more specific rule. E.g., if a reporting rule triggers the report that average treatment costs for patients with diabetes mellitus of type 2 patients in Austria are twice as high than in Germany last year, it is presumed that this will hold in general for each province in Austria. Minor deviations are generally not of interest in comparative data analysis, but major ones are. The knowledge what kind of exceptions should be reported can be expressed by a negative activation condition of a more specific rule. In our example such a rule may state that for comparing treatment costs for patients in a province of Austria with patients in Germany, a percentage difference of less than 20 % should not be reported. Assume a positive activation condition of a more general rule has led to report a striking difference of average treatment costs per patient in Austria versus Germany. Based on deviating observation for comparing Tyrol (a province of Austria) with Germany the more specific rule will report (if the difference is less than 20 %) that contrary to Austria as a whole, average treatment costs per patient have been similar when comparing Tyrol with Germany.

Example 43 (Reporting rules – presumed evaluation strategy). AR2 of Fig. 14 represents a reporting rule, which is evaluated in a presumed manner. If the positive activation condition ($val \geq 1.1$) is true for a province, the province is reported, but districts of the province are only reported, if the evaluation of the negative activation condition ($val < 1.05$) is true for a district of that province. In this case the district is reported as an exception.

Prerogative and presumed evaluation of analysis rules along roll-up hierarchies in top-down manner can be combined with evaluation along hierarchies of sets points at the same granularity according to the specialization principle described above for guidance rules. Again we assume for simplicity, that analysis rules are triggered for points along a single-drill-down path of granularities. At each granularity all analysis rules defined for points at that granularity are considered, and for each fact the most specific one is chosen (It is assumed here, as that the specialization hierarchy is consistent such that a single most specific rule exists). We also expect that the rules have been specialized in a consistent way according to the decision scope approach such that the positive activation condition of a more general rule implies the activation condition of a more specific rule, and the same holds for negative activation rules. In the prerogative

strategy, once a decision has been determined for a fact at a given granularity, analysis stops; if no decision could be made due to an open decision scope, analysis continues with drill-down facts at a lower-level granularity for which an analysis rule is defined. In the presumed evaluation strategy, a reporting rule that once fired for a fact, is not triggered again for drill-down facts to avoid information overload, unless reporting an exception is justified by a negative activation condition.

Example 44 (Specialization of an action rule). In Fig. 14 action rule AR1 for cube C1 is overridden by the same named action rule AR1 for cube C2. If the rule is applied to a multi-dimensional point with year 2012, the specialized rule AR1 for cube C2 is evaluated.

8 Conclusion

Existing BI tools are well-suited for reporting and for performing complex analysis tasks but lack an explicit formalization of knowledge about business terms, comparison, and analysis processes. Commonly, business analysts define business terms in an ad-hoc manner rather than explicitly capturing business terms and their meaning in a central, shared repository. An unambiguous formalization of business terms can be used for the definition of measures, which facilitates the analysis task of a business analyst. Furthermore, the definition of meaningful comparisons should not be left to human intuition but captured as a first-class citizen. Similarly, the analysis process itself could be formalized. The experience of business analysts should be made explicit in order to benefit all business analysts within a company.

Rather than solving each issue separately, we presented an integrated and coherent approach for ontology-driven comparative data analysis.

The centerpiece of the Semantic Cockpit approach is the multi-dimensional ontology (MDO) which enriches a data warehouse with a set of concepts. These MDO concepts unambiguously define business terms and their meaning in the context of data analysis. Existing domain ontologies can be integrated as semantic dimensions. Analysts use MDO concepts for the definition of measures and scores. Measures quantify real-world facts of interest. Scores contrast measure values of a group of interest with a comparison group. Generic measures and scores reduce the number of measures that have to be defined. Judgement rules provide possible explanations for striking results that are encountered during an analysis process. Analysis rules trigger particular analysis processes or report facts.

The semCockpit project extends the enterprise data warehouse by a static semantic layer which assists analysts in formulating analytical queries in high-level business terms. During the project we identified the need to also capture knowledge about analysis processes. Future research concerns modeling and sharing of such knowledge in a semantic BI process layer based on the ideas of analysis graphs and guidance rules sketched herein.

Acknowledgments. This work is funded by the Austrian Ministry of Transport, Innovation, and Technology in program FIT-IT Semantic Systems and Services under grant FFG-829594 (*Semantic Cockpit:* an ontology-driven, interactive business intelligence tool for comparative data analysis).

References

1. Anderlik, S., Neumayr, B., Schrefl, M.: Using domain ontologies as semantic dimensions in data warehouses. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 88–101. Springer, Heidelberg (2012)
2. Bellatreche, L., Giacometti, A., Marcel, P., Mouloudi, H., Laurent, D.: A personalization framework for OLAP queries. In: Song, I.-Y., Trujillo, J. (eds.) DOLAP, pp. 9–18. ACM, New York (2005)
3. Bentayeb, F., Favre, C.: RoK: roll-up with the K-means clustering method for recommending olap queries. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2009. LNCS, vol. 5690, pp. 501–515. Springer, Heidelberg (2009)
4. Berger, S., Schrefl, M.: Feddw: A tool for querying federations of data warehouses - architecture, use case and implementation. In: Cordeiro, J., Filipe, J. (eds.) ICEIS (1), pp. 113–122 (2009)
5. Buchheit, M., Nutt, W., Donini, F.M., Schaerf, A.: Refining the structure of terminological systems: Terminology = schema + views. In: Hayes-Roth, B., Korf, R.E. (eds.) AAAI, pp. 199–204. AAAI Press/The MIT Press (1994)
6. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The mastro system for ontology-based data access. Semant. Web **2**(1), 43–53 (2011)
7. Ceri, S., Brambilla, M., Fraternali, P.: The history of webML lessons learned from 10 years of model-driven development of web applications. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 273–292. Springer, Heidelberg (2009)
8. Das, S., Chong, E.I., Eadon, G., Srinivasan, J.: Supporting ontology-based semantic matching in rdbms. In: Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B. (eds.) VLDB, pp. 1054–1065. Morgan Kaufmann (2004)
9. Fikes, R., Kehler, T.: The role of frame-based representation in reasoning. Commun. ACM **28**(9), 904–920 (1985)
10. Geerts, F., Kementsietsidis, A., Milano, D., et al.: *i*MONDRIAN: a visual tool to annotate and query scientific databases. In: Böhm, C. (ed.) EDBT 2006. LNCS, vol. 3896, pp. 1168–1171. Springer, Heidelberg (2006)
11. Giacometti, A., Marcel, P., Negre, E., Soulet, A.: Query recommendations for OLAP discovery driven analysis. In: Song, I.-Y., Zimányi, E. (eds.) DOLAP, pp. 81–88. ACM (2009)
12. Golfarelli, M., Maio, D., Rizzi, S.: The dimensional fact model: a conceptual model for data warehouses. Int. J. Coop. Inf. Syst. **7**(2–3), 215–247 (1998)
13. Golfarelli, M., Rizzi, S., Biondi, P.: myOLAP: an approach to express and evaluate OLAP preferences. IEEE Trans. Knowl. Data Eng. **23**(7), 1050–1064 (2011)
14. Heer, J., Mackinlay, J.D., Stolte, C., Agrawala, M.: Graphical histories for visualization: supporting analysis, communication, and evaluation. IEEE Trans. Vis. Comput. Graph. **14**(6), 1189–1196 (2008)
15. Heer, J., Shneiderman, B.: Interactive dynamics for visual analysis. Commun. ACM **55**(4), 45–54 (2012)

16. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation, 27 October 2009. http://www.w3.org/TR/owl2-primer/
17. Hürsch, W.L.: Should superclasses be abstract? In: Pareschi, R. (ed.) ECOOP 1994. LNCS, vol. 821, pp. 12–31. Springer, Heidelberg (1994)
18. Hurtado, C.A., Mendelzon, A.O.: Reasoning about summarizability in heterogeneous multidimensional schemas. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, p. 375. Springer, Heidelberg (2001)
19. Jerbi, H., Ravat, F., Teste, O., Zurfluh, G.: Preference-based recommendations for OLAP analysis. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 467–478. Springer, Heidelberg (2009)
20. Khouri, S., Bellatreche, L.: A methodology and tool for conceptual designing a data warehouse from ontology-based sources. In: Song, I.-Y., Ordonez, C. (eds.) DOLAP, pp. 19–24. ACM (2010)
21. Lehner, W., Albrecht, J., Wedekin, H.: Normal forms for multidimensional databases. In: Rafanelli, M., Jarke, M. (eds.) SSDBM, pp. 63–72. IEEE Computer Society (1998)
22. Lim, L., Wang, H., Wang, M.: Unifying data and domain knowledge using virtual views. In: Koch, C., Gehrke, J., Garofalakis, M.N., Srivastava, D., Aberer, K., Deshpande, A., Florescu, D., Chan, C.Y., Ganti, V., Kanne, C.-C., Klas, W., Neuhold, E.J. (eds.) VLDB, pp. 255–266. ACM (2007)
23. Malinowski, E., Zimányi, E.: Hierarchies in a multidimensional model: from conceptual modeling to logical representation. Data Knowl. Eng. **59**(2), 348–377 (2006)
24. Nebot, V., Llavori, R.B.: Building data warehouses with semantic web data. Decis. Support Syst. **52**(4), 853–868 (2012)
25. Nebot, V., Berlanga, R., Pérez, J.M., Aramburu, M., Pedersen, T.B.: Multidimensional integrated ontologies: a framework for designing semantic data warehouses. In: Spaccapietra, S., Zimányi, E., Song, I.-Y. (eds.) Journal on Data Semantics XIII. LNCS, vol. 5530, pp. 1–36. Springer, Heidelberg (2009)
26. Neuböck, T., Neumayr, B., Rossgatterer, T., Anderlik, S., Schrefl, M.: Multidimensional navigation modeling using BI analysis graphs. In: Castano, S., Vassiliadis, P., Lakshmanan, L.V.S., Lee, M.L. (eds.) ER Workshops 2012. LNCS, vol. 7518, pp. 162–171. Springer, Heidelberg (2012)
27. Neumayr, B., Schrefl, M., Thalheim, B.: Hetero-homogeneous hierarchies in data warehouses. In: Link, S., Ghose, A. (eds.) APCCM. CRPIT, vol. 110, pp. 61–70. Australian Computer Society (2010)
28. Neumayr, B., Schütz, Ch., Schrefl, M.: Semantic enrichment of OLAP cubes: multidimensional ontologies and their representation in SQL and OWL. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) OTM 2013. LNCS, vol. 8185, pp. 624–641. Springer, Heidelberg (2013)
29. Niinimäki, M., Niemi, T.: An etl process for olap using rdf/owl ontologies. In: J. Data Semantics [39], pp. 97–119
30. Pardillo, J., Mazón, J.-N., Trujillo, J.: Extending OCL for OLAP querying on conceptual multidimensional models of data warehouses. Inf. Sci. **180**(5), 584–601 (2010)
31. Romero, O., Abelló, A.: Automating multidimensional design from ontologies. In: Song, I.-Y., Pedersen, T.B. (eds.) DOLAP, pp 1–8. ACM (2007)
32. Romero, O., Abelló, A.: Open access semantic aware business intelligence. In: Zimányi, E. (ed.) eBISS 2013. LNCS, vol. 7911, pp. xx–yy. Springer, Heidelberg (2014)

33. Romero, O., Marcel, P., Abelló, A., Peralta, V., Bellatreche, L.: Describing analytical sessions using a multidimensional algebra. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 224–239. Springer, Heidelberg (2011)
34. Saggion, H., Funk, A., Maynard, D., Bontcheva, K.: Ontology-based information extraction for business intelligence. In: Aberer, K. (ed.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 843–856. Springer, Heidelberg (2007)
35. Sapia, C.: On modeling and predicting query behavior in OLAP systems. In: Gatziu, S., Jeusfeld, M.A., Staudt, M., Vassiliou, Y. (eds.) DMDW. CEUR Workshop Proceedings, vol. 19, pp. 1–10. CEUR-WS.org (1999)
36. Schrefl, M., Neumayr, B., Stumptner, M.: The decision-scope approach to specialization of business rules: Application in business process modeling and data warehousing. In: Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling (APCCM 2013) (2013)
37. Silver, B.: BPMN Method and Style, 2nd edn., with BPMN Implementer's Guide: A Structured Approach for Business Process Modeling and Implementation Using BPMN 2.0. Cody-Cassidy Press (2011)
38. Skoutas, D., Simitsis, A., Sellis, T.K.: Ontology-driven conceptual design of etl processes using graph transformations. In: J. Data Semantics [39], pp. 120–146
39. Spaccapietra, S., Zimányi, E., Song, I.-Y. (eds.): Journal on Data Semantics XIII. LNCS, vol. 5530. Springer, Heidelberg (2009)
40. Spahn, M., Kleb, J., Grimm, S., Scheidl, S.: Supporting business intelligence by providing ontology-based end-user information self-service. In: Duke, A., Hepp, M., Bontcheva, K., Vilain, M.B. (eds) OBI. ACM International Conference Proceeding Series, vol. 308, p. 10. ACM (2008)
41. Staudt, M., Jarke, M., Jeusfeld, M.A., Nissen, H.W.: Query classes. In: DOOD, pp. 283–295 (1993)
42. Thalhammer, T., Schrefl, M., Mohania, M.K.: Active data warehouses: complementing OLAP with analysis rules. Data Knowl. Eng. **39**(3), 241–269 (2001)
43. Thollot, R.: Dynamic Situation Monitoring and Context-Aware BI Recommendations. PhD thesis, Ecole Centrale Paris (2012)
44. Trujillo, J., Gómez, J., Palomar, M.S.: Modeling the behavior of OLAP applications using an UML compliant approach. In: Yakhno, T. (ed.) ADVIS 2000. LNCS, vol. 1909, pp. 14–23. Springer, Heidelberg (2000)

Open Access Semantic Aware Business Intelligence

Oscar Romero(✉) and Alberto Abelló

Universitat Politècnica de Catalunya, BarcelonaTech, Barcelona, Spain
{oromero,aabello}@essi.upc.edu

Abstract. The vision of an interconnected and open Web of data is, still, a chimera far from being accomplished. Fortunately, though, one can find several evidences in this direction and despite the technical challenges behind such approach recent advances have shown its feasibility. Semantic-aware formalisms (such as RDF and ontology languages) have been successfully put in practice in approaches such as Linked Data, whereas movements like Open Data have stressed the need of a new open access paradigm to guarantee free access to Web data.

In front of such promising scenario, traditional business intelligence (BI) techniques and methods have been shown not to be appropriate. BI was born to support decision making within the organizations and the data warehouse, the most popular IT construct to support BI, has been typically nurtured with data either owned or accessible within the organization. With the new linked open data paradigm BI systems must meet new requirements such as providing on-demand analysis tasks over any relevant (either internal or external) data source in *right-time*. In this paper we discuss the technical challenges behind such requirements, which we refer to as *exploratory BI*, and envision a new kind of BI system to support this scenario.

Keywords: Semantic web · Business intelligence · Data warehousing · ETL · Multidimensional modeling · Exploratory business intelligence · Data modeling · Data provisioning

1 Introduction

The Internet empowered the interconnection of different systems and contributed to the bloom of massive and heterogeneous distributed systems that brought new challenges of data integration. The data integration problem [1] aims at providing users with a single unified view of different and interconnected data repositories. This is an old and recurrent topic for the database community and as such it has been thoroughly studied in the past. As discussed in [2], data integration must overcome a number of heterogeneities present in the systems to be interconnected that can be classified in two categories: *system* and *semantic heterogeneities*. On the one hand, system heterogeneities include differences in

E. Zimányi (Ed.): eBISS 2013, LNBIP 172, pp. 121–149, 2014.
DOI: 10.1007/978-3-319-05461-2_4,

hardware, operating systems, database management systems (DBMS) and so on. On the other hand, semantic heterogeneities include differences in the way the real-world is modeled in terms of the database schema.

In the recent past, when the database world was mostly relational, different architectures were developed for data integration, such as federated databases, multidatabase systems and wrappers and mediators [3]. However, semantic heterogeneities are still an open issue for relational systems. Data semantics gathered (as metadata[1] in the database catalog) by relational systems (RDBMS) are not enough as to enable automatic design decisions to overcome semantic heterogeneities and most of the burden to get rid of such heterogeneities still relies on the designer's shoulders.

The arrival of the Internet did nothing but worsen the problem. Massive distributed scalable Web-systems put the traditional relational systems under stress and nowadays relational systems co-exist with non-relational systems, commonly known as NOSQL (to be read Not Only SQL and never as ¬SQL). The first systems that coined the NOSQL term were born on the Web but soon the idea spread to many other areas and currently it claims for specialized database solutions for specific problems. Although NOSQL is mainly a buzzword referring to a way of doing (perfectly captured in the "*one size does not fit all*" motto) rather than new technical solutions, there have been some attempts to classify and find common aspects of these systems. One of the most spread features among NOSQL systems is being *schemaless*. A schemaless database does not have a explicit schema created by the user (e.g., by means of the `CREATE TABLE` statement for RDBMS). Nevertheless, an implicit schema remains. In general, this situation is not desirable at all, as semantics are lost. In terms of the ANSI / SPARC architecture, schemaless databases do not define external and conceptual schemas and query languages must access the DBMS internal data structures, which violates the logical and physical independence principle. Therefore, data is a black-box with no *meaning* for the DBMS and, for example, in some extreme cases such as key-value stores, the value becomes a meaningless chunk of bytes. As consequence, one gains in flexibility but misses even more semantics as metadata gathered by most NOSQL systems is almost non-existent.

As consequence, the current picture is that of massive (independent) systems gathering few metadata, known as *data silos*, which ideally should intercommunicate in order to solve bigger problems. However, data integration becomes even tougher as a wider range of system and semantic heterogeneities must be considered.

Far away from this scenario stands Tim-Berners Lee's vision of an *interconnected* and *open* Web of data [4]. The Semantic Web envisioned by Berners Lee considered *linking* data in such a way a machine could process the links and automatically explore the data without human intervention. Relating data *opens* the data silos and allows navigating, crossing, exploring and analysing data in order to

[1] Metadata, or data about data, keeps track of any relevant information regarding data. For example, a value of 4 means nothing by itself. But if the system knew it refers to the number of children of a certain person as of 2013 it becomes information.

produce unexpected results and relevant (hidden) knowledge. Berners Lee referred to this feature as data fusion [5], and its most popular implementation is by means of *linked open data*. On the one hand, enriching data with machine processable metadata so that machines can *understand* (i.e., manipulate) data is known as linked data [5]. On the other hand, similar to the open source concept, open data describes data that is freely available and can be used as well as republished by everyone without restrictions from copyright or patents [6].

From the database point of view, the linked open data wave demands new systems able to store linked data (thus, semantically rich metadata must be stored together with data) and support the open data initiative (these systems should be easily identified and accessed by anyone). Such systems would provide foundations for more elaborated applications such as mashups, linkeable browsers and semantic-aware search engines. In this paper, we focus on the need for new generation decision support systems built on top of that, which can also be used as foundation for any traditional data fusion application on the Web.

2 Business Intelligence: Past, Present and Future

Decision support systems play a key role in many organizations. These systems provide accurate information (understood as the result of processing, manipulating and organizing data in a way that adds new knowledge to the person or organization receiving it) that leads to better decisions and gives competitive advantages. In the past, the ability of decision makers for foreseeing upcoming trends was crucial for any organization but this largely subjective scenario changed when the world became digital. Actually, any event can be recorded and stored for later analysis, which provides new and objective business perspectives to support managers in the decision making process. Hence, (digital) information is a valuable asset to organizations and gathering, transforming and exploiting such information is nowadays a technological challenge commonly known as *Business Intelligence* (BI). Thus, BI embraces many different disciplines (e.g., from databases to data mining) and solutions meant to support decision making based on (digitally recorded) evidences.

Among all architectural solutions proposed for BI, data warehousing is possibly the most popular one. According to [7], *Data Warehousing is a collection of methods, techniques and tools used to support knowledge workers -senior managers, directors, managers and analysts- to conduct data analyses that help with performing decision making processes and improving information resources.*

Data warehousing mainly focus on decision making and data analysis, and at the same time, these systems abstract technical challenges like data heterogeneity or data sources implementation. This is a key factor in data warehousing in particular, and business intelligence in general. Nowadays, many events can be recorded within organizations but the way each event is stored differs in every organization and it depends on several factors such as relevant attributes for the organization (i.e., their business needs), technology used (i.e., implementation), analysis task performed (i.e., data relevant for decision making), etc. These systems gather and assemble relevant data available within the organization from

various and probably heterogeneous sources in order to produce a single, detailed view of the organization that can be used for enabling better strategic management. In other words, data warehousing overcomes the data integration problem by means of data consolidation.

Although data warehousing ecosystems are highly complex systems, one can say that data warehousing is mature enough (even if data warehousing projects are still far away from being well-controlled). After 20 years, several methods and tools to support the design, deployment and maintenance of such systems have been introduced (e.g., [8–11]) as long as methodologies and good practices (e.g., [12,13]). As result, there exist data warehousing experts who have been working and mastering these problem for several years.

Building a data warehouse system consists of designing the data warehouse and creating the ETL (Extract-Transform-Load) processes to populate the data warehouse from the sources. Ideally, the data warehouse schema should subsume any analytical requirement that end-users may pose to the system. Past experience has highlighted three main challenges that complicate designing and deploying data warehousing systems [7,8]:

1. Like in any other information system, the design process starts by eliciting and gathering the end-user requirements. The burden of such process relies on the DW designer and the end-user does not actively participate in this task. However, it has been shown that IT people (i.e., the DW designer) and non-IT people (i.e., the DW end-user) do not understand each other easily and it is rather common that this step fails to meet the end-user requirements and, in turn, the whole DW ecosystem fails.
2. After gathering the end-user requirements, the designer proceeds to design and create the DW. Next, the ETL processes to populate the DW from the available sources are designed and implemented. In this step, the DW designer is meant to understand the available data sources, identify the data with which the requirements gathered can be meet and construct the ETL flows. This step may take up to 70 % of the whole DW project span of time and they entail complex and time-consuming transformations. Accordingly, the update window of a data warehouse (the time it takes to load data into the data warehouse) can be of hours, and executed daily or even weekly. Also, ETL constructs are several times directly implemented at the logical or physical level, which troubles its maintenance.
3. Finally, once the DW ecosystem is running, intuitive and non-technical means to query the DW are mandatory. End-users tend to be non-IT people and require ad hoc formalisms to understand and exploit the data warehouse. For example, one cannot assume that the company CEO masters the SQL language as to query a database on his / her own.

For all these reasons data warehousing projects are traversal and people from different areas must work together in order to produce a flexible, powerful, and successful system. Ideally, the construction of the DW and ETL designs must undergo several rounds of reconciliation and redesigning, until all business needs – even some that are not described from the beginning of the project– are satisfied.

Thus, a DW project is rarely completed. Typically, a DW is an alive ecosystem and thus, maintainability and evolution aspects should be considered too [14].

Beyond its complexity, a successful DW project is a valuable asset for any organization that can effectively exploit the objective evidences stored in its data sources. However, traditional data warehousing techniques have shown not to be adequate for the next generation of BI systems. In terms of the previous section, traditional data warehousing techniques are meant to create independent data silos, whereas current trends claim for interconnecting different data sources (including external sources) and provide a global unified view. This evolution of the data warehousing systems is referred to as DW 2.0, new generation BI and similar concepts [6,15–20] that have bloomed recently. According to them, ideally, (i) the end-user should dynamically explore any data source of potential interest (no matter if it is internal or external to the organization) that is available and (ii) any desired analysis task should be made in *right-time* (i.e., according to the user needs, which usually means near real-time). A thorough analysis of these requirements elicit new needs for the next generation BI systems. Specifically:

1. Any desired task in right-time implies:
   - The analysis must be conducted over fresh data and ideally, get rid of the update window concept.
   - The end-user must be able to state his / her analysis requirements in a non-technical language. Ideally, the end-user must state what data want to analyze and from which perspectives without the intervention of the DW designer.
2. Analyze any source means to reconsider how ETL processes should work:
   - Extraction: If any source can be considered, flexible extraction techniques must be able to extract data from structured, semi-structured and non-structured data.
   - Transformation: According to the end-user analytical needs, the extracted data must undergo different transformation rounds to meet the desired quality threshold to be agreed with the end-user.
   - Load: Loading data into the data warehousing can be costly. For this reason, and honouring the right-time requirement, next generation systems may decide not to materialize the consolidated data into a data warehouse but dynamically consume it.

Of course, all these requirements must be put into perspective and they will be rarely met. A graphical representation of traditional and new BI flows is depicted in Fig. 1, and a discussion follows.

As shown in Fig. 1, data sources can be globally divided into two groups: internal and external data. Internal data is that owned by the company and it traditionally consisted of relational databases and tabular data (e.g., Excel or CVS files). With the time, semi-structured data (such as XML) or non-structured data (e.g., plain text files, pdf files, videos or images) were also incorporated and, nowadays, NOSQL sources (e.g., graph databases or key-value / document

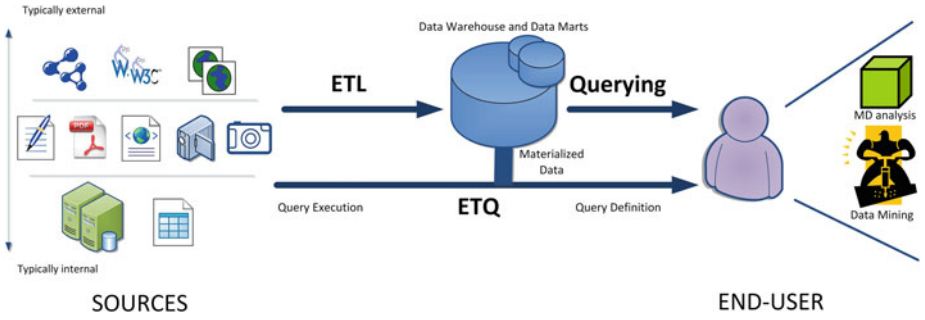


Fig. 1. BI Elements and Data Flows

stores) may need to be considered. External data mainly refers to open and / or linked data (ideally, open linked data). Open data normally consists of semi-structured data. Its structure typically depends on who delivers the data. For example, governmental institutions[2] usually open tabular data or PDF files, while social networks such as Facebook or Twitter usually deliver XML or JSON files[3]. As special case, external data coming from other organizations after subscribing an agreement may also be considered. This is a typical case in e-science scenarios where research institutes tend to collaborate and share their data. In these cases, an agreement is signed and external access to any kind of source may follow. In any case, a vast amount of heterogeneous data sources may need to be considered.

Next, after undergoing several transformations, the extracted consolidated data might be materialized in a DW. In such scenario, smaller, specific data marts might be modeled. Analytical tools (i.e., data mining, OLAP or query & reporting) are available on top of the DW (or data marts) to allow non-technical end-users to query, navigate and exploit the data. This suits traditional data warehousing techniques. However, next BI systems claim for a more flexible architecture where the end-user can define a query (what data to be analyzed and how) and the system would dynamically decide what sources should be considered, extract data (maybe also from the available DW), do the corresponding transformations and visualize the results. To stress that this approach is query-oriented, we refer to it as ETQ (Extract-Transform-Query). Furthermore, given that the data warehouse could not even exist, it is sometimes meaningless to use the data warehousing term to refer to this new kind of systems, which are better defined as Business Intelligence systems. From here on, we will refer to traditional data warehousing systems as traditional BI and next generation systems as exploratory BI. Thus, we stress the fact that in these systems the aim is at *exploring* data sources to perform right-time analytical tasks.

[2] For example: http://data.gov.uk/ and http://www.data.gov/

[3] See http://www.w3.org/DesignIssues/LinkedData.html for a detailed description of the 5-stars of linked data.

Importantly, note that exploratory BI requires a huge degree of automation in both design and implementation of the query and ETQ tasks. In traditional BI systems the designer is responsible for properly understanding the business requirements, look for the proper data in the sources needed to answer requirements and produce correct data warehouse / ETL models meeting all requirements. Similarly, the designer is responsible for a correct integration of new requirements that may appear. This scenario must change and exploratory BI systems must support the designer as much as possible, and automate the most possible tasks. Of course, the designer (ideally, the end-user) will need to supervise the process and, eventually, accept or disregard different modeling options automatically produced. However, automation requires machine-processable metadata.

2.1 Challenges of Exploratory BI

In the previous section we have just discussed the characteristics of next generation exploratory BI systems. In this section we further elaborate on the IT challenges behind exploratory BI and focus on the technical challenges to overcome in order to (i) *allow the end-user dynamically explore any data source of potential interest (either internal or external to the company)* and (ii) *perform any analysis task in right-time*. More specifically, we divide the technological challenges in three, according to the typical life-cycle of IT systems: requirement engineering, modeling and physical deployment.

Requirement Elicitation and Specification. Exploratory BI systems are goal-oriented (one may consider the analysis needs stated by the end-user as the system goals). Thus, the end-user must express his / her needs without the help of IT people (*requirement elicitation*) and should be internally translated into a machine processable formal specification (*requirement specification*). By needs we either refer to *information needs* (that roughly speaking can be mapped to a query retrieving data) and *quality needs* (some quality criteria that the information retrieved must fulfill and typically expressed in terms of non-functional requirements).

In a truly exploratory system, requirements gathered must be used to lead the next stages (design and implementation of the system) and consequently, the language used to express the user needs (ideally, close to the domain vocabulary used by end-users) must be also *understood* by computers (i.e., it must have precise associated semantics). As later discussed in Sect. 3, semantic-aware formalisms are the most promising means to describe such *reference domain language*.

All in all, these systems must be *user-centric* and *semantic-aware*. On the one hand, the end-user must express analysis needs using his / her own words. On the other hand, the system must also talk the user language and automatically process the requirements, which will lead the next two stages. A comprehensive list of challenges related to this area follows:

- End-users must state their *information and quality analysis needs* in terms of a high-level language close to their own vocabulary (i.e., *the reference domain language*).
- The reference domain language must be computer understandable in order to enable the desired automation described in the next two stages.

Automatic Design. From the end-user requirements gathered in the previous stage, exploratory systems must automatically design the system and make this inherent complex task transparent to the end-user. Assuming a data warehousing architecture, it entails to design the integration layer schema (if any) and the ETL (ETQ) processes. More specifically, the system must be able to explore the available (or potential) data sources and, according to the end-user requirements expressed in terms of the reference domain language and some *design quality criteria*, re-arrange the data source schemas in the shape of an integration layer (which can be thought as a view over the data sources) meeting the analysis needs at hand. Once the integration schema is available, the schema transformations identified must be lowered to the instance level and produce the ETL (ETQ) design needed to populate (answer) end-user analysis needs.

Relevantly, note that this entails that the system must be able to understand what data and how is stored at each potential data source. Thus, the system must track the potential data sources and, in order to allow automatic processing, understand what data and how is stored at each source. Thus, it must be able to *map* the data sources to the reference domain language used by the system. Ideally, such map should follow a local-as-view (LAV) approach (similar to that followed by linked data) and thus, concepts in the data sources must point to the reference domain language. In order to automatically consider the end-user quality needs the data sources should also *declare* what quality criteria they might meet.

As consequence, an exploratory system should keep track of potential sources and understand what data and how is stored in the data sources. In other words, to open the black box. To achieve such goal, the data sources schema should be mapped to the reference domain language used by the system. Then, by considering the end-user needs gathered in the previous stage, the exploratory system should be able to design the integration layer and ETL (ETQ) flows needed to answer the analysis needs at hand. A comprehensive list of challenges related to this area follows:

- Potential data sources must be tracked into the system by mapping to the reference domain language both their schematas and the quality criteria they might meet. Ideally, a LAV approach is needed.
- Automatic algorithms for the design of the integration layer and the ETL (ETQ) processes are needed.
- Means to express some design quality criteria to lead the automatic creation of potential designs are mandatory.

Automatic Deployment. The physical deployment of such systems should also be transparent to the end-user. Such deployment should consider the design created in the previous stage and according to some quality needs (either those explicitly stated by the end-user or introduced by the *system administrator*) choose the execution engine to compute the answer to the analysis needs posed to the system. On the one hand, exploratory systems should explore self-tuning systems to the limit and dynamically choose the best storage option as well as auxiliary storage techniques (such as indexing or materialized views) in order to improve performance. On the other hand, an optimizer should be available to optimize ETL (ETQ) executions (similar to the query optimizer of a database). Both issues can drastically benefit from the reference domain language and use the semantic-aware metadata gathered during the whole process in order to determine the best design / execution strategies.

Thus, the system should be able to trigger self-tuning tasks in accordance with the metadata gathered during the whole process, as well as implement an optimizer to improve the ETL (ETQ) internal processes. A comprehensive list of challenges related to this area follows:

- Advanced self-tuning techniques are needed.
- ETL (ETQ) optimizers need to be developed.
- Means to express some quality criteria to guide the system self-tuning and ETL (ETQ) optimization techniques internally carried out are mandatory.

As the reader may note, exploratory systems are user-centered and thus, the ultimate goal of such systems is to allow the end-user state his / her analysis needs (both from the point of view of the information needed and the associated quality metrics associated to it) and accordingly produce the design and physical implementation of the needed constructs in an automatic fashion. As discussed, this must be achieved by means of semantic-aware systems that, by means of a reference domain language, are able to map the end-user requirements and the data source schemas to a common reference domain language. All in all, the database expert role may seem to dilute in this new approach. However, nothing further from the truth. Exploratory systems place the focus on the end-user, who is the real domain expert, and makes him / her play an active role when building the system. The database expert, however, is still essential in this approach. The automatic stages carried out by exploratory systems are responsible for designing and deploying the system. As discussed, some general quality metrics to guide the design and deployment are mandatory. Such guidance must be continuously monitored by a database expert who is expected to react in front of unexpected changes or system misbehaviour (e.g., by changing the design or deployment quality criteria to be met by the automatic created solutions). As consequence, although moved away from the focus, the database expert is still needed to monitor and tune the system.

3 An Introduction to Semantic Web Formalisms

Nowadays, we can find several formalisms to capture semantics in a machine-processable format. Typically, these formalisms come from the Semantic Web (SW), which is aimed at providing the necessary representation languages and tools to express semantic-based metadata. Prior to the SW, there were several efforts to provide metadata formats to the web contents, such as Dublin-Core[4], whose main purpose was to improve information discovery and retrieval. However, these formats were shown very limited mainly due to their poor expressivity and little Web-awareness. As result, the W3C proposed new representation formats, all relying on XML[5], to overcome the limitations of existing metadata formats. The main idea behind these formats is that any concept or instance used for describing a Web object must be referred through a unique resource identifier (URI). Thus, the most basic way to describe an object consists of creating a link to the URI that represents the intended semantics. With the resource description framework (RDF)[6], we can create more complex metadata elements allowing the representation of relationships between descriptors (i.e., triples). Additionally, the RDFS[7] extension allows users to define a schema for RDF descriptions. More expressive semantic descriptions have been also proposed by adopting logic-based frameworks: DAML+OIL[8] and the Ontology Web Language (OWL)[9]. Contrary to RDFS, all these languages rely on description logics, which are tractable subsets of the first order logic (FOL). In this context, metadata is governed by logic axioms over both classes and instances (assertions). Like in RDFS, logic axioms in these formats must be defined over Web-based references (i.e. URIs).

In the next sections we further elaborate in the most popular SW formalisms: RDF and ontology languages.

3.1 RDF(S)

In RDF there are three kinds of elements: resources, literals, and properties. Resources are web objects (entities) that are identified through a URI, literals are atomic values such as strings, dates, numbers, etc., and properties are binary relationships between resources and literals. Properties are also identified through URIs. The basic building block of RDF is the triple: a binary relationship between two resources or between a resource and a literal. For example, consider the following triples depicted in Fig. 2.

In this example, the concept (object) eBISS 2013 is represented by the `http://uri-repository/eBISS2013` URI and it is related through the `http://`

[4] http://dublincore.org/
[5] http://www.w3.org/XML/
[6] http://www.w3.org/RDF/
[7] http://www.w3.org/TR/rdf-schema/
[8] http://www.w3.org/TR/daml+oil-reference/
[9] http://www.w3.org/TR/owl2-overview/

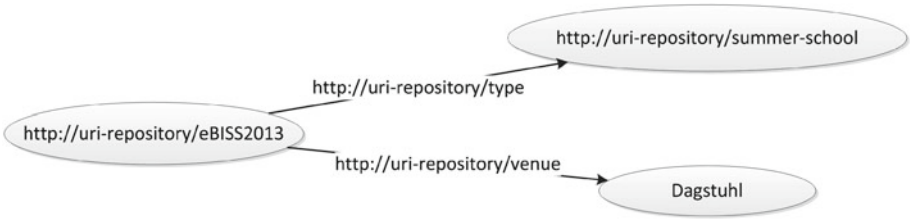


Fig. 2. A graph representing RDF triples

`uri-repository/type` property to another resource representing the summer school concept `http://uri-repository/summer-school`. Similarly, it is related to the literal *Dagstuhl* by the `http://uri-repository/venueproperty`. The resulting metadata can be seen as a graph where nodes are resources and literals, and edges are properties connecting them. RDFS extends RDF by allowing triples to be defined over classes and properties. In this way, we can describe the schema that rules our metadata within the same description framework. It is worth mentioning that the semantics of RDFS are based on type systems, similar to object-oriented formalisms and, for example, we can specify *classes*, *subclasses* and typed properties.

3.2 Ontology Languages

Two main families of logic-based languages currently underlie most of the research done in this direction: Description Logics (DL) and datalog-related logics (see [21] and [22], respectively). For example, OWL is founded in DL.

Both, DL and datalog, seek the same objective, but from different points of view. While DL focuses on representing knowledge, datalog is more focused on capturing the instances (and, in this sense, closer to the database field). As discussed in [23], both paradigms can be used to establish ontologies.

On the one hand, DL (or DL-based languages) assume a decentralized approach and information is stored separate from data. Thus, one talks about *terminology* and *instances asserted* (in terms of the terminology). DL also follow the open-world assumption and, accordingly, a DL ontology can have many different interpretations. On the other hand, datalog follows a centralized viewpoint, the closed-world assumption (there is a single interpretation which corresponds to the current database state) and the unique name assumption (two instances differing in their identifier are automatically assumed to be different).

A direct consequence is that DL ontologies are more difficult to model (as "unexpected information" could be inferred from the asserted instances) but they better deal with incomplete data (such as Web data), whereas datalog ontologies are more intuitive for the database community but might not be that interesting for integration cases with missing or partial information.

Modeling in DL and datalog deserve further discussion and basic knowledge on FOL. We address the interested reader to [24] for a discussion on the

expressivity of two popular ontology languages for data modeling: DL-Lite and Datalog±.

All in all, the open-world assumption in DL and the fact that these logics do not follow the unique name assumption suits the essence of Web data (by definition incomplete and where different repositories could identify the same real-world instance by means of different identifiers). From the point of view of BI, DL suits what-if analysis and scenarios with lack of information. In turn, scenarios where the data gathered is known to be complete (to some extent, what is stored in a DW) may be better captured in a datalog ontology.

Need vs. Feasibility Besides being very expressive, ontology language provide reasoning techniques to infer non-explicit knowledge from already asserted knowledge. Although logic-based languages are very appealing for their semantic-awareness and reasoning features, it is also true that reasoning is known to be computationally hard. Nowadays, it is well established that we must balance the language expressiveness and reasoning services provided according to each scenario.

The main reasoning services provided by DL are concept satisfiability, subsumption and query answering [21]. Concept satisfiability checks if a concept is non-contradictory (regarding the ontology terminology) and it may have, at least, one instance. For example, concept satisfiability (or unsatisfiability) is useful for validating the correctness of the ontology concepts. Subsumption checks if an ontology concept $\mathcal{C}$ is subsumed by another concept $\mathcal{D}$ (i.e., if $\mathcal{D}$ is more general than $\mathcal{C}$). For example, subsumption can be used to identify concept taxonomies and equivalence (if two concepts subsume each other). Finally, query answering finds all the asserted instances that satisfy a concept description and thus, it is extremely useful to pose arbitrary queries over the ontology.

Concept satisfiability and subsumption sit at the terminological level, whereas query answering also deals with instances. Relevantly, very few DL languages (e.g., DL-Lite in [25] and the OWL2 QL profile, based on DL-Lite) properly support query answering which means that, in practice, query answering is prohibitively costly for large data sets, such as those in BI scenarios. Thus, most DL languages are typically used at the terminological level.

Regarding datalog, since terminology and instances are not separated, its reasoning services are query-oriented and its most typical inference is query answering.

4 One Step towards Exploratory BI

Exploratory BI is, by nature, challenging. However, the current state of the art on BI systems envision a promising future. In this section, we present a functional architecture based on existing approaches towards the creation of a truly exploratory BI. Nonetheless, several challenges remain open but our aim is to show that exploratory BI is no longer a chimera but a feasible challenge.

In the recent past, we have been working on bringing new flexible and powerful BI capabilities to the end-user. Our focus has been set on accessing different data sources (inter / intra organizations / open linked data sources) in a flexible manner: the end-user poses his / her analytical needs and the system is able to produce data cubes on-demand. Ideally, the end-user should state his /her analytical needs and some quality criteria and receive the required data, while the inherent internal complexity of such system should be transparent to him / her. As consequence, the system would be responsible for identifying the data sources, extract and transform the data prior to show it to the user. Performing such tasks on the fly would be extremely costly and therefore, we propose here our vision of a self-tunable architecture that automatically performs optimization tasks to guarantee the feasibility of exploratory BI.

4.1 Narrowing the Focus: Assumptions Made

Before introducing our vision in detail, we need to narrow the focus and properly define what kind of BI systems we do tackle in this architecture. As the reader will note we talk about *cubes*, which is a multidimensional (MD) concept. The multidimensional model was introduced by Kimball [8]. Specifically, multidimensionality is based on the fact / dimension dichotomy. The fact, or subject of analysis is placed in the n-dimensional space produced by the analysis dimensions. We consider a dimension to contain an aggregation hierarchy of levels representing different granularities (or levels of detail) to study data, and a level to contain descriptors (i.e., level attributes). We differentiate between identifier descriptors (univocally identifying each instance of a level) and non-identifier. In turn, a fact contains analysis indicators known as measures (which, in turn, can be regarded as fact attributes). One fact and several dimensions conform a star-schema. Several star-schemas conform a constellation. A specific level of detail for each dimension produces a certain data granularity or data cube, in which place the measures. Thus, one can think of a cube as a query over a star-schema.

Despite its simplicity, the multidimensional schema has been shown to suit analytical tasks [26] and for this reason, we consider the multidimensional model as the *de facto* standard for BI data modeling. For modeling data flows, such as ETL processes, we do not use the multidimensional model (data-oriented) but BPMN (process-oriented). BPMN (Business Process Model and Notation)[10] is an OMG standard that has already been successfully applied to model BI processes in general, and ETL processes in particular [27]. BPMN is a graphical notation that provides constructs to control and manage data flows with enough detail as to be easily translated into an executable flow (for example, using BPEL, Business Process Execution Language).

All in all, the assumptions made in this architecture are as follows:

– A new generation data warehousing system is considered (i.e., we will talk about the integration layer and the ETL/ETQ layer).

[10] See http://www.bpmn.org/

– We consider the multidimensional model as the de facto data model for data analysis.
– We consider BPMN as the de facto data model for data workflows.

Obviously, other similar assumptions could be made for alternative solutions.

4.2 Functional Architecture

Figure 3 sketchs our proposal for an open-access semantic-aware platform for exploratory business intelligence.

First, note that our system is built following two principles previously discussed and motivated in this paper: open data and semantic-aware systems. We say our system is open-access because we aim at providing foundations for exploratory BI on top of freely available and accessible data sources, whereas it must be semantic-aware to enable automation. In our case, the common semantic framework is provided by a reference ontology.

In this system, the user is meant to interact by providing a *seed* or key concept for his / her analysis needs. This concept reaches the first module, AMDO (*Automating Muldimensional Design from Ontologies*) that looks for this concept in the reference ontology. Next, AMDO exploits the knowledge captured in the ontology to propose a list of potential facts, dimensions, measures and descriptors of interest related to the seed concept. This information is shown to the user in a comprehensive way. In short, most relevant concepts (according to some internal rules) are properly ranked and shown to the user, who selects those of his / her interest in a dynamic, interactive manner.

The choices made by the user out of AMDO's suggestions (from now on, the end-user requirement) are forwarded to the GEM module (*Generating ETL and Multidimensional Models*), which is responsible for producing the data cube design (both the conceptual data cube schema and the conceptual design of ETL

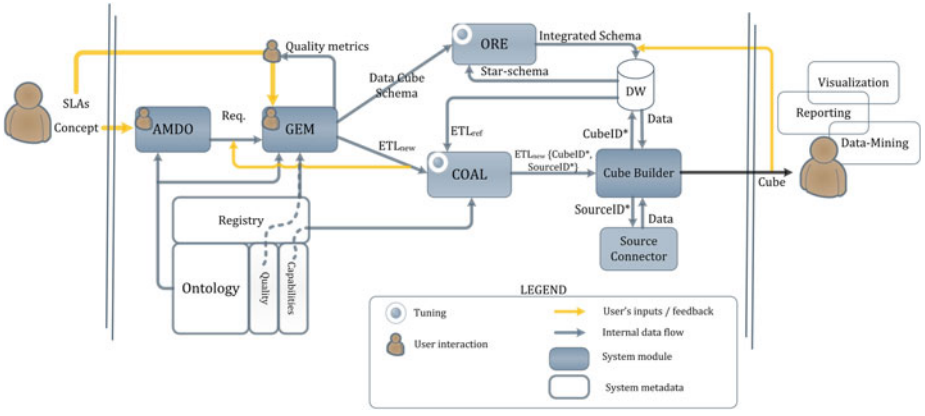


Fig. 3. An open access semantic-aware architecture for business intelligence

flows to provision the cube with data). As consequence, GEM needs to be aware of the end-user non-functional requirements (such as freshness or quality thresholds for the data retrieved) and the candidate sources from where to extract data. Non-functional requirements are expressed in terms of SLAs (service-level agreements), whereas data sources are identified from the registry module. In our system, any data source to be *federated* into the system must be properly registered. In our vision, we should follow a local-as-view (LAV) approach (similar to that followed by linked data) and thus, concepts in the data sources must point to the reference ontology (thus we also assume semantic-aware sources). Accordingly, when registering, the sources must *declare* what ontology concepts they refer to. From now on, we will refer to these ontology concepts as *linked concepts* (as, at least, one source is *linking* them). Furthermore, some quality criteria about the data source (again, as SLAs) must be provided and finally, the registry also must keep trace of the source technical capabilities (such as underlying technology -e.g., relational, triple-store, key-value-, query language, etc.), which will be needed to later query the source. GEM queries the registry to know what sources may provide the required data and the available sources are presented to the end-user together with the quality metrics (extracted from the SLAs) gathered for each of them.

As output, GEM designs a data cube schema and the corresponding ETL flows, as well as SLAs for the query at hand. Note, however, that some non-functional requirements may not be met and at this point the user would be prompted to relax, reinforce or disregard them. Next, the data cube schema is forwarded to the ORE module and the ETL flow to the COAL module. Now, COAL could query the sources and retrieve the needed data according to the ETL flow received. On the contrary, our system performs some optimizations to avoid data shipping (from the sources) whenever possible by materializing some queries in order to improve performance. The ORE module selects relevant pieces of information worth to materialize (e.g., if they are queried regularly) and iteratively consolidates them to produce a complete MD star-schema[11] (potentially, a constellation) subsuming all the cube schemas to be materialized so far. ORE's goal is to create the minimal MD design (e.g., by fusing adjacent facts and /or dimensions, hiding irrelevant concepts, etc.) meeting some tuneable quality criteria. Similarly, COAL is an incremental cost-based method for consolidating individual ETL designs into a unified ETL flow (from now on, the reference ETL) minimizing the execution cost. As result, ORE and COAL generate and maintain a data warehouse.

The system control flow goes to COAL once GEM has generated its outputs. There, COAL is responsible for deciding either to query the data sources or the data warehouse. Whenever possible, the latter will be prioritized. To do so, COAL checks if the new ETL at hand is subsumed by the data warehouse ETL flows. We say the new ETL is subsumed by the reference ETL if after

[11] Note we clearly differentiate between a data cube schema and a star-schema. The first one describes the schema of a query, whereas the second one describes a data warehouse schema that can answer many different queries.

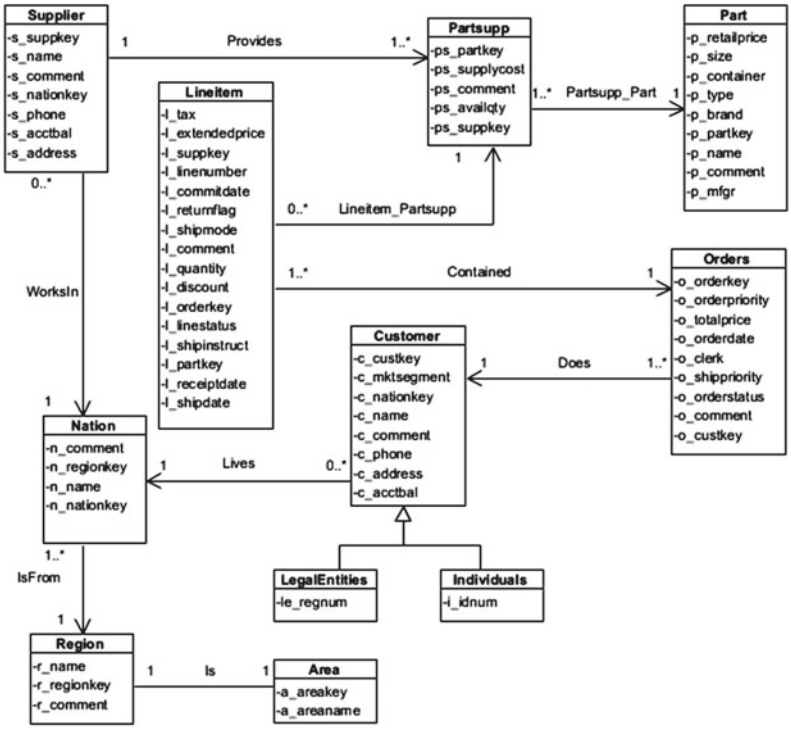


Fig. 4. TPC-H diagrammatic representation

consolidating both of them the output exactly coincides with the reference ETL. In other words, if no changes must be made in order to incorporate the new ETL. In practice, COAL does not look for a full matching but also partial matchings to identify what parts of the new ETL flow can be answered from the data warehouse and what parts must be queried from the sources (in the figure, this has been represented as a set of *cubeIDs* and *sourceIDs*). Intuitively, one may say that COAL tries to *rewrite* the new ETL in terms of the already available ETL flows.

Finally, the cube builder module can be thought as a query executor. It gathers the data retrieved from the sources and the data warehouse and, according to the ETL conceptual schema produced by GEM, builds the cube. If a source must be queried, the source connector triggers a query to retrieve the data according to the registered data source capabilities and capability-based optimization techniques [28]. Note the data sources are black-boxes for our system and, beyond ORE and COAL, we pass the responsibility for optimizing the query to the underlying system. To some extent, one may say ORE and COAL are responsible for the global optimization, whereas data sources are responsible for the local optimization of queries.

Once the cube has been shown to the user, the system (or the user) might decide to materialize it. Then ORE and COAL come into action to integrate the data cube at hand into the data warehouse. Obviously, any visualization or analytical tool might be used to further analyze the produced data cube from the end-user point of view.

Last, but not least, note some relevant features of our approach. According to Fig. 1, our ultimate goal essentially corresponds to an ETQ scenario, but the internal optimization techniques enable other data flows besides ETQ. Specifically, COAL and ORE correspond to the ETL arrow and the data warehouse repository respectively, whereas the ETL rewriting technique (i.e., the *CubeID* list produced by COAL) corresponds to the materialized data arrow. From the point of view of use cases, the user is meant to interact with AMDO and GEM, whereas the system administrator (the former DW designer) is meant to interact with ORE and COAL (for example, tuning the internal metrics used by ORE and COAL to consolidate the designs). Furthermore, both ORE and COAL generate and store valuable metadata of interest for the end user. For example, the data warehouse MD schema and ETL processes can be retrieved and visualized. This feature has many applications and, for example, the ETL flows can be used to tackle traceability and show what sources were considered, what transformations applied and how it was combined and visualized as a data cube, whereas the star-schemas can be used to tackle collaborative BI and query recommendation (based on past evidences).

4.3 A (Toy) Usage Example

Prior to detailing each of the modules contained in our system (see next sections), we introduce an example to show how such a system would work from the point of view of the end-user. For this example we will consider the TPC-H benchmark [29]. In our approach, we need an ontology (our common semantic framework) whose diagrammatic representation is depicted in Fig. 4. Suppose now an end-user interested in *orders lineitems*. It may start dropping a query by writing the word *lineitem* in the GUI. Immediately, AMDO looks for this word in the system reference ontology and proposes a set of dimensions, facts and measures of potential interest. For example, let assume the output shown in Table 1 (in brackets, the relevance computed by AMDO).

Table 1. An Example of AMDO's outputs

Proposed fact: *lineItem* (100 %)

Measures	Dimensions
ExtendedPrice (100 %)	Orderdate (93 %)
Quantity (95 %)	Shipdate (90 %)
Discount (80 %)	Nation (of the customer) (87 %)
...	Supplier (83 %)
	...

Usually, AMDO will propose different facts (not only one) ranked by relevance (see Sect. 5.1 for further details) and for each fact it will propose different measures and dimensions. Similarly to most BI dashboards, the user may drag and drop the concepts to build the desired cube schema. For example, suppose the user chooses *extendedPrice* and *discount* as measures and the *customer nation* as single dimension. At this point, AMDO will allow the user to add slicers (e.g., *nation = 'Serbia'*) and specify derivation functions (e.g., *price = extendedPrice*(1−discount)*) and aggregation functions (e.g., compute the *average price*). Once done, GEM creates the data cube schema. For example, consider the example depicted in Fig. 5. GEM identifies the ontology subset (bolded in colours) needed to satisfy the end-user requirement forwarded from AMDO (in words, *the average price paid* (measure) *by Serbian customers* (dimension)). For further information on GEM see Sect. 5.2 but note that additional ontology concepts (e.g., *orders* in our running example, which is bolded in green), not chosen by the end-user, may be needed to properly relate the concepts at hand (bolded in orange) and produce a single data cube. Once the schema has been created, GEM looks for those registered sources *linked* to the ontology concepts participating in the cube. These sources are presented to the user together with the data source quality evidences registered as SLAs. Suppose now three sources: A, B and C, but B presents some quality issues: its servers are frequently down and data provided is generated by a small community that cannot guarantee its correctness. For this reason, the user decides not to consider B and proceed with the other two sources. Next, GEM designs the needed ETL processes to provision the data cube schema with data from the selected sources.

Next, COAL is launched to check what parts of the new ETL flow can be rewritten in terms of the data warehouse ETL flows and what others need to query the data sources. Suppose we need to query a source. Then, according to the metadata regarding A and C it asks the source connector to wrap a query to obtain the needed data. For example, if A is a relational database that contains a table `M(orderkey, partkey, suppkey, extendedPrice, discount, ...)` (like the one provided in the TPC-H schema for *lineitem*) and we want to obtain the measures needed from it, it would trigger an SQL query such as `SELECT orderkey, partkey, suppkey, extendedPrice, discount FROM M` (where the set {`orderkey, partkey, suppkey`} is considered to be the table primary key). Data gathered from the data warehouse and the data sources is properly assembled according to the ETL flow created by GEM.

Once the data cube is ready any visualization or analytical tool can be used to show it to the user. If the user eventually decides to integrate this data cube into the data warehouse, our system would launch first ORE and then COAL to perform the needed evolution tasks in the data warehouse (further details in the next section).

Finally, suppose now that we decided to materialize the current query and, in the future, the user is interested in the *discount* (measure) obtained for each *customer* (dimension). Clearly, this query can be rewritten in terms of the inputs needed for the query discussed above. Thus, the COAL module will use

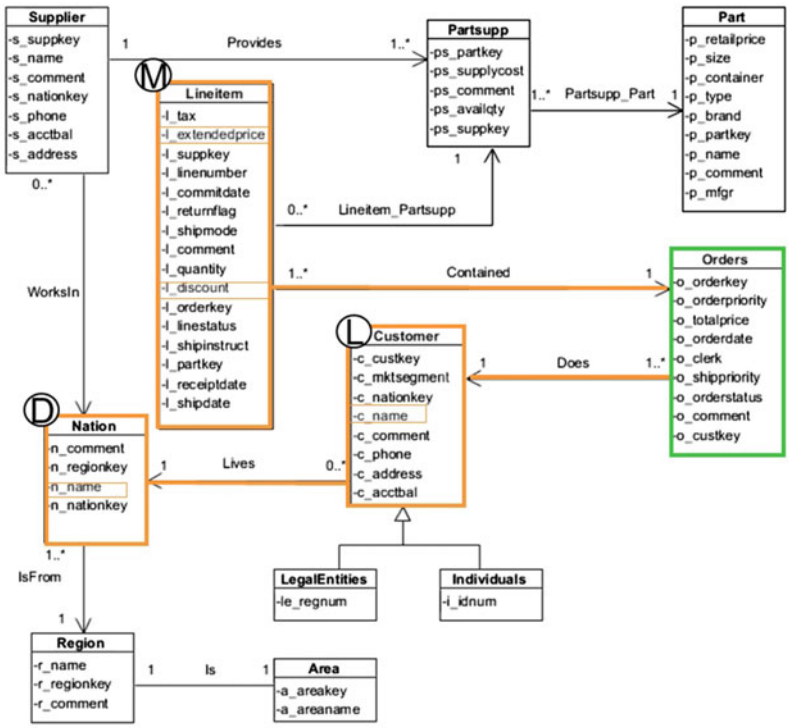


Fig. 5. GEM example

equivalence rules to rearrange the ETL operations in both flows (always preserving the semantics) and maximize the overlapping area. In this case, the new ETL is completely subsumed by the reference one and therefore, COAL will find a full match (i.e., complete overlapping). See Sect. 5.4 for further details.

5 An Open-Access Semantic-Aware System

Our system sets a common semantic framework by means of an ontology. Note that, up to now, we have assumed a single reference ontology. However, in practice, several ontologies may co-exist. Ontology matching techniques can be used to combine such ontologies and, in the end, the more inter and intra-relationships captured, the better. Regarding the sources, by now, we only assume semantic-aware repositories (e.g., linked data). Nevertheless, it might be possible to wrap other sources with additional semantics and enable their integration into our system. For example, tools like *Triplify*[12] can be used to semi-automate such task. This opens the door for a polyglot system consisting of heterogeneous data sources.

[12] See http://triplify.org/Overview

In the rest of this section we will focus on the main modules of our proposal (i.e., AMDO, GEM, ORE and COAL) and we present them in more detail.

5.1 The AMDO Module

AMDO [30] looks for ontology concepts that together with the seed concept (provided by the end-user) may produce meaningful data cubes. In practice, it looks for concepts likely to play a valid multidimensional role (with regard to the seed concept). Specifically, dimensions arrange the multidimensional space where the fact of study is depicted. Each instance of data is identified (i.e., placed in the multidimensional space) by a point in each of its analysis dimensions. Conceptually, it implies that a fact must be related to each analysis dimension (and by extension, to any dimensional concept) by a many-to-one relationship. That is, every instance of the fact is related to, at least and at most, one instance of an analysis dimension, and every dimension instance may be related to many instances of the fact.

AMDO looks for potential facts, dimensions, measures and descriptors by means of *topological patterns* guaranteeing the MD constraints above discussed. Importantly, AMDO does not perform a blind search but a guided search from the seed concept. Furthermore, some internal metrics are used to rank concepts found. For example, a fact containing many measures and dimensions, closeness to the seed concept, etc.

Internally, AMDO exploits standard reasoning services to compute the topological patterns.

5.2 The GEM Module

GEM [31] receives the facts, dimensions, measures and descriptors identified by AMDO, which we will refer to as the input requirement from now on.

The process of creating the MD and ETL designs for the input requirement is a semi-automatic process comprising four main stages (see Fig. 6). The outcome of each stage is validated and then, either it is propagated to the next stage or undergoes a correction process. The correction process may be done automatically, it may suggest changes, or it may require user feedback.

Stage 1: Requirement verification. First, the system checks if there is a mismatch among the input requirement and the ontology linked concepts. For each concept in the input requirement, GEM checks if, at least, there is a source linked to it. In case of mismatch, it may suggest relaxation of the requirement or alternatives (e.g., choosing subclasses).

Stage 2: Requirement completion. After mapping the input requirement onto the ontology and verifying it, the system complements it with needed additional information. This stage identifies *intermediate concepts* that are not explicitly stated in the business requirement, but are needed in order to retrieve data. Intuitively, it identifies all the ontology concepts needed to eventually build

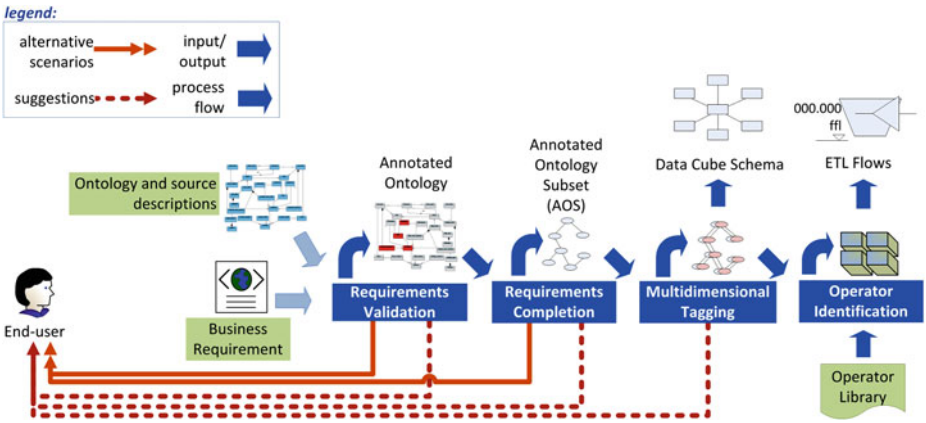


Fig. 6. GEM in a Nutshell

a *query* (or access plan) to answer the requirement. User feedback is welcomed for ensuring correctness and compliance to the end-user needs in case of ambiguity.

Stage 3: Multidimensional verification. Next, we look for a MD interpretation of the ontology subset identified in the previous stage. To do so, we check the MD integrity constraints and verify the correctness of the requirement according to MD design principles. Hence, we check two issues: (i) whether the factual data is arranged in a MD space (i.e., it forms a data cube and thus, if each instance of factual data is identified by a point in each of the analysis dimensions [32]); and (ii) whether data summarization performed is correct by examining whether the following conditions hold [33]: (a) *disjointness* (the sets of objects to be aggregated must be disjoint); (b) *completeness* (the union of subsets must constitute the entire set); and (c) *compatibility* of the dimension, the type of measure being aggregated and the aggregation function.

Stage 4: Operator identification. The ETL operations are identified in three phases. First, we use the annotations generated by the previous steps for extracting schema modification operations. Then, the cubes are built. And finally, we complement the design with additional information that might be found in the sources and with typical ETL operations such as surrogate key and slowly changing dimensions. Similar to Stage 1, once the sources have been identified, this step looks for mismatches between the end-user and data source SLAs and may suggest alternatives or relaxation of some non-functional requirements.

5.3 The ORE Module

ORE [34] is responsible for integrating new data cube schemas into the data warehouse. It comprises four stages, namely *matching facts*, *matching dimensions*, *complementing the MD design*, and *integration* (see Fig. 7). The first three

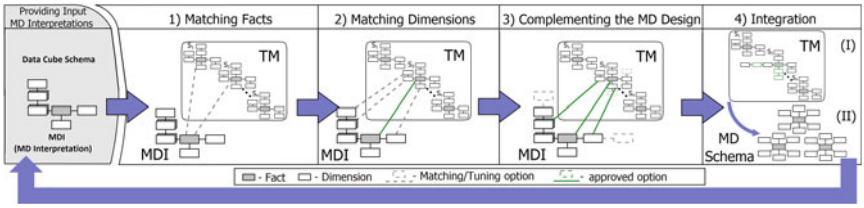


Fig. 7. ORE in a Nutshell

stages gradually match different MD concepts and explore new design alternatives. The last stage considers these matchings and designer's feedback to generate the final MD schema that accommodates a new information requirement.

In all stages, we keep and maintain a structure, namely *traceability metadata* (TM), for systematically tracing everything we know about the MD design integrated so far, like candidate improvements and alternatives. With TM, we avoid overloading the produced MD schema itself.

Stage 1: Matching facts. We first search for different possibilities of how to incorporate the data cube schema at hand (i.e., the MD interpretation produced by GEM) to TM. The matching between factual concepts is considered –i.e., the system searches the fact(s) of TM producing an equivalent set of points in the MD space– as the one in the given MD interpretation. Different possibilities to match the factual concepts results with the appropriate sets of integration operations. The costs of these integration possibilities are weighted and prioritized.

Stage 2: Matching dimensions. After matching facts, we then conform the dimensions of the paired facts. Different matchings between levels are considered (i.e., "=", "1 - 1", "1 - *" and "* - 1") and thus, the different valid conformation possibilities are obtained. With each possibility, a different set of integration operations for conforming these dimensions is considered and weighted.

Stage 3: Complementing the MD Design. We further explore the reference ontology and search for new analytical perspectives related to the new concepts. Different options may be identified to extend the current schema with new levels, descriptors, and measures). The user is then asked to (dis)approve the integration of the discovered concepts into the final MD schema.

Stage 4: Integration. The MD schema is finally obtained in two phases of this stage. First, possible groupings of the adjacent concepts containing equivalent MD knowledge is identified to minimize the MD design. Finally, the final MD schema is produced by folding and collapsing grouped concepts to capture only the minimal information relevant to the user. Nevertheless, the complete TM is still preserved in the background to assists further integration steps (e.g., to handle future evolution events).

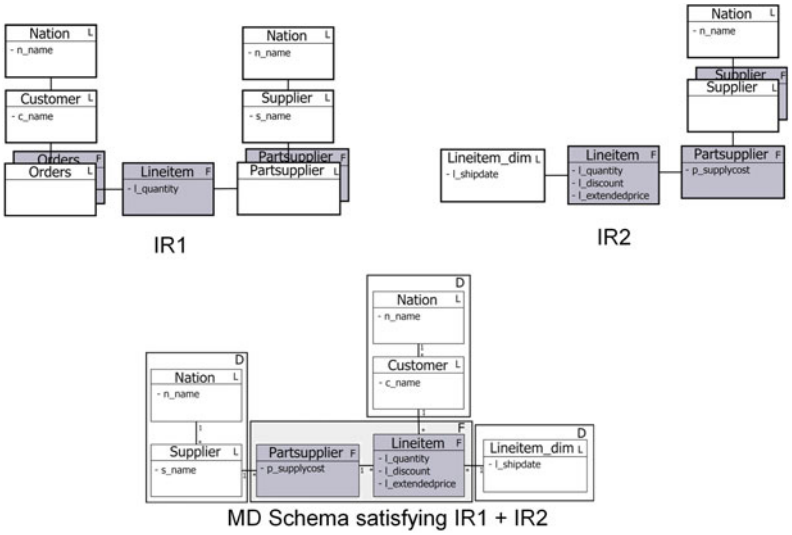


Fig. 8. ORE example

Example: The general idea behind ORE can be visualized in the example depicted in Fig. 8. This figure shows two outputs produced by GEM for two given requirements (IR1 and IR2) over the TPC-H example. As output, ORE will produce an integrated schema meeting the two cube-schemas of the requirements at hand.

5.4 COAL

COAL [35] is responsible for integrating the new ETL flow at hand with the data warehouse ETL flows. Typically, an ETL design is modeled as a directed acyclic graph. The nodes of the graph are data stores and operations, and the graph edges represent the data flow among the nodes. Intuitively, for consolidating two ETL designs, a referent G_1 and a new G_2 designs, we need to identify the maximal overlapping area in G_1 and G_2. However, this is not a typical graph-matching problem, as the MD interpretation of the ETL flow must be preserved. Therefore, we proceed as follows. First, we identify the common source nodes between G_1 and G_2. For each source node, we consider all paths up to a target node and search for common operations in both designs. In these paths, we search for common operations that could be consolidated into a single operation in the resulting design.

Deciding what operations can be consolidated and how is not an easy task. If two operations, each placed in a different design, can be matched, then we have a *full match* (e.g., the very same or equivalent operations). If two operations, one in the reference design and the other in the new design, partially overlap, then we have a *partial match* (e.g., one operation subsuming the other).

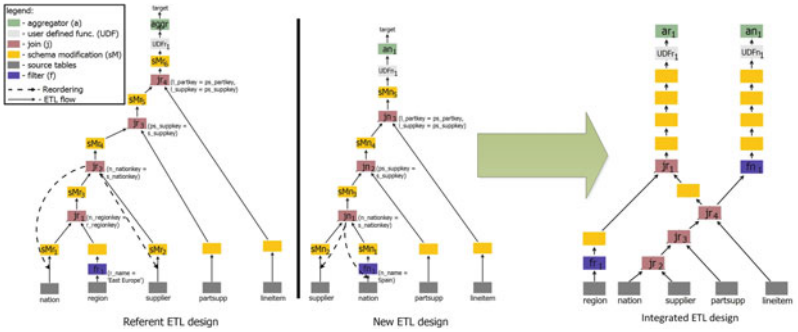


Fig. 9. COAL example

To detect the maximum number of full and partial matchings, we should also look at the operations performed before the ones considered for the matching; i.e., COAL must consider restructuring both designs by moving operations to maximize the number of overlapping operations. Restructuring the ETL designs must be performed by guaranteeing the same semantics as result and this is achieved by means of a set of predefined equivalence rules between operations (e.g., selections can always be pushed down a join operation, but a selection cannot be pushed down a projection if the selection attribute coincides with the projected attributed).

Example: The general idea behind COAL can be visualized in the example depicted in Fig. 9. There, a new and a reference ETL flow are presented. In general, these two designs may have a number of common operations. COAL's internal algorithms look for a minimal resulting integrated ETL such that common operations are executed once and the semantics of both flows are preserved by means of the above discussed equivalence rules. What minimal exactly means depends on the quality criteria determined by the system administrator.

5.5 Open Questions

In our functional architecture we have shown the feasibility of some of the challenges behind exploratory BI (see Sect. 2.1). However, several challenges still remain open for exploratory BI and deserve further attention. More specifically, the most important ones can be summarized as follows:

- **Integration of schemas:** In practice, it is unfeasible to assume that a single common semantic framework (where any potential data source of interest is mapped) does exist. Indeed, several reference languages may co-exist and automatic mappings should be discovered (ideally, by means of reasoning).
- **LAV vs. GAV:** Clearly, a LAV approach is desired for mapping the data sources to the semantic framework but nowadays most approaches follow a GAV approach (i.e., the mappings are in the ontology concepts). Further research on semantic-aware formalisms to allow LAV is desirable.

- **Reasoning:** Tightly related to the previous item, computationally feasible reasoning techniques are needed in order to infer knowledge not explicitly stated in the reference layer. In our work, we used the DL-Lite family [25], which provides a good trade-off between expressivity (similar to that of UML) and computational complexity. However, only basic inference algorithms (such as subsumption) were feasible in practice. Others, like computing the transitive functional closure, needed ad-hoc algorithms to be computed. Thus, creating reasoning facilities over DL / datalog families especially designed to capture the multidimensional model is a must. In this sense, how to exploit parallelism and benefit from distributed computation when computing reasoning is also an open challenge.
- **ETL Operators:** The ETL flows automatically generated in our approach mainly consider the relational algebra and a bunch of additional operations (create surrogates, dictionary look-ups, etc.). Ideally, any transformation should be able to be specified and automatically considered by our framework in an automatic way (right now, the ETLs produced need to be manually enriched).
- **Non-Functional Requirements:** How to specify non-functional requirements in a machine readable format and include them all over the process is still also an open challenge. Traditionally, non-functional requirements are considered and the database is correspondingly tuned by database administrators.

6 Semantic Aware Business Intelligence: State of the Art

In the recent past there has been a bloom of new techniques and methods for BI relying on semantic-aware formalisms. Semantic-aware data warehouses are nowadays hot topics of research (e.g., among many others [36–39]). These works can be understood as the cog wheels forming the exploratory BI machine and they span from requirement engineering, conceptual design to physical design in many and disparate areas. Surveying all these works is completely unfeasible and, for this reason, in this section, we will focus on those approaches presenting similar systems to what we have called exploratory BI (i.e., the big picture) and how they propose to combine different techniques to produce similar systems. Indeed, in the literature we can find several equivalent or similar terms to exploratory BI. For example, "live BI" [15], "on-demand BI" [16], "ad-hoc BI" [17], "open BI" [18], "situational BI" [19], or "fusion cubes" in [20].

Mazón et al. [18] presents a platform to analyze linked open data and, similar to our approach, they assume semantic-aware sources. Data modeling and provisioning are achieved by means of a traditional data warehouse architecture (which is loaded with data from the sources) following model-driven techniques and their focus is on providing advanced support to non-technical users to trigger data mining algorithms over the gathered data. To support non-expert data mining users, a knowledge base conforming quality criteria of the sources is created and used as main source to recommend the user data explorations.

Essaidi [16] presents *on-demand* BI services and adopt models related to Software-as-a-Service (SaaS) as technical solution. Their approach consists of a multi-layered architecture to support different business intelligence needs, namely: the designer tasks (designing the environment, perform data integration and manage metadata) based on a model-driven approach and the end-user analytical needs. They add a third tier of services related to security (which includes authorities, roles, users, groups and grants). From the point of view of BI, this solution embeds and supports traditional BI into a SaaS architecture.

Berthold et al. [17] focuses on the technical challenges of *ad-hoc* BI, namely, a global business data model, data source integration and enrichment (in which they distinguish between business configuration at design time, and data provisioning at run time) and finally, support for ad-hoc (self-oriented) and collaborative BI.

Castellanos et al. [15] presents a unified data management and analytics platform for *live* BI. This paper presents a flexible architecture that allows to specify and define ETL flows from highly heterogeneous sources. They introduce the concept of extraction operators so that unstructured or semi-structured data can be extracted and integrated with structured data. In this paper, the data flow is defined in terms of a pipeline transforming input streams into analytics results. It is presented in terms of an event discovery stage (based on historical evidences) from the input stream and a second stage of further analysis of the events to detect and predict non-explicit patterns. Special emphasis is put on the need of a powerful physical design and optimizer that could deal with heterogeneous and mixed data flows.

Löser et al. [19] presents a platform to correlate data from an organization data warehouse with external sources. This platform is database-inspired and enables traditional BI queries (ad-hoc and aggregate queries) over cloud architectures. Their approach consists of a common algebraic core to describe, plan, optimize and execute queries, and similar to the previous approach, they focus on unstructured sources and how to extract and integrate data from them. Also, an optimizer and a parallel executor engine are introduced.

Finally, [20] envisions a highly heterogeneous BI system where a query is formulated, then relevant sources are discovered and selected and data provisioning and integration flows are triggered before presenting the resulting data to the user. An abstract architecture is also presented and data from external and internal sources is ETL-ed into *stationary* (i.e., pre-defined cubes) and *fusion cubes* (similar to the ETQ term in this paper) before querying them. Different from the rest of approaches, this is a visionary paper presenting future research trends and priorities for data modeling and provisioning for BI processes but no specific solution is discussed.

All in all, these approaches present a similar ultimate goal and some common trends can be identified. For example, they agree on the need to make transparent to the end-user all the technical complexity behind advanced BI solutions and provide flexible and intuitive conceptual formalisms in order to allow end-users specify and design their queries. These solutions mainly differ though in the

assumptions made. We can identify a key assumption that conditions the solution presented: loosely coupled - tightly coupled data sources. In this aspect, we can find a very wide range of solutions but the consequences are clear: the more tight the data sources are the better internal optimizations can be made.

Regarding our approach, we present an *instance* of the abstract architecture envisioned in [20] and use a reference ontology to relate loosely coupled sources and, at the same time, enable some global optimizations. Furthermore, special emphasis is put on automatic MD discovery, design and deployment.

7 Conclusions

In this paper we have motivated the need for new generation BI systems and coined the exploratory BI term, which should enable end-users to trigger right-time analytical tasks over disparate and heterogeneous sources. The challenges behind exploratory BI are manifold, but we have focused on two key aspects: open-access (following the open data movement) and semantic-aware repositories to enable automation.

As a proof of concept, we have sketched the architecture of an open-access semantic-aware system to support exploratory BI and highlighted the main areas of research for enabling exploratory BI.

References

1. Lenzerini, M.: Data integration: a theoretical perspective. In: Popa, L., Abiteboul, S., Kolaitis, P.G., (eds.) PODS, pp. 233–246. ACM (2002).
2. Bukhres, O.A., Elmagarmid, A.K. (eds.): Object-Oriented Multidatabase Systems: A Solution for Advanced Applications. Prentice-Hall, Engle- wood Cliffs (1996)
3. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 3rd edn. Springer, New York (2011)
4. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intell. Syst. **21**(3), 96–101 (2006)
5. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J. Semant. Web Inf. Syst. **5**(3), 1–22 (2009)
6. Eberius, J., Thiele, M., Braunschweig, K., Lehner, W.: Drillbeyond: enabling business analysts to explore the web of open data. PVLDB **5**(12), 1978–1981 (2012)
7. Golfarelli, M., Rizzi, S.: Data Warehouse Design: Modern Principles and Methodologies, 1st edn. McGraw-Hill, New York (2009)
8. Kimball, R., Reeves, L., Thornthwaite, W., Ross, M.: The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing. Developing and Deploying Data Warehouses. Wiley, New York (1998)
9. Giorgini, P., Rizzi, S., Garzetti, M.: Grand: a goal-oriented approach to requirement analysis in data warehouses. Decis. Support Syst. **45**(1), 4–21 (2008)
10. Mazon, J.N., Trujillo, J., Lechtenborger, J.: Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. Data Knowl. Eng. **23**(3), 725–751 (2007)

11. Jensen, C.S., Pedersen, T.B., Thomsen, C.: Multidimensional Databases and Data Warehousing. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, San Rafeal (2010)
12. Winter, R., Strauch, B.: A method for demand-driven information requirements analysis in DW projects. In: Proceedings of 36th Annual Hawaii International Conference on System Sciences, pp. 231–239. IEEE (2003).
13. Golfarelli, M., Rizzi, S., Turricchia, E.: Modern software engineering methodologies meet data warehouse design: 4WD. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 66–79. Springer, Heidelberg (2011)
14. Wrembel, R.: A survey of managing the evolution of data warehouses. IJDWM **5**(2), 24–56 (2009)
15. Castellanos, M., Dayal, U., Hsu, M.: Live business intelligence for the real-time enterprise. In: Sachs, K., Petrov, I., Guerrero, P. (eds.) From Active Data Management to Event-Based Systems and More. LNCS, vol. 6462, pp. 325–336. Springer, Heidelberg (2010)
16. Essaidi, M.: ODBIS: towards a platform for on-demand business intelligence services. In: Proceedings of the ACM International Conference on EDBT/ICDT Workshops. ACM (2010).
17. Berthold, H., Rösch, P., Zöller, S., Wortmann, F., Carenini, A., Campbell, S., Bisson, P., Strohmaier, F.: An architecture for ad-hoc and collaborative business intelligence. In: EDBT/ICDT Workshops (2010).
18. Mazón, J.N., Zubcoff, J.J., Garrigós, I., Espinosa, R., Rodríguez, R.: Open business intelligence: on the importance of data quality awareness in user-friendly data mining. In: EDBT/ICDT Workshops, pp. 144–147. ACM (2012).
19. Löser, A., Hueske, F., Markl, V.: Situational business intelligence. In: Castellanos, M., Dayal, U., Sellis, T. (eds.) Business Intelligence for the Real-Time Enterprise. Lecture Notes in Business Information Processing, vol. 27, pp. 1–11. Springer, Berlin Heidelberg (2009)
20. Abelló, A., Darmont, J., Etcheverry, L., Golfarelli, M., Mazón, J.N., Naumann, F., Pedersen, T.B., Rizzi, S., Trujillo, J., Vassiliadis, P., Vossen, G.: Fusion cubes: towards self-service business intelligence. Int. J. Data Warehouse. Min. **9**(2), 66–88 (2013)
21. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York (2003)
22. Ceri, S., Gottlob, G., Tanca, L.: What you always wanted to know about datalog (and never dared to ask). IEEE Trans. Knowl. Data Eng. **1**(1), 146–166 (1989)
23. Patel-Schneider, P.F., Horrocks, I.: Position paper: a comparison of two modelling paradigms in the semantic web. In: 15th International Conference on World Wide Web (WWW), pp. 3–12. ACM (2006).
24. Calì, A., Gottlob, G., Lukasiewicz, T.: Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In: Fagin, R. (ed.) ICDT. ACM International Conference Proceeding Series, vol. 361, pp. 14–30. ACM, New York (2009)
25. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-lite family and relations. J. Artif. Intell. Res. **36**, 1–69 (2009)
26. Romero, O., Marcel, P., Abelló, A., Peralta, V., Bellatreche, L.: Describing analytical sessions using a multidimensional algebra. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 224–239. Springer, Heidelberg (2011)

27. El Akkaoui, Z., Mazón, J.-N., Vaisman, A., Zimányi, E.: BPMN-based conceptual modeling of ETL processes. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 1–14. Springer, Heidelberg (2012)
28. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book. Pearson Prentice Hall, Upper Saddle River (2008)
29. TPC: TPC-H specification (2012). http://www.tpc.org/tpch/
30. Romero, O., Abelló, A.: A framework for multidimensional design of data warehouses from ontologies. Data Knowl. Eng. **69**(11), 1138–1157 (2010)
31. Romero, O., Simitsis, A., Abelló, A.: GEM: requirement-driven generation of ETL and multidimensional conceptual designs. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 80–95. Springer, Heidelberg (2011)
32. Mazón, J., Lechtenbörger, J., Trujillo, J.: A survey on summarizability issues in multidimensional modeling. In: DKE, pp. 1452–1469 (2009).
33. Lenz, H., Shoshani, A.: Summarizabilty in OLAP and Statistical Data Bases. In: SSDBM, pp. 132–143 (1997)
34. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: ORE: an iterative approach to the design and evolution of multi-dimensional schemas. In: Song, I.Y., Golfarelli, M., (eds.) DOLAP, pp. 1–8. ACM (2012).
35. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: Integrating ETL processes from information requirements. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 65–80. Springer, Heidelberg (2012)
36. Pérez, J.M., Llavori, R.B., Aramburu, M.J., Pedersen, T.B.: Integrating data warehouses with web data: a survey. IEEE Trans. Knowl. Data Eng. **20**(7), 940–955 (2008)
37. Nebot, V., Llavori, R.B.: Building data warehouses with semantic web data. Decis. Support Syst. **52**(4), 853–868 (2012)
38. Spaccapietra, S. (ed.): Journal on Data Semantics XII. LNCS, vol. 5480. Springer, Heidelberg (2009)
39. Gallinucci, E., Golfarelli, M., Rizzi, S.: Meta-stars: multidimensional modeling for social business intelligence. In: Proceedings of DOLAP 2013, pp. 11–18. ACM (2013).

Transparent Forecasting Strategies in Database Management Systems

Ulrike Fischer(✉) and Wolfgang Lehner

Technische Universität Dresden, Database Technology Group, Dresden, Germany
{ulrike.fischer,wolfgang.lehner}@tu-dresden.de

Abstract. Whereas traditional data warehouse systems assume that data is complete or has been carefully preprocessed, increasingly more data is imprecise, incomplete, and inconsistent. This is especially true in the context of big data, where massive amount of data arrives continuously in real-time from vast data sources. Nevertheless, modern data analysis involves sophisticated statistical algorithm that go well beyond traditional BI and, additionally, is increasingly performed by non-expert users. Both trends require transparent data mining techniques that efficiently handle missing data and present a complete view of the database to the user. Time series forecasting estimates future, not yet available, data of a time series and represents one way of dealing with missing data. Moreover, it enables queries that retrieve a view of the database at any point in time — past, present, and future. This article presents an overview of forecasting techniques in database management systems. After discussing possible application areas for time series forecasting, we give a short mathematical background of the main forecasting concepts. We then outline various general strategies of integrating time series forecasting inside a database and discuss some individual techniques from the database community. We conclude this article by introducing a novel forecasting-enabled database management architecture that natively and transparently integrates forecast models.

1 Introduction

We can observe the transition of traditional data warehouse systems to big data stores where massive amount of data arrives continuously in real-time from vast data sources. Whereas traditional systems assume that data is complete or has been carefully preprocessed using ETL tools, this is not true any more in the context of big data and real-time requirements. Nowadays, data is increasingly characterized by incompleteness, inconsistency, and imprecision. Nevertheless, we can observe the requirements for sophisticated adhoc queries that extract higher-level information out of the vast and incomplete data sets. This leads to tremendous challenges of dealing with missing (past, present, and future) data that arrives at a later point in time or might even never be available.

In fact, we require data mining techniques that fill in such non-existent data. Various techniques from the statistical field aim at dealing with different kinds of incomplete data, for example:

E. Zimányi (Ed.): eBISS 2013, LNBIP 172, pp. 150–181, 2014.
DOI: 10.1007/978-3-319-05461-2_5,

- *Forecasting* estimates future, not yet available, data of a time series.
- *Imputation* replaces missing data with substituted values based on similar values observed in the same data set.
- *Interpolation* estimates a function between known data points to construct new data points.
- *Extrapolation* estimates the characteristics of a whole population based on the selection of a subset of individuals.
- *Recommendation* provides user ratings of items that have not been rated yet.

In the past, these techniques were often performed by highly qualified statistical experts, with long experience in the company, who manually experiment with different algorithms and parametrization. Such manual approaches are infeasible or even impossible for large data sets that grow and evolve at a rapid pace. Moreover, the non-expert user does not care about advanced data mining techniques. The user expects a complete view of the data set at any point in time, independent of its actual characteristics. Along with the traditional ANSI/SPARC architecture of a DBMS, which has the goal to separate a users' view of the data from its physical representation, incomplete data should be handled transparently to the user by the underlying database system. As a consequence, data mining techniques dealing with incomplete data should be seamlessly integrated into the existing infrastructure of a database management system, maintaining the declarative interface of a DBMS.

For example, assume a market research company that collects sales data according to various retailers and product lines. Retailers communicate their data to the market research company in regular time intervals. In this context, it happens quite regularly that retailers fail to deliver data, provide incomplete data or delay data delivery. Suppose a decision manager wants to create an aggregated report of products sold between yesterday and tomorrow:

```
SELECT pname, time, SUM(salesunits)
FROM facts f, products p
WHERE f.product_id = p.product_id
AND time in (yesterday(), tomorrow())
GROUP BY pname, time
```

Within this time interval, some stores might have failed to deliver their sales data, so here an interpolation model could fill in such missing data. Alternatively or additionally, an imputation model can be used to derive missing data from similar stores. Finally, the data for tomorrow is not available at all, therefore, forecast models can be used to estimate future data. All these models are created and transparently processed by the query engine, the user just receives a traditional relational table containing the complete result set.

In fact, as most mining techniques are based on some kind of statistical model, this will lead to a *model-based database system*, where incomplete, inconsistent or imprecise data is represented trough statistical models. New query processing, optimization, and execution techniques are required that work with models instead of real data. Besides query processing, such a model-based database system also impacts the design of a database to select and parametrize models that

allow efficient query processing as well as accurate query results. Finally, new data has to be efficiently incorporated into the existing model configuration.

The design of such a model-based database system opens up many challenges. We have to deal with large amount of data incorporating real-time data streams from many data sources. There are loads of existing statistical models, ranging from very simple to rather complex ones with many parameter and tuning opportunities. Both aspects, data size and parameter possibilities, pose high challenges to the design of such a model-based database system. In addition, data characteristics as well as query workloads are rapidly changing requiring a self-adaptive and self-tuning approach.

Along with this overall goal of designing a model-based database system, in this article, we turn our attention to one specific statistical technique, which is time series forecasting. Hence, we are only interested in providing future, not yet available data of a time series. Hereby, we focus on integrating *forecast models* into a database management system in order to *transparently* support queries on a future time interval, i.e., *forecast queries*.

The remainder of this article is organized as follows: We first outline the main challenges of time series forecasting using three application examples (Sect. 2). We continue by discussing the mathematical foundations of forecasting, including frequently used statistical methods in this area (Sect. 3). We then dive into technical aspects of extending database systems and give first of all a high level overview of integrating statistical methods, but not necessarily time series forecasting, into database management systems (Sect. 4). Subsequently, in Sect. 5, we review specific database techniques that explicitly address forecasting of time series data. Based on the discussion of existing work, in Sect. 6, we introduce a novel forecasting-enabled database management system that aims to fully and transparently integrate time series forecasting within a DBMS. We finally conclude in Sect. 7 and outline some further research challenges.

2 Forecasting Applications

Time series data appears in numerous domains and often forecasting of such data is required for planning and decision making processes. In this section, we discuss the characteristics of time series forecasting on three selected application areas, namely production planning, energy load balancing, and online display advertisement.

2.1 Production Planning

Typically, large volumes of historical sales data is stored in data warehouse systems and collected according to various characteristics of products, stores, and customers. Often a multidimensional data model is used for such kind of applications where different facts (often called measures) are organized along several dimensions [23]. The measures within a dimension are further divided into hierarchies to support multiple granularities. In this context, analytical database

queries are not interested in single measures but in some form of summarized data (e.g., sales in a certain area). The dimension hierarchies provide the key navigation paths for interactive OLAP (Online Analytical Processing) on the data, allowing for meaningful query formulation via drill-down, roll-up, or slice-and-dice operations [47]. Besides querying historical data, forecasts of sales figures form the basis for planning in many commercial decision-making-processes in logistics and supply chain management.

According to Mentzer and Bienstock [57], a sales forecasting system should follow several principles. First, sales forecasts should be provided in a central system and tightly coupled with the database management system, allowing fast access by various departments, such as production, distribution, and marketing. Second, forecasts have to be available for various horizons (short-, mid-, and long-term) and hierarchical levels, depending on the company's needs. For example, supply chain managers require long-term sales forecast to plan production and storage facilities, whereas short-term forecasts are required for timely transportation decisions. Third, the complexity of forecasting should be hidden from a decision manager, who is usually not an expert in the statistical area. Forecast results need to be provided in an easy-to-use format. Fourth, a sales forecasting system should include a suite of time series techniques and provide a combination of different techniques to benefit from their specific advantages, where Mentzer and Bienstock see time series, regression and qualitative techniques (see Sect. 3) as the most important approaches in sales forecasting. Finally, the best forecasting techniques for a time series should be selected automatically, by trying a number of different techniques and selecting the technique that provides the best forecast accuracy.

2.2 Energy Load Balancing

As another example, consider the energy market domain. One major challenge is the constantly increasing capacities of renewable energy sources (RES) due to governmental regulation efforts (e.g., climate saving propositions) and excessive funding policies [21]. Renewable energy sources pose the challenge that production depends on external factors (wind speed, amount of sunlight, etc.). Hence, available power can only be predicted but not planned, which makes it rather difficult for energy distributors to efficiently include RES into their daily schedules.

The key to balance an energy distribution network successfully is to predict as many of the most influencing (correlated) parameters for operations as possible. Such forecasts are often made by domain-specific forecasting techniques specifically designed for energy demand or supply. To forecast energy demand, for example, Ramanathan et al. [72] propose a multi-equation forecasting technique that creates a different statistical model for each hour of a day and includes various variables that capture the seasonality of the data as well as external influences (e.g., temperature).

In the past, energy balancing was typically done once per day at a specific time and, accordingly, one-day ahead demand forecasts were calculated only on

a daily basis. The need for fast response times to react to new market situations (e.g., weather changes) as well as the continuous streams of new demand and supply measurements poses additional real-time demands on the forecasting process [26]. Thus, the runtime of forecasting is very critical and, more importantly, forecast models have to be continuously adapted to changes in the time series behavior. Moreover, the hierarchical organization of the energy market requires a careful selection of the forecasting granularity (e.g., single wind installation vs. complete regions), the efficient handling of real-time mass prediction processes and the guarantee of consistency between hierarchy levels.

2.3 Online Display Advertisement

As a third example, online display advertisement allows advertisers to promote products to users by having publishers display their graphical ads on web pages. For example, a brokerage firm may wish to target males from California who visit a Finance web site, and show an ad promoting its special offers to those users. Such kind of targeted ads are channeled to users via ad networks — intermediates that package and sell ad space from multiple publishers' websites [82]. In order to be able to accept contracts and allocate inventory, an ad network has to have access to reliable forecasts of user visits. Overestimating user visit volumes may result in penalties for the publisher if guarantees are not met, whereas underestimating user visits may leave unsold user visits that often result in substantial revenue loss.

The forecasting problem in online display advertisement has several challenges [1]. First, the data to be forecasted is very high-dimensional. Specifically, each user visit is characterized by hundreds of attributes, including the demographics of the user (e.g., age, gender, location), explicitly stated interests of the user (e.g., travel, spots), implicitly inferred interests of the user (e.g, planning a vacation), characteristics of the web page being visits (e.g., sports page, travel page), and characteristics of the system being used by the user (e.g., PC vs. mobile, IP address location). Second, as a consequence of the high-dimensionality of the data, the number of combinations that needs to be forecasted is of the order of trillions. A forecast can be requested for any combination of the hundreds of attributes using arbitrary forecasting methods, ranging from traditional time-series forecasting techniques up to latent class models [12]. Nevertheless, forecasts have to be returned in real-time, of the order of a few hundred milliseconds. Several queries are issued to the forecasting system within a short span of time to decide if an advertisers' contract should be acceptable.

3 Mathematical Foundations of Time Series Forecasting

All three application areas are based on the traditional model-based time series forecasting process, which we outline in this section. After introducing the basic idea and terminology of forecasting (Subsect. 3.1), we detail the three main steps of forecasting, namely model creation (Subsect. 3.3), model usage (Subsect. 3.4),

and model maintenance (Subsect. 3.5). For further readings we refer to standard literature about time series analysis and forecasting [9,10,13].

3.1 Basic Idea and Terminology

A *time series* is sequence of observations taken sequentially in time, spaced at *equidistant* time intervals. A time series up to the *current time* t is denoted as:

$$X = (x_1, x_2, ..., x_t). \tag{1}$$

In general, each point in time might be associated with multiple observations, where we distinguished *dependent* and *independent* observations or variables. Dependent variables, also known as *measure* or *output* variables, are those variables we actually want to forecast (e.g., product sales, solar energy production). In contrast, those variables that we believe influence the value of the dependent variables are referred to as independent or explanatory variables (e.g., product price, sun radiation).

Forecasting refers to the estimation of values of a time series at some future point in time. These future values are called *forecast values* or short *forecasts* $\hat{x}_{t+k}$. The forecasts $\hat{x}_{[t+1;t+h]}$ in the interval from the next point in time $t+1$ up to time $t+h$ are usually of most interest. The length of the interval is denoted as *forecast horizon* h. Note that the term *prediction* is often used in a more general sense and covers different problem types, e.g., classification, recommendation or moving object prediction. In contrast, the term forecasting is only concerned with estimating the next and future values of a sequence, i.e., a time series.

The quality of forecast values can be expressed by calculating *prediction intervals*, which, for a certain probability, give an estimate of the interval in which forecast values will fall. If new real data is available, the error of the forecasts can be calculated by comparing forecasts with real time series data using an error metric, e.g., the *mean squared error* (MSE).

A *forecasting method* is a procedure for computing forecasts from present and past values. Most forecasting methods are based on a *forecast model*, which is learned over historical training data and used to compute the forecast values (*model-based forecasting*). Examples of approaches that do not belong to this class of methods are judgmental or similarity-based forecasting methods.

A *forecast model* consists of the following parts:

- definition of *input* and *output* time series,
- definition of the *forecasting method*, which determines how forecast values are calculated,
- *model parameters* of the forecasting method that have to be determined in the model estimation step, and
- *model state*, representing internal variables of the forecasting method that change with time.

The *input* of a model consists of n dependent and m independent variables, whereas the *output* yields from the associated forecasts of the n dependent variables. The *model parameters* and *model state* directly depend on the used forecasting method.

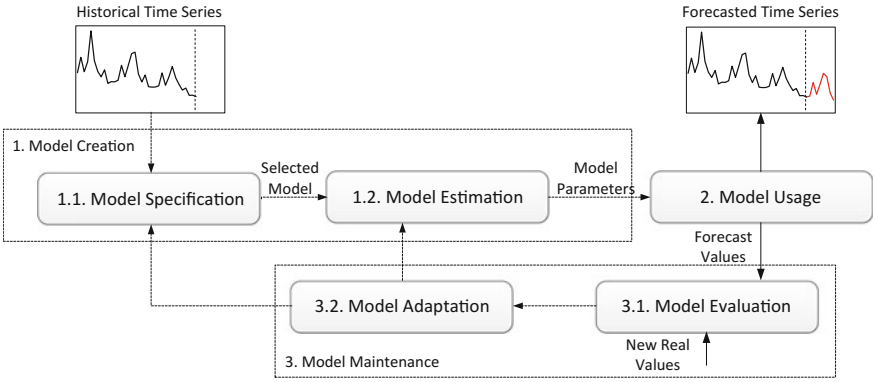


Fig. 1. General forecasting process

Finally, the different steps of forecasting can be summarized into a general *model-based time series forecasting process* (Fig. 1):

1. *Model Creation:* A forecast model is created by defining input, output as well as the forecasting method (*model specification*), and by estimating the model parameters (*model estimation*).
2. *Model Usage:* Forecast values based on the created forecast model are calculated.
3. *Model Maintenance:* The forecast model is evaluated by comparing real time series data with forecast values (*model evaluation*) and, optionally, *model adaption* is triggered by recalculating the model parameters or choosing a new model.

In what follows, we first discuss the characteristics of different forecasting methods and describe two of them in more detail. We then have a closer look at the three main steps of the forecasting process.

3.2 Overview Forecasting Methods

Numerous forecasting methods have been proposed in the literature, e.g., Gooijera and Hyndman [36] summarized over 940 papers in the period 1982–2005. Forecasting methods are often classified into *time series methods* (or univariate methods) and *causal methods* (or multivariate methods) [13]. Time series methods assume that forecasts depend only on present and past values of the single series being forecasted, possibly augmented by a function of time such as linear trend. In contrast, in causal methods forecasts depend, at least partly, on one or more additional (independent) variables (e.g., price, weather). Finally, forecasting can also be implemented by *machine learning approaches* that exploit artificial intelligence techniques, e.g., neural networks [8]. Machine learning approaches are not specifically designed for time series data and might be applied for

Table 1. Classification of forecasting methods

Time series methods	Causal methods	Machine learning
ARIMA [9]	Multivariate linear	Neural networks [88]
Exponential	Regression [86]	Support vector
Smoothing [40]	ARMAX [9]	Machines [60]
Multiple linear		Bayesian networks [87]
Regression [34]		Decision trees [56]

arbitrary predictive tasks, e.g., classification. Table 1 summarizes the different forecasting methods and gives some literature examples where these methods have been applied for time series forecasting. Many extensions to the basic formulation of those forecasting methods have been developed, including domain specific methods that are specifically designed to solve a prediction task in a certain domain.

Various studies have compared the accuracy of the different forecastingarbitrary methods with varying results depending on the domain and forecasting target. Exponential smoothing and ARIMA models have shown empirically to be able to model a wide range of real world time series [54], and are usually computationally more efficient than elaborate machine learning approaches. The main idea of both approaches is sketched in the following.

Exponential Smoothing. Exponential Smoothing is a popular scheme to produce smoothed time series, where past observations are weighted with exponentially decreasing weights [33]. In other words, recent observations are given more weight in forecasting than older observations. Different variants of exponential smoothing have been proposed, varying in the number and characteristics of the smoothing weights. The most common variants are called *single*, *double*, and *triple* exponential smoothing.

For example, single exponential smoothing has only one weight parameter, also known as the smoothing constant α:

$$a_t = \alpha x_t + (1 - \alpha) a_{t-1} \qquad \text{with } a_0 = x_0. \tag{2}$$

The forecast of single exponential smoothing is a constant value based on the last smoothed value a_t, independent of the forecast horizon h:

$$\hat{x}_{t+h} = a_t. \tag{3}$$

This method is mainly used for stationary time series fluctuating around a constant mean. In contrast, double exponential smoothing introduces an additional trend component, whereas triple exponential smoothing (also known as Holt-Winters [40]) further applies a seasonal component.

ARIMA Models. ARIMA models describe the behavior of time series using an auto-regression process [9] and consist of three main parts; the autoregression part (AR), the integration part (I) and the moving average part (MA).

The autoregressive part $AR(p)$ is computed by a linear combination of previous values up to a defined maximum lag (denoted p) combined with a random error term ϵ_t:

$$x_t = \sum_{i}^{p} \phi_i x_{t-i} + \epsilon_t. \tag{4}$$

In contrast, the moving average part $MA(q)$ describes a time series as a random error term plus some linear combination of previous random error terms up to a defined maximum lag (denoted q):

$$x_t = \sum_{i}^{q} \phi_i \epsilon_{t-i} + \epsilon_t. \tag{5}$$

Hereby, the random error terms result from a white noise process, i.e., a set of uncorrelated, normal-distributed, random variables with an assumed equal variance. Most time series can not be described solely by an AR or MA forecast model, as they show behavior of both models at the same time. For this reason, a combination of both models is useful. Additionally, an integrated part I adjusts the model for non-stationary time series, leading to so called $ARIMA(p, d, q)$ models. This basic formulation of ARIMA models can be extended to include seasonality (SARIMA) or exogenous data (ARIMAX).

A variety of research has studied the relationship between exponential smoothing and ARIMA models [33,59]. In general, all linear exponential smoothing methods have equivalent ARIMA models. However, exponential smoothing is often preferred over ARIMA due to its simplicity, robustness and the surprising accuracy that can be obtained with minimal effort in model selection.

3.3 Model Creation

Model creation is the process of defining (model specification) and training (model estimation) a forecast model for given time series data. In this section, we discuss each of these steps separately. However, in automatic model identification approaches these two steps are often combined as models are built and evaluated iteratively.

Model Specification. Model specification requires the definition of the input and output time series as well as the definition of the forecasting method.

Input and Output Selection. The output time series (i.e., the dependent variables) depends on the aim of forecasting and has to be specified by the application, whereas the input time series (i.e., the independent variables) might be manually or automatically selected. Hereby, the goal is to find those variables (often called features) that are highly correlated and significantly contribute to the time series

to be forecasted. Statistical techniques, such as principal component analysis, are often applied to find relationships between different features and to reduce the number of features [32]. Another issue in input selection is given by the time series history, e.g., how much history should be used for model estimation. Hereby, existing research has studied the minimal historical length required for specific forecasting methods [43] as well as the influence of the history length on the forecast accuracy [5] and the runtime of the parameter estimator [77].

Forecasting Method Selection. For a given data set, we need to select the forecasting method with the highest accuracy, which can be done manually or automatically. The manual approach requires knowledge of statistical theory and uses diagnostic tools such as the correlogram. Choosing a model manually is not possible if a large number of time series is involved or if the forecast is done by non-experts in the statistical area. In the automatic approach, the best model is selected empirically according to the in-sample error or a model selection criteria such as the Akaike's Information Criterion (AIC). AIC chooses the model that maximizes the so-called likelihood function and includes a regularization term, which basically avoids overfitting by increasing the training error with the number of parameters fitted in the model. Hyndman et al. [42] developed a state space framework for the class of exponential smoothing methods, where the best method is chosen automatically based on AIC. Heuristic model identification approaches for the class of ARIMA models have also been developed [41]. All automatic approaches still have the drawback that they select a single model that has to be best at all times. Ensemble approaches increase robustness by combining forecasts of several models using weighted linear combinations [36].

Model Estimation. In model estimation, the parameters of the forecasting method (called *model parameters*) are fitted to a given training time series. Thus, model estimation tries to find the best parameter combination for the training data. This process involves two main components — an *optimization function* (to specify which parameter combination is best) and an *optimization approach* (to control the search strategy). Most common optimization approaches follow an iterative search process based on the steepest descent or hill climbing technique. In each iteration one or several parameter combinations are evaluated using the optimization function and subsequently, based on the outcome of the evaluation, a new parameter combination is chosen for the next iteration. After termination, the best parameter combination according to the optimization function is outputted (Fig. 2).

The optimization function is composed of the forecasting method and the error metric used to evaluate the forecast values. Commonly used error metrics are least squares or maximum likelihood approaches.

Optimization approaches are mainly distinguished into derivative-based and derivative-free algorithms. Derivative-based methods (e.g., gradient descent, quasi-newton) exploit the optimization function's first or second derivations to move directly into the direction of the steepest descent. If the optimization function is not derivable gradient approximation techniques can be applied.

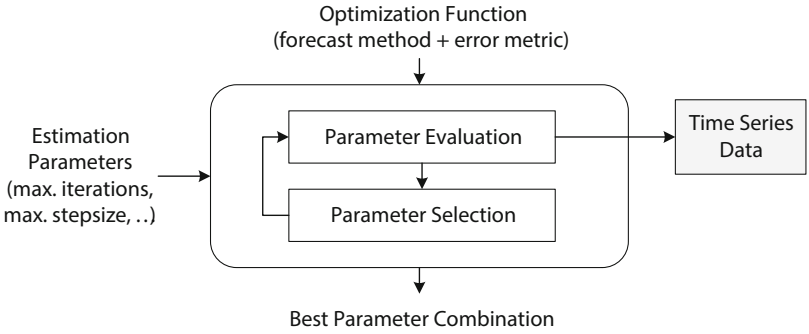


Fig. 2. Parameter estimation process

In contrast, derivative-free algorithms (e.g., simulated annealing, nelder-mead) make only direct evaluations of the optimization function, i.e., treat it as black-box. Finally, the optimization approach can be configured with various parameters, such as the maximum number of iterations or the maximum step size for selecting the next parameter combination.

3.4 Model Usage

Model usage applies the created model to forecast future values of the time series for a given forecast horizon h. In model-based forecasting, the time-consuming part is given by the model creation step, whereas model usage requires only the application of a function (with the trained parameters).

A more complex aspect of model usage concerns the aggregation of time series data. Time series data may be aggregated either across time, called *temporal aggregation*, or across several time series, called *contemporaneous aggregation* [13]. For example, suppose we have sales figures for different brand sizes of different products in successive weeks. Such data may be quite volatile and difficult to forecast without some form of aggregation, either across time (e.g. over successive 4-week periods) or across products (e.g. sum all brand sizes for the same brand). A common problem in inventory control is whether to develop a summary forecast for the aggregate of a particular group of items and then allocate this forecast to individual items based on their historical relative frequency, called the *top-down approach*, or make individual forecasts for each item, called a *bottom-up approach*. This line of research is called *hierarchical forecasting* [31].

3.5 Model Maintenance

As time proceeds, new values of the time series are observable, which impact the forecast model. First, the state of the model has to be updated to the current time series values, which we refer to as *model update*. Second, the parameters of the model or even the forecasting method might change, which is meant by

the term *model maintenance*. Complex seasonal patterns or unexpected changes in time series' characteristics (also called concept drift) like customers' buying preferences or the influence of weather predictions usually require such an adaption of the model. Model maintenance exhibits two major challenges: (1) when to trigger forecast model maintenance (model evaluation) and (2) how to efficiently adapt the forecast model parameters (model adaption).

Model Evaluation. If new real data is available, the model can be evaluated by calculating the forecast error using a specified error metric. Commonly known error metrics are the mean absolute error (MAE) and mean squared error (MSE). Such error metrics, however, depend on the scale of the time series values, are hard to judge and do not allow comparisons between time series of different scale or mean. In contrast, percentage error metrics evaluate the error's magnitude instead of its size. One example is the *symmetric mean absolute percentage error* (SMAPE) [53], which was also used in the M3-competition — a mayor time series competitions in the business forecasting domain [54].

Simple model evaluation approaches trigger model adaption independently of the actual time series data and might be time-based (after a time interval), update-based (after a fixed number of new values) or event-based (after updating an exogenous variable or on request). More advanced approaches monitor the temporal development of the time series. For example, error-based approaches evaluate the forecast error after each new real value and trigger adaption whenever the error surpasses a predefined threshold [16]. Other approaches use statistical information about the time series like minimum and maximum values [39] or statistical tests [51].

Model Adaption. A simple way to realize model adaption is by starting from scratch and by re-executing the model estimation and, optionally, the model identification step as done in the initialization. This can easily be improved by reusing previous information, e.g., by providing the last parameters of the model as starting parameters to the model estimation step. A more advanced approach stores previous model parameters in a decision tree according to specific context information (e.g., type of day, temperature) and uses them as starting point in the estimation step if the same context reoccurs [17]. Another approach extends genetic algorithms with dynamical features, where previously good models are given an advantage in future selection rounds [83]. Orthogonal approaches adapt the training set of the models and include only recent observations in the model creating step, either employing fixed-size or dynamic windows [85]. All these approaches use offline parameter estimation approaches, where the parameters are fully reestimated using any of the previously discussed optimization approaches. As an alternative, approximate online optimization algorithms [90] may be applied, which evaluate the objective function after each new time series value and alter the parameter estimate made so far accordingly. However, the parameters by this approach are only approximative and will deviate from parameter estimates produced by full optimization. Finally, the forecast model

itself might be designed in an adaptive way, which aims at completely avoiding the need for recalculating the model parameters. Self-adaptive forecasting methods extend existing forecasting methods with time-dependent parameters (e.g., [72]). Ensemble methods combine forecasts of several models and adapt the weights of the ensemble members over time.

4 Architectural Integration

Recall our vision of a model-based database system outlined in the introduction. In terms of time series forecasting, our overall objective is the transparent integration of forecast models inside a database management system. We discussed the main challenges of typical forecasting applications in Sect. 2, such as high-dimensional data, real-time requirements, complex and domain-specific forecasting methods, diverse workloads, non-expert users, and fast evolving data sets. Moreover, in Sect. 3, we highlighted the most important steps of the forecasting process and outlined challenges in this context, such as the selection of the best forecasting method, long parameter (re-)estimation times, and the importance of a smart maintenance strategy.

We now turn our attention to the actual integration of forecasting inside a database management systems. First of all, in this section, we review general solutions to the integration of any kind of statistical method, not necessarily time series forecasting, into a DBMS and outline to what extent they support our overall objective as well as the discussed challenges. We classify existing methods into (1) no integration approaches (Subsect. 4.1), (2) partial integration approaches that try to keep changes to the database as small as possible (Subsect. 4.2), and (3) full integration approaches that actually extend the functionality of a database system (Subsect. 4.3).

4.1 No Database Integration

No database integration approaches refer to the use or integration of statistical approaches in other system categories, such as external software or Map-Reduce environments.

Statistical Software Environments. Traditionally, statistical computations have been performed outside the database system by specialized software, which uses the DBMS primarily as backend data server. Ganesan and Shenoy [50] provide a survey of forecasting software up to the year 2006. Probably the most well-known commercial software environments are Matlab [55], SAS [75] and SPSS [78]. All include a large variety of specific forecasting methods including approaches for automatic and ensemble forecasting.

A popular open-source statistical software package is the R framework [71]. With over 2,000 add-on packages, it is comparable to the big commercial packages SAS and SPSS. In the context of time series forecasting, it contains a wide

variety of model types and parameter estimators. Model types range from linear models, exponential smoothing, ARIMA up to machine learning approaches. Additionally, automatic model identification approaches for ARIMA and exponential smoothing models are available [41]. A general-purpose optimization functions including five different optimization approaches (e.g., Nelder-Mead, Simulated Annealing) is used to estimate the parameters of the various model types. As R is open-source it can be easily extended with new, domain-specific, forecasting methods. A number of approaches aim at improving the handling of large amount of data in R [76]. The proposed techniques range from simple ones that require rewriting and adapting of existing scripts and functions up to more complex ones that try to adapt the R environment in a transparent manner. For example, an approach called RIOT [89] focuses on storing and querying arrays, and tries to make R more I/O efficient by introducing a new expression algebra.

Map-Reduce Environments. Massive data sets and and large clusters of machines have led to an increased interest in implementing statistical algorithms on Map-Reduce environments. Several research has investigated the implementing of scalable versions of machine learning algorithms on Map-Reduce, ranging from proprietary (e.g., [67]) to open source implementations [6]. In contrast, declarative machine learning approaches try to avoid the low-level implementations of specific algorithms on Map-Reduce. For example, SystemML [35] provides a declarative high-level language for writing machine learning algorithms, which is automatically compiled and optimized into a set of Map-Reduce jobs. Another declarative approach is MLbase [49], which proposes an optimizer that selects and dynamically adapts the choice of the learning algorithm.

All discussed approaches exhibit the general problem of being outside the database system. This might be valid in application scenarios where data is stored in external files or fits into main memory, and database characteristics such as transaction management are not required. However, if data is managed in

```
INSERT INTO YD
 SELECT i,j,
  sum((YV.val-C.val)**2)
 FROM YV,C
 WHERE YV.l = C.l
 GROUP BY i,j;
```

(a) Computing the Euclidean distance for k-means in SQL [64]

```
SELECT svm_regression(
 'input_table',
 'model_table',
 parallel,
 'kernel_func',
 verb DEFAULT false,
 eta DEFAULT 0.1,
 nu DEFAULT 0.005,
 slambda DEFAULT 0.05);
```

(b) Training SVM model in MAD [15]

```
initialize(ModelCoef *w)
{ .... }

transition(ModelCoef *w,
           Example e)
{ ...
 wx = Dot_Product(w,e.x);
 c = stepsize * e.y;
... }

terminate(ModelCoef *w)
{ .... }
```

(c) Implementation of SVM in BISMARCK [24]

Fig. 3. Exploiting SQL and UDFs

a traditional database management system those approaches have several drawbacks. They require data transfer from the database to the statistical software system and vice versa, might lead to inconsistencies between data and models and miss optimization potential such as the reuse of models by multiple queries. Surely, some or all of this functionality could be implemented in the external system. This, however, requires the re-implementation of existing concepts of the DBMS and will eventually lead to the design of a new database system outside of the actual database system.

4.2 Partial Database Integration

Partial database integration approaches try to leave the database system itself unchanged or include advanced analytical functionality by keeping the changes to the databases as small as possible. We distinguish three approaches in this area: SQL extensions and UDFs, customized functions, and bi-directional communication approaches.

Exploiting SQL and UDFs. The first set of approaches uses database query languages to express linear algebra functions or even higher-level algorithms and, thus, try to get a database system to act like a statistical software environment. Such approaches either (1) use directly SQL to implement data mining algorithms, (2) hide mining functionality behind user defined functions and provide high-level SQL extensions to interact with mining models and results, or (3) support the implementation of mining algorithms by providing low-level language extensions (Fig. 3).

First, SQL can be used to directly implement data mining algorithms, such as Bayesian classifiers [65] and clustering approaches [64]. Figure 3(a) shows an excerpt of the k-means clustering algorithm in SQL, namely the computation of the Euclidean distance between the data points and the centroids.

Second, high level language constructs for specific mining tasks have been proposed. The MAD approach [15] consists of a hierarchy of mathematical concepts in SQL that enable vector and matrix operations, simple functions as well as sophisticated analytical methods such as ordinary least squares, conjugate gradient, or support vector machines (Fig. 3(b)). The Splash system [22] views statistical models, such as probability density functions, as SQL aggregation operations and proposes extensions to the relational data model and SQL query language for interaction with such models. Ordonez and Pitchaimalai [66] propose a general system that integrates statistical models such as correlation, linear regression, principal component analysis, and clustering into a database using SQL queries and UDFs. Besides these general approaches, a large number of research papers has addressed specific data mining methods. Examples include association rule mining [45] and sequential patterns [74]. All approaches provide a high-level query language that hide statistical details from less sophisticated users.

In contrast, the ATLaS system [84] introduces a lower-level language, where the user can integrate simple data mining algorithms with user-defined aggregates by implementing three standard functions in SQL — initialize, iterate, and terminate. The ATLaS language processor optimizes and translates ATLaS SQL programs (e.g., decision tree classifier, apriori) into C++ code. In another work, Feng et al. [24] propose a unified architecture (called BISMARCK) for convex programming problems, such as support vector machines, where local solutions are always globally optimal. Their main component is an in-RDBMS implementation of the incremental gradient descent optimization approach that allows to solve a number of convex programming tasks in a unified way. Analogue to ATLaS, a developer can integrate analytic tasks by implementing three standard functions using user-defined aggregates, using any language supported by the DBMS (Fig. 3(c)). Additionally, performance optimizations, namely partitioning and parallelization schemes, are studied.

Exploiting SQL and UDFs for data mining algorithms has the advantage of being flexible: the analyst is able to develop algorithms independently on top of the database and nevertheless is able to profit from performance gains by running analytical methods inside the database [15,66]. However, SQL itself is not designed to express statistical computations. SQL follows a declarative logic (e.g., it lacks a convenient syntax for iteration), whereas statistical computing requires imperative and functional programming logic. This leads to a high overhead of statistical computations and makes it impossible to express sophisticated time series methods in SQL. In contrast, UDFs allow arbitrary programming languages supported by the DBMS and might be used to implement advanced time series methods. However, within all approaches models are explicitly queried and not transparently processed as first class citizens inside the database. Furthermore, within an UDF, all decision have to be made locally, whereas a DBMS exhibits a global view over all queries and operators, allowing joint optimization techniques such as model reuse and maintenance in multidimensional data sets.

Customized Functions with Proprietary Languages. Instead of developing SQL extensions, another possible approach is to implement data mining functionality internally as customized black box functions and offer proprietary languages to the corresponding methods. This approach has been used by most commercial database management systems, which provide advanced time series forecasting methods to some extend.

Microsoft SQL Server offers a Data Mining Extension (DMX) for creating models for various mining tasks such as association rule mining, clustering, and also time series forecasting [79]. Two explicit time series forecasting methods are included, autoregressive trees and ARIMA models, which are by default combined to a hybrid forecasting method. DMX supports a set of functions that allow to query a forecast model for predictions and additional statistical information (Fig. 4 (a)). Chaudhuri et al. [14] propose optimizations for queries on classification and clustering models in SQL Server. Using model-specific

```
SELECT
 [Forecasting_MIXED].[ModelRegion],
  PredictTimeSeries
   ([Forecasting_MIXED].[Quantity],6),
  PredictTimeSeries
   ([Forecasting_MIXED].[Amount],6)
FROM  [Forecasting_MIXED]
WHERE [ModelRegion] = 'M200 Europe'
  OR  [ModelRegion] = 'M200 Pacific'
```

(a) Forecasting using DMX in SQLServer

```
DEFINE fcst.sales DECIMAL <month>
LIMIT month
TO year 'Yr95' 'Yr96'
FORECAST LENGTH 12
METHOD WINTERS
PERIODICITY 12
ALPHA .5 BETA .5 GAMMA .5
TIME month
FCNAME fcst.sales sales
```

(b) FORECAST command in Oracle

Fig. 4. Comparison of forecast functionalities in SQLServer and Oracle

algorithms, predicates on data mining models are transformed to simple selection predicates, which can then be exploited for access path selection.

Oracle Data Mining (ODM) provides twelve data mining algorithms that address classification, regression, association rules, clustering, attribute importance, and feature selection problems [63]. ODM provides PL/SQL and Java application programming interfaces for model building and model scoring functions as well as a Oracle Data Miner graphical user interface for data analysts who want to use a GUI. Additionally, Oracles offers a `FORECAST` command as part of its OLAP DML (Fig. 4 (b)), which supports linear as well as non-linear regression methods or exponential smoothing [61].

The IBM DB2 Warehouse data mining capabilities provide algorithms for mining tasks such as clustering, classification, association rule mining and regression [7], but no specific time series forecasting methods are supported. Data mining models are represented using the Predictive Model Markup Language (PMML) and stored in relational tables.

Commercial database systems increase the efficiency by pushing statistical computation closer to the database and also offer some advanced time series forecasting methods. However, due to the usage of proprietary languages, forecasting is not integrated within the relational processing and optimization of SQL queries. Additionally, the black box approach makes it difficult to customize, extend and optimize data mining functionality, including the whole forecasting life cycle. Subsequently, models are not handled as first class citizens leading to same drawbacks as discussed for UDFs.

Bi-directional Communication. Finally, a third possibility is to reuse existing statistical tools like R and improve the cooperation between the database and the statistical software system.

Ricardo [18] focuses on large-scale data management systems such as Hadoop and proposes a system where large-scale computations are expressed in JAQL, a high level query interface on top of Hadoop, while R is called for smaller-scale single-node statistical tasks. This requires the programmer to identify scalability of different components of an algorithm, and re-express large-scale matrix operations in terms of JAQL queries.

A second example is the integration of R into the SAP in-memory computing engine. Große et al. [37] developed a shared memory-based data exchange to reduce the communication overhead between R and the database, and, additionally, included R scripts as part of the database execution plan. The latter approach allows multiple R runtimes in parallel processing advanced analytic functionality.

On the commercial side, Oracle R Enterprise [62] embeds the functionality of R inside the Oracle database. A transparency layer supports mapping of R data types to Oracle Database objects and generates SQL transparently from R expressions. Additionally, R scripts can be executed inside the database and the Oracle R Connector for Hadoop enables R users to work with a Hadoop cluster.

Bi-directional communication approaches avoid the re-implementation of statistical functionality and reuse well established statistical software environments, which usually provide advanced time series forecasting functionality. As proposed by Große et al. [37], R scripts can be encapsulated into a native database operator, allowing the processing and optimization of statistical computations within the traditional query execution plan. However, again, the whole statistical computation is treated as black box within the R operator and statistical models are hidden within R scripts. Therefore, models can not be transparently precomputed, materialized and managed within the database system, and optimization possibilities on the forecasting process itself are limited.

4.3 Full Database Integration

In contrast to partial integration approaches, full integration approaches either design a completely new special-purpose database system or extend the core functionality of a traditional database system.

Special-Purpose Database System. A representative of a special-purpose database system is SciDB [11], which targets application domains that involve very large array data such as scientific applications. The SciDB database is organized as collections of n-dimensional arrays and addresses challenges like array storage and partitioning as well as parallel processing of array operations. SciDB supports query patterns such as array slicing and dicing, array scans, and binary

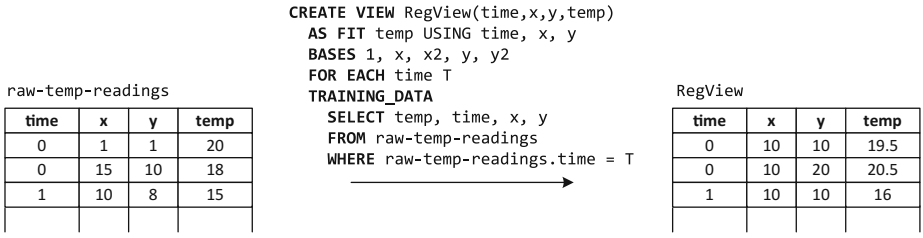


Fig. 5. Model-based views in MauveDB

array operations, but no advanced statistical methods like time series forecasting. The approach of SciDB adapts and optimizes the database system specifically to the target use case and goes far beyond the idea of integrating statistical methods inside a database system. Our goal is to provide time series forecasting within traditional database processing for various use cases and to benefit from existing database technologies, requiring a more general full database integration approach.

Traditional Database System. The MauveDB project [19] integrates statistical modeling inside a DBMS using so-called model-base views. Model-based views generalize the view concept and allow the definition of views as statistical models using extensions to SQL (Fig. 5). Such views can be queried like traditional view leading to new classes of view access operators inside the DBMS. Additionally, MauveDB provides different maintenance strategies that keep models consistent with changes to the data, e.g., no, partial, or full materialization. A similar motivation follows FunctionDB [80], where mathematical functions are treated as first-class citizens inside a DBMS. Queries are answered with discrete points that are computed from piecewise polynomial functions, where the data is discretized as late as possible. These leads to various new relational operations that operate directly on the symbolic representations of the functions.

Akdere et al. [3] expand the idea of MauveDB and propose a Predictive Database Management System (PDBMS), called Longview, that enables declarative predictive queries as well as automatic model training and selection. Longview provides two interfaces for access to its predictive functionality. A direct interface offers direct access to the functionality of the prediction models (regression and classification), whereas a declarative interface is used for high-level access by non-expert users. Prediction models can be built using the `CREATE PREDICTOR` command and then directly queried or referenced in traditional views. Internally, a model manager is responsible for creating materialized models or selecting models in an ad-hoc fashion. However, Akdere et al. [3] provide only a high-level overview over such a system and identify several open research aspects, including automatic selection of materialized models for given cost and accuracy constraints as well as execution and optimization of predictive queries.

Both projects, MauveDB and Longview, integrate statistical methods within a database by viewing models as first class citizens. However, they target statistical methods such as interpolation or classification, and not time series forecasting. Forecasting requires specific time series forecasting methods as well specific model identification, model evaluation, and model maintenance strategies. Furthermore, both approaches require the explicit selection of a model in a query and do not realize declarative and transparent forecast queries.

Besides these general approaches, a number of papers has addressed database aspects of specific mining models. For example, the HAZY project [48] builds upon the work of MauveDB [19] and addresses the incremental maintenance of classification views. The Monte Carlo Database System [46] allows the creation of arbitrary stochastic models for uncertain data and focuses on Monte Carlo

analysis of such models. Simple regression methods have been natively supported by relational database systems for about a decade, and have been incorporated into the SQL language [4]. Other approaches focus on support vector machines [58] and interpolation functions [38].

Additionally, specific time series approaches have been proposed, which we discuss in more detail in the next section.

5 In-DBMS Time Series Forecasting Techniques

We now review dedicated time series forecasting techniques in the database context. Such approaches usually address a specific forecasting method or scenario and discuss individual aspects of the forecasting process in this context.

The processing of declarative forecast queries in traditional databases was first introduced within the Fa System [20]. The main contribution of Duan and Babu [20] is an automatic feature selection approach for forecasting multidimensional time series. A query execution plan in Fa consists of a sequence of transformers, which shift or remove attributes from the input data set; a builder, which computes a forecast model from the transformed data set; and a predictor to make the forecast itself. Fa's plan search is based on an iterative algorithm. Each iteration selects a set of attributes using several heuristics and empirically evaluates five different forecasting methods (regression and machine learning approaches) for the selected attributes, leading to more and more accurate plans over time. Furthermore, an adaptive version of the plan search algorithm for continuous forecast queries is proposed.

Later, Ge and Zdonik [34] proposed an automatic model selection approach for multivariate regression models. Their approach is based on the observation that the best history length of the time series, in terms of accuracy and efficiency, varies according to the requested forecast horizon. An empirical approach iteratively increases the history as well as the number of data points and uses statistical tests of hypotheses to build a single regression model. A skip-list data structure supports the efficient selection of the data at a certain granularity. Additionally, a randomized algorithm is provided that chooses a set of forecast models for a given query workload and maintenance constraints. Maintenance either involves rebuilding the regression model or choosing new properties of the models, i.e., history length and data granularity, where the latter is done after a fixed number of new time series values. Ge and Zdonik also introduce query optimization techniques for range, aggregation, and join queries exploiting the properties of regression models. To compute a join over a future time range, for example, a simple approach would generate all future data points using the regression models and then perform a traditional join on the raw data. In contrast, the second relation could simply use the regression functions of the first relation and solve an equality condition to retrieve the matching tuples. However, such optimization techniques can only be exploited by simple regression function and are not applicable to more sophisticated time series methods, which require additional input data to compute the forecast values (e.g., auto-regressive models).

A formal definition of a forecast operator was developed by Parisi et al. [69] Also, the integration of forecast operators with standard relational operators was explored by identifying simple plan restructuring rules for three relational operators; selection, projection, and union.

In another work, Akdere et al. [2] present optimization techniques for continuous prediction queries using Bayesian Networks as forecasting method. They propose to model point and range-based prediction queries as query plans and introduce different materialization options within a plan. A selection approach finds an execution plan with minimum computation costs for given memory constraints.

The challenge of forecasting high-dimensional data was addressed in the area of online display advertisement [1]. Hundreds of attributes and trillions of attribute combinations have to be forecasted, making it impossible to build a forecast model for each single time series in the database. To solve this issue, only forecast models for a small subset of attribute combinations are built, which are selected manually for seasonality and historical importance. Forecasts for remaining attributes are obtained by exploiting correlations between the attributes. Specifically, three different correlation approaches are evaluated: a Naive Bayes approach that assumes attribute independence, a partwise independence approach that infers combinations of correlated attributes, and a sampling-based approach that computes correlations for a sample of the data.

In the area of data stream management, research has investigated the joint forecasting of multiple data streams. The MUSCLES method [86] uses multivariate linear regression to forecast values of one stream based on the previous values of all streams. MUSCLE is able to adapt to changing correlations among time sequences. SPIRIT [68] finds correlations among data streams by computing the principal components. An auto-regressive model is built directly over the principal components and used for the estimation of missing values.

In terms of model maintenance, Rosenthal and Lehner [73] developed an incremental model adaption approach for simple auto-regressive models and propose a generic approach, called on-demand estimation, for more complex ARIMA models. The parameters of ARIMA models can be estimated using the maximum likelihood approach, which tries to maximizes the probability of reproducing the training data from the given parameters. On-demand estimation incrementally maintains the so called likelihood function and triggers model adaption if new time series values lead to significant changes in the function's optimum.

Maintenance issues have also been discussed in the context of streaming databases and sensor networks. Tulone and Madden [81] propose an error-based model evaluation strategy for auto-regressive models. Model adaption is triggered based on two thresholds, which distinguish outlier values and distribution changes in the data. The latter suggest that re-learning the model might be necessary. In contrast, the sensor data management architecture PRESTO [52] retrains models periodically. Ikonomovska et al. [44] propose an incremental

stream mining algorithm for regression and model trees, including drift detection and model adaptation to maintain accurate and updated regression models at any time.

To sum up, the need for integrating analytical methods into traditional databases has been identified by many existing research projects, addressing general approaches as well as specific forecasting methods. However, non of the existing approaches provide a complete solution for in-DBMS forecasting, including declarative forecast queries, arbitrary forecasting methods, relational query processing, query optimization, forecast model maintenance, transparent model reuse, and automatic model selection. Following the discussion of the general forecasting process from Sect. 3.1, we now introduce an architecture that integrates the whole forecasting life cycle natively into an existing DBMS and, additionally, benefits from existing work on in-DBMS forecasting techniques.

6 A Flash-Forward Database System

In contrast to flash back queries that allow a view on the data in the past, we developed a *Flash-Foreward Database System*. We explain the necessary extensions to a traditional DBMS from two angles. First, in Subsect. 6.1, we investigate changes to the different types of schemas of a DBMS, which are usually described by the ANSI/SPARC architecture. Subsequently, in Subsect. 6.2, we discuss the actual implementation of a forecast-enabled database management system.

6.1 ANSI/SPARC Architecture

The ANSI/SPARC architecture forms an abstract design standard for a database management systems and gives a general architecture for database functions, interfaces, and usages. The objective of the three-level architecture is to separate the users' view of the data from the way that it is physically represented. Specifically, the use of the data is described in the external schema, the meaning of the data in the conceptual schema, and the data storage in the internal schema. Time series forecasting consists of two major data entities — time series and forecast models — that have to be arranged within the ANSI-SPARC architecture. We now systematically study each of the three levels of the architecture and discuss where we have to add new concepts and where we can reuse existing concepts from the ANSI-SPARC architecture (see Fig. 6) [25].

External Schema. The external schema in the traditional ANSI/SPARC architecture consists of user-defined data views, which can be seen as virtual tables storing the results of specific queries. Time series can just be seen as a special view that ensure the representation of the data as time series. A *time series view* requires at least a time attribute containing discrete points in time and another attribute exhibiting the measurements at these specified moments, for example:

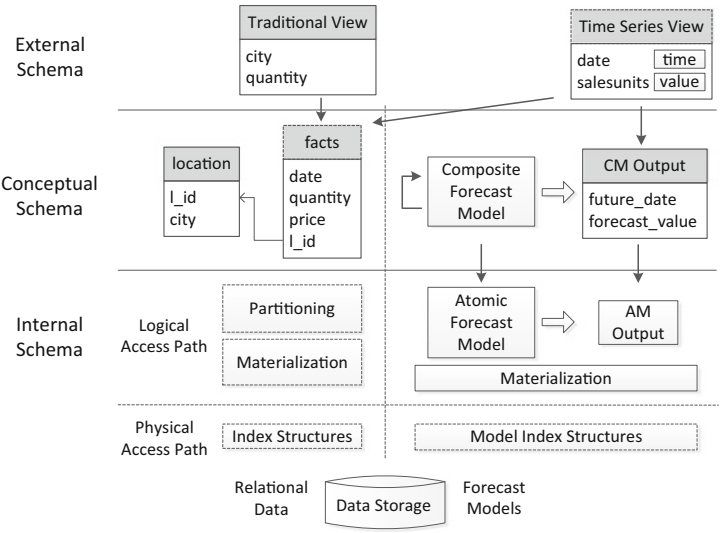


Fig. 6. Integration of forecasting within the ANSI/SPARC architecture

```
CREATE VIEW timeSeriesView AS
  SELECT    MONTH(date) AS month, SUM(salesunits) as sales
  FROM      facts f, products p
  WHERE         f.product_id = p.product_id
  AND           p.pname = 'audio'
  AND             month in (now() - 3 months, now() + 3 months)
  GROUP BY  month
```

A time series view can represent historical values, forecast values, or both. If forecast values are involved, the time series view has to be defined by a query requesting future values. It might contain further information such as standard deviation or prediction intervals, which clearly distinguish future from historical values. Once real values are available, they replace the forecast values.

Conceptual Schema. The conceptual level includes a data schema that describes available entities, their relationship and contained attributes and can be seen as an abstraction from the internal data representation. Likewise, a *composite forecast model* is defined as a conceptual abstraction from a concrete *atomic forecast model*. A composite model might directly refer to a single atomic forecast model from the internal schema, representing a simple forecast of a time series, e.g., sales units of audio devices. However, composite forecast models can also describe a (hierarchical) forecast model composition. When forecasting sales units of audio devices in Germany, for example, the forecasting can be decomposed into forecasts of the sales units for all German states, or further down in the hierarchy, sales units of all German cities. The composite forecast model can define a hierarchical forecast composition referring further composite models on multiple hierarchy levels and, on the leaf level, ultimately refer to atomic forecast

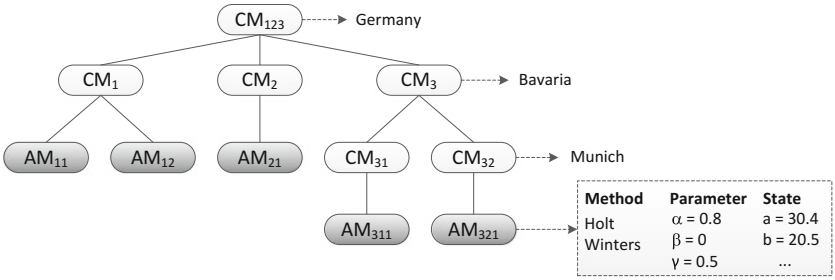


Fig. 7. Example model composition

models defined in the internal schema (see Example in Fig. 7). Multiple atomic forecast models, with different forecasting methods or parameter combinations, might be referenced by one composite model on the leaf level, enabling ensemble forecasting.

With respect to the external layer, each composite forecast model is linked with a single time series view from the external schema. It further defines a single output, the *composite model output* (CM output), which is a special table complying to the same rules as the time series view and exhibits the same hierarchical forecast model composition as defined for the associated composite forecast model. The forecast values, i.e., the composite forecast model output, are computed by a weighted linear combination of the referenced forecast models according to the defined forecast model composition.

Internal Schema. The logical and the physical data access paths are defined in the internal schema of the ANSI-SPARC architecture. Logical access paths refer to data organization aspects like partitioning and materialization, whereas the physical access paths define low level access structures like indexes. Likewise, atomic forecast models are defined that represent a non-decomposable forecast model. A single atomic forecast model is represented by input and output definitions, the forecasting method, model parameters, and the current model state (see Subsect. 3.1). Here, the input is the data as defined in the associated time series view, referenced through the connected composite forecast model. The forecasting method is chosen from a *forecast model catalog* that represents all forecasting methods available in the DBMS and is predefined with respect to the application domain (similar to the approach of Longview [3]). The output of atomic forecast models is represented by a special data structure called *atomic forecast model output* (AM Output). Optionally, additional attributes might be included in the model output (e.g., prediction intervals).

Traditionally, materialization is performed to precompute complex database queries. Similarly, composite and atomic models can be materialized for faster query response times. Materialized models might store composition rules, model parameters, model states, or even the model output, i.e., forecast results. On the physical access paths, specific model specific index structures might be

applied [27,34]. Additionally, traditional or customized time series index structures ensure efficient processing of time series data and access of time series values in a subsequent order.

6.2 DBMS Architecture

In contrast to the ANSI/SPARC architecture, which mainly describes interfaces, we now discuss the actual realization of a flash-forward database system [28]. Figure 8 shows the main components of a database management systems, exemplary on the open-source DBMS PostgreSQL [70]. Our extensions to traditional database components are shown by grey boxes. In what follows, we shortly outline the core idea of the main components, more details can be found in [27–30].

First of all, declarative forecast queries require the extension of the parser so that forecast-specific keywords (e.g., forecast horizon, forecasting attributes, forecasting method) are recognized. After parsing, the statement is identified as complex (e.g., select, insert, delete) or simple (e.g., create table) by the traffic cop. Simple utility commands are processed by a dedicated component, which contains forecast model specific utility commands (e.g., create model, drop model). This enables a database administrator to explicitly create and delete models. Complex statements are planned by the optimizer. Hereby, the existing optimizer is extended with (1) new forecast-specific physical operators, (2) new cost models for those operators as well as (3) new optimizer rules.

Following the general forecasting process, forecast operators decouple model creation and model usage functionality. The `CreateModel` operator is responsible for model creation. It receives a time series view as input and outputs a set of forecast models. Subsequently, the `Forecast` operator receives as input a set of forecast models and outputs a time series relation containing corresponding forecast values. In case of ad hoc forecast queries, these two operators appear jointly, with the `Forecast` operator sitting on top of the `CreateModel` operator. However, the separation of both operators enables the transparent reuse of materialized models.

Materialized models are handled by two new components — *model matching* and *model maintenance* [27]. Model matching is responsible for finding suitable models for a given forecast query, including atomic forecast models and potential model compositions. Depending on the query type, model matching can be accessed by either the optimizer or executor. In contrast, model maintenance is performed after insert statements. It is responsible for finding models that are based on those inserts. Model maintenance includes a model update step, an evaluation step and, optionally, a parameter re-estimation step.

Besides, the transparent reuse of materialized models, the optimizer is also responsible for processing ad hoc forecast queries that require the creation of a model at query runtime. Hereby, we exploit traditional database sampling techniques to reduce the amount of processed data by the `CreateModel` operator [29]. For example, one optimization techniques reduces the time series length for parameter estimation.

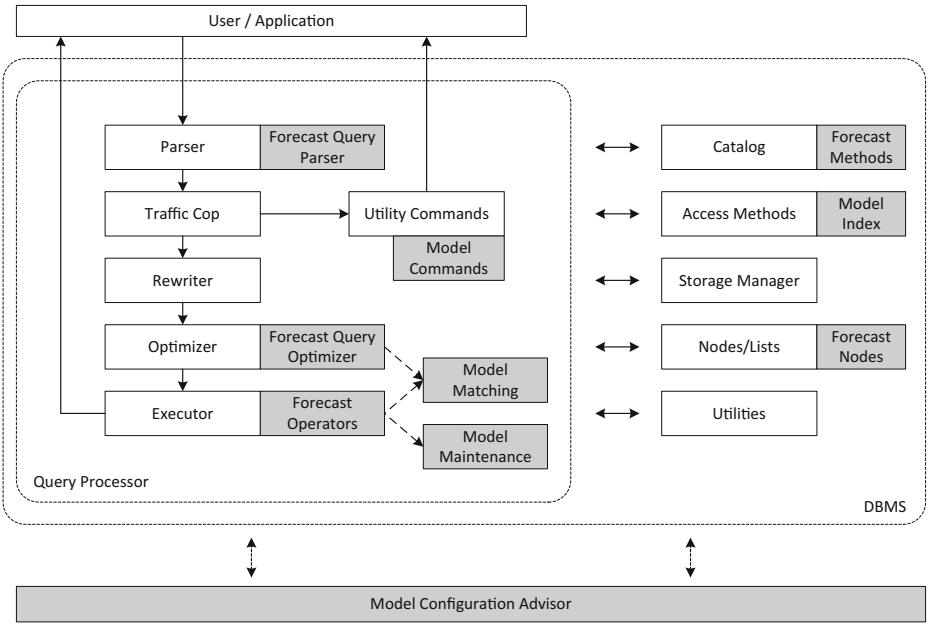


Fig. 8. Architecture of a flash-forward database system

The discussed query processing components are supported by other modules. The catalog stores meta data about atomic and composite forecast models as well as the previously mentioned forecasting method catalog. As discussed in Subsect. 6.1 access methods are extended with model-specific access structures, supporting the model matching and model maintenance components [27]. Information about internal query structures and query plans are stored in nodes and lists in PostgreSQL. Thus, new nodes for forecast queries have to be added. The remaining two components, the storage manager and utilities, containing support functions, are currently not touched by our extensions. Traditional tables are used to store time series data as well as models, which enables the direct reuse of the different functionalities of a storage manager.

Besides the extension of internal components, one additional external component, the *model configuration advisor*, is available [30]. Similar to a traditional index or materialized view advisor, which proposes a physical design of indexes and materialized views to the database system, the model configuration advisor recommends a physical design of forecast models. In contrast to traditional advisers, which usually focus on minimizing the runtime of a given workload (optionally giving some storage constraints), the optimization objective of the model configuration advisor is twofold — minimize the query runtime and maximize the forecast accuracy. The technical challenge of the advisor comes from the fact that there are no known ways to estimate the accuracy of a physical design of forecast models without actually deploying and querying it. Hence, the

model advisor is based on an iterative process that, based on heuristics, selects a set of candidate time series in each iteration for which a model should be built and analyzed. Parameters of the advisor like the number of candidate models are automatically tuned in a control phase.

7 Conclusions and Future Work

The need for integrating statistical methods into databases has been identified by many existing research projects, addressing general approaches as well as specific statistical methods. In this article, we provided a review of existing work and discussed its applicability for supporting a transparent model-based database system, where we specifically focused on forecast models. Based on the traditional ANSI/SPARC architecture, we introduced a novel forecast-enabled database management system, the flash-forward database system. Our approach belongs to the class of full database integration approaches and integrates the whole forecasting life cycle.

Recall the application scenarios and associated requirements presented in Sect. 2. The proposed flash-forward database systems enables forecasting by non-expert users (e.g., supply chain managers) and hides the complexity of the forecasting process. It allows the integration of a suite of forecasting techniques and domain-specific forecasting methods (e.g., for energy demand and supply forecasting). The reuse of materialized forecast models enables the processing of forecast queries in real-time as required, for example, in energy load balancing or display advertisement. The maintenance component continuously and efficiently adapts models to changes in the time series behavior (e.g., weather changes in the energy domain). Finally, the model configuration advisor selects a physical design of forecast models for large multi-dimensional data sets (e.g., in producting planning), balancing query efficiency and forecast accuracy.

Although we specifically focused on forecast models in this article, many of the discussed challenges, foundations, and concepts can be applied to other statistical models. We have taken a first step towards a model-based database system and opened up interesting opportunities for further research. We conclude by mentioning a few of them:

- *Configuration Maintenance:* How can we maintain a configuration of forecast models in an online fashion? Can we develop incremental algorithms that avoid the complete re-execution of the forecast model advisor?
- *Parallelized Query Execution:* How can we improve the execution of adhoc forecast queries that require the creation of a new model? Can we develop efficient operators that exploit parallelization opportunities of modern (heterogeneous) hardware environments?
- *Lineage:* Can we provide the user with information about the origin as well as reliability of the query result?

References

1. Agarwal, D., Chen, D., Lin, L., Shanmugasundaram, J., Vee, E.: Forecasting high-dimensional data. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 1003–1012 (2010)
2. Akdere, M., Çetintemel, U., Upfal, E.: Database-support for continuous prediction queries over streaming data. Proc. VLDB Endowment **3**, 1291–1301 (2010)
3. Akdere, M., Cetintemel, U., Riondato, M., Upfal, E., Zdonik, S.B.: The case for predictive database systems: opportunities and challenges. In: Fifth Biennial Conference on Innovative Data Systems Research, pp. 167–174 (2011)
4. Alur, N., Haas, P., Momiroska, D., Read, P., Summers, N., Totanes, V., Zuzarte, C.: DB2 UDB's High Function Business Intelligence in e-Business. IBM Redbook Series (2002)
5. Andersen, T.G., Bollerslev, T., Lange, S.: Forecasting financial market volatility: sample frequency vis-a-vis forecast horizon. J. Empirical Finan. **6**, 457–477 (1999)
6. Apache. Apache Mahout (2013). http://mahout.apache.org/
7. Ballard, C., Rollins, J., Ramos, J., Perkins, A., Hale, R., Doerneich, A., Milner, E.C., Chodagam, J.: Dynamic Warehousing: Data Mining Made Easy. IBM Redbooks Series (2007). http://www.redbooks.ibm.com/redbooks/pdfs/sg247418.pdf
8. Bontempi, G., Ben Taieb, S., Le Borgne, Y.-A.: Machine learning strategies for time series forecasting. In: Aufaure, M.-A., Zimányi, E. (eds.) eBISS 2012. LNBIP, vol. 138, pp. 62–77. Springer, Heidelberg (2013)
9. Box, G.E.P., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control, 4th edn. Wiley, New York (2008)
10. Brockwell, P.J., Davis, R.A.: Introduction to Time Series and Forecasting. Prentice Hall, Englewood Clifs (2002)
11. Brown, P.G.: Overview of sciDB: large scale array storage, processing and analysis. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 963–968 (2010)
12. Cetintas, S., Chen, D., Si, L., Shen, B., Datbayev, Z.: Forecasting counts of user visits for online display advertising with probabilistic latent class models. In: International Conference on Research and Development in Information Retrieval, pp. 1217–1218 (2011)
13. Chatfield, C.: Time-Series Forecasting. Chapman & Hall, Boca Raton (2000)
14. Chaudhuri, S., Narasayya, V., Sarawagi, S.: Efficient evaluation of queries with mining predicates. In: Proceedings of the 18th International Conference on Data Engineering, pp. 529–540 (2002)
15. Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C.: MAD skills: new analysis practices for big data. Proc. VLDB Endowment **2**, 1481–1492 (2009)
16. Dannecker, L., Böhm, M., Lehner, W., Hackenbroich, G.: Forcasting evolving time series of energy demand and supply. In: Eder, J., Bielikova, M., Tjoa, A.M. (eds.) ADBIS 2011. LNCS, vol. 6909, pp. 302–315. Springer, Heidelberg (2011)
17. Dannecker, L., Schulze, R., Böhm, M., Lehner, W., Hackenbroich, G.: Context-aware parameter estimation for forecast models in the energy domain. In: Bayard Cushing, J., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 491–508. Springer, Heidelberg (2011)
18. Das, S., Sismanis, Y., Beyer, K.S., Gemulla, R., Haas, P.J., McPherson, J.: Ricardo: integrating r and hadoop. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, pp. 987–998 (2010)

19. Deshpande, A., Madden, S.: MauveDB: supporting model-based user views in database systems. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 73–84 (2006)
20. Duan, S., Babu, S.: Processing forecasting queries. In: Proceedings of the VLDB Endowment, pp. 711–722 (2007)
21. European Commission. Energy Roadmap 2050. Brussels (2011)
22. Fang, L., LeFevre, K.: Splash: ad-hoc querying of data and statistical models. In: Proceedings of the 13th International Conference on Extending Database Technology, pp. 275–286 (2010)
23. Feng, H.: Performance problems of forecasting systems. In: 15th East-European Conference on Advances in Databases and Information Systems, pp. 254–261 (2011)
24. Feng, X., Kumar, A., Recht, B., Ré, C.: Towards a unified architecture for in-rdbms analytics. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 325–336 (2012)
25. Fischer, U., Dannecker, L., Siksnys, L., Rosenthal, F., Boehm, M., Lehner, W.: Towards integrated data analytics: time series forecasting in dbms. Datenbank-Spektrum, 1–9 (2012)
26. Fischer, U., Kaulakienė, D., Khalefa, M.E., Lehner, W., Pedersen, T.B., Šikšnys, L., Thomsen, C.: Real-time business intelligence in the MIRABEL smart grid system. In: Castellanos, M., Dayal, U., Rundensteiner, E.A. (eds.) BIRTE 2012. LNBIP, vol. 154, pp. 1–22. Springer, Heidelberg (2013)
27. Fischer, U., Rosenthal, F., Böhm, M., Lehner, W.: Indexing forecast models for matching and maintenance. In: IDEAS, pp. 26–31 (2010)
28. Fischer, U., Rosenthal, F., Lehner, W.: F2DB: the flash-forward database system. In: Proceedings of the 28th International Conference on Data Engineering, pp. 1245–1248 (2012)
29. Fischer, U., Rosenthal, F., Lehner, W.: Sample-based forecasting exploiting hierarchical time series. In: Proceedings of the 16th International Database Engineering and Applications Sysmposium, pp. 120–129 (2012)
30. Fischer, U., Schildt, C., Hartmann, C., Lehner, W.: Forecasting the data cube: a model configuration advisor for multi-dimensional data sets. In: Proceedings of the 29th International Conference on Data Engineering (2013)
31. Fliedner, G.: Hierarichal forecasting issues and use guidelines. Ind. Manage. Data Syst. **101**, 5–12 (2001)
32. Ganapathi, A., Kuno, H., Dayal, U., Wiener, J.L., Fox, A., Jordan, M., Patterson, D.: Predicting multiple metrics for queries: better decisions enabled by machine learning. In: Proceedings of the 25th International Conference on Data Engineering, pp. 592–603 (2009)
33. Gardner Jr, E.S.: Exponential smoothing: the state of the art. Int. J. Forecast. **4**, 1–28 (1985)
34. Ge, T., Zdonik, S.B.: A skip-list approach for efficiently processing forecasting queries. Proc. VLDB Endowment **1**, 984–995 (2008)
35. Ghoting, A., Krishnamurthy, R., Pednault, E., Reinwald, B., Sindhwani, V., Tatikonda, S., Tian, Y., Vaithyanathan, S.: SystemML: declarative machine learning on mapreduce. In: Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, pp. 231–242 (2011)
36. Gooijera, J.G.D., Hyndman, R.J.: 25 years of time series forecasting. Int. J. Forecast. **22**, 443–473 (2006)

37. Große, P., Lehner, W., Weichert, T., Färber, F., Li, W.-S.: Bridging two worlds with rice integrating r into the sap in-memory computing engine. Proc. VLDB Endowment **4**, 1307–1317 (2011)
38. Grumbach, S., Rigaux, P., Segoufin, L.: Manipulating interpolated data is easier than you thought. In: Proceedings of the 26th International Conference on Very Large Data Bases, pp. 156–165 (2000)
39. Harries, M., Horn, K.: Detecting concept drift in financial time series prediction using symbolic machine learning. In: Proceedings of the 8th Australian Joint Conference on Artificial Intelligence, pp. 91–98 (1995)
40. Holt, C.C.: Forecasting seasonals and trends by exponentially weighted moving averages. Int. J. Forecast. **20**, 5–10 (2004)
41. Hyndman, R.J., Khandakar, Y.: Automatic time series forecasting: the forecast package for R. J. Stat. Softw. **27**, 1–22 (2008)
42. Hyndman, R.J., Koehler, A.B., Snyder, R.D., Grose, S.: A state space framework for automatic forecasting using exponential smoothing methods. Int. J. Forecast. **18**, 439–454 (2002)
43. Hyndman, R.J., Kostenko, A.V.: Minimum sample size requirements for seasonal forecasting models. Foresight: the Int. J. Appl Forecast. **6**, 12–15 (2007)
44. Ikonomovska, E., Gama, J., Džeroski, S.: Learning model trees from evolving data streams. Data Min. Knowl. Discov. **23**, 128–168 (2011)
45. Imieliński, T., Virmani, A.: Msql: a query language for database mining. Data Min. Knowl. Discov. **3**, 373–408 (1999)
46. Jampani, R., Xu, F., Wu, M., Perez, L.L., Jermaine, C., Haas, P.J.: Mcdb: a monte carlo approach to managing uncertain data. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 687–700 (2008)
47. Kimball, R., Ross, M.: The Data Warehouse Toolkit. Wiley, New York (2002)
48. Koc, M.L., Ré, C.: Incrementally maintaining classification using an rdbms. Proc. VLDB Endowment **4**, 302–313 (2011)
49. Kraska, T., Talwalkar, A., Duchi, J., Griffith, R., Franklin, M.J., Jordan, M.: Mlbase:a distributed machine learning system. In: 6th Biennial Conference on Innovative Data Systems Research (2013)
50. Kusters, U., McCullough, B., Bell, M.: Forecasting software: past, present and future. Int. J. Forecast. **22**, 599–615 (2006)
51. Lazarescu, M.M., Venkatesh, S., Bui, H.H.: Using multiple windows to track concept drift. Intell. Data Anal. J., 1–28 (2003)
52. Li, M., Ganesan, D., Shenoy, P.: Presto: feedback-driven data management in sensor networks. In: Proceedings of the 3rd Conference on Networked Systems Design & Implementation, pp. 23–23 (2006)
53. Makridakis, S.: Accuracy measures: theoretical and practical concerns. Int. J. Forecast. **9**, 527–529 (1993)
54. Makridakis, S., Hibon, M.: The M3-Competition: results, conclusions and implications. Int. J. Forecast. **16**, 451–476 (2000)
55. Matlab. The language of technical computing (2012). http://www.mathworks.com/products/matlab/
56. Meek, C., Chickering, D.M., Heckerman, D.: Autoregressive tree models for time-series analysis. In: SIAM International Conference on Data Mining (2002)
57. Mentzer, J.T., Bienstock, C.C.: The seven principles of sales-forecasting systems. Supply Chain, Manage. Rev. **11**, 76–83 (1998)

58. Milenova, B.L., Yarmus, J.S., Campos, M.M.: Svm in oracle database 10g: removing the barriers to widespread adoption of support vector machines. In: Proceedings of the VLDB Endowment, pp. 1152–1163 (2005)
59. Mills, T.C.: Time Series Techniques for Economists. Business & Economics (1991)
60. Müller, K.-R., Smola, A.J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: Gerstner, W., Hasler, M., Germond, A., Nicoud, J.-D. (eds.) ICANN 1997. LNCS, vol. 1327, pp. 999–1004. Springer, Heidelberg (1997)
61. Oracle OLAP DML Reference 11g. Forecast - dml statement (2012). http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1052.htm
62. Oracle R. Enterprise user's guide (2012). http://docs.oracle.com/cd/E27988_01/doc/doc.112/e26499.pdf
63. Oracle White Paper. Oracle data mining 11g release 2 - competing on in-database analytics (2012)
64. Ordonez, C.: Programming the k-means clustering algorithm in sql. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 823–828 (2004)
65. Ordonez, C., Pitchaimalai, S.K.: Bayesian classifiers programmed in sql. IEEE Trans. Knowl. Data Eng. **22**, 139–144 (2010)
66. Ordonez, C., Pitchaimalai, S.K.: One-pass data mining algorithms in a dbms with udfs. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp. 1217–1220 (2011)
67. Panda, B., Herbach, J.S., Basu, S., Bayardo, R.J.: Planet: massively parallel learning of tree ensembles with mapreduce. Proc. VLDB Endowment **2**, 1426–1437 (2009)
68. Papadimitriou, S., Sun, J., Faloutsos, C.: Streaming pattern discovery in multiple time-series. In: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 697–708 (2005)
69. Parisi, F., Sliva, A., Subrahmanian, V.S.: Embedding forecast operators in databases. In: Benferhat, S., Grant, J. (eds.) SUM 2011. LNCS, vol. 6929, pp. 373–386. Springer, Heidelberg (2011)
70. PostgreSQL (2012). http://www.postgresql.org/
71. R Development Core Team. R: A language and environment for statistical computing, reference index version 2.1.1. R Foundation for Statistical Computing (2012). http://www.r-project.org
72. Ramanathan, R., Engle, R., Granger, C.W.J., Vahid-Araghi, F., Brace, C.: Short-run forecasts of electricity loads and peaks. Int. J. Forecast. **13**(2), 161–174 (1997)
73. Rosenthal, F., Lehner, W.: Efficient in-database maintenance of ARIMA models. In: Bayard Cushing, J., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 537–545. Springer, Heidelberg (2011)
74. Sadri, R., Zaniolo, C., Zarkesh, A.M., Adibi, J.: A sequential pattern query language for supporting instant data mining for e-services. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 653–656 (2001)
75. SAS. Business intelligence software (2012). http://www.sas.com
76. Schmidberger, M., Morgan, M., Eddelbuettel, D., Yu, H., Tierney, L., Mansmann, U.: State-of-the-art in parallel computing with R. J. Stat. Softw. **31**, 1–27 (2009)
77. Shalev-Shwartz, S., Srebro, N.: SVM optimization: inverse dependence on training set size. In: Proceedings of the 25th International Conference on Machine Learning, pp. 928–935 (2008)
78. SPSS. IBM SPSS Statistics (2012). http://www-01.ibm.com/software/analytics/spss/

79. SQL Server. Data Mining Algorithms - Books Online for SQL Server 2012 (2012). http://msdn.microsoft.com/en-us/library/ms175595.aspx
80. Thiagarajan, A., Madden, S.: Querying continuous functions in a database system. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 791–804 (2008)
81. Tulone, D., Madden, S.: PAQ: time series forecasting for approximate query answering in sensor networks. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 21–37. Springer, Heidelberg (2006)
82. Turner, J.: The planning of guaranteed targeted display advertising. Oper. Res. **60**, 18–33 (2012)
83. Wagner, N., Michalewicz, Z., Khouja, M., McGregor, R.: Time series forecasting for dynamic environments: the dyfor genetic program model. IEEE Trans. Evol. Comput. **11**, 433–452 (2007)
84. Wang, H., Zaniolo, C., Luo, C.R.: ATLAS: a small but complete sql extension for data mining and data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 1113–1116 (2003)
85. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Mach. Learn. **23**, 69–101 (1996)
86. Yi, B., Sidiropoulos, N.D., Johnson, T., Jagadish, H.V., Faloutsos, C., Biliris, A.: Online data mining for co-evolving time sequences. In: Proceedings of the 16th International Conference on Data Engineering, pp. 13–22 (2000)
87. Zhang, C., Sun, S., Yu, G.: A bayesian network approach to time series forecasting of short-term traffic flows. In: Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems, pp. 216–221 (2004)
88. Zhang, G., Eddy-Patuwo, B., Hu, M.Y.: Forecasting with artificial neural networks: the state of the art. Int. J. Forecast. **14**, 35–62 (1998)
89. Zhang, Y., Zhang, W., Yang, J.: I/O-efficient statistical computing with RIOT. In: Proceedings of the 26th International Conference on Data Engineering, pp. 1157–1160 (2010)
90. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th International Conference on Machine Learning, pp. 928–936 (2003)

On Index Structures for Star Query Processing in Data Warehouses

Artur Wojciechowski and Robert Wrembel(✉)

Institute of Computing Science, Poznan University of Technology, Poznań, Poland
{Artur.Wojciechowski,Robert.Wrembel}@cs.put.poznan.pl

Abstract. One of the important research and technological issues in data warehouse performance is the optimization of analytical queries. Most of the research have been focusing on optimizing such queries by means of materialized views, data and index partitioning, as well as various index structures including: join indexes, bitmap join indexes, multidimensional indexes or index-based multidimensional clusters. These structures neither well support navigation along dimension hierarchies nor optimize joins with the *Time* dimension, which in practice is used in the majority of analytical queries. In this chapter we overview the basic index structures, namely: a bitmap index, a join index, and a bitmap join index. Based on these indexes, we show how to build another index, called *Time-HOBI*, for optimizing queries that address the *Time* dimension and compute aggregates along dimension hierarchies. We further discuss the extension of the index with additional data structure for storing aggregate values along the hierarchical structure of the index. The aggregates are used for speeding up aggregate queries along dimension hierarchies. Furthermore, we show how the index is used for answering queries in an example data warehouse. Finally, we discuss its performance-related characteristics, based on experiments.

Keywords: Data warehouse · Query optimization · Star query · Hierarchical index · Bitmap index · Join index · Bitmap join index · Time-HOBI

1 Introduction

A traditional data warehouse architecture has been developed in order to analyze heterogeneous and distributed data managed by an enterprise. A core component of this architecture is a database, called a *data warehouse* (DW) that stores the integrated data, both current and historical ones. The content of a DW is analyzed by various analytical queries for the purpose of discovering trends (e.g., demand and sales of products), discovering patterns of behavior (e.g., customer habits, credit repayment history) and anomalies (e.g., credit card usage) as well as for finding dependencies between data (e.g., market basket analysis, suggested buying, insurance fee assessment). These techniques are commonly referred to as On-Line Analytical Processing (OLAP).

E. Zimányi (Ed.): eBISS 2013, LNBIP 172, pp. 182–217, 2014.
DOI: 10.1007/978-3-319-05461-2_6,

Analytical queries, commonly known as *star queries* process large volumes of data. The queries join a central table with multiple reference tables (called dimension tables) that define the context of the analyses. The queries next aggregate data at various levels of granularity, from fine grained to coarse - by means of roll-up operations and from coarse to fine grained - by means of drill-down operations. Since a query response time is one of the key factors of a DW performance, providing means for reducing the time is one of the research and technological challenges. In this area, different mechanisms have been proposed in the research literature, e.g., [27] and applied in commercial data warehouse management systems (DWMSs), i.e., materialized views and query rewriting, e.g., [20], data partitioning and parallel processing, e.g., [17,45,55] as well as advanced indexing, e.g, [6]. The research on indexing resulted in multiple index structures. From these structures, the successfully applied ones in commercial DWMSs include: join indexes, e.g., [57], bitmap indexes, e.g., [38,53], bitmap join indexes, e.g., [9,39], various multidimensional indexes, like for example R-tree [21], Quad-tree [15], and K-d-b-tree [46], as well as index-based multidimensional clusters [42].

We argue that the flat structure of a bitmap index can still be better optimized to match a hierarchical structure of dimensions, in order to facilitate the roll-up and drill-down operations. Moreover, the existing implementations of the aforementioned indexes do not exploit the fact that most of the analytical queries analyze data in time and thus require costly operations of joining a central table with a dimension table, which stores time data.

Although bitmap indexes, join indexes, and bitmap join indexes substantially decrease execution times of analytical queries, not all commercially available database management systems support them. For example, Oracle implements the bitmap index and the bitmap join index. IBM DB2 and SQL Server support implicitly created temporary bitmap indexes only, which are used to optimize joins.

Chapter contribution. In this chapter we overview the basic index structures, namely: a bitmap index, a join index, and a bitmap join index. Based on these indexes, we show how to build another index, called *Time-HOBI*, for optimizing queries that compute aggregates along dimension hierarchies and that analyze data in time. The index was originally presented in [12,36]. In this chapter, we introduce the following additional contributions:

- the extension of *Time-HOBI* with additional data structure for storing aggregate values along the index hierarchy (the aggregates are used for speeding up aggregate queries along dimension hierarchies),
- the analysis of how the index is used for answering queries in an example data warehouse,
- the experimental evaluation of the extended *Time-HOBI*.

Chapter content. This chapter is organized as follows. Section 2 presents basic concepts on data warehousing used in this chapter. Section 3 outlines basic index

structures applied in data warehouses. Section 4 discusses the components of the *Time-HOBI* index and shows how the index is used in a query execution plan. Section 5 discusses performance characteristics of *Time-HOBI* obtained from multiple experimental evaluations. Section 6 presents related work in the area of indexing DW data. Finally, Sect. 7 summarizes the chapter.

2 Data Warehouse Basics

In this section we present the basic concepts and definitions in the area of data warehousing, i.e., a multidimensional data model and its relational implementations, as well as star queries.

2.1 DW Model and Schema

In order to support various analyses, data stored in a DW are represented in the *multidimensional data model* [22,24]. In this model an elementary information being the subject of analysis is called a *fact*. It contains numerical features, called *measures* that quantify the fact. Values of measures are analyzed in the context of *dimensions*. Dimensions often have a hierarchical structure composed of levels, such that $L_i \rightarrow L_j$, where $\rightarrow$ denotes hierarchical assignment between a lower level L_i and upper level L_j, also known as a roll-up or an aggregation path [32]. Following the aggregation path, data can be aggregated along a dimension hierarchy. Level data are called *level instances*. Hierarchically connected level instances form a *dimension instance*.

The multidimensional model is often implemented in relational databases (ROLAP) [11], where fact data are stored in a *fact table*, and level instances are stored in *dimension level tables*. In a ROLAP implementation two basic types of conceptual schemas are used, i.e. a star schema and a snowflake schema [11]. In the *star schema*, each dimension is composed of only one (typically denormalized) level table. In the *snowflake schema*, a dimension is composed of multiple normalized level tables connected by foreign key - primary key relationships. The two basic DW conceptual schemas can be used for creating a *starflake schema* [25]. In this schema some dimensions are composed of normalized and some of denormalized level tables. Star schemas store redundant data and are generally more efficient for queries that join upper levels of dimensions with a fact table. Conversely, for such queries snowflake schemas offer worse performance but there is no data redundancy.

The example "Auctions" DW snowflake schema is shown in Fig. 1. It includes the fact table *Auctions* that stores data about finished Internet auctions. The schema allows to analyze auctions and to aggregate values of measures *price* and *quantity*, with respect to three dimensions, namely: *Time*, *Location*, and *Product*. To this end, the *Auctions* fact table is connected to the dimensions via foreign keys: *dateID*, *cityID*, and *prodID*, respectively. The dimensions have hierarchical structures. For example, dimension *Product* is composed of two

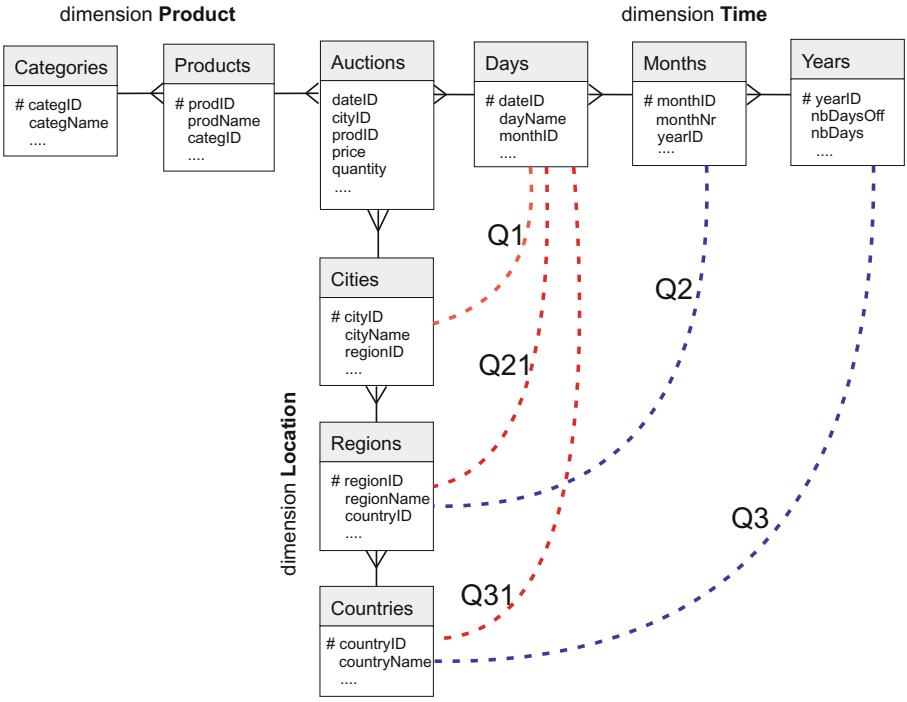


Fig. 1. The example data warehouse snowflake schema

level tables, namely *Products* and *Categories*, such that *Products* → *Categories*. For simplicity reasons only the most important tables' attributes are shown. Notice that the *price* measure represents the current price that was paid for the set of identical items in a given auction, whereas *quantity* is the number of the sold items.

The example "Auctions" DW star schema is shown in Fig. 2. The *Product*, *Location*, and *Time* dimensions were denormalized. Thus, each of them is implemented as a single table.

Figure 3 shows the instance of dimension *Product*. It includes the instances of level *Categories* and level *Products*. Level *Categories* include 2 instances, namely 'Ultrabook' and 'Tablet'. Level *Products* include 7 instances, namely 'Asus Zenbook', 'Dell XPS Duo', 'Toshiba Portege Z930-14T', etc. 'iPad mini', 'Samsung Galaxy Note', and 'Asus Vivio Tab' belong to category 'Tablet' and the others belong to category 'Ultrabook'.

Notice that throughout the paper we use the snowflake schema for illustration purposes only. The *Time-HOBI* index, discussed in Sect. 4 is applicable to the star, snowflake, and starflake schemas. Moreover, in Sect. 5 we evaluated the index for both the star and snowflake schemas.

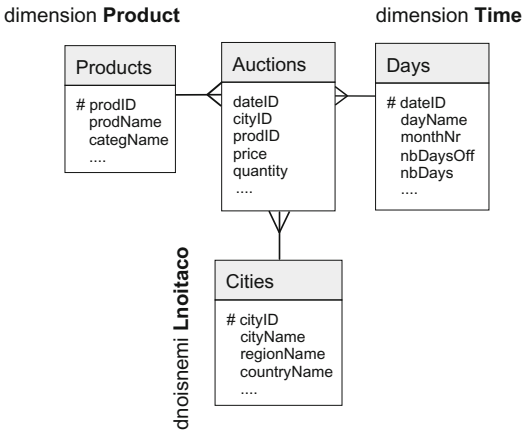


Fig. 2. The example data warehouse star schema

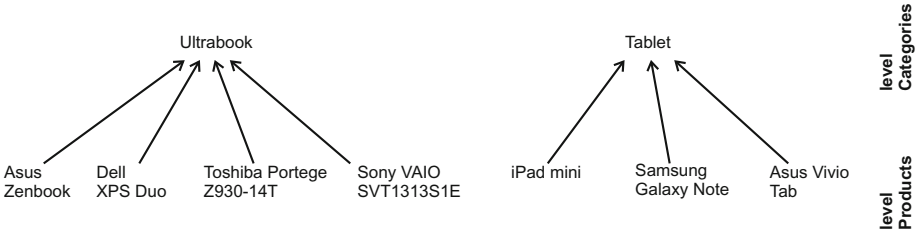


Fig. 3. The example of the *Product* dimension instance

2.2 Star Queries

Star queries, executed on any of the aforementioned DW schemas, join a fact table with multiple level tables. In Fig. 1, we marked (by means of dashed lines) the tables joined by various star queries. For example, Q1 joins tables *Auctions*, *Cities*, and *Days*, whereas Q3 joins *Auctions*, *Cities*, *Regions*, *Countries*, *Days*, *Months*, and *Years*. As an example let us consider the star query Q2 that computes monthly auction sales per region, as shown in Fig. 4.

3 Index Data Structures

Star queries can profit from applying some indexes in the process of retrieving data. In this section we outline three indexes that inspired us while developing *Time-HOBI*. They include: a join index, a bitmap index, and a bitmap join index. We also outline how the oracle implementation of the bitmap join index supports star queries exemplified by query pattern Q3.

```
SELECT r.regionName, m.monthID, m.yearID, sum(a.price), sum(a.quantity)
FROM
   Auctions a,
   Days d, Months m,
   Cities c, Regions r
WHERE
   a.dateID=d.dateID
   AND d.monthID=m.monthID
   AND a.cityID=c.cityID
   AND c.regionID=r.regionID
GROUP BY
   r.regionName, m.monthID, m.yearID
```

Fig. 4. The example query Q2

3.1 Join Index

A *join index* represent the materialized join of two tables, say R and S. As defined in [31,57], a join index is a table composed of two attributes. It stores the set of pairs (r_i, s_j) where r_i and s_j denote identifiers of tuples from R and S, respectively, that join on a given predicate. For the purpose of searching the join index faster, it is physically ordered (clustered) by one of the attributes. Alternatively, the access to the join index can be organized by means of a B-tree or a hash index [39]. Typically, in a DW the index joins a dimension table and a fact table. The index is created either on a join attribute (typically a primary key) or on another attribute (typically storing unique values) of a dimension level table. In order to illustrate the idea behind the join index let us consider the Example 1.

Example 1. Let us consider the *Products* and *Auctions* tables from the DW schema shown in Fig. 1. Their content is shown in Table 1. For explanatory reasons, both tables include also explicit column *ROWID* that stores physical addresses of records. *ROWID*s also play the role of row identifiers. The join index defined on column *ProdID* is shown in Table 2.

As one can observe from the above example, the join index stores a materialized (precomputed) join of tables *Products* and *Auctions*. Thus, it will optimize queries like:

```
select ...
from Auctions a, Products p
where a.prodID=p.prodID ...
```

Table 1. Example tables in the "Auctions" data warehouse (from Fig. 1)

table Auctions			
ROWID	price	cityID	prodID
0AA0	...	POZ	100
0AA1	...	WRO	230
0AA2	...	POZ	100
0AA3	...	WAW	300
0AA4	...	WAW	300
0AA5	...	WRO	230

table Products			
ROWID	prodID	prodName	categID
BFF1	100	HP Pavilion	ELE
BFF2	230	Dell Inspiron	ELE
BFF3	300	Acer Ferrari	ELE

Table 2. Example join index on *Products.prodID*

Products.ROWID	Auctions.ROWID
BFF1	0AA0
BFF1	0AA2
BFF2	0AA1
BFF2	0AA5
BFF3	0AA3
BFF3	0AA4

3.2 Bitmap Index

Analytical queries not only join data, but also filter data by means of query predicates. Efficient filtering of large data volumes may be supported by *bitmap indexes* [13,38,53,60]. Conceptually, a bitmap index created on an attribute a_m of table T is organized as the collection of bitmaps. For each value val_i in the domain of a_m a separate bitmap is created. A bitmap is a vector of bits, where the number of bits is equal to the number of records in table T. The values of bits in bitmap for val_i are set as follows. The n-th bit is set to 1 if the value of attribute a_m for the n-th record is equal to val_i. Otherwise the bit is set to 0. At the implementation level, access to bitmaps can be realized either by means of a B-tree whose leaves store pointers to bitmaps [38] or as simple arrays in a binary file [48].

Example 2. In order to illustrate the idea behind the bitmap index let us review the fact table *Auctions*, shown in Table 3. The table contains attribute *prodID*, whose values are from the set {100, 230, 300}. A bitmap index created on this attribute will be composed of three bitmaps, denoted as *Bm100*, *Bm230*, and *Bm300*, as shown in Table 3.

Bitmap *Bm100* describes rows whose value of attribute *prodID* is equal to *100*, i.e., the first bit in this bitmap is equal to 1 since the value of *prodID* of the first row in table *Auctions* is equal to *100*. The second bit in *Bm100* is equal to 0 since the value of *prodID* of the second row in *Auctions* does not equal *100*, etc. In exactly the same way the bits are set in *Bm230* and *Bm300*. Such a bitmap index will offer a good response time for a query selecting for example data on auctions concerning products identified by *100* or by *300*.

Table 3. The example table *Auctions* and the bitmap index created on attribute *Auctions.prodID*

table Auctions				bitmap index on Auctions.prodID		
				Bm100	Bm230	Bm300
price	prodID	...		prodID=100	prodID=230	prodID=300
...	100	...	⟵	1	0	0
...	230	...	⟵	0	1	0
...	100	...	⟵	1	0	0
...	300	...	⟵	0	0	1
...	300	...	⟵	0	0	1
...	230	...	⟵	0	1	0

In order to find auction rows fulfilling this criterion, it is sufficient to OR bitmaps *Bm100* and *Bm300* to construct the final result bitmap. Then, records pointed to by bits equal to '1' in the result bitmap are fetched from the *Auctions* table.

Bitmap indexes allow to answer queries with the `count` function without accessing tables, since answers to such queries can be computed by simply counting bits equal to '1' in a result bitmap.

The size of a bitmap index strongly depends on the cardinality (domain width) of an indexed attribute, i.e., the index size increases when the cardinality of an indexed attribute increases. Thus, for attributes of high cardinalities (wide domains) bitmap indexes become very large. In order to reduce the size of bitmap indexes defined on attributes of high cardinalities, the two following approaches have been proposed in the research literature, namely: (1) extensions to the structure of the basic bitmap index, e.g., [10,29,41,54,62,63], and (2) bitmap index compression techniques, e.g., [4,14,37,52,59,61]. Discussing these techniques is out of scope of this chapter and their overviews can be found in [53,58].

3.3 Bitmap Join Index

A *bitmap join index* [5,39,41] combines concepts of the join index and the bitmap index. Thus, the bitmap join index takes the advantage of the join index since it allows to materialize a join of tables. It also takes the advantage of the bitmap index with respect to efficient data filtering by means of AND, OR, and NOT operations on bitmaps. Conceptually, this index is organized as the join index, but instead of ROWIDs of a fact table's rows the index stores bitmaps that point to the appropriate fact table's rows. A lookup entry to the bitmap is by the ROWID of a row from a dimension level table (or an attribute uniquely identifying a row in that table). Similarly as for the ordinary join index, the access to the bitmap join index lookup column can be organized by means of a B-tree or a hash index. In order to illustrate the idea behind the bitmap join index let us consider the Example 3.

Example 3. Let us return to Example 1 and let us define the bitmap join index on attribute *prodID* of level table *Products*. Conceptually, the entries of this index are shown in Table 4. The lookup attribute of the index is *prodID*. A bitmap is associated with every value of this attribute. For example, the bitmap for *prodID*=100 points to the rows from table *Auctions* that concern this product, i.e., the 1st and 3rd row in table *Auctions* concern a product of *prodID*=100.

3.4 Indexes in Star Query Processing

In order to asses how star queries utilize the indexes discussed above, we executed in Oracle11g queries Q1, Q2, Q21, Q3, and Q31, as shown in Fig. 1. We selected Oracle as it supports user-managed both bitmap indexes and bitmap join indexes, whereas other systems, including IBM DB2 and Microsoft SQL Server,

Table 4. Example bitmap join index organized as a lookup by attribute *Products.prodID*

Products.prodID	bitmap
100	1
	0
	1
	0
	0
	0
230	0
	1
	0
	0
	0
	1
300	0
	0
	0
	1
	1
	0

support only system defined temporal bitmap indexes [58]. In this section we outline the execution of query Q3, expressed by means of the SQL code shown in Fig. 5. Notice that Q3 allows to parameterize its selectivity by means of the WHERE clause. We run the query for selectivities from 5 to 60 %.

In order to provide a query optimizer the means for optimizing the query, we defined the following indexes: (1) the bitmap index on attribute *year*, (2) the bitmap index on attribute *countryName*, (3) the concatenated bitmap join index on *year* and *countryName*, using the SQL command shown in Fig. 6.

```
SELECT y.yearID, co.countryName, sum(a.price), sum(a.quantity)
FROM
   Auctions a,
   Days d, Months m, Years y,
   Cities ci, Regions r, Countries co
WHERE
   y.yearID in (year1, ..., yearN)
   AND co.countryName in (country1, ..., countryN)
   AND a.cityID=ci.cityID
   AND ci.regionID=r.regionID
   AND r.countryID=co.countryID
   AND a.dateID=d.dateID
   AND d.monthID=m.monthID
   AND m.yearID=y.yearID
GROUP BY
   y.yearID, co.countryName
```

Fig. 5. The example query Q3

```
CREATE BITMAP INDEX bmi_a_years_countries
ON auctions(y.yearID, co.countryName)
FROM
   Auctions a,
   Days d, Months m, Years y,
   Cities ci, Regions r, Countries co
WHERE
   a.cityID=ci.cityID
   AND r.regionID=ci.regionID
   AND r.countryID=co.countryID
   AND a.dateID=d.dateID
   AND d.monthID=m.monthID
   AND m.yearID=y.yearID
```

Fig. 6. The example concatenated bitmap join index

The query execution plan for selectivity equal 7 % is shown in Fig. 7. It was constructed by the Oracle11g (Enterprise Edition Release 11.2.0.1.0 - 64bit Production) cost query optimizer with full statistics available. We can notice that the plan is quite complex. Even with the defined bitmap join index, 6 joins were executed. For other tested query selectivities, their respective execution plans were complex and expensive as well.

The analysis of execution plans of other star queries, including Q1, Q2, Q21, and Q31 reveals that also these queries could be better optimized, i.e., the costly join operations with hierarchical dimensions, including *Time*, could be

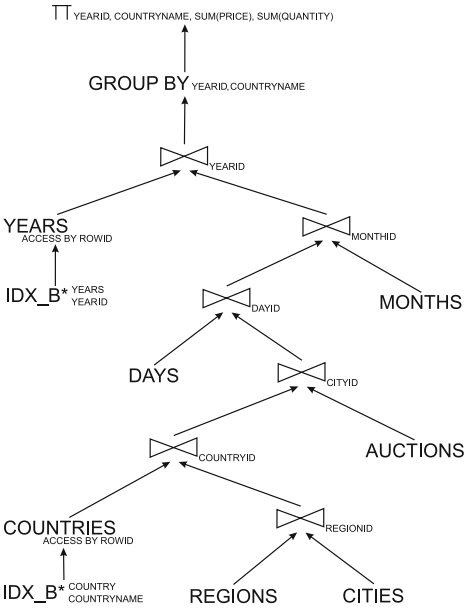


Fig. 7. Execution plan obtained from Oracle for star query Q3 with the selectivity equal 7 %

eliminated or minimized. This observation led us to the development of the index called *Time-HOBI*, originally proposed in [36].

4 Index Time-HOBI

The *Time-HOBI* index is build of three components, namely:

- Hierarchically Organized Bitmap Index (HOBI), where one bitmap index is maintained for one dimension level [12],
- Time Index (TI) that implicitly encodes time in every dimension [36],
- Partial Aggregates (PA) - that store precomputed aggregates along dimension hierarchies that is a new contribution introduced in this chapter.

In this section we present the three aforementioned components of *Time-HOBI*, show how a star query can be executed based on the *Time-HOBI* index, relate our index to a materialized view, and finally, we outline some alternative implementations of HOBI, TI, and PA.

4.1 Hierarchically Organized Bitmap Index

HOBI belongs to the class of bitmap join indexes as the index is defined on a dimension attribute and its bitmaps point to fact rows. *HOBI* is composed of bitmaps organized in a hierarchy that reflects the hierarchy of a dimension. Bitmaps on a lower level of a hierarchy are aggregated at an upper level.

Example 4. In order to illustrate the concept of *HOBI* let us consider dimension *Product*, such that *Products* → *Categories* and the dimension instance, as shown in Fig. 3. For this dimension, *HOBI* consists of two levels. At the lower level - *Products* there exist 7 bitmaps, each of which describes auction sales of one product, i.e., 'Asus Zenbook', 'Dell XPS Duo', etc. At the upper level - *Categories* there exist 2 bitmaps, i.e., 'Ultrabook' and 'Tablet', one bitmap for one category of sold products. The bitmaps are illustrated in Fig. reffig:TimeHobiExample in the box entitled "HOBI for dimension Product".

The upper level bitmap for 'Ultrabook', at level *Categories*, is computed by OR-ing the four bitmaps from level *Products*, i.e., 'Asus Zenbook', 'Dell XPS Duo', 'Toshiba Portege Z930-14T', and 'Sony VAIO SVT1313S1E'. Similarly, the 'Tablet' bitmap describes auction sales of products from this category and it is constructed by OR-ing bitmaps for 'iPad mini', 'Samsung Galaxy Note', and 'Asus Vivio Tab'.

4.2 Time Index

The *Time* dimension plays a special role as it is used in most of the star queries. In order to eliminate the frequent join operation of a fact table with the *Time* dimension, in [36] we proposed to implicitly encode the *Time* dimension in other dimensions. Similarly as in [1,19,33] we assume that data stored in a fact table

are sorted by a selected attribute, typically storing time. This assumption is realistic since a DW is loaded incrementally in time intervals. Moreover, data can be easily sorted by time in the ETL layer before being loaded into a DW. The *Time Index* (TI) takes advantage of data ordering by time. It is created on an attribute used to join a fact table with the *Time* dimension. *TI* stores ranges of bit numbers belonging to a given time interval. The time intervals in *TI* are identical as in the *Time* dimension.

The concept of *TI* is illustrated in Fig. 8. We assume that the *Time* dimension is composed of the following implicit hierarchy $Days \rightarrow Months \rightarrow Years$. Dimension D_i has only one denormalized level L with k instances. Thus, *HOBI* defined for D_i is composed of k bitmaps (denoted as $B_1, \ldots, B_k$) and they describe rows in a fact table. Let us assume that there are z such rows in the fact table. Therefore, every bitmap in *HOBI* is composed of z bits.

TI organizes bits in the bitmaps into intervals (segments) defined in the *Time* dimension. Thus, bits $b_1, \ldots, b_i$ point to fact rows that come from day_1, bits $b_{i+1}, \ldots, b_{i+x}$ point to fact rows that come from day_2, etc. Moreover, $day_1, \ldots, day_n$ aggregate to $month_1$. For this reason, bits $b_1, \ldots, b_{j+x}$ point to fact rows that come from $month_1$. Similarly, $month_1, month_2, \ldots, month_{12}$ aggregate to $year_1$ and bits b_1 to b_{o+x} point to fact rows that come from $year_1$.

Notice that: (1) all the bitmaps point to the same number of rows in a fact table, i.e., the length of every bitmap is identical, and (2) all the bitmaps in *HOBI* are divided into identical time intervals. For these reasons, *TI* is shared by all bitmaps in *HOBI*. *Time Index* eliminates the joins of a fact table with dimension *Time* as bit numbers representing fact rows that fulfill selection criteria on time may be easily retrieved with the support of *TI*.

Example 5. In order to illustrate the concept of *Time Index* let us consider 10 auctions (stored in the table *Auctions*) held on some days in months from February until July in the year 2010, as shown in Fig. 9. The *TI* maps the 9 distinct dates in the *Time* dimension into bit numbers. As the *Auctions* fact table stores 10 rows, every bitmap in *HOBI* includes 10 bits. Thus, the date '25-Feb-2010' maps to bit b_1, '2-Mar-2010' maps to the range of bits b_2–b_3, etc. At the level *Months*, 'February' maps to bit b_1, 'March' maps to the range of bits b_2–b_4, 'April' maps to b_5–b_6, etc. At the level *Year*, '2010' maps to b_1–b_{10}.

4.3 Partial Aggregates

Inspired by the concepts of Small Materialized Aggregates [33], Zone Maps [1], and Zone Filters [19] (Sect. 6), we augmented *HOBI* and *TI* with sets of aggregates that we call *Partial Aggregates* (PA). *PA* are computed for: (1) selected measures, (2) selected aggregation functions, (3) a given dimension, (4) a given dimension level, (5) a given dimension level instances, (6) a given time interval. The aggregates are associated with *HOBI* and they also have a hierarchical structure, which is identical to the structure of *HOBI*.

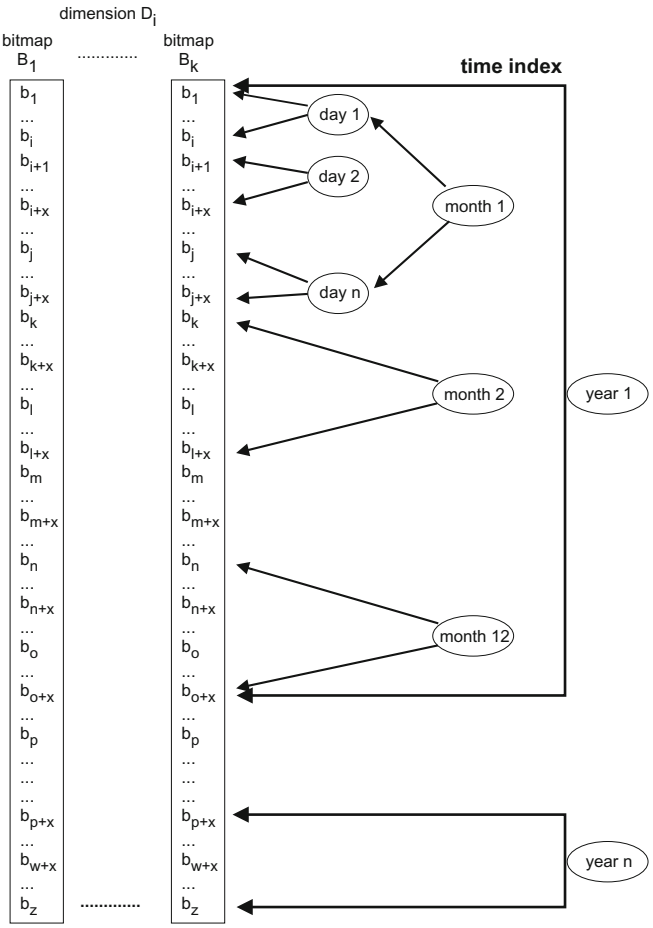


Fig. 8. The concept of Time Index

Let, in a given data warehouse schema:

- $\mathbb{M}$ denote the set of measures (e.g., *price*, *quantity*) and $m_i \in \mathbb{M}$,
- $\mathbb{AF}$ denote the set of aggregate functions (e.g., min, max, avg, sum, count) and $aggF_i \in \mathbb{AF}$,
- $\mathbb{D}$ denote the set of dimensions (e.g., *Time*, *Product*, *Location*) and $d_i \in \mathbb{D}$,
- $\mathbb{DL}$ denote the set of dimension levels (e.g., *Products*, *Categories*, *Cities*, *Regions*, *Countries*) and $dl_i \in \mathbb{DL}$,
- $\mathbb{LI}$ denote the set of dimension level instances (e.g., 'Ultrabook', 'Asus Zenbook', 'iTablet', 'iPad mini', 'Poznan', 'Warsaw') and $li_i \in \mathbb{LI}$,
- $\mathbb{TI}$ denote the set of time intervals in the *Time* dimension (e.g., 01-JAN-2013, JAN-2013, 2013) and $t_i \in \mathbb{TI}$.

Then, formally *PA* is a function $F : (m_i, aggF_i, d_i, dl_i, li_i, t_i) \mapsto v$ $(v \in \mathbb{R})$.

For example, F: (*price*, *sum*, *Location*, *Cities*, $Warsaw$, $MAR-2013$) maps to the aggregate - sum of sales prices in 'Warsaw' (in dimension *Location*, level *Cities*) in March 2013.

Partial Aggregates may be used:

- for selecting these segments of bitmaps that fulfill selection criteria based on aggregates and intersect the segments with segments selected by selection criteria defined by the intervals from the *Time* dimension, e.g.,

```
SELECT ...
WHERE sum(price) > 5000 AND yearID=2013
GROUP BY monthNr
```

- for computing aggregates on an upper level based on aggregates from a lower level of a dimension hierarchy (for distributive and algebraic aggregate functions), e.g., based on aggregate sales price per product in JAN 2013 computing aggregate sales per product category in the same time interval,
- in aggregate queries without selection predicates, e.g.,

```
SELECT sum(price), c.countryName
FROM Auctions a, Countries c, ...
GROUP BY c.countryName
```

- in aggregate queries with multiple selection predicates defined by means of dimensions, like for example Q3.

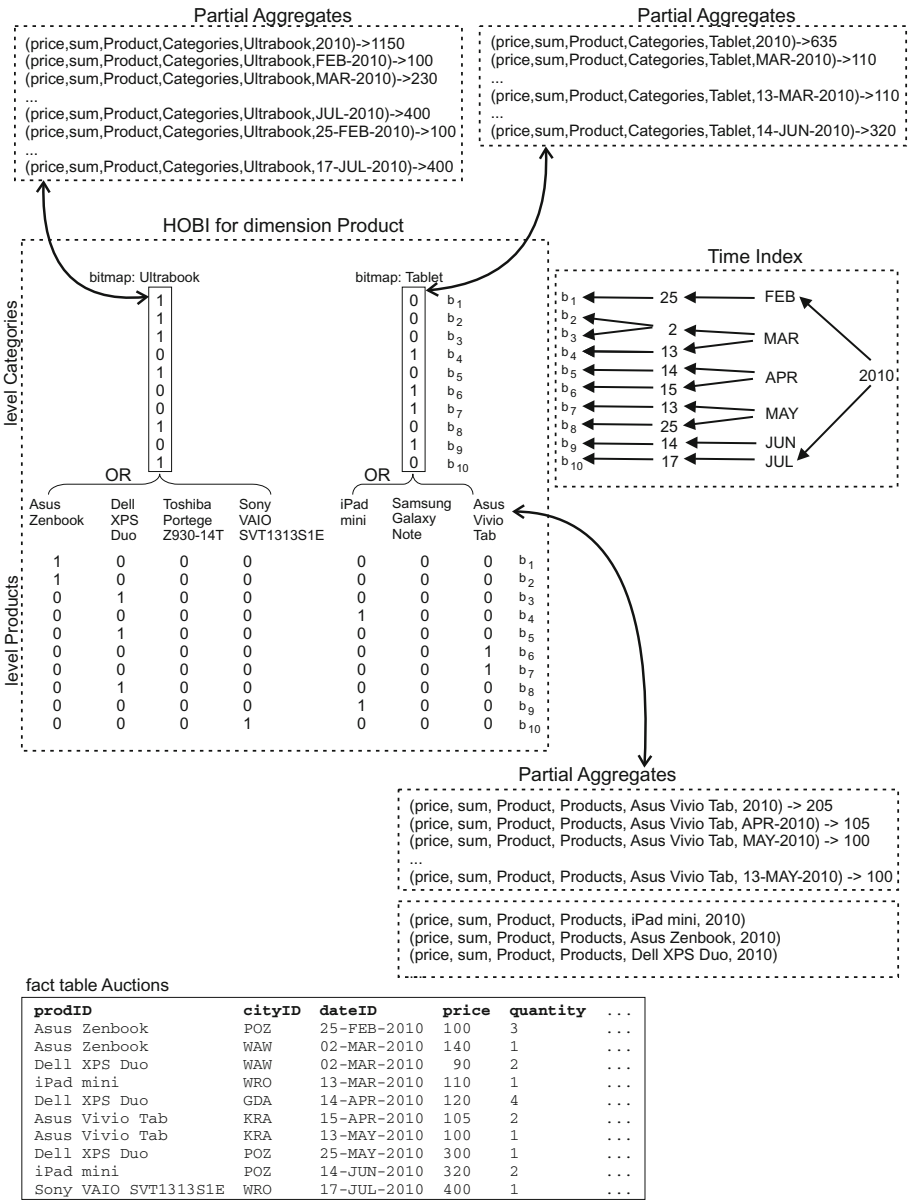
The application of *Time-HOBI* to index the tables in our example schema (Fig. 1) is shown in Fig. 9. We assume that fact table *Auctions* stores 10 rows. *HOBI* for dimension *Product* is composed of 7 bitmaps stored on level *Products* and 2 bitmaps on level *Categories*, as explained in Sect. 4.1. Since every bitmap is composed of 10 bits, *Time Index* points to 10 rows. Auction row from 25-FEB-2010 is represented by bit b_1, auction rows from 02-MAR-2010 are represented by bits b_2 and b_3. Auction rows from March 2010 are represented by bits $b_2, \ldots, b_4$, etc.

Following the definition of *Partial Aggregates*, they are stored for every bitmap and for every time interval in *Time Index*, i.e., day, month, and year. For example, at the level of bitmap 'Asus Vivio Tab', for 13-MAY-2010 there exist the following partial aggregate: *(price, sum, Product, Products, Asus Vivio Tab,13-MAY-2010)* $\rightarrow$ 100, for MAY-2010 there exist the following partial aggregate: *(price, sum, Product, Products, Asus Vivio Tab,MAY-2010)* $\rightarrow$ 100, and for the whole year 2010 there exist the following partial aggregate: *(price, sum, Product, Products, Asus Vivio Tab, 2010)* $\rightarrow$ 205. For simplicity reasons we assumed that only the *price* measure is aggregated by the SUM aggregate function.

Notice that *HOBI* (being the part of *Time-HOBI*) is applicable to any dimension but *Time*, since the *Time* dimension is used to organize bits in *HOBI*.

4.4 Star Query Optimization with the Support of Time-HOBI

As we have shown in Sect. 3.4, the optimization of a simple star query Q3 requires a complex execution plan with 6 joins. While applying *Time-HOBI*, we could optimize the query more efficiently since *Time-HOBI*:



fact table Auctions

prodID	cityID	dateID	price	quantity	...
Asus Zenbook	POZ	25-FEB-2010	100	3	...
Asus Zenbook	WAW	02-MAR-2010	140	1	...
Dell XPS Duo	WAW	02-MAR-2010	90	2	...
iPad mini	WRO	13-MAR-2010	110	1	...
Dell XPS Duo	GDA	14-APR-2010	120	4	...
Asus Vivio Tab	KRA	15-APR-2010	105	2	...
Asus Vivio Tab	KRA	13-MAY-2010	100	1	...
Dell XPS Duo	POZ	25-MAY-2010	300	1	...
iPad mini	POZ	14-JUN-2010	320	2	...
Sony VAIO SVT1313S1E	WRO	17-JUL-2010	400	1	...

Fig. 9. *Time-HOBI* for the *Product* dimension

– eliminates joins of a fact table with the *Time* by applying *TI*;
– allows efficient processing of bitmaps in *HOBI*;
– allows to fetch only the segments of bitmaps that are relevant to a time period selected in a query, by applying *TI* to *HOBI*;
– eliminates joins of a fact table with other dimensions by applying *HOBI*;
– eliminates or reduces the costs of computing aggregates of measures at various levels of a dimension hierarchy by applying *PA*.

The theoretical execution plan of Q3 with the support of *Time-HOBI* is shown in Fig. 10. The WHERE clause included the following selection criteria, resulting in 7 % query selectivity:

```
WHERE
   y.yearID in (2010, 2011)
   AND countryName in ('Poland', 'Germany')
```

As it can be noticed, no joins are needed in this plan. The query can be answered by accessing the following *PA*:

– $(price, sum, Location, Countries, Poland, 2010)$,
– $(price, sum, Location, Countries, Poland, 2011)$,
– $(price, sum, Location, Countries, Germany, 2010)$,
– $(price, sum, Location, Countries, Germany, 2011)$.

Addressing the partial aggregates is symbolized by:

– $PA\text{-}Address(SUM)^{YEARS}_{year=2010\ or\ year=2011}$,
– $PA\text{-}Address(SUM)^{COUNTRIES}_{countryName='Poland'\ or\ countryName='Germany'}$,

whereas fetching the aggregates is symbolized by *PA-Fetch*.

4.5 *Time-HOBI* vs. Materialized View

Time-HOBI takes advantage of materialized partial aggregates. With this respect, our index is similar to a materialized view as it can be applied to answering queries

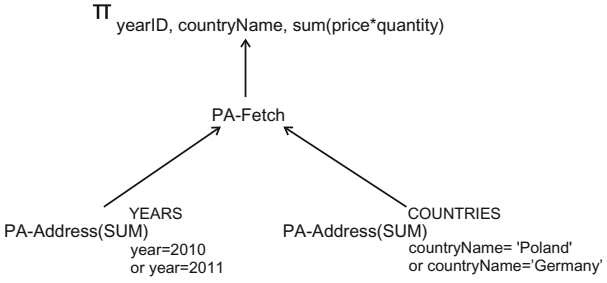


Fig. 10. Query execution plan for query Q3 constructed with the support of *Time-HOBI*

computing the aggregates being materialized in *PA*. It also associates the aggregates with their proper dimension levels and organizes the access to these aggregates by means of *PA*. Unlike a materialized view, *Time-HOBI* additionally can facilitate the execution of queries that filter fact data based on restrictions defined at various levels of dimensions. It is done by means of *HOBI* - being a kind of bitmap join index. In such cases, queries can be executed more efficiently with the support of *Time-HOBI* than with the support of bitmap, join, or bitmap join indexes, as it was shown in [36] and in Sect. 5. Finally, *Time-HOBI* eliminates the need to join a fact table with the *Time* dimension, especially in snowflake schemas. Such queries take advantage of *Time Index* in their execution plans. To this end, the following star query processing algorithm is applied.

Star query processing with the support of *Time-HOBI*

1. By means of the *Time Index* find the bit numbers that correspond to the given time interval $< t_k, t_{k+m} >$; let $[b_k : b_{k+m}]$ denote the corresponding range of bit numbers.
2. $\forall i = (j, \ldots, j+m)$ fetch fragments of bitmaps d_i pointed to by bit numbers $[b_k : b_{k+m}]$; let the fetched fragments be denoted as $f(d_j)$ for dimension instance d_j and $f(d_{j+m})$ for dimension instance d_{j+m}.
3. Compute the final bitmap fragment F from $f(d_i)$ $(i = (j, \ldots, j+m))$ by applying logical operators that were defined in the *where* clause of the star query.
4. $\forall b_i \in [b_k : b_{k+m}]$: if the bit value is equal to 1, then transform b_i into the physical addresses of the corresponding row and fetch the row.

To sum up, we believe that *Time-HOBI* offers slightly more flexibility than materialized views as it combines materialized aggregates at multiple levels of dimension hierarchies, bitmap join indexes, and encodes time on other dimensions.

4.6 Implementation Issues

There are three major implementation issues that impact query performance with the support of *Time-HOBI*, namely: (1) organizing access to bitmaps in *HOBI*, (2) implementing *Time Index*, and (3) organizing access to *Partial Aggregates*.

In the simplest case, when the domain of an indexed attribute is narrow, *HOBI* bitmaps can be accessed by means of an array or list sorted by the bitmap name (i.e., the value of the indexed attribute). In either of these implementations, an element of an array cell or a list stores: (1) the value of an indexed attribute and (2) a pointer to an appropriate bitmap. For wide domains, an access to the bitmaps may be organized either as a B-tree-like index or hash function.

Regarding *Time Index*, a crucial implementation issue is to organize ranges of bits for every time interval defined in the *Time* dimension. There are two straightforward methods of implementing *TI*. The first one is based on a record

structure and the second one is based on a tree structure. In a *record-based* implementation every instance of the *Time* dimension is stored as a record in a table. The record-based implementation can be altered in order to use a nested table (in the spirit of Oracle), where the days records are nested in months, which in turn are nested in a year record. A record-based and a nested table-based implementations can be further indexed in order to reduce access time to data. In the *tree-based* implementation, ranges of bits in the *Time Index* are organized in n trees, where n is equal to the number of years in the *Time* dimension. Each tree has a root that stores the year. A root points to the lower level time intervals, e.g., months, that in turn point to the lower level time intervals, e.g., days. This implementation is visualized in Fig. 9.

A straightforward storage implementation of *Partial Aggregates* is a table with columns $m_i, aggF_i, d_i, dl_i, li_i, t_i, v$ (Sect. 4.3). The access to the aggregates may be organized by an index either on all the columns but v or on the leading columns. Alternatively, *PA* may be accessed by a hash function on the same columns. *PA* may be implemented also as a n-dimensional array, dimensioned by $m_i, aggF_i, d_i, dl_i, li_i, t_i$, with the cells storing v. Thus, the values of the dimensions are used as indexes to the cells of interest.

4.7 Time-HOBI Limitations

As already mention, we assumed that data in a fact table must be sorted by the value of an attribute storing time. The values of the ordering time attribute are used to construct *Time Index*. For this reason, *Time-HOBI* can only be created on the ordering time attribute.

For fine grained instances of the *Time* dimension, like seconds, milliseconds, etc., *Time Index* would include millions or billions of items, resulting in a huge and inefficient size. Moreover, the number of *PA* would also contributed to the size of the whole *Time-HOBI*. These issues needs further investigation in the future.

5 Experimental Evaluation

Time-HOBI was implemented as an application layer in Oracle11g that stored all the data structures discussed in Sect. 4. In the experiments we use a sorted list for *HOBI*, the record-based storage for *TI*, and table storage for *PA*.

The Oracle instance used 3.4 GB of SGA (i.e., the main memory allocated to handle various instance buffers), 1.7 GB of which was allocated to a data cache. The experiments were conducted on a computer equipped with: processor - Intel Core i7-820QM 1.7 GHz, disk - Seagate ST9320423AS, and 8 GB RAM, under Windows7 Server. The DW schemas for the experimental evaluation are shown in Figs. 1 and 2. In the snowflake schema, the tables included the following number of rows: *Years*: 6, *Months*: 72, *Days*: 2190, *Cities*: 10 000, *Regions*: 200, *Countries*: 10, *Categories*: 230, *Products*: 48 000, and *Auctions*: 500 000 000, of the total size equal 30 GB. In the star schema, the tables included the following

```
SELECT ci.cityName, d.dayName, d.dayNr,
   sum(a.price), sum(a.quantity)
FROM
   Auctions a, Cities ci, Days d
WHERE
   ci.cityName like 'pattern'
   AND a.cityID=ci.cityID
   AND a.dateID=d.dateID
GROUP BY
   ci.cityName, d.dayName, d.dayNr
```

Fig. 11. The example query Q1 used in the experiments for the snowflake and star schemas

number of rows: *Days*: 2190, *Cities*: 10 000, *Products*: 48 000, and *Auctions*: 500 000 000, of the total size slightly over 30 GB, due to the redundancy in the dimension tables. The data used in the experiments were artificially generated but data distributions reflected real data that we used in [12].

In this schema we run five different query patterns, as shown in Fig. 1 and mentioned in Sect. 2.2 as well as the equivalents of the query patterns in the star schema. For example, the pattern of Q1 is shown in Fig. 11. Its selectivity is parameterized in the WHERE clause by means of attribute *Cities.cityName*. Notice that Q1 is identical for the snowflake and the star schema.

The patterns of Q3 for the snowflake and the star schema are shown in Figs. 12 and 13, respectively. Their selectivities are parameterized in the WHERE clause by means of attributes *countryName* and *yearID*. Since the filtering predicates are defined by means of the highest levels (i.e., *Countries*, and *Years*) of the *Location* and *Time* dimensions, the lowest possible selectivity for Q3 is 1.6 %.

In the snowflake schema, the Q2 query pattern computes the aggregates *sum(a.price)* and *sum(a.quantity)* at the levels of *Months* and *Regions* (cf. Fig. 1. It is parameterized by means of attributes *Regions.regionName* and *Months.*

```
SELECT y.yearID, co.countryName,
   sum(a.price), sum(a.quantity)
FROM
   Auctions a,
   Days d, Months m, Years r,
   Cities ci, Regions r, Countries co
WHERE
   NOT (co.countryName like 'pattern')
   AND (y.yearID > v_year)
   AND a.cityID=ci.cityID
   AND ci.regionID=r.regionID
   AND r.countryID=co.countryID
   AND a.dateID=d.dateID
   AND d.monthID=m.monthID
   AND m.yearID=y.yearID
GROUP BY
   y.yearID, co.countryName
```

Fig. 12. The pattern Q3 in the snowflake schema

```
SELECT d.yearID, ci.countryName,
   sum(a.price), sum(a.quantity)
FROM
   Auctions a, Days d, Cities ci
WHERE
   NOT (ci.countryName like 'pattern')
   AND (d.yearID > v_year)
   AND a.cityID=ci.cityID
   AND a.dateID=d.dateID
GROUP BY
   d.yearID, ci.countryName
```

Fig. 13. The pattern Q3 in the star schema

monthNr. Pattern Q21 computes the same aggregates at the levels of *Days* and *Regions* and it is parameterized by means of attributes *Regions.regionName* and *Days.dayNr*. Pattern Q31 computes the same aggregates as Q2 but at the levels of *Days* and *Countries*. It is parameterized by *Countries.countryName* and *Days.dayNo*.

In the star schema, the query patterns Q2, Q21, and Q3 compute the same aggregates as their equivalents in the snowflake schema, however, *Auctions* is joined with the denormalized dimension tables *Days* and *Cities*.

In the experiments we decided no to use any of the standard benchmarks like TPC-H, TPC-DS [3], SSB [40], and [2] for two reasons. First, because the benchmarks are typically designed to measure an overall system's response time and query processing efficiency for given query workloads. We found that it is inadequate to our setting where we aimed at comparing the performance of particular indexes with respect to parameterized selectivities of the queries and parameterized number of joins. Second, the patterns of queries that we applied in our experiments reflect real queries that were run in a real system [12] that we modeled with the snowflake and the star schema.

The experiments that we conducted aimed at comparing the following characteristics of *Time-HOBI* with respect to:

1. the performance of the aggregate query patterns Q1, Q2, Q21, Q3, and Q31,
2. index sizes,
3. index creation times.

We related the three characteristics to the competitors being:

- Oracle indexes in the snowflake schema,
- Oracle indexes in the star schema,
- Oracle materialized views in the snowflake schema,
- Oracle materialized views in the star schema.

In each of the experiments described below, for comparison, we defined the following Oracle indexes:

- the bitmap index on attribute *yearID* in both the snowflake and the star schema,
- the bitmap index on attribute *countryName* in both the snowflake and the star schema,
- the concatenated bitmap join index on *year* and *countryName* that joined tables *Years*, *Months*, *Days*, *Countries*, *Regions*, *Cities*, and *Auctions* in the snowflake schema,
- the bitmap index on attribute *monthNr* in both the snowflake and the star schema,
- the bitmap index on attribute *regionName* in both the snowflake and the star schema,
- the concatenated bitmap join index on *monthNr* and *regionName* that joined tables *Months*, *Days*, *Regions*, *Cities*, and *Auctions* in the snowflake schema,

- the bitmap index on attribute *dayNr* in both the snowflake and the star schema,
- the bitmap index on attribute *cityName* in both the snowflake and the star schema,
- the concatenated bitmap join index on *dayNr* and *cityName* that joined tables *Days*, *Cities*, and *Auctions* in the snowflake schema,
- the concatenated bitmap join index on *yearID*, *monthNr*, *dayNr*, *countryName*, *regionName* and *cityName* that joined tables *Locations*, *Time* and *Auctions* in the star schema.

Additionally, we created the set of materialized views in the snowflake schema. Their structures are shown in Fig. 14. The corresponding views in the star schema computed the same aggregates.

```
CREATE MATERIALIZED VIEW MV_A_CITY_DAY ... AS
SELECT ci.cityId, d.dateId, ci.cityName, d.dayName, d.dayNr,
  sum(a.price) as pricesum, sum(a.quantity) as quantitysum, count(*) as count
FROM Auctions a, Cities ci, Days d
WHERE a.cityId=ci.cityId AND a.dateId=d.dateId
GROUP BY ci.cityId, d.dateId, ci.cityName, d.dayName, d.dayNr

CREATE MATERIALIZED VIEW MV_A_REGION_MONTH ... AS
SELECT r.regionID, m.monthID, r.regionName, m.monthNr,
  sum(a.pricesum) as pricesum, sum(a.quantitysum) as quantitysum, sum(a.count) as count
FROM MV_A_CITY_DAY a, Cities ci, Regions r, Says d, Months m
WHERE a.cityID=ci.cityID AND a.dateID=d.dateID AND ci.regionID=r.regionID
  AND d.monthID=m.monthID
GROUP BY r.regionID, m.monthID, r.regionName, m.monthNr

CREATE MATERIALIZED VIEW MV_A_COUNTRY_YEAR ... AS
SELECT c.countryID, y.yearID, c.countryName,
  sum(a.pricesum) as pricesum, sum(a.quantitysum) as quantitysum, sum(a.count) as count
FROM MV_A_REGION_MONTH a, Regions r, Months m, Countries c, Years r
WHERE a.regionID=r.regionID AND a.monthID=m.monthID AND r.coutnryID=c.coutnryID
  AND m.yearID=y.yearID
GROUP BY c.coutnryID, y.yearID, c.countryName

CREATE MATERIALIZED VIEW MV_A_LOCATION_TIME ... AS
SELECT l.countryName, l.regionName, l.cityName, locationID,
  t.year, t.monthNr, t.dayNr, timeID,
  sum(a.price) as pricesum, sum(a.quantity) as quantitysum, count(*) as count
FROM Auctions a, Time t, Locations l
WHERE a.timeid=t.timeid AND a.locationID=l.locationID
GROUP BY l.countryName, l.regionName, l.cityName, locationID,
  t.year, t.monthNr, t.dayNr, timeID

CREATE MATERIALIZED VIEW MV_A_LOCATION_TIME_R2 ... AS
SELECT countryName, regionName, yearID, monthNr,
  sum(pricesum) as pricesum, sum(quantitysum) as quantitysum, sum(count) as count
FROM MV_A_LOCATION_TIME
GROUP BY countryName, regionName, yearID, monthNr

CREATE MATERIALIZED VIEW MV_A_LOCATION_TIME_R3 ... AS
SELECT countryName, yearID, sum(pricesum) as pricesum,
  sum(quantitysum) as quantitysum, sum(count) as count
FROM MV_A_LOCATION_TIME_R2
GROUP BY countryName, yearID
```

Fig. 14. The materialized views created in the snowflake schema

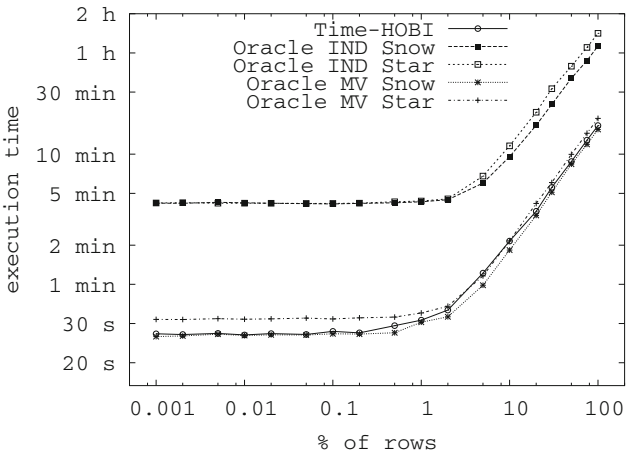


Fig. 15. Elapsed execution times of query pattern Q1 with the parameterized query selectivity

5.1 Query Performance

In these experiments we measured the performance characteristics of the indexes for query patterns Q1, Q2, Q21, Q3, and Q31. The selectivities of these queries were parameterized and ranged from 0.001 % to 100 % of rows in the *Auction* fact table. *Time-HOBI* indexes were defined in every dimension used in the queries, thus the indexes included the *Partial Aggregates* by (Product, Time) as well as (Location, Time), for all the levels in these dimensions. *PA* were created in the snowflake and the star schema.

The obtained results for query pattern Q1 are shown in Fig. 15. Notice that the query performance with the support of *Time-HOBI* is the same in the snowflake and star schema. It is because, the structure of the index is the same for both of the schemas. This observation is true also for the rest of the test queries.

From Fig. 15 we can observe that the elapsed execution times of the queries are much lower with the support of *Time-HOBI* than with the support of the set of Oracle indexes, for the whole range of the query selectivity. Moreover, the execution times of the queries are almost the same for *Time-HOBI* and the materialized views, for the same range of the selectivity.

The performance characteristics of query patterns Q2, Q21, Q3, and Q31 are shown in Figs. 16, 17, 18, and 19. From the figures we can observe that *Time-HOBI* also offers better performance than the Oracle indexes in both DW schemas and offers similar performance to the Oracle materialized views.

For the above characteristics we computed ratio $\gamma = \frac{t^{IND}_{Oracle}}{t_{Time-HOBI}}$, where t^{IND}_{Oracle} and $t_{Time-HOBI}$ denote elapsed query execution times with the support of the set of Oracle indexes and *Time-HOBI*, respectively. Similarly, we computed

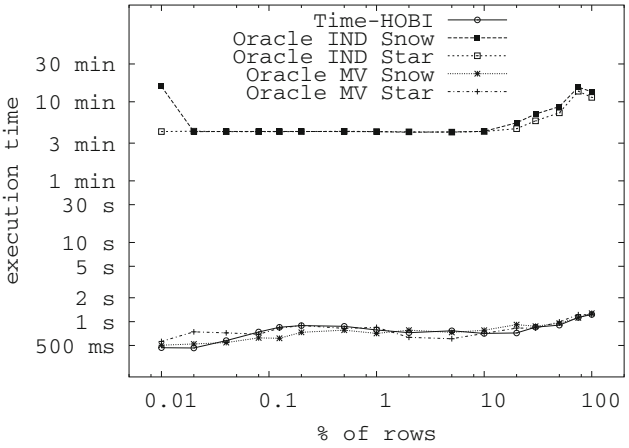


Fig. 16. Elapsed execution times of query pattern Q2 with the parameterized query selectivity

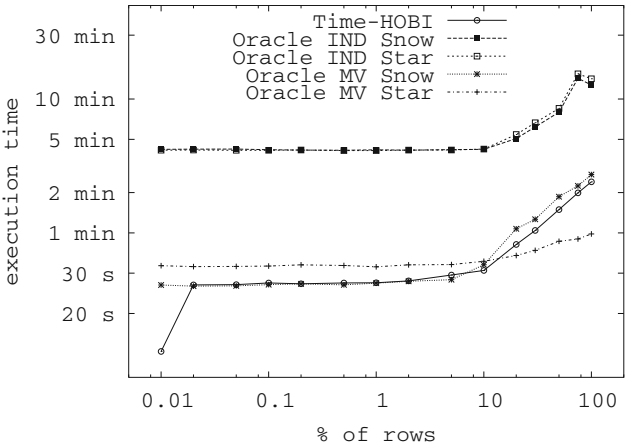


Fig. 17. Elapsed execution times of query pattern Q21 with the parameterized query selectivity

ratio $\lambda \frac{t^{MV}_{Oracle}}{t_{Time-HOBI}}$, where t^{MV}_{Oracle} denotes elapsed query execution times with the support of the set of Oracle materialized views.

The minimum and maximum values of the ratios γ and λ obtained for query patterns Q1, Q2, Q21, Q3, and Q31 are shown in Table 5. For every minimum and maximum value we indicated the query selectivity.

From the performance characteristics shown above we conclude that:

– *Time-HOBI* outperforms the Oracle indexes for aggregate queries, for the whole tested range of selectivities. As the experiments confirmed, this statement is true for the snowflake and the star schema.

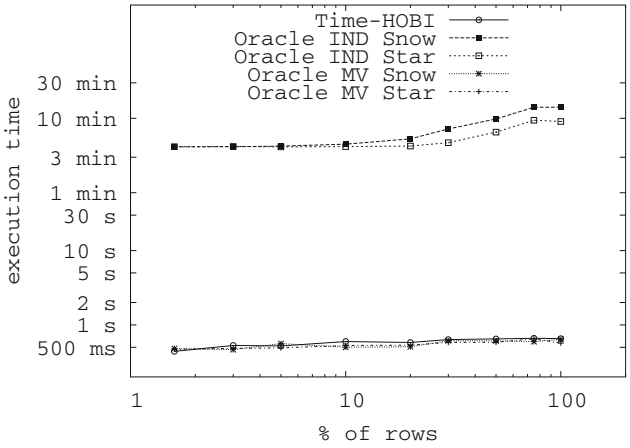


Fig. 18. Elapsed execution times of query pattern Q3 with the parameterized query selectivity

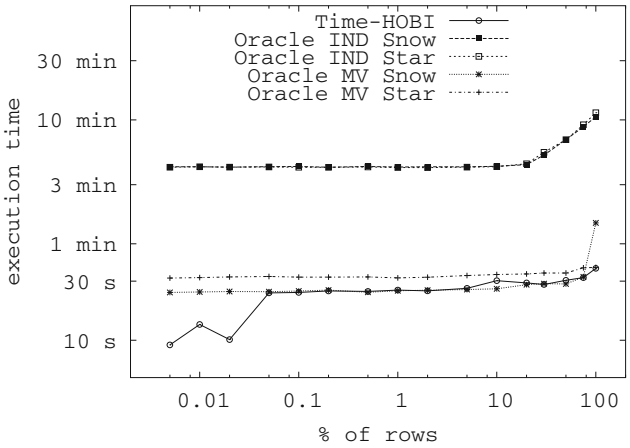


Fig. 19. Elapsed execution times of query pattern Q31 with the parameterized query selectivity

- *Time-HOBI* offers similar performance characteristics as the Oracle materialized views for aggregate queries, for the whole tested range of selectivities. As the experiments confirmed, this statement is true for the snowflake and the star schema.
- *Time-HOBI* offers the same query performance for the snowflake and the star DW schema as the structure of the index is the same for both of the schemas.

Table 5. The minimum and maximum values of the γ and λ ratios obtained for the test queries

	γ snow sch.		γ star sch.		λ snow sch.		λ star sch.	
	min	max	min	max	min	max	min	max
Q1	4.09	10.35	5.11	10.29	0.81	0.99	0.95	1.32
	sel.75–100 %	sel.0.01 %	sel.0.01 %	sel.0.01 %	sel.0.01 %	sel.0.005–0.05 %	sel.5 %	sel.0.05 %
Q2	283	1997	282	721	0.73	1.27	0.79	1.60
	sel.0.5 %	sel.0.01 %	sel.0.2 %	sel.75 %	sel.0.125 %	sel.20 %	sel.5 %	sel.0.02 %
Q21	5.29	32.37	5.71	31.90	0.92	3.14	0.41	4.37
	sel.100 %	sel.0.01 %	sel.50 %	sel.0.01 %	sel.5 %	sel.0.01 %	sel.100 %	sel.0.01 %
Q3	450	1292	417	853	0.84	1.09	0.87	1.07
	sel.50 %	sel.100 %	sel.10 %	sel.75 %	sel.10 %	sel.1.6 %	sel.100 %	sel.1.6 %
Q31	8.37	27.36	8.32	27.13	0.86	2.65	1.02	3.47
	sel.10 %	sel.0.005 %	sel.10 %	sel.0.005 %	sel.10 %	sel.0.005 %	sel.100 %	sel.0.005 %

5.2 Index Sizes

In these experiments we measured the sizes of *Time-HOBI* and compared them to the sizes of Oracle indexes and Oracle materialized views, for various data volumes being indexed (from 1 GB to 18 GB). The results are visualized in Fig. 20. As it can be observed from the figure, the size of *Time-HOBI* is larger than the size of the materialized views for the whole range of the fact table sizes and for both DW schema implementations. It is not surprising as *Time-HOBI* stores not only aggregates but also bitmaps. As compared to the Oracle indexes, in the snowflake schema our index is about 2.1 times larger in the whole tested range of DW size. In the star schema our index is about 1.5 times larger in the whole tested range of DW size. Table 6 shows the exact sizes of the tested data structures.

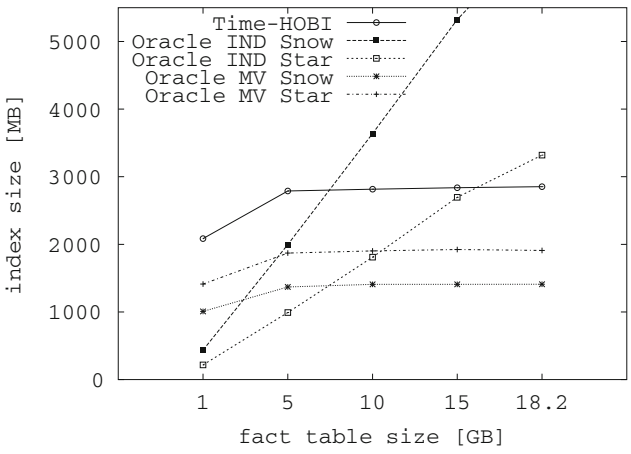


Fig. 20. The sizes of *Time-HOBI*, Oracle indexes, and Oracle materialized views for, a parameterized data volume

Table 6. The sizes of *Time-HOBI*, the Oracle indexes, and the Oracle materialized views, for the snowflake and the star schema

Data volume	PA [MB]	HOBI [MB]	Time-HOBI [MB]	Oracle IND snow [MB]	Oracle IND star [MB]	Oracle MV snow [MB]	Oracle MV star [MB]
1 [GB]	980	1105	2085	435	216	597	1412
5 [GB]	1310	1479	2789	1997	991	1644	1871
10 [GB]	1310	1505	2815	3638	1809	2508	1902
14 [GB]	1380	1458	2838	5324	2694	3356	1923
18 [GB]	1380	1472	2852	6644	3319	4026	1009

Recall that *Time-HOBI* stores three sets of data, i.e., time in *Time Index*, bitmaps in *HOBI*, and precomputed aggregated values of selected measures in *Partial Aggregates*. Table 6 shows the sizes of the aforementioned data structures, the Oracle indexes and materialized views, for five data volumes in the snowflake schema. The size of *TI* for 10 years time period is about 85 kB and it can be neglected, [36]. The size of *PA* should be constant for a given constant number of rows in dimension level tables. In our experiments, the size of *PA* changes slightly with the increase of the data volume, cf. Table 6. In our opinion it may be caused by a data allocation in database blocks (with different percentage free for different data volume sizes). Thus with the almost constant size of *PA* w.r.t. a data volume, for data volume sizes greater than 18 GB *Time-HOBI* will be smaller than the Oracle indexes, for both the snowflake and the star schema. Since the number of *PA* and the number of bitmaps in *HOBI* depends on the number of levels in dimensions and does not depend on the schema implementation, the size of *Time-HOBI* remains the same for the snowflake and star schema.

5.3 Index Creation Times

In these experiments we measured the creation times of *Time-HOBI* and compared them to creation times of Oracle indexes and Oracle materialized views, for various data volumes being indexed (from 1 GB to 18 GB). The results are shown in Fig. 21. The creation time characteristics are consistent with the index size characteristics, i.e., the larger the index is the longer it takes to create it. However, for data volumes larger than 5GB, *Time-HOBI* is created faster than the Oracle indexes. Moreover, its creation time is comparable to the creation time of the materialized views in the whole range of the test data volume.

As our index was implemented at an application layer, the procedure for creating *Time-HOBI* was not optimized and data volumes processed for creating *HOBI* were not shared while creating *PA*. In fact, the procedure executed joins on *Auctions* and its dimensions (*Time* and *Location*) twice (two independent queries). We believe that this procedure can be optimized at some extent, thus reducing the index creation time.

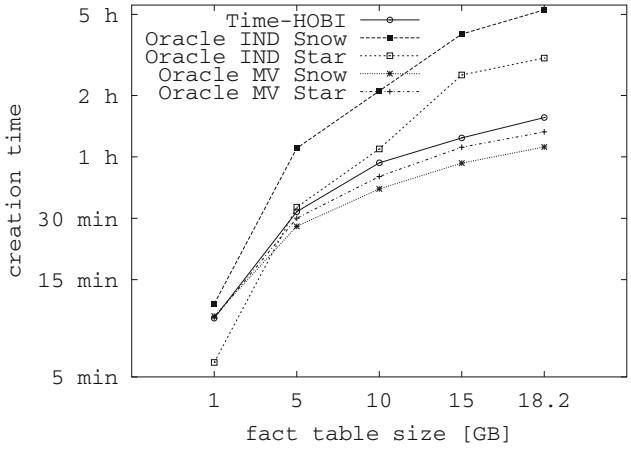


Fig. 21. Creation times of *Time-HOBI*, Oracle indexes, and Oracle materialized views, for a parameterized data volume

5.4 Experiments Summary

Query performance comparison of *Time-HOBI* to the Oracle indexes reveals that:

- the star queries that we tested in the snowflake DW schema were executed from 4 to 1292 times faster (depending on a query) with the support of *Time-HOBI*,
- the queries in the star DW schema were executed from 5 to 853 times faster with the support of our index.

Query performance comparison of *Time-HOBI* to the Oracle materialized views reveals that:

- the queries in the snowflake schema were executed in comparable times (λ ranges from 0.81 to 2.65),
- the queries in the star schema were executed also in comparable times (λ ranges from 0.79 to 4.37).

Size comparison of *Time-HOBI* to the Oracle indexes reveals that:

- in the snowflake schema our index is up to 4.8 times larger for a DW volume 1–7.5 GB, and it is up to 2.3 times smaller for a DW data volume 8–18 GB,
- in the star schema our index is up to 9.6 times larger for a DW volume 1–16 GB, and it is up to 1.2 times smaller for a DW data volume over 16 GB.

Size comparison of *Time-HOBI* to the Oracle materialized views reveals that:

- in the snowflake schema our index is about 2.1 times larger in the whole tested range of DW size,

– in the star schema our index is about 1.5 times larger in the whole tested range of DW size.

Creation time comparison of *Time-HOBI* to the Oracle indexes reveals that:

– in the snowflake schema our index is created from 1.18 to 3.37 times faster in the whole tested range of DW size,
– in the star schema our index is created from 1.17 to 2 times faster for a DW data volume greater than 5 GB.

Creation time comparison of *Time-HOBI* to the Oracle materialized views reveals that:

– in the snowflake schema our index is created from 1.2 to 1.4 times slower in the whole tested range of DW size,
– in the star schema our index is created from 1.1 to 1.5 times slower in the whole tested range of DW size.

6 Related Work

Assuring an efficient access to large volumes of data is an important research problem. Various physical structures have been proposed in the research literature to solve the problem. In this section we outline the physical structures that inspired the development of *Time-HOBI*. The structures include indexes and materialized aggregates.

6.1 Traditional Indexes

Various indexes have been proposed for different application domains. In relational databases and data warehouses, most widely applied index structures in practice include: B-tree like indexes, bitmap indexes, and join indexes. They gained popularity due to their relatively simple structures and maintenance algorithms. In geographical databases, various multi-dimensional indexes have been developed. For advanced data analysis in statistical databases, for data mining, as well as for object databases hierarchical indexes have been developed.

The indexes from the B-tree family [26] are efficient only in indexing data of high cardinalities (i.e., wide domains) and they well support queries of high selectivities (i.e., when few records fulfill query criteria). However, for OLAP queries that are often expressed on attributes of low cardinalities (i.e., narrow domains), B-tree indexes do not provide an acceptable performance. For this reason, for indexing data of low cardinalities, for efficient filtering large data volumes, and for supporting OLAP queries of low cardinalities, bitmap indexes have been developed (Sect. 3.2). A drawback of a bitmap index is that its size increases when the cardinality of an indexed attribute increases. As a consequence, bitmap indexes defined on attributes of high cardinalities become very large or too large to be efficiently processed in main memory [63]. In order to improve the efficiency of accessing data with the support of bitmap indexes

defined on attributes of high cardinalities, either different kinds of bitmap encodings have been proposed, e.g., [10,29,47,54,62] or compression techniques have been developed, e.g., [4,52,53,61].

For efficient executions of star queries, a join index was developed [31,39,57]. It can be perceived as the materialized join of a level table and a fact table. The index is created on a join attribute of a level table. The index is typically organized as a B-tree. It differs from a traditional B-tree with respect to the content of its leaves. The leaves of the join index store physical addresses of records from all the joined tables. An extension to the join index was proposed in [39] where the authors represented precomputed joins by means of bitmaps. The join index whose leaves store bitmaps rather than ROWIDs is called a bitmap join index. Bitmap join indexes provide an efficient optimization mechanism for star queries not only in traditional data warehouses but also in spatial data warehouses [8,51].

6.2 Multi-level Indexes

Concepts similar to the join index were developed for object databases for the purpose of optimizing queries that follow the chain of references from one object to another ($o_i \rightarrow o_{i+1} \rightarrow \ldots o_{i+n}$). Persistent (precomputed) chains of object references are stored either in an access support relation [28] or in a join index hierarchy [23]. In [66] two index structures for indexing hierarchies of classes were described. Both of them are based on tree-like structures.

In [34,35,49,50] indexes of multi-level structures have been proposed. A multi-resolution bitmap index was presented in [49,50] for the purpose of indexing scientific data. The index is composed of multiple levels. Lower levels are implemented as standard bitmap indexes offering exact data look-ups, while upper levels, are implemented as binned bitmaps, offering data look-ups with false positives. An upper level index (the binned one) is used for retrieving a dataset that totally fulfills query search criteria. A lower level index is used for fetching data from boundary ranges in the case when only some data from bins fulfill query criteria.

In [34,35], a hierarchical bitmap index was proposed for set-valued attributes for the purpose of optimizing subset, superset, and similarity queries. The index, being defined on a given attribute, consists of the set of index keys, where every key represents a single set of values. Every index key comprises signature S. The length of the signature, i.e. the number of bits, is equal to the size of the domain of the indexed attribute. S is divided into n-bit chunks (called index key leaves) and the set of inner nodes. Index key leaves and inner nodes are organized into a tree structure. Every element from the indexed set is represented once in the signature by assigning value '1' to an appropriate bit in an appropriate index key leaf. The next level of the index key stores information only about these index key leaves that contain '1' on at least one position. A single bit in an inner node represents a single index key leaf. If the bit is set to '1' then the corresponding index key leaf contains at least one bit set to '1'. The i-th index key leaf is represented by j-th position in the k-th inner node, where $k = \lceil i/l \rceil$

and $j = i - (\lceil i/l \rceil - 1) * l$. Every upper level of the inner nodes represents the lower level in an analogous way. This procedure repeats recursively up to the root of the tree.

6.3 Multidimensional Indexes

Since more than 40 years of research in this domain a few dozens of multidimensional indexes have been proposed, including the most frequently applied families of R-trees [43,44,64] grid files, and K-D-trees. Excellent overviews of existing multidimensional indexes have been proposed in [7,18]. Most of the indexes have been designed for the support of access methods to spatial data in geographical databases, mostly in a two dimensional space. Multidimensional indexes are also applied to supporting top-k (e.g., [65]), k-NN queries (e.g., [67]), and data mining (e.g., [30]).

Most of the multi-dimensional indexes are very complex data structures. They are difficult to implement and to maintain and sometimes their complexity penalizes the performance. For these reasons, some research efforts focus on mapping the multidimensional indexes into relational DBMSs [7,16].

6.4 Materialized Aggregates

The second data storage structure that inspired our work on *Time-HOBI* are materialized aggregates. Two types of such aggregates can be distinguished, namely: (1) materialized views and (2) summary data. A *materialized view* is a precomputed query whose result is stored in a database. Typically, various data warehouse queries take advantage of materialized views in the process of query rewriting. An overview of multiple research problems on materialized views can be found in [20].

Summary data are materialized sets of aggregates, typically associated with storage units of data, e.g., segments, extents (like in Oracle), buckets [33], or zones [1,19]. The first concept, called *Small Materialized Aggregates* (SMA) was proposed in [33]. The concept of SMA assumes that data are ordered on disk by the value of a selected attribute, typically date. Physically, a disk is divided into logical storage units called buckets. Every bucket stores at most n rows. SMA is associated with each bucket. For each bucket, SMA typically include aggregates like minimum and maximum value of the ordering attribute as well as the number of rows in the bucket. The min and max values allow to check whether the bucket fulfills selection criteria on an ordering attribute. If so, the bucket is fetched from disk, otherwise it is skipped.

A concept similar to SMA, called *Zone Maps*, was implemented in Netezza [1]. Netezza organizes disk space in zones, each of which has its own zone map. The zone map includes minimum and maximum values of every attribute of a table stored in the zone. Zone maps are used for verifying whether a given zone qualifies to be accessed by a query.

Finally, [19] extends the two aforementioned concepts by means of *zone filters* and *zone indexes*. The first mechanism generalize SMA and zone maps. It is

similar to zone maps but it stores n minimum and n maximum values of every attribute in a zone (where $n>2$). An separate index - a zone index is dedicated to every zone to facilitate searching data within the zone. All the aforementioned structures assume that the summary data are maintained within by a process that loads a data warehouse.

Although SMA, zone maps, and [19] inspired the development of *PA*, it differs from these three concept as *PA* store aggregated values of measures at all levels of dimension hierarchies, whereas in the three concepts minimum and maximum values of attribute values are stored for every bucket/zone. Moreover, *PA* organizes the aggregates in hierarchies, whereas the three concepts not.

6.5 The Missing Functionality

From the index structures discussed above none was proposed for indexing hierarchical dimensional data in a data warehouse. Moreover, none of them reflects the hierarchy of dimensions. Such a feature may be useful for computing aggregates in an upper level of a dimension based on data computed for a lower level. Furthermore, none of them exploits the fact that the *Time* dimension is used in most of the star queries in predicates and is used for aggregating measures.

From commercial DBMSs, IBM DB2 supports a cluster index and a multidimensional cluster. Both data structures use B-tree based indexes to order data by the values of selected attributes. Oracle11g supports a sorted hash cluster used for the same purpose. Nonetheless, none of these data structures support indexing hierarchical dimensions.

7 Summary

In this chapter we gave an overview of the indexes typically applied to the optimization of star queries in a data warehouse, i.e., the bitmap, join, and bitmap join indexes. We showed how an example star query is executed in Oracle, which motivated us to develop an alternative index structure. We also presented the previously developed *Time-HOBI* index. This chapter introduced as additional contributions: (1) an extension to *Time-HOBI* by means of *Partial Aggregates* and (2) experimental evaluation of the extended *Time-HOBI*. The performance of the extended *Time-HOBI* was compared to the Oracle bitmap and bitmap join indexes as well as to materialized views, for the snowflake and the star DW schema.

In summary, *Time-HOBI* offers the following functionality:

- it eliminates joins of a fact table with the *Time* dimension as *Time Index* stores bit ranges for all the time intervals in the *Time* dimension;
- it eliminates joins of a fact table with other dimensions as *HOBI* stores bitmaps at all levels of a given dimension and the bitmaps point to rows in a fact table;

– it eliminates or reduces the costs of computing aggregates of measures at various levels of a dimension hierarchy as *Partial Aggregates* provide access to precomputed aggregates;
– it offers the same query performance for the snowflake and the star schema as the structure of *Time-HOBI* is the same for both of the schemas.

The experimental evaluation of *Time-HOBI* shows that:

– the index outperforms the Oracle indexes for aggregate queries, for the whole tested range of selectivities, which is true for the snowflake and the star schema;
– the index offers similar performance characteristics as the Oracle materialized views for aggregate queries, for the whole tested range of selectivities, which is true for the snowflake and the star schema;
– for the DW data volume over 7.5 GB - for the snowflake and over 16 GB - for the star schema the size of our index is smaller than the size of the Oracle indexes;
– *Time-HOBI* is larger from 2 to 3.4 times than the set of materialized views;
– our index was created from 1.1 to 1.5 times slower than the set of materialized views (as the size impacts the creation time), however, *Time-HOBI* was created from 1.17 to 3.37 times faster than the Oracle indexes.

Notice that for the experiments *Time-HOBI* was implemented as an application on top of DBMS Oracle, resulting in some additional time overheads. One may expect yet better performance while building the index in the DBMS so that it could be efficiently managed by a system and used by a query optimizer. Our ongoing work is focused on developing efficient algorithms for maintaining all the components of *Time-HOBI* as well as on embedding the index and the maintenance algorithms in the Oracle DBMS by means of the data cartridge technology [56]. Next, we plan to evaluate the performance of the implementation w.r.t. query processing and maintenance. Our research in the future will concentrate also on analyzing the impact of fine grained *Time* dimensions on the size and performance of the index.

Acknowledgment. This work was supported from the Polish National Science Center (NCN), grant No. 2011/01/B/ST6/05169. The authors express their gratitude to the anonymous Reviewers whose very thorough comments greatly improved the quality of this paper.

References

1. Netezza underground: zone maps and data power. www.ibm.com/developerworks/community/blogs/Netezza/entry/zone_maps_and_data_power20?lang=en. Accessed 27 June 2013
2. OLAP council APB-1 OLAP benchmark release 2. www.olapcouncil.org/research/APB1R2_spec.pdf. Accessed 20 Dec 2012

3. Transaction processing performance council. www.tpc.org/information/benchmarks.asp
4. Antoshenkov, G., Ziauddin, M.: Query processing and optimization in Oracle RDB. Int. J. Very Larg. Data Bases **5**(4), 229–237 (1996)
5. Aouiche, K., Darmont, J., Boussaïd, O., Bentayeb, F.: Automatic selection of bitmap join indexes in data warehouses. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2005. LNCS, vol. 3589, pp. 64–73. Springer, Heidelberg (2005)
6. Bellatreche, L., Missaoui, R., Necir, H., Drias, H.: A data mining approach for selecting bitmap join indices. J. Comput. Sci. Eng. **1**(2), 177–194 (2007)
7. Böhm, C., Berchtold, S., Kriegel, H., Urs, M.: Multidimensional index structures in relational databases. J. Intell. Inf. Syst. **15**(1), 51–70 (2000)
8. Brito, J.J., Siqueira, T.L.L., Times, V.C., Ciferri, R.R., de Ciferri, C.D.: Efficient processing of drill-across queries over geographic data warehouses. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 152–166. Springer, Heidelberg (2011)
9. Bryla, B., Loney, K.: Oracle Database 11g DBA Handbook. McGraw-Hill Osborne Media, New York (2007). ISBN 0071496637
10. Chan, C., Ioannidis, Y.: Bitmap index design and evaluation. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 355–366 (1998)
11. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. **26**(1), 65–74 (1997)
12. Chmiel, J., Morzy, T., Wrembel, R.: Time-HOBI: indexing dimension hierarchies by means of hierarchically organized bitmaps. In: Proceedings of ACM International Workshop on Data Warehousing and OLAP (DOLAP), pp. 69–76 (2010)
13. Davis, K.C., Gupta, A.: Indexing in data warehouses: bitmaps and beyond. In: Wrembel, R., Koncilia, C. (eds.) Data Warehouses and OLAP: Concepts, Architectures and Solutions, pp. 179–202. Idea Group Inc., London (2007). ISBN 1-59904-364-5
14. Deliège, F., Pedersen, T.B.: Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. In: Proceedings of International Conference on Extending Database Technology (EDBT), pp. 228–239. ACM (2010)
15. Finkel, R.A., Bentley, J.L.: Quad trees: a data structure for retrieval on composite keys. Acta Informatica **4**, 1–9 (1974)
16. Fonseca, M.J., Jorge, J.A.: Indexing high-dimensional data for content-based retrieval in large databases. In: Proceedings of International Conference on Database Systems for Advanced Applications (DASFAA) (2003)
17. Furtado, P.: Workload-based placement and join processing in node-partitioned data warehouses. In: Kambayashi, Y., Mohania, M., Wöß, W. (eds.) DaWaK 2004. LNCS, vol. 3181, pp. 38–47. Springer, Heidelberg (2004)
18. Gaede, V., Günther, O.: Multidimensional access methods. ACM Comput. Surv. **30**(2), 170–231 (1998)
19. Graefe, G.: Fast loads and fast queries. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 111–124. Springer, Heidelberg (2009)
20. Gupta, A., Mumick, I.S. (eds.): Materialized Views: Techniques, Implementations, and Applications. MIT Press, Cambridge (1999)
21. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 47–57. ACM (1984)

22. Gyssens, M., Lakshmanan, L.V.S.: A foundation for multi-dimensional databases. In: Proceedings of International Conference on Very Large Data Bases (VLDB), pp. 106–115 (1997)
23. Han, J., Xie, Z., Fu, Y.: Join index hierarchy: an indexing structure for efficient navigation in object-oriented databases. IEEE Trans. Knowl. Data Eng. **11**(2), 321–337 (1999)
24. Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of Data Warehouses. Springer, Heidelberg (2003). ISBN 3-540-42089-4
25. Jensen, C.S., Pedersen, T.B., Tomsen, C.: Multidimensional Databases and Data Warehousing. Morgan & Claypool Publishers, San Rafael (2010). ISBN 978-1-60845-537-9
26. Johnson, T., Sasha, D.: The performance of current B-tree algorithms. ACM Trans. Database Syst. (TODS) **18**(1), 51–101 (1993)
27. Karayannidis, N., Tsois, A., Sellis, T.: Advanced ad hoc star query processing. In: Wrembel, R., Koncilia, C. (eds.) Data Warehouses and OLAP: Concepts, Architectures and Solutions, pp. 136–156. Idea Group Inc., London (2007). ISBN 1-59904-364-5
28. Kemper, A., Moerkotte, G.: Access support in object bases. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 364–374 (1989)
29. Koudas, N.: Space efficient bitmap indexing. In: Proceedings of ACM Conference on Information and Knowledge Management (CIKM), pp. 194–201 (2000)
30. Li, C., Tang, C., Yu, Z., Liu, Y., Zhang, T., Liu, Q., Zhu, M., Jiang, Y.: Mining multi-dimensional frequent patterns without data cube construction. In: Yang, Q., Webb, G. (eds.) PRICAI 2006. LNCS (LNAI), vol. 4099, pp. 251–260. Springer, Heidelberg (2006)
31. Li, K.A., Ross, Z.: Fast joins using join indices. Int. J. Very Larg. Data Bases **8**(1), 1–24 (1999)
32. Malinowski, E., Zimányi, E.: Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications. Springer, Heidelberg (2008). ISBN 9783540744047
33. Moerkotte, G.: Small materialized aggregates: a light weight index structure for data warehousing. In: Proceedings of International Conference on Very Large Data Bases (VLDB), pp. 476–487 (1998)
34. Morzy, M.: Advanced database structure for efficient association rule mining. Ph.D. thesis, Poznan University of Technology, Institute of Computing Science (2004)
35. Morzy, M., Morzy, T., Nanopoulos, A., Manolopoulos, Y.: Hierarchical bitmap index: an efficient and scalable indexing technique for set-valued attributes. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 236–252. Springer, Heidelberg (2003)
36. Morzy, T., Wrembel, R., Chmiel, J., Wojciechowski, A.: Time-HOBI: index for optimizing star queries. Inf. Syst. **37**(5), 412–429 (2012)
37. Nourani, M., Tehranipour, M.H.: RL-Huffman encoding for test compression and power reduction in scan applications. ACM Trans. Des. Autom. Electron. Syst. (TODAES) **10**(1), 91–115 (2005)
38. O'Neil, P.E.: Model 204 architecture and performance. HPTS 1989. LNCS, vol. 359, pp. 39–59. Springer, Heidelberg (1989)
39. O'Neil, P., Graefe, G.: Multi-table joins through bitmapped join indices. SIGMOD Rec. **24**(3), 8–11 (1995)

40. O'Neil, P., O'Neil, E., Chen, X., Revilak, S.: The star schema benchmark and augmented fact table indexing. In: Nambiar, R., Poess, M. (eds.) TPCTC 2009. LNCS, vol. 5895, pp. 237–252. Springer, Heidelberg (2009)
41. O'Neil, P., Quass, D.: Improved query performance with variant indexes. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 38–49 (1997)
42. Padmanabhan, S., Bhattacharjee, B., Malkemus, T., Cranston, L., Huras, M.: Multi-dimensional clustering: a new data layout scheme in DB2. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 637–641 (2003)
43. Papadias, D., Kalnis, P., Zhang, J., Tao, Y.: Efficient OLAP operations in spatial data warehouses. In: Jensen, C.S., Schneider, M., Seeger, B., Tsotras, V.J. (eds.) SSTD 2001. LNCS, vol. 2121, pp. 443–459. Springer, Heidelberg (2001)
44. Papadias, D., Tao, Y., Kalnis, P., Zhang, J.: Indexing spatio-temporal data warehouses. In: Proceedings of International Conference on Data Engineering (ICDE), pp. 166–175. IEEE Computer Society (2002)
45. Rao, J., Zhang, C., Megiddo, N., Lohman, G.: Automating physical database design in a parallel database. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 558–569 (2002)
46. Robinson, J.T.: The K-D-B-tree: a search structure for large multidimensional dynamic indexes. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 10–18. ACM (1981)
47. Rotem, D., Stockinger, K., Wu, K.: Optimizing candidate check costs for bitmap indices. In: Proceedings of ACM Conference on Information and Knowledge Management (CIKM), pp. 648–655 (2005)
48. Scientific Data Management Research Group: FastBit: an efficient compressed bitmap index technology. http://sdm.lbl.gov/fastbit/. Accessed 10 Nov 2006
49. Sinha, R.R., Mitra, S., Winslett, M.: Bitmap indexes for large scientific data sets: a case study. In: Proceedings of Parallel and Distributed Processing Symposium. IEEE (2006)
50. Sinha, R.R., Winslett, M.: Multi-resolution bitmap indexes for scientific data. ACM Trans. Database Syst. (TODS) **32**(3), 1–38 (2007)
51. Siqueira, T.L., Ciferri, C.D.D., Times, V.C., Ciferri, R.R.: The sb-index and the hsb-index: efficient indices for spatial data warehouses. Geoinformatica **16**(1), 165–205 (2012)
52. Stabno, M., Wrembel, R.: RLH: bitmap compression technique based on run-length and Huffman encoding. Inf. Syst. **34**(4–5), 400–414 (2009)
53. Stockinger, K., Wu, K.: Bitmap indices for data warehouses. In: Wrembel, R., Koncilia, C. (eds.) Data Warehouses and OLAP: Concepts, Architectures and Solutions, pp. 157–178. Idea Group Inc., London (2007). ISBN 1-59904-364-5
54. Stockinger, K., Wu, K., Shoshani, A.: Evaluation strategies for bitmap indices with binning. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, pp. 120–129. Springer, Heidelberg (2004)
55. Stöhr, T., Rahm, E.: Warlock: a data allocation tool for parallel warehouses. In: Proceedings of International Conference on Very Large Data Bases (VLDB), pp. 721–722 (2001)
56. Technical Documentation. Oracle Database Data Cartridge Developer's Guide 11g Release 1(11.1)
57. Valduriez, P.: Join indices. ACM Trans. Database Syst. (TODS) **12**(2), 218–246 (1987)

58. Wrembel, R.: Data warehouse performance: selected techniques and data structures. In: Aufaure, M.-A., Zimányi, E. (eds.) eBISS 2011. LNBIP, vol. 96, pp. 27–62. Springer, Heidelberg (2012)
59. Wu, K., Otoo, E.J., Shoshani, A.: An efficient compression scheme for bitmap indices. Research report, Lawrence Berkeley National Laboratory (2004)
60. Wu, K., Otoo, E.J., Shoshani, A.: On the performance of bitmap indices for high cardinality attributes. In: Proceedings of International Conference on Very Large Data Bases (VLDB), pp. 24–35 (2004)
61. Wu, K., Otoo, E.J., Shoshani, A.: Optimizing bitmap indices with efficient compression. ACM Trans. Database Syst. (TODS) **31**(1), 1–38 (2006)
62. Wu, K., Yu, P.: Range-based bitmap indexing for high cardinality attributes with skew. In: International Computer Software and Applications Conference (COMPSAC), pp. 61–67 (1998)
63. Wu, M., Buchmann, A.: Encoded bitmap indexing for data warehouses. In: Proceedings of International Conference on Data Engineering (ICDE), pp. 220–230 (1998)
64. Xu, X., Han, J., Lu, W.: RT-tree: an improved R-tree index structure for spatiotemporal databases. In: International Symposium on Spatial Data Handling, pp. 1040–1049 (1990)
65. Yiu, M.L., Mamoulis, N.: Efficient processing of top-k dominating queries on multidimensional data. In: Proceedings of International Conference on Very Large Data Bases (VLDB), pp. 483–494 (2007)
66. Yu, T.C., Meng, W.: Principles of Database Query Processing for Advanced Applications. Morgan Kaufmann, San Francisco (1998). ISBN 1-55860-434-0
67. Zhuang, Y., Zhuang, Y., Li, Q., Chen, L., Yu, Y.: Indexing high-dimensional data in dual distance spaces: a symmetrical encoding approach. In: Proceedings of International Conference on Extending Database Technology (EDBT), pp. 241–251 (2008)

Intelligent Wizard for Human Language Interaction in Business Intelligence

Morten Middelfart(✉)

TARGIT, Tampa, FL, USA
morton@targit.com

Abstract. This chapter presents a novel extension to TARGIT's patented meta-morphing called "The Intelligent Wizard". Firstly, we compare the currently patent-pending Intelligent Wizard to prior art. Secondly, we present the Intelligent Wizard as implemented in a real-world industrial Business Intelligence (BI) application. Finally, we demonstrate by concrete examples that the Intelligent Wizard allows a user to navigate a real-world data warehouse using only human language and knowledge of business terms, thus significantly simplifying the generation of analytics and reports.

1 Introduction

Organizations that learn from their information pose a challenge to the task of successfully deploying Business Intelligence, since by definition, it's a project that never ends. In fact, the problem grows exponentially throughout the implementation process as user functional requirements multiply as deployment advances from strategic to operational levels, and as the requirements generated multiply through the very learning that the deployment enables. The pace of learning (and the number of questions) is higher at the operational level due in large part to the accelerated decision cycle as compared to the strategic level. Failure to recognize the dynamics of learning and the nature of decision cycles at both the strategic and operational levels will make implementation of Business Intelligence unsuccessful.

From personal experience, implementations fail most often when participation in output creation is limited. If a project starts at the strategic level, it will appear to be successful at the outset, but the (limited) report builders will inevitably feel that report and analysis requests pile up as an increasing number of members of the organization see the value in their work. Unless the group of report-builders is unrealistically close to the number of decision-makers, requests for new information will snowball into a workload that prevents the report-builders from ever fully meeting the expectations the rest of the organization will set. Report-builders will spend eons making changes to ground they've already covered as strategic decision-makers will demand revisions upon revisions. Operational users will feel they don't get the information they need and their world will remain virtually unchanged despite Business Intelligence.

E. Zimnyi (Ed.): eBISS 2013, LNBIP 172, pp. 218–242, 2014.
DOI: 10.1007/978-3-319-05461-2_7,

The way to avoid this "Create/Maintain Equilibrium" is to recognize that implementation is all about users' decision cycles, rather than individual reports and chart. In principle, if we think about anything we do as supporting a better and faster decision (as opposed to responding to a specific report), then an entirely new type of system emerges. For example, in an organization that learns, there cannot be a report without the need to analyze. Whenever something goes wrong or an opportunity presents itself, it will be something new to learn from. Therefore, prefabricated reports will not accommodate changes occurring in the real world, but analytics provides that flexibility. On the other hand, reports and dashboards are excellent tools for creating a frame of reference that aligns the organization's efforts and keep focus on its goals. Recognizing that all these disciplines are tightly interlinked is the key to a successful Business Intelligence implementation that renders an organization informed, powerful, and agile. If users can move freely between predefined goals using reports, dashboards, and analytics that copes with problems and opportunities, any user will be able to travel a decision cycle end-to-end.

Further evidence of the need to re-think traditional delivery of analytics, dashboards, and reports can be found with leading industry analyst, Gartner. Gartner identifies "ease of use" as the dominant Business Intelligence platform buying criterion in 2011, surpassing "functionality" for the first time [6]. This change represents a shift from prioritizing the IT department's need for standardization to prioritizing the ability for casual users to conduct analysis and reporting.

Ease of use in the decision processes that managers and employees work through has been the focal point in the development of TARGIT Decision Suite since its early version in 1995. However, distinguishing from other solutions with the same objective, TARGIT has applied the CALM philosophy [2], which seeks to create synergy between humans and computers as opposed to using computers simply as a tool to create efficiency. An organization following the CALM philosophy is divided into multiple observe-orient-decide-act (OODA) loops, and computing is applied to enable users to cycle these loops as fast as possible (i.e., with as few interactions as possible). TARGIT's patented meta-morphing [3,4] allows users to analyze data by stating their goal, and thus facilitates users cycling OODA loops with few interactions.

Chapter contribution. This chapter presents a novel extension to TARGIT's patented meta-morphing called "The Intelligent Wizard". Firstly, we motivate the need for the Intelligent Wizard. Secondly, we present the Intelligent Wizard as implemented in a real-world industrial Business Intelligence (BI) application. Third, we show in detail how a "free-text" string in human language is converted into specific actions with metadata as parameters. Finally, we demonstrate by concrete examples that the Intelligent Wizard allows a user to navigate a real-world data warehouse using only human language and knowledge of business terms, thus significantly simplifying the generation of analytics and reports. We conduct the presentation using the industry proven software TARGIT Decision Suite.

Chapter content. This chapter is organized as follows. Section 2 motivates the Intelligent Wizard and presents related work. Section 3 presents the implementation of the Intelligent Wizard. Section 4 demonstrates human interaction with the Intelligent Wizard. Finally, Sect. 5 summarizes the chapter.

2 Motivation

In traditional Business Intelligence software, users perform two tasks to create the report desired. First, the user defines a query to examine a subset of data stored within the organization's data warehouse. And second, the user defines a graphical representation of the data that was retrieved. Typically, the presentation means are selected from a report generator or "chart wizard", from which default layout properties (or user-specified layout properties) are identified before making the presentation.

Conventional Business Intelligence software requires users to adhere closely to formal syntax, as the underlying databases require syntactically correct queries to properly retrieve information. Conventional Business Intelligence solutions are frequently configured with user interfaces that guide a user directly to a presentation of retrieved data, often through a wizard. US Patent 7,783,628 [4] discloses such a method and a user interface for making a presentation of data using meta-morphing. The technology allows to enter various business questions via the user interface, e.g.:

1. I would like to see 'cost' grouped by 'time, month'
2. I would like to see 'revenue' grouped by 'time, month', 'customer, group' and 'product, name'
3. I would like to see 'revenue'
4. I would like to see 'country'

Questions are forwarded to a data determination unit, which is configured to identify metadata items by parsing the question. Based on the identified metadata items, the data determination unit is able to look up a storage memory of previously used combinations of metadata, which is used as an entry to retrieve previously used presentation properties that fit the identified metadata items from an earlier use.

This allows some flexibility in the way a user could request a presentation of data. However, the options provided to a user in phrasing a question like the ones above were basically limited to possible combinations of measures, dimensions and criteria (where "I would like to see" was part of a rigid syntax).

US Patent 8,468,444 [5] introduces so-called Hyper Related OLAP which leverages the meta-morphing functionality, and thus allows a user to apply meta-morphing in one interaction (typically mouse click) on any report or dashboard in a Business Intelligence implementation. This functionality adds a lot of analytical freedom to a decision process since it essentially moves the user from the

observation to the orientation phase in one interaction. However, the freedom is not complete since the analytics is limited to the dashboard or reporting context from which the Hyper Relation was initiated.

Today, there is a greater demand for new freedom in phrasing what a user wants to do with requested data. In particular, there is a subordinate but important demand that this greater freedom in phrasing requires as few user interactions as at all possible. In fact, if a user perceives a user interface to be too rigid in its way of accepting input, it may be a reason for discarding a Business Intelligence application.

Conventionally, a user compared different Business Intelligence applications based on what features or functions the application provided. Today, even if all the functions that a user needs are present, the user increasingly considers how smoothly and quickly the application gets to a desired result.

3 The Intelligent Wizard

3.1 Overview

There is an unmet demand for providing more data analysis functions in Business Intelligence applications while, on the other hand, enabling more direct access to a desired end-result for users looking to rapidly create Business Intelligence items such as analyses, dashboards, and reports.

The Intelligent Wizard processes a natural-language user query and translating it into a graphical representation of desired data presented in a format identified as preferred by the user and in harmony with the nature of the data. Broken into software-driven tasks, the Intelligent Wizard:

- "listens for a user's input and process said input to identify input elements thereof (said input elements comprising at least a first and second input element)"
- "searches among predefined named executable functions for a matching executable function that match the first identified input element"
- "searches in a predefined set of named metadata items for a matching metadata item that match the second identified input element"
- "eliminates, from a set of syntax patterns with syntactically valid combinations of named executable functions and types of metadata, patterns that do not comprise the matching executable function and the matching metadata item"
- "wherein a first set of patterns remains, assigns a matching metadata item to one or more of the patterns in the first set"
- "selects a pattern among first set of patterns and execute the function named in the selected predefined pattern with assigned metadata"

As a result, the user is given an option of taking different actions on data at a point in time where the data have not yet been retrieved. The Intelligent Wizard accepts both different executable functions and different metadata as input in

the same user dialogue. The Intelligent Wizard thereby supports different actions that may be taken on data the very first time the data are requested. This saves the user from dealing with prolonged step-by-step dialogues with the application. Thus the user can directly, in a very first user dialogue, obtain the result (s)he is after. For instance, in connection with user interfaces on a mobile device, it is expedient in this way to avoid many steps in a user dialogue.

3.2 Prerequisites

Users have a greater freedom in phrasing their desired end-result. On a first interaction, users can request not only analyses or reports, but early warnings of conditions met in the data, scheduled deliveries of reports, or video sequences of different data views. The application responds by identifying and running respective executable functions, incorporating identified metadata as input. This flexible way of interpreting and intuiting what a user is looking for can save the user valuable time. In this context, we simply use whatever metadata is already stored in (or in conjunction with) a given database; if none is provided we use actual field names from the databases available to the system.

The predefined patterns of named executable functions with their parameters and type of metadata (i.e. their formal syntax), define a paradigm for capturing a user's possibly disarrayed input elements into an ordered structure. This relieves the user from recalling a specific syntax and gives the user a smooth and quick way to get to a desired end-result.

Named executable functions are programmed functions configured to receive a metadata item as its input, retrieve data defined by the metadata item from a database, and output a result of processing the data. The functions are respectively programmed to process the data by preparing an analysis, a report, a warning of some alarming condition being met in the data, a scheduled delivery of a report or a video sequence of different data views. These functions are selectively invoked by the user right from the first point of a user's interaction with the application to request data.

As examples, the functions could be named 'report', 'analysis', 'storyboard', 'notify' and 'schedule', respectively. The functions take one or more parameters as input in the form of a named metadata item, which could be a metadata item in the form of a measure denoted 'Sales' and a dimension denoted 'time' and/or it could be a criterion stating that time is limited to year 2013. The functions retrieve data defined by said named metadata item from a database and outputs a visual presentation of said data.

In general, metadata are data that describes other data. In the context of multidimensional databases, 'measures' are a class of data containing measured values, whereas so-called dimensions are a class of data containing divisions of typically time or some predefined structure. Criteria defining a limit or a range of data or data values are also metadata. Formatting properties for visual presentations are also metadata. Thus, metadata can define which data to retrieve and how to present them. Metadata may thus remain unchanged while the data are updated on a running basis.

The set of predefined patterns of named executable functions and formal syntax can be represented in software in various ways, e.g. by expanding all combinations of executable functions in formal syntax, wherein metadata parameters are represented by parameter names. For instance, a single function named 'analyze' may be expanded to the following predefined patterns:

- "analyze mmm"
- "analyze mmm ddd"
- "analyze mmm ddd ccc"
- "analyze mmm ccc"

Where 'mmm' designates a metadata item of the measures type, 'ddd' designates a metadata item of the dimensions type, and 'ccc' designates a metadata item that is a criterion. 'fff' designates an array of items that can be either 'mmm' or 'ddd'. Once the name and type of the identified metadata items is determined, it can be assigned to a best matching pattern. This pattern may be selected and executed by executing the function named in the selected predefined pattern with assigned metadata.

Listening for a user's input is conducted either through the operating system's innate keyboard software or through speech recognition systems. Sometimes, metadata will be assigned immediately (and follow-up tasks begun) following the termination of user input, which can prove troublesome if a user pauses during input.

3.3 Walkthrough

The step of assigning a matching metadata item to one or more matching executable functions assigns a matching metadata item, of an identified type, firstly mentioned in the input to a firstly selected field requiring the identified type of metadata item. However, other predefined ways of assigning metadata may be feasible.

As a result, users can present input in any order desired and with an informal syntax, and the Intelligent Wizard will reorganize the data to properly fit formal syntax. Superfluous input elements are ignored (chiefly those elements that are not recognized as executable functions or metadata).

The first selected field corresponds to the first type that occurs in the traditional syntax. In other configurations, there is an additional step of searching among the input for matching metadata items, first for named metadata items unique in the complete set, and second for metadata items comprising at least a measure, dimension, and formatting item.

The first step searches for individual metadata items and enables the user to combine metadata items freely. This search is convenient since metadata can be correctly recognized and their types determined before a formal query is formed and submitted to a database.

The second step searches for a collection of metadata items in *documents*. In other words, documents in this context means pre-defined analyses, reports,

dashboards, and other ways of combining and presenting metadata in a business intelligence context. Documents may be stored, for instance, when a user has defined it such. Documents may be defined according to the eXtendend Markup Language (XML). The application is configured to provide the user with graphical user interface tools for defining the metadata stored in a document. In this configuration, a document comprising multiple sets of metadata items must be rendered to provide visualization of multiple sets of data.

In this way, a metadata item input can be as a key to collections of metadata items comprising the metadata item. If for instance a metadata item 'sales' is searched for, a document may be found comprising the metadata item 'sales' in combination with the dimension 'time' and another measure, 'costs', in combination with the dimension 'country'. Firstly, the measure 'sales' is combined with a dimension which makes it possible to complete an association of a measure and a dimension and then in turn to retrieve data for a first graph or other type of presentation. Secondly, a further association is found with the measure 'costs' and the dimension 'country'. Thus, inputting only a single measure reveals a document comprising four measures defined in combination in a document. The metadata item reveals a document defining two presentations along the two dimensions 'time' and 'country'. The document may also comprise formatting metadata and criteria.

The step of executing the function named in the selected predefined pattern with its assigned metadata is performed: in a first manner, wherein metadata are retrieved from the input and from documents, or in a second manner, wherein metadata are determined from the input and where content in documents is disregarded. In the second manner, the Intelligent Wizard presents the user a selectable choice between the first option or the second option.

In this way the user can intervene and decide whether to use a stored document or not, allowing the user to freely define a new combination of metadata. The first option may be implemented by showing a list of stored documents uncovered by a search to the user. The list of documents is set up with links to the documents.

However, the method comprises determining at least one association, of a dimension and a measure, by combination of a measure and dimension identified in the user's input, or by addition of a dimension or measure to an identified measure or dimension, respectively.

The association thus comprises a measure and a dimension which is sufficient for retrieving a set of data. In case the user fails to input both a measure and a dimension, but inputs only one of them, the application performs a search to identify a stored association (stored perhaps in a document which comprises the missing input measure or dimension). The other portion of the association can then be determined by selecting the dimension or measure most frequently used in combination with the respective measure or dimension that was given in the input. It is therefore convenient when a document or an association is stored with a usage counter, and that this counter is updated when this association is used

in connection with an action or when a document containing this association is loaded or stored.

A document may comprise multiple associations. The above step of determining an association may also comprise determining other associations that are stored in a document.

The Intelligent Wizard may also compute a linguistic distance between an input element and an executable function or a named metadata item and compare the linguistic distance to a threshold to decide whether there is a match. The linguistic distance is computed as the "Levenstein distance". Experiments have shown that a threshold of about 60-70 % match allows a good trade-off between matching what the user is after and avoiding catching another executable function or another metadata item. As an example, this threshold makes it possible to match the executable function 'analyze' despite the user's input element 'analysis'. Other algorithms for computing the linguistic distance may be used.

The application comprises a data structure with a group of synonyms for a respective, named metadata item or a named executable function. The processing of said input comprises identifying an input element by looking up the group of synonyms and identifying the input element as the respective, named metadata item or a named executable function.

The group of synonyms comprises a list of synonyms or alternative names for a named metadata item or a named executable function. The Intelligent Wizard receives a user's input, looks up the group of synonyms, and if the user's input is found among the synonyms, takes the appropriate, named metadata item or named executable function as an identified input element. This expands the likelihood of capturing the user's input accurately and provide an appropriate response, which greatly enhances the user experience. The user is more likely to get a desired result, despite being less experienced with databases.

The method needs to compute a linguistic distance between an input element and at least one of the synonyms, compare the linguistic distance to a threshold to decide whether there is a match, and if so, identify the input element as the respective, named metadata item or a named executable function.

The steps of computing a linguistic distance and comparing to a threshold make the application more tolerant to typographical errors and spelling mistakes in the user's input. Combining these steps with a list of synonyms further improves the likelihood of catching the user's input.

The method comprises the step of identifying which types of metadata items in the syntax pattern remain unassigned and providing input elements for an unassigned fields by assigning a default value to the parameter or type of metadata or prompting a user to input a value for an unassigned field.

The application comprises: a user interface configured to receive a user's input in sequential form; named executable functions that are configured to: receive a parameter in the form of a named metadata item, retrieve data defined by said named metadata item from a database, output a result of processing the data, a set of predefined patterns of named executable functions and their formal syntax; and a set of named metadata items.

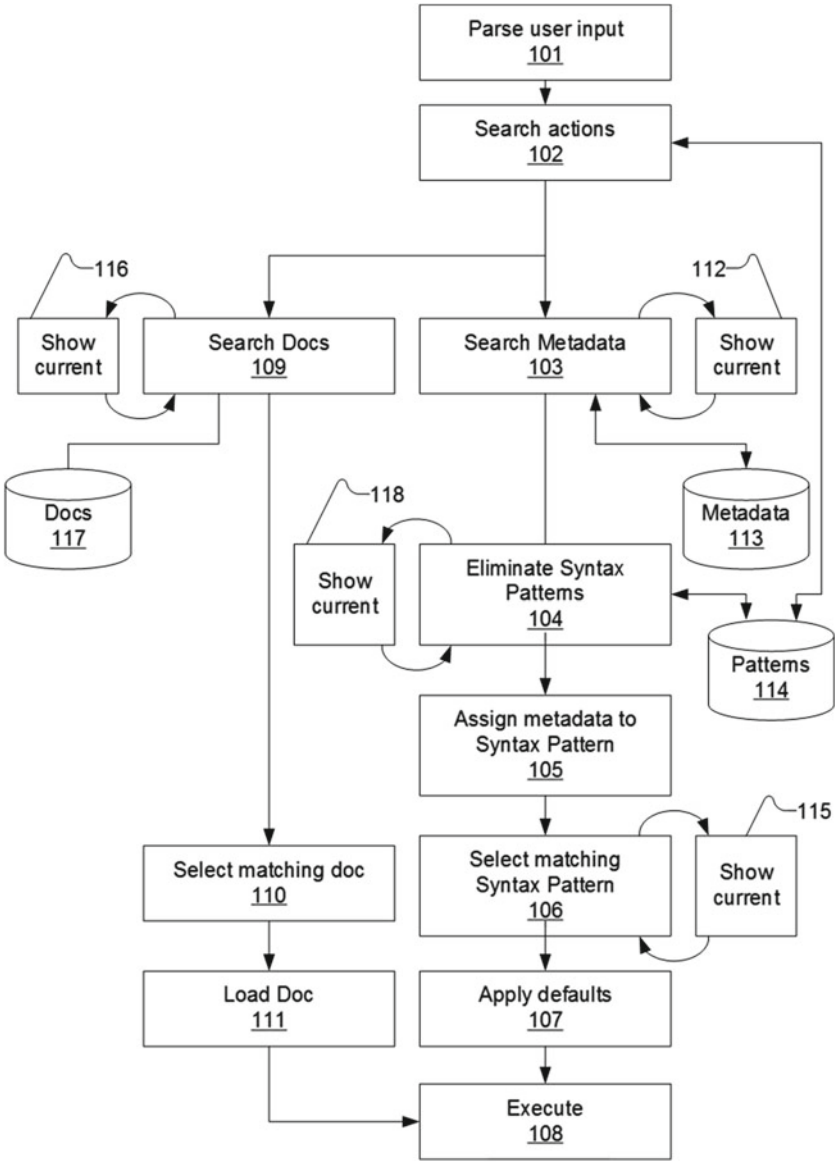


Fig. 1. Flowchart for processing an input via a user interface

Figure 1 shows a flowchart for processing input through a user interface of a Business Intelligence application. The application comprises a user interface configured to receive a user's input in sequential form; a collection of named executable functions and denoted actions, a set of predefined syntax patterns

of named executable functions with respective parameters, and a set of named metadata items.

The user's input can be received through textbox input, in which the user types input via a keyboard or via a speech processor configured to receive a user's spoken input. In general, the user's input is given in form of one or more input elements, which may be letters (characters), words (commands or items), terms, numbers or fractions thereof. Input element is the general term. The phrase 'in sequential form' means that input elements are received one after another.

3.4 Matching Executable Functions (Actions) with Parameters

The Business Intelligence application offers a multitude of executable functions or actions which the user may execute via the user interface by giving his input as described above. The Business Intelligence application has those executable functions embedded in program code or has an interface for accessing the executable functions remotely through a web service. The executable functions are configured to receive a parameter in the form of a named metadata item, retrieve data defined by said named metadata item from a database, and output a result of processing the data. Thereby the user is given an option of taking different actions on data at a point in time where the data are yet not retrieved. This option saves the user from making many tedious user interactions. The executable functions are configured to create:

- an on-screen analysis showing one or more graphs or other data presentations for graphical user interaction,
- a report for a printed media,
- a so-called dashboard,
- a so-called storyboard with a video rendering of data presentations;
- a scheduled delivery of a report, a dashboard, or a storyboard, or
- a warning, triggering any of the above functions in case a predefined criterion is met.

The different executable functions process the data defined by metadata input by the user in respective ways. The different executable functions provide their output at respective destinations, for example, in a data structure of a graphical user interface, in a report, in a video, in a message object (email object), or to a data transport interface for a remote data system. Processing of the data may comprise graphical rendering, statistical signal processing or other types of processing.

In step 101, the method listens for a user's input and processes the input as it is received to identify input elements. This is performed by parsing the user's natural input and translating it into proper syntax. As an example a user may enter the text: "give me a revenue analysis". This input is parsed and output is given in multiple input elements: "give", "me", "a", "revenue", and "analysis". Input elements are separated by a space-character or a temporal pause. A filter may remove input elements such as "give", "me", and "a". Such words may be

removed or ignored if they are comprised by a predefined set of so-called stop-words. Stop-words may also be defined as words that are not included by a list of accepted words that comprises predefined names of metadata, names of actions, and other commands or instructions for the application.

Then in step 102, a search for actions or executable functions is performed. The search is performed in storage memory (114) wherein a set of syntax patterns is stored. An exemplary set of syntax patterns is shown in Table 1 below:

Table 1. Syntax patterns

#	**Pattern**
1	analyze mmm
2	analyze mmm grouped by ddd
3	analyze mmm grouped by ddd selected by ccc
4	analyze mmm selected by ccc
5	report fff
6	report fff grouped by ddd
7	report fff grouped by ddd selected by ccc
8	report fff selected by ccc
9	dashboard showing mmm
10	dashboard mmm ddd
11	dashboard mmm ddd selected by ccc
12	storyboard fff

The syntax patterns comprise syntactically valid combinations of named executable functions and parameters comprising different types of metadata. The executable functions named 'analyze' and 'report' are included in four respective syntax patterns. The functions 'dashboard' and 'storyboard' are included in three and one syntax patterns, respectively. Some actions may accept more parameters than shown above; the action 'analyze' may accept multiple parameters of the measures type. For an implementation of the method, such a complete set of syntax patterns is stored.

Continuing the above example, revenue and analysis remain as input items after removal of stop-words. The search for actions in step 102 then reveals that revenue was not identified as an action, but that analysis is identified as a action. The input element revenue was not identified in step 102, but it is then subject for a search for metadata in step 103. The search is performed in metadata storage memory (113), which comprises a list of metadata items with an indication of their metadata type. Metadata storage memory (113) may comprise the items revenue, cost, and profit of the measures type and the items time, countries of the dimension type. Additionally, the metadata storage memory (113) may comprise metadata items with formatting properties for graphs or other types of presentations. Thereby, the input element revenue can be identified as a measure by the code 'mmm'. By means of step 112, a current assessment of identified metadata items and actions may be shown to the user via the user interface,

e.g., by listing the items analyze and revenue with a checkmark. Thereby the user is notified that the elements are identified or recognized.

In the following step 104, those syntax patterns that do not comprise the matching executable function and the matching metadata item are eliminated from a full set of syntax patterns. Once the action 'analyze' is recognized, patterns 5 through 12 of the above 12 patterns can be eliminated from further search on the input element given by the user. Patterns 1 through 4 remain since they contain the action 'analyze'. The result of the search for metadata is then taken into account, and the identified metadata item revenue is assigned as a parameter for the function analyze in step 105. The step of assigning a metadata item to one or more matching executable functions assigns a matching metadata item (of an identified type) from step 103, firstly mentioned in the input to a firstly selected field requiring the identified type of metadata item.

In connection with step 104, those patterns that are not eliminated may be shown to the user by step 118 (i.e. patterns 1 through 4 above). The user interface shows to the user that the action 'analyze' can be executed since at least one of the patterns (pattern 1 above) is completely assigned with metadata.

The method identifies that a user has stopped entering or giving input and a pattern with all parameters assigned to a value is automatically selected for being executed. Alternatively, in step 107, the method assigns default values to parameters that remained unassigned. Some actions may have parameters assigned to them that are hidden from the user (at least at this stage of user interaction) and that a default is assigned with default parameters. Default parameters may be retrieved in a context-sensitive manner and/or according to a frequency of use. This is described in more detail below in connection with Fig. 4.

Action and metadata items are stored in documents, which are searched in step 109. Step 109 may be entered once the search for actions in step 102 has revealed that at least one action is identified or that a predefined category of an action is identified. Documents are stored in document storage memory (117). Typically, documents comprise collections of metadata items comprising one or more associations of a dimension type metadata item and a measure type metadata item and formatting type metadata e.g. specifying a graph type and properties thereof. Typically, a document contains the information that would have been applied in step 107 in case the user (or the application) selected an action. A document can comprise a collection of graph objects or other types of presentations with its data content specified by means of metadata.

Documents revealed by the search may be listed for the user to select; a selected document is loaded in step 111 and executed in step 108. Execution of actions or documents comprises querying a database to retrieve and render a presentation of the data defined by the metadata items. This is described in more detail below in connection with Fig. 5.

In the event the user inputs either a measure or a dimension, but fails to input both a measure and a dimension, the step of applying defaults may comprise a step of creating an association comprising both a measure and a dimension. This is explained in greater detail in the below in connection with Fig. 4.

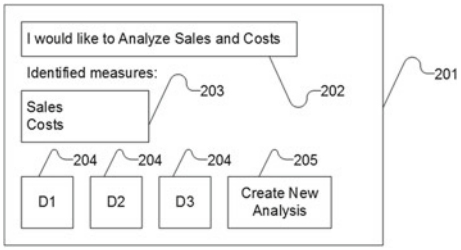


Fig. 2. First portion of user interface

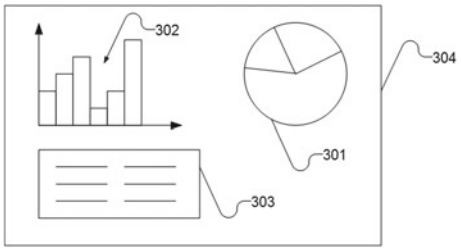


Fig. 3. Second portion of user interface

Figure 2 shows a first portion of a user interface of a Business Intelligence application. The user interface is shown in a simple form with a user interface window 201 comprising a textbox input 202.

The user can give an input in a textual form phrasing what (s)he would like the Business Intelligence application to do for her/him, for example, writing: "give me a revenue analysis" or "I would like to analyze sales and costs". Processing of the input is performed as described above.

The textbox 203 shows metadata items of the measures type or dimensions type as they are identified in the user's input. The metadata items in textbox 203 are shown by means of step 112 as described above.

3.5 Retrieving and Presenting Data

Once an action is identified in the user's input, a button 205 is enabled or shown on the user interface. Pressing the button causes the method to proceed to execute the designated action with the measures listed in the textbox 203 as its parameters. In this way, the user selects a predefined syntax pattern comprising the action and the identified measures.

The result of the search for documents may be shown by graphical buttons 204 designating titles of the documents revealed by the search, perhaps designated "D1", "D2" and "D3". By pressing a button, the user can select a respective document for execution.

Figure 3 shows a second portion of a user interface of a Business Intelligence application. The first portion of the user interface, shown in Fig. 2, is configured

to receive a user's input and to give the user feedback on identified metadata items and documents. The second portion, shown in Fig. 3, is configured to display the result of executing an action outputting a visual presentation output from an action.

In the shown example, the user interface displays three presentation objects in the form of a so-called piechart 301, a bar chart 302 and a table 303. The second portion 304 of the user interface is displayed as a result of executing a document or executing an action in connection with step 108 described above. The presentation objects or the user interface may comprise controls (not shown) configured to adjust or change properties of the presentation objects, for example, to change the type of the presentation object, its color properties, font type for text matter with the object, etc., as it is known in the art. For textual presentations the objects may comprise text sections, headings, tables with columns and rows, etc. The formatting metadata and denoted properties may comprise font size, line spacing, etc. For graphical presentations, the means may comprise different types of charts and diagrams such as bar charts, line charts, pie charts, scatter charts, radar diagrams and other known diagrams or charts based on graphical elements. The properties may comprise tick marker spacing on an axis, legend font size, etc.

The user interface can handle different situations. In a first situation, a user can give his input by means of the first portion of the user interface as described above, whereas in a second situation a user can request further data by an action directed directly to a presentation object to thereby explore data by further analysis.

The processing of metadata in Fig. 4 comprises determining an association and determining a presentation. The method of processing metadata may be performed in connection with step 107 wherein defaults are applied. Defaults are parameter values that the user has not explicitly specified in a present input, but that may be retrieved by some type of expert system as exemplified by the flowchart. Default values may be predefined values comprised of values programmatically entered in the application or values previously selected by a user. The first time a user gives a certain input, values programmatically entered in the

Table 2. Metadata and presentation properties

Data	Presentation	Frequency

Time, Level 1
Revenue
	type=Barchart; legend=off;
	labels=off;
	3D-effects=Orthogonal
		3

Country; Contribution Margin	type=map; legend=off; labels=on; 3D-effects=None	3
Customer, Level 0; Revenue; Cost; Contribution Margin	Type=Crosstab; legend=off; labels=off; 3D-effects=None	2
...	...	...

application are applied. Then (s)he adjusts metadata as necessary and stores the result in a document. A second time the user gives the same certain or a similar input, default values may be retrieved from such a document. Default values are stored in a dedicated storage memory of previously used combinations of metadata and presentation properties and/or other parameters.

The contents of such a storage memory with previously used combinations of metadata and presentation properties can have the following form as shown in Table 2.

By searching the storage memory, with contents e.g. as shown in Table 2 for default values, it is possible to determine whether a previous action has been used. Thereby a user's preferred presentation properties can be found. If, for instance, it is determined that a question involves the data item 'time, level 1' and 'revenue', it can be deduced that the preferred presentation of these data items is a bar chart with properties as shown in Table 2 above.

The flowchart in Fig. 4 is divided into a data determining section 408, a presentation determining section 410 and a step 415 of making a presentation based on determined data and presentation properties. In the event an action does not provide a graphical presentation output, the method may skip the presentation determining section 410, and may terminate when steps of the data determining section 408 have been performed.

In the data determining section 408, metadata 401 identified in step 103 are input. That is, the metadata and their category and associations of data items (if any) are input. Optionally, the metadata 401 can comprise an identification of dimension levels, criteria (if any), and combinations of associations. In step 402 it is determined whether any associations were identified. In the positive event (Y) it can be deduced that sufficient data were present in the question and that a search for presentation properties can be initiated in step 411. In the negative event (N), however, it can be deduced no association is found in the representation of the input metadata the method continues in section 409 of the flowchart to create an association. In section 409, it is examined in step 403 whether an identified data item is a dimension. In the positive event (Y) of step 403, a data item measure is selected in step 405. In the negative event (N) of step 403, it is examined in step 404 whether an identified data item is of the measures type; and in the positive event (Y) thereof a data item dimension is selected in step 406. If no measure is identified, and consequently no dimension is identified, it can be decided that the input metadata has not even incomplete information, but is lacking information. In this case the situation can be handled by prompting the user to enter valid input metadata with at least one data item, or alternatively, by applying the most frequently used metadata and applied presentation properties.

In step 405 and step 406 a previously used association (if any) involving the identified data item may be detected. However, in the event no previously used association involving the data item is found, a most frequently used data item can be selected or all data items of the respective type can be selected. An association based on the identified data items and determined data items

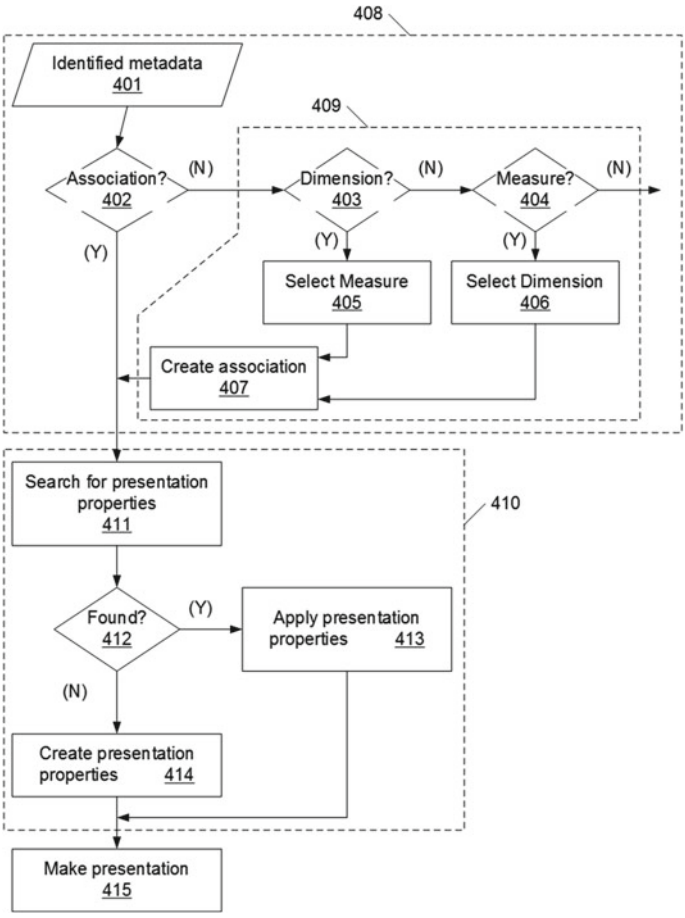


Fig. 4. Flowchart for presentation of metadata

is created in step 407. This step can involve creating a memory object for each data item of the dimensions type.

If a data item measure or dimension has been selected in either step 405 or step 406, respectively, a search for presentation properties can be initiated in step 411. In step 412, the system examines whether a previously used presentation is found. If so (Y), presentation properties of a previously stored, like association is found, the presentation properties are applied in step 413 to make a presentation of the data specified by the association in step 415. Alternatively (N), if no like association is found, presentation properties are created either by selecting the association that is most like an existing association and/or by using an expert system. As a result, even a question with incomplete data identification can lead to retrieval and application of preferred presentation properties to make a presentation.

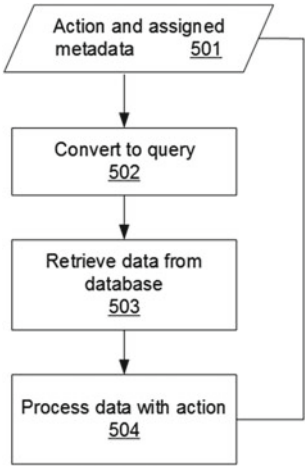


Fig. 5. Flowchart for retrieving data based on metadata

The method can simply parse the input metadata to identify associations and proceed directly to step 411 to search for presentation properties. Presentation properties can be retrieved based on associations identified in the input metadata. Thus the steps 402 and steps in section 409 can be omitted.

A dimension or measure as the case may be can be selected to create a complete association. Also, an identified association, dimension or measure can be expanded to multiple associations.

The expanding can be based on user preferences. User preferences can be determined by maintaining a list of combinations of associations that are used concurrently or in the same data report. Thus, when an association is determined it can be expanded by looking up other associations which have previously been used in combination with the identified association.

Alternatively, the list can include a number which for an identified association reflects the likelihood that a user will apply another identified association. This number can be a relative or absolute number of times the identified association has previously been used in combination with the other identified association. Thus, when an association is determined, it can be expanded by looking up other associations which, with a given likelihood, have been used in combination with the identified association. For instance, the given likelihood can be expressed as a threshold value of, say, 50 %; for an association A1 it is stated that in 10 % of its uses association A2 is also used, whereas in 90 % of its uses, association A3 is also used. Thus it can be deduced that A1 should be expanded to be combined with A3.

When a measure is identified, firstly, a complete association is created by selecting a dimension. Secondly, this created association can be expanded as set forth above. This also applies to dimensions.

The aspect of expanding an association from an identified association and/or measure and/or dimension can be embodied as an individual step between

section 408 and 410. Alternatively, the aspect can be embodied by means of section 410. In any event, when an association is expanded, presentation properties should be applied to make a presentation of the multiple associations obtained by expanding an identified association.

It should be noted that step 402 can be applied iteratively to identify multiple associations and/or measures and/or dimensions in the metadata.

Figure 5 shows a flowchart for a method of retrieving data from metadata. As mentioned above, the executable functions or actions are configured to receive a parameter in the form of a named metadata item; retrieve data defined by said named metadata item from a database; and output a result of processing the data.

As a result of assigning metadata in step 105 and applying defaults in step 107, or as a result of loading a document in step 111, the action(s) can be executed in step 108. As a step thereof, data are retrieved from a database using the metadata. Sometimes data are also denoted underlying data to more clearly distinguish the data that are retrieved from metadata.

As a step of executing the action in step 108, an action and assigned metadata 501, are converted to a query to retrieve data from a database in step 502. In step 503, the data are retrieved from the database, and in step 504, the data are processed according to program steps defined in a selected action.

For example, if the database contains the following data items, wherein the date items are categorized as measures or dimensions and wherein a dimension exists at different levels such as day, month, and year.

In Table 3, the measures 'revenue' and 'cost' contain data along the different dimensions. A data set is defined by specifying both a measure and a dimension. For instance, the association of the measure 'revenue' and 'country' defines a data set where 'revenue' is represented by means of data values per 'country' as they may be defined in the database.

Generally, and in connection with the present invention, it should be noted that a dimension is a collection of data of the same type: it allows one to structure a multidimensional database. A multidimensional database is typically defined as a database with at least three independent dimensions. Measures are data structured by dimensions. In a measure, each cell of data is associated with one single position in a dimension.

Table 3. Syntax patterns

Measures	**Dimensions**
'revenue'	'time' (level 0: Year; level 1: Month; level 2: Day)
'cost'	'Customer' (level 0: Group; level 1: Name)
	'Product' (level 0: group; level 1: Name)
	'Country'

4 Demonstration

Having defined the workings of the Intelligent Wizard in theory in previous section, this demonstration shows it in a real-world scenario running on a Microsoft SQL/Analysis Server 2012 data warehouse. The fields and relationships in the data warehouse staging area are shown in Fig. 6, and these load into 6 cubes with a total of 37 dimensions and 52 measures.

Analytics: In Fig. 7 we see a screenshot of TARGIT Decision Suite version 2013, with the Intelligent Wizard shown on the entire screen below the top menu. The interaction with the Intelligent Wizard is arranged around the OODA loop, as described in Sect. 1, and thus the options will emerge as we type a sentence in the field labeled "Decision Search". However, we note that when the search string is blank, the various Business Intelligence disciplines supported by TARGIT

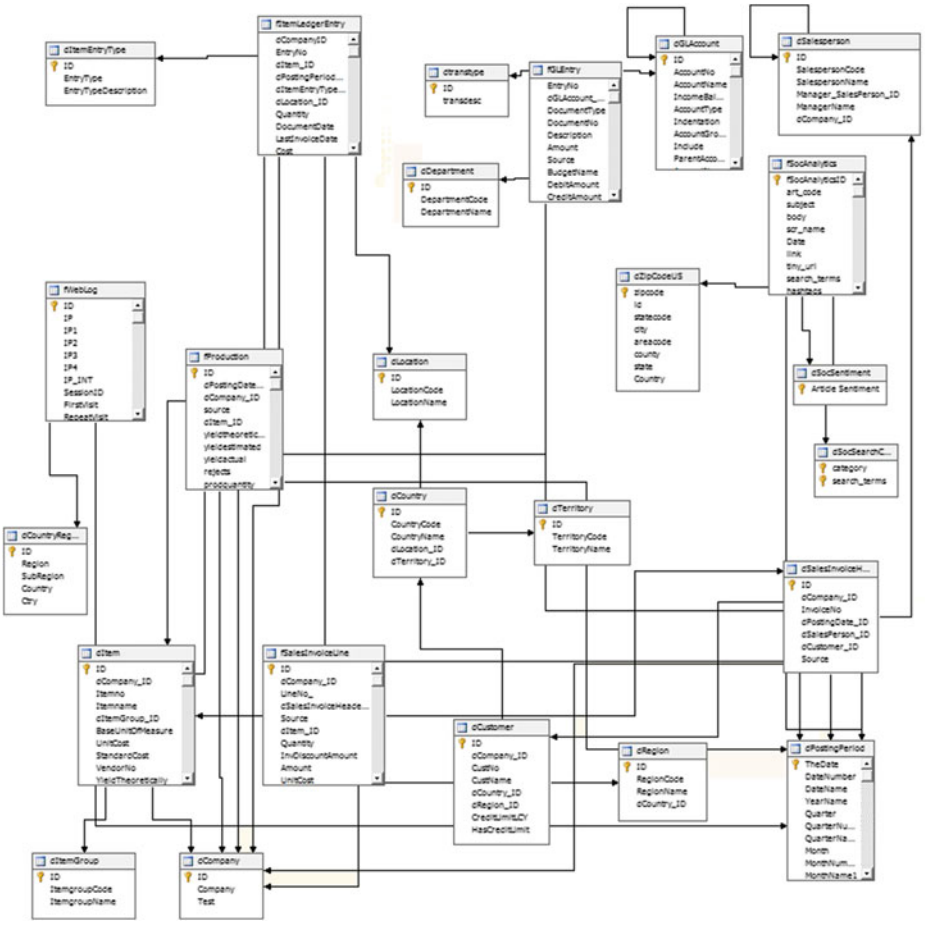


Fig. 6. Datamodel used in demonstration

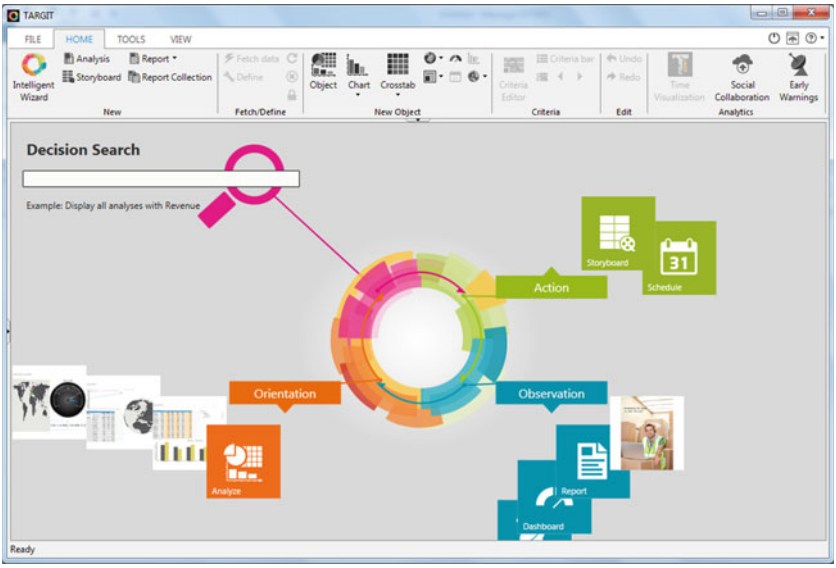


Fig. 7. TARGIT decision suite with intelligent wizard

Decision Suite are displayed in accordance with their role in the OODA loop. We also note that if some existing objects (such as analysis, reports, and dashboards) exist, these will be shown as snapshots in accordance with their role in the OODA loop.

In Fig. 8 we have entered the sentence "show me an analysis with revenue managers and items" in the Decision Search. We note that now only the options under the "Orientation" phase in the OODA loop are shown. This is due the Intelligent Wizard identifying the word "analysis". We also note that a list of metadata appears under the search field. This list allows the user to manually select metadata of particular interest to focus the Intelligent Wizard on these metadata. In this demonstration we do not make use of this option.

In Fig. 9 we see the resulting screen from clicking the "Analyze" icon in Fig. 8. Observe that the metadata have been organized in two objects, in which the measure Revenue is displayed over Sales Managers as well as over Items (products). The Intelligent Wizard automatically selected the most appropriate visualization in both objects, including selection between chart and type hereof, maps (not limited to geographical), and tables with sorting. Selection of visualization is done to as large extent as possible in accordance with "best practices in presentation of data", as described by Stephen Few [1].

Note that getting to the presentation in Fig. 9 took only the writing of the sentence "show me an analysis with revenue managers and items" and one click (or touch depending on device). The intent is that Fig. 9 represents a fast and pretty good "initial visualization", from which the user can now start analyzing further, either by adding more objects for visualization, or by interacting

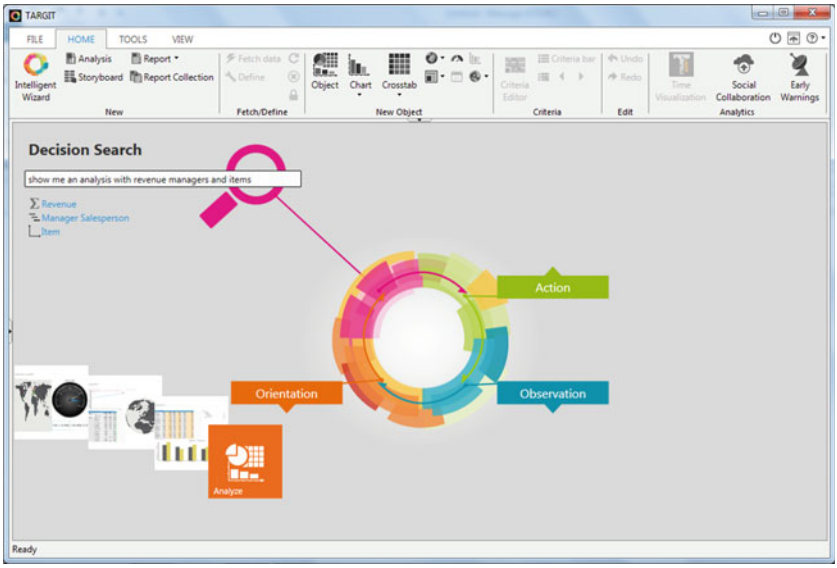


Fig. 8. Analytics selection

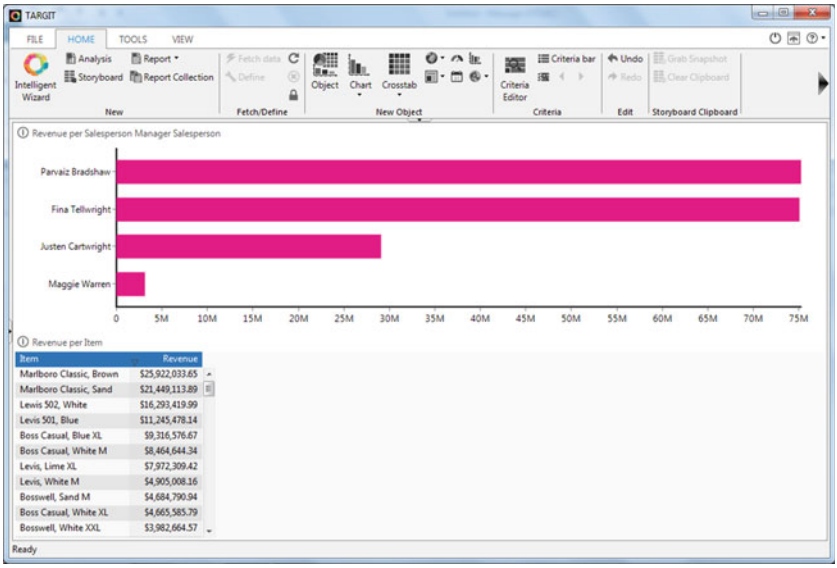


Fig. 9. Analytics visualization output

with the ones that are already present. In Fig. 10 we click in the "Paviaz Bradshaw" member of the Salesperson Manager dimension, and by doing so, we see the object below reflect only the Items (Products) generating revenue for this particular manager. We have now effectively enabled a user with no knowledge

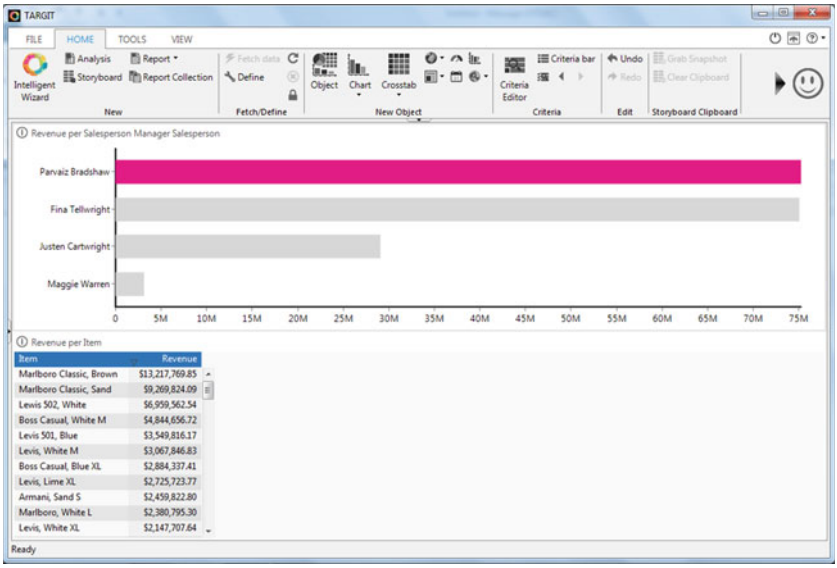


Fig. 10. Analytics interaction

about the complicated data structures in Fig. 6 to interact with the data to create business insights, and to do so using only natural human language and business terms used widely in the organization/company.

Reporting: Returning to the initial screen shown in Fig. 7, we now type "make a report showing customers revenue and profit" as shown in Fig. 11. The Intelligent Wizard recognizes the word "report". However, in this case there are no predefined reports that match the criteria, so only the option of creating a new report is available in this example. Similar to the previous example, we note the metadata available are displayed and further selection is possible.

We do not make an additional selection, but simply click the (create a) "Report" icon under the "Observation" phase, and the report shown in Fig. 12 appears. In other words, using human language we have requested a report and created it from scratch in just a sentence and one click (or touch).

Demonstration Summary: In this demonstration, we have shown how the Intelligent Wizard allows a user self-service within the two Business Intelligence disciplines: analytics and reporting. The Intelligent Wizard successfully supported users in creating both an interactive analysis and a report using human language with no predefined syntax. Furthermore, the Intelligent Wizard needed only one interaction in addition to the initial input to create the intended output.

Download: The fully functional software used in this demonstration can be downloaded at www.targit.com

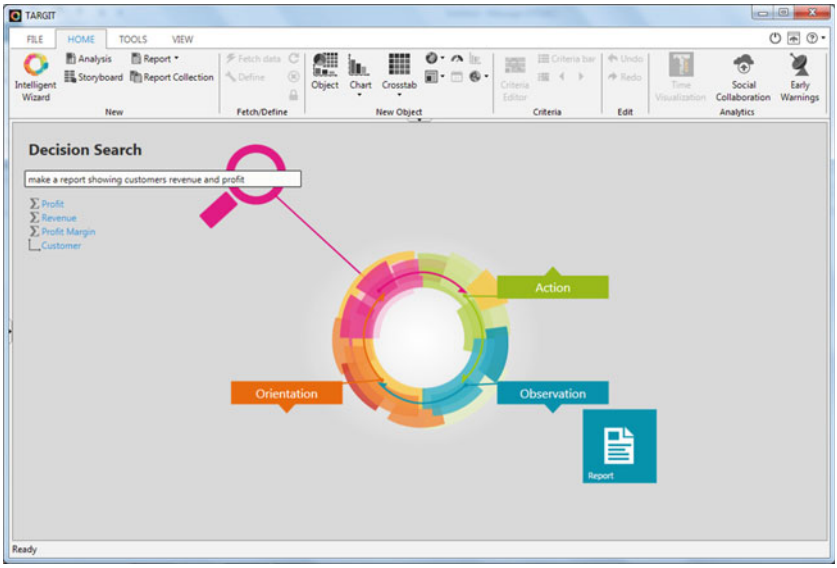
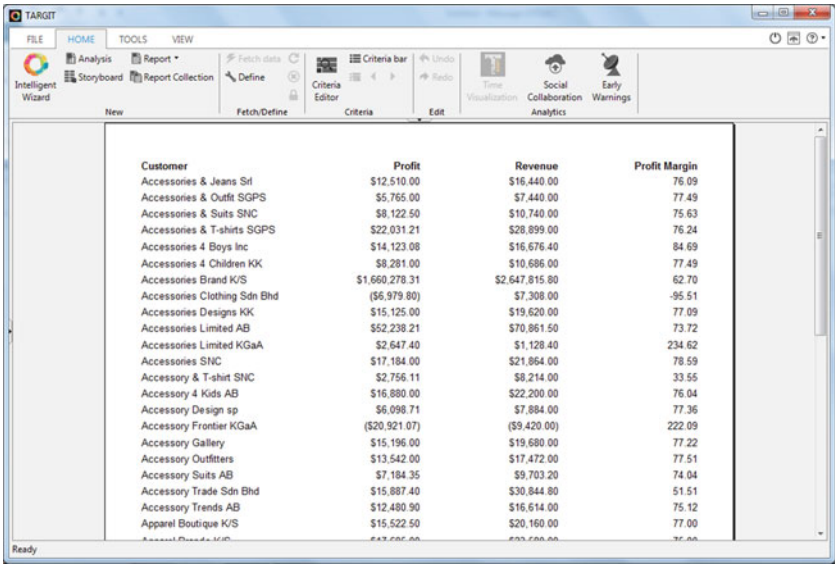


Fig. 11. Reporting selection



Customer	Profit	Revenue	Profit Margin
Accessories & Jeans Srl	$12,510.00	$16,440.00	76.09
Accessories & Outfit SGPS	$5,765.00	$7,440.00	77.49
Accessories & Suits SNC	$8,122.50	$10,740.00	75.63
Accessories & T-shirts SGPS	$22,031.21	$28,899.00	76.24
Accessories 4 Boys Inc	$14,123.08	$16,676.40	84.69
Accessories 4 Children KK	$8,281.00	$10,686.00	77.49
Accessories Brand K/S	$1,660,278.31	$2,647,815.80	62.70
Accessories Clothing Sdn Bhd	($6,979.80)	$7,308.00	-95.51
Accessories Designs KK	$15,125.00	$19,620.00	77.09
Accessories Limited AB	$52,238.21	$70,861.50	73.72
Accessories Limited KGaA	$2,647.40	$1,128.40	234.62
Accessories SNC	$17,184.00	$21,864.00	78.59
Accessory & T-shirt SNC	$2,756.11	$8,214.00	33.55
Accessory 4 Kids AB	$16,880.00	$22,200.00	76.04
Accessory Design sp	$6,098.71	$7,884.00	77.36
Accessory Frontier KGaA	($20,921.07)	($9,420.00)	222.09
Accessory Gallery	$15,196.00	$19,680.00	77.22
Accessory Outfitters	$13,542.00	$17,472.00	77.51
Accessory Suits AB	$7,184.35	$9,703.20	74.04
Accessory Trade Sdn Bhd	$15,887.40	$30,844.80	51.51
Accessory Trends AB	$12,480.90	$16,614.00	75.12
Apparel Boutique K/S	$15,522.50	$20,160.00	77.00

Fig. 12. Reporting output

5 Summary

In this chapter, we presented a novel Intelligent Wizard that allowed users to interact with a Business Intelligence system via natural human language and very few interactions. We demonstrated how the Intelligent Wizard is different from prior technologies in that it:

1. Covers all contemporary Business Intelligence disciplines: analytics, reporting, dashboards, and agents, and novel disciplines like storyboards.
2. Allows users to interact with a Business Intelligence system via natural language.

We also demonstrated with concrete examples that a user is fully self-service capable in a real-world Business Intelligence application, as the Intelligent Wizard allows the user to navigate with natural language, including organizationally-known business terms. Furthermore, we demonstrated an implementation of the Intelligent Wizard in a real-world industrial Business Intelligence application with improved user-friendliness as the effect.

TARGIT Decision Suite 2013, which is the first product to have the Intelligent Wizard, was launched on July 4th 2013. Therefore the feedback from the market is still limited at the time of writing. However, one very skilled TARGIT partner with deep industry knowledge about both TARGIT and competing platforms have stated [7]:

TARGIT has always been known for its ease of use. In Decision Suite 2013 I can just ask TARGIT to find business answers for me by typing in "laymans terms". You dont have to be a technology wiz to find the answers you need. The new Intelligent Wizard works just as I expect it. I type Analyze salesperson profit over last five years and Im presented with a beautiful analysis of my sales. This will definitely help more people and business users find valuable insights in their data.

Given the feedback from the market and the demonstration in this chapter, we will conclude that the Intelligent Wizard is indeed a useful step towards allowing everyone self-service in Business Intelligent and Analytics. For future work, we intend to further develop the interface to include more functionality as well as capturing more parameters from a freely formulated user sentence. In addition, we intend to combine the current interface with speech recognition, thus supporting a use case where one can simply speak to, e.g., a mobile device and ask for Business Intelligence. The speech use case can already be seen in this video demonstration recorded in September 2013: `youtu.be/32KE0rbGZ9c`

Acknowledgments. This work was supported by TARGIT US inc.

References

1. Few, S.: Show Me the Numbers: Designing Tables and Graphs to Enlighten, 2nd edn. Analytics Press (2012)
2. Middelfart, M.: CALM: Computer Aided Leadership & Management. iUniverse (2005)
3. Middelfart, M.: Presentation of data using meta-morphing. United States Patent 7,779,018. Issued August 17, 2010
4. Middelfart, M.: Method and user interface for making a presentation of data using meta-morphing. United States Patent 7,783,628. Issued August 24, 2010
5. Middelfart, M.: Hyper related OLAP. United States Patent 8,468,444. Issued June 18, 2013
6. Sallam, R.L., Richardson, J., Hagerty, J., Hostmann, B.: Magic Quadrant for Business Intelligence Platforms. www.gartner.com/technology/media-products/reprints/oracle/article180/article180.html April 28, 2011
7. The TARGIT Blog. Entry by Jens-Jacob Thuun Aarup, Sales & Marketing Director, Inspari. http://www.targit.com/en/resources/targitcommunity/targit-blog July 31, 2013

Author Index