

Designing and implementing a data warehouse for internal reporting in Sievo

Markus Koivuniemi



Tekijä Markus Koivuniemi	
Koulutusohjelma Tietojenkäsittelyn koulutusohjelma	
Opinnäytetyön otsikko Tietovaraston suunnittelu ja toteutus sisäiselle raportoinnille Sievossa	Sivu- ja liitesivumäärä 41
Opinnäytetyön otsikko englanniksi Designing and implementing a data warehouse for internal reporting in Sievo	
<p>Opinnäytetyö tehtiin SaaS (ohjelmiston myynti palveluna) yritys Sievolle, joka on erikoistunut yritysten kulujen ja hankintatoimen analysointiin sekä hallinnoimiseen. Sievossa käytetään "Business Intelligence" – raportointia sisäisten prosessien seuraamiseen, mutta ajan kuluessa sen kasvattamisesta ja hallinnoimisesta on tullut työlästä. Tämä johtuu tietokantarakenteesta, joka toimii raportoinnin lähteenä. Tietovarasto toteutettiin edellä mainittujen puutteiden ratkaisemiseksi ja siitä tulee uusi lähde raportoinnille.</p> <p>Projekti alkoi tutustumisella teoriataustaan, johon kuului Business Intelligence, tietovarastointi ja ETL – prosessi. Sitä seurasi kahden rakenteeltaan erilaisen testiversion luominen, joita testattiin samalla datalla. Testaamisen jälkeen päätettiin, kumpaa rakennetta lähdetään kehittämään. Lopulliselle versiolle suunniteltiin ja tehtiin ensin relaatiomalli Microsoft Visiolla, joka sisälsi tietovarastoon tulevat taulut, kolumnit ja datatyypit. Relaatiomallia seurasi tietovaraston luonti tietokantaan ja tietovaraston testaus paljon suuremmalla ja vaihtelevammalla datalla kuin aiempien versioiden. Lopuksi tietovarasto indexoitiin.</p> <p>Lopputuloksena syntyi käyttövalmis tietovarasto Sievon tietokantaan. Tietovarasto on suunniteltu tukemaan erilaista ja vaihtelevaa dataa sekä tarjoamaan joustavuutta raportointiin. Tietovaraston SQL-kyselyjen suorituskyky on pyritty optimoimaan aina rakenteen salliessa.</p> <p>Tietovarastoa ei tulla ottamaan heti käyttöön Sievossa, koska se vaatii ETL-prosessien rakentamisen jatkuvaan käyttöön sekä itse raporttien teko Microsoft Reporting Service:llä. Näiden rakentaminen ei kuulunut osaksi tätä projektia.</p>	
Asiasanat Tietovarasto, tietokanta, Business Intelligence, ETL	

Author Markus Koivuniemi	
Degree programme Business Information Technology	
Report/thesis title Designing and implementing a data warehouse for internal reporting in Sievo	Number of pages and appendix pages 41
<p>The thesis was done for a Software as a Service company Sievo that specialises in spend and procurement analysis and management. Sievo uses a business intelligence report to track internal processes but over time it has become a burden to expand and maintain. The issue lies in the database structure that is used as the source to generate the report. The data warehouse was developed to tackle these problems and to replace the old database structure.</p> <p>The project was started off with familiarizing with the related theory background that included Business Intelligence, ETL process and Data Warehousing concepts. Then two different test data warehouse structures were created and tested with same set of data. After testing was done it was decided which version was to undergo further development. For the final version a relational database model was done in Microsoft Visio that included all the tables, columns and data types to be used in the data warehouse. Then the data warehouse was implemented to the database and tested with different kinds of data sets. Finally indexing was created to the data warehouse.</p> <p>As the final result there is a ready to use data warehouse created in Sievo's database. The data warehouse was designed to support different kinds of data and to offer better flexibility. Query performance was also optimized whenever it did not interfere with the wanted structure.</p> <p>The data warehouse will not be taken into use instantly as it still requires ETL processes to be built for a continuous use. Building these processes was not in the scope of this project. The actual reports have to be made with the Microsoft Reporting Services as well.</p>	
Keywords Data Warehouse, database, Business Intelligence, ETL	

Table of contents

1	Introduction	3
2	Business Intelligence.....	4
2.1	BI Goals.....	5
2.2	Business Intelligence architecture	6
3	ETL process.....	8
3.1	Extract (E).....	8
3.2	Transform (T).....	9
3.3	Load (L)	10
4	What is a data warehouse and why does it exist	11
4.1	Data warehouse schemas.....	12
4.1.1	Fact and dimension tables	12
4.1.2	Hierarchies.....	13
4.1.3	Star schema.....	14
4.1.4	Snowflake schema.....	15
4.1.5	Star and snowflake schema comparison	16
4.2	Granularity	17
4.3	Database indexing	17
5	Project planning	19
5.1	Tools used in the project.....	19
5.2	Old reporting database setup and the requirements for the data warehouse.....	20
6	First test versions and testing.....	21
6.1	First test version.....	22
6.2	Second test version	22
6.3	Testing and conclusions.....	24
7	Final version.....	27
7.1	Overview of the relational model	27
7.2	Dimension tables	28
7.2.1	Customer and User dimensions	29
7.2.2	Event type, event source and measure type dimensions.....	30
7.2.3	Extraction tool and load process dimensions.....	30
7.2.4	Time dimension.....	32
7.3	Fact tables: Event type, measure type and latest event type.....	33
7.4	Indexing	35
7.5	Testing of the final version	35
8	Conclusions and pondering.....	37
8.1	What was learned	37
9	Bibliography	39

Terms

A list of terms that are used in the text to assist reading:

- **Ad Hoc Query** is a query that is created to get information for a certain issue. Ad hoc means “for this purpose” in Latin and in general it refers to anything that is designed to return to specific problem
- **Aggregate** is a function where multiple rows are grouped to form a set of data in database management
- **Business Intelligence** refers to a software or group of softwares, solutions and processes that are used to turn raw data into information used to help business decisions
- **Column** is a set of values that are of same type for all the rows in a same table
- **Database** is a data collection that is organized so that it is easily and quickly accessed with a computer program
- **Data type** is a label for a column to describe what kind of data it holds
- **Data warehouse** is a repository that is commonly used in business intelligence data analysing and reporting
- **Duplication** means a copy of something. In this thesis duplication refers to a table having two identical rows
- **ERP** (Enterprise Resource Planning) means a software that is used to manage business processes
- **ETL** (Extract, Transform, Load) is a process where data is first taken from a source and then put to a destination through a transform process. A common process in data warehousing
- **Foreign key** has a value that contains the value of a primary key in another table
- **Granularity** in database means the level of detail the data is presented
- **Indexing** is a structure for the data so that it will ease the data retrieval and thus improve query performance
- **Normalization** is a process where data is organized to lessen or remove redundancy in database. This usually increases the number of tables. Denormalization is the opposite that aims to improve query performance
- **Null** represents data that doesn't exist in SQL. However it does not mean that a column with null value is empty
- **Populate** means adding data to a database
- **Primary key** is a value (or combination of values) that identifies unique rows in a relational database table

- **SQL** (Structure Query Language) is a standard language used to access relational databases. Queries are statements that have different functions in the database like return data or modify data
- **Table** is a set of data in database
- **Table joins** combine rows between two or more tables together over certain fields
- **Row** is a single record in a database table

1 Introduction

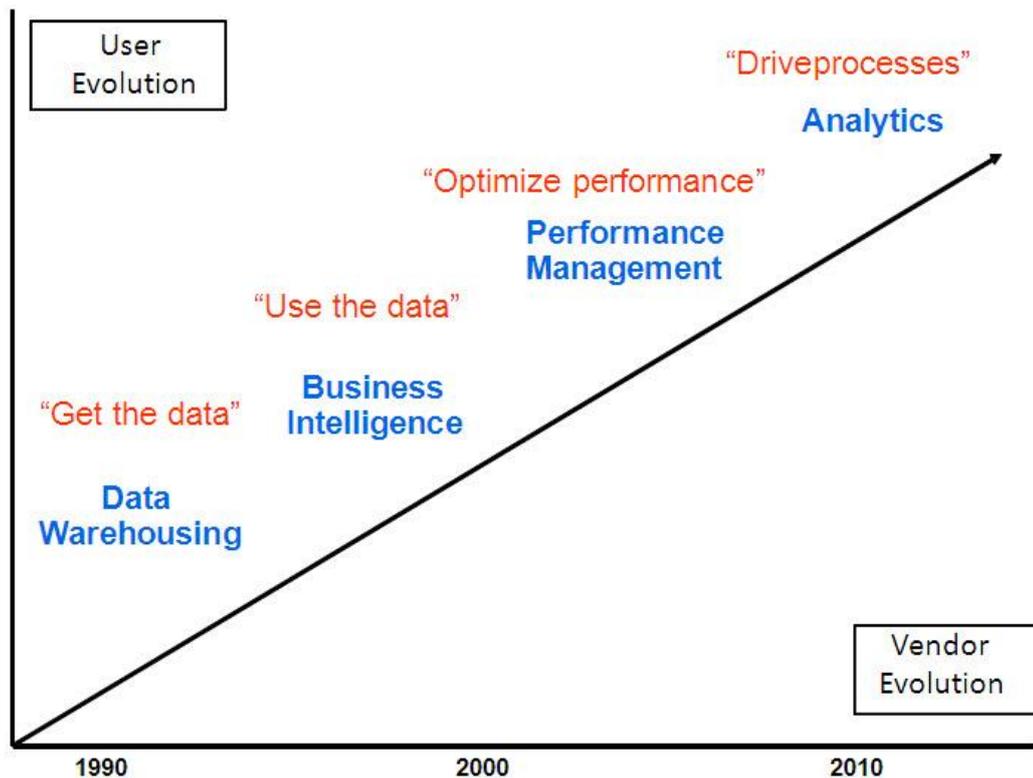
The goal in this thesis is to design and implement a data warehouse used as the source for a business intelligence report. The report gathers a lot of varying information to one place that is used by multiple different teams to track internal processes in a company called Sievo. This kind of report exists already in Sievo but over time it has become a burden to manage and maintain it. The bottleneck of the current report is the database structure behind the report. The main target in this thesis is to develop a more flexible and more expandable data warehouse to replace the current database.

The thesis is done for a SaaS (Software as a Service) company Sievo that specialises in spend and procurement management. Their software aims to help their customers in identifying savings, analysing past purchases, foresee future spend and assist pushing savings ideas into measurable profit. Clients of Sievo are usually larger companies that vary across dozens of industries and come from around the world.

This thesis first introduces the theory behind data warehousing and business intelligence. After that there will be discussions about the requirements for the structure but also learnings from the current reporting process on a technical level. Two different test versions were generated and both versions were tested with same set of data in both. The pros and cons of both test versions were analysed and another version developed further. Only the final version will be described in a much more precise detail than the test versions. The final version was also tested with much more varying data than the test versions. Finally the conclusions are summarized and the different learnings are gone through. Generating the report itself or creation processes for continuous data loading won't be in the scope of this thesis. The author has learned more about the database and SQL world and specifically widened his knowledge in these areas. In practice this means learning different methods to achieve similar results.

2 Business Intelligence

It is no longer a problem for companies to store massive amounts of data but using it effectively and getting the most out of it is a problem. That is where Business Intelligence steps in. In Business Intelligence (BI) a set of operational raw data is combined and converted into useful information that is then, through analysing, turned into knowledge. BI is used to support and speed up the decision making inside enterprises. Business Intelligence has seen an explosive growth in the available services and products but also in adoption of these technologies in the past two decades as seen in the picture 1. This is caused by the development happening on both sides. The suppliers have better offerings and the users are getting better at using the BI products so users also want better products to fulfil their needs. (Chaudhuri, Dayal, Narasayya 2011, 88). *“Even during the recent financial struggles, the demand for BI applications keeps on growing when the demand for other Information Technology products is soft”* (Negash 2004, 77).



Picture 1. The recent evolution of BI (Eckerson 2014)

The modern BI can be considered to have been born in the late 1980s where businesses used to generate run-the-business order entries, manufacturing resource planning and accounting solutions. In the early 1990s these were put into Enterprise Resource Planning (ERP) applications and data warehouses. In the early 2000s two new obstacles triggered the rapid development of business intelligence. Firstly the growing importance for new and

refreshed reports as often as possible (daily) put a lot of pressure on the IT to provide these reports. Secondly the rising popularity of Web meant that the “customer-facing” solutions had to operate around the clock. A great example about this is social media. From there on the data amounts and technology and applications have grown really fast with multiple different solutions that get deeper and deeper into analysing the data. (Kemochan 2014)

The purpose of business Intelligence (BI) solutions is pretty much the same as the purpose of “military intelligence”: to give commanders at every stage of a campaign a good view of the battlefield and the pros and cons of their options. In the business context, this means telling the business on an ongoing basis the state of the business (production, sales, profits, losses) and the factors that affect success (success of a sales campaign, customer satisfaction, manufacturing efficiency, planning and budgeting). (Kemochan 2014)

2.1 BI Goals

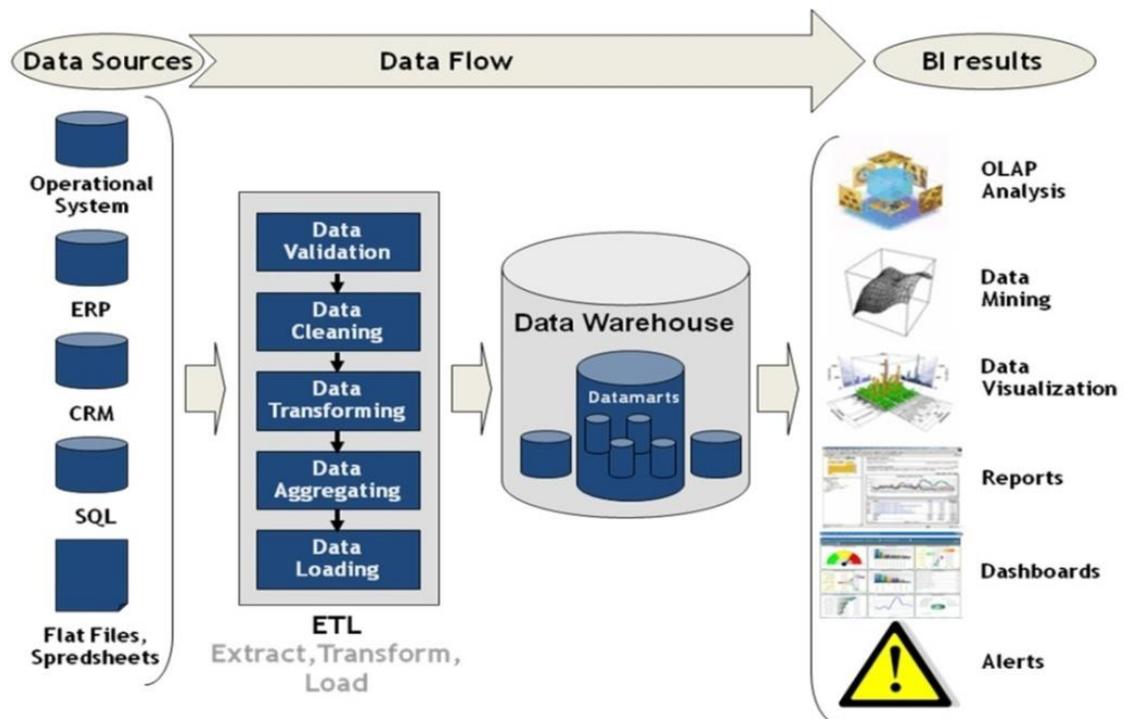
The biggest goals for BI are accelerating and improving decision making, optimization of the internal business processes and increasing operational efficiency. It can also help to drive new revenues, gain advantage over a competitor or spot problems within the business. With some BI tools future market trends can also be identified. (Rouse 2015)

The goals also differ between different end users and the size of the company. Very small companies with less than 100 employees are usually more interested in increasing their revenue and more likely to target customers with their BI deployments. Bigger companies on the other hand are looking to better their operational efficiency because in their scale small improvements there can lead to bigger savings in the long term. Similar differences can also be found on individual level within the company. The CEO is going to be interested in different kinds of analysis and reports than a middle management or a line manager. (Lachlan 2013)

Lastly the whole BI process should generally work in a way that the end users do not need to know what goes on “under the hood” of their analysing and reporting applications. This means that the structure of the data warehouse for example doesn’t matter for the end users as long as their queries and reports are working and loading fast. This doesn’t mean that their needs shouldn’t be taken into account when building the whole BI process but the technical side of how the process is build is usually none of their concern. (Rouse 2015)

2.2 Business Intelligence architecture

In general, BI refers to the processes, technologies and tools that are used to get useful information out of raw data. This information is then used for business analysis. Data for BI use is always taken from some kind of source. The most common source, especially when it comes to traditional sets of data, is a database that is behind an ERP system. ERP is a business management software or a set of softwares that is used to track, manage and interpret data from different sorts of business activities like inventory management, service delivery, sales or product costs. The database behind an ERP can be anything for example Microsoft SQL server, Oracle or IBMs DB2 which are the three most common ones. Outside of different databases the data source can also be a file such as CSV or a Microsoft Excel file. CSV stands for Comma Separated Value which is a text file or. These are more common with much smaller amounts of data. It is also possible to get the data out of a cloud system. (Rouse 2014)



Picture 2. An example of a BI architecture and the BI process (Super Develop 2014)

The data is normally loaded into the database through some kind of process. One common example is an ETL process that will be described in detail later on. When data is in the database which, in many BI cases is a data warehouse, the data can be already analysed with different kinds of ad hoc queries. This means that in all cases there isn't need for BI reporting applications. This is normally only done by the more IT minded professionals or data scientists. For the higher management and other BI end users there is usually one or multiple different 3rd party applications and tools. These applications can be

used for analysing and especially they are most of the time required for the reporting. Some examples of these are data mining where the goal is to gather previously unknown patterns and information from the data that will most likely have use in the future, OLAP (Online Analytical Processing) that tries to make looking at a set data from different angles easier for the user or data warehousing that is aimed for analysing data and to report it. An example of a chain from source to BI report can be seen in picture 2. (Chaudhuri, Dayal, Narasayya, 2011)

3 ETL process

ETL stands for Extract (E), Transform (T) and Load (L). In ETL the first thing to do is to take data out of some system, which means extracting the data. Then the data or set of data is modified to meet the BI requirements, so the data is transformed. Finally the data is put into the destination database for BI reporting and for the BI users to query, so the data is loaded to the database. The ETL in most cases is an automated process which means that keeping the database up to date requires no updating or actions from the end users. (Kimball, Ross & Thornthwaite, 2011, 369)

When building a Data Warehouse and Business Intelligence environment the ETL is estimated to take about 70 percent of the time spent on the DW/BI project. This is caused by the large amount of outer expectations and constraints. Few examples of the outer constraints: BI requirements, budget, timeframe, source data systems, skillset of the staff working on ETL. (Kimball 2011, 369)

To explain the constraints a little further. For example, many organizations have multiple data sources that are distributed globally. The data quality in these systems may not be identical and the business needs and pre-set requirements for the ETL can't be met. This could be discovered when exploring the data in the data source or after the data has been extracted and validation is being done to the data. This will force the people working on the project back to the drawing board to figure out a solution to the situation. (Kimball 2011, 371-374)

In this thesis the weight is not on working on the ETL. This is because the data is coming straight from Sievo's own database. This means that the extraction is very easy and the amount of data we are handling per extraction is on the smaller side in the ETL world. Lastly the data doesn't need much cleaning or modifying and we use the data in the same format as it is already in the database. Some kind of simple load processes need to be created to be able to test the structure of the database. These load processes won't be intended for continuous use.

3.1 Extract (E)

The main purpose for the extract process is to collect all the necessary data and to be sure that enough data is being collected. The extraction is always done to adapt to the source system. The extraction should also be created in a way that it does not interfere with the source system. This means that the extraction should use as little of the source

systems resources as possible. It should not affect the source system's respond times or cause locking of any kind. (Dataintegration.info)

When planning the extraction it should also be decided what will be the frequency that the data is extracted from the source system. The extraction can be a full extraction where all the available data is taken or then it can be restricted over something. For example, it could be set for a certain time period like a week, month or a year. The amount of extracted data is usually a good indicator on how the extraction should be set up. With smaller data sets extracting everything won't be an issue. (Dataintegration.info)

3.2 Transform (T)

In transform step the transactional data is turned into analytical data. The data can be run against multiple conditions and rules, datatypes can be changed and additional information can be added to the data such as NULLs to zeroes or splitting null columns not to be loaded or not to include in calculations. In transformation the data can also be combined from multiple data sets to just one or one column can be split into multiple columns or sets of data and vice versa. The bigger the data masses are the more thought and time should be put into creating the transform process. Otherwise the runtimes for transformation can shoot up exponentially. (Dataintegration.info)

In the transform phase the data can also be aggregated which is quite common. Aggregation in its most simple form would be the "group by" SQL query. For example, counting the sum of money each individual customer has spent for a certain company in 2014 instead of having that information spread over 20 individual purchases. This would result in one row for a certain customer instead of 20. Another example of simple aggregation would be to count the average amount the former mentioned customer has spent in 2014. The main advantage gained from aggregating data is big improvements on the query performance. (Wan, 2007)

"The single most dramatic way to affect performance in a large data warehouse is to provide a proper set of aggregate (summary) records that coexist with the primary base records. Aggregates can have a very significant effect on performance, in some cases speeding queries by a factor of one hundred or even one thousand. No other means exist to harvest such spectacular gains". (Kimball 1996)

It is common for ETL-process to use "staging area" between Extract and the actual transform phase. In staging the data is first loaded to staging tables in the database after being extracted. There are advantages for using the staging area: In extractions that take a long

time it is possible for the connection to disconnect in the middle of an ETL process or maybe data that isn't extracted at the same time from another source needs to be joined in the data. The staging area usually has the same column names and structure as the source and nothing is actually done to the data before it is loaded to staging area. The real transform phase happens between the staging tables and the destination tables in the data warehouse. (Oracle 2002, 12)

3.3 Load (L)

In the loading phase of ETL the data is loaded to its destination, which usually is a database. In loading it is important to pay attention so that the data is correctly loaded to the destination. Same as in the transform process, the load process should also be performed with as little resources as possible, especially as the data amounts get bigger and bigger. (Dataintegration.info, 2015)

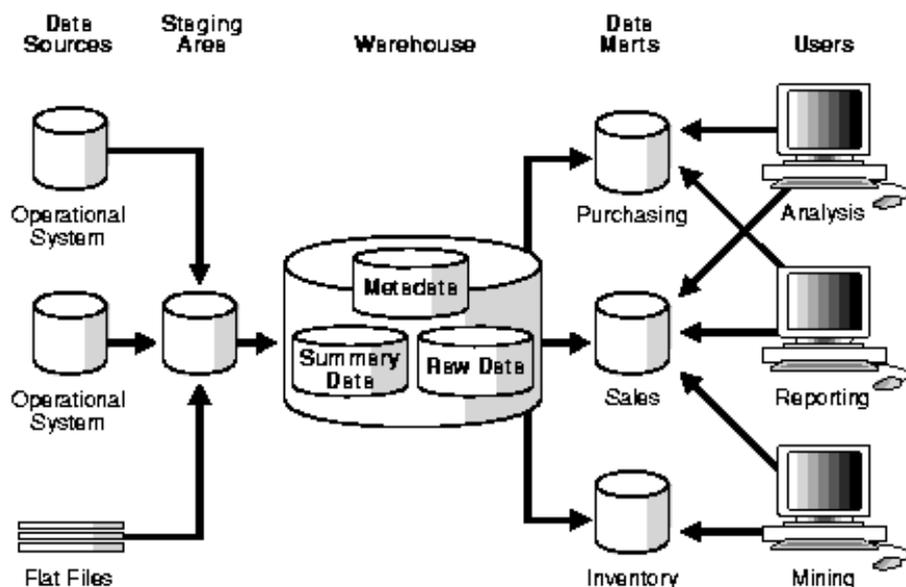
There are multiple ways for how to write the data to the destination in loading. One way is to delete all existing data and then insert all the data available or simply just add all the rows that follow the transformation. The latter is good if it is ensured that the data isn't duplicating in the extraction phase. The former usually works with really small data sets where all available data is extracted every time. Other way is to set delta keys that identify each row from one another and then either updating rows according to matching delta keys, inserting new rows that don't have matching delta keys or a combination of both of them. There is no right way to do the data load and it is usually decided uniquely for each case though in a perfect world append would be used every time. It should be noted though that different methods use resources in different amounts. (Kimball, Ross, Thornthwaite, 2011, 389)

4 What is a data warehouse and why does it exist

Data warehouse is a repository that BI reports and analysis tools use as a source. This means that it is usually a relational database that is used for querying instead of processing transactions. Data warehouses are designed to support this kind of ad hoc querying and the schemas are in general built to make this faster. (Oracle 2002, 1)

A lot of data warehouses end up storing big amounts of data. That is mostly due to the fact that data analysing simply requires large amounts of data. The amount of data usually depends on how far back historical data is stored in the data warehouse, how many data sources the data warehouse has and what are the requirements of the reporting for end users. The more needs the end users have the more data could be required, or if the needs of the end users differ a lot. Data warehouse also usually keeps growing as more recent data is loaded and updated to the warehouse through ETL-processes. (Oracle 2002, 1)

Sometimes it is possible to modify the data warehouse architecture to contain different groups (picture 3). It is done by adding data marts to the data warehouse structure. Data marts goal is to meet the particular demands of a specific group of users within the organization. (Kimball 2011, 248)



Picture 3. Data warehouse and ETL structure with staging area and data marts (Oracle 2002, 1)

4.1 Data warehouse schemas

Schema refers to the structure of a database system. It logically stores definitions for tables, views, columns, indexes and synonyms and it also stores the relations between those. These are called database objects. Different users can have different kinds of schemas depending on the user permissions. Some object can for example be restricted for some users and allowed for some. (Oracle 2002, 2)

4.1.1 Fact and dimension tables

Fact tables consist of mostly numeric values and measurements that vary between entries and foreign keys that refer to the dimensions around the fact table. Usually the identifying key is also generated for every row in a fact table to separate the rows from each other. It can either be a combination of foreign keys or then a totally own id that isn't dependant on other keys. Example can be seen in picture 4.

Normalization is a technique to organize the table contents in relational database (like data warehouse). It aims to get rid of inaccuracy in the database. This is done via scattering data into smaller and less redundant tables. In this process are also created "primary keys" and "foreign keys". Primary keys are created to the new table and those primary keys refer to the foreign keys in the old tables. It is important to note that this should not lead to losing any kind of data. A good example of this is that an identity for customer Sievo is 3. The name Sievo is only used in one table where the primary key is 3. The identity 3 can then be used in the foreign key columns in other tables. In denormalization the opposite is made and more information is brought together. This is usually done to improve the performance. (Lande 2014)

Dimensional tables usually consist of descriptive and textual attribute values and a primary key for each row. Same as fact table the primary key helps to identify rows but it also helps joining dimension tables to fact tables because fact tables carry the primary key as a foreign key in one column. Dimensions tables are meant to store all the chunky and clumsy description fields. Attributes in the dimension table have two main purposes: labeling query results and constraining and filtering the queries. In general the columns in dimension table are always filled with some information and very rarely do they contain null values. Dimension can also be used multiple times. For example, one time dimension is enough for many columns that define time. Each column does not require its own dimension. (Kimball 2011, 241)

Fact table	Call Id (Primary key)				
	Customer Id (foreign key)	Start time	Duration in seconds	Number called	
	34	2	7.6.2015 15:16	55	123 123 123
	35	2	8.6.2015 0:31	13	12442 111 1
36	1	8.6.2015 8:49	127	91 11 2233	

Dimension table	Customer Id (Primary key)		
	Customer Name	Customer Address	Customer city
	1 Eric Example	Street 1 A 3	Helsinki
	2 Erin Sample	Street 27 B 1	Tampere

Picture 4. An example of fact table and dimension table

4.1.2 Hierarchies

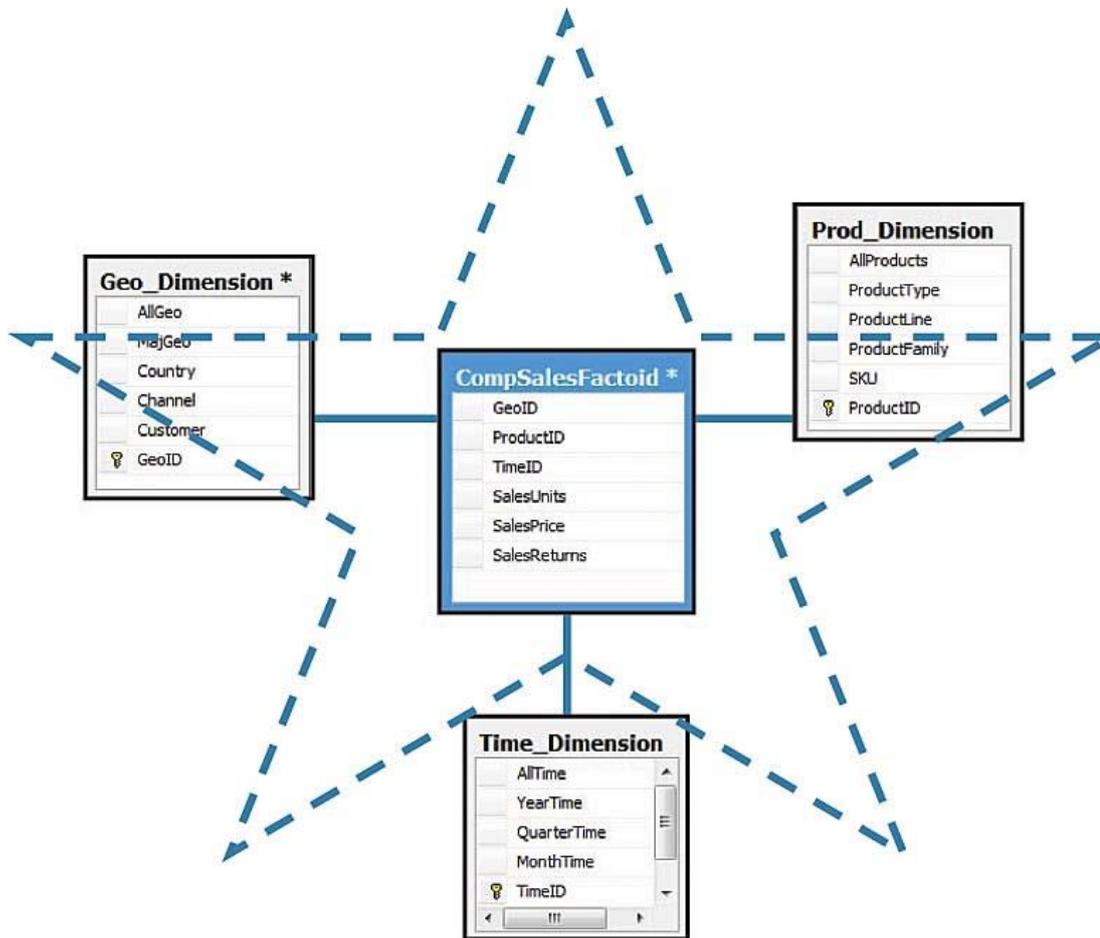
Dimension tables might also consist of hierarchies. These hierarchies are used to categorize data. Hierarchies consist of logical structure. For example, in picture 5 for phone records fact table there would be a customer dimension table that has additional information for customers such as full name, address, billing information etc. This customer dimension links to fact table's foreign key over a unique id. In customer dimension there could also be a hierarchy for customer type such as corporate or personal. Then under personal there could be a lower level hierarchy for different kinds of connections. (Oracle, 2002, 5)

Fact table for phone call	Customer dimension	Customer type dimension	Corporate type
Id	Id	Personal	CEO
CustomerId	Name	Corporate	Employee
Date	Number		Personal type
StartTime	Address		Adult
EndTime	Billing		Child
NumberCalled			

Picture 5. Example of dimension with hierarchies.

4.1.3 Star schema

Star schema gets its name from its diagram that resembles a star (picture 6). A star schema has either just one or multiple fact tables which have references in any number of dimension tables. It is the simplest but at the same time most common way to model a data warehouse. It is possible to create a relation between fact and any dimension table with just one join. Star schema's dimension tables are almost always de-normalized and the fact tables are in third normal form. (Inmon 2002, 137-141)



Picture 6. An example of star schema (databasejournal.com, 2010)

The ultimate strength of star schema is its query performance which provides fast responses with rather simple queries. Aggregate actions like sum, count and average perform really quickly in the star schema. Lastly as the most common data warehouse schema it is the easiest to integrate with some other tool or tools. (Oracle, 2002, 17)

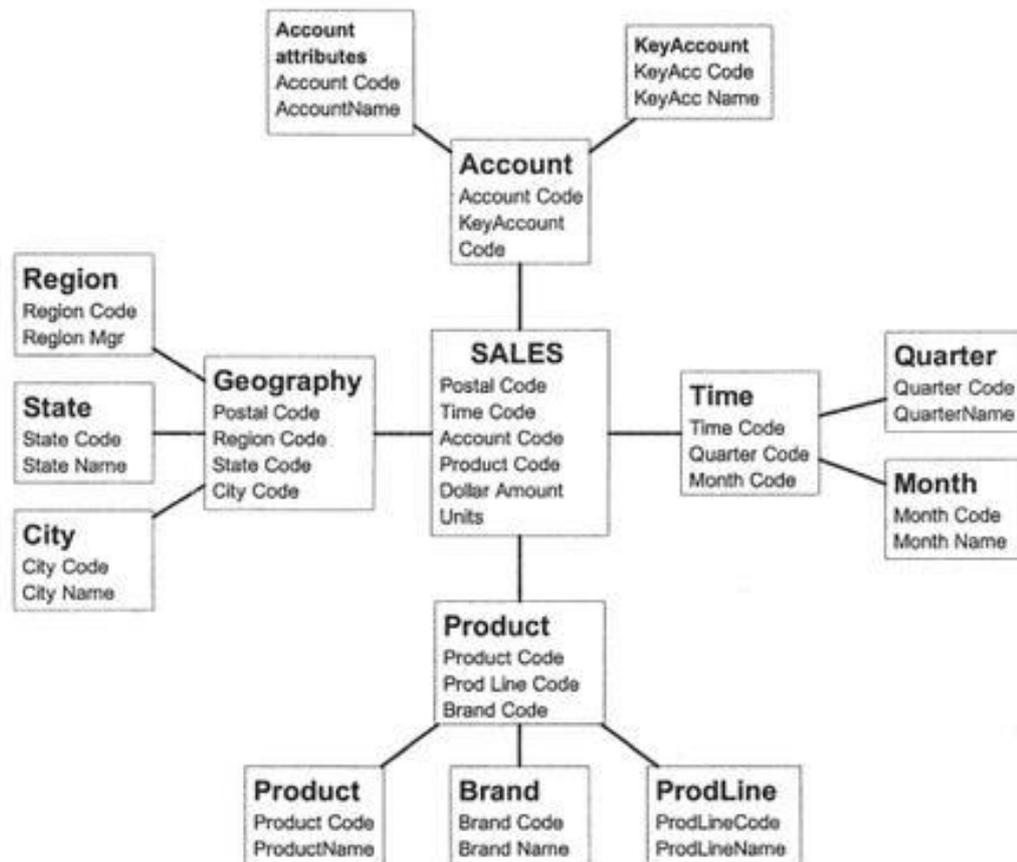
Star schema has one disadvantage which is that its data integrity is poor. This means that the information in the dimension table can create replicates quite easily and it is due to the dimension table being de-normalized. Additional controlling processes are usually re-

quired to avoid these said replicates. Star schema also lacks the ability to create more complex queries in order to analyse the data further. Star schema also doesn't support many-to-many relations between tables and it requires more disk space. The latter usually isn't a problem or at least isn't worth the risk of slowing query times and making them more complex. (Oracle, 2002, 17)

4.1.4 Snowflake schema

Snowflake resembles the star schema a lot and they pretty much present the same data. It also has a fact table which has dimensions but the difference is that snowflake schema is multidimensional (picture 7) and the dimensions are normalized. This means that the dimension tables are divided into many dimension tables. The dimensions can have connections to other dimensions. (Tehcnopedia, Snowflake schema)

The biggest advantage of snowflake schema is that it allows for much more flexible queries and such analysing. Another major thing is that it improves the data quality and so creates less replicates. It will also use less disk space. On the other hand the queries will be more complex and taxing, both for the user and for the query performance. Ralph Kimball who is considered "The Father of Business Intelligence" has stated that in general snowflake dimensions should be avoided. This is explained with snowflakes being harder to understand and navigate for business users and the same information can be stored in a de-normalized dimension table. (Kimballgroup, snowflaked dimensions)



Picture 7. An example structure of a snowflake schema (Athena.ecs.csus.edu)

4.1.5 Star and snowflake schema comparison

Here is a comparison to summarize the differences between star and snowflake schema.

Star schema is better at:

- Queries are less complex and easier to understand
- Query performance is better due to lower amount of foreign keys and star schema requires less amount of joins
- Fairs well with data marts and simple one-to-one or one-to-many relations

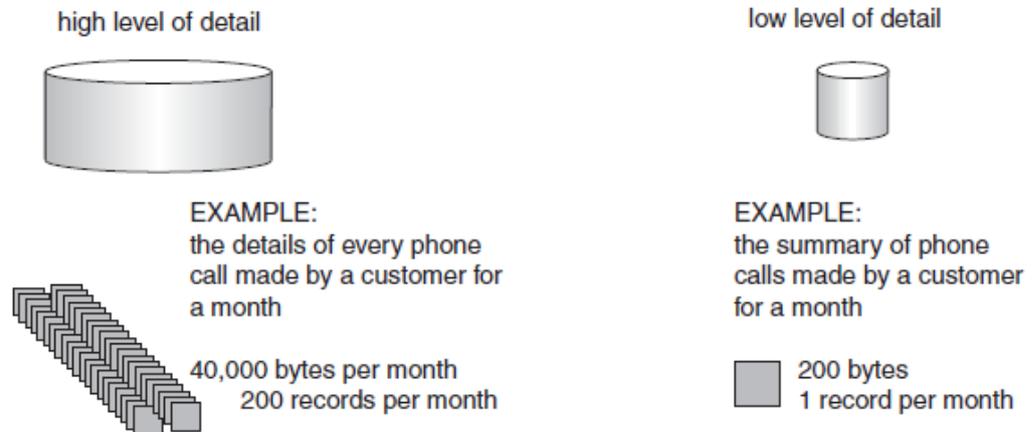
Snowflake schema is better at:

- Snowflake schema has no redundancy and so it is easier to maintain and make changes whereas star schema has redundancy
- Better at many to many data warehouse relations
- Dimensions are normalized

(Diffen.com 2015)

4.2 Granularity

A fact table always has a certain granularity. Granularity is the level of detail that the fact table describes. The lower the level of granularity the more details the fact table consists.



Picture 8. An example for high and low level granularities (Inmon 2002, 44)

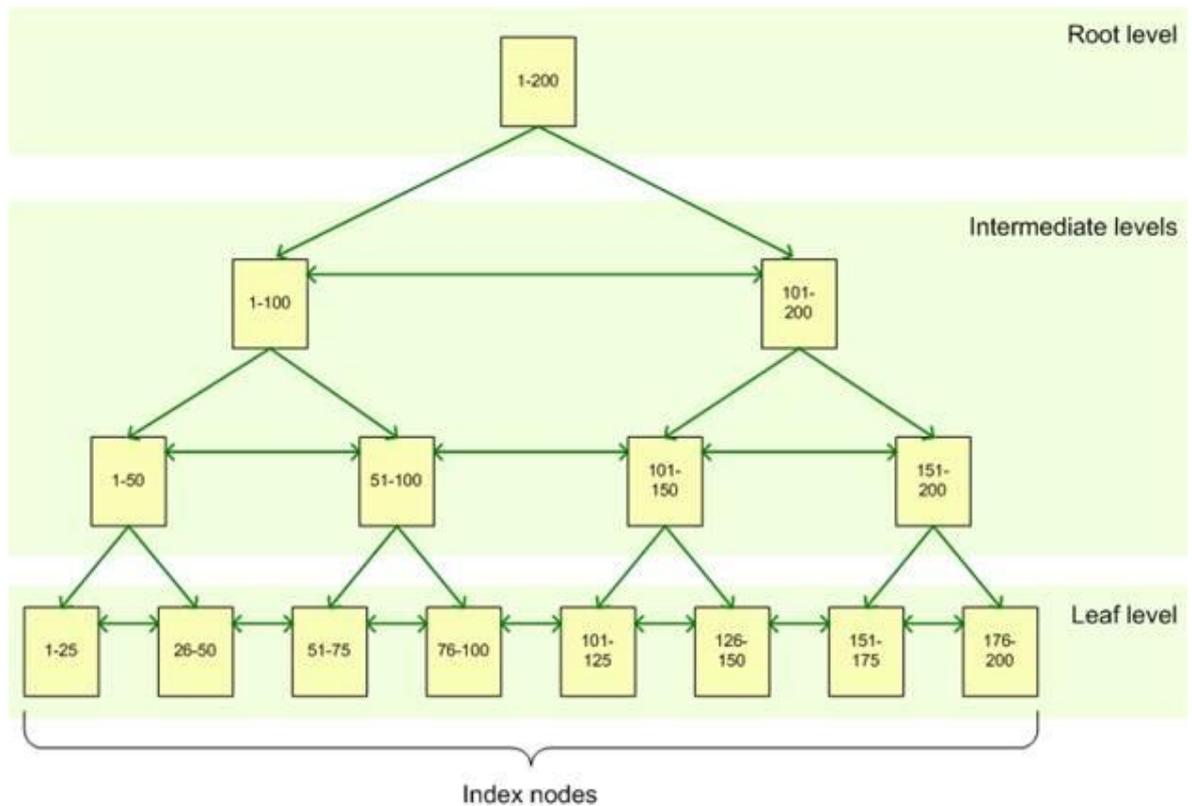
In the picture 8 in high level of detail there is one row of information for every phone call made by a certain customer in a month. These rows contain information like duration of the call, date, start and end time of the call, what number was the call made to etc. In lower level of detail we would have the amount of calls made, average length of a call, what month was it etc.

Figuring out the granularity for fact tables is usually the biggest challenge encountered when designing a data warehouse environment. The typical issue is to bring too high level granularity data into the data warehouse. If the level of granularity is too low, it can be hard to meet the BI needs as there aren't enough details. With too high level of granularity the query performance will suffer due to the bigger amounts of data. This will then affect the scalability as well. (Inmon 2002, 44)

4.3 Database indexing

Indexing is one if not the most important factor in a high performance database and high performance querying. Indexes are created on columns for tables or views to allow a fast access to rows in the data. An index is a set of index nodes that are organized in a so called "B-tree structure" that can be seen in the picture 9 below. The B-tree structure works so that when a query is issued to an indexed column the search engine will start navigating down through the nodes in the intermediate levels. For example, when searching for a row with a value of 57 in an indexed column the engine will first go to node "1-100" then to node "51-100" then to node "51-75". The node "51-75" will either contain all

the data for the rows between values 51 and 75 or then it will just have a pointer that will point to the location where the value 57 is. The former is called “clustered index” and the latter is called “nonclustered index”. When creating primary keys for tables a clustered index is automatically created for the table if the table or view doesn’t already have a clustered index. (Sheldon 2008)



Picture 9. An example of B-tree structure (Sheldon 2008)

The indexes have tremendous value in them and are almost mandatory in bigger data warehouses but they should be designed with caution. Indexes can take up a remarkable amount of disk space. This means that only indexes that will be used should be created. When data rows are updated, new rows are inserted or deleted the indexes update as well. This means that the indexes require maintenance because otherwise over time the changes may affect the performance of an index. (Sheldon 2008)

5 Project planning

The project team included Markus Koivuniemi (author of this thesis), thesis responsible at Sievo and the creator of the current reporting process. The project started with studying phase to get familiar with selected literature and articles related to data warehousing and business intelligence. This was done to build a solid knowledge background to understand the topic better and deeper and to provide as good as possible end results. The studying phase was followed with discussions about what is needed and required from the new database structure. Possible ideas were tossed around and discussed throughout the project as well. These ideas included both new aspects and needs that could come up in the future but also how to work around obstacles and problems when doing the actual planning, implementation and testing. Several meetings were organized over the project about these topics.

Even though all project members use the visual reporting at least weekly it was important to get familiar with the system behind it. This meant being introduced the technical side of reporting such as the database structure and the ETL process for it.

From the business point of view the requirements were somewhat following:

- Changing needs require more flexible structure so adding new data becomes easier and the available data can be taken advantage of better
- More detailed reporting
- Get rid of some “unnecessary” alarms with the current reporting
- Possibility to make the reporting faster in different areas

The project came up with two different structures that were going to be tested with the same sets of data. Then it was to be decided which one of them was preferred and what kind of tweaks would be required to be made. The test versions were to be done and planned fairly fast to have more time on planning the actual structure. When the structure was implemented in the database a bigger and more varying set of data was loaded in the database for final testing.

5.1 Tools used in the project

In this project the following tools were used:

- Microsoft SQL Server
 - Management Studio 2014
- Microsoft Visual Studio 2013
- Microsoft Word 2010

- Microsoft Excel 2010
- Microsoft Visio 2010

Excluding Visio all of these tools are used by the project members in their work. The main weight being on the Microsoft SQL server management studio 2014 and the Microsoft Visual Studio 2013 as these are used on a daily basis.

Management Studio is the tool that is used to create the database, to run the queries against the database and to view the source data and to view the data that is put into the new database. In Visual Studio “SSIS packages” are created to handle the ETL process in this project with the main focus being on loading the data into the database for testing purposes. SSIS comes from SQL Server Integration Services and it is a platform for data integration and workflow applications. This includes the ETL operations for data warehousing. SSIS package is an object that implements the functionalities such as populating tables. Populate simply means adding new data. (Microsoft 2015)

Microsoft Word was used to take notes during the project and to write the documentation for this project. Excel was found useful to create some of the example attachments in this thesis. Visual Studio was used to create the relational database model. This was extremely useful when doing the planning for the data warehouse.

5.2 Old reporting database setup and the requirements for the data warehouse

The current report that is used by the end users reports the different internal steps in Sievo for a so called “source-to-screen” process. Source-to-screen in this case starts with extracting the data from Sievo’s customers and ends with releasing Sievos reports that are available for the customers through Sievo’s software. The current internal report is available in the internal network and it is also running on a big monitor on the wall at the office.

The report gathers the process data from all customers to one place and covers the steps formerly mentioned. The same data could also be dug up from all over Sievo’s own databases but this report makes it visually a lot more appealing and easier to use but most importantly it centralizes the data from hundreds of processes to one place. It is also highly possible that without the report some of the possible errors wouldn’t be spotted as fast as it currently is. On a related note the report is also set to generate alarms based on the information in the data. The report is used by many different teams within the company and adding new aspects and data to the report could be really useful considering the future.

The current setup was kind of a test run that was once done to see whether this kind of reporting is possible to do with the tools available at the time. After that it has kind of grown due to the small additions and changes. Overtime it has gotten to the point that managing the database has become quite a burden as there seems constantly to pop up new requests and ideas.

At the moment the report is based on flat tables of varying width. Width of a table means the number of columns and flat table means that the table contains all the information and it has no dimensional structure. The more columns a table has the wider it is. Adding new stuff is still manageable. In practice this means adding a whole new flat table that gets reported as it is. Trying to bend the report in any way is about impossible. This works up to a certain point but managing a growing number of tables that keep growing all the time might become a burden over time. The main reason the report has stuck in its current form for so long is that it does what it does really well. The biggest flaw is the lack of flexibility and this is what is going to get addressed with this thesis. A dimensional data warehouse will be the most optimal for these requirements. This should tackle both, the scalability and flexibility issues but also make adding new and possibly unexpected data smoother.

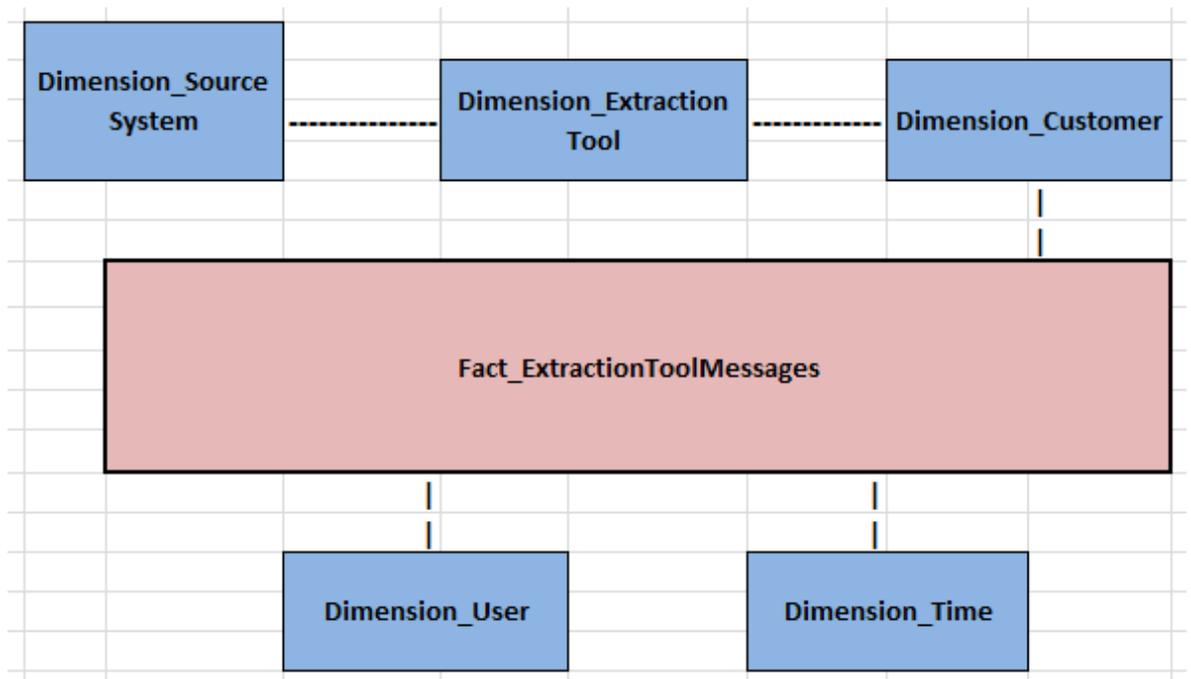
For the requirements it was also decided that there is no need to build ETL process for continuous and for the future use due to the project acknowledging that the schedule will be rather tight. Very simple and “quick and dirty” ETL processes were done to load varying amounts of data to test and play around in the database with queries. It was also noted that getting familiar with the Microsoft Reporting Services was not required in this thesis.

6 First test versions and testing

After initial thought and discussions two kinds of different models came up that were going to get tested. The test data included different kinds of messages that were related to a tool used by Sievo to extract data from the customers. The data set only had data to some of the dimensions in the data warehouse so it was decided to only include the necessary dimensions to save time. For the same reason it was decided to not care about datatypes for the test versions which meant that almost all the data and columns were in text format.

6.1 First test version

The first version (picture 10) was more similar to the current model but with dimensions. A complete version would have around 3-4 fact tables with multiple dimensions. These fact tables would end up really wide. This is caused by the different kinds of messages have different kind of data which would result in multiple columns that can only be used by certain kind of message. This model mainly combined few tables from the current structure to one table in the new. Some of the columns were similar between the tables but a lot would be added as well. Only one fact table was included in the test version.



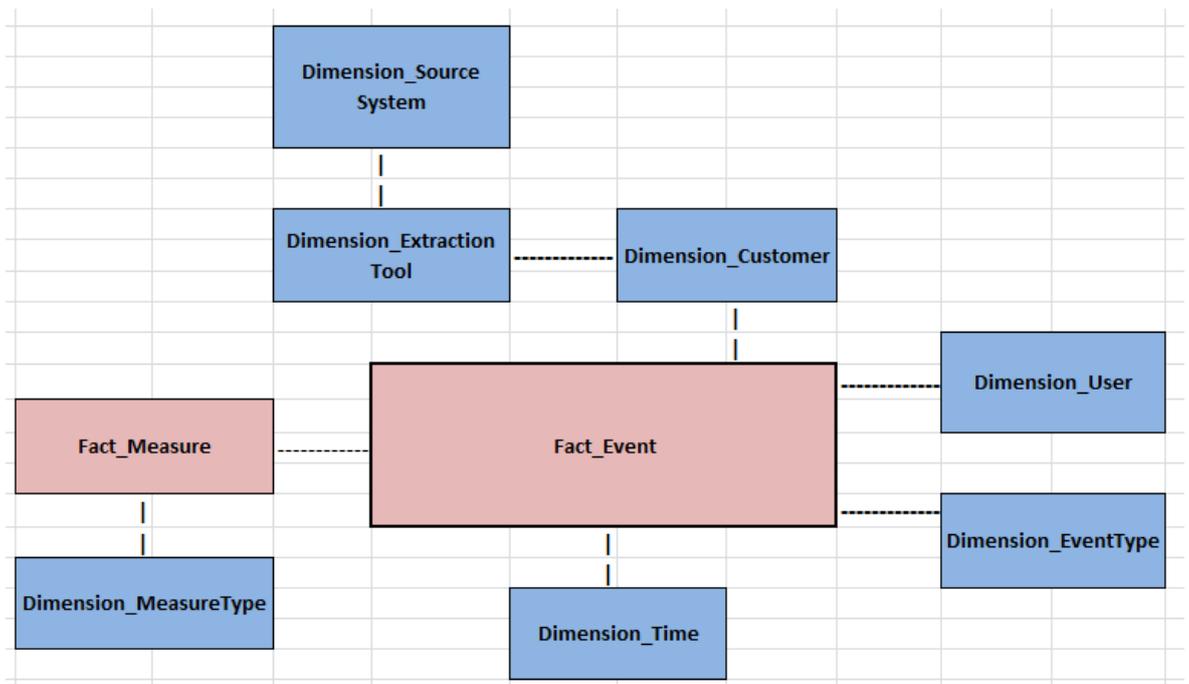
Picture 10. Relational database model for the first test version

6.2 Second test version

The other version (picture 11) had one main fact table **Fact_Event** that was kept pretty narrow. The **Fact_event** consisted of different kinds of events. With this set of test data one event would be one message received from the software. The event types are defined in an event type dimension. For example, one event type would be an extraction message. The idea for this model was also that there would be another fact table that would measure certain values in numeric format for the events, for example the amount of rows extracted in the extraction. One event could have one, multiple or no values in this table and they are linked to the **Fact_Event** table over one events unique id.

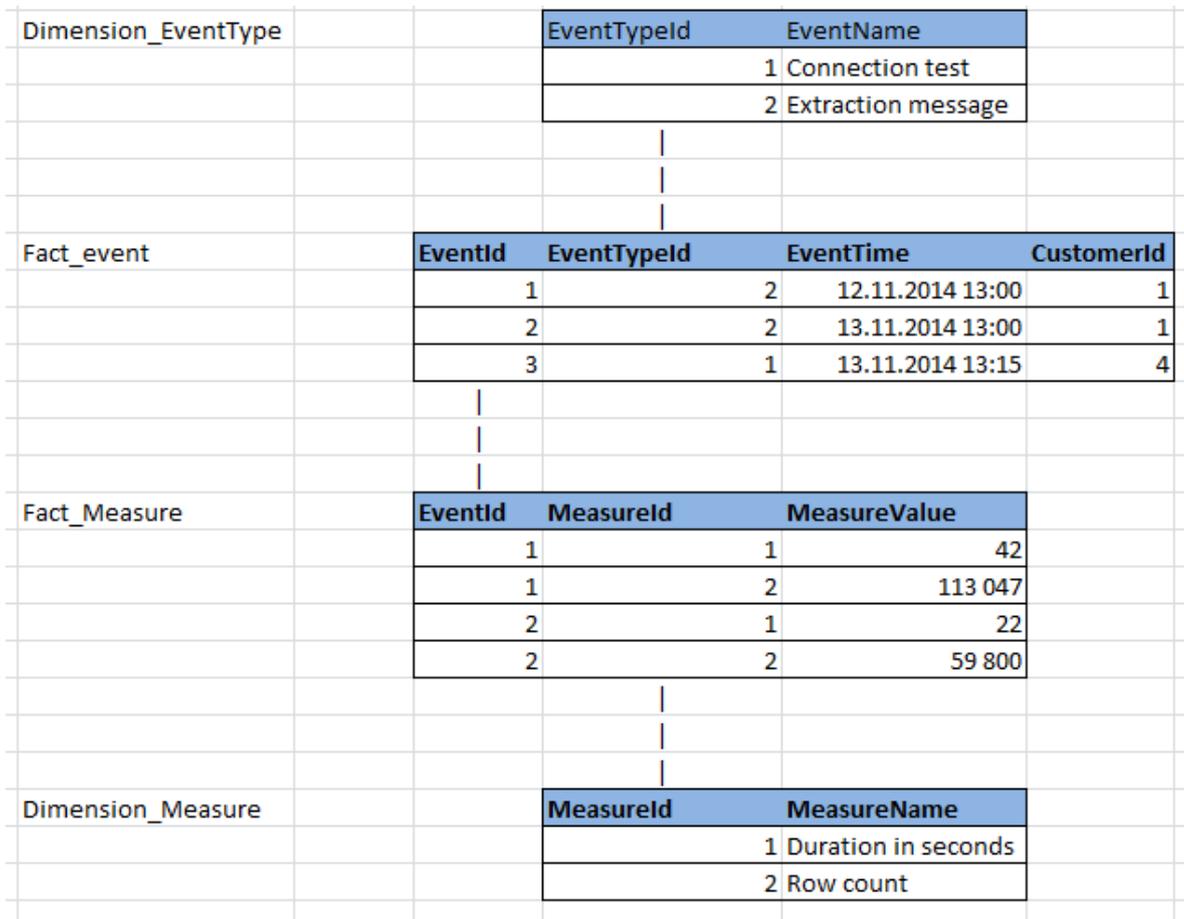
Usually in data warehousing links between two fact tables are not preferred if done over a fact table foreign key or foreign keys. However in this case the join is done over a fact

table's primary key. Also this second version has been tried to make in a way that it will adapt to the data that is coming from different kinds of sources. These sources don't always have for example technical link over each other. It would also be preferred to keep the amount of fact tables to a minimum and some kind of way is required to keep track of the varying values between different kinds of events. This is why the idea of Fact_Measure came up. For the Fact_Measure table there is similar dimension table as there is for event type that will have definitions for different kinds of measures.



Picture 11. Relational database model for the second test version

For example, for the row where EventId is "1" it can be seen that it is an extraction message because the EventTypeId for it is "2". From the measure table it can be seen that for the EventId "1" the extraction took 42 seconds and that 113 047 rows were extracted. In the actual test version a lot more different types of measures were used than what is shown in the picture 12.



Picture 12. Example to demonstrate the logic between Fact_Event and Fact_Measure tables

6.3 Testing and conclusions

As previously mentioned both databases held the same data set. The testing was done with SQL queries which were aimed to return similar result that the current reporting system produces. Then the queries were compared for their complexity and usability. Because of the limited data the query performance couldn't be completely trusted and this is why the execution plans for different queries were analysed as well. Execution plans are visual presentations for the operations that are performed by the database engine. The complexity for the queries was not a deal breaker in this case because the tables are not meant for frequent user querying but new ideas and feedback definitely arose when doing the queries. Ideas for table columns and structure came up as well when doing the load process to these.

For the first version the feelings were really bland. It was extremely easy to use and load data into but it felt too similar to the current setup. Also there were doubts regarding the flexibility for the structure in the long run.

For the second version the first major issue was that the Fact_Event table was kept too narrow and too many repetitive values were put into the Fact_Measure table. This made querying the Fact_Event really heavy process with a lot of joining a table to itself. This was caused by the fact that for the Fact_Measure table data is first “turned” from horizontal to vertical in the loading process and then again turned from vertical to horizontal with the SQL queries as explained in the picture 13. Even though previously was mentioned that the querying the data warehouse can be user unfriendly querying the Fact_Measure felt like it was seriously getting out of hand and there had to be a way to make it in a smarter way. On a positive note about the horizontal to vertical turning was that it was found easier to do when building the load process in ETL than what was thought beforehand. Another point that was found a bit tough when querying the Fact_Event was showing the latest event information for a certain event and/or customer as an example.

<u>Data in source:</u>					
RowId	MeasureValue1	MeasureValue2	MeasureValue3	MeasureValue4	
1	23	33	43	53	
2	11	22	33	44	
3	21	32	43	54	
<u>Data in Fact Measure:</u>					
RowId	MeasureValueId	Value			
1	1	23			
1	2	33			
1	3	43			
1	4	54			
2	1	11			
2	2	22			
2	3	33			
<u>Data in report:</u>					
RowId	MeasureValue1	CustomerId	TimeId	MeasureValue2	MeasureValue3
1	23	3	2015040	33	43
2	11	7	2015040	22	33

Picture 13. Example for horizontal to vertical to horizontal data turning

After discussions and going through the pros and cons for both structures it was decided to go forward with the second test version but with changes. Some of the changes would drive the final structure to be more of a compromise between the two versions. For example looking back, trying to keep the Fact_Event as normalized as possible was probably not the best of ideas and it would be denormalized a lot for the final version. The second

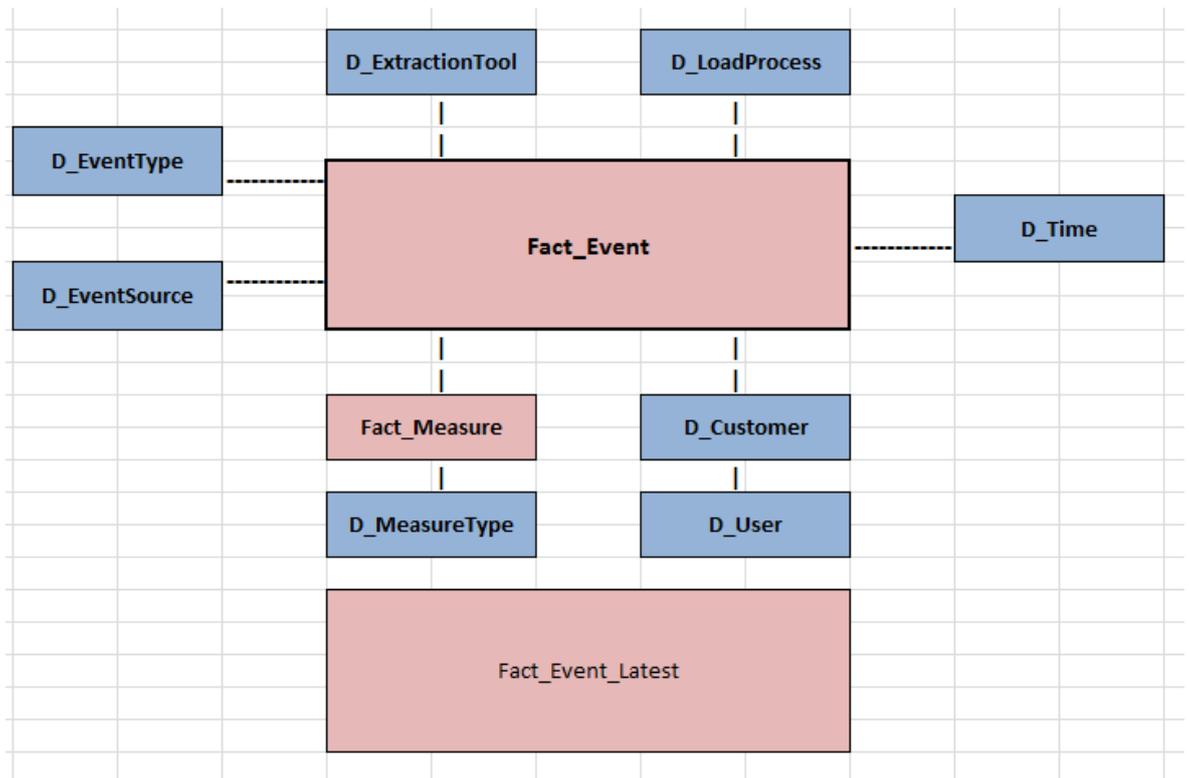
version had a lot more potential considering the future which was the major factor for the project team to further develop it.

7 Final version

The relational model for the database was done in Microsoft Visio. For the test versions no real documentation was done. The Visio documentation was found vital because it helped the project team to picture different kinds of events and plan whether those were technically possible to do or if creating them made logical sense. For this version all the columns and data types for them were also planned. The final Visio model can't be shown due to it containing confidential material but the structure will be described with similar example pictures that have been used before.

7.1 Overview of the relational model

The overall structure for the final version is really similar to the second test version. The point was to create dimensions for the changing values. For the fact and dimension tables there was no restrictions regarding the width of the tables but unnecessary columns were avoided with the fact tables. As end result, the new additions are the event source dimension and the load process dimension. The dimension for the extraction tool has also been changed to its own dimension from being a child hierarchy for the customer and the user dimension has been moved to be the child for the customer dimension. The structure for extraction tool felt a bit clunky in the second test version. In order to get data from source system, the fact table was required to have a foreign key column for customer, extraction tool and source system. A good idea came up with the structure for the import process dimension that would suit the extractor tool dimension as well. This will be expanded a bit later. It is a common good practice to avoid table or column names with spaces so the name references in this text are equal to the names in the actual database.



Picture 14. Relational database model for the data warehouse, dimension tables are in blue and fact tables in red

A whole new fact table has also been added, Fact_Event_Latest (picture 14). This table is structurally identical to the Fact_Event table and it will link to the dimensions in the same way. The point for it is to always hold the latest event for each different event type and customer. This means that every time there is a new event the old data is replaced with the new. This new table should make obtaining the latest event easier as it is likely going to be one of the most common thing checked from the report.

7.2 Dimension tables

All dimension tables have identity (ID) column that is used as a unique primary key for those tables. All IDs are set as not null columns, which means that when inserting data into the table the column can't be null for any row but it always has to contain data. They are all in data type "integer" (int) that is the recommended datatype to be used with primary keys. Int data type is a number that doesn't have integral values. They don't take much space and thus are fast for querying. For cases where the dimension is expected to stay relatively small, a data type "smallinteger" or "tinyint" is used. They are identical to int with the only difference being that they store less numbers in total but they also take less space. The less space the data type uses the faster the queries perform. (Poollet 1999)

All dimension tables excluding time have also a “name” column that is used to describe the dimension row. For name column data type “nvarchar” is used in length 50. Data type nvarchar can store unicode characters, which cover the need for inserting text. Length means the maximum number of characters for that column. With nvarchar the data will only use space according to the amount of characters inserted. For example in picture 15 name “Sievo” will only take space of five characters and not the maximum length of 50.

Many of the dimensions also include some information that might not yet be needed in the current reports but could be found useful in the near future. Storing this kind of information is really easy in these dimensions as it does the same as the current report meaning it gathers the information to one place from many different sources and is easily available from there. For example, storing information about what kind of adapter is in use for the extraction tool to connect to the ERP database. This information can be used with new customers to make setting up the tool easier because we can easily see what adapter has been used before.

7.2.1 Customer and User dimensions

These two dimensions have the same structure apart from user dimension not having a foreign key. They have the previously mentioned ID and name –columns. They both also have the ID –column set as “auto increment” which means that the database ensures the uniqueness of the key. It also means that the value in ID is populated automatically every time a new row is inserted. It is set to start from 1. For example if there was 100 rows in user dimension with IDs between 1 and 100 and 10 new rows were added they would get IDs between 101 and 110 in order they are inserted.

The data to these dimensions is added from Sievo’s own databases. The business key column in this case refers to the ID column for the row in the source database tables. The need for this kind of business keys was found when the load process was being made for the test versions and its purpose is to assist the load process and make validating the data easier. These keys will not be used when generating the new reports.

D_Customer	CustomerId (PrimaryKey) smallint	CustomerName nvarchar(50)	UserId (ForeignKey) smallint	CustomerDatabase nvarchar(50)	BusinessKey (smallint)
	1	Haaga-Helia	7	sievodatabase\DB-1	14
	2	Sievo	11	sievodatabase\DB-7	2
	3	Company123	2	sievodatabase\DB-1	5

Picture 15. Example of customer dimension

7.2.2 Event type, event source and measure type dimensions

Event type and measure type dimensions set the IDs for different kinds of events and measures. Event source is kind of a “nice to know” dimension. It will tell where the source rows are from. For example source system connection test comes in extractor tool message so the source would be extractor tool message. For event source the ID 1 is reserved for “n/a” as can be seen in picture 16 which means not available as seen in picture. Since the event source can be considered more of an additional dimension it isn’t necessary to specify a source for all of the data.

D_EventType		EventTypeId (Primary key) smallint	EventTypeName nvarchar(50)
		1	Connection test
		2	Extraction
		3	Load process
D_EventSource		EventSourceId (PrimaryKey) smallint	EventSourceName nvarchar(50)
		1	n/a
		2	ExtractionTool message

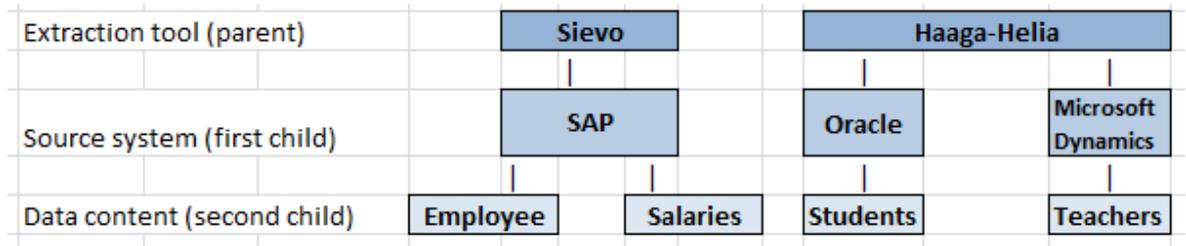
Picture 16. Example of event type and event source dimensions

All of these dimensions are really simple and only have the previously mentioned two columns, ID and name. For these dimensions the data rows must be added manually for both columns and they aren’t available anywhere else. This is something to be taken into account when doing the load processes. Adding totally new event type begins with adding new ID and name in the event type and then adding the data to the fact event table for previously added event type ID. Despite EventType being really simple dimension it makes adding new and possibly unexpected kind of data really easy.

7.2.3 Extraction tool and load process dimensions

These two dimensions have many similarities to the ones introduced before. They have auto increment IDs in int data type with the ID 1 reserved for n/a. They also have the same name column in nvarchar and similar business keys. The table also has bunch of

columns to store varying information regarding extraction tool. The gist for these dimensions is a “parent-child” hierarchy built into both of the tables.



Picture 17. Example of a parent-child hierarchy in extractor tool dimension

The extraction tool dimension is planned to have three levels: Extraction tool where one ID reflects one extraction tool, source system or systems which is the database or databases that the tool is connected to and lastly the extraction data content that holds the extracted data. In the picture 17 is an example of such hierarchy. We have two extraction tools, one for Sievo and one for Haaga-Helia. The tool for Sievo will extract data from an imaginary SAP and the tool for Haaga-Helia will extract from Haaga-Helias two imaginary ERPs, Oracle and Microsoft Dynamics. Employee and salary information are extracted from the SAP, student information is gotten out of Oracle and from MS Dynamics comes information for teachers.

D_ExtractionTool	Extraction-ToolId (PrimaryKey) int	ExtractionToolName nvarchar(50)	ParentId smallint	BusinessKey smallint
	1	n/a		
	2	Sievo		11
	3	Haaga-Helia		12
	4	SAP	2	
	5	Oracle	3	
	6	Microsoft Dynamics	3	
	7	Employees	4	
	8	Salaries	4	
	9	Students	5	
	10	Teachers	6	

Picture 18. Example of extraction tool dimension

Picture 18 shows how the information in picture 17 would look like in the extraction tool dimension with the parent-child –hierarchy. For example, for ID 5 (Oracle) the “ParentId” –column shows values 3. This parent ID 3 refers to the “ExtractionToolId” –column where

ID is 3 which is Haaga-Helia. The ParentId also works in the other way. For example, to figure all data contents under SAP first it must be looked that the ExtractionToolId is 4 for SAP. Then check the whole table where ParentId is 4 which results in ExtractionToolIds 7 (employees) and 8 (salaries). The highest (extraction tool) level has no parents so they don't have ParentIds. The load process dimension is planned to have the same structure with a three level hierarchy, but for the load process.

The reason for choosing this kind of hierarchy in one table instead of "snowflaked" dimension was that adding new levels to both ends is extremely easy for this. It requires no changes in the table structure. When adding a new highest level new rows can be added and then refer to these in the ParentId column for the old highest level rows. If a new lowest level is added the data can be added and just referred to the old lowest level in the ParentId column.

7.2.4 Time dimension

There are a lot of different ways do the time dimension. The time dimension for this data warehouse was tried to be kept rather simple. It could be expanded with for example quarters and quarter names or names for every column or some kind of holiday marking or then the granularity could be on hour or even minute level. As can be seen from picture 19 the table is rather narrow and the granularity in this time dimension is on a daily level. As said before it doesn't bother to have wide dimension but it was felt that in this case there isn't and there won't be a need for more detailed time dimension.

D_Time	TimeId (Primary Key) int	Year smallint	Month tinyint	Week nvarchar(20)	Day tinyint	Date date
	1	0	0	n/a	0	0001-01-01
	20140101	2014	1	2014-1	1	2014-01-01
	20140102	2014	1	2014-1	2	2014-01-02
	20140103	2014	1	2014-1	3	2014-01-03
	...					
	20150208	2015	2	2015-6	8	2015-02-07
	20150209	2015	2	2015-7	9	2015-02-08

Picture 19. Example of time dimension

Instead of running ID that starts from 1 it was decided to have the ID as YYYYMMDD where YYYY is year MM is month and DD is day. This way the ID will be unique for each row and it will make querying the database much easier and it will also ease the load pro-

cess a little. Querying will be easier because the Timeld will be known even though the table content was unknown. The ID 1 will be reserved for not available in this dimension as well. The datatypes are all really simple and integers are preferred because they don't consume much space.

7.3 Fact tables: Event type, measure type and latest event type

This data warehouse ended up with three different fact tables. How the Fact_Measure table operates has already been explained in picture 12. The Fact_Measure table forms its unique primary key over two foreign keys. One Foreign key linking to table Fact_Event and the other one to D_MeasureType as can be seen in picture 20. The third column is MeasureValue that stores the changing number values that change between same and different event types. The data type is a numeric(18,6) which accepts number values with max length of 18. From the 18 numbers, 12 can be integer and 6 can be decimal at the most.

Fact_Measure	EventId (PrimaryKey/ ForeignKey) int	MeasureTypeId (PrimaryKey/ ForeignKey) smallint	MeasureValue numeric(18,6)
	333	2	111
	333	3	222
	333	4	333
	343	2	121
	557	7	555,777

Picture 20. Example of Fact_Measure

The Fact_Event_Latest has identical structure as the fact Fact_Event table as said before. It will have the same primary and foreign keys but it will be much smaller in size. The granularity for Fact_Event table is one event and the types and descriptions for those events can be found from the event type dimension.

About half of the Fact_Events columns are foreign keys that link to the primary keys of dimensions introduced earlier. The datatypes for these columns are the same as they are in the dimensions and all of them are not null columns so they don't accept null values. In case the row would not have value for one or some of these columns, it would use the ID 1 for not available like in the picture 21 in column ImportProcessId on row where EventId

is 3. The EventId is the primary key for the Fact_Event table. Its data type is “int” and it is an auto increment column.

	EventId (PrimaryKey) int	EventTypeId (ForeignKey) smallint	EventSourceId (ForeignKey) smallint	TimeId (ForeignKey) int	CustomerId (ForeignKey) smallint	ImportProcessId (ForeignKey) int	ExtractionToolId (ForeignKey) int
Fact_Event							
Keys	1	2	1	20140611	3	1	7
	2	5	3	20140611	3	1	77
	3	5	4	20140611	3	3	1

Picture 21. Example of Fact_Event keys

The other half of the Fact_Event table consists of the changing values within events that are not numeric or that had no point of creating an own dimension. These include data type datetime2 columns: EventCreationTime, EventTime and NextEventTime. EventCreationTime is the time it is put into the Fact_Event table. EventTime is the time that the event has originally happened. NextEventTime is a time when the next event is expected to happen. The NextEventTime is not available for all event types as is emphasized in the picture 22 in the first row. The datetime2 is a preferred datatype nowadays over datetime as it takes less space, is the standard and has bigger range.

	BusinessKey int	EventCreationTime datetime2	EventTime datetime2	NextEventTime datetime2	Text nvarchar(255)	ErrorText nvarchar(255)	Status bit	Deleted bit
Fact_Event								
Fact values	223	2015-09-11 01:00:00	2015-09-10 23:15:00				1	0
	224	2015-09-11 01:00:00	2015-09-10 23:15:00	2015-09-15 23:15:00		connection failed	0	0
	225	2015-09-11 01:00:00	2015-09-10 23:15:00	2015-09-15 23:15:00			1	0

Picture 22. Example of Fact_Event fact values

Some events have some kind of text attached to them and the Text column is reserved for that. It is in data type nvarchar with max length of 255 which is the default length for nvarchar column. A second text field ErrorText was needed as well that is reserved for error values. At first it was thought to have one text column in the Fact_Event to keep the columns containing text to a minimum as those consume the most memory but it came up that there are some event types that can have both, the error text and some kind of other text attached.

Finally there are two columns with data type “bit”, “Status” and “Deleted”. The data type bit only allows three values in them: true (1), false (0) or not available (null). The numbers 1 and 0 is just a way for SQL Server to show them. When Status is 1 the event has completed successfully and when it is 0 it means there has been some kind of error. It is a good practice in data warehousing to use some kind of column to mark if the row is deleted or not as it can usually be configured into the reporting softwares to not show those rows to the end users. This way the rows don’t have to be deleted permanently from the

warehouse. These bit columns are in this sense used as “flags”. By default they Status expects value 1 and deleted expects 0.

7.4 Indexing

As explained earlier indexes speed up the searches for queries in database. In case when a table is not indexed some queries will have to go through the whole table before returning the results. Indexes make the area that has to be read smaller. All tables having primary key as auto increment have that column by default as clustered index in SQL server. If the index contains multiple columns they are arranged in a priority order. For this data warehouse an additional nonclustered index was created for each fact table but measure and all dimension tables.

Topic was approached with the mindset of figuring out what columns were going to be the most commonly used in the “WHERE” condition of the queries. For example for Fact_Event the nonclustered index was created over EventType, Timeld and Customer – columns in that priority order. Indexes can also have a column that they include and return. This should be a column that is most likely needed from the queried table. In the Fact_Event table’s case this would be the EventId, status and deleted so those were chosen for the index to include.

For all dimensions except the time the index was created over the name and ID was put as the included column. It was thought that there could be multiple cases where for example customer is looked by the name instead of the identity but the ID is the one that is wanted out from the query.

7.5 Testing of the final version

The testing was carried out in a similar manner to the first versions testing so the ETL-processes created for testing aren’t meant for continuous use. Only difference being that there were more and different kinds of data sets to see how different kinds of events could be put into the database and if it would be possible without making changes. Only tweaks that had to be made were that a lot more business key columns were added to help with the load process. This is not a problem because adding columns to existing tables is easy and it doesn’t affect the structure of the tables otherwise. Also some staging tables were created and used to support the ETL process. This was not considered a problem and it was to be kind of expected. A totally new kind of event and related data was also inserted into the database that wasn’t considered beforehand. This loaded in just fine with a little help from the staging tables. The project team was positively surprised about how well the

database adopted different kinds of data and how easy it was to add new kind of data into it as it was the main goal for this project.

Only few reports have been generated so far with the Microsoft reporting tool and dive. Generating these reports is not expected to create too many issues as similar results can be done with querying the new structure compared to the old tables.

8 Conclusions and pondering

The main goal was to create a flexible and scalable data warehouse to replace the current database structure. Planning and creating the data warehouse goal was accomplished. Building a complete ETL –process and then configuring the Microsoft Reporting Services will be required before taking the data warehouse into use. Adding more and new kinds of event types should not cause much work or any issues. In case new dimensions are needed or the already existing need to be expanded the required actions shouldn't take much effort. This data warehouse model should be easily scalable and flexible to many directions.

The project start got a bit delayed from what was originally planned and the whole project was done over roughly four months. This made for a rather strict schedule. Time had to be reserved also for the author to get familiar with the theory background. At first the idea was to include creating the ETL-process that could be taken into use as well as planning and developing the data warehouse. Later on it was decided that only planning and establishing the data warehouse would suit best for this project's scope.

Already during the project some questions and requests regarding the ability to report more information arose. To these requests the project team has gladly been able to respond that this should be possible with the new data warehouse once it gets implemented. There was also an idea to have the ability to comment on events and thus possibility to neutralize alarms and warnings. Implementing this falls more on the ETL –side than data warehouse so it was dropped from the project scope. However, it has been taken into account so that adding should not be big effort. The user dimension was also partly implemented because of this.

Altogether the testing was done with about half a million rows of data and various different types of events. The project team feels that this is adequate proof for the functionality and thus the project can be considered done successfully. All in all the project team and author of this thesis are really satisfied with the results.

8.1 What was learned

My current database knowledge is based on 6 months working experience at Sievo and on few database related courses at school. As I have gotten rather familiar with how things are done in Sievo on the technical level I was eager to learn about other possible ways of achieving the same or similar results and learn more about databases and data-

base structures in a more general level. The project also improved my SQL skills. I feel learning targets were accomplished.

When writing the thesis I had to constantly watch that I don't cross the line and slip anything confidential in there. Balancing around this area was surprisingly hard. Due to this I also had to make a lot of the example pictures myself and I couldn't just use something that already existed or was created during the project. I also found it hard to estimate how much base knowledge can be expected from reader. It affected much issues can be described only on how generic level without going into details. This was just something that caught me a bit off guard and I wasn't totally prepared for it.

I knew already before the project that the topic would be a bit challenging. The biggest obstacles would definitely be on the technical side of things as I had no previous experience of this type of work. I got great support and feedback during the project whenever things felt or seemed to be stalling. Otherwise the end results would be most likely quite different or at least the results would not be as well prepared for a production use as they are now. Overall I am extremely happy that I chose this topic and am at least as happy and proud with the outcome of this thesis.

9 Bibliography

Athena.ecs.csus.edu

<http://athena.ecs.csus.edu/~olap/olap/snowflakeschema.jpg>

Chaudhuri S., Dayal U., Narasayya V. 2011. An Overview of Business Intelligence Technology. Readable:

<http://web.ist.utl.pt/~ist13085/fcsh/sio/casos/BI-Tech2.pdf>. Read 11.11.2015

Communications of the ACM 2011. An Overview of Business Intelligence Technologies.

Readable:

http://delivery.acm.org/10.1145/1980000/1978562/p88-chaudhuri.pdf?ip=91.154.153.143&id=1978562&acc=OPEN&key=4D4702B0C3E38B35.4D4702B0C3E38B35.4D4702B0C3E38B35.6D218144511F3437&CFID=527807180&CFTOKEN=83591280&_acm_ =1443373609_e9ba3701a1465caaed253809db7b248c

Databasejournal 2010

<http://www.databasejournal.com/features/mssql/article.php/3911476/Examples-of-SQL-Server-Implementations.htm>

Diffen. Snowflake schema vs. star schema. Readable:

http://www.diffen.com/difference/Snowflake_Schema_vs_Star_Schema. Read 23.10.2015

Eckerson W. 2014, <http://www.b-eye-network.com/blogs/eckerson/BI%20Evolution%20-%20User%20Trends.jpg>

ETL (Extract-Transform-Load). Readable:

<http://www.dataintegration.info/etl> Read 20.10.2015

Inmon W. H. 2002. Building the data warehouse, 3rd edition. Wiley

Kemochan W. 2014. Business Intelligence 101: A Brief History. Readable:

<http://www.enterpriseappstoday.com/business-intelligence/business-intelligence-101-a-brief-history.html>. Read 11.11.2015

Kimball Group. Multipass SQL to Avoid Fact-to-Fact Table Joins. Readable:

<http://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/multipass-sql/>. Read 2.11.2015

Kimball R. 1996. Design requirements for aggregate environments. Readable:
<http://www.kimballgroup.com/1996/08/aggregate-navigation-with-almost-no-metadata/>
Read 24.10.2015

Kimball R., Ross M. & Thornthwaite W. 2011. John Wiley & Sons.
The Data Warehouse Lifecycle Toolkit 2nd Edition

Lachlan J. 2013. How organization size affects Business Intelligence goals, deployments, adoption and success. Readable:
<https://www.yellowfinbi.com/YFCommunityNews-How-organization-size-affects-Business-Intelligence-goals-deployments-adoption-a-142925>. Read 22.10.2015

Lande J. 2014. Design a Better SQL Database With Database Normalization. Readable:
<http://joshualande.com/database-normalization/>. Read 11.11.2015

Microsoft 2015. SSIS Package Essentials. Readable:
<https://msdn.microsoft.com/en-us/library/cc280511%28v=sql.105%29.aspx>. Read
31.10.2015

Negash, S., 2004. Kennesaw State University. Readable:
<http://aisel.aisnet.org/cgi/viewcontent.cgi?article=3234&context=cais>. Read: 27.9.2015

Oracle 2002. Oracle9i Data Warehousing Guide Release 2 (9.2). Readable:
http://docs.oracle.com/cd/B10500_01/server.920/a96520/concept.htm. Read: 17.9.2015

Poolet M. 1999. SQL by Design: How to Choose a Primary Key. Readable:
<http://sqlmag.com/database-administration/sql-design-how-choose-primary-key>. Read
[7.11.2015](#)

Rouse M. 2014, ERP (enterprise resource planning) definition. Readable:
<http://searchsap.techtarget.com/definition/ERP>. Read 22.10.2015

Rouse M. Business Intelligence Definition. Readable:
<http://searchdatamanagement.techtarget.com/definition/business-intelligence>. Read
[7.10.2015](#)

Sheldon R. 2008. SQL server index basics. Readable: <https://www.simple-talk.com/sql/learn-sql-server/sql-server-index-basics/>. Read 25.10.2015

Snowflaked dimensions, Kimballgroup. Readable:

<http://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/snowflake-dimension/>. Read 14.10.2015

Super Develop 2014.

<http://superdevelop.com/databases-understanding-data-cleansing/>

Technopedia. Snowflake schema. Readable:

<https://www.techopedia.com/definition/28068/snowflake-schema>. Read 14.10.2015

Wan D. 2007. Aggregation for Data Warehouse. Readable:

<https://dylanwan.wordpress.com/2007/04/04/aggregation-for-data-warehouse-part-i/>.

Read 24.10.2015