Detecting similarity patterns in OLAP business databases

With a case study on super market sales data

Master Thesis

Arjen Huberts

Studentnumber: 279704ah
Email: a_huberts@hotmail.com
supervisors: E. Caron & R. Potharst

Economic & Informatics
Erasmus University Rotterdam
The Netherlands.

Abstract

In this paper we describe and implement the detection of similarity patterns in an OLAP cube. This functionality (semi-)automates the generic explanation formalism for a cell in the cube. The explanation formalisms are generated using a greedy algorithm or a top-down algorithm. The explanation formalism is represented as an explanationt tree. The explanation trees across a dimension level can be compared with each other to detect a similarity tree. This similarity can be a structural similarity tree or a weighted similarity tree that uses fractions and standard deviation. Similarity patterns makes managerial decision making based on structural causes possible. At the moment the parent cell is used to find structural similarity. But large incidental causes can than be mistaken as structural. The functionalities were implemented and tested on an artificial, but realistic, case study involving the sales data of a food market database.

Contents

Chapter 1

Introduction

1.1 Background

In today's economy it is getting more and more important to have quick access to the correct information at the right moment for managerial decision making. However the increasing volume of data, and the data being stored at a variety of systems, create difficulties for efficient and effective managerial decision-making. This notion has led to companies implementing Enterprise Information Systems (EIS). EIS is a collection of systems that collect and process data for managerial purposes. The data warehousing part of EIS are the Corporate Information Factory (CIF). The framework of CIF can be seen in Figure 1.1 [Inm96] and consists out of four successive phases

- Production,
- Assembly, Logistics and Storage,
- Processing, Analysing and Consumption,
- Decision making.

The production phase is a collection of systems that collect, store and maintain production data, for example Enterprise Resource Planning Systems (ERP Systems) and On-Line Transactional Processing Systems (OLTP Systems) [BE08]. These different systems work independently of each other and collect, store and maintain the data differently.

The Assembly, Logistics and Storage phase consists of two sub phases. In the ETL Staging Area the data of the different systems are processed to an universal format. The loading format has pre-defined rules and functions [KH05]. In the data warehouse collects the processed data from the ETL staging Area and stores the data. Once the data is stored the data cannot be modified.

In the third phase a collection of front-end applications for data analysis that can be used by managers and business analysts for managerial decision making. The applications can divided into statistical, data mining, querying & reporting and multi-dimensional database applications. The statistics and data mining applications try to detect abnormalities. The querying & reporting applications allow the manager to select a view over the data and generate a report. In multi-dimensional database applications multiple queries are executed after each other. It allows the manager to get multi-dimensional views on the data with so called data cubes. This allows the manager to aggregate and inspect the data visually with a number of managerial reports.

In the last phase of the framework the actual business decision making takes place by managers and business analysts that use the results generated by the front-end applications.

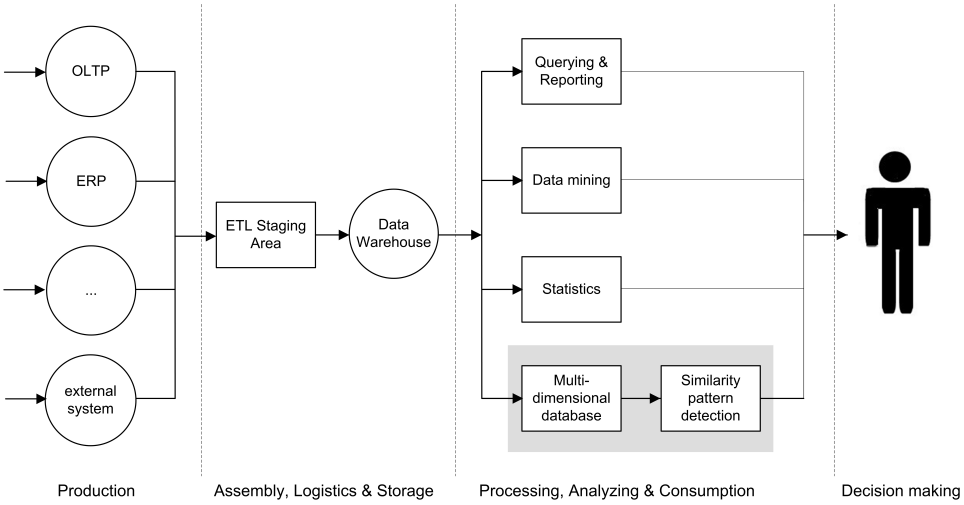


Figure 1.1: Business Intelligence framework based on [Inm96]

In the literature a number of techniques are developed to improve the analysis of multi-dimensional databases. In this thesis we add to this body of knowledge by proposing a technique for the detection of similarity patterns.

Suppose the analyst wants to find a similarity pattern in the monthly profit. In a typical multi dimensional database the analyst would have to take 5 to 7 dimension with on average three dimension levels into account to explain every month's profit [CD08]. The analyst would first have to find an explanation for every month's profit and then has to compare these explanations with each other to find a similarity pattern. At the moment this is a manual task that is time consuming and prone to errors, such as overlooking interesting patterns. It's especially time consuming and prone to errors with large and complex databases. Therefore we semi automate the

detection of similarity patterns in a multi dimensional database, allowing the analyst to consider the complete database and find the patterns of interest for the analyst.

1.2 Research question

The main research question in this thesis is:

How can the analysis functionality of multi-dimensional databases be extended with the detection of similarity patterns?

In this thesis we first have to answer questions on what we mean with terms such as multi-dimensional databases and similarity patterns. We therefore break down the research questions into the following sub-questions:

1. *What are multi-dimensional databases?*
2. *What is (greedy) explanation in the OLAP context?*
3. *How can the similarity operator be defined?*
4. *How can the similarity operator be implemented?*
5. *How can this operator be applied in a case study?*

The answer to the sub questions are answered in separate chapters. The first two subquestions are used to get a good understanding of the existing theory. They are required to create a good understanding on how we attempt to define similarity patterns in the context of multi-dimensional databases. With the third and fourth questions a prototype is developed and built. The last question tests the prototype and implements the prototype into a case study.

1.3 Methodology

In the chapters of this thesis we apply various methods to answer the research questions.

Chapters 2 and 3 are mainly a literature review. The literature review is needed to be able to write a model for detecting similarity patterns in an OLAP business database. In chapter 3 the multi-dimensional database is explained using Caron's notations [Car09].In chapter 3 the explanation formalism is created for a cell in the database. In chapter we continue with explaining how the most important explanations are selected.

In chapter 4 we develop a model for similarity detection. We discuss what the model's input is, the working of the model and output.

In the next chapter the prototype software is described. The prototype is built in Visual Basics and uses Ms Excel 2003 and Ms Access 2003 to show the results. The conceptual design is further developed and explained. Finally the prototype is tested a database with super market sales data. We use the different heuristics to find an explanation tree for an arbitrary cell and a similarity tree for a given set of cells.

1.4 Relevance

The relevance on this thesis can be divided into the scientific relevance and the economic relevance.

At the moment similarity patterns have to be found manually in multi-dimensional databases. Because of time and cognitive limitations the analyst is clearly restricted in finding similarity patterns. Semi-automate detection of similarity patterns allows a more detailed similarity patterns to be found. The semi-automation also prevents errors to be made. This makes it possible to improve managerial decision making based on the information received. The application is an advantage for companies because patterns in databases such as a sales database can be detected quickly and correctly. Because correct patterns are detected quickly they can react quickly to adjust their policy if needed.

Detecting similarity patterns as a topic is not new in science. Many algorithms and techniques are used to detect similarities for various applications. For example, web graph similarity [PAH08] where similarity is used to check how well a search engine performed in finding the requested search. But the detection of similarity patterns in an OLAP database is a new application for where there are no publications to our knowledge.

1.5 Thesis Outline

The thesis is organized as follows. In Chapter 2 a general introduction is given to multi dimensional databases and give background information. In Chapter 3 we describe the explanation formalism. This explanation formalism, for exceptional values, is based on the model of [CD08]. We extend this formalism using the greedy algorithm. In Chapter 4 we explain graph similarity and explain how this can be used in an OLAP datacube. In Chapter 5 we briefly describe how we built the prototype with case and UML diagrams. We than use a food market database and implement the prototype. We describe how this is done and explain why certain choices were made with a case study. Finally we summarize our findings in Chapter 6, and present our final conclusions and possible directions for further research.

Chapter 2

Multidimensional databases

The purpose of this chapter is to explain the basic concepts behind multi-dimensional or OLAP databases. Because OLAP, *On-Line Analytical Processing*, is used as analysis tool we review the most important OLAP terminology and equations. The definitions introduced in this chapter are used in chapter 3 and 4 to define the explanation formalism (chapter 3) and similarity patterns (chapter 4) in an OLAP database.

2.1 Introduction

With traditional databases it takes a long time before a business report is generated. These reports are inflexible and when another view is required it may take a while before this other view is created. With a multi-dimensional model the data can be accessed quickly and from different perspectives, because the data is structured in categories. Therefore a multidimensional database has become a popular tool for data warehouse applications and business analysis tools such as OLAP. OLAP is different than other business tools such as OLTP, because OLTP systems are used to create snapshots of ongoing business processes, whereas OLAP is used for decision support based on historical data.

The first OLAP business analysis tool was Express in 1970 [Pen02], however, the term OLAP was introduced in 1993 for the first time by Codd [CCC93]. In the late 1990's OLAP became more and more popular, and in 1998 OLAP became widely accepted with the introduction of Microsoft Analysis Services. Today Microsoft is one of the largest OLAP vendors, with IBM, Oracle and SAP as their main competitors. Their products analyse why an event occurred and can detect trends, but none of the products detect similarity patterns in the data.

OLAP can be categorized into three storage systems; MOLAP, ROLAP

and HOLAP. MOLAP, Multidimensional OnLine Analytical Processing, is designed to analyse data using a multidimensional database. All calculations are performed when the cube is created, this limits the amount of data. But it also allows the data to be extracted quickly and complex calculations are possible and extracted quickly. Products such as Microsoft Analysis Service and Cognos Powerplay use MOLAP.

ROLAP systems, Relational OnLine Analytical Processing, manipulate the data stored in the database simulating the more traditional MOLAP system. Because it manipulates the stored data, large amount of data can be processed, but it takes more time to extract the data. With ROLAP models it is possible to created new database tables or remove existing database tables. Products such as Oracle BI and Microsoft Analysis Service use ROLAP.

The last system is HOLAP, Hybrid OnLine Analytical Processing, and is a combination of the two systems mentioned above. The system stores new data in MOLAP ensuring quick data extraction and older data is stored in the ROLAP system. Most of the products use this system.

In this thesis we use MOLAP system because we want to analyse data in the cube and extract data quickly from the database.

OLAP uses a partially denormalized database, allowing the database to be viewed from different angles. It typically stores data in a fact table with multiple dimension tables associated with it. This formal structure is known as a schema. The fact table has a column for each dimension and for each measure. The dimension column contains a foreign key relating to the relevant dimension table's primary key. Dimensions and measures are explained further on this chapter. In the following chapters we use a simplified dataset of the food market database with dimensions *Item*, *Time* and *Location* as running example to illustrate the terminology.

The most popular schema is the star schema. Here each dimension table is directly linked to the fact table. An example of the star schema representing the running example is shown in Figure 2.1. An extension of the star schema is the snowflake schema. It splits the dimension table further into additional tables in order to reduce the redundancy level. Some schemas have a multiple fact tables, but these schemas can be seen as a collection of star schemas. That's why these schemas are sometimes called galaxy schemas.

Sometimes OLAP databases are compared with statistical databases. The statistical database are similar to an OLAP database. However the statistical database are more close to a relational database instead of multidimensional database. The statistical database also use statistical analysis techniques that go beyond SQL statements. The last major difference is that statistical databases are mainly socio-economic oriented and a multi-

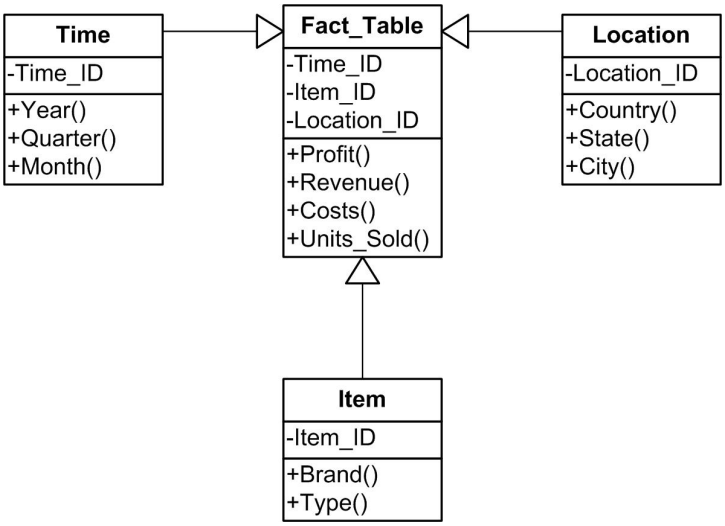


Figure 2.1: The star schema of the supermarket sales data example

dimensional database are business oriented.

2.2 OLAP Terminology

In this section we discuss the OLAP terminology needed in the rest of the thesis. Here we give a summary of the most important concepts on the work of Caron and Daniels article [CD08].

2.2.1 Dimensions & Dimension hierarchies

A multidimensional database is modelled in such a way the data can be viewed from different perspectives allowing the end users to answer business questions. The different perspectives are known as dimensions. The dimensions are represented as dimension tables in the schema. Examples of dimensions are Time, Item and Location.

The dimensions have dimension levels allowing the end users to have a detailed view or a more general view of the data. The dimension levels are organized in an hierarchy with the most detailed level at the lowest level. In general

$$D_k^0 \prec D_k^1 \prec ... \prec D_k^{max_k}. \tag{2.1}$$

Every dimension level $D_k^{i_k}$ has an unique categoric label $A_k^{i_k}$ that corresponds with a column name from the dimension table. In Figure 2.2 an example of the Location's dimension hierarchy is shown.

The values of the dimension levels are called instances or members. They are denoted by $d_k^{i_k}$ where $d_k^{i_k} \in D_k^{i_k}$. For example, the instances of the di-

mension level countries are *USA* and *Mexico*. The total number of instances in $D_k^{i_k}$ is denoted by $|D_k^{i_k}|$.

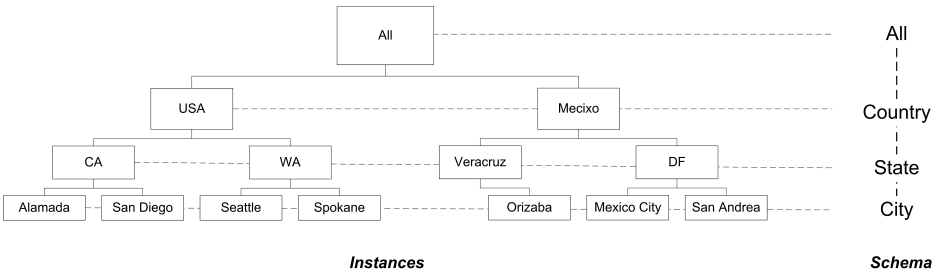


Figure 2.2: Example of the Location's dimension hierarchy

The dimension level structure is based on a parent-child relationship. This is a 1:n relationship where the child can have only one parent, but the parent can have multiple children. With a single roll-up operator the parent can be found [Car09]

$$r^{+1}(D_k^{i_k}) = D_k^{i_k+1}. \tag{2.2}$$

Reversely with a drill-down operator the child can be found [Car09]

$$r^{-1}(D_k^{i_k}) = D_k^{i_k-1}. \tag{2.3}$$

The operators can also be used on a subset $X_k^{i_k} \in D_k^{i_k}$ or on instance level $d_k^{i_k}$. For example the roll-up operator on the dimension level as $r^{+1}(D_L^{City}) = D_L^{State}$, or the drill-down operator on the instance level as $r^{-1}(D_L^{Veracruz}) = D_L^{Orizaba}$.

Moreover, the dimension hierarchy structure can also be in a dot notation showing the levels of the dimension e.g. the dimension *Location*'s structure is *Country.State.City*. The notation is defined as

$$D_k^{i_k} = A_k^{i_{max_k}}.A_k^{i_{max_k}-1}.....A_k^{i_{0_k}}. \tag{2.4}$$

For the dimension level instance hierarchy the dot notation is similar, e.g. USA.CA. It is defined as

$$d_k^{i_k} = a_k^{i_{max_k}}.a_k^{i_{max_k}-1}.....a_k^{i_{0_k}}. \tag{2.5}$$

2.2.2 Cubes

The data in a multidimensional database is structured as a cube. It allows the data to be viewed from the different dimensions. Although the name suggests a three dimensional view a typical cube has five to seven dimensions. The cube C is the Cartesian product of all the dimension levels in the cube [Car09]

$$C = X_1^{i_1} \times X_2^{i_1} \times ... \times X_n^{i_n},\ where\ X_k^{i_k} \subseteq D_k^{i_k}. \tag{2.6}$$

When $X_n^{i_n} = D_n^{i_n}$ the full cube is found

$$C_F = D_1^{i_1} \times D_2^{i_1} \times ... \times D_n^{i_n}. \tag{2.7}$$

The cube can also be represented as $(X_1^{i_1}, X_2^{i_1}..., X_n^{i_n})$ or $[i_1, i_2, ..., i_n]$. In the running example the cube can be shown as $Location^3 \times Product^2 \times Time^3$.

The combination of the different dimensions' instances results in the cube's cell c

$$c = d_1^{i_1} \times d_2^{i_1} \times ... \times d_n^{i_n},\ where\ d_k^{i_k} \in X_k^{i_k}. \tag{2.8}$$

There are two special cubes; the base cube where all the instances are at the lowest level of each dimension and the top, or apex, cube with the maximum level of each dimension. Note the apex cube exists out of only 1 cell.

2.2.3 Navigational Operators

The dimensions and their levels provide the user to view the data from different views. These views can be generated with typical OLAP operators. In this section we discuss the most relevant operators and illustrate them with an example. The opertors we discuss are the drill down, roll up, slice and unslice operators.

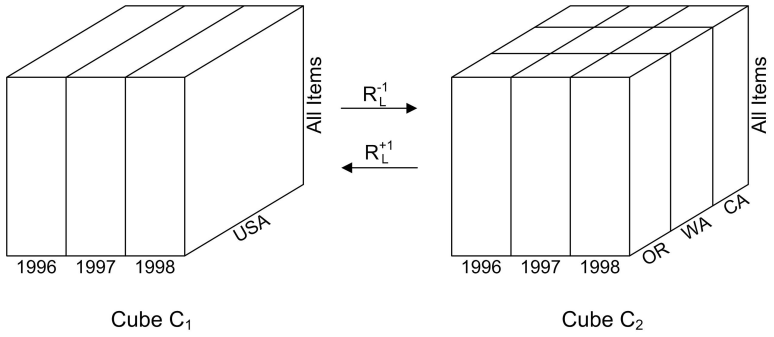


Figure 2.3: Example of the drill down and roll up operators

The *drill-down* operators de-aggregate the cube to a lower, more detailed, level. It is possible to do this by stepping down the dimension hierarchy, or by introducing an additional dimension level [PJ01]. The operator is defined as

$$R_q^{-1}(X_1^{i_1} \times ... \times X_q^{i_q} \times ... \times X_n^{i_n}) = X_1^{i_1} \times ... \times r^{-1}(X_q^{i_q}) \times ... \times X_n^{i_n}. \tag{2.9}$$

This could be seen as the cube C_1 in Figure 2.3 being drilled down over the location dimension with cube C_2 being the result; $R_L^{-1}(Country \times Brand \times Year)$ = *[State, Brand, Year]*. When the cube needs to be aggregated to

an higher, less detailed, level the *roll-up* operator is used. It is the exact opposite of the drill-down operator. The Roll-up operator is defined as

$$R_q^{+1}(X_1^{i_1} \times ... \times X_q^{i_q} \times ... \times X_n^{i_n}) = X_1^{i_1} \times ... \times r^{+1}(X_q^{i_q} \times ... \times X_n^{i_n}. \quad (2.10)$$

This could be seen as the cube C_2 in Figure 2.3 being rolled up over the location dimension with cube C_1 being the result; $R_L^{+1}(State \times Brand \times Year)$ = *[Country, Brand, Year].*

The *slice* operator performs a selection on one dimension's level instance creating a new sub cube.

$$S^{X_q=Y_q}(X_1^{i_1} \times ... \times X_q^{i_q} \times ... \times X_n^{i_n}) = X_1^{i_1} \times ... \times Y_q^{i_q}) \times ... \times X_n^{i_n} \; where \; Y_q^{i_q} \in X_q^{i_q} \quad (2.11)$$

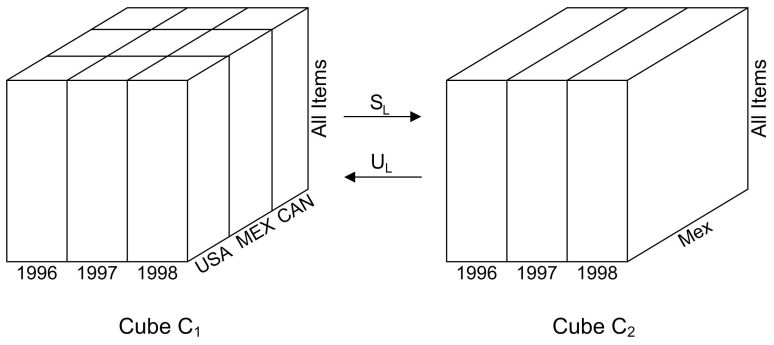


Figure 2.4: Example of the slice and unslice operators

Using the slice operation with the criteria *Location = Mexico* results in the sub cube;
$S^{Loc=Mex}(Country \times Brand \times Year)$ = *[Mexico, Brand, Year].* In Figure 2.4 the slice operator is shown from C_1 to C_2

A combination of of slice operators is called a dice operator. The order of the slice operators doesn't matter because the operator is commutative. Combining the slice operators *Location = Mexico* and *Year = 1997* results in
$MS^{Loc=Mex,Year=1997}(Country \times Brand \times Year)$ = *[Mexico, Brand, 1997].*

The *un-slice* operator performs the reverse of a slice operator, and is given by

$$U^{X_q=D_q}(X_1^{i_1} \times ... \times X_q^{i_q} \times ... \times X_n^{i_n} = X_1^{i_1} \times ... \times D_q^{i_q}) \times ... \times X_n^{i_n}. \quad (2.12)$$

Unslicing the previous example results in the initial cube;
$S^{Mex=Loc}(Mexico \times Product \times Time)$ = *[Location, Product, Time].* This operator can be seen in Figure 2.4 where cube C_2 is unsliced to cube C_1. Performing multiple unslice operators will create the context cell $(d_1^{i_1}, ..., d_n^{i_n})$.

If enough unslice operators are performed the top cube is the final result.

The matrix slice operator is basically the same as a normal slice operator but then on a specific dimension level instance within a hierarchy,

$$S^{A_q^{i_q}=a_q^{i_q}}(X_1^{i_1}\times...\times(A_q^{max_q}.\cdots.A_q^{i_q})\times...\times X_n^{i_n}) = X_1^{i_1}\times...\times(A_q^{max_q}.\cdots.a_q^{i_q})\times...\times X_n^{i_n}. \tag{2.13}$$

The matrix slice over the Location dimension specifies the state. $S^{State=Veracruz}(Country.State \times Product \times Time) = [Country.Veracruz, Product, Time]$.

The matrix unslice operator is the opposite of the matrix slice operator and generalizes the dimension,

$$U^{a_q^{i_q}=A_q^{i_q}}(X_1^{i_1}\times...\times A_q^{max_q}.....a_q^{i_q}\times...\times X_n^{i_n}) = X_1^{i_1}\times...\times A_q^{max_q}.....A_q^{i_q}\times...\times X_n^{i_n}. \tag{2.14}$$

The matrix unslices the Location dimension of previous examples. $S^{Veracruz=State}(Country.Veracruz\times Brand\times Year) = [Country.State, Brand, Year]$.

There are many other OLAP operations such as drill-accross, drill-through and ranking. We will briefly discuss these operations. When cubes share dimensions a drill-across operation is done. This is the same as the join operator [PJ01] The drill-through uses relational SQL facilities to drill through the bottom level of the cube to its back-end relational-tables [HK00]. The ranking operator only returns the cell that return at the top/bottom of a specified order. For example, selection of the top five best sold items.

Using the operators discussed multiple dimensions can be viewed from different perspectives and data can be grouped and selected interactively, making it an ideal analysis tool.

2.2.4 Aggregation lattices

The aggregation lattice L is a graph that is defined by aggregating all the dimension across their dimension levels. The aggregation lattice is also known as a data cube lattice. The aggregation lattice is represented as an ordered tuple. The tuple is the combination of the dimension levels. The base aggregation lattice of a n-dimensional database is s $[0, \ldots, 0]$ and the top level is $[i^{1_{max}}, i^{2_{max}}, ..., i^{n_{max}}]$ [Kam09]. The aggregation lattice level is the sum of all the dimension levels. The maximum lattice level of the running example is $3+2+3 = 8$.

The lattice is defined by the following sequence operations applied to the base cube [Car09]

$$R_q^{+n} = R_q^{+1} \circ R_q^{+1} \circ ... \circ R_q^{+1} \tag{2.15}$$

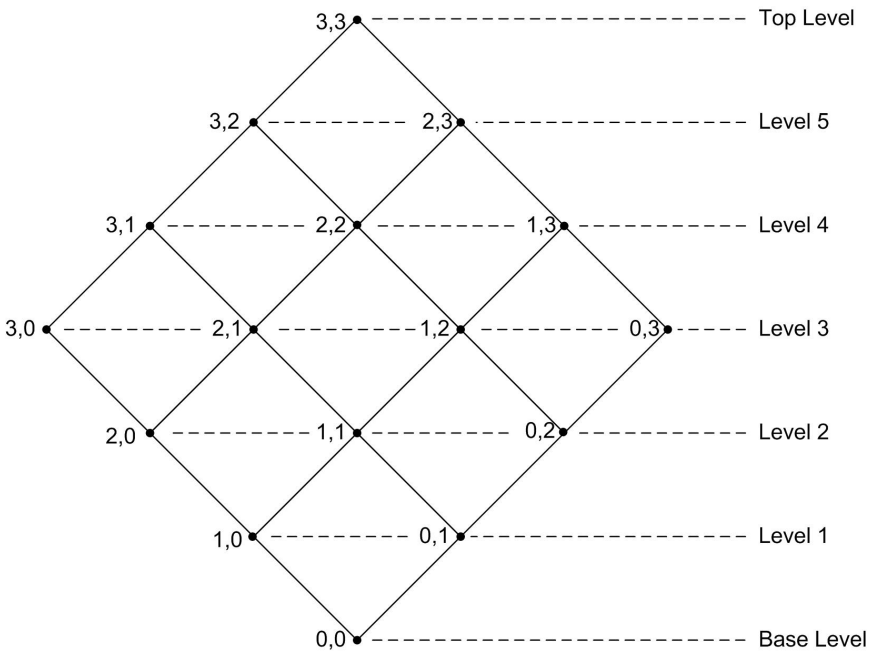


Figure 2.5: An aggregation lattice of 2 dimensions where each dimension has 3 levels

The *upset* of a cube $C = [i_1, i_2, ..., i_n]$ is the lattice of all ancestors of cube C. Where the base cube of this sub lattice is C and the top cube $[i^{1_{max}}, i^{2_{max}}, ..., i^{n_{max}}]$. The different cubes in the sub lattice can be found using the roll up operator. An example of an aggregation lattice is shown in Figure 2.5.

The *downset* of a cube $C = [i^{1_{max}}, i^{2_{max}}, ..., i^{n_{max}}]$ is the lattice of all descendants from cube C. The base cube of the sub lattice being $[0, 0, ..., 0]$ and the top cube being C. The cubes in the sub lattice can be found using the drill down operator [Car09].

2.2.5 Analysis path

In Figure 2.5 there are different roll-up(drill down) operation sequences to reach the top (base)cube from the base (top) cube. The sequence results in an analysis path P. For example [0, 0 ,0] $\rightarrow$ [0, 1 ,0] $\rightarrow$ [0, 1 ,1] $\rightarrow$ [1, 1, 1]. The length of the analysis path is dependant of number of dimensions and the maximum dimension's level. The total number of paths is dependent on the number of nodes in the lattice, and is given by

$$T = \Pi(i^{n_{max}} + 1). \tag{2.16}$$

Accordingly we can calculate the number of possible lattice paths with [Car09]

$$\#\ analysis\ paths = \frac{(n_1 + n_2 + .. + n_k)!}{n_1!n_2!...n_k!}. \tag{2.17}$$

The running example has 3 dimensions with respectively three, two and three levels. The maximum length of the analysis path is 3+2+3 = 8. Which has $\frac{(3+2+3)!}{3! \times 2 \times 3!} = \frac{40320}{72} = 5604$ different paths. This example shows the number of analysis paths rapidly increases as the number of dimensions and dimension levels increase.

2.2.6 Measure

Measures are the measurable variables in the fact tables and are sometimes called facts or variables. Typical business measures are *sales, costs* and *number of items*. A measure y is defined as a function on cube C as follows

$$y^{i_1,i_2,...,i_n} : D_1^{i_1} \times D_2^{i_2} \times ... \times D_n^{i_n}. \tag{2.18}$$

When a measure is related to the base cube then the dimension hierarchies can be used for aggregating (e.g. sum, count, max, avg) the measure values creating different views on the data.

2.3 OLAP equations

OLAP equations are formed by performing specific aggregation functions on a single measure. The most common additive OLAP equations are the sum and count function. The most common non-additive OLAP equation is the average function.

2.3.1 Additive Drill-Down equation

A function is additive when the operation f(x+y) = f(x) + f(y) is valid. For the cube C this implies the measure y can be summarized across all dimensions. This implies y for each cube in the lattice, except for the base cube, the following holds [CD08];

$$y^{i_1...i_q...i_n}(C) = \sum_{j=1}^{J} y^{i_1...(i_q-1),...i_n}(S_q^j(R_q^{-1}(C))). \tag{2.19}$$

In equation 2.19 the measure is the sum of the children corresponding to the cube. For the running example we can calculate the items sold this year using cube $C = Country \times Years \times Brand$ and measure $y = items\ sold$

$$y^{2,2,1}(C) = \sum_{j=1}^{4} y^{2,1,1}(S^{Time=Quarter_j}(R_T^{-1}(C))).$$

If the cube and its cells are instantiated the Equation 2.19 is rewritten to

$$y^{i_1...,i_q...i_n}(..., A, ...) = \sum_{j=1}^{J} y^{i_1...(i_q-1)...i_n}(..., A.a_j, ...). \tag{2.20}$$

Using Equation 2.20 on the running example we calculate the number of 'Faux' Products sold in USA in the year 1997 as

$$y^{2,2,1}(USA \times 1997 \times Faux) = \sum_{j=1}^{4} y^{2,1,1}(USA \times 1997.Quarter_j \times Faux).$$

If it is clear over which dimension(s) the measure is summarized we use the short hand notation. In the short hand notation the summarization over the dimension is represented with the plus sign. For example using cube $C = Country \times Year \times Brand$ we get

$$y(Country, +, Item) = \sum_{j=1}^{4} y(Country, Year.Quarter_j, Item).$$

2.3.2 Non-Additive Drill-Down equation

Non-additive equations are equations that cannot be added across multiple dimensions. The most popular non-additive equation is the average function. The average function is non-additive because the instance average is not equal to the sum of his children's averages. The instance's average is calculated by the summation of its children divided by the number of children [Car09]

$$\overline{y}^{i_1...i_q...i_n}(..., A, ...) = \frac{1}{J} \sum_{j=1}^{J} y^{i_1...i_q-1...i_n}(..., A.a_j...). \tag{2.21}$$

In a more general formulation that holds for each cube C

$$\overline{y}^{i_1...i_q...i_n}(C) = \frac{1}{|R_q^{-1}(C)|} \sum_{j=1}^{J} y^{i_1...(i_q-1)...i_n}(S_q^j(R_q^{-1}(C))). \tag{2.22}$$

2.4 Conclusion

OLAP allows data to be viewed quickly and viewed from different dimensions. The views can be modified by using the operators roll up, drill down, slice and unslice on one or more dimensions. Following a sequence of operators is known as a path. The lattice of the cube exists of all the possible paths between the base and sub cube. Because the data can follow different paths different views can be created allowing the analyst to make better informed decisions.

Chapter 3

Explanation generation

3.1 Introduction

OLAP systems do not have standard functionality for the automated automated explanation of exceptional cells. Currently the analyst has to use manual drill down and roll up operators to discover reasons for the exception.

In this chapter the OLAP system is extended with an automated explanation formalism. The formalism finds the best explanation of unexpected behaviour, symptoms, of a system under study. The notation is based on the literature of [CD04].

There are two important classes of research in this. The first is from Sarawagi [SAM98]. In Sarawagi an operator was presented that finds a summarized reason for drops or increases observed at an aggregated level. They introduced a model which compared the actual value with the expected, reference, value and the compact summary of the difference table A, with for each row of A being the ration between y^a and y^r. The idea of Surawagi is to find A in such a way that the reference value built up from table A and the actual value incurs the smallest amount of errors as possible. The second class in based on Feelders and Daniels notion. We explain this notion in more detail in this chapter.

3.2 Explanation Formalism

Our exposition on causal explanation is based on Feelders & Daniels notion of explanations [FD01], which is based on Hesslow's theory of explaining differences [Hes84] and Humpreys' aleatory explanation notation [Hum97]. The explanation takes place from the whole to the parts. In the OLAP context the formalism explains why the event's actual value is different than its reference value. The causal explanation is written as

$$< a, f, r > \texttt{ Occurred because } \textit{Cb} \texttt{ despite } \textit{Ca}. \tag{3.1}$$

The formalism describes a symptom as a three-place relation between an actual object a, a property f, and a reference object r. The reference object is obtained from a normative model R. This model is a managerial or statistical normative model. An example of a managerial normative model is an historical model and an example of a statistical normative model is a multi-way ANOVA model.

The explanation itself consists of a non-empty set of *contributing* causes, *Cb*, and a, possibly empty, set of *counteracting* causes, *Ca*. The counteracting causes do not explain the event, however they clarify the contributing causes.

In the OLAP context property f is measure y. Now the analyst wants to specify the difference between the actual value and reference value as a qualitative difference. The qualitative difference δy can be { 'high', 'normal', 'low' } and are determined as follows

$$\begin{aligned} &y^a(c) > y^r(c) \rightarrow \delta y = \texttt{`high'}, \\ &y^a(c) = y^r(c) \rightarrow \delta y = \texttt{`normal'}, \\ &y^a(c) < y^r(c) \rightarrow \delta y = \texttt{`low'}. \end{aligned} \tag{3.2}$$

An explanation is given when $\delta y =$ 'high' or $\delta y =$ 'low'. We can now rewrite Equation (3.1), which is used for explanation of $\delta y(c)$ in a cube C,

$$< y^a(c), \delta y = \{high, low\}, y^r(c) > \ \texttt{occurred because Cb}\textit{despite}Ca. \tag{3.3}$$

3.3 Influence Measure

To quantitatively determine the contributing an counteracting causes that explain the qualitative difference between the actual and reference value a measure of influence is given by [FD01]

$$\texttt{inf}(x_i, y) = f(\mathbf{x}^r_{-i}, x^a_i) - y^r, \tag{3.4}$$

where $\inf(x_i, y)$ indicates what the difference between the actual and reference value of y would have been if only x_i would have deviated from its reference value. The influence measure should satisfy the conjunctiveness constraint. This constraint says that the influence of a single variable cannot turnaround when it is with the influence of multiple variables [FD01]. Furthermore, the function f is additive in the OLAP context because an OLAP datacube is a system of additive drill down equations, see Equation 2.19. Now we can rewrite Equation 3.4 as

$$\begin{aligned} &\texttt{inf}(y^{i_1 \ldots i_q \ldots i_n}(c), y^{i_1 \ldots (i_q+1) \ldots i_n}(R^{+1}_{D_q}(c))) = \\ &y^{a;i_1 \ldots i_q \ldots i_n}(c) - y^{r;i_1 \ldots i_q \ldots i_n}(c). \end{aligned} \tag{3.5}$$

Equation 3.5 also holds the transitivity property. The *transitivity property implies that the influence of a variable* $y^{i_1 \ldots i_q \ldots i_n}(c)$ *on its parent* $y^{i_1 \ldots i_q+1 \ldots i_n}(R^{+1}_{D_q}(c))$ *is equal to its influence on any other ancestor in its upset.* The proof of this property can be foud in Caron's paper [Car09].

Using the influence measure the causes can be quantified as contributing or counteracting causes, defined by [FD01].

$$\begin{aligned} &\texttt{inf}(y(c), y(R^{+1}_{D_q}(c)) \times \delta y > 0 \rightarrow Cb \\ &\texttt{inf}(y(c), y(R^{+1}_{D_q}(c)) \times \delta y < 0 \rightarrow Ca. \end{aligned} \tag{3.6}$$

Now we give an example where we apply the concepts introduced above.

Example 3.1 The analyst wants to explain why the profit of USA × All Items × 1997.Q4 is higher than the profit of USA × All items × 1997.Q3, the symptom is given by

⟨profit(USA, All Items, 1997.Q4), 'high', profit(USA, All Items, 1997.Q3)⟩.

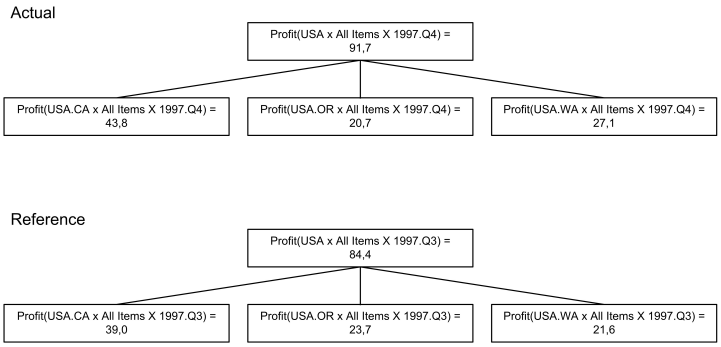


Figure 3.1: Actual and reference objects for the cell (USA × All Items

The analyst wants the difference to be explained at the state level in the location dimension. First the actual value and reference value for the states are determined, as depicted in figure 3.1. With these values the analyst calculates the influence for each state.

Dimension	profit$(y^a(c) - y^r(c))$	Influence
USA.CA	27.11 - 23.72	3.39
USA.OR	20.71 - 21.61	-0.89
USA.WA	43.84 - 39.04	4.80

From the table it can be concluded that the following set of causes Cb = {USA.CA, USA.WA} and Ca = {USA.OR} are contributing causes. The set of counteracting causes exists of { USA.OR }

3.4 General explanation generation

The general approach for the explanation of an exceptional cell consists of three main steps

1. The creation of an aggregated table with all possible causes;
2. The selection of significant causes using a greedy heurist or top-down heuristic;
3. The creation of the explanation tree.

The general approach is described in Algorithm 3.1. In the remainder of Section 3.4 we elaborate on each of the algorithm's basic steps.

Algorithm 3.1 General Explanation generation

Select cell to be explained
Select heuristic for Aggregated Table
Create Aggregated Table
Select Significant causes
Create Explanation Tree

3.4.1 The aggregated table

In the previous section influence of a child on an ancestor in the upset was explained. In the explanation generation proces all the objects in the symptom's downset are considered. For this purpose the aggregated table has to be determined. The aggregated table is an overview of all possible causes in a drill down path with their actual value, reference value and influence. We use a standardized lay-out for the influence table. The first columns shows the dimension levels used to explain a cause. If an object does not use a certain dimension level, the description 'All' is filled in the dimension level instance cell. Next to these columns there is a column for the object's actual value and for the reference's value. The last column shows the influence measure. In Figure 3.2 the structure of an aggregated table is shown. It is possible multiple dimensions are used in the aggregated table.

D_q^{i-1}	D_q^{i-2}	...	D_q^{0}	Actual Value	Reference Value	Influence

Figure 3.2: Example of the aggregated table structure for a single dimension $D_q^{i_q}$

Because it is often impossible to check all the paths in the symptom's downset, different heuristics are applied to determine the drill down path. With the drill down path the aggregated table is created. These different heuristics derermine which part of the downset are significant enough to be considered. The heuristics are discussed in Section 3.4.2 and Section 3.4.3.

3.4.2 Selection of significant causes

The first step is to sort the influence values in the aggregated table in descending order to find the largest contributing causes at the top of the aggregated table and the largest counteracting causes at the bottem of the aggregated table. Different sorting algorithms can be used for this, e.g. bubblesort algorithm [LC06]. After sorting the causes, the analyst can easily select the top n largest contributing/counteracting causes from the aggregated table.

The selection of significant causes can be done in several ways. The most straightforward approach is to select the top n contributing causes of the aggregated table. We use this approach in our thesis. But in some cases the analyst also wants to view the top m counteracting causes of the aggregated table. The counteracting causes can help the analyst to clarify the set contributing causes. Another selection approach is to select the top n % contributing causes of the total number of contributing causes in the aggregated table, and top m % counteracting causes of the total number of counteracting causes in the aggregated table.

3.4.3 Explanation tree

The explanation tree is a tree of causes with the explained cell as the root of the tree. Its successor nodes are the selected causes. The explanations are chained together from one level to the next level and a tree is formed. Counteracting causes are denoted with a dotted line. In Figure 3.3 an example explanation tree is given that shows how the profit is explained over the location dimension.

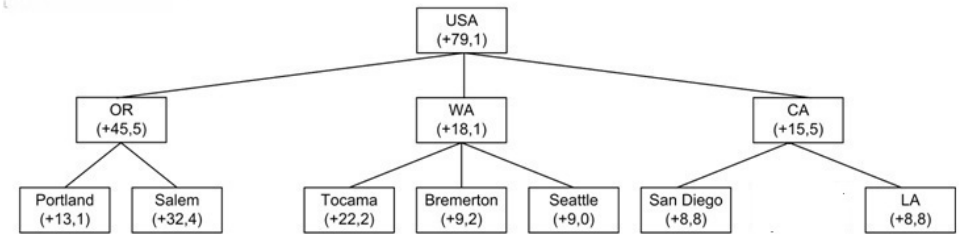


Figure 3.3: An explanation tree from the running example over the location dimension with USA as the root cell, with n=10

The generation of the explanation tree is shown in algorithm 3.2. The algorithm generates an explanation tree based on the set of significant causes and the symptom that is explained. For each instance in the set of significant causes the algorithm finds the nearest ancestor in the set, or the symptom. If the nearest ancestor is also the parent of the instance an edge is added between the two instance. If the nearest ancestor is not the parent of the instance all the instances between the nearest ancestor and instance are added to the set. Between each child and parent an edge is added. Because the symptom is at the top level of the explanation tree we are certain an explanation tree is created.

Algorithm 3.2 Explanation Tree

```
Set of significant causes AND symptom → SetInst
for each sinificant cause do
  Find nearest ancestor in SetInst
  if ancestor ≠ parent cell then
    Add instances between significant cause and ancestor in SetInst.
    Add edge for each between child and parent in path between signifi-
    cant cause and nearest ancestor.
  else
    Add Edge beyween cause and ancestor
  end if
end for
```

3.5 Greedy Explanation Generation

Here we discuss two heuristics applicable for explanation generation; dimension hierarchy heuristic and user defined heuristic.

3.5.1 Dimension hierarchy heuristic

The dimension hierarchy heuristic creates an aggregated table that considers only one specific dimension hierarchy. In this heuristic a cube is drilled down over a single dimension. For example, to generate an explanation over the dimension D_q. Because only one dimension is selected the path in the downset is always clear.

This is an heuristic that is useful for an analyst when he is focussed on explaining a symptom over one specific dimension. The analyst often wants the strongest causes in one dimension shown. For example, the analyst only wants the measure profit to be explained over the Time dimension, and is not interested in an explanation over the other dimensions.

The input for the algorithm is one dimension $D_q^{i_q}$ from the symptom's downset. The heuristic fills in the dimension level instance names and for

the columns that are used for lower dimension levels are filled in with 'All'. After that, it calculates the actual value, reference value and influence for the causes in the subcube.

The heuristic drills down one dimension level. It finds all the instances at the new dimension level and calculates the actual value, the reference value and the influence for all the instances. The columns that are used for the lower dimension level are filled with 'All'. This is repeated until the dimension level is 0. The heuristic in pseudocode is given in Algorithm 3.3

Algorithm 3.3 aggregated table based on one dimension

```
Define cube → C
Define drill down dimension → D_q
Define current dimension level → i_q
Create Empty Aggregated Table
Number of columns in table = i_q + 4
while i_q ≥ 0 do
  for ∀ instances ∈ D_q^{i_q} do
    fill in all known dimension levels
    column D_q^{i_q − 1} to column D_q^0
    Fill in 'All'
    calculate y^a(C), y^r(C)
    store values in correct columns
    Inf = y^a(C) − y^r(C)
    store values in correct column
    next row in aggregated table
  end for
  i_q = i_q -1
  C= R_q^{-1}(C)
end while
```

Example 3.2 Here we present an example where we construct the aggregated table for the explanation of ⟨profit(USA, All, Q4), 'high', profit(USA, All,Q3)⟩. The Aggegrated table is created over the Location dimension path and is shown in Figure 3.4.

country	state	city	1997 Q4 (actual)	1997 Q4 (reference)	Influence
USA	All	All	91,67	84,37	7,30
USA	CA	All	27,11	23,72	3,39
USA	OR	All	20,71	21,61	-0,89
USA	WA	All	43,84	39,04	4,80
USA	WA	Bellingham	0,91	0,64	0,27
USA	CA	Beverly Hills	8,90	6,08	2,82
USA	WA	Bremerton	8,76	7,70	1,06
USA	CA	Los Angeles	9,16	8,35	0,81
USA	OR	Portland	8,91	7,06	1,86
USA	OR	Salem	11,80	14,55	-2,75
USA	CA	San Diego	8,34	8,59	-0,24
USA	CA	San Francisco	0,71	0,71	0,00
USA	WA	Seattle	8,48	7,98	0,50
USA	WA	Spokane	8,15	7,69	0,46
USA	WA	Tacoma	12,72	11,21	1,51
USA	WA	Walla Walla	0,86	0,67	0,18
USA	WA	Yakima	3,96	3,14	0,82

Figure 3.4: Aggregated table for location dimension

3.5.2 User Defined heuristic

In the second heuristic the aggregated table considers multiple dimensions from the exceptional cell's downset. The analyst has predetermined the analysis path. For example, such path can look like [2,1,2] → [2,1,1] → [1,1,1] → [0,1,1]. It is possible to display the path in matrix notation. The analysis matrix columns represent dimensions of the cubes, $D_1, D_2, ..., D_n$ and the rows represent the levels of lattice L and shows over which dimension the heuristic drills down one level. The path in the example is given by the list of causes

$$\left(\begin{array}{c|ccc} \text{Level} & L & T & P \\ 7 & 0 & -1 & 0 \\ 6 & -1 & 0 & 0 \\ 5 & -1 & 0 & 0 \\ & & & \end{array}\right) \quad (3.7)$$

In 3.7 each column represents 1 dimension, e.q. Location, Time and Product, and are defined in the first row. In the second row the first drill down of the path is selected. The selected dimension is shown with -1. The second drill down is shown in the third row and so on. This is heuristic is useful for an analyst when he knows which dimension are important and which sequence the dimensions should be drilled down over. For example, the analyst drills down the Time dimension level quarter and than continue with the Location dimension state to finish the path with the Time dimension level month.

Similar to the one dimensional heuristic the user defined heuristic fills the dimension level instance names and the columns that are used for lower dimension levels are filled in with 'All'. It then calculates the object value, the reference value and the influence of the selected (sub)cube. This is repeated until the last step of the path is reached. The heuristic in pseudo code can be seen in Algorithm 3.4.

Algorithm 3.4 User defined path aggregated table

```
Define subcube → C
path → [P_1, P_2, ..., P_n]
Create Empty aggregated table
for i = 1 to length Path do
  for ∀ instances ∈ C_q do
    next row in aggregated table
    store in all known dimension levels column D_q^{i_{q-1}} to column D_q^0
    Fill in 'All'
    calculate y^a(C)
    calculate y^r(C)
    calculate Influence
    store values in the correct columns
  end for
  Determine next dimension selected in path → q
  C = R_q^{-1}(C)
end for
```

Example 3.3 Here we present an example where the analyst wants to construct an aggregated table for ⟨profit(USA, All, Q4), 'high', profit(USA, All,Q3)⟩, based on the following predefined path: [2,1,2] → [2,1,1] → [1,1,1] → [0,1,1]. Using this path the aggregated table is shown in Figure 3.5.

3.6 Top Down Explanation

The top down algorithm is a different approach than the greedy algorithm. The greedy algorithm calculates the influence of each cause and than selects the top n significant causes. The top down algorithm breaks down the calculation en selection of significant causes per level. The top down approach only uses the selected significant causes to calculate the next significant causes.

With the greedy algorithm the analyst was certain the most significant causes were always found, but with the top down approach a significant cause can not be shown if the parent is not a significant cause.

country	state	city	brand_name	product_name	1997.Q4 (actual)	1997.Q4 (reference)	Influence
USA	All	All	All	All	91,67	84,37	7,30
USA	All	All	ADJ	All	0,07	0,08	-0,01
USA	All	All	Akron	All	0,11	0,09	0,02
USA	All	All	American	All	0,90	0,78	0,12
USA	All	All	Amigo	All	0,20	0,15	0,06
USA	All	All	Applause	All	0,17	0,15	0,02
USA	All	All	Atomic	All	0,52	0,45	0,07
USA	All	All	BBB Best	All	1,65	1,53	0,13
USA	All	All	Best	All	0,25	0,21	0,03
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
USA	CA	Los Angeles	Washington	All	0,08	0,03	0,05
USA	CA	San Diego	Washington	All	0,04	0,08	-0,03
USA	CA	San Francisco	Washington	All	0,01	0,01	-0,01
USA	OR	Portland	Washington	All	0,07	0,05	0,01
USA	OR	Salem	Washington	All	0,06	0,09	-0,03
USA	WA	Bellingham	Washington	All	0,01	0,00	0,01
USA	WA	Bremerton	Washington	All	0,08	0,04	0,04
USA	WA	Seattle	Washington	All	0,02	0,07	-0,05
USA	WA	Spokane	Washington	All	0,04	0,05	0,00
USA	WA	Tacoma	Washington	All	0,09	0,08	0,00
USA	WA	Walla Walla	Washington	All	0,02	0,00	0,01
USA	WA	Yakima	Washington	All	0,04	0,01	0,03

Figure 3.5: Aggregated table created with user defined heuristic

3.6.1 Specificity heuristic

In some cases the analyst wants to take multiple dimensions into account but does not know what the best path should be. The specificity heuristic automates the path selection. This is an heuristic that can be useful for an analyst when the analyst has no preference and wants to determine the most specific explanation.

The specificity heuristic works as follows. The symptom is drilled down in each possible dimension. It than counts the possible causes in each dimension. After that, it selects the minimum possible number of causes for each dimension to explain the symptom for a certain fraction, T^+. The reduced set of causes in a dimension is known as the set of contributing causes (Cb_p) [Car09].

$$\frac{\texttt{inf}(Cb_p, y^{i_1\ i_2 \cdots\ i_n}(C)}{\texttt{inf}(Cb, y^{i_1\ i_2\ \cdots\ i_n}(C)} \geq T^+. \tag{3.8}$$

T^+ is normally between 0.7 and 1, or so. If the analyst wants to increase/decrease the number of causes in the parsimonious set the analyst can increase/decrease T^+. The counteracting parsimonious set is similar to the contributing parsimonious set. After that the heuristic calculates the measure of specificity for the symptom. The specificity measure is defined as

$$\texttt{S} = \frac{\sharp\texttt{possible causes}}{\sharp\texttt{actual causes}}. \tag{3.9}$$

The number of actual causes is the number of causes that is in the parsimonious subset. In general the explanation with the highest specificity measure is preferred.

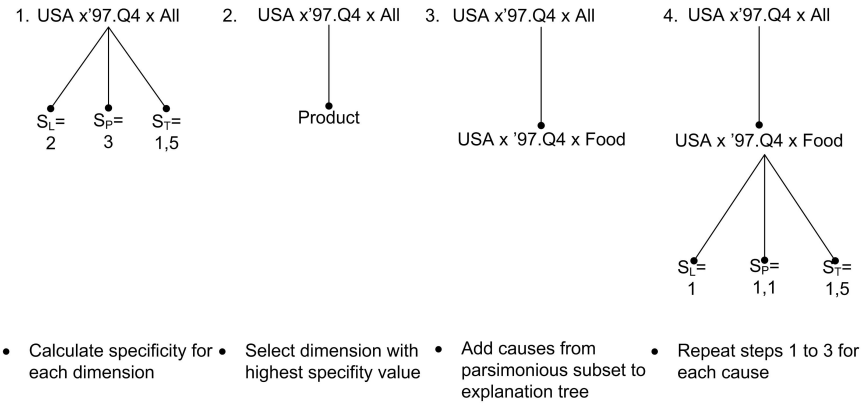


Figure 3.6: A possible built up of an explanation using the specificity heuristic

Once the dimension is selected the heuristic drills down this dimension and calculates the influence of all the instances in the parsimonious subset. To determine the next dimension for each cause the heuristic repeats the specificity measure. An example of how an explanation tree would be createdusing the specificity heuristic is shown in Figure 3.6. The heuristic is shown in algorithm 3.5.

Algorithm 3.5 Specificity aggregated table

Define symptom $\rightarrow$ c
Define max depth explanation tree $\rightarrow$ maximum
Cb_{TMP} = c
while $Cb_{TMP} \neq 0$ AND length $\leqslant$ maximum **do**
  $Cb_{NEW} = Cb_{TMP}$
  $Cb^{+}_{TOT} = Cb_{TOT} + Cb_{TMP}$
  Cb_{TMP} = empty
  length = 0
  for all causes in Cb_{NEW} **do**
    Calculate specificity for each dimension
    Select dimension with highest specificity cell
    Drill down cause over selected dimensions
    Add parsimonious set of cause to Cb_{TMP}
    length = length + 1
  end for
end while
create explanation tree based on Cb_{TOT}

Example 3.4 The analyst wants an explanation for the profit in the USA × 1997.Q4 × All items compared with the profit in the USA × 1997.Q3 × All items. The analyst does not know which path can be follow best, and uses the specificity heuristic to generate the explanation tree. The analyst selrcts $T^+ = 0.7$. The heuristic first calculates the influence in USA × 1997.Q4 × All. The influence is 7.30. The minimum parsimonious subset should at least have explained the influence for 0.7 * 7.30 = 5.11. After that, the specificity heuristic calculates the specificity value of the location dimension and product dimension. The location dimension specificity was 1.5 and the product dimension specificity was 4.11. The heuristic selects the dimension with the highest specificity value. It therefore drills down the product dimension and calculates the influence for each instance in the parsimonious subset. It then repeats this for each instance in the parsimonious subset. The heuristic continues untill a stop condition is met. A stop condition could be the maximum depth of the explanation tree.

One of the cause in the parsimonious set is the product 'Dairy'. Here time dimension specificity was 1.5 and the product dimension specificity was 1.67. For this cause the drill-down path is over the product dimension. But this does not mean all other products also are drilled down over the product dimension.

3.7 Conclusion

The explanation formalism introduced in section 3.2 makes it possible to view an symptom in the OLAP datacube as a general problem. The explanation formalism shows the explanation represented as an explanation tree which makes it intuitive for inspection by the analyst. The explanation formalism and creation of the explanation tree can be created using one of the heuristics in 3.5 and 3.6. Although the ideas behind the heuristics are different, both heuristics create a rooted explanation tree. These different paths could cause three major different explanation trees for one symptom in a datacube.

The explanation tree heuristics makes it possible to generate multiple explanation trees for a cell its context. A set of explanation trees can be used to detect similarity patterns.

Chapter 4

Detecting similarity patterns

In this chapter we discuss the basic notions of graph matching. We introduce the basic ideas behind graph matching and give an overview on what has been done in the literature related to this topic. We than explain two algorithms that are used to detect tree similarity. Subsequently, we explain how these algorithms can be applied in an OLAP datacube to detect similarity patterns.

4.1 Introduction

Graphs are used in many subfields of science to represent the structure of an object. Example of such subfields are chemistry, engineering and databases. When patterns or graphs are compared with each other, for example comparing molecules with each other, this is known as graph similarity [Lev72]. There are many approaches found in the literature to solve the graph matching problem.

There are *subgraphs* that allow a certain degree of error. Most of these algorithms are based on the A* algorithm. These algorithms use different heuristic lookahead techniques to calculate the similarity between two subgraphs, in order to generate the similarity graph. These methods will find the optimal solution but need exponentional time and space due to the NP-completeness problem [Bun09].

Another approach is based on approximating the optimal solution to find the similarity graph. The Hopfield network [FLD94] uses neural networks to approximate the optimal solution. Another method uses a genetic algorithm to find the optimal solution [CWH96]. These methods do not guarantee to find the optimal solution but find a solution close to the optimal solution. With this approach it is possible to get stuck in a local minima.

The last sort of approaches use eigendecomposition and linear programming. This approach will find the optimal solution but can quickly get complicated and is not intuitively.

4.2 Graph Similarity basic notions

In this section we discuss the topic of graph similarity in graph theory. There are a couple of standard concepts in graph similarity that we first have to introduce, such as *graph isomorphism, subgraph isomorphism* and *maximum common subgraph*, before we can continue with our discussion on how graph similarity is detected.

The graphs G_1 and G_2 are isomorphic if there is a one-to-one mapping of the graph nodes that preserves the structure of the graph.

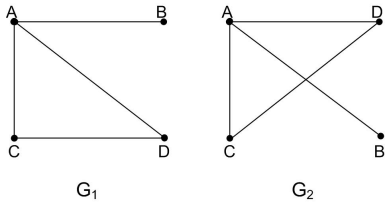


Figure 4.1: Example of isomorphism between a graph G_1 and G_2

More formally the graphs are isomorphic if a bijective function exists with for every node in G_1, u, there is exactly one node in G_2, v, such that $f(u) = v$ exists. In Figure 4.1 an example of isomorphism between two graphs is given.

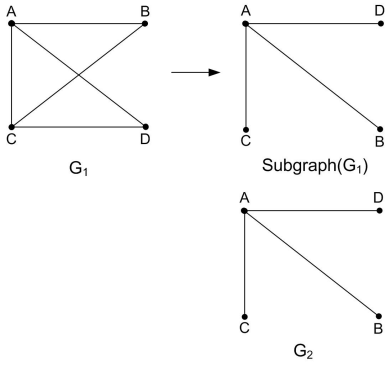


Figure 4.2: Example of subgraph isomorphism

Subgraph isomorphism occurs when the graph of G_1 and a subgraph of G_2 are isomorphic. A subgraph of graph G is a graph whose set of nodes and edges are a subset of G's set of nodes and edges. In Figure 4.2 the subgraph of G_2 is isomorphic with graph G_1.

The last notion is the maximum common subgraph. A common subgraph of G_1 and G_2 is the subgraph that is a subgraph of both graphs.

The maximum common subgraph, G_{mcs}, is the common subgraph with the maximum number of nodes compared to all the other common subgraphs. In figure **??** the maximum common subgraph is shown for the graphs G_1 and G_2.

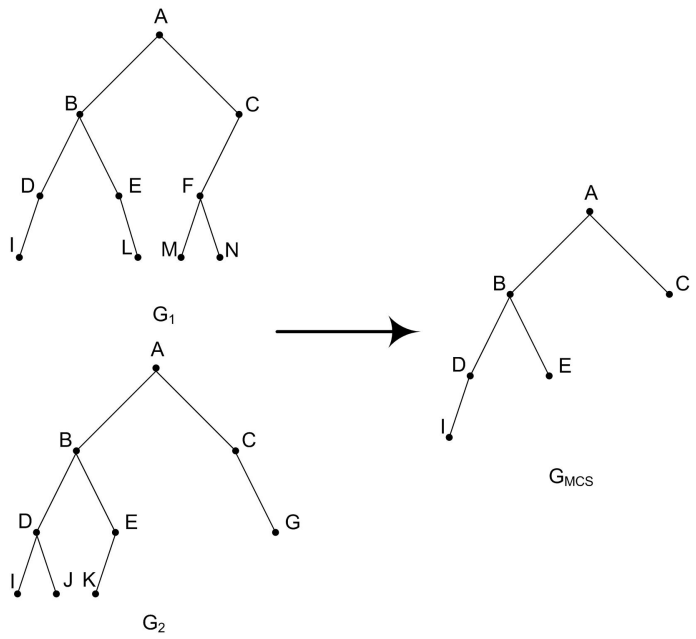


Figure 4.3: Example of maximum common subgraph

In an OLAP datacube the explanations generated are always structured rooted trees, see chapter 3. This means the graphs are acyclic and one node forms the root. The edges have a natural orientation away from the root. Therefore we can simplify the graph matching to a breadth first approach.

4.3 Tree Similarity

In this section we discuss two different sorts of tree similarity. The first approach is based on structural similarity and the second approach is based on weighted structural similarity. In both cases we assume a given set of trees, $(T_i|i = 1...n)$, where all trees are rooted trees. We also assume that each cell in the set follow a similar drill down path.

4.3.1 Structural similarity

Because a node always has one parent we simplify the finding of the similarity tree, and find the similarity tree with a breadth first approach. The depth first approach is also possible, However, we select to adopt a breadth

Algorithm 4.1 Structural similarity

```
Input set of trees {T_1...T_n}
Input minimum support threshold → δ
Intitialize similarity tree; root node of trees
Get maximum level of trees → i_max
for dimension level i = 1 to i_max do
  Find all possible unique nodes in dimension i → set of nodes
  for all nodes in set of nodes do
    Calculate support
    if parent node ⊈ similarity tree OR support ≤ δ then
      Remove node from set of nodes
    end if
    Add nodes from set of nodes to similarity tree
  end for
  Empty set of nodes
end for
Output; show T_MCS
```

first approach. Than all nodes at level *i+1* related to the successful nodes at level i are considered. This is repeated until the top level is reached, or when no new nodes are detected that are suitable for the similarity tree.

More formally similarity is defined as the set of common contributing causes, i.e. common edges between the similarity trees. Similarity between the sets of contributing causes for expceptional cell values in the set of trees $\partial y(c_1)$, $\partial y(c_2)$, ... $\partial y(c_n) \cap$ in the context cube C $= i_1, i_2, ..., i_n\,]$ is given by

$$Cb_{sim} = \cap Cb_p(\gamma y(c_i)) \tag{4.1}$$

If we want to add an support to determine whether a node is succesfull we calculate the support a node has in all the explanation trees. A node is seen as a successful node if the node has a support above a certain threshold. The support for an arbitrary node u in a set of trees, ST is given as

$$\texttt{sup}(u, ST) = \frac{1}{n} \sum \sigma(u, T_i), \tag{4.2}$$

where $\sigma(u, T_i) = 1$ if the node u can be found in T_i, otherwise $\sigma(u, T_i) = 0$. The support is compared with a threshold. If the support is higher than the threshold the node is added to the similarity tree. If the threshold is set to 1 the similarity tree is equal to the maximum common subgraph.

Because the first part of the structural similarity is a special case of the second part we give an algorithm for detecting the similarity tree with a support in Algorithm 4.1.

The general problem of finding similar graphs is reduced to the problem of finding similar trees. The tree is always an acyclic graph where each

nodes has zero or more children and at most one parent. The advantage of this approach is that it will always find the maximal common subgraph. But the disadvantages are the memory requirements and the execution time. For small OLAP datacubes this is not an issue, however for large datacubes with large explanation trees it could cause some problems.

4.3.2 Weighted similarity

In the second approach we take the weights of the nodes into account. The weight is the relative influence compared to parent's influence. In the OLAP context the weight is the relative influence value a cell has on its ancestor. This algorithm is based on Yang's algorithm [YBV05] which is a weighted tree algorithm for similarity matching between buyers and sellers. The approach calculates the edges' weights. This has the advantage that cells with different sizes can be compared. For example, it is possible to compare USA with the Netherlands. The fraction is the influence of the child relative to its parent influence, we also make a difference between contributing and counteracting causes.

Algorithm 4.2 Structural weighted similarity

```
Input set of trees {T_1...T_n}
Input minimum support threshold → δ
Input maximum spread
Run Algorithm 4.1 to create structural similarity
Calculate standard deviation for each edge in the structural similarity tree
for All edges in structural similarity tree do
  if standard deviation edge ⩾ maximum spread then
    Remove edge
  end if
end for
Add nodes to ensure all nodes are connected with parent
Calculate Average weights of each edge
Output; Structural weighted similarity tree
```

If the two trees T_1 and T_2 are compared and T_1 does not have a node that tree T_2 does have, then the fraction for that particular node in T_1 is 0.If the node would be considered in the previous algorithm, the spread of the normalized weights are calculated as the standard deviation. If the edge's spread is below a Maximum spread value and the corresponding node has a support higher than the minimum threshold, the node is added to the similarity tree. If the maximum spread value would be set to 1, the weighted similarity tree would be the same as the structural similarity tree. The general algorithm for the structural weighted similarity is shown in Algorithm 4.2

4.4 Detecting similarity in OLAP

In the OLAP datacube we detect similarity patterns between cells that are in each other's context. In a sub cube of the aggregation lattice multiple explanation trees are created for a number of cells that are in each others context. The context can be one, or a combination of, dimensions. Detecting similarities between explanation trees in OLAP might answer typical business questions. For example finding an explanation why the first six months performed worse structurally. In this section we discuss in detail how similarity patterns in OLAP are discovered. The detection of similarity patterns can be divided into four steps;

1. determine the context,
2. selection of the appropriate explanation tree heuristic for all cells in the context,
3. construction of the maximum common subtree,
4. display similarity tree.

We now discuss the steps in more detail.

First the analyst selects an arbitrary cell in the cube and selects the desired context. The selected context is obtained using the unslice operator, or a combination of unslice operator (see Equation 2.12). The set of cells in the neighbourhood of the selected cell is denoted as C'. the number of cells in C' is denoted as $|C'|$.

For each cell in the set C' an explanation tree is created. There are in total $|C'|$ explanation trees generated, one for each cell in the context. The explanation trees are generated by Algorithm 3.1. The generation of the explanation trees is restricted to the application of the same heuristic to create the aggregated table. This restriction is needed to guarantee the structure of all the explanation trees in the set are the same. If the restriction would not apply, it would be possible that different tree structures are present in the set of explanation trees. If these explanation trees would be compared with each other a strange, or none, pattern would be found.

This explanation tree is similar in structure as a normal tree defined in section 4.3. The causes in the explanation trees are similar to the nodes of a normal tree. The normalized influence values are similar to the weighted edges of a weighted tree. Assuming this is correct, we can use Algorithm 4.1 or Algorithm 4.2 to find the (weighted) structural similarity tree. If the minimum support is set to 1, the similarity tree is equal to the maximum common subtree.

The similarity tree is finally shown graphically. The four steps described are shown in pseudocode in algorithm 4.3.

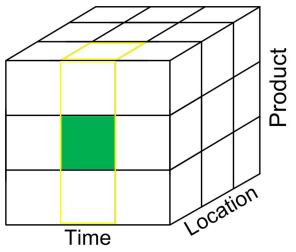


Figure 4.4: Shows the cells of a cube with three dimensions. The yellow lines show a possible selection the analyst can make

In Figure 4.4 the green cell is selected and is unsliced over the product dimension resulting in the selection shown with the yellow lines. The analyst creates an explanation tree for each cell in C' this would result in three explanation trees. In the next step the Algorithm 4.1 or Algorithm 4.2 is used to find the similarity pattern between these three explanation trees. The similarity pattern shows the structural similarity between the cells instead of showing the most important causes between the three cells.

Algorithm 4.3 Create Similarity tree in OLAP

Input; minimum support threshold $\rightarrow \delta$
Input; heuristic for aggregated table
Input; Select type similarity
Set of selected cells $\rightarrow C'$
for i = 1 to $|C'|$ **do**
  Create Explanation tree with selected heuristic using Algorithm 3.1 $\rightarrow T_i$
end for
if Selected similarity is structural **then**
  Create similarity tree using Algorithm 4.1 $\rightarrow T_{mcs}$
else
  Create similarity tree using Algorithm 4.2 $\rightarrow T_{mcs}$
end if
Output; T_{mcs}

4.5 Conclusion

There are many algorithms for the detection of the similarity pattern between graphs. These algorithms have to take into account that the graphs are cyclic and the structure is unpredictable. Because in an OLAP datacube the explanation tree structure is predictable the tree similarity is more simple to solve. In this thesis the problem is solved using a breadth first approach.

The algorithm ensures that all possible nodes are considered. The children of a node that is not in the similarity tree will not be considered. Adding the weights of the edges to find similarity patterns can be regarded as a restriction on the normal pattern similarity, but gives extra information to the tree which can be valuable.

Chapter 5

Software Implementation

In this chapter we present the most important concepts of the prototype software implementation in MS Excel and MS Access. We give these concepts to get a better understanding of the case study in the next chapter.

5.1 Use Case diagram

The use case describes how the analyst creates the pivottable. It also shows how the explanation trees and similarity trees are created. We assume the data needed for the pivottable is stored in access. In the use case diagram the anlayst is the only actor.

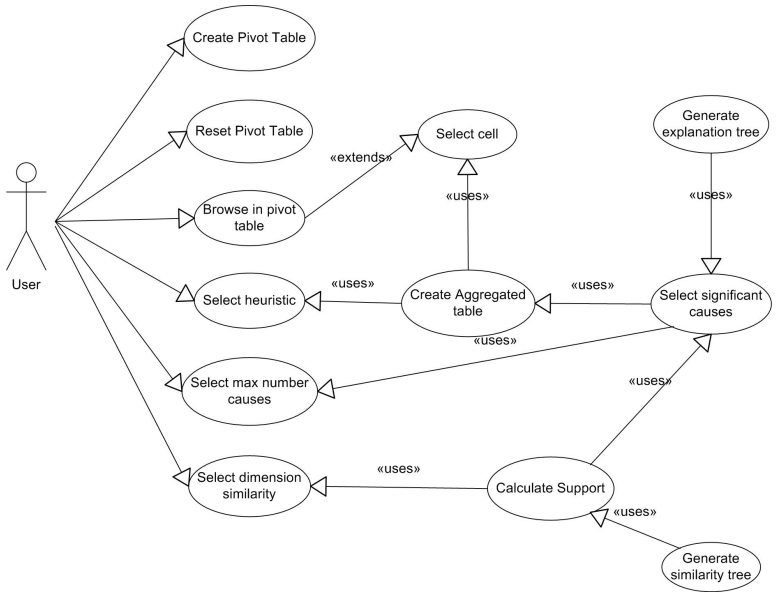


Figure 5.1: Use case diagram

The analyst can create a new pivotTable. With this pivot table he can browse through the data. He can use the different navational operators to browse through the pivot table. With one simple click on a button he can reset the pivot table.

There are two extensive flows in the use diagram. The first flow is the generation of an explanation tree. The second flow is the generation of the similarity tree. The first flow starts with the selection of a cell. The analyst also selects the heuristic that has to be used and determines the number of causes (n) shown in the explanation tree. With the input an aggregated table is created. For the selected cell the top n significant causes are selected. With these significant causes an explantion tree is generated.

With the second flow the similarity tree is created. The analyst select the dimension over which the similarity tree is created. For each cell in the dimension level the significant causes are calculated. These significant causes are compared with each other and the similarity tree is created.

5.2 The class diagram

Figure 5.2 depicts the class diagram of the software program. This diagram shows the OLAP datacube and explains how tree similarity fits in. The model itself is made up by measures and the dimensions are made up by dimension hierarchies.

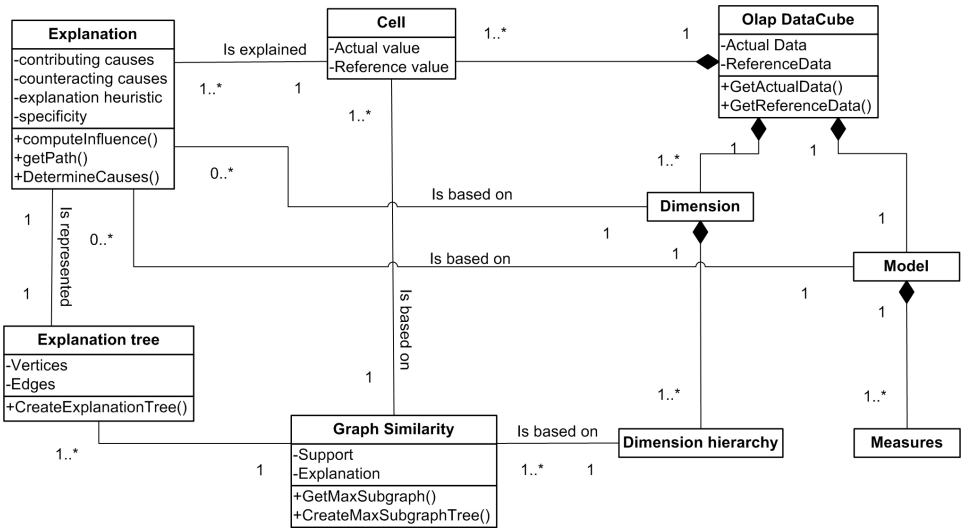


Figure 5.2: The dimensions and dimension levels used in the case study

The OLAP datacube is created based on the model discussed in the previous chapters. The dimensions are predetermined and use to built up

the OLAP datacube. The OLAP datacube exists of multiple cells, where every cell can be explained based on a reference object. The explanation can be created based on contributing or counteracting causes. With a selected heuristic the significant causes are selected. The significant causes are represented as an explanation tree. For cells from a similar dimension level different explanation trees can be explained. The different explanation trees are compared with each other and a (weighted) similarity tree is generated.

5.3 The software

In this section we explain the software based on screenshots and discuss the main code behind the software. In Figure 5.3 the startpage of the software is shown.

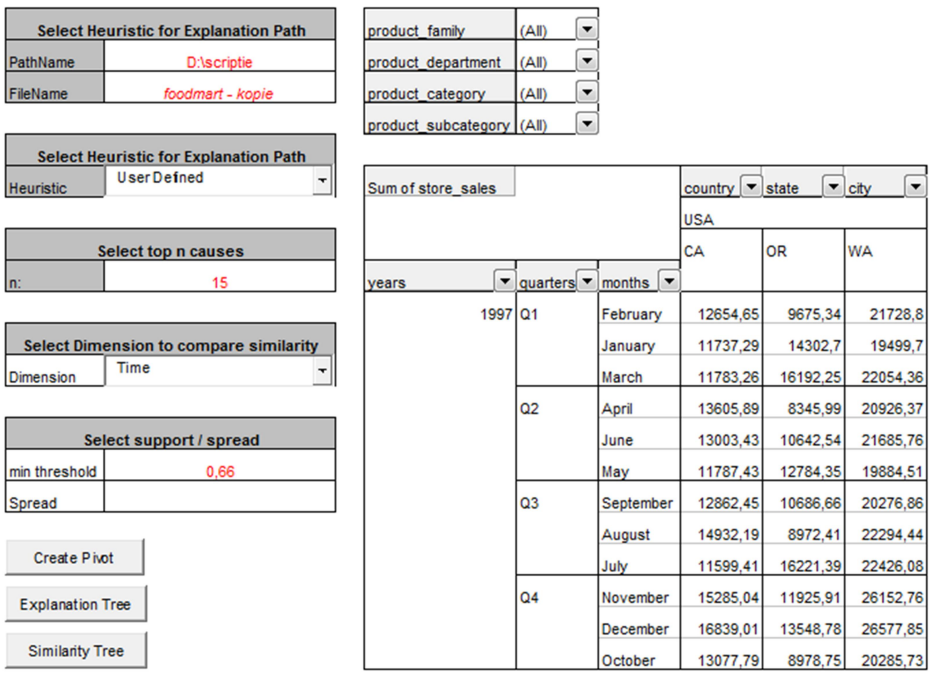


Figure 5.3: The screenshot of the main GUI

In this screenshot the data is already loaded from the database stored in the folder *scriptie*, and represented in the pivot table. The data for the pivottable is retrieved using the SQL statement shown in Figure 5.4. Because this is a prototype we simplified the SQL problem with the assumption the prototype can only create a pivot table for the foodmarket database. The query selects the time by day (time), store (location) and product (product). Using this data and the fact table a pivot table is created.

```
Private Function CreatePivotQuery()

'Create Array
CreatePivotQuery = "SELECT *" & Chr(13) & "" & Chr(10) & _
             "FROM time_by_day, Product, sales_fact_1997, store" & Chr(13) & "" & Chr(10) & _
             "WHERE time_by_day.time_id = sales_fact_1997.time_id AND  sales_fact_1997.store_id = store.store_id " & _
             "AND sales_fact_1997.product_id = Product.product_id"

End Function
```

Figure 5.4: The SQL string used to retrieve the data from the database

This prototype is limited to the measure sales. The user can use any of the navigational operators discussed in Section 2.2.3 to navigate through the pivot table to find any arbitrary cell.

The user can select which heuristic is used to generate an explanation tree. If the user selects the one dimensional heuristic he fills in the first letter of the dimension drilled down over. For the user defined path the user enter the first letter of the dimension the path follows. In Figure 5.3 the user has selected the path Location, Product, Location and Time. The user defined path in matrix notation is

$$\left(\begin{array}{c|ccc} \text{Level} & L & T & P \\ 9 & -1 & 0 & 0 \\ 8 & 0 & 0 & -1 \\ 7 & -1 & 0 & 0 \\ 6 & 0 & -1 & 0 \end{array}\right)$$

In the select top n causes the user selects the n biggest causes for the aggregated table. Once this is all entered the user selects a cell in the pivot table and hits the Explanation tree button. Using the SQL statement shown in Figure 5.5 the data is retrieved from the database. The the Actual value can be calculated like

$$= sum(if(months =' Q1', store_sales))as'actual'.$$

The most important part in the SQL statement is the WHERE part of the SQL statement. With the PivotPositionString the position of the cell in the pivot table is determined and added as a restriction to which the SQL statement must satisfy. The data in the SQL statement can be grouped over the location dimension and product dimension.

In the 'group by' part, the data is summed over the dimensional levels. Following the path defined earlier the SQL statement is repeated at every path. After executing the SQL statements several times the Aggregated table is generated.

Then the data is ordered descending for the influence value, and the top n causes are selected by the user. The program checks if the parent of each of these causes exists, and if not this causes is added. The causes are shown

```
Actual = GetActualString()
Reference = GetReferenceString()

 'The 1st raw SQL-statement to be executed.

stSQL1 = "SELECT " & DimLocation & ", " & DimProduct & ", " & Actual & ", " & Reference & "" & _
         "FROM store, product, time_by_day, sales_fact_1997 " & _
         "WHERE sales_fact_1997.store_id = store.store_id AND sales_fact_1997.product_id = product.product_id " & _
         "AND sales_fact_1997.time_id = time_by_day.time_id " & PivotPositionString & _
         "GROUP BY " & DimLocation & ", " & DimProduct & ";"

With cnt
.Open (stConn) 'Open the connection.
```

Figure 5.5: The SQL string used to create the aggregated table

ordered in dimensions. This is also how the explanation tree is represented in the prototype.

If the user wants to find the similarity pattern he selects the dimension over which the user wants to find the similarity pattern. The similarity pattern finds the siblings of the selected cell over the chosen dimension and creates a explanation tree for each cell.

The minimum support is entered by the user, and the maximum spread for the weights. If the spread is left empty the similarity is pure structural, otherwise it is a weighted similarity tree.

At the moment the explanation trees and similarity trees are respresented in a table and not as a tree. With some extra programming it is possible the explanation/similarity tree is shown correctly.

Chapter 6

Case Study

6.1 Introduction

The case study is not a 'real case study', but a study on an artificial, but representative, database. For the case study we use the foodmart 2000 dataset. The dataset's original schema is a galaxy schema, this is a dataset that has multiple fact tables. In our case study we simplified the dataset and we focus on the fact table sale_fact_1997 , which consist of 86,837 data records. The schema was originally a snowflake schema, but in order to clarify the example the schema is changed into a star schema. The modified schema is shown in Figure 6.1. In the modified dataset the costs and sales of the foodmart are shown over three dimensions.

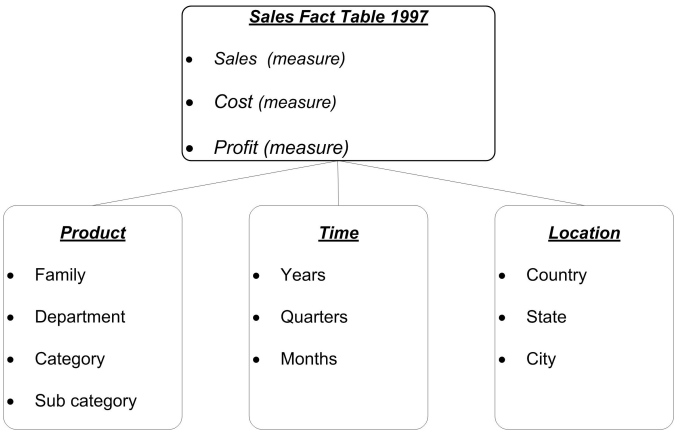


Figure 6.1: The star schema of the foodmart's sales fact table

In the case study we use the algorithms from Chapter 3 and 4 to create explanation trees and similarity tree.

6.2 Generate Explanation Trees

We use the same cell to generate an explantion with the different heuristics. We explore the cell '1997 × All Products × USA' in the datacube. We drill down over the time dimension; R_T^{-1}(1997× All Products × USA) = (1997.Quarter × All Products × USA), and compare the quarterly sales with the average quarterly sales in dollars.

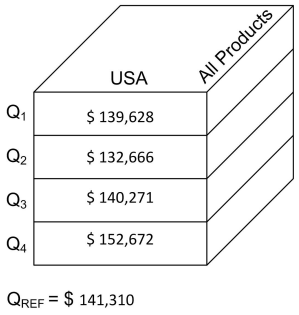


Figure 6.2: The starting cell is drilled down over the time dimension

When comparing the 4 quarters in Figure 6.2 we notice the sales in the fourth quarter was the only quarter that performed better than average. The average value can be seen as a chain of reference objects because the average is calculated for each cell. We want to generate an explanation why this occurred. More formally the symptom that has to be explained is;

⟨ Sales(1997.Q4, All, USA), 'high', Avg(Sales(1997.Quarter , All, USA)) ⟩.

We use the three heuristics discussed in Chapter 3 to generate an explanation for this symptom. For each heuristic we show the aggregated table, the top 10 contributing causes and the explanation tree. We finally give a business interpertation for the explanation tree.

6.2.1 Dimension Hierarchy heuristic

The first heuristic we used is the dimension hierarchy heuristic. With this heuristic we generate an explanation over the location dimension. The analysis path used is [1,4,2] → [1,4,1] → [1,4,0]. We use the heuristics introduced in Chapter 3.4 to generate the explanation tree.

The aggregated table

In the first step of the heuristic the aggregated table is generated. This is shown in Figure 6.3

Location			Product				Measure		
Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence
USA	All	All	All	All	All	All	152672	141310	11362
USA	CA	All	All	All	All	All	45202	39792	5410
USA	OR	All	All	All	All	All	34453	35569	-1116
USA	WA	All	All	All	All	All	73016	65948	7068
USA	CA	Beverly Hills	All	All	All	All	14840	11438	3403
USA	CA	Los Angeles	All	All	All	All	15255	13636	1619
USA	CA	San Diego	All	All	All	All	13913	13608	306
USA	CA	San Francisco	All	All	All	All	1192	1110	82
USA	OR	Portland	All	All	All	All	14826	13765	1061
USA	OR	Salem	All	All	All	All	19628	21805	-2177
USA	WA	Bellingham	All	All	All	All	1522	1185	337
USA	WA	Bremerton	All	All	All	All	14580	13224	1356
USA	WA	Seattle	All	All	All	All	14089	13161	928
USA	WA	Spokane	All	All	All	All	13563	12409	1154
USA	WA	Tacoma	All	All	All	All	21254	18711	2543
USA	WA	Walla Walla	All	All	All	All	1423	1176	247
USA	WA	Yakima	All	All	All	All	6585	6082	503

Figure 6.3: The aggregated table based on the Location dimension

In the aggregated table we see all the contributing and counteracting causes of the symptom. The order of the causes in the aggregated are determined by the drill-down path.

Selection significant causes

In the next step the influence of the causes are ordered in descending order and $n = 10$. The 10 contributing causes are shown in Figure 6.4. In order to create the explanation tree all causes should have a symptom. Because the cell 'USA.Or.Portland' does not have a parent in the top 10 contributing causes its parent is added. The parent is the counteracting cause 'USA.Or'.

	Location			Product				Measure		
	Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence
Symptom	USA	All	All	All	All	All	All	152672	141310	11362
1	USA	WA	All	All	All	All	All	73016	65948	7068
2	USA	CA	All	All	All	All	All	45202	39792	5410
3	USA	CA	Beverly Hills	All	All	All	All	14840	11438	3403
4	USA	WA	Tacoma	All	All	All	All	21254	18711	2543
5	USA	CA	Los Angeles	All	All	All	All	15255	13636	1619
6	USA	WA	Bremerton	All	All	All	All	14580	13224	1356
7	USA	WA	Spokane	All	All	All	All	13563	12409	1154
8	USA	OR	Portland	All	All	All	All	14826	13765	1061
9	USA	WA	Seattle	All	All	All	All	14089	13161	928
10	USA	WA	Yakima	All	All	All	All	6585	6082	503
counteracting cause	USA	OR	All	All	All	All	All	34453	35569	-1116

Figure 6.4: The top 10 contributing causes from the aggregated table are selected

Generate Explanation Tree

With the top 10 contributing causes and the counteracting cause we generate the explanation tree. To make clear the the cause 'USA.Or' is not a top 10 contributing cause, it is coloured grey. And because it is a counteracting cause, the edge between the counteracting cause and its parent is a dotted line. The explanation tree is shown in Figure 6.5

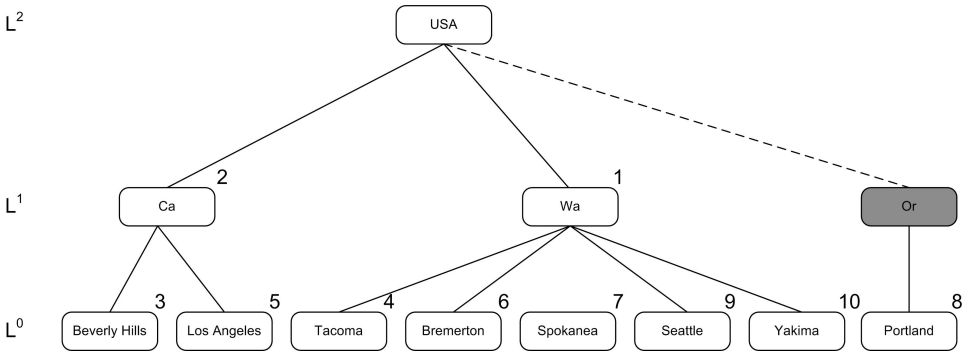


Figure 6.5: The explanation tree based on the location Dimension with n=10

Conclusion

With the generated explanation tree the analyst has a clear explanation based over the location dimension. The fourth quarter had a higher sales compared to average mainly due to the state 'WA'. From the 7 cities in the state 'WA', there were 5 cities that performed better than the quarterly average. In the state 'Ca' the city ''Beverly Hills' performed good. It is noteworthy to mention the city 'Portland'. It performed better than average but its parent performed worse than the average. In a top-down approach this cause would not have been shown.

With the generated explanation tree in Figure 6.5 the analyst has no information over the products that performed better, or worse than expected. He also does not know in which month the sales was the highest. The heuristic is powerful for creating an explanation based on one dimension but is too limited for general use.

6.2.2 User Defined heuristic

In the next explanation tree we also take the product dimension into account. We create an user defined path to generate the explanation tree. The user defined path in matrix notation is given by

$$\left(\begin{array}{c|ccc} \text{Level} & L & T & P \\ 9 & 0 & 0 & -1 \\ 8 & -1 & 0 & 0 \\ 7 & 0 & 0 & -1 \\ 6 & -1 & 0 & 0 \end{array}\right) \quad (6.1)$$

The first column exists of the drill down level and the three rows next to it shows the dimension involved (Location, Time and Product). So in the first row the product dimension is drilled down one level, $(R_P^{-1}(USA, 1997.Q4, All)) = (USA, 1997.Q4, product\_family)$.

The aggregated table

Location			Product				Measure		
Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence
USA	All	All	All	All	All	All	152672	141310	11362
USA	CA	All	All	All	All	All	45202	39792	5410
USA	OR	All	All	All	All	All	34453	35569	-1116
USA	WA	All	All	All	All	All	73016	65948	7068
USA	CA	All	Drink	All	All	All	4060	3551	509
USA	CA	All	Food	All	All	All	32745	28798	3946
USA	CA	All	Non-Consumabl	All	All	All	8397	7443	954
USA	OR	All	Drink	All	All	All	3004	3034	-31
USA	OR	All	Food	All	All	All	24820	25641	-822
USA	OR	All	Non-Consumabl	All	All	All	6630	6894	-263
USA	WA	All	Drink	All	All	All	6278	5624	654
USA	WA	All	Food	All	All	All	52967	47819	5147
USA	WA	All	Non-Consumabl	All	All	All	13772	12505	1267
USA	CA	Beverly Hills	Drink	All	All	All	1302	985	317
USA	CA	Beverly Hills	Food	All	All	All	10673	8356	2317
USA	CA	Beverly Hills	Non-Consumabl	All	All	All	2865	2096	769
USA	CA	Los Angeles	Drink	All	All	All	1453	1206	247
USA	CA	Los Angeles	Food	All	All	All	10896	9797	1100
USA	CA	Los Angeles	Non-Consumabl	All	All	All	2906	2633	272
USA	CA	San Diego	Drink	All	All	All	1203	1266	-63
USA	CA	San Diego	Food	All	All	All	10305	9826	479
USA	CA	San Diego	Non-Consumabl	All	All	All	2405	2515	-110
USA	CA	San Francisco	Drink	All	All	All	101	93	8
USA	CA	San Francisco	Food	All	All	All	870	819	51
USA	CA	San Francisco	Non-Consumabl	All	All	All	221	198	23
USA	OR	Portland	Drink	All	All	All	1274	1186	89

Figure 6.6: The aggregated table based on the Location dimension

With the drill down path an aggregated table can be generated using Algorithm 3.3. The first 25 causes of the aggregated table are shown in Figure 6.6. The complete aggregated table exists of more than 300 causes and can be seen in Appendix A.

Selection significant causes

The causes are ordered in descending order and the top 10 contributing causes are selected, these are shown in Figure 6.7. The explanation tree doesn't need any extra causes to be added.

	Location			Product				Measure		
	Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence
Symptom	USA	All	All	All	All	All	All	152672	141310	11362
1	USA	WA	All	Food	All	All	All	52967	47819	8272
2	USA	WA	All	Non-Consumabl	Household	All	All	7642	6964	5147
3	USA	OR	All	Food	All	All	All	24820	25641	3946
4	USA	CA	All	Food	All	All	All	32745	28798	1958
5	USA	OR	All	Non-Consumabl	All	All	All	6630	6894	1267
6	USA	All	All	Food	All	All	All	110531	102259	1133
7	USA	WA	All	Non-Consumabl	All	All	All	13772	12505	954
8	USA	CA	San Diego	Non-Consumabl	Household	All	All	1475	1389	789
9	USA	WA	Tacoma	Food	Produce	All	All	2974	2706	787
10	USA	CA	Los Angeles	Non-Consumabl	Household	All	All	1663	1523	682

Figure 6.7: The top 10 contributing causes

The Explanation Tree

The explanation tree is generated using Algorithm 3.3. The generated explanation tree is shown in Figure 6.8.

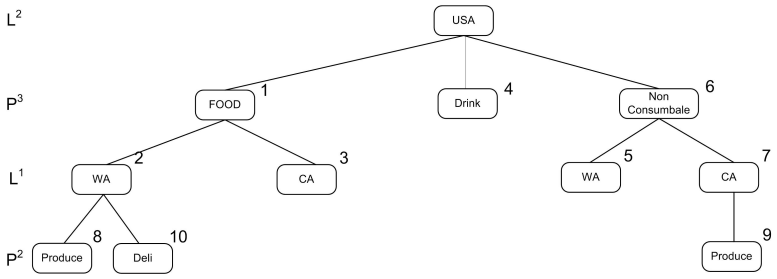


Figure 6.8: The explanation tree based on the user defined path

Conclusion

The higher sales can be explained due to the higher sales in the product family 'Food'. The top 3 causes can be found in the 'Food' branch. The product family Drink did also perform better than average but did not perform exceptionally good in one of the states. We again can conclude with the explanation tree that the states 'Ca' and 'Wa' performed better than average which corresponds with the findings of the explanation tree generated with the one dimension heuristic.

The advantage of this heuristic is that the analyst can manually select the drill-down path. This allows the analyst to see the causes of the dimensions

that are most important for the analyst in the top of the explanation tree. However, this advantage can also be a disadvantage when a dimension is selected in the top that does not contain much information. To ensure the most interesting drill-down path is selected, the analyst can use the specificity heuristic.

6.2.3 Specificity heuristic

In this last subsection we create an explanation tree using the specificity heuristic, see Chapter 3.4.1. This algorithm is different compared to the greedy algorithms because it is based on a top-down algorithm. The algorithm allows different dimensions te be shown in the same level of the explanation tree.

For each new cause the heuristic calculates the specificity values for the different dimensions and selects the dimension with the highest specificity value. The fraction T^+ is 0.9. For the symptom (1997.Q4, All, USA) the specificity is calculated for the location ($S_L = \frac{3}{2}$) and product dimension ($S_P = \frac{3}{3}$).

The Location has the highest specificty value. Therefore we drill down the location dimension and select the causes in the parsimonious set. For each of these causes the specifcity is calculated and we drill down over the dimension with the highest specificity value.

With this approach we cannot specificy the number of causes to be shown or the maximum level of depth. Because we are interested in the most important causes we only drill down two levels.

Generate Explanation tree

With the top 10 contributing causes the explanation tree can be generated, there is no need to add any other causes. The explanation tree is shown in Figure 6.9.

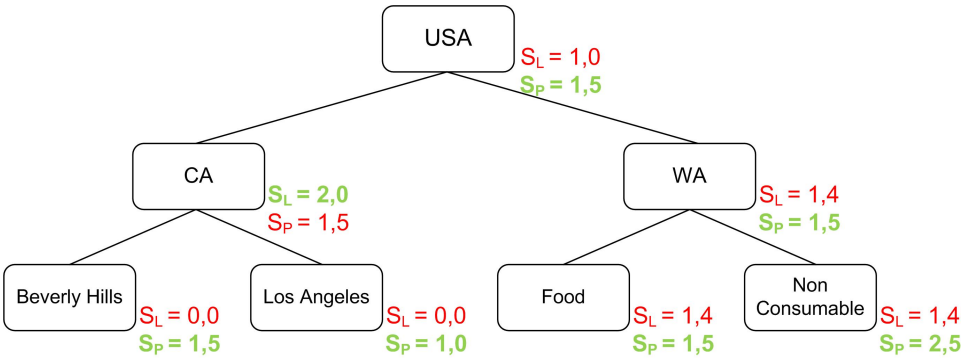


Figure 6.9: The explanation tree based on the specificity heuristic

Aggregated Table

For the 6 causes selected with the specificity heuristic an table with the influence of each of these causes can be generated, This table is shown in Figure 6.10.

	Location			Product				Measure			
	Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence	Fraction
Symptom	USA	All	All	All	All	All	All	152672	141310	11362	-
	USA	CA	All	All	All	All	All	45202	39792	5410	0,48
	USA	WA	All	All	All	All	All	73016	65948	7068	0,62
	USA	WA	All	Food	All	All	All	52967	47819	5148	0,73
	USA	WA	All	Non-Consumable	All	All	All	13772	12505	1267	0,18
	USA	CA	Beverly Hills	All	All	All	All	14840	11438	3402	0,63
	USA	CA	Los Angeles	All	All	All	All	15255	13636	1619	0,30

Figure 6.10: The explanation tree based on the specificity heuristic

Conclusion

With the explanation tree in Figure 6.10 it becomes clear that the higher sales in the USA are mainly caused by the states 'Wa' and 'Ca'. But the way the sales are explained per state is different. In the state 'Ca' two cities performed very well, this explains the higher sales in the state 'Ca'. But in the state 'Wa' the higher sales are better explained by looking at the product families. Here the higher sales was due to the higher sales in Food and Non-Consumable. Looking at the specificity values of the causes (1997.Q4, Food, USA.Wa) and (1997.Q4, Non-Consumable, USA.Wa) we can see that these are more specific explained over the product dimensions instead of the location dimension. The analyst can therefore conclude the cities in the state 'Wa' performed in general better than average and the state 'Ca' performed better due to 2 specific cities.

A typical property of explanation trees generated with the specificity heuristic is that nodes on the same level in the tree can come from different dimensions.

6.3 Generate Similarity Pattern

In this section we take a closer look at the explanation trees created with the user defined path and wants to know if there is a structural explanation why the first three quarters performed worse than average. More generally speaking;s were there in the first three quarters structural causes that performed worse than average? We create the explanation trees for the Q1, Q2 and Q3 with the a user defined path. The path used is shown in Equation 6.2

$$\left(\begin{array}{c|ccc} \text{Level} & L & T & P \\ 7 & -1 & 0 & 0 \\ 6 & 0 & 0 & -1 \\ 5 & -1 & 0 & 0 \\ 4 & 0 & 0 & -1 \end{array}\right) \quad (6.2)$$

Explanation trees

Because we want to know why the quarters performed worse than expected we have to take a look at the causes with the lowest influence values. The top 10 selection are similar as shown in the previous section. Therefore we do not show the top 10 tables here, they can be found in Appendix A. The generated explanation trees are shown in Figure 6.11, Figure 6.12 and Figure 6.13.

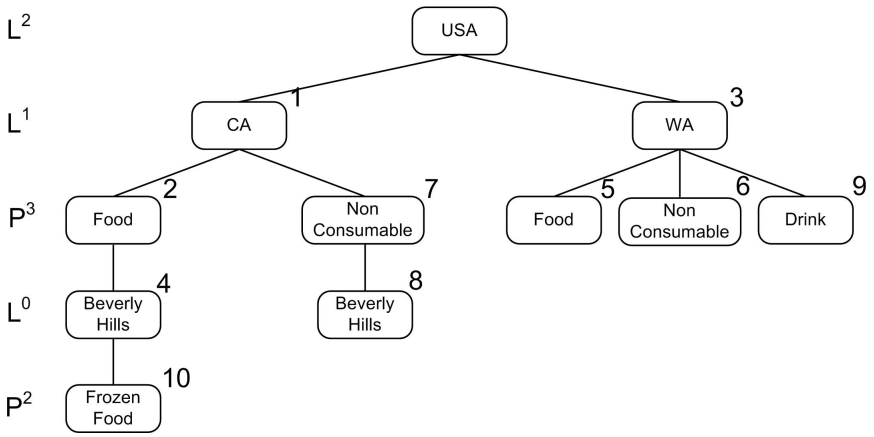


Figure 6.11: The explanation tree for Q1

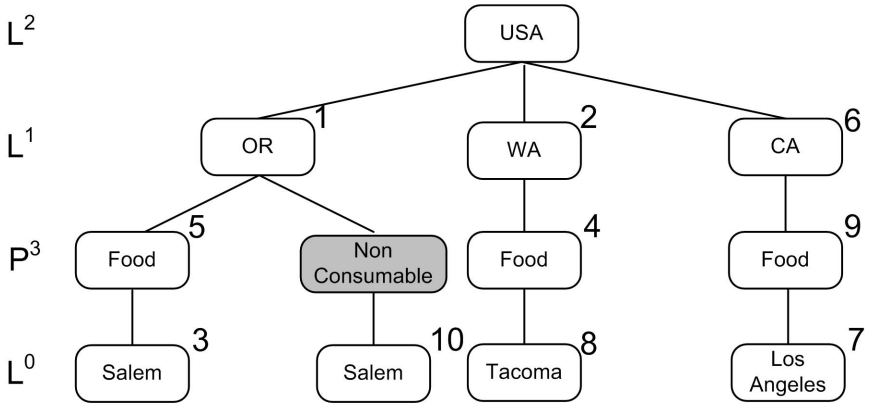


Figure 6.12: The explanation tree for Q2

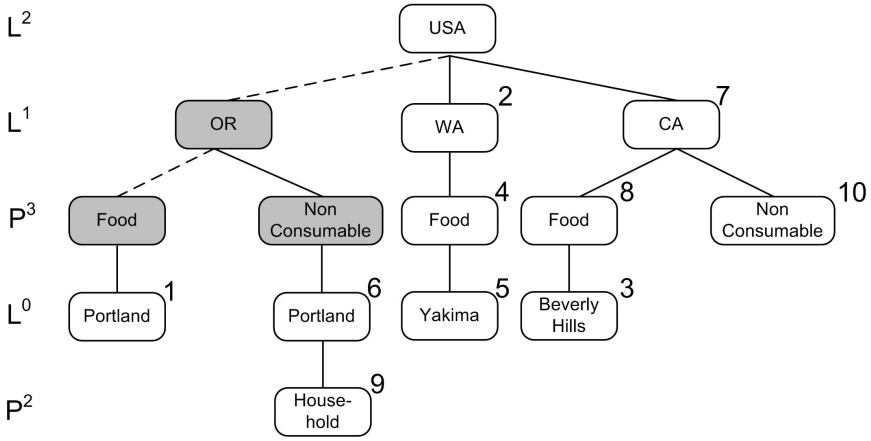


Figure 6.13: The explanation tree for Q3

Interestingly we see in the third quarter that the city 'Portland' was the main reason why 'Q3' performed worse thant average. In a normal top-down approach this cause would never been shown in the explanation tree because it would never drill down in the location dimension.

Similarity tree

When looking at the three different explanation trees there are little similarities detected. But taking a closer look a similarity tree can be generated. The similarity tree with support = 1 is shown in 6.14.

It is possible to relax the similarity support level. Relaxing the support to 2/3 would add the causes (1997.Q4, Non Consumable, USA.CA) and (1997.Q4, Food, USA.CA.Beverly Hills) to the explanation tree.

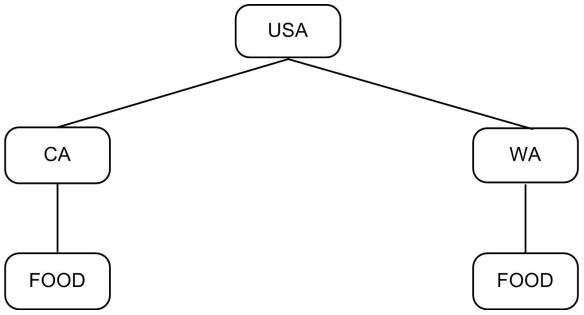


Figure 6.14: The similarity tree

Conclusion

The first three quarters performed worse because the states 'CA' and 'WA', and than especially the product family 'Food', performed worse than average. The business analist should find underlying explanations, that can't be found in the dataset, why this occured.

This similarity tree confirms the idea that was previously created by the explanation tree of 'Q4'. In this explanation tree we saw the states CA and WA perform better than average.

6.3.1 Weighted Similarity tree

The similarity tree created in the previous section shows only information about structural similarity. It just shows the causes represented in all three explanation trees. But it does not contain any information related to the magnitude of the causes.

To detect if the weighted structure is similar between the three quarters the fractions of the causes are compared with each other. If we compare the fractions we can create a similarity tree based on the weighted structure. An advantage of this approach that it considers all causes, including counteracting causes.

The first three quarters are again used to create a similarity tree. We calculate the average influence for each cause in the first two levels. And than calculate the spread with the standard deviation. This results in the table shown in in Figure 6.15. The support is also based on how many times the cause occured in the individual explanation trees.

If the standard deviation is lower than 0.25 and the support higher than 0.6 the cause will be added into the similarity tree. In the edge the average weight is shown for the particular cause has. The similarity tree generated with this approach is shown in 6.15.

The weighted similarity tree is shown in Figure 6.16. The states 'Wa' and 'Ca' were also structurally similar over the three quarters. The weights

Location			Product				Measure									
Country	State	City	Family	Department	Category	Sub Category	INFQ1	INFQ2	INFQ3	INFAVG	FRACQ1	FRACQ2	FRACQ3	FRACAVG	StDev FRAC	Support
USA	All	All	All	All	All	All	-1681	-8643	-1038	-3787	1	1	1	1,00	0,00	1
USA	CA	All	All	All	All	All	-3617	-1395	-398	-1803	0,58	0,16	0,3	0,35	0,21	1
USA	OR	All	All	All	All	All	4601	-3796	311	372	-1	0,44	-1	-0,52	0,83	0,33
USA	WA	All	All	All	All	All	-2665	-3452	-951	-2356	0,42	0,4	0,7	0,51	0,17	1
USA	CA	All	Drink	All	All	All	-241	-221	-47	-170	0,07	0,16	0,12	0,12	0,05	0
USA	CA	All	Food	All	All	All	-2753	-918	-274	-1315	0,76	0,66	0,69	0,70	0,05	1
USA	CA	All	Non-Consumable	All	All	All	-622	-256	-76	-318	0,17	0,18	0,19	0,18	0,01	0,67
USA	OR	All	Drink	All	All	All	135	-73	-32	10	-0,03	0,02	0,11	0,03	0,07	0
USA	OR	All	Food	All	All	All	3095	-2884	610	274	-0,67	0,76	-1	-0,30	0,94	0
USA	OR	All	Non-Consumable	All	All	All	1371	-840	-268	88	-0,3	0,22	0,89	0,27	0,60	0,33
USA	WA	All	Drink	All	All	All	-518	0	-136	-218	0,19	0	0,14	0,11	0,10	1
USA	WA	All	Food	All	All	All	-1339	-3021	-787	-1716	0,5	0,88	0,83	0,74	0,21	0,33
USA	WA	All	Non-Consumable	All	All	All	-809	-430	-28	-422	0,3	0,12	0,03	0,15	0,14	0,33

Figure 6.15: The influence, fractions, averages and standard deviation of the causes.

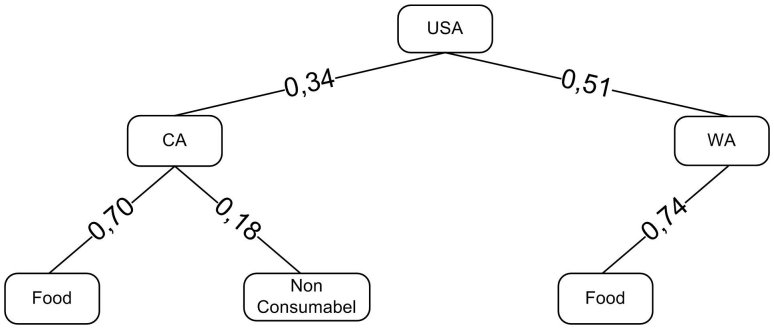


Figure 6.16: The weighted similarity tree.

confirm that the product family 'Food' was the most important reason to perform worse than average in both states. When we also look at the table with the fraction and influence values, we also see that the product family 'Drink' only plaid a minor role in explaining the lower profits in the first three quarters.

Conclusion

The advantage of the weighted similarity tree is that it does not only show the causes that were similar but also the average weight of these causes. In Figure 6.16 it became clear that 'Wa.Food' and 'Ca.Food' are one of the most important causes why the sales were below average in the first three quarters. It also shows the Product Family 'Drink' was structurally an cause.

Chapter 7

Conclusions and Future work

7.1 Conclusion

In the previous chapters we created three heuristic to (semi) automatically generate an explanation tree. The three heuristics used were the dimension heuristic, the user defined heuristic and the specificity heuristic. The heuristics generate the explanation tree in complete different ways. But they all generate an explanation that can be reviewed in a quick and convenient way by the analyst. Showing the top n significant causes in the explanation tree allows the analyst to quickly review the explanation tree and ensure there is no information overload.

The first two heuristics are based on a greedy algorithm. With this greedy algorithm the analyst knows for sure all important causes are presented in the explanation tree. This is useful because contributing causes can be cancelled out by counteracting causes at lower dimension levels.

The last heuristic, the specificity heuristic, is based on a top-down approach. This approach does not guarantee all significant causes are shown, but it gives a clear explanation for each cause individually. This is done automatically. This is why it is a powerful explanation heuristic. For symptoms with a large number of dimensions, and dimension levels, this heuristic is also the most interesting, because this is too complex for an analyst to create the most interesting analysis path.

When different cells in a dimension level are compared with each other we first create an explanation tree for each cell. The structural similarity pattern found is useful for an analyst as ignores the large incidental causes and only look at the structural causes. With the detection of a similarity pattern a symptom can not be explained with an incidental cause anymore. In some cases it can be convenient to allow some relaxation of the definition similarity and use a support level. This can be interesting if a large number of trees are compared with each other.

If a more detailed eplanation is needed the analyst can use the weighted similarity. It shows the structural similarity also and it shows the general influence a child has on its parent. If this varies to much this cause will not be shown in the weighted similarity tree.

7.2 Future work

For the explanation generation we discussed the greedy explanation tree and one implementation of the top-down approach. Other explantion generation algorithm could be considered and different algorithms can be compared.

The similarity pattern created in this thesis used the assumption the drill down path of the compared explanation trees were the same. But with the specifcity heuristic the explanation trees can be different. It is interesting to find out if it possible to detect a maximum sub graph for these explanation trees although they have different drill down paths.

It might be possible to use the similarity tree to predict the structure of a new cell in the dataset. This can be useful, for example, to deterimine the purchase of products in a new store. This can than be compared with other predictive models.

Bibliography

[BE08] Wagner B. and Monk E. *Enterprise Resource Planning.* Course Technology, 2008.

[Bun09] H Bunke. Graph matching: Theoretical foundations, algorithms, and applications. */www.ai.rug.nl*, 2009.

[Car09] E.A.M. Caron. Explanation of exceptional values in multi-dimensional business databases. Haveka, 2009.

[CCC93] E.F. Codd, S.B. Codd, and Salley C.T. Providing olap (on-line analytical processing) to user-analysts: An it mandate. *fpm*, 1993.

[CD04] E.A.M. Caron and H.A.M. Daniels. Automated business diagnosis in the olap context. *ERIM Report Series*, 2004.

[CD08] E.A.M. Caron and H.A.M. Daniels. Explanation of exceptional values in multi-dimensional business databases. *European Journal of Operational Research*, 188(3):884–897, August 2008.

[CWH96] A. Cross, R. Wilson, and E. Hancock. Genetic search for structural matching. *Springer Verlag*, (31), 1996.

[FD01] A. J. Feelders and H. A. M. Daniels. A general model for automated business diagnosis. *European Journal of Operational Research*, 130(3):623–637, 2001.

[FLD94] J. Feng, M. Laumy, and M Dhome. Inexact matching using neural networks. *Comparative Studies and Hybrid Systems*, 1994.

[Hes84] Hesslow. Explaining differences and weighting causes. *Theoria*, 1984.

[HK00] J. Han and M. Kamber. Data mining: Concepts and techniques (the morgan kaufmann series in data management systems). Morgan Kaufmann, September 2000.

[Hum97] P.W. Humphreys. The chances of explanation. *Princeton University Press*, 1997.

[Inm96] W.H. Inmon. Building the data warehouse. 1996.

[Kam09] M.J. Kamfonas. An effective aggregation methodology. *kamfonas.com*, 2009.

[KH05] A. Karakasidis and I. Hellas. Etl queues for active data warehousing. In *In Proc. of IQIS*, 2005.

[LC06] J. Lewis and J. Chase. *Java software structures*. Addison Wesley, 2006.

[Lev72] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 1972.

[PAH08] Papadimitriou P., Dasdan A., and Garcia-Molina H. Web graph similarity for anomaly detection. *Stanford*, 2008.

[Pen02] N. Pendse. The origins of todayï¿½s olap products. *DSSResources.COM*, 2002.

[PJ01] T.B. Pedersen and C.S. Jensen. Multidimensional database technology. *Computer*, 34(12):40–46, 2001.

[SAM98] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of olap data cubes. In *EDBT '98: Proceedings of the 6th International Conference on Extending Database Technology*, pages 168–182. Springer-Verlag, 1998.

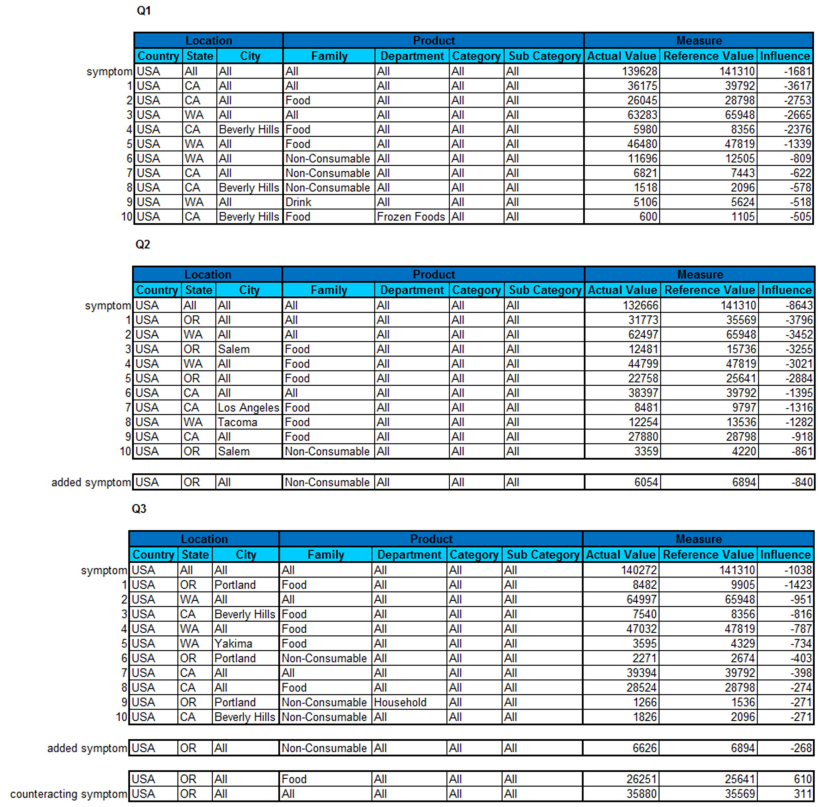
[YBV05] L. Yang, M. Ball, and Bhavsar V.C. Weighted partonomy-taxonomy trees with local similarity measures for semantic buyer-seller match-making. *Business Agents and the Semantic Web (BASeWEB) Workshop*, 2005.

Appendix A

Aggregated Tables

Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence
USA	All	All	All	All	All	All	152672	141310	11362
USA	All	All	Drink	All	All	All	13342	12209	1133
USA	All	All	Food	All	All	All	110531	102259	8272
USA	All	All	Non-Consumable	All	All	All	28799	26842	1958
USA	CA	All	Drink	All	All	All	4060	3551	509
USA	CA	All	Food	All	All	All	32745	28798	3946
USA	CA	All	Non-Consumable	All	All	All	8397	7443	954
USA	OR	All	Drink	All	All	All	3004	3034	-31
USA	OR	All	Food	All	All	All	24820	25641	-822
USA	OR	All	Non-Consumable	All	All	All	6630	6894	-263
USA	WA	All	Drink	All	All	All	6278	5624	654
USA	WA	All	Food	All	All	All	52967	47819	5147
USA	WA	All	Non-Consumable	All	All	All	13772	12505	1267
USA	CA	Beverly Hills	Drink	All	All	All	1302	985	317
USA	CA	Beverly Hills	Food	All	All	All	10673	8356	2317
USA	CA	Beverly Hills	Non-Consumable	All	All	All	2865	2096	769
USA	CA	Los Angeles	Drink	All	All	All	1453	1206	247
USA	CA	Los Angeles	Food	All	All	All	10896	9797	1100
USA	CA	Los Angeles	Non-Consumable	All	All	All	2906	2633	272
USA	CA	San Diego	Drink	All	All	All	1203	1266	-63
USA	CA	San Diego	Food	All	All	All	10305	9826	479
USA	CA	San Diego	Non-Consumable	All	All	All	2405	2515	-110
USA	CA	San Francisco	Drink	All	All	All	101	93	8
USA	CA	San Francisco	Food	All	All	All	870	819	51
USA	CA	San Francisco	Non-Consumable	All	All	All	221	198	23
USA	OR	Portland	Drink	All	All	All	1274	1186	89
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
USA	WA	Walla Walla	Non-Consumable	Checkout	All	All	0	4	-4
USA	WA	Walla Walla	Non-Consumable	Health and Hygiene	All	All	98	62	36
USA	WA	Walla Walla	Non-Consumable	Household	All	All	152	125	27
USA	WA	Walla Walla	Non-Consumable	Periodicals	All	All	8	12	-4
USA	WA	Yakima	Drink	Alcoholic Beverages	All	All	170	178	-7
USA	WA	Yakima	Drink	Beverages	All	All	418	339	79
USA	WA	Yakima	Drink	Dairy	All	All	78	71	7
USA	WA	Yakima	Food	Baked Goods	All	All	133	142	-9
USA	WA	Yakima	Food	Baking Goods	All	All	410	406	4
USA	WA	Yakima	Food	Breakfast Foods	All	All	95	73	22
USA	WA	Yakima	Food	Canned Foods	All	All	502	439	63
USA	WA	Yakima	Food	Canned Products	All	All	50	46	4
USA	WA	Yakima	Food	Dairy	All	All	334	342	-9
USA	WA	Yakima	Food	Deli	All	All	328	258	70
USA	WA	Yakima	Food	Eggs	All	All	95	98	-3
USA	WA	Yakima	Food	Frozen Foods	All	All	686	599	88
USA	WA	Yakima	Food	Meat	All	All	55	46	9
USA	WA	Yakima	Food	Produce	All	All	957	833	125
USA	WA	Yakima	Food	Seafood	All	All	18	27	-9
USA	WA	Yakima	Food	Snack Foods	All	All	738	727	11
USA	WA	Yakima	Food	Snacks	All	All	170	160	10
USA	WA	Yakima	Food	Starchy Foods	All	All	139	132	8
USA	WA	Yakima	Non-Consumable	Carousel	All	All	19	15	4
USA	WA	Yakima	Non-Consumable	Checkout	All	All	15	25	-10
USA	WA	Yakima	Non-Consumable	Health and Hygiene	All	All	379	393	-14
USA	WA	Yakima	Non-Consumable	Household	All	All	704	640	64
USA	WA	Yakima	Non-Consumable	Periodicals	All	All	90	94	-3

Figure A.1: A more detailed view of the defined aggregated table



Q1

	Location			Product				Measure		
	Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence
symptom	USA	All	All	All	All	All	All	139628	141310	-1681
1	USA	CA	All	All	All	All	All	36175	39792	-3617
2	USA	CA	All	Food	All	All	All	26045	28798	-2753
3	USA	WA	All	All	All	All	All	63283	65948	-2665
4	USA	CA	Beverly Hills	Food	All	All	All	5980	8356	-2376
5	USA	WA	All	Food	All	All	All	46480	47819	-1339
6	USA	WA	All	Non-Consumable	All	All	All	11696	12505	-809
7	USA	CA	All	Non-Consumable	All	All	All	6821	7443	-622
8	USA	CA	Beverly Hills	Non-Consumable	All	All	All	1518	2096	-578
9	USA	WA	All	Drink	All	All	All	5106	5624	-518
10	USA	CA	Beverly Hills	Food	Frozen Foods	All	All	600	1105	-505

Q2

	Location			Product				Measure		
	Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence
symptom	USA	All	All	All	All	All	All	132666	141310	-8643
1	USA	OR	All	All	All	All	All	31773	35569	-3796
2	USA	WA	All	All	All	All	All	62497	65948	-3452
3	USA	OR	Salem	Food	All	All	All	12481	15736	-3255
4	USA	WA	All	Food	All	All	All	44799	47819	-3021
5	USA	OR	All	Food	All	All	All	22758	25641	-2884
6	USA	CA	All	All	All	All	All	38397	39792	-1395
7	USA	CA	Los Angeles	Food	All	All	All	8481	9797	-1316
8	USA	WA	Tacoma	Food	All	All	All	12254	13536	-1282
9	USA	CA	All	Food	All	All	All	27880	28798	-918
10	USA	OR	Salem	Non-Consumable	All	All	All	3359	4220	-861
added symptom	USA	OR	All	Non-Consumable	All	All	All	6054	6894	-840

Q3

	Location			Product				Measure		
	Country	State	City	Family	Department	Category	Sub Category	Actual Value	Reference Value	Influence
symptom	USA	All	All	All	All	All	All	140272	141310	-1038
1	USA	OR	Portland	Food	All	All	All	8482	9905	-1423
2	USA	WA	All	All	All	All	All	64997	65948	-951
3	USA	CA	Beverly Hills	Food	All	All	All	7540	8356	-816
4	USA	WA	All	Food	All	All	All	47032	47819	-787
5	USA	WA	Yakima	Food	All	All	All	3595	4329	-734
6	USA	OR	Portland	Non-Consumable	All	All	All	2271	2674	-403
7	USA	CA	All	All	All	All	All	39394	39792	-398
8	USA	CA	All	Food	All	All	All	28524	28798	-274
9	USA	OR	Portland	Non-Consumable	Household	All	All	1266	1536	-271
10	USA	CA	Beverly Hills	Non-Consumable	All	All	All	1826	2096	-271
added symptom	USA	OR	All	Non-Consumable	All	All	All	6626	6894	-268
	USA	OR	All	Food	All	All	All	26251	25641	610
counteracting symptom	USA	OR	All	All	All	All	All	35880	35569	311

Figure A.2: The user defined aggregated tables for Q1, Q2 and Q3