FREE UNIVERSITY OF BOLZANO
FACULTY OF COMPUTER SCIENCE

Clinical Data Warehousing with QlikView: A Case Study

Author:
Michael DEJORI

Supervisor:
Prof. Johann GAMPER

Academic Year 2009/2010

Dedicated to my family

Acknowledgements

I would like to thank everyone who helped me realizing this work during the last three years of study.

- My parents, Martha and Walter Dejori, who supported me all the time and allowed me to pursue this study.
- My university tutor Prof. Johann Gamper who proposed this thesis to me and led me in the right direction.
- Christian Krüger from the DIS (Database and Information Systems) research group who was always open for questions and discussions when problems arised.
- A special thank to my girlfriend Nora, who was always there for me in the last years.

Last but not least I would like to thank the team from the Hospital of Merano, my fellow students with whom I had a great time and the University of Bolzano, which also gave me the possibility to spend a wonderful semester in the United States as an exchange student.

Thank you all! Michael

Abstract

We are deluged by data - scientific data, medical data, financial data and business data. Due to the computerization of business, financial and medical transactions, the amount of data increases rapidly. Therefore, we need additional tools that can analyze the data and discover trends from the relationship of the data. As a concrete example we consider QlikTech's Business Intelligence tool QlikView together with the OncoNet data warehouse, which collects data of cancer therapies at the Department of Hematology at the Hospital of Merano. In this thesis we examine the suitability of QlikView with the underlying OncoNet data warehouse. Moreover, we present a QlikView template application for a multidimensional data model that can be used by people without QlikView experience for an automated report generation. Non computer-experts, for example doctors, can interact at a highly abstracted level and generate their own statistics. The idea is that the user can see the star schema of the underlying data model where he can directly select the dimension attributes that he wants to have in his analysis. After specifying the aggregation function and the diagram type, QlikView macros written in VBScript will automatically create the report and the user does not have to cogitate about QlikView components and its properties.
Afterwards we design a model that supports OLAP (on-line analytical processing) or cube operations and implement this model in our QlikView template application so that the user can aggregate between the defined concept hierarchies in the created diagrams. This makes a flexible view changing by modifying the granularity of the data possible.

Contents

List of Figures

List of Tables

1 Introduction

Clinical databases accumulate large quantities of information about patients and their medical conditions and the amount of data increases every day. Thus it is unimaginable in Health Care to do without a data warehouse which collects the data from the operational databases and keeps it for statistical report generation and data analysis. Analytical tools can then retrieve important medical knowledge from the relationships of the data. These tools are often called Business Intelligence (BI) tools and they can be used to simplify the information reporting of the underlying operational data. BI tools are mostly used in business companies to control and survey the business performances and allow to managers, analysts and decision makers improving the business performances.

The Department of Hematology in the Hospital of Merano has legacy systems which daily collect new data from the cancer patients during cancer treatments and investigations. In a previous work, a data warehouse was constructed which loads the data from the operational databases and preserves the data in a data warehouse. This historical data can contain interesting information and is the base for statistical data analysis. Until now, the data can only be retrieved by standard SQL queries, directly applied to the data warehouse. Therefore, is needed an application which can visualize the underlying data to non-computer experts. The application should allow the user to navigate through the data, visualize the data in a more abstract way and generate statistics and reports he is interested in.

In this thesis I will examine the suitability of the Business Intelligence tool QlikView for a clinical data warehouse with the collaboration of the Hospital of Merano. Moreover, I will present a QlikView template application for a multidimensional data structure that supports cube operations. This template application can be adapted to any multidimensional model and allows to generate reports at a highly abstracted level of user interaction i.e. that QlikView analysis diagrams can be created without any experience in QlikView.

The contributions of this thesis are (i) investigating the suitability of QlikView, a fast evolving Business Intelligence tool with a clinical data warehouse, focusing on the generation of reports and statistics about cancer therapies. (ii) Creating a general QlikView template application for multidimensional data models. (iii) Designing a model for multidimensional data structures which supports OLAP or cube operations and (iv) implementing the model in the template application.

The organization of the work is the following. In the next chapter you will get some background knowledge about data warehouses and multidimensional data models in order to understand the contributions of the work. In addition, we will shortly take a look at the market position of QlikView in comparison to similar products. In chapter 3 we will focus on the basic design of the OncoNet data warehouse we are dealing with in this thesis. In chapter 4 we will discuss the construction of the general QlikView template application for multidimensional data models. Finally, in chapter 5 we will design a model that supports OLAP or cube operations and we will show the implementation of the model in our QlikView template application.

2 Background

2.1 Data Warehouse

Data Warehouses have been defined in many different ways. Crudely speaking, a data warehouse refers to a database that is maintained separately from an organization's operational databases. Operational systems are where the data is put in, and the data warehouse is where we get the data out[3]. According to William H. Inmon, a leading architect in the data warehouse construction, "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process"[2]. These four keywords describe the main features of a data warehouse:

- Subject oriented: a data warehouse represents data around a major subject, such as medical investigations, treatments and drug therapies in a clinical data warehouse. Data warehouses provide a clean and understandable view around a particular subject to decision makers by excluding irrelevant data.
- Integrated: a data warehouse integrates data from more heterogeneous sources. When the data is extracted from the operational sources, it is cleaned and loaded consistently.
- Time-variant: the data in a data warehouse depend explicitly upon time and every key structure in the data warehouse contains an element of time to provide historical information.
- Nonvolatile: due to the separation from the operational databases, data warehouses do not require transaction processing. The two operations which are performed on the data are initial loading and access of data.

The major reason for the separation of the data from the operational databases is to promote high performance in both systems[1]. Moreover, the design of these two systems is different. While operational records deal with particular records, data warehouse queries are more complex and involve large amounts of summarized data.

2.2 Multidimensional Model

Data warehouses are based on a multidimensional data model. Such a model can exist in form of a star schema, snowflake schema or a fact constellation schema. The most common modeling paradigm is the star schema which consists of a centralized fact table and several dimension tables.[1]

The primary table in a multidimensional model is the **fact table**, where the numerical measurements (which are also known as variables, metrics, or facts[6]) are stored. The facts represent the organization's measures and can be the sales of a company or investigation measures of a hospital. The measurements are defined over dimensions which tell us what the scope of a measurement is[3].

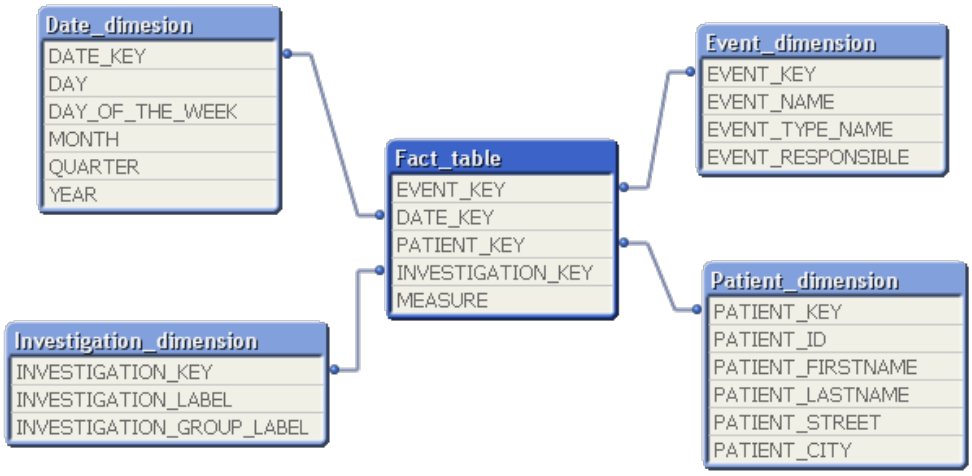


Figure 1: Star schema of a clinical data warehouse. Notice that in the centralized fact table there is the measure and the keys pointing to the dimension tables.

Dimension tables are integral companions to the fact table and contain the textual descriptor of the business[3]. Generally speaking, dimensions are entities the records can be referred to. For example, date, investigation, drug and patient might be the entities which are relevant in a medical data warehouse. For each dimension of a multidimensional model there exists a dimension table containing different levels of aggregation and some other properties of these levels[5].

2.3 QlikView, a Challenger in the Business Intelligence Software Market

Business Intelligence (BI) tools are application software designed to report and analyze data. They read data generated by organization's business performances which are mostly collected in data warehouses.

Among the market share leaders in the area of BI software tools are Microsoft, Oracle, SAP and IBM. Another big competitor of the top performing performance management software solutions is QlikTech[11]. In the last years QlikTech's BI software called QlikView has increased its market share and has become one of the biggest challengers.

According to the magic quadrant for business intelligence platforms published by Gartner Inc., a leading information technology research and advisory company, QlikView has moved from a visionary product in 2008 to one of the biggest challengers in 2010 (figure 2). Since it is a very fast evolving tool, we can expect that it will increase its value and market share even more. The success of QlikView derived from the intuitive interface, being easy to use for developers and end-users, and from the patented in-memory data calculation capability that promotes the high-performance of the tool.

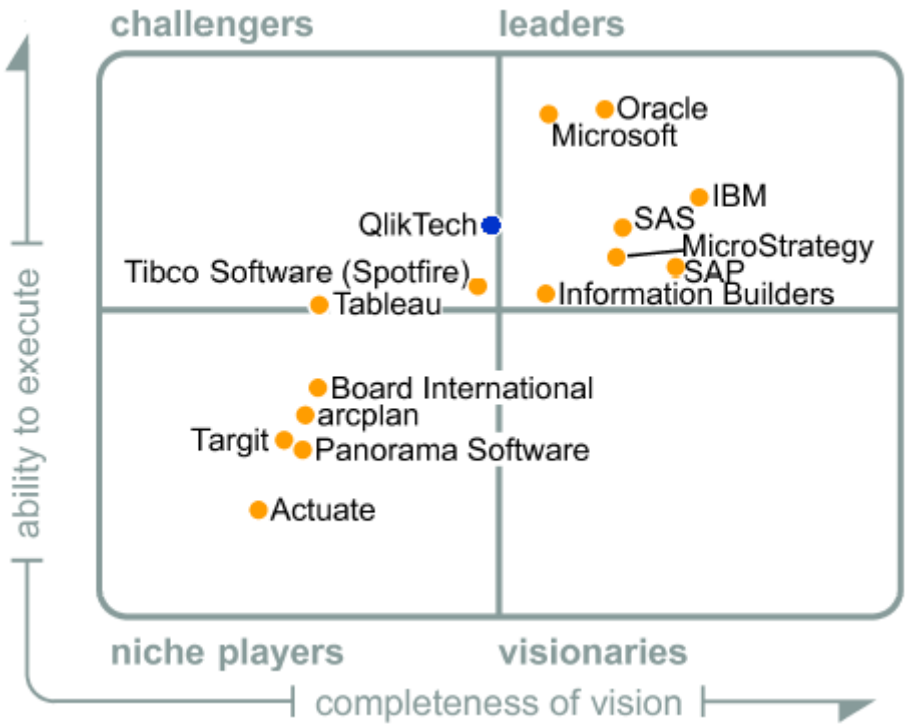


Figure 2: Quadrant for Business Intelligence Platforms published in January 2010 by Gartner Inc.[12]

3 OncoNet Data Warehouse

In this section we will present the OncoNet data warehouse and will see which information can be retrieved from it. The clinical data warehouse we are using in the thesis collects the data from an application called OncoNet, which the Department of Hematology at the Hospital of Merano currently uses. During a chemotherapy treatment plan a patient must go through different steps in order to combat the cancer over several years. During these steps data about the state of health of a patient (questionnaire surveys where the interviewer can choose from different given answers) is collected. Further, all the drug takings a person receives during a treatment plan are recorded. The data warehouse includes also laboratory tests and collects the measurements of different investigations such as pulse, blood pressure or blood compositions.

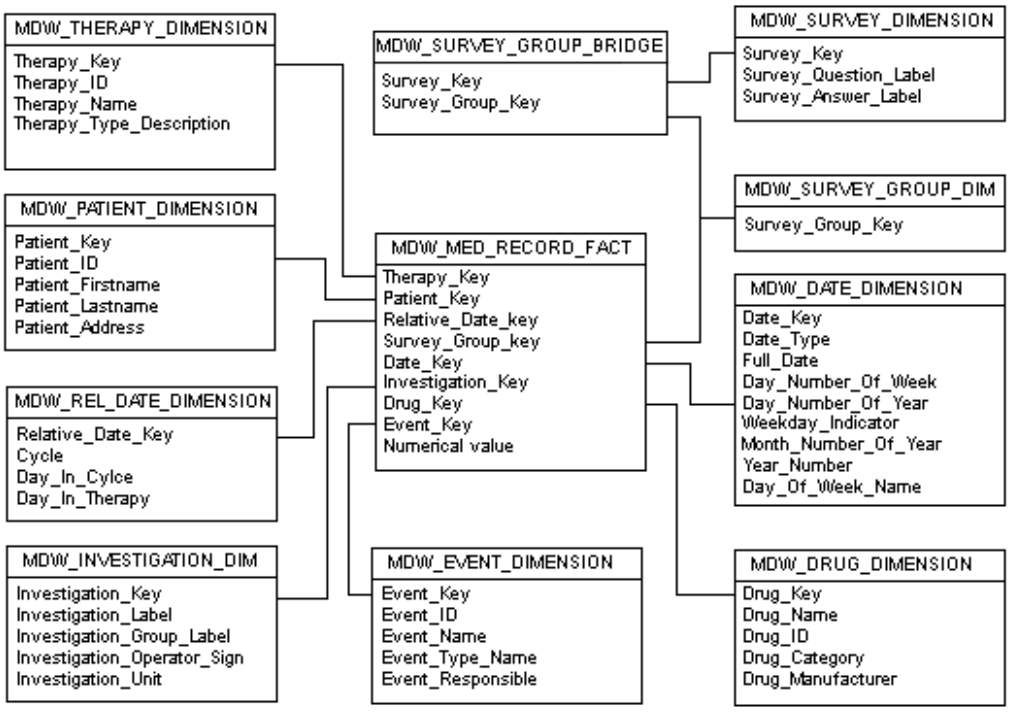


Figure 3: Shows the multidimensional model of the underlying OncoNet data warehouse. In many dimension tables there are shown only a subset of the columns.

The central fact table contains the measurements (numerical value). The measurements can describe two dimensions.

Firstly, it is a numerical value of an investigation such as a blood pressure measurement. In this case the foreign key for the investigation dimension is set:

Table 1: Fact Table Entry for an Investigation Measure

Date Key	Patient Key	Therapy Key	Drug Key	Event Key	Investigation Key	Survey Group Key	Numerical value
123	223	-2	-1	4608	**4118**	-1	140

Table 2: Investigation Dimension Entry

Investigation Key	Investigation Group	Investigation Label	Investigation Operator
4118	Blood Pressure	Systolic	=

Secondly, the measurement can describe the amount of an intake of medication. In this case the investigation key will point to *Not Applicable* and the foreign key for the drug dimension is set:

Table 3: Fact Table Entry for a Drug Measure

Date Key	Patient Key	Therapy Key	Drug Key	Event Key	Investigation Key	Survey Group Key	Numerical value
3425	2123	6138	**7242464**	-1	-1	-1	8

Table 4: Drug Dimension Entry

Drug Key	Drug ID	Drug name	Drug category	Drug quantity unit
7242464	23006	Zofran	Gastrointestinali Antiemetici	mg

3.1 Statistics of the Health State

One important information that can be retrieved from the OncoNet data warehouse is the questionnaire data collected during a chemotherapy treatment. The survey is based on fixed questions. For each question a patient's state of health can be indicated by selecting one of the five provided answers. Figure 4 displays the statistics of the health state.

Figure 4: This diagram shows the survey statistics. There is shown the partial sum, which displays immediately the number of patients to which a state of health was given.

To get an impression how QlikView makes the development of powerful analysis simple, we will compare the obtained result of figure 4 with an equivalent SQL query.

To create the SQL query, we must join the fact dimension with the event dimension and the multivalued dimensions survey_group_dimension, survey_group_bridge and survey_dimension in order to get all question and answer labels. Finally, we group the result by event name, question label and answer label to get the respective count.

```
select event_name,sd.survey_question_label, sd.survey_answer_label, count(*)
from mdw_event_dimension ed
inner join mdw_medical_record_fact mrf
  on ed.event_key = mrf.event_key
inner join mdw_survey_group_dimension sgd
  on sgd.survey_group_key = mrf.survey_group_key
inner join mdw_survey_group_bridge sgb
  on sgb.survey_group_key = sgd.survey_group_key
inner join mdw_survey_dimension sd
  on sd.survey_key = sgb.survey_key
where survey_key <> -1
group by (event_name,sd.survey_question_label, sd.survey_answer_label)
```

To get the corresponding result in QlikView as presented in figure 4 we create a pivot table with the dimensions *event name*, *survey question* and *survey answer*. As expression *count(Event_Key)* is used to count how often an answer was given to a certain question. We do not have to consider the joins, which can be fairly complex in large data warehouses, but we only need to specify the attributes of the dimensions we are interested in. Because of the fact that the OncoNet data warehouse contains also fact entries, where the survey dimension is not relevant as dimension entity, we want exclude those entries from the result shown by the pivot table (see SQL query: *where survey_key <>-1*). Therefore, we have to expand the dimension where we inserted *survey question* to an expression:

if("Survey_Question"<>'Not applicable',"Survey_Question")

Notice that in our case Survey_Question <>'Not applicable' does mean the same as Survey_Key<>-1.

Anyway, if we take into consideration that we are dealing with a data warehouse where all entities are relevant for the measurements, the creation of analysis in QlikView would be much easier than making SQL queries. Moreover, the results are shown in nice diagrams which can be used directly for reports.

4 General QlikView Template for Multidimensional Data Models

4.1 Description

Qlikview simplifies the analysis of data and after having made some experience as a QlikView application developer reports and analysis can be generated in a few days or weeks. Anyway, it would be desireable that, for instance a doctor, who has no experience in QlikView and programming, can generate his own reports and doing evaluations without consolidating a QlikView application developer. The idea is that the user sees the underlying data of a multidimensional data model. Then he can directly click on the attributes that he wants to have in the analysis diagram. Additionally, he can

choose how he wants to have represented the diagram, for instance as table or pie chart. The advantage of such an application is that the user does not need any background knowledge about QlikView and its diagram components. Moreover, he sees the entire star schema and according to it he can directly choose the information he is interested in. This template application should not only be the basis for the OncoNet data warehouse but it should also be usable with any other multidimensional data model.

4.2 Construction of a General QlikView Template for Multidimensional Data Models

The most basic sheet object in a QlikView application is the list box. Each list box represents a column (field) of a loaded database table. Once a list box is associated with a field, it shows all values of this field and the main way of making queries is through the selection of field values. After selecting field values, the program instantaneously shows all the values that are related to that selection in the document. Taking the data from table 5, we can insert, for instance, a list box of the field *therapy type* and correspondingly its values *Antidiarrheal* and *Hydration* are shown. Then we can make a query and by selecting therapy type *Antidiarrheal* we will see all information according to that selection: all therapy names that are part of that therapy type such as *Octreotide* and *Loperamid* from table 5 and other related information that is displayed in sheet objects.

Table 5: Therapy Dimension Table

Therapy Key	**Therapy ID**	**Therapy Name**	**Therapy Type**
535	2463	Octreotide	Antidiarrheal
145	2674	Loperamid	Antidiarrheal
147	3657	250 Ml Nacl 0,9%	Hydration

To construct a QlikView template application we need to allow interaction at a higher level of abstraction. This means that the user can create diagrams and reports by specifying which fields (or also called dimensions in relation to the diagrams) he wants to use as grouping attributes for the analysis. The user should not select possible values of therapy type but rather therapy type itself. QlikView does not provide a way where a selection on the level of column names is possible. Therefore, we need a corresponding attribute-list table for each database table where the column names are inserted as values of a field.

To make this more comprehensible we will look at an example. Looking at table 5 the corresponding attribute-list table contains all column names or field names as tuple:

Table 6: Therapy Attribute-List Table

Therapy Dimension
Therapy Key
Therapy ID
Therapy Name
Therapy Type

We have to create this additional attribute-list table for each dimension table and also for the fact table of the data warehouse. We can run a SQL script (in our use case Oracle SQL), which creates these additional tables automatically. Basically, the script runs through each table of the data warehouse, then it creates the corresponding attribute-list table and inserts the column names as tuples in the newly created table. In our OncoNet data warehouse all tables start with "MDW" (medical data warehouse) and so we can query in a outer loop all table names that start with "MDW" and create the corresponding attribute-list table with the same name and an "AL" (attribute list) prefix. In the inner loop, we run through all column names of the original table and insert these as tuples in the new attribute-list table.

```
begin
  for t in (select object_name from user_objects where object_type='TABLE'
                                AND object_name like 'MDW%') loop
    newtablename := 'AL_' || t.object_name;
    createtable := 'CREATE TABLE ' || newtablename || '(' ||
                SUBSTR(t.object_name,5) || ' varchar2(100) NOT NULL, '';
    execute immediate createtable;
    for  colname  in  (select COLUMN_NAME from USER_TAB_COLUMNS where
                                TABLE_NAME = t.object_name) loop
      insertvalue := 'INSERT INTO ' || newtablename || '(' ||
       SUBSTR(t.object_name,5) || ') VALUES (''' || colname.COLUMN_NAME || ''')';
      execute immediate insertvalue;
    end loop;
  end loop;
end;
```

The attribute-list tables solve two problems: the column names of the tables can be displayed in list boxes and they can be selected by the user. This allows us to create a template application where the user can select the grouping attributes of the diagrams he wants to create. For instance, he can select therapy type and therapy name as grouping attributes. Additionally, the attribute-list tables represent the structure of the underlying data model and so it should be easy for the user to select the attributes when he can see the underlying data model.

After importing the new attribute-list tables with the load script of our QlikView application, we can display the fields of the attribute-list tables in list boxes and create a panel (see figure 5), which provides the user the possibility to select the attributes of a certain dimension (such as month in date dimension and therapy name in therapy dimension).

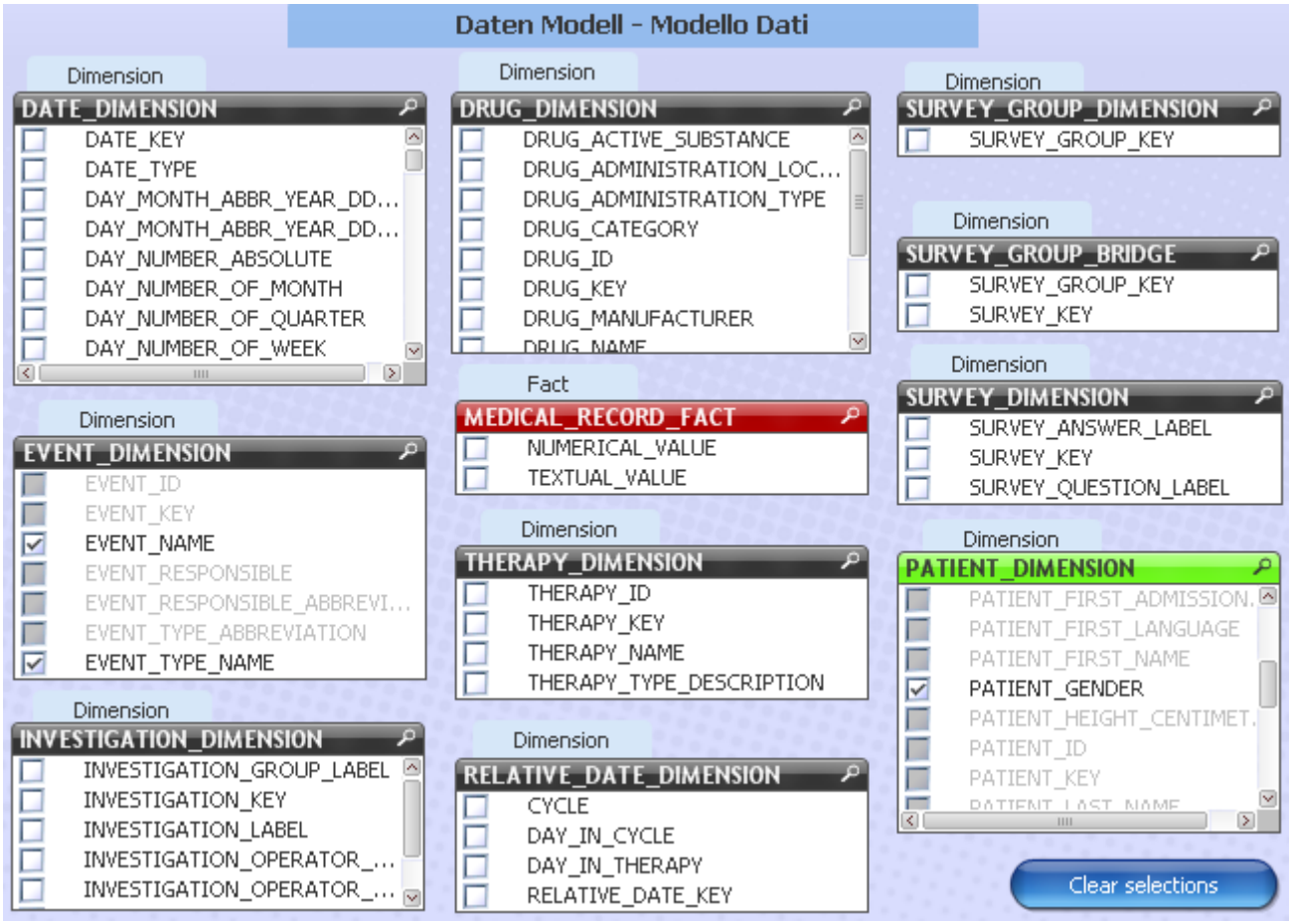


Figure 5: Shows the panel with the attributes of each dimension. The user has the possibility to select different attributes. In this example *event name*, *event type* and *patient gender* were selected. For an intuitive perception, the fact table is colored differently and centralized and does not show the foreign keys because they are already contained in the dimensions.

In addition to the panel where the data model is shown, we will need an input panel (figure 6) where the user can insert additional information after having selected the dimension attributes that he wants to include in the analysis:

- List of chosen attributes: the user should see the dimension attributes which he has chosen. He can also see the order of the selected attributes, which is important for the grouping of the analysis report. The order of the grouping is the same as the order of the selection, i.e. the first chosen attribute is also the first grouping attribute. As QlikView object we can use a customized input box and display the QlikView variables, where the chosen dimensions are stored. The variables must be defined in advance and its values are set by a macro when a selection in a dimension occurred. Additionally, there is a *clear selections* button which will deselect all chosen dimensions.

- Diagram type: contains a list with all possible diagram types that can be created with QlikView Automation Reference for analysis diagrams. In the load script of the application we can define the possible diagram types and create a list box where these values are shown and only one value should be selectable.

```
diagrams:
LOAD * INLINE [
Diagram type
Bar Chart
Line Chart
Combo Chart
Radar Chart
Gauge Chart
Scatter Chart
Grid Chart
Pie Chart
Pivot Table
Straight Table];
```

- Aggregation function: contains a list with the common aggregation functions like sum, average, maximum, minimum and count. In a classic multidimensional model the aggregation function is mostly applied to the measures. Nevertheless, it is necessary for some analysis to apply the aggregation function count on other attributes, for instance, to sum up the number of patients making a certain therapy (*count(PatientID)*). The approach to get the aggregation functions in a list box is the same as we have already seen for the list box of the diagram types. Additionally, we need to show the values of all attribute-list tables for specifying which attribute the aggregation function should be applied to. Since the values of the attribute-list tables for each dimension are in separate fields, they can not be shown together in a QlikView component. Therefore, we need a new field where all attributes are coalesced. In the load script we define the field All_Attributes and select the tuples of all attribute-tables:

```
All_Attributes:
SQL SELECT DATE_DIMENSION AS ALL_ATTRIBUTES
FROM ONCO."AL_MDW_DATE_DIMENSION";
SQL SELECT DRUG_DIMENSION AS ALL_ATTRIBUTES
FROM ONCO."AL_MDW_DRUG_DIMENSION";
SQL SELECT EVENT_DIMENSION AS ALL_ATTRIBUTES
FROM ONCO."AL_MDW_EVENT_DIMENSION";
SQL SELECT INVESTIGATION_DIMENSION AS ALL_ATTRIBUTES
FROM ONCO."AL_MDW_INVESTIGATION_DIMENSION";
...
```

- Sheet selection: the user should be able to create new sheets and select a sheet where the newly generated analysis diagrams are created. The existing and newly created sheets are stored in a table that is created in the load script.

```
SheetTable:
LOAD * INLINE [
Sheets
];
```

At runtime, the created sheets are integrated into the table with the dynamic update command and are displayed in the multi box.

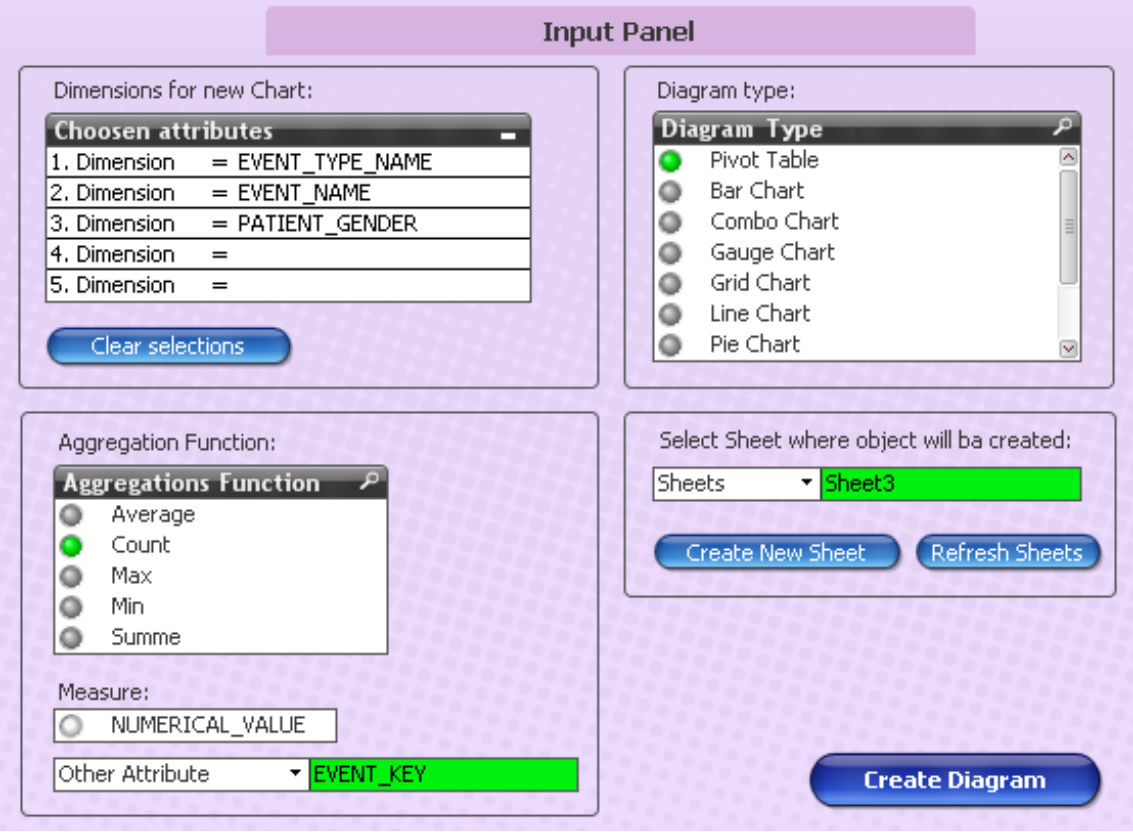


Figure 6: Shows the input panel.

4.3 Implementation

QlikView has integrated an Automation interface which allows internal macros to access and control objects of an QlikView application[9, 10]. Macros can be written either in JScript or VBScript (Visual Basic Script) programming language. Given that all the examples in the Automation Reference API are in VBScript, it is the most used script language in QlikView and therefore we will also use it for our QlikView template application. It is important to know that the macros can be invoked as an action trigger in different ways, for example, after selections in any field or pressing a button or when a sheet is activated. For a deeper understanding of the QlikView functions I would suggest to look in the QlikView Refence API.

Once we want to make a new diagram, we need all the chosen attributes (or dimension levels). Since we want to be able to show the currently chosen attributes at any point in the panel of the chosen attributes, we need to keep track of any selection. We can define a field event trigger on the fields of the attribute-list tables that are shown in table 5 and on the event *OnSelect* we call a macro *SelectionInDimensionField* that stores the selections in QlikView variables. We can assume that when the event is triggered, the list box, where the selection occurred, changes its status to active. In this case a selection means

that either a value was selected or a chosen value was deselected. Moreover, we only know in which field the selection occurred but we do not know which value was the last selected one in a case when more are selected. Therefore, the method *appendToChoosenFields*, which assigns the selected value to the QlikView variables, should append the value only if it was not appended so far. At each selection event there is also the possibility that the macro is called due to a deselected value. The method *removeDeselectedValues* runs through all variables and checks if its value or attribute is still selected. Otherwise, the variable is emptied and the order rearranged.

```
sub SelectionInDimensionField
   set doc = ActiveDocument
   activeObject = getActiveSheetObjectFromMain
   ' check if activeElement of mainSheet
   if activeObject <> -1 then
      set listbx = doc.GetSheetObject(activeObject)
      set field = listbx.getField
      set selections = field.GetSelectedValues
      for i = 0 to selections.Count - 1
         appendToChoosenFields(selections.Item(i).text)
      next
      removeDeselectedValues
   end if
end sub

sub appendToChoosenFields(field)
   i = 1
   exist = false
   while i <= MAX_DIMENSION and exist = false
      varCont = ActiveDocument.Variables(i & ".␣Dimension").GetContent.String
      if strcomp(varCont,"") = 0 then
         ' no entry in dimension dimension i
         ActiveDocument.Variables(i & ".␣Dimension").SetContent field, true
         exist = true
      else
         ' there is an entry in dimension i
         if strcomp(varCont,field) = 0 then
            ' this entry of the field already exists
            exist = true
         end if
      end if
      i = i+1
   wend
end sub
```

The attributes are stored in QlikView variables and can be easily retrieved when the user wants to instantiate a new diagram. As we have seen, the user has to insert additional information such as the diagram type and the aggregation function. This information can be read directly from the QlikView objects when the user pushed the *Create Diagram* button and according to the selected diagram type the corresponding

QlikView object is instantiated. Then, the dimensions, which after the previous work can be easily read from the variables, are added to the diagram. The following code lines are part of the function which creates the new diagram. In this example the function *makeDiagram()* creates the new diagram of the selected type and on the specified sheet. Finally, we add the selected dimension attributes and the aggregation function as expression to the diagram and then we get the desired result.

```
sub createDiagram
   ...
   set allattr = doc.Fields("ALL_ATTRIBUTES").GetSelectedValues
   aggregationAtt = allattr.Item(0).text
   set diagramTyp = ActiveDocument.Fields("Diagramm␣Typ").GetSelectedValues
   set newDiagram = makeDiagram(diagramTyp.item(0).Text, sheetName)
   ' append selected dimension levels to the diagram
   for each present in createArrayOfSelectedDimensions
      newDiagram.AddDimension present
   next
   ' set expression
   set aggregationExp = ActiveDocument.Fields("Agg␣Expression").GetPossibleValues
   newDiagram.AddExpression aggregationExp.item(0).text &"("& aggregationAtt &")"
   ...
end sub

function makeDiagram(dia, sheetName)
   On Error Resume Next
   set sheet = ActiveDocument.GetSheet(sheetName)
   if Err.number <> 0 Then
      set sheet = ActiveDocument.GetSheet("Main")
   end if
   On Error Goto 0
   if dia = "Bar␣Chart" then
      set makeDiagram = sheet.CreateBarChart
   ...
   elseif dia = "Pie␣Chart" then
      set makeDiagram = sheet.CreatePieChart
   elseif dia = "Straight␣Table" then
      set makeDiagram = sheet.CreateStraightTable
   else
      set makeDiagram = sheet.CreateLineChart
   end if
end function

function createArrayOfSelectedDimensions
   countDimensions = getCountSelectedDimensions
   redim arrDim(countDimensions - 1)
   for i = 0 to ubound(arrDim)
      arrDim(i) = ActiveDocument.Variables(i + 1  & ".␣Dimension").GetContent.String
   next
   createArrayOfSelectedDimensions = arrDim
end function
```

5 QlikView Template Supporting Cube Operations

5.1 Description

In the last section we have seen how we can construct a template application for a multidimensional model in QlikView. Now we will expand the model further, so that we can do some OLAP or cube operations on the selected data. Cube operations allow a flexible view changing on the created diagrams. Sometimes, the user wants to see the selected data in a more detailed way or even in a more general way. The user does not have to create new diagrams and entering the new dimension attributes but he can select the diagram and perform a cube operation in order to change the view of the selected data.

In the next subsections we will get the idea of concept hierarchies and look in more detail at cube operations. After designing a model that supports the most important cube operations we will focus on the implementation of some primary functions in QlikView.

5.2 Concept Hierarchies

A concept hierarchy can be defined as a sequence of mappings from a set of low-level concepts to higher-level concepts[1]. A common example of a concept hierarchy is the date dimension which can be divided into the dimension-levels *day*, *month* and *year* (figure 7). We can aggregate data according to the low-level day and map it afterwards to a more general level such as month or year. Another example used in our clinical data warehouse is the investigation dimension (see table 7). The finest granularity is the investigation label which could be *diastolic* and *systolic*. Both investigation labels are part of the investigation group label *blood pressure* which then corresponds to a higher-level, more general concept.

Table 7: Investigation Dimension

Investigation Group Label	**Investigation Label**
blood pressure	diastolic
blood pressure	systolic
lymph node	inguinal size left
lymph node	inguinal size right
lymph node	supraclavicular size left
lymph node	supraclavicular size right

In [4, 6] a dimension D is formally defined as a lattice of levels $(L,<)$ such as: $L=(L_1, L_2...L_n)$ where L_i represent the different levels. So the dimension date would be a lattice $(L,<)$:$L=(day, month, year)$ and $day<month<year$ is an example of a partial order (figure 7).

Now, if we define our date dimension in a more detailed way and expand it with the attributes week and quarter the concept hierarchy becomes more complicated. Since a week often crosses the boundary of a month and therefore it belongs to two months, sometimes

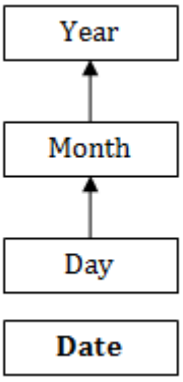


Figure 7: Date dimension with dimension levels.

weeks are not considered as a lower abstraction of the month level. It is then treated as a lower level of year and a year is then approximated to 52 weeks. Consequently, the new partial orders of the date dimension would be $day < month < quarter < year$ and $day < week < year$ as we can see from figure 8.

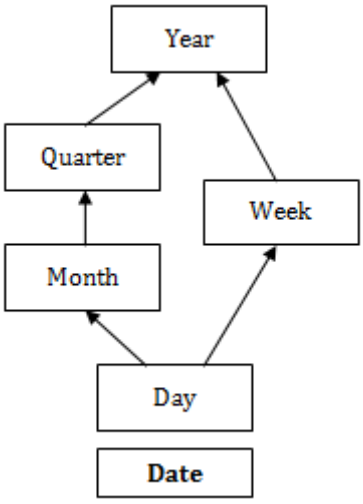


Figure 8: Date dimension with dimension levels.

One task will be to define manually the concept hierarchies of the multidimensional model. This will be important since we later define cube operations which can be performed only on the predefined concept hierarchies. Beside the many concept hierarchies that can be defined in each date dimension, possible hierarchies in the OncoNet data warehouse are shown in figure 9.

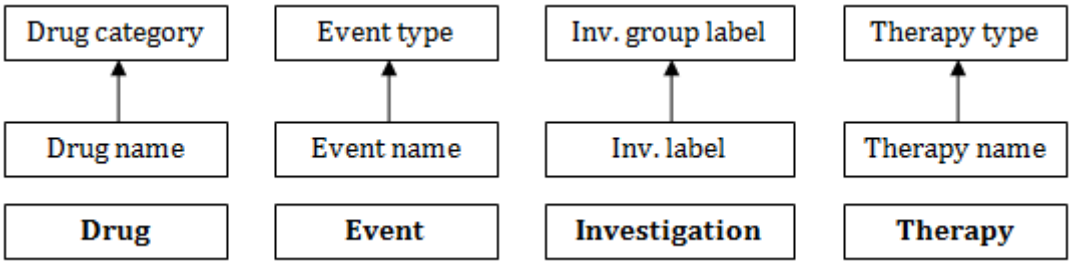


Figure 9: Shows possible concept hierarchies in the use case of the OncoNet data warehouse.

5.3 Cube Operations

Cube operations or OLAP operations are well known operations in OLAP tools, which provide multidimensional analysis to the underlying information[5]. As we have seen, the dimensions contain different levels of aggregation. This concept of logical hierarchies is useful because it provides a flexible view of the data from different perspectives. A number of well-known cube operations, which support interactive view changing, exist:

- Roll-up: the roll-up operation (drill-up) corresponds to an aggregation of the data from a lower level of hierarchy to a higher level of hierarchy within a dimension[4]. In the hierarchy of the investigation dimension *investigation label*<*investigation group label* the roll-up operation from the investigation label is defined as the climbing up in the concept hierarchy to investigation group label.

- Drill-down: the drill-down is the reverse of the roll-up operation. It corresponds to a navigation from a higher level to a more detailed level of abstraction. For example, the resulting attribute of a drill-down operation from the *quarter* attribute in the concept hierarchy of *day*<*month*<*quarter*<*year* (figure 8) would be *month*. Consequently, the data will be displayed in a more detailed way.

- Slice and dice: The slice operation would correspond to a selection on one dimension. For instance, by selecting *quarter = "1"* in the date dimension, it will only show those data sets that are in the first quarter of a year. The dice operation is defined as a selection in two or more dimensions such as *quarter = "1"* and *investigation group label = "blood pressure"*.

5.4 A Model Supporting Cube Operations

As we have seen, each dimension is a lattice of levels. We name a path in a dimension hierarchy as dimension path[4] or hierarchy path and define it as a path from a most detailed level to a most general level.

From figure 8 we can extract two dimension paths: $day \rightarrow week \rightarrow year$ and $day \rightarrow month \rightarrow quarter \rightarrow year$.

To allow cube operations in our QlikView application we must add to each dimension in the model its dimension hierarchies. So we define the existing hierarchy paths in our data warehouse as we have seen in section 5.2 and expand the attribute-list tables of each dimension with two additional columns, the dimension level and the dimension hierarchy. The dimension level indicates the level of aggregation of the corresponding attribute and the dimension hierarchy gives us the concept hierarchy to which the attribute belongs.

Table 8: Therapy Attribute-List Table

THERAPY_DIMENSION	DIMENSION_LEVEL	DIMENSION_HIERARCHY
THERAPY_KEY	-1	-1
THERAPY_ID	-1	-1
THERAPY_NAME	1	1
THERAPY_TYPE	2	1

The placeholder *-1* indicates that we do not want that this attribute belongs to a dimension path. In table 8 we defined the only possible dimension path as *therapy name<therapy type*.

Now we will see how the table might look like when there are more dimension paths such as in the example of the date dimension. The basic idea is that we extract each possible path. In the case of figure 9 we define two dimension paths and we can recognize that day and year are contained in both dimension paths. So we also need these attributes to appear twice in the attribute-list table of our date dimension. The dimension hierarchy indicator tells us that we defined two concept hierarchies, one consisting of $day \rightarrow week \rightarrow year$ and the other of $day \rightarrow month \rightarrow quarter \rightarrow year$.

Table 9: Date Attribute-List Table

DATE_DIMENSION	DIMENSION_LEVEL	DIMENSION_HIERARCHY
DAY	1	1
WEEK	2	1
YEAR	3	1
DAY	1	2
MONTH	2	2
QUARTER	3	2
YEAR	4	2

To complete the model we will expand it with a few functions. First of all, if we want to perform a cube operation on a dimension level (or attribute) of any dimension, we need to know which dimension the attribute belongs to. So we define the function

$$D = dimension(attribute)$$

which returns the dimension name where the attribute is contained; *D = dimension('Therapy Name')='Therapy Dimension'.*

Moreover, let

DP = dimension_paths(dimension, attribute)

be a function which gives us all dimension paths (dimension hierarchies) where the passed attribute is contained; *dimension_paths('Date Dimension', 'Day')=[Day, Week, Year]; [Day, Month, Quarter, Year].*

Then we introduce the function

level(dimension, attribute, hierarchy) = k

where k is a number representing the level in a dimension path, starting with 1 for the lowest level (for instance in figure 9, the level of the attribute *Week* in the dimension path [Day, Week, Year] would be *2*). Knowing the dimension *D*, the dimension paths *DP*, and the level *k*, it is possible to do the cube operation roll-up to level *k+1* or a drill-down to level *k-1*.

Concluding this model we define a function for the roll-up operation as

newAttribute = rollUpInDimensionPath(dimension, level, dimension_path);

rollUpInDimensionPath('Date Dimension', 1, [Day, Week, Year])= 'Week' and *rollUpInDimensionPath('Date Dimension', 1, [Day, Month, Quarter, Year]) = 'Month'*. It is important to obtain all dimension paths with the function we defined above and then retrieve all possible attributes for a roll-up operation, because it might occur that a roll-up is ambiguously defined such as in our date dimension.

Analogously, we can do the same for a drill-down operation, we just introduce a new function

newAttribute = drillDownInDimensionPath(dimension, level, dimension_path)

which gives us the new attributes when performing a drill-down operation.

5.5 Implementation of the Model for Cube Operations

In this subsection we will see some implementation examples in QlikView. We will focus on the functions that we defined in subsection 5.4.

The function *dimension(attribute)* will retrieve all list boxes that contain the dimension levels. From a list box it is possible to retrieve the field of which the list box shows the values. Once the field is retrieved, we can search if the attribute is contained in that dimension and if this is the case return that dimension.

```
function dimension(attribute)
   arr = getDimensionListBoxIDs
   for each present in arr
      set field = ActiveDocument.GetSheetObject(present).GetField
      count = field.SearchFor(attribute, True, 1)
      if uBound(count) >= 0 then
         dimension = field.Name
      end if
   next
end function
```

In our model we have defined the function *dimension_paths* as a function that returns the entire paths with all dimension levels. In the implementation of the function we are not retrieving the entire path but only the indicators of the dimension hierarchy (see column dimension hierarchy of table 8 and table 9), since each path is uniquely defined by the indicator and so the path can be easily retained. If we want to retrieve all dimension hierarchies where *Day* is contained, we immediately think that we can make an SQL query, but it is impossible to execute an SQL query in a QlikView macro. Nevertheless, we can execute the select command and select *Day* in the date dimension field which is equivalent to an SQL query. After this selection, all values from dimension_level and dimension_hierarchy, which are not related to *Day*, are excluded and we can use the function *GetPossibleValues* in the field of dimension_hierarchy. *GetPossibleValues* will return an array with all possible hierarchy values where *day* is contained; in our example it will be (1;2).

```
function dimension_paths(dimension, attribute)
   ActiveDocument.Fields(dimension).Select attribute
   set x = ActiveDocument.Fields(dimension &"_HIERARCHY").GetPossibleValues
   redim arrHierarchies(x.Count - 1)
   for i = 0 to x.Count - 1
      arrHierarchies(i) = x.item(i).text
   next
   dimension_paths = arrHierarchies
end function
```

In order to make a roll-up (or drill-down) operation we need to know the level index of a dimension level. The implementation of the function *level(dimension, attribute, hierarchy)* is similar to the function *dimension_path*. We select the dimension attribute and the hierarchy and can request the possible values of the level index.

```
function level(dimension, attribute, hierarchy)
   ActiveDocument.Fields(dimension).Select attribute
   ActiveDocument.Fields(dimension &"_HIERARCHY").Select hierarchy
   set x = ActiveDocument.Fields(dimension &"_LEVEL").GetPossibleValues
   if x.Count > 0 then
      level = x(0).text
   else
      level = -1
   end if
end function
```

Finally, we have to implement the function *newAttribute = rollUpInDimensionPath(dimension, level, dimension_path)*. We can evaluate an expression by setting the level indicator to level + 1 to obtain the roll-up attribute (level - 1 in drill-down operation).

```
function rollUpInDimensionPath(dimension, level, hierarchy)
   expression = "Only(if("& dimension &"_LEVEL="&  level + 1 &"AND"
        & dimension &"_HIERARCHY="&hierarchy&",dimension))"
   rollUpInDimensionPath = ActiveDocument.Evaluate(expression)
end function
```

As soon as the new attribute is found, we only need to retrieve the diagram where the roll-up (or drill-down) operation is performed and reset its dimensions by replacing the old attribute with the new attribute.

In parallel with the additional macros that contain the functions for performing cube operations, we will need to expand the QlikView application itself.

Since we want to perform cube operations on the created diagrams, we need an additional panel with a list box, which shows the created diagrams. After selecting a diagram there appears another list box with the dimensions belonging to this diagram (figure 10). Then, if there is a possibility to perform a roll-up or drill-down operation, the buttons are enabled properly.

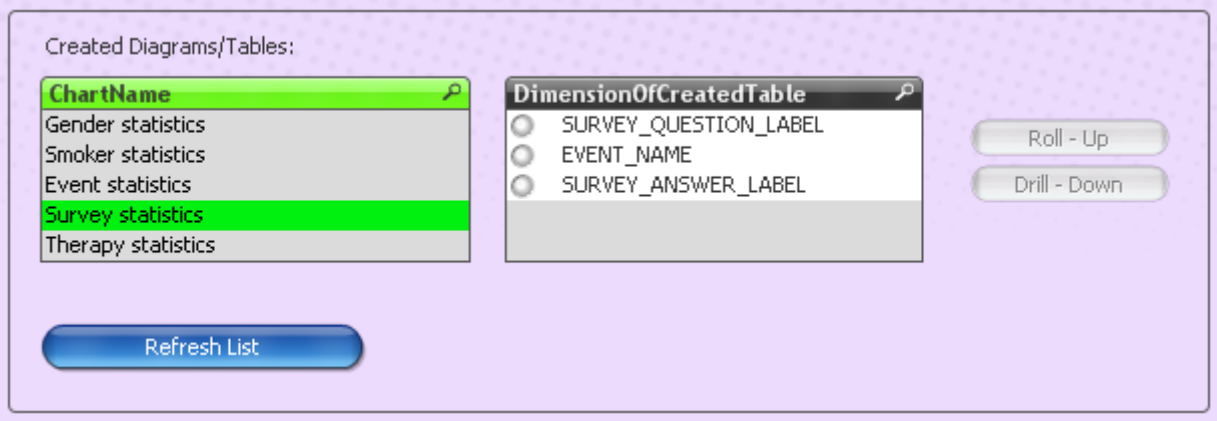


Figure 10: Shows the panel with the created diagrams and after a selecting one, its dimension attributes show up in the other list box. The buttons for performing a cube operations are disabled since no dimension was selected.

To be able to show the created diagrams, the application has to record them after they are instantiated. In the load script of the application we define two empty tables, one for the diagrams and another for the dimensions which belong to a diagram and are referenced by its unique *ChartID*.

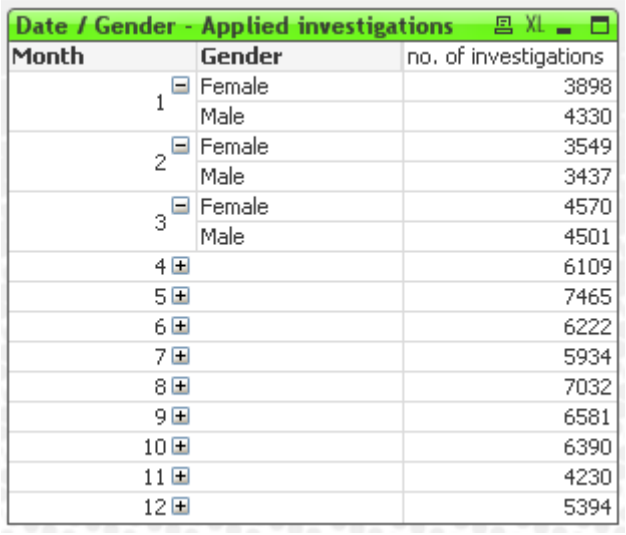
```
// Table containing all diagram objects
ChartTable:
LOAD * INLINE [
ChartName, ChartID
];

// Dimensions of a created diagrams
DimensionOfCreatedTables:
LOAD * INLINE [
DimensionOfCreatedTable, ChartID, DimNumber
];
```

After each diagram creation the diagram name, its unique ID that is needed for retrieving the QlikView objects and its dimensions are stored in these tables with the dynamic update command. Furthermore, we need also an event trigger that calls a macro which removes all entries of deleted tables.

Finally, we will see two examples in QlikView, where we can perform a cube operation. For instance, we will create a diagram that shows the number of applied investigations in the year 2006 (select 2006 in a year list box) grouped by month and gender. We select *Month* and *patient gender* as dimension attributes and *count(investigation key)* as

expression. As diagram type we can specify a pivot table and get the following diagram.

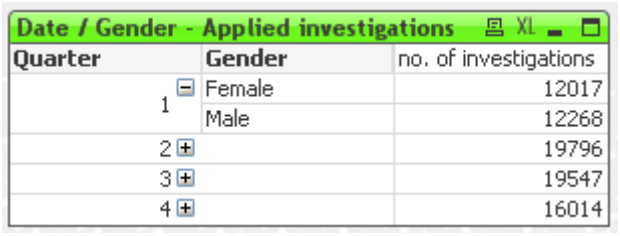


Date / Gender - Applied investigations

Month	Gender	no. of investigations
1	Female	3898
	Male	4330
2	Female	3549
	Male	3437
3	Female	4570
	Male	4501
4		6109
5		7465
6		6222
7		5934
8		7032
9		6581
10		6390
11		4230
12		5394

Figure 11: Pivot table showing the number of applied investigations grouped by month and gender.

After selecting this diagram, we can perform a roll-up operation from *month* dimension level to *quarter*. Respectively, the pivot table is changed and we can see the results grouped by quarter and gender:

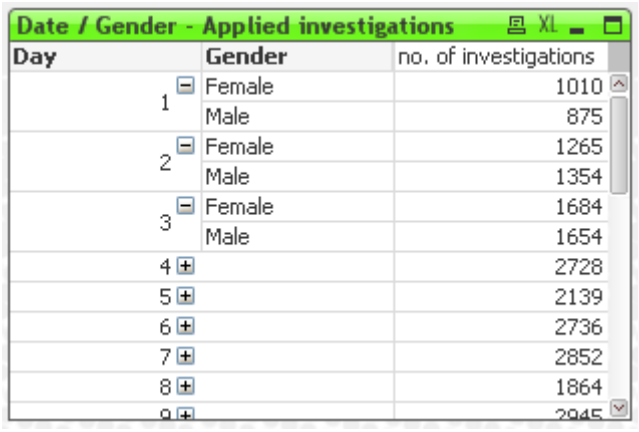


Date / Gender - Applied investigations

Quarter	Gender	no. of investigations
1	Female	12017
	Male	12268
2		19796
3		19547
4		16014

Figure 12: Pivot table showing the number of applied investigations grouped by quarter and gender.

Equally, we can also perform a drill-down operation from *month* to *day* in the initial table. Then, we get the data aggregated in a more detailed way, where the number of applied investigations is grouped according to day and gender.



Date / Gender - Applied investigations

Day	Gender	no. of investigations
1	Female	1010
	Male	875
2	Female	1265
	Male	1354
3	Female	1684
	Male	1654
4		2728
5		2139
6		2736
7		2852
8		1864
9		2945

Figure 13: Pivot table showing the number of applied investigations grouped by day and gender.

Another example is the diagram in figure 14 that shows the gender statistics about patients who did the therapy type "Chemioterapia E Sostanze Immunostimolanti". We can see how many of the patients are males and how many are females.

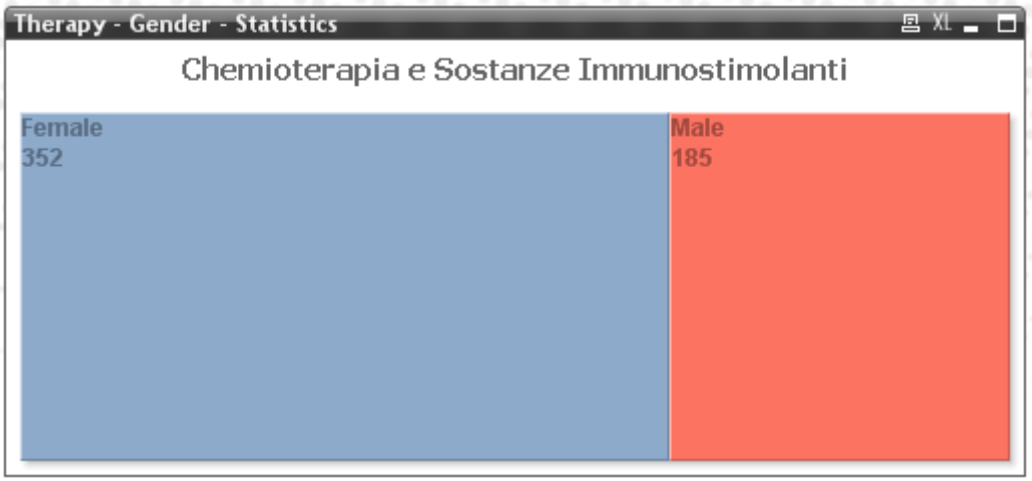


Figure 14: Block chart showing gender statistics about therapy type "Chemioterapia E Sostanze Immunostimolanti".

Since we defined therapy type as an upper level of therapy name, we can perform a drill down operation and look at the data in a more detailed way by granulating from therapy type to therapy name.

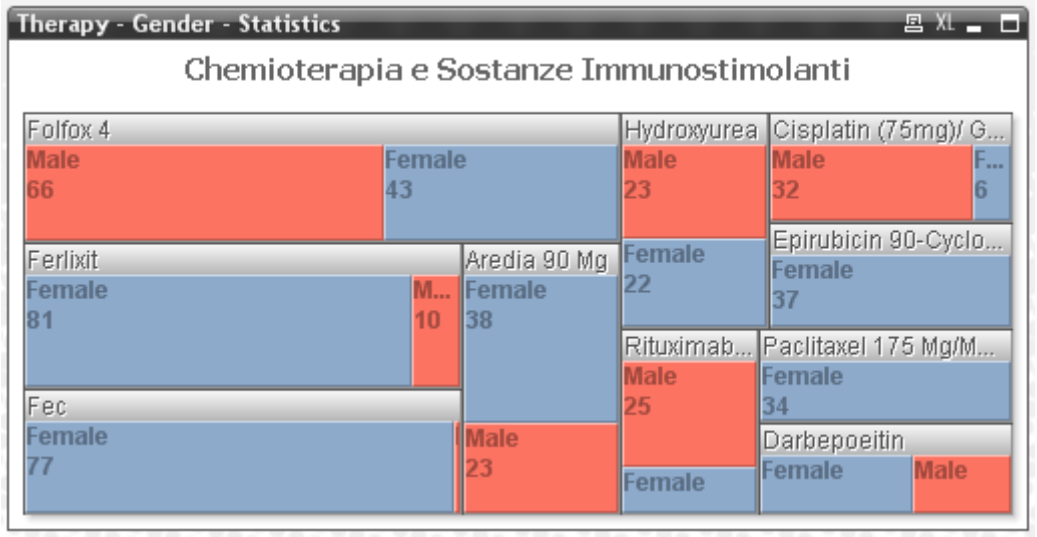


Figure 15: Block chart showing gender statistics about the therapies of type "Chemioterapia E Sostanze Immunostimolanti".

Now we got a detailed gender statistic about the therapy type "Chemioterapia E Sostanze Immunostimolanti", where the data is grouped according to the more detailed level therapy name.

6 Related Work

Inmon[2] and Kimball[3] are probably the most cited authors regarding data warehousing and multidimensional modeling. Inmon provides a clear definition of the data warehouse and describes its main features. Kimball focuses on areas, where data warehouses are used and additionally he describes the design of a data warehouse when dealing with medical records.

Jiawei Han and Micheline Kamber[1] describe different technologies of data analysis and present the basic concepts and architectures of data warehouse and on-line analytical processing (OLAP).

Vassilidias[4, 5, 6] published many research papers focusing on on-line analytical processing and cube operations. In [6] Vassilidias describes a model for multidimensional databases and introduces dimension hierarchies and cubes. Furthermore, he designs a model that supports a series of operations on cubes such as level climbing, function application and navigation.

In [7] the possibility of using data warehousing in OLAP technologies in public health care and how interactive exploration and analysis can be enabled are discribed.

On the page [8] there can be found a case study about QlikView in the public health

care. Some good examples that demostrate how QlikView can be used in the public health care for report generation and statistics gathering are provided. Moreover, it demostrates the vary possibilities of different diagrams that can be used to represent the data.

Gartner Inc.[12] and Aberdeen Group[11] are information technology research and advisory companies and both do market analysis for Business Intelligence tools. Gartner Inc. uses its Magic Quadrant for BI platforms providing a qualitative analysis of the market and its participants. Similar to the Magic Quadrant, the Aberdeen Group uses its Business Intelligence Performance Management (BIPM) axis to classify the products according to value delivered and market readiness. These documents describe the strenghts and weaknesses of the different solutions that exist on the Business Intelligence software market.

7 Conclusion and Future Work

In this thesis we have seen how we can import data of a clinical data warehouse in QklickView. Moreover, we generated some analysis reports and investigated the suitability of QlikView together with the OncoNet data warehouse. We underlined that QlikView can not only be used by application developers for report generation, but there exists the possibility to make a template application using VBScript macros, which allows non-experts to create reports and diagrams by interacting at a higher level of abstraction. The template is not only the basis for the use case of the OncoNet data warehouse but can be applied also to any other multidimensional data model. Additionally, the template application supports cube operations on the level of diagrams and a flexible view changing can be carried out by the user.

At the moment, to set up the template application for another underlying multidimensional data model, some work has to be done manually in order to get the template application to work, for instance, importing the structure tables (attribute-list tables) in list boxes and linking the QlikView components to the macros.

A future work could be to concentrate on the definition of an installation macro which automatically instantiates the QlikView components of the template application after the user has indicated the tables of the star schema and associates them to the defined macros.

References

[1] Jiawei, Han / Kamber, Micheline: *Data mining. Concepts and Techniques*. Second Edition. 2006.

[2] Inmon, W.H.: *Building the Data Warehouse*. John Wiley & Sons. 2006.

[3] Kimball, Ralph / Ross, Margy: *The Data Warehouse Toolkit*. Second Edition. 2002. John Wiley & Sons.

[4] Vassilidias, Panos / Skiadopoulos, Spiros: *Modelling and Optimization Issues for Multidimensional Databases*. CAiSE 2000: international conference on advanced information systems engineering No. 12, Stockholm. 2000.

[5] Vassilidias, Panos / Sellis, Timos: *A Survey of Logical Models for OLAP Databases*. SIGMOD Record. Vol. 28, No. 4. December 1999.

[6] Vassilidias, Panos: *Modeling Multidimensional Databases, Cubes and Cube Operations*. In Proceedings of the 10th SSDBM Conference. 1998.

[7] Hristovski, Dimitar / Rogac, Mitja / Marakota Mladen: *Using Data Warehousing and OLAP in Public Health Care*. In Proceedings of the AMIA Symp. 2000.

[8] http://www.4s-dawn.com/clinicalperformance/Report.htm current as of July 1, 2010.

[9] QlikTech International: *QlikView Reference Manual*. Version 9.0 for Windows. Sweden. October 2009.

[10] QlikTech International: *QlikView Automation Interface Reference*. Version 9.0 for Windows. Sweden. June 2009.

[11] Aberdeen Group, Inc., David Hatch, Michael Lock: *Business Intelligence (BI): Performance Management AXIS*. 2009.

[12] Gartner RAS Core Research, Joseph Feiman, Neil MacDonald: *Magic Quadrant for Business Intelligence Platforms*. January 2010.