This degree project was done in cooperation with
Nordicstation
Mentor at Nordicstation: Christian Isaksson

NORDICSTATION

Flexible Data Extraction for Analysis using Multidimensional Databases and OLAP Cubes

Flexibelt extraherande av data för analys med multidimensionella databaser och OLAP-kuber

TOBIAS HULTGREN
ROBERT JERNBERG

Degree Project in
Computer Engineering
Undergraduate, 15 Credits
Mentor at KTH: Reine Bergström
Examiner: Ibrahim Orhan
School of Technology and Health
TRITA-STH 2013:23

Royal Institute of Technology
School of Technology and Health
136 40 Handen, Sweden
http://www.kth.se/sth

Abstract

Bright is a company that provides customer and employee satisfaction surveys, and uses this information to provide feedback to their customers. Data from the surveys are stored in a relational database and information is generated both by directly querying the database as well as doing analysis on extracted data. As the amount of data grows, generating this information takes increasingly more time. Extracting the data requires significant manual work and is in practice avoided. As this is not an uncommon issue, there is a substantial theoretical framework around the area.

The aim of this degree project is to explore the different methods for achieving flexible and efficient data analysis on large amounts of data. This was implemented using a multidimensional database designed for analysis as well as an OnLine Analytical Processing (OLAP) cube built using Microsoft's SQL Server Analysis Services (SSAS). The cube was designed with the possibility to extract data on an individual level through PivotTables in Excel.

The implemented prototype was analyzed, showing that the prototype consistently delivers correct results several-fold as efficient as the current solution as well as making new types of analysis possible and convenient. It is concluded that the use of an OLAP cube was a good choice for the issue at hand, and that the use of SSAS provided the necessary features for a functional prototype. Finally, recommendations on possible further developments were discussed.

Keywords: OLAP, cube, multidimensional database, business intelligence, data warehouse, SSAS

Sammanfattning

Bright är ett företag som tillhandahåller undersökningar för kund- och medarbetarnöjdhet, och använder den informationen för att ge återkoppling till sina kunder. Data från undersökningarna sparas i en relationsdatabas och information genereras både genom att direkt fråga databasen såväl som att göra manuell analys på extraherad data. När mängden data ökar så ökar även tiden som krävs för att generera informationen. För att extrahera data krävs en betydande mängd manuellt arbete och i praktiken undviks det. Då detta inte är ett ovanligt problem finns det ett gediget teoretiskt ramverk kring området.

Målet med detta examensarbete är att utforska de olika metoderna för att uppnå flexibel och effektiv dataanalys på stora mängder data. Det implementerades genom att använda en multidimensionell databas designad för analys samt en OnLine Analytical Processing (OLAP)-kub byggd med Microsoft SQL Server Analysis Services (SSAS). Kuben designades med möjligheten att extrahera data på en individuell nivå med PivotTables i Excel.

Den implementerade prototypen analyserades vilket visade att prototypen konsekvent levererar korrekta resultat flerfaldigt så effektivt som den nuvarande lösningen såväl som att göra nya typer av analys möjliga och lättanvända. Slutsatsen dras att användandet av en OLAP-kub var ett bra val för det aktuella problemet, samt att valet att använda SSAS tillhandahöll de nödvändiga funktionaliteterna för en funktionell prototyp. Slutligen diskuterades rekommendationer av möjliga framtida utvecklingar.

Acknowledgments

We would like to thank Nordicstation for the opportunity to do our degree project with them. Special thanks to Christian Isaksson at Nordicstation for mentoring and technical advice. We would also like to thank Bright for providing the problem this degree project set out to solve. Special thanks to Kent Norman who was our contact at Bright for describing the requirements and giving feedback during the work. Lastly we would like to thank KTH and our lecturers for the education we have received over these years. Special thanks to Reine Bergström at KTH for his advice and mentoring during our degree project.

Abbreviations

BI: Business Intelligence. A set of technologies and methods for extracting relevant business intelligence from data sources.

OLAP: On-Line Analytical Processing. A database designed for analytical purposes.

OLTP: On-Line Transaction Processing. A database designed for transactional purposes.

DBMS: DataBase Management System. A system used to manage a database, such as SQL Server.

ETL: Extract/Transform/Load. A set of actions intended to extract information from a source, transform it if necessary, and finally load it into a destination.

SQL: Structured Query Language. Language standard for querying databases. This standard is to some degree followed by major DBMS's.

SSAS: SQL Server Analysis Services. Microsoft's BI tool for data analysis.

SSIS: SQL Server Integration Services. Microsoft's BI tool for data migration.

SSRS: SQL Server Reporting Services. Microsoft's BI tool for producing graphical reports.

MDX: MultiDimensional eXpressions. A query language similar to SQL used to query OLAP databases.

GUI: Graphical User Interface. A graphical user interface used for user interaction.

API: Application Programming Interface. An interface to application functions accessible by other applications.

JDBC: Java DataBase Connectivity. A standard API for SQL database connectivity.

XMLA: Extensible Markup Language for Analysis. Industry standard XML based language for OLAP.

MOLAP: Multidimensional Online Analytical Processing. Storage mode where both atomic data and aggregates are stored in the cube.

ROLAP: Relational Online Analytical Processing. Storage mode where both atomic data and aggregates are stored in the relational database.

Contents

Chapter 1

Introduction

1.1 Background

Bright is a company that performs customer and employee satisfaction surveys. They collect answers to questions regarding satisfaction from millions of users for companies in different industries, and uses this data to provide feedback to the respective companies on what their customers and employees are satisfied with and also to investigate what can be done to improve relations with them. These surveys are carried out using the web and automated telephone calls.

Because of the large amounts of data in their relational database it has become a challenge to extract the sought information, and to do this in a fast and flexible way.

1.2 Goals

The goal of the task is to develop a solution that can extract the sought information and to do this in an efficient manner without adversely affecting the customers' use of the database. There are two major kinds of information that are sought; aggregates and correlations.

Aggregates refers to the summarized information of a specific range of data in the database. For example, the total number of surveys for company X in June 2012 or the average general customer satisfaction for company Y in 2013.

Correlations in our case refers to correlations between answers given from one customer. For example, "how does the waiting time for customer support affect the customer satisfaction?". Correlations may also be a correlation between different aggregates, eg. average waiting time compared to average customer satisfaction, but the further away from individual surveys the analysis takes place, the less valuable

it is for Bright's analysts.

Furthermore, the solution must be compatible with the in-place systems used by Bright, which are mostly based on the Microsoft platform.

1.3 Delimitations

The solution was based on the problems as stated above and the system was developed with Bright's needs in mind. Feature requests that did not serve to further the depth of this degree project were left to be implemented on a future improvement basis. The time span for developing a prototype was limited to six weeks and the goals were limited to this.

The database provided by the company runs the SQL Server 2008 R2 DBMS. This was the database that was to be used without further analysis of which DBMS might be optimal for the purpose of data analysis.

With consideration to the great amount of tools available for the purpose of data analysis, this degree project will not provide an exhaustive list or review of each of them. Only a selection of tools will be briefly reviewed for the purpose of picking one that provides the necessary functionalities we require.

1.4 Solutions

The problem was solved by implementing a dimensional database in SQL Server and a data cube on top of it using Microsoft SQL Server Analysis Services (SSAS). The dimensional database provides an efficient storage format for large queries as well as being a format easily used in SSAS. The data cube provides an intuitive and efficient way of browsing the data with the necessary aggregates available. Extract / Transform / Load (ETL) operations can be run routinely to populate the dimensional database with up-to-date data from the transactional database. The ETL operations can be run nightly to not adversely affect customer use of the database. All analyses is done on the cube and dimensional database ensuring that the transactional database is not adversely affected.

Chapter 2

Current Situation

Currently, there are two entities which are using information from the database; the web interface and internal analysis.

The **web interface** is accessible by all of Bright's customers and serves to present different aggregates; eg. the average customer satisfaction for different sections in the company. The users can browse different company hierarchies and filter on time ranges. Which data that can be browsed is pre-defined and rarely changed. As of now this data is derived partly from specially prepared tables in the database and partly from directly querying the transactional data. Even though the prepared tables speed up the queries it can still take several seconds to execute them when aggregating data for a large company or over a large time span.

The **internal analysis** serves to let Bright's analysts further analyze the data to find interesting information and be able to give their customers recommendations on how to improve customer and employee satisfaction. It is also used for internal reporting such as billing. To get all the data needed for the internal analyzing in a useful format, some fairly complex queries has to be run against the database. Much of the work is done using Microsoft Access databases and then copying data into Excel spreadsheets. As this is done manually, it takes a lot of time and has to be repeated every time new data is needed. It introduces a risk of human error each time the data is handled.

Chapter 3

Theoretical Framework

In a field as expansive as Business Intelligence[1], there are many relevant concepts and technologies. These concepts are briefly explained in the following sections.

3.1 Business Intelligence

Business Intelligence (BI) is a generic term which refers to a wide array of technologies and methods used to extract relevant business information from different data sources that can be used to help making informed decisions in a business.[2]

3.2 Data Warehouse

While a transactional database is intended and designed for processing (mostly small) transactions, a data warehouse is intended and designed for processing (often large) queries and data analysis. It is common that a data warehouse contains historical information as opposed to a transaction database which may only contains the current situation.[3, 4]

Why use a Data Warehouse? IBM estimates that, as of January 2012, that 90% of the world's data was created in the past two years. This data comes from all kind of source including purchase transactions, social media and climate information sensors. Clever solutions are necessary to process this data in an efficient manner in order to extract relevant information and acquire sought knowledge. This is done using a data warehouse which stores all historical data and is designed to process it.[5]

3.3 Data Mart

A Data Mart refers to stored data for a single business process. For example, one Data Mart can keep information about customer satisfaction, and another can keep

information on employee satisfaction. It is common that a number of data marts make up a data warehouse.[4]

3.4 Extract / Transform / Load

Extract / Transform / Load - or just ETL for short - refers to the process of **extracting** data from data source(s), **transforming** it to the format expected by the destination(s), and then **loading** it into the destination(s). In some organizations there can be a large number of data sources and destinations involved in the ETL process including databases using different DBMS, plain text files, spreadsheets using different formats, web services etc. In some cases the ETL process can end up being incredibly complex. For this reason there are a number of ETL tools available to simplify and automate this process.

3.5 Data Mining

Data Mining is the process of extracting interesting information from large amounts of data. Such information is not limited to pre-defined questions such as "What is the average customer happiness served by employee X?" but instead attempts to find patterns, anomalies or other significant structures in the data. Thus, data mining (also more descriptively known as Knowledge Discovery in Data) can help answer questions that go beyond simple queries.[6]

3.6 OLTP vs OLAP

OnLine Transaction Processing (OLTP) is a term used for everyday use of a database, often to modify or access individual elements. This is usually done on a specific transactional database designed with transactions in mind. Such a database follows certain rules to avoid redundancy and ensure consistency in the database, but may perform poorly if subjected to large queries affecting a large number of rows. For this purpose, the data in the OLTP database is copied to an OnLine Analytical Processing (OLAP) database which is designed with respect to a different set of rules for the purpose of analytical performance.

3.7 Dimensional database

A dimensional database design is based around one or more fact tables surrounded by a number of dimension tables. The fact table contains quantifiable values such as numbers, and the connected dimensions contains information that give meaning to those values. A dimensional database can be designed with two similar patterns: **Star** or **Snowflake** schema.

The two design patterns for a dimensional database both include a central fact table surrounded by dimensions. The difference is that a snowflake design allows dimensions to be connected to other tables whereas dimensions in a star schema cannot be connected to any other tables. This can make the snowflake schema less efficient in queries on account of more joins being necessary, but may also reduce space requirements for the database because of less redundant data. A snowflake schema can also be more normalized to ensure data consistency, a welcome feature in a transactional database. Examples of star and snowflake designs can be found in figures 3.1 and 3.2 respectively[1].

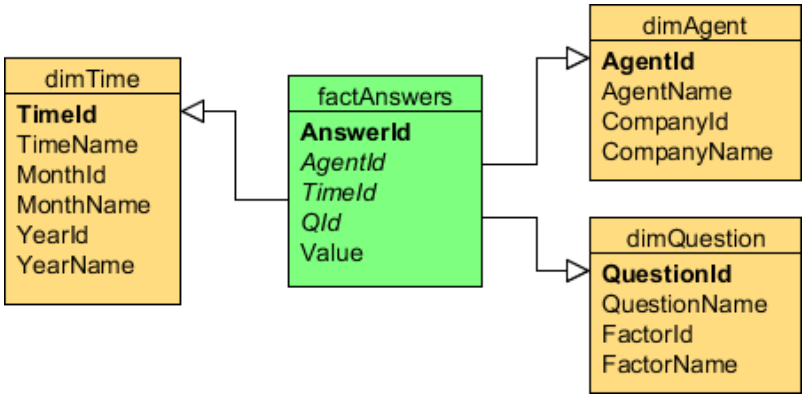


Figure 3.1. An example of a star schema.

Transactional databases are designed with respect to the third normal form (3NF) in order to minimize redundancy, ensure data consistency and perform fast inserts, edits and deletes of single rows. A dimensional database on the other hand allows redundancy for the sake of performing faster queries of large amounts of data. Because of this it if is often used with analysis tools, which may query large portions of the database to extract interesting information.

3.8 OLAP Cube

An OLAP cube can generally be viewed as a multidimensional spreadsheet. Each axis in this spreadsheet would be a dimension such as "Time", and each cell would be a number of measures (described below). An OLAP cube pre-processes a database to generate metadata about it which can be used to make filtering and displaying data much easier for the user. Cubes work with dimensions and are thus easier and faster to use with a dimensional database, but may also use other database

[1]This and following examples are adapted to provide a relevant context with Bright's database, but acts only as examples and may not be consistent with the actual database.

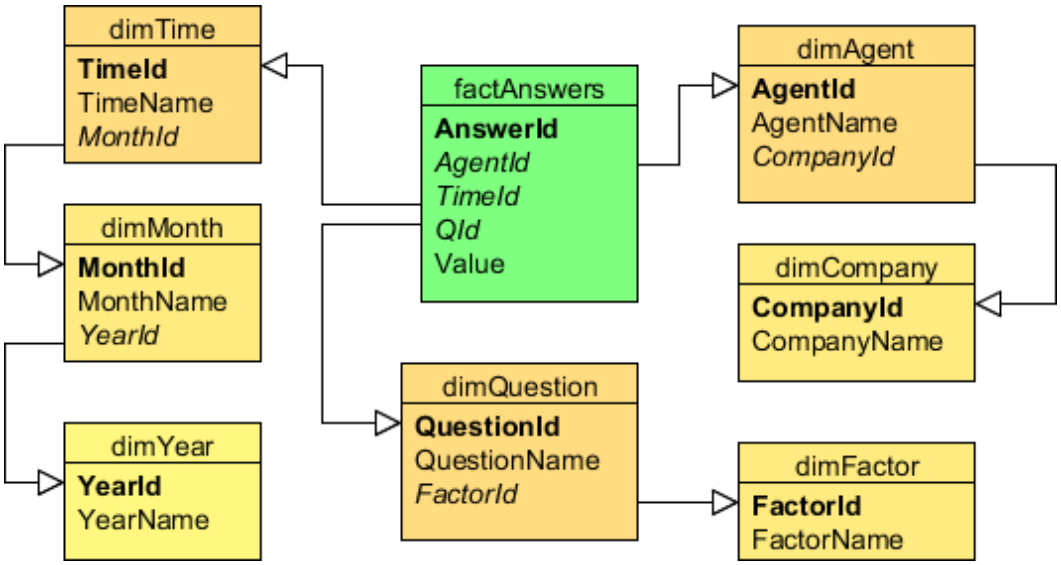


Figure 3.2. An example of a snowflake schema.

structures. There are a number of concepts and concerns regarding these cubes that will be discussed in this section.

Measures. Central in an OLAP cube are **measures**. These are quantifiable values, i.e. boolean or numeric, stored in a fact table and used to measure the stored data. For example the score given by the responding customer.

Dimensions. In order to filter, group and sort the stored information as well as to give meaning to the measures, one or more dimensions are used. A dimension may contain a number of attributes to describe it. Each dimension can be viewed as one of the axises of the cube. Example: A time dimension may have attributes for Year, Month and Date allowing information to be extracted dependent on the time the survey was submitted.

Hierarchies. Attributes in a dimension may be organized in one or more **hierarchies**. Hierarchies often exist naturally in the data and defining them in the cube allows users to browse it in a intuitive way. For example, in a time dimension a reasonable hierarchy is $Year \rightarrow Month \rightarrow Day$.

Pre-calculated measures. Even with a highly efficient database structure such as discussed in section 3.7, computing aggregates of huge ranges of data takes time. Those aggregates may also be accessed frequently and thus put a significant strain on the DBMS. In order to be able to present measure aggregates quickly, they can be pre-calculated in the cube.

Data explosions. In order to pre-calculate all measures, one value for all possible intersections of the cube dimensions, known as the Cartesian product, would have to be stored. Since this will grow very fast as more dimensions are added and the number of rows in each dimension increases, the space needed to store all pre-calculated values can become a problem. This phenomena is know as a **data explosion**, and OLAP cube tools often have ways to solve this issue.[7]

Chapter 4

Methods

As with most problems there are many ways to solve it. Based on the given goals and circumstances, different methods that could be used were considered. After choosing a method, available tools for the development of a solution were examined.

4.1 Requirements

To be able to meet the given goals of this degree project, the chosen method must meet several requirements. These requirements can be divided into three categories; simplicity, performance and flexibility.

Simplicity. The users of the prototype are not to be expected to be knowledgeable in programming languages, database structure or other technical fields.

Performance. While the current solution of directly querying the transactional database for most aggregates can complete within a few seconds for now, it is certain to get worse as the size of the database increases. Even now there can be serious delays during heavy usage, which is not trivial as these queries are made by users through a web interface. Thus, the generation of these aggregates needs to be more efficient to ensure adequate performance even as the database scales in terms of size and user activity.

Flexibility. There is currently no convenient solution of generating other statistics than aggregates from the data such as correlation analysis, as explained in chapter 2. The prototype must be capable of retrieving data in a flexible and efficient manner, in a format usable for Bright's analysts. Alternatively, the prototype should be capable of doing user-defined analysis on the data and present results.

4.2 Candidate methods

4.2.1 Modifying the current solution

One method is to modify the current solution to meet the specified requirements. This would involve both improving the performance of the aggregate generation as well as implementing an application which can extract data in a flexible way through a simple user interface. The extracted information would need to be in a format which is useful for analysis. There are a number of issues present when attempting a solution with this method.

- Implementing a simple and flexible user interface for users to choose which data they want to retrieve is not a straightforward task. What types of data a user wants, how it should be grouped and filtered, as well as the format in which it should be delivered can come to change with time. It is not reasonable to build a solution that is generic enough to cover all these cases within the limited time span.

- The authors of the current solution have already put significant effort into improving the database performance. Finding ways to further improve it without fundamental changes would be a great challenge.

4.2.2 Using an OLAP tool

Another method would be to use an OLAP tool which is designed to handle analysis work on large amounts of data. An OLAP tool may work directly against an OLTP database or a database designed and dedicated for OLAP. It can handle aggregation of values, generate metadata for making data navigation intuitive as well as allowing multidimensional queries. OLAP tools may also include functionality for doing calculations, analysis and put together reports on the server side. Furthermore, it is possible to implement pre-calculated aggregates in an OLAP cube to prepare answers to frequently asked queries. Some of the issues of using an OLAP tool are as follows.

- Because there is such a large selection of OLAP tools, choosing the optimal one is a challenge. Because of the limited time span of this task it is not possible to fully evaluate several tools. Instead, the comparison between the candidates would have to be largely literary.

- Developing a completely new system could turn out to be significantly more work than to simply modify the in-place solution to meet the new requirements.

4.3 Chosen method

Using a OLAP tool has a number of advantages over building a custom solution based on using the current OLTP database.

Simplicity. An OLAP tool provides a intuitive way of browsing the available data and retrieving it in a flexible way, something that is very hard to design in an application.

Performance. An OLAP tool removes the need to try to improve the efficiency of the current SQL queries. Given that significant efforts have already been put into this, it is unlikely that the performance can be significantly increased. OLAP tools are designed for efficient analysis and are likely to perform significantly better than a custom-built solution.

Flexibility. Because an OLAP cube work with dimensions it is possible to retrieve data in a flexible way and in a table format which is useful for analysis. Developing an application with this characteristic is not trivial.

As such, using an OLAP tool was deemed the most suitable solution for this problem.

4.4 Tools for the chosen method

There are plenty of tools available for OLAP and business intelligence with varying functionality and design choices. Many BI software packages come as part of a set of tools extending beyond only the data analyzing. This degree project, including this chapter, will focus on data retrieval and analyzing. With respect to Bright's circumstances and requirements, a comparison was made between a handful of tools to find which would be most suitable for an implementation on the given task. The limited time that was used to evaluate the tools means that it can not be concluded that the chosen tool is the optimal choice for this implementation, but simply that it was deemed most suitable.

4.4.1 Technical requirements

- In order to increase the performance of the web interface, the tool must be able to quickly deliver aggregated values over large amounts of data.
- Much of the data fall into a natural hierarchy such as $Industry \rightarrow Company \rightarrow Unit \rightarrow Group \rightarrow Employee$ and the in-place web interface relies heavily on these hierarchies. For this reason the analysis software should work with hierarchies.

- In order not to compromise the function and performance of the transactional database and to make analysis more efficient the analysis should be made on a separate database to which data is migrated on a nightly basis.

- It must be possible to extract information on an individual level in order to do correlational analysis.

- It must be possible to retrieve the data in a format useful for analysis.

- The source database runs the SQL Server 2008 DBMS and as such the OLAP software must support it.

4.4.2 Examined tools

4.4.2.1 Oracle Essbase

Oracle provides a large number of tools for Business Intelligence. This section will focus on Essbase (Extended Spread Sheet dataBASE) which is their OLAP analytics software[8]. Essbase can use SQL Server as a data source, making it possible to use directly with the current data source. It provides the possibility to create dimensional and measure hierarchies in an OLAP cube to form a good structure for navigating the data[9]. Using Essbase data can be viewed in Excel through their Oracle Hyperion Smart View for Office software[10]. To integrate Essbase with different applications, MDX queries are supported[11].

Attempts were made to get Essbase up and running for testing purposes. However, no successful installation could be made in the time span dedicated to testing it.

4.4.2.2 Microsoft SSAS 2008 R2

Microsoft offers a Business Intelligence software suit consisting of SQL Server Integration, Analysis and Reporting Services. This section will focus on SQL Server Analysis Services (SSAS). SSAS works with dimensions in an OLAP cube to process the data. The dimensions can be organized into different hierarchy levels. SSAS also provides a set of Data Mining methods to discover information not specifically queried for[7]. The software package also includes tools for migrating data (SSIS) as well as for producing graphical reports based on the analysis (SSRS). For use as an API to SSAS, MDX queries can be used[12].

A development environment for SSAS was already installed and a test was done by following chapters three to five in SSAS 2005 step by step[7]. The data was possible to browse using PivotTable in Excel. Individual answers were not possible to browse during the evaluation because of the design of the cube.

4.4.2.3 Palo 3.2 and Jedox 5.0

Palo is an open source OLAP solution developed by Jedox AG and there is also a proprietary version named Jedox that is based on the same open source OLAP server as Palo. The OLAP server stores data in memory and allows real-time processing of the data. Data is stored in a multidimensional structure, and the tool supports organizing the dimensions into hierarchies[13]. Data can be imported to the OLAP server using the Palo ETL tool[14]. To connect the ETL tool to a Microsoft SQL Server, JDBC has to be installed. The design of cubes and dimensions are done through graphical interfaces in Excel and the Palo Web interface[13, 15]. There are available APIs for C, Java, .NET and PHP in the Jedox and Palo SDK, easing the development of applications. MDX queries can be used to integrate Jedox with different applications[16].

A trial installation using Jedox was installed and tested with the included demos. Demo data could be browsed using the PivotTable in Excel. Attempts were made to connect the ETL tool to the SQL Server database but were ultimately unsuccessful during the evaluation phase.

4.4.2.4 PowerPivot

PowerPivot is an analytics plugin for Microsoft Excel. It allows data to be imported from a variety of sources, including SQL Server. The imported data can have relations set up, hierarchies structured and measures added. When creating a PowerPivot workbook[1] the data is stored on the local machine. However, it can be published using Microsoft SharePoint which allows it to be accessed as a server. SharePoint can automatically refresh the workbook data when published[17].

There are some limitations to the file size of a PowerPivot solution. When working locally, there is a 4 GB limit to save the workbook[18]. The maximum file size when uploading to SharePoint is 2 GB[19]. When using PowerPivot with SharePoint, there are some added features such as the possibility to programmatically access the data using MDX queries. However, the MDX support is limited and there are no development tools other than Excel to design a PowerPivot solution[20].

A test implementation was made using the PowerPivot plugin for Excel 2010. Data was taken from a multidimensional database on SQL Server and hierarchies were designed. The data could be browsed using the PivotTable in Excel. No attempts were made to add the PowerPivot workbook to a SharePoint server during the evaluation phase.

[1] A workbook is the name of an Excel file. In this case it holds all the data used for a PowerPivot solution.

4.4.3 Choice of tool

Oracle Essbase seems like a viable alternative on paper, but since no successful installation could be made during the evaluation phase it was decided that it would be a poor choice given the limited time span for the degree project.

Palo and Jedox was an interesting alternative because it does support all the required functionalities. Using a web interface for configuration allows for configurations from different platforms, but it is not relevant in this case because all workstations run Windows. Storing the data in-memory allows for real-time processing, but this is not a feature that will be used in this case as analysis is done on historical data. Ultimately, Jedox is a smaller vendor in the business intelligence industry. Although the documentation covers much, there is few books and tutorials in English. Plentiful resources are necessary to complete the prototype in the given time span.

Although PowerPivot was user friendly to work with, it did have some limitations when it came to using it for the development of this solution. Since it is currently is unknown if the size limitations would be reached, it is undesirable to take the risk of ultimately not being able to use the solution. It is also preferable to be able to use a development environment and not to do it inside an application aimed at non-developers such as PowerPivot. This is partly because PowerPivot has restricted features, and a development environment is easier to revision control. Because the MDX support is limited, integrating the developed solution could prove problematic.

For developing a prototype, Microsoft SSAS was chosen as the tool to be implemented. There are a number of advantages to this choice.

- SSAS provides the functionalities specified in the technical requirements.
- Nordicstation works with Microsoft solutions and therefore they already have the programs and licenses to use them.
- Nordicstation has development environments for the development of SSAS installed.
- SSAS integrates well with SQL Server which is the DBMS used by Bright.
- There is plenty of documentation and tutorials to use during implementation.

Chapter 5

Implementation

The implementation of the data warehousing solution was done in several steps.

1. Convert relevant parts of the OLTP database to a dimensional database
   a) Identify relevant schemas, tables and columns
   b) Design a dimensional database using a star schema design
   c) Design ETL operations to populate the dimensional database
2. Implement the OLAP software
   a) Map fact table and dimension tables to software and define hierarchies
   b) Create a cube using fact table and dimension tables
3. Design and implement an appropriate interface

These steps are described in detail in the following sections.

5.1 Database conversion

5.1.1 Relevant source data

To be able to extract the relevant data from the OTLP database its design has to be understood correctly. A good understanding of what is needed for analysis and the web interface is also necessary. For Bright, the most important factual data is the answers provided from the surveys. In this solution only a subset of the answers are useful. For instance, no attempt will be made to extract information from voice messages or free text answers since it is not straightforward how to quantify this data in a way so it can be used in the database. Instead, answers based on yes or no as well as numerical questions (on a scale from 1 to 5) are used. These are the same type of answers currently used on Brights website.

5.1.2 Dimensional database design

The dimensional database will consist of one fact table holding the quantified values from given answers. The fact table includes foreign keys to each of the dimension tables which provides information to the answer. In this case there were three fundamental questions that needed to be answered in order to be able to analyze the data; **who**, **what** and **when**. These three questions are answered by the **Respondent**, **Question** and **Time** dimensions respectively and will be explained in the following three sections. Section 5.1.2.4 will explain the design of the **Answers** fact table which connects the dimensions. Figure 5.1 illustrates the resulting tables and their relations in the dimensional database.

All attributes in a dimension will have a key value to identify it. For presentation to the user this key value may not be intuitive enough, and thus a matching Name column is created for most columns. For example, the Day_Of_Week column has an integer value between 1 and 7 and has a matching name column Day_Of_Week_Name with a value of "Monday" to "Sunday" instead.

5.1.2.1 Respondent dimension

The Respondent dimension is designed to answer the question **who**, and its most fine grained member is the respondent. In this case, the respondent defines a respondent of a single survey. Even if the same person responds to several surveys, they will be stored as a different respondent each time. This has the immediate consequence that every answer from a single respondent also comes from the same survey which enables analysis on the level of individual surveys.

Each respondent is part of a hierarchy defined as $Industry \rightarrow Account$[1] $\rightarrow Unit \rightarrow Group \rightarrow Agent \rightarrow Respondent$.

5.1.2.2 Question dimension

The Question dimension is designed to answer the question **what**, and its most fine grained member is the **question**. A question can, for instance, be "How satisfied are you in general with our services?". Each question is a child to a parent **factor** which may for instance be "General satisfaction". Each factor can have several often similar child questions. In some cases a question can have several factors in the source database resulting in one separate question is created for each factor in the dimensional database. This also means that a QuestionId is not always unique making it necessary to use a separate, generated QId as a primary key. This dimension has one short hierarchy of $Factor \rightarrow Question$.

[1] An "account" is a company. The name "account" is chosen for consistency with the source data.

5.1.2.3 Time dimension

The Time dimension is designed to answer the question **when**, and its most fine grained member is the time with an hour precision. The source data enables a much higher precision, but in this case an hour precision is sufficient. Using this lower precision allows for a smaller Time table resulting in improved performance. Each hour is part of the natural hierarchies $Year \rightarrow Month \rightarrow Date \rightarrow Time$ and $Year \rightarrow Week \rightarrow Date \rightarrow Time$, where both are useful for the web interface as well as analysis.

In excess of the values necessary to create these two hierarchies a number of other values are included as well for other kinds of analysis.

Day_Of_Year Day number of current year.

Day_Of_Month Day number of current month.

Day_Of_Week Day number of current week.

Week_Of_Year Week number of current year.

Month_Of_Year Month number of current year.

These values can be used to filter the data in order to answer questions such as "Are customers generally more satisfied on Fridays than Mondays?" and "Does customers of company X generally feel the response time is higher during these months?".

5.1.2.4 Answers fact table

One row in the Answers fact table corresponds to an answer to a single question, resulting in a value from 1 to 5 if it's a numeric question or 1 or 5 if it's a yes/no question. In excess of this number each row in the fact table also includes a reference to one row in each of the three dimensions.

5.1.3 ETL operations

ETL operations were used to extract the relevant data from the OLTP database used by Bright to the new dimensional database schema. Data that was not deemed useful for analysis purposes were filtered out, such as answers that does not follow the numeric or yes/no format and respondents that have no answers connected to them. Redundant data was added where deemed necessary, most prominently the time dimension in which all columns are derived from the submission time of the survey (see section 5.1.2.3).

Since both the data source and destination were on SQL Servers, the simplest solution for ETL operations was to write an SQL script. The alternative of using

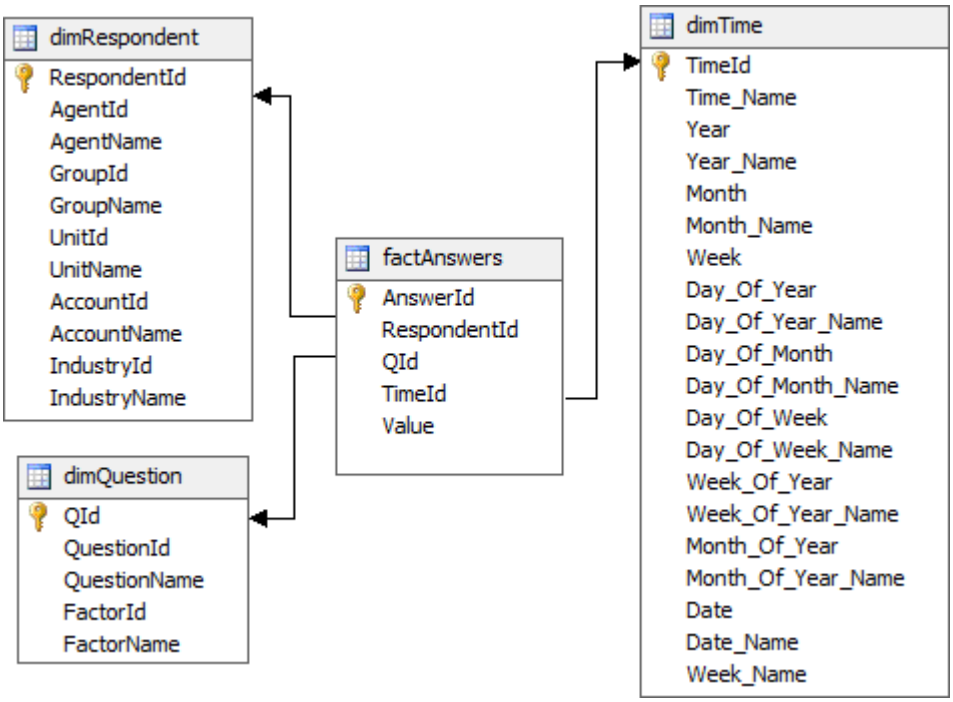


Figure 5.1. The design of the dimensional database, fact table in the middle with surrounding dimensions.

an ETL tool would require a study of available tools as well as learning how to use it with no apparent benefit. An SQL script for ETL operations was written which performs the following steps.

Clear tables. The tables of the dimensional database are emptied of all content.

Populate dimensions. The dimension tables are populated with no specific order since there are no dependencies between them.

Populate fact table. The fact table is populated last because of its dependence on all dimension tables.

A notable design choice is that all select statements for the data migration are based on the **Answers** table. When migrating the dimensions there are also a number of constraints to only migrate the necessary data. This is to ensure that no unnecessary questions, respondents or time rows are present in the dimensional database as they will not be useful for analysis. This increases analysis performance as there are less rows in the dimensional database tables, but can also significantly

increase the time to perform ETL operations because of more complexity in the ETL script. This is analyzed further in chapter 6.

When migrating the **question** dimension, there can be a number of translations for both factors and questions. Primarily, Swedish translations are used to name questions in the cube when they are available. However, sometimes these are not available or are explicitly named "Not translated" or similar. In these cases, translations for other languages are used instead in the order of preference $Swedish \rightarrow English \rightarrow Norwegian \rightarrow Danish \rightarrow Finnish$. In case none of these languages have the relevant factor or question translated, it is simply specified as "Not translated". The respondent dimension exhibit a similar system for the **industry** names, but these are expected to always have translations or are otherwise specified as "Not translated".

5.2 Data cube implementation

5.2.1 Defining dimensions

Which dimensions will be used, as well as what data these dimensions will contain, was decided in the dimensional database design phase (section 5.1.2). As discussed in section 4 the tool used for implementing the cube is SSAS. Defining the dimensions takes place in a number of steps[2].

Select source table. The source table of the dimension is selected and the dimension is created.

Attribute roles. The table's primary key is chosen as the key for the dimension. Other columns that will be used as attributes for the dimension are chosen and corresponding name column (if any, ending with "Name" in figure 5.1) are specified. The value from the name column is the one seen when browsing the cube.

Dimension hierarchies. The hierarchies which will be used to browse the dimension are defined as designed in section 5.1.2. Each item in the hierarchy aggregate the measures of its child items. Using hierarchies makes browsing the cube more intuitive for the user by adding structure to the data and also makes it possible to define the attribute relationships in a way that makes these queries efficient.

Attribute relationships. The relationship between the attributes are defined as parent-child relationships. The relationships can be defined as **Flexible** or **Rigid**. A flexible relationship is one where a child attribute can have several

[2]This is the process when using a dimensional database with a star schema design, and may differ with other schema designs.

parent attributes. For example, a question has a flexible relationship to a factor. This is because the same question can have several factors. A rigid relationship is one where a child attribute only have one parent attribute. For example, a month has a rigid relationship to a year - the month May 2013 will always relate to the year 2013. A rigid relationship may be defined as a flexible one without errors, but the reverse is not true. Choosing the correct attribute relationship is important to avoid incorrect computations and optimize query and processing performance[21].

Subsections 5.2.1.1 through 5.2.1.3 will briefly describe these steps for each of the dimensions. In the attribute relationship figures arrows are used to denote attribute relationships. A filled arrow means a relationship is rigid and a hollow arrow means it's flexible.

5.2.1.1 Respondent dimension

The respondent dimension have a number of attributes: RespondentId, AgentId, GroupId, UnitId, AccountId and IndustryId. All of these attributes have a corresponding name column except for Respondent which uses its Id as a name. This is because the name of the respondent is not relevant for analysis purposes, and it is often lacking in the source data. All of these attributes are used in the hierarchy as illustrated in figure 5.2. The attribute relationships are illustrated in figure 5.3.

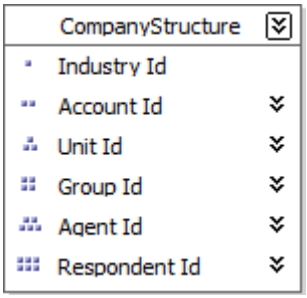


Figure 5.2. The company structure hierarchy in the respondent dimension.

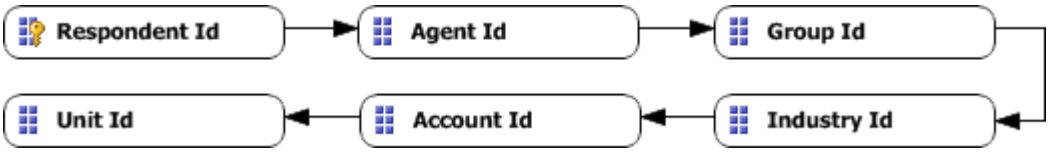


Figure 5.3. The attribute relationships in the respondent dimension.

5.2.1.2 Question dimension

The question dimension only have three attributes: QId, QuestionId and FactorId. QuestionId and FactorId have corresponding name columns and are used in the hierarchy as illustrated in figure 5.4. The attribute relationships are illustrated in figure 5.5.

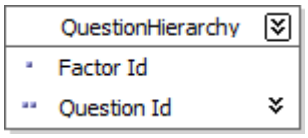


Figure 5.4. The question hierarchy in its dimension.

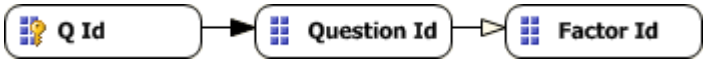


Figure 5.5. The attribute relationships in the question dimension.

5.2.1.3 Time dimension

The weeks follow the ISO 8601 standard of week enumeration because ISO week numbers are unique per year[22]. This allows for a rigid relationship between year and week. The two hierarchies of the time dimension are illustrated in figure 5.6. The attribute relationships are illustrated in figure 5.7.

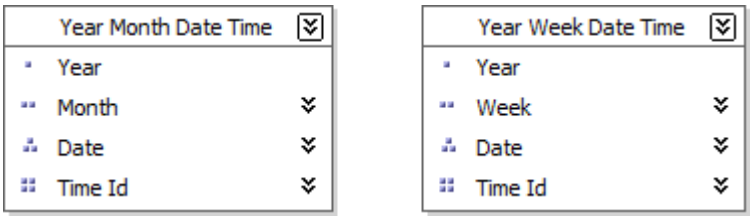


Figure 5.6. The $Year \rightarrow Month \rightarrow Date \rightarrow Time$ and $Year \rightarrow Week \rightarrow Date \rightarrow Time$ hierarchies in the time dimension.

5.2.2 Creating the cube

The cube was created by selecting the fact table and choosing the measures that should be included from it. All three connected dimensions are then chosen so the cube can be browsed using them. In SSAS there are some standard measures that can be used whereof three were used in this prototype.

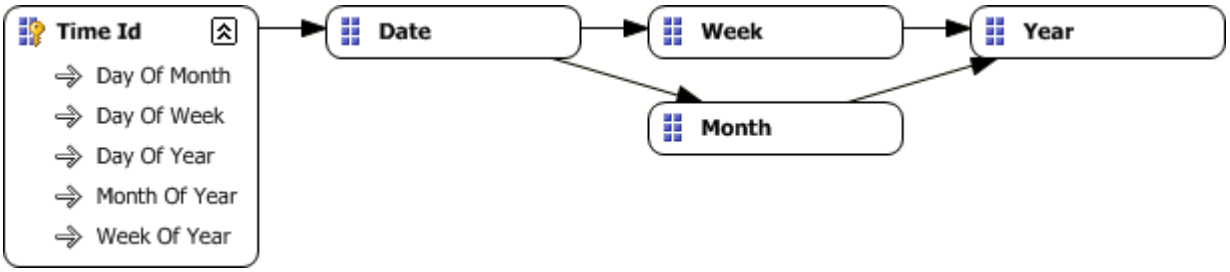


Figure 5.7. The attribute relationships in the time dimension. The attributes listed below TimeId have a flexible relationship to TimeId, but are not part of a hierarchy.

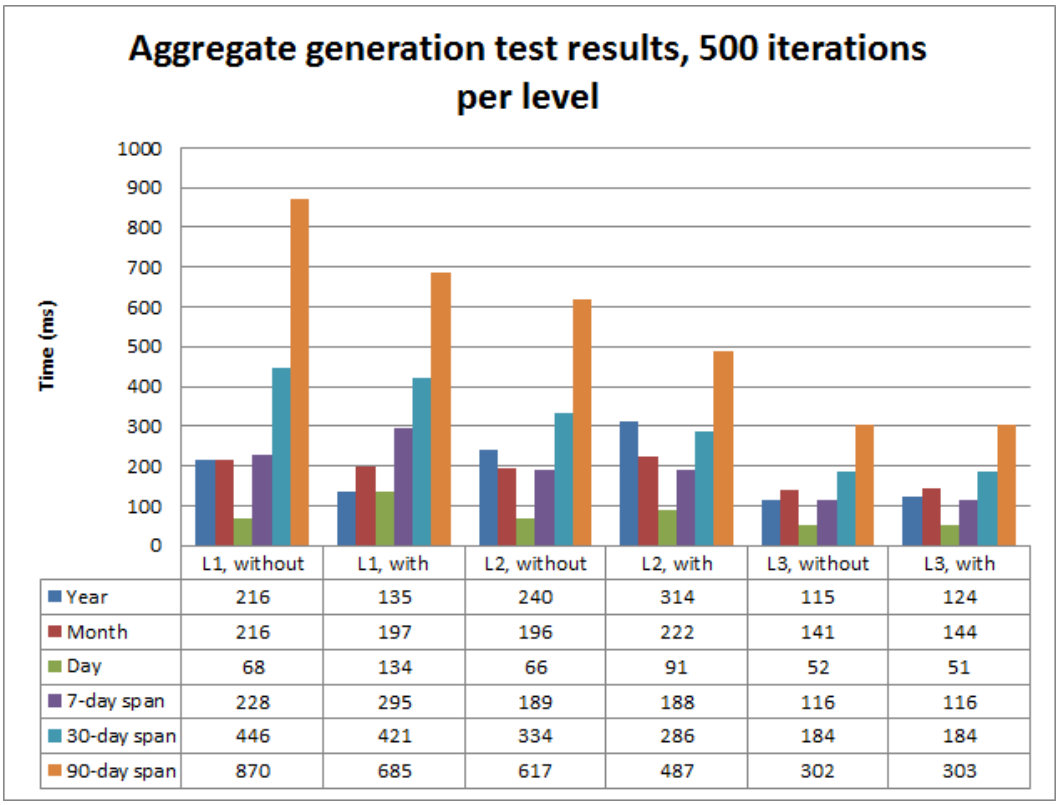
Sum. Aggregates the sum of all children values. Used to calculate the average value.

Non-empty count. Aggregates the number of non-empty data points found. A regular count does not work because respondents do not always answer all questions, and thus different questions need different counts.

Distinct count. Aggregates the number of distinct items. In this case, the number of distinct **respondents**.

After these base measures and all the dimensions are chosen, calculated members can be added where the formulas that calculate the member are written by the developer. The only calculated member necessary in this prototype is the average, calculated as the **sum** divided by the **non-empty count**.[23]

By default, no pre-calculated aggregations are used. Default pre-calculated aggregations can be set for the measure groups to let SSAS choose the optimal aggregates to pre-calculate based on the data set. By testing the solution both with and without these pre-calculated aggregates, a number of differences are noticed. The reader is referred to figure 5.8 for a chart over the performance difference. The test is run for all three levels, whereas a "level" refers to a level in the organization hierarchies (abbreviated "Ln" where n is the level number). 500 different date/organization sets each level, and the time required averaged. Averaging these averages, it turns out that the average time required is 255 ms without the pre-calculated aggregates and 243 ms with them - a performance increase of as little as 4.9%. This is not necessarily because pre-calculated aggregates are not efficient, but rather that the default aggregates are not fit to the way the cube is used. In addition to the performance difference, the cube takes 1866 MB as opposed to 759 MB without the pre-calculated aggregates, and takes 11:06 minutes instead of 07:08 minutes to process. Because of the minimal performance gain (<5%) and costs in space and processing, no pre-calculated aggregates were used. The methods of testing and



	L1, without	L1, with	L2, without	L2, with	L3, without	L3, with
Year	216	135	240	314	115	124
Month	216	197	196	222	141	144
Day	68	134	66	91	52	51
7-day span	228	295	189	188	116	116
30-day span	446	421	334	286	184	184
90-day span	870	685	617	487	302	303

Figure 5.8. A comparison between using no pre-calculated aggregates and using default pre-calculated aggregates in SSAS .

analyzing are discussed in more detail in chapter 6, and pre-calculated aggregates are discussed further in section 8.6.

5.3 Interface

5.3.1 User Interface

A number of available user interfaces can be connected to a SSAS cube solution, partly because it uses the Extensible Markup Language for Analysis (XMLA) standard including the Multidimensional Expressions (MDX) query language[12]. These include SQL Server Reporting Services (SSRS), PowerPivot and software from third parties. The main interface used in this solution was Microsoft Excel which has built in support for connecting and working with OLAP cube sources[24]. The way data from the cube can be accessed, what data is available and other features such as actions and drill through are already decided through the design of the cube. Hence the importance of having this in mind when designing dimensions, hierarchies and

measures in the cube.

The goal of this implementation has been to provide a way to group data belonging to a single respondent and list the given answers by question. This is achieved by using pivot tables in Excel. The respondent hierarchy is used for the rows, the question hierarchy is used for the columns and the data is filtered using the time dimension. Lastly the desired measures are added to the pivot table, being displayed for each cell that has a value. An example of the result can be seen in figures 5.9 and 5.10[3].

5.3.2 Application Interface

In order to allow other applications such as the web interface to access values in the cube in a convenient way, an Application Programming Interface (API) is needed. When browsing the cube through a Graphical User Interface (GUI) such as Excel, what actually happens is that Excel generates appropriate MDX statements, queries the OLAP server and presents the returned result. The used MDX queries can be identified by using the SQL Server Profiler[25]. Similarly, MDX queries can be used in applications to access the SSAS cube.

[3]The data presented in the examples have had confidential information changed.

	A	B	C	D	E	F	G
1	Year Month Date Time	2013					
2							
3	**Answer Average**	**Column Labels**					
4	**Row Labels**	**Generell nöjdhet**	**Engagemang**	**Kunskap**	**Svarstid**	**Första gången**	**Grand Total**
5	Industry 1	4,440556495	4,547886563	4,491510202	3,81687193	4,065479749	4,280154792
6	Industry 2	4,210443979	4,403363867	4,306116734	3,72211035	3,817034277	4,182154432
7	Industry 3	4,57557628	4,666725321	4,584198487	3,900404716	4,101410343	4,393465578
8	Industry 4	4,386672068	4,544865303	4,424448045	3,628705942	4,205014959	4,267640449
9	Industry 5	4,452325127	4,571254702	4,477044908	3,810731421	4,038898451	4,268960937
10	Industry 6	3,3	3,590909091	3,590909091	3		3,453125
11	Industry 7	4,347673054	4,456424254	4,414115812	3,598156296		4,214693472
12	Industry 8	4,546394384	4,622718571	4,603573708	4,25488194	4,356763735	4,479575057
13	Industry 9	4,417512013	4,52642819	4,43780032	3,957287774	4,195234944	4,311340664
14	Industry 10	4,724637681	4,729728295	4,665493953	4,211307137	3,785787193	4,350339924
15	**Grand Total**	**4,35032681**	**4,506066616**	**4,463717986**	**3,828604484**	**3,938039688**	**4,260256021**

Figure 5.9. An example of coarser aggregated cube values as viewed in Excel.

	A	B	C	D	E	F	G
1	Year Month Date Time	2013					
2							
3	**Answer Average**	**Column Labels**					
4	**Row Labels**	**Generell nöjdhet**	**Engagemang**	**Kunskap**	**Svarstid**	**Första gången**	**Grand Total**
5	**Industry 3**	**4,705035971**	**4,755395683**	**4,741007194**	**3,820143885**	**4,309352518**	**4,46618705**
6	**Company 15**	**4,705035971**	**4,755395683**	**4,741007194**	**3,820143885**	**4,309352518**	**4,46618705**
7	Unit 8	4,705035971	4,755395683	4,741007194	3,820143885	4,309352518	4,46618705
8	**Group 3**	**4,705035971**	**4,755395683**	**4,741007194**	**3,820143885**	**4,309352518**	**4,46618705**
9	Agent 47	4,705035971	4,755395683	4,741007194	3,820143885	4,309352518	4,46618705
10	923457	5	5	5	1	5	4,2
11	923557	5	5	5	1	5	4,2
12	925160	5	5	5	4	5	4,8
13	931615	5	5	5	5	5	5
14	932120	5	5	5	5	5	5
15	932959	4	4	5	2	5	4
16	950834	3	5	5	4	5	4,4
17	951174	5	5	5	4	5	4,8
18	951794	5	4	5	5	5	4,8

Figure 5.10. An example of a detailed view of individual respondents as viewed in Excel.

Chapter 6

Analysis

The analysis phase of this degree project was divided into two parts; aggregates and correlational analysis. The reason for this distinction is that the calculation of aggregates is fully implemented and can be accessed using an API in both the current solution and the solution developed for this degree project, and thus it can be tested and compared programmatically. The correlational analysis on the other hand is in the current solution subject to a significant amount of manual processing and less so in the solution developed for this degree project. For that reason it is not feasible to test the difference programmatically, but the difference will rather be analyzed in discussion with Bright's analysts.

6.1 Aggregate analysis

In order to analyze the difference between the current solution and the solution implemented for this degree project in terms of aggregates, C# Test Units were used. The tests attempts to answer the questions of the implemented solution validity (i.e. if the generated answers are correct) and performance (i.e. how much faster or slower the solution is) compared to the current solution.

The tests are adapted to provide a meaningful comparison between the current solution and the developed prototype. The web interface allows the user to select a company, unit, group or agent in a hierarchy as well as a time span ranging over the previous and current year. The web interface then provides the user with the answer average from respondents on a set of factors for the chosen entity company, unit, group or agent as well as a comparison to the corresponding upper level in the hierarchy. For example, choosing a **group** would give a comparison to the corresponding **unit**. The tests of both the current solution and prototype provides the same comparison.

Measurements are taken for a random time interval in each of six categories; A

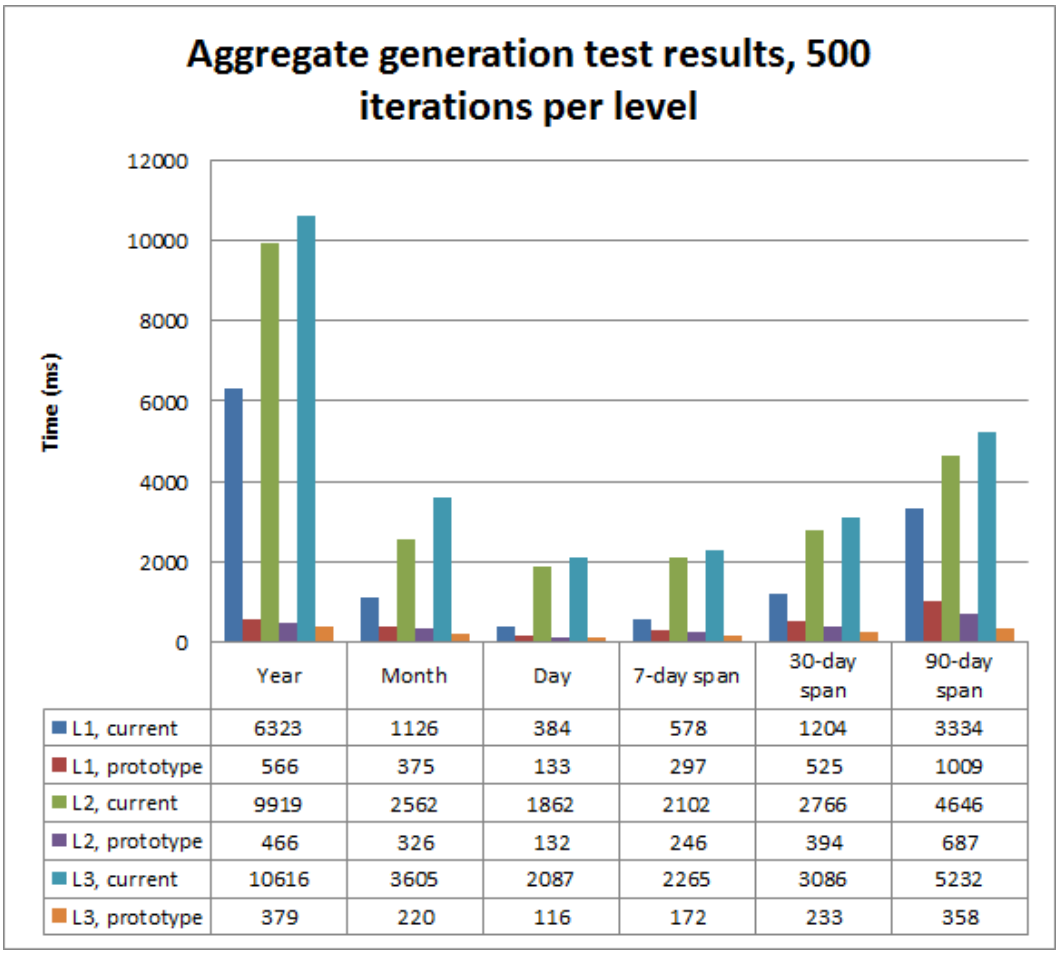
full year (such as 2012), month (such as May 2012), day (such as 2012-07-13) as well as randomly chosen time spans of 7, 30 and 90 days respectively. The reason for having both full years, months and days as well as random time spans is to analyze how the prototype performance differs from random spans to possibly pre-aggregates values in the defined hierarchies. While a full month such as $2012-06-01 \rightarrow 2012-06-31$ is reasonably performance-wise equal to $2012-06-08 \rightarrow 2012-07-07$ in the current solution, the prototype might exhibit different performance.

The tests are conducted for three separate levels in the hierarchy. The first test compares level 1 and 2; company and unit, referred to as "L1". The second test compares level 2 and 3; unit and group, referred to as "L2". The third tests compares level 3 and 4; group and agent, referred to as "L3". The tests also compare the results of the queries, ensuring validity of the answers given by the prototype. The tests are repeated 500 times each and the average is stated for both the current and prototype solution. The results of these tests can be viewed in figure 6.1.

It should be noted that while it was attempted to make the tests as similar and fair as possible between the current solution and the prototype, they are very different and some problems were encountered. The current solution stores additional interface-specific information on which factors to include or exclude from showing in the web interface as well as which companies, units or groups to include. It only retrieves data for the factors which are actually shown. To make the two tests more similar, the test program for the prototype queried the transactional database for which companies, units and groups to show as well as which factors to show for them. However, there were occasions when the prototype did not receive the same number of factors from the transactional database as the current solution. As such only the factors that did match were compared.

According to these test results, the prototype retrieves the results 9.6 times faster. However, it should be noted that these are fully random tests and may not accurately reflect how the real-world solution usage. Looking at the individual test levels, L1, L2 and L3 are 4.46, 10.6 and 18.18 times faster respectively. An interesting trend that can be noted in figure 6.1 is that the current solution is faster in L1 than L3 while the prototype is faster in L3 than L1. The reason the current solution is faster at a higher level despite more data is not certain, but a reasonable explanation can be how the indexes are set up as well as that querying a lower level is a more complex query. The prototype is faster at the lower levels simply because there is less data to process.

The total number of distinct respondents is also important and included in the analysis, but the test does not assert that the results from the current solution equals those of the prototype. This is because there were problems when using the interface-specific settings described above together with the prototype. At times a different set of factors are retrieved by the prototype, resulting in an incorrect



	Year	Month	Day	7-day span	30-day span	90-day span
L1, current	6323	1126	384	578	1204	3334
L1, prototype	566	375	133	297	525	1009
L2, current	9919	2562	1862	2102	2766	4646
L2, prototype	466	326	132	246	394	687
L3, current	10616	3605	2087	2265	3086	5232
L3, prototype	379	220	116	172	233	358

Figure 6.1. Test of average factor retrieval times comparing current solution and prototype for all three test levels.

distinct respondent count. Despite of this, in the random tests conducted, the correct number of distinct respondents was retrieved in 96% of the tests.

Furthermore, because it is possible for rounding errors to occur during the different processing stages, factor accuracy is only asserted within a 10^{-9} error margin. During the tests, all compared factor results were consistent between the current solution and the prototype.

6.2 Correlational analysis

In the current solution, as discussed in chapter 2, there is no reasonable way to extract information in a format that allows correlational analysis. This has restricted

analysis to be made primarily on aggregates which proves less valuable than if it is done on individual data. The prototype opens up possibilities to generate reports based on correlational analysis in a flexible and convenient way by presenting the information in a useful format in Excel (see figure 5.10).

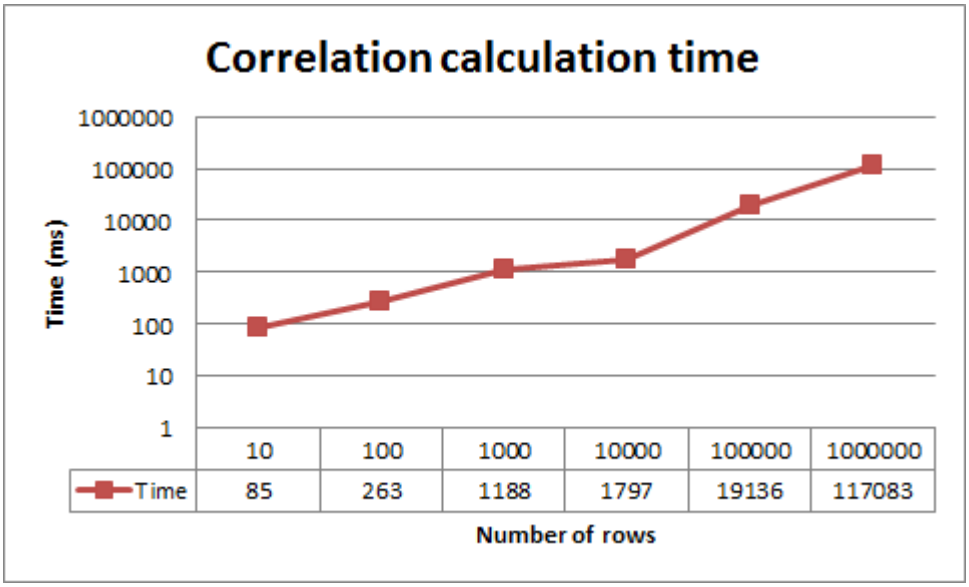


Figure 6.2. Chart over the correlation calculation time for different number of rows. Note that both axises are exponential. The retrieved number of rows are not exactly as stated, but chosen to be as close as possible.

Once data is retrieved in a usable format in Excel, adding the formulas for analysis calculations is simple and straightforward for users experienced with Excel. Depending on the number of rows involved in the calculations, retrieval can take significant time. In order to assess the performance of correlational analysis, a typical test case was set up and tested for different number of rows. Each row in the test is an individual respondent. The test case aims to calculate the correlation between two often used factors. For retrieving a smaller number of rows, a short time span for a single agent, group or unit is used. For a large number of rows, a long time span for an industry or a company is used. Refer to figure 6.2 for the test results.

Most of the correlational analysis done by Bright's analysts is focused on specific groups in organizations and specific time spans, and thus the number of rows being used are generally not excessively high. In these cases, correlational analysis is highly responsive with times below 2 seconds. In the rare cases that the number of rows used exceeds 10,000, the calculation delays will be noticeable but well within acceptable boundaries. As of Excel 2010, correlational analysis is limited to $2^{20} =$

1,048,576 rows[26].

6.3 ETL operations and cube processing

The total time for processing the prototype includes the stages of clearing the dimensional database tables, populating the three dimensions dimRespondent, dimQuestion and dimTime as well as the fact table factAnswers. Finally, the cube is processed. The total time for this varies somewhat depending on when it is run, and will naturally change (increase) in the future as more data is added. In the current situation it takes approximately 14:47 minutes. How this time is divided among the different stages can be viewed in figure 6.3.

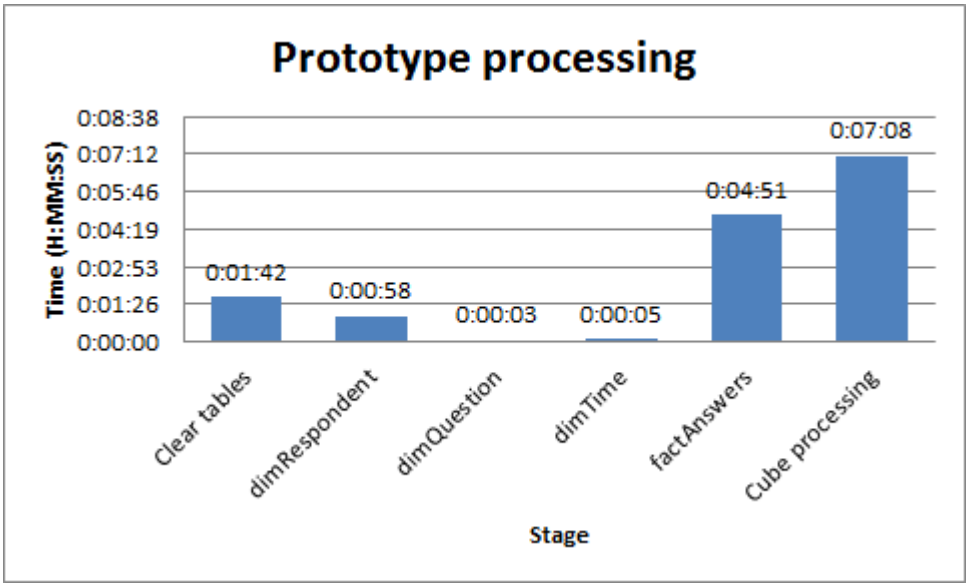


Figure 6.3. Chart over how prototype processing times are divided.

6.4 Storage size

Using a dimensional database as well as a data cube in excess of the in-place transactional database requires additional storage space. This section will compare the prototype to the solution in terms of required storage space. For the current solution only the tables specifically used for analysis are measured because both solutions are equally dependent on the source data.

The current solution uses two tables to speed up analysis; **Answers_Fact** and **Benchmark_Fact**. The first table includes detailed information on all answers in a format easily used for analysis (much redundancy), and the second provides ready

numbers for different factors for all industries. These tables take up 15,091,816 KB and 600 KB respectively, adding up to 15,092,416 KB or approximately 14.39 GB. Most of this space, approximately two thirds, are indexes used for speeding up queries.

The prototype uses both the dimensional database and a data cube for analysis. The dimensional database is made up of four tables; the three dimensions dimRespondent, dimQuestion and dimTime as well as the fact table factAnswers. These tables take up 459,408 KB, 240 KB, 9,232 KB and 654,992 KB respectively. The data cube takes up approximately 759.92 MB. In total, this adds up to approximately 1.81 GB.

Chapter 7

Conclusions

The developed prototype consistently delivers correct results in an efficient manner. According to the tests in chapter 6, analysis performance is improved severalfold. It does so using significantly less storage space than the current solution, and is kept up-to-date using nightly procedures well within acceptable boundaries in terms of processing times. Because all analysis is done on the data warehouse solution, customer use of the transactional database is not adversely affected beyond the nightly procedures.

In addition to improving performance, the prototype allows for flexible data extraction making new kinds of analysis possible. This allows Bright to offer new services to their customers that they do desire, but have previously been restricted from implementing because the current solution was never intended for such flexibility.

Looking back at the choice of method made in chapter 4, using OLAP software specialized in data analysis and choosing SSAS over one of the alternative tools such as Palo, Jedox or Oracle Essbase turned out to be a good choice leading to a well-functional result that was possible to develop within the given time frame. Because there has not been an equal amount of time spent on each of the candidate tools we cannot determine if we made the optimal choice, but only that it sufficed.

One of the major features of an OLAP cube that made us turn to such a solution was the possibility of having pre-calculated aggregates that could speed up analysis significantly. As it turns out, we ended up not implementing pre-calculated aggregates in the prototype as the performance gains were minimal as discussed in section 5.2.2. The structure of the cube make analysis efficient enough even without those pre-calculated aggregates.

The prototype is only intended as a prototype and as such lacks some of the features needed for it to be implemented in a production environment, as discussed in section 1.3. Possible future developments of the prototype are discussed in chapter 8 dedicated to this subject.

Chapter 8

Recommendations

This chapter will discuss possible future developments for the solution, organized into separate sections.

8.1 Selective ETL

In the current solution the ETL script simply clears the dimensional database and re-copies everything from the OLTP. This is a simple solution that works well for now considering that the total ETL time is less than 10 minutes and is done nightly. However, if the database continues to grow rapidly the ETL time may become unacceptably long. Furthermore, the current dimensional database is implemented as its own schema in the OLTP database allowing high transfer speeds. In case the dimensional database was to be moved to a different server in a different location, this means that network transfer speeds could bottleneck the ETL operations resulting in more time needed to perform the nightly ETL.

It is recommended that a selective ETL solution is developed to only copy the changed data if these conditions are to change. This would significantly lower the time required for ETL and ensure that it is not overwhelmed by future database sizes and slower network connectivity.

8.2 Real-time synchronization

In the current solution the ETL script is only run nightly, and section 8.1 suggests a more optimal means to the same end. A different approach would be to have the OLTP database update the dimensional database every time a change is made, allowing for real-time analysis. This approach, however, has implications for the cube as different sections would have to be reprocessed as well. There are a number of approaches to address this issue.

- It is possible to use an incremental process in SSAS to only process changed values. This approach may introduce significant processing overhead.

- Another alternative is to use the ROLAP storage mode in SSAS for real-time analysis. ROLAP stands for Relational Online Analytical Processing, and in short means that both atomic data and aggregations are stored in the relational database. This approach would allow for real-time analysis, but may also decrease performance significantly because the relational database is not optimized for large queries.

- The cube is currently using the MOLAP storage mode. MOLAP stands for Multidimensional Online Analytical Processing and means that both atomic data and aggregations are stored in a highly optimized format in the cube. It is possible to use this storage mode for historical data and corresponding queries, and use the ROLAP storage mode for a small partition for real-time queries.

The third option is most likely to be the optimal solution, but more research around the are is required to make an informed decision on which approach to choose.

8.3 Redundant questions

Because the questions in the Bright database are indirectly administrated by many third parties, and there is a common need to ask approximately the same questions (e.g. "Are you generally satisfied with our customer support?"), it has come to being many identical or similar questions in the transactional database. While this doesn't have any direct consequences, it could certainly complicate analysis as answers to "basically the same question" are scattered across several questions. This could be improved by implementing fuzzy logic in the ETL solution to remove such redundancy.

This may however not be a relevant problem. Multiple similar questions are most likely part of the same factor on which the analysis could equally well be made. Implementing fuzzy logic to help reduce this redundancy could also present new problems if the fuzzy logic makes mistakes.

8.4 Web interface using the cube

As of now the web interface still uses the transactional database to generate the aggregate data that's presented to the users. The web interface could use the cube instead to reap the benefits of increased performance as well as not disrupt transactions done on the transactional database.

Because the web interface has several interface-specific settings defining which information to show for specific companies, units and groups, the cube must be modified to fit those needs. One approach would be to tailor the cube to the needs of the web interface, but this could have negative implications in terms to analysis and increase cube complexity significantly. A preferable approach would be implement a layer between the web interface and the cube that handles interface-specific settings. This middle layer could simply query the transactional database where these settings are stored (such queries are trivial in comparison to calculating aggregates) and use them to extract only the necessary information from the cube using OLAP queries. By getting interface-specific information from the transactional database, changes in the interface (made by users) can be made in real-time even if the cube is not processed as such.

The main problem of this approach is that queries can be made for the current day, for which the OLAP cube does not have information. Because real-time information is important in the web interface, disabling this feature is not an option. There are two distinct approaches to this problem. The first approach is to update the cube in real-time as discussed in section 8.2 allowing the cube to fully replace the current solution. The second approach is to combine the cube with the current solution, retrieving historical data from the cube and the current day's data from the current solution.

8.5 Move data warehouse to specific database

The data warehouse is currently hosted on the same database as the transactional database, only under a different schema. In production, this could be hosted on a different server in order to completely separate the OLTP and OLAP solutions. This would be an interesting option if the analysis is using up considerable resources from the shared server, adversely affecting the use of the transactional database.

8.6 Optimize cube pre-calculated aggregates

The cube in SSAS is currently not using any pre-calculated aggregates. As discussed in chapter 6, the performance in the prototype is greatly increased as compared to the previous solution. However, the performance might be possible to increase significantly by optimizing the choices of which aggregates to include as well as the structure of these.

The optimal configuration of this is highly dependent on the usage of the cube and requires co-operation with the end users of the cube. If the necessary research on cube usage is made, informed decisions on which aggregates to include and how to partition them can be made and better performance achieved.

8.7 Server side correlational calculations

In the developed solution, it is easy to extract information in a highly flexible and efficient manner that is useful for correlational analysis. Selecting which rows to show in Excel with which columns allows the user (assumed familiar with Excel syntax) to simply use Excel syntax to calculate different things. However, with a huge amount of rows in the magnitudes of tens or hundreds of thousands, this becomes a poorly responsive approach because huge amounts of data needs to be transferred and stored in an Excel spreadsheet. This is an issue that can be addressed by removing the need to view the individual rows and instead do the calculations on the server side, delivering only the sought answer to the user.

One option to achieve this is to use the built-in data mining available in SSAS. This option would allow further automation of correlational analysis done completely on the server side.[27]

References

[1] *Gartner Says Worldwide Business Intelligence Software Revenue to Grow 7 Percent in 2013*, Rob van der Meulen and Janessa Rivera, 2013-02-19, http://www.gartner.com/newsroom/id/2340216, Retrieved 2013-04-15

[2] *Business Intelligence (BI)*, Gartner IT Glossary, http://www.gartner.com/it-glossary/business-intelligence-bi/, Retrieved 2013-04-18

[3] *Data Warehousing Concepts*, Oracle Documentation, http://docs.oracle.com/cd/B10500_01/server.920/a96520/concept.htm, Retrieved 2013-04-18

[4] *Data Warehouse*, Gartner IT Glossary, http://www.gartner.com/it-glossary/data-warehouse/, Retrieved 2013-04-18

[5] *What is big data?*, IBM, http://www-01.ibm.com/software/data/bigdata/, Retrieved 2013-04-18

[6] *What is Data Mining?*, Oracle Documentation, http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/process.htm, Retrieved 2013-04-15

[7] *SQL Server 2005 Analysis Services - Step by Step*, Reed Jacobson and Stacia Misner, 2006, Hitachi Consulting

[8] *Oracle Business Intelligence Tools and Technology*, Oracle, http://www.oracle.com/us/solutions/business-analytics/business-intelligence/overview/index.html, Retrieved 2013-05-01

[9] *Essbase Studio User Guide*, Oracle, 2012-08, http://docs.oracle.com/cd/E38445_01/doc.11122/est_user.pdf, Retrieved 2013-04-26

[10] *Oracle Hyperion Smart View for Office User Guide*, Oracle, 2013-03, http://docs.oracle.com/cd/E38438_01/epm.111223/sv_user.pdf, Retrieved 2013-04-26

[11] *Essbase MDX Queries*, Oracle Documentation, http://docs.oracle.com/cd/E10530_01/doc/epm.931/html_esb_dbag/frameset.htm?dmaxldml.htm, Retrieved 2013-05-21

[12] *SSAS 2008 R2 - Technical Reference*, Microsoft Documentation, http://technet.microsoft.com/en-us/library/bb500210(v=sql.105).aspx, Retrieved 2013-04-30

[13] *First Steps with Jedox for Excel version 5.0*, Jedox Documentation http://www.jedox.com/en/jedox-downloads/documentation/5/first-steps-with-jedox-for-excel-5-manual.html, Retrieved 2013-04-26

[14] *Jedox ETL Server Manual version 4.0*, Jedox Documentation, http://www.jedox.com/en/jedox-downloads/documentation/jedox-etl-server-4-manual.html, Retrieved 2013-04-26

[15] *First Steps with Jedox Web version 5.0*, Jedox Documentation http://www.jedox.com/en/jedox-downloads/documentation/5/first-steps-with-jedox-web-5-manual.html, Retrieved 2013-05-21

[16] *Jedox 3rd Party Software*, Jedox Documentation, http://www.jedox.com/en/jedox-downloads/documentation/5/jedox-3rd-party-software-5-manual.html, Retrieved 2013-05-21

[17] *Creating a PowerPivot Workbook on SharePoint*, Microsoft Documentation, http://technet.microsoft.com/en-us/library/gg413412.aspx, Retrieved 2013-05-22

[18] *PowerPivot capacity specification*, Microsoft Documentation, http://technet.microsoft.com/en-us/library/gg413465.aspx, Retrieved 2013-05-22

[19] *Software boundaries and limits for SharePoint 2013*, Microsoft Documentation, http://technet.microsoft.com/en-us/library/cc262787.aspx, Retrieved 2013-05-22

[20] *PowerPivot Features*, Microsoft Documentation, http://msdn.microsoft.com/en-us/library/ee210639(v=sql.105).aspx, Retrieved 2013-05-22

[21] *Configure Attribute Relationship Properties*, Microsoft Documentation, http://msdn.microsoft.com/en-us/library/ms176124.aspx, Retrieved 2013-04-29

[22] *A summary of the international standard date and time notation*, Markus Kuhn, http://www.cl.cam.ac.uk/~mgk25/iso-time.html, Retrieved 2013-04-29

[23] *Defining Calculated Members*, Microsoft Documentation, http://msdn.microsoft.com/en-us/library/ms166568.aspx, Retrieved 2013-04-30

[24] *Microsoft Office OLAP overview*, Microsoft Documentation, http://office.microsoft.com/en-001/excel-help/overview-of-online-analytical-processing-olap-HP010177437.aspx, Retrieved 2013-04-30

[25] *Monitoring Analysis Services with SQL Server Profiler*, Microsoft Documentation, http://msdn.microsoft.com/en-us/library/ms174779.aspx, Retrieved 2013-05-03

[26] *Excel specifications and limits*, Microsoft Documentation, http://office.microsoft.com/en-us/excel-help/excel-specifications-and-limits-HP010073849.aspx, Retrieved 2013-04-15

[27] *Data Mining in SSAS*, Microsoft Documentation, http://msdn.microsoft.com/en-us/library/bb510516.aspx, Retrieved 2013-05-14