Web vs. Standalone Application
- A maintenance application for Business Intelligence

Martin Löfberg, Patrik Molin
This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 40 (2*20) weeks of full time studies.

Contact Information:
Authors:
Martin Löfberg
E-mail: martin@ball.se

Patrik Molin
E-mail: patrik@ball.se

External advisor:
Tony Jönsson
WM-data Sverige AB
Address: Boplatsgatan 4, Box 9116, 200 39 Malmö
Phone: 0733-983252

University advisor:
Lars Lundberg
Sektionen för teknik, Avd för programvarusystem.

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25
ABSTRACT

Many companies want a lot of functionality over the web. Is it possible to achieve the same functionality on the web compared to an ordinary windows application?

Our work aims towards evaluating which one of the solutions is the best. Many customers want a standalone application rich of functionality and demands to have the same functionality on the web. Is it always possible to achieve the customer’s requirements on a web-based solution or do you have to settle with an implementation of a standalone application? There are some factors that the answer depends on: performance, security, usability and implementation.

The application that will be tested is developed in .Net and is a maintenance application for Business Intelligence (BI). We will have a short introduction to the Business Intelligence field to make you understand the purpose of the application.

Keywords: Data Warehouse, web based, standalone, .NET, Business Intelligence
CONTENTS

ABSTRACT ............................................................................................................................... I

CONTENTS ................................................................................................................................... II

1 INTRODUCTION .................................................................................................................. 1

2 BACKGROUND .................................................................................................................... 3
  2.1 BUSINESS INTELLIGENCE .............................................................................................. 3
      2.1.1 The different steps in a Business Intelligence solution ........................................... 4
  2.2 DATA WAREHOUSE....................................................................................................... 4
  2.3 STANDALONE VS. WEB BASED APPLICATION ............................................................ 5
      2.3.1 Standalone application ......................................................................................... 5
      2.3.2 Web based application ......................................................................................... 6
      2.3.3 Web or not from a Business Intelligence perspective .......................................... 7

3 METHOD ............................................................................................................................... 9
  3.1 IMPLEMENTATION ...................................................................................................... 9
  3.2 PERFORMANCE .......................................................................................................... 9
  3.3 SECURITY .................................................................................................................... 9
  3.4 USABILITY & LAYOUT ............................................................................................... 10

4 RESULTS ............................................................................................................................. 11
  4.1 IMPLEMENTATION .................................................................................................... 11
  4.2 PERFORMANCE ....................................................................................................... 13
  4.3 SECURITY ................................................................................................................... 16
  4.4 USABILITY & LAYOUT ............................................................................................ 16

5 ANALYSIS .......................................................................................................................... 19
  5.1 IMPLEMENTATION .................................................................................................... 19
  5.2 PERFORMANCE ....................................................................................................... 19
  5.3 SECURITY .................................................................................................................. 20
  5.4 USABILITY & LAYOUT ............................................................................................ 21
  5.5 VALIDITY .................................................................................................................. 21

6 CONCLUSION ..................................................................................................................... 22

7 REFERENCES ..................................................................................................................... 23

8 APPENDIX ........................................................................................................................... 24
  8.1 APPENDIX A – THE DIFFERENT STEPS IN BI .......................................................... 24
  8.2 APPENDIX B – STAR SCHEMA .................................................................................. 26
  8.3 APPENDIX C – THE DATA WAREHOUSE MODEL ................................................... 27
  8.4 APPENDIX D – USABILITY AND LAYOUT FORM ...................................................... 28
  8.5 APPENDIX E – DATABASE DIAGRAM ................................................................. 30
  8.6 APPENDIX F – UML DIAGRAM .............................................................................. 31
  8.7 APPENDIX G – EXAMPLE OF A XML-FILE .............................................................. 32
  8.8 APPENDIX H – INPUTHANDLER ............................................................................ 34
  8.9 APPENDIX I – GLOSSARY ...................................................................................... 36
1 **INTRODUCTION**

Business Intelligence (BI) and Data Warehouse (DW) are underestimated tools for analyzing data in a company. BI and DW are designed to help the company making business related decisions. Large volume of data is stored in databases which then are assembled into a structure for analyzing.

In the late 1990s the term Business Intelligence got its share of attention and became popular. At this time the organizations realized what information could do for there businesses. Business Intelligence as a concept is older but for the first time due to the cost of hardware and disc space it was possible for organizations to deploy BI solutions and Data Warehouses [1].

As Jonathan Wu says in his article “Business Intelligence: Differences Between Web and Non-Web Applications” (2000) BI applications have evolved over the last years and more features and functionality have been implemented in these applications. A lot of the new BI applications are Web based solutions. Are you supposed to choose the web or standalone approach for your BI application? Most BI tools today have two widely accepted versions; one full client (standalone), and one thin client (web based) application.

When a customer contacts a software company regarding the development of an application it often sounds like: “I want an application with this and that functionality…and I want the same functionality on the web.”

Is this possible to achieve or do you have to compromise with some of the functionality on the web?

The functionality is the customer’s only concern and the customer does not care how it is implemented. Therefore it is up to the developers to explain to the customer that it is not always possible to achieve the same functionality for both solutions. It is often easier to implement a lot of functionally in a standalone application, a so called thick client than in a web based application, a thin client solution. It is often harder or impossible to achieve the same functionality in a web solution because it is a client server architecture where the functionality is on the server side.

The main advantage of a web implementation today is the fact that it is centralized. This makes it easy to distribute and maintain. This is a good thing but is it good enough to make it a better solution than a standalone solution where you can have more functionality?

The application that will be implemented and tested is a tool that structures information in a relation database so it becomes useful in BI solutions. The standalone application runs in a Windows environment and both solutions are to be developed in C# using the .NET framework.

The application should be database independent and the information that the application will process will be used as a ground for dimensions in different OLAP databases.

Our objective in this thesis is to compare the differences between a standalone and a web application. The comparison will be done according to these research questions:

- Implementation
- Performance
- Security
Usability & Layout

To evaluate the differences we will conduct some tests and analyze the results. To see the difference in Usability and Layout we will interview people with different backgrounds and different outlooks.
2 Background

This chapter will discuss the meaning of Business Intelligence and Data Warehouse. It will give you a short introduction of how it works and what it can be used for. We will also talk about the differences between a standalone and a web application. How they differ in implementation, security, usability and performance.

2.1 Business Intelligence

What is Business Intelligence? There are a lot of definitions on Business Intelligence. For example the one from Informatica (www.informatica.com):

“Business Intelligence (BI) is a category of applications and technologies for gathering, storing, analyzing, and providing access to data to help enterprise users make better business decisions. BI applications include decision support systems, query and reporting, online analytical processing, statistical analysis, forecasting, and data mining.” [2]

Another definition from wikipedia:

“Business Intelligence (BI) is the process of gathering information in the field of business. It can be described as the process of enhancing data into information and then into knowledge. Business Intelligence is carried out to gain sustainable competitive advantage, and is a valuable core competence in some instances.” [3]

It is hard to define Business Intelligence because it is a vague expression and is used in many different contexts. To understand what Business Intelligence is and what the benefits of BI are we will give you a more detailed description on the subject.

The difference between intelligence and data/raw information can be explained by these short sentences:

- What has happened is history (data)
- What is happening is news (data)
- What will happen in the future (intelligence)

The function of Business Intelligence is the organizations ability to identify and solve future problems by gathering and analyzing business information. Intelligence is the refined information that makes it possible to see into the future. Therefore BI is a very important factor in order for a company to win the competition against its rivals. Without a good BI function it is impossible to do good businesses in the long run.

The entire organization benefits from Business Intelligence. The head management can for example see the organizations profitability of the products/customers, “executive” reports and balanced score card. The sales persons in the organization can benefit from BI with sales reports and the analysts can see deviations and plan campaigns. Last but not least the employees can for example make better budgets and product investments.
2.1.1 The different steps in a Business Intelligence solution

There are many models that describe BI and its different steps. We have chosen the one in figure 1 which has six steps. It describes how data is gathered and processed from the company’s source systems to suite the users needs. To see a description of all the steps in the figure, go to appendix A.

![Figure 1. The different steps in BI](image)

The application that will be implemented and tested will be a tool for the Storage area (step 3). This is where the correct and quality assured data is stored, the so called Data Warehouse. The stored data is the foundation for the rest of the BI solution. Sometimes it must be possible to look at this data and change it in order to achieve the requirements.

2.2 Data Warehouse

In the field of Data Warehousing there are two gurus, Bill Inmon and Ralph Kimball. They have different philosophies of DW and neither of them is wrong.

**Bill Inmon's paradigm:** Data Warehouse is one part of the overall Business Intelligence system. An enterprise has one Data Warehouse, and data marts source their information from the Data Warehouse. In the Data Warehouse, information is stored in 3rd normal form.

**Ralph Kimball's paradigm:** Data Warehouse is the conglomerate of all data marts within the enterprise. Information is always stored in the dimensional model.

Bill Inmon defines the concept as follows: “A Data Warehouse needs to serve the needs of all of its users, not just one class of users.”

With this he means that in an enterprise environment there are many classes of users such as accounting, finance, marketing, production etc. These different users have different looks on the data in the Data Warehouse. This requires that the Data Warehouse have relationally designed tables for the data [4].

As said in the paradigm of Bill Inmon the data should be in 3rd normal form and should be at a low level of granularity. Low level of granularity means that the Data Warehouse is more detailed. The granularity is important in the design because it affects the volume of data in the DW. The volume of data in a warehouse is traded off against the level of detail of a query [5].

Kimball’s way of seeing DW is closer to the real world. This is because most Data Warehouses originated as data mart. Only when more data marts are built later do they evolve into a Data Warehouse. This is the big difference when comparing to Inmon. He says that even if you combine many data marts you do not get a Data Warehouse.
According to Kimball the Data Warehouse consists of two different types of data marts, the atomic data marts which hold multidimensional data at the lowest level and the aggregated data marts that can store data according to a core business process. In both atomic and aggregated data marts the data is stored in a star schema like the one in appendix B.

A Data Warehouse is a way of storing data for later retrieval. This retrieval is almost always used to support decision-making in the organization. That is why many Data Warehouses are considered to be DSS (Decision Support Systems). Some people argue that not all Data Warehouses are DSS, and that’s true. Some Data Warehouses are only store copies of data. The primary objective of Data Warehouses is to gather information from different sources and put it together into a format that helps in the decision making.

2.3 Standalone vs. web based application

To conduct our tests whether to use a standalone or web based solution for a Business Intelligence application two different applications will be developed. To understand the different architectures they will be described in this chapter.

2.3.1 Standalone application

A standalone or thick client refers to an application running on a desktop environment such as windows or Mac platforms. When the Graphical User Interface (GUI) was developed by Apple in the 1980s, it made it possible to do things in an easier way on a desktop computer. The users could now perform 95% of there work without having to remember cryptic commands [7].

The thick client architecture, where the code runs on the client as well as the processing of data, was once the norm. With the spreading of internet the thin client model became more popular [8]. The thin client also became more popular because the standalone applications grew more complex and depended on third party controls.

Everything has some drawbacks, and that also includes thick clients. One of the top arguments against thick clients is when it comes to deployment. Imagine a thousand clients that must be updated. This is not an easy task and it takes a lot of time to update all clients. In the future this argument will no longer be true because Microsoft has come up with something called ClickOnce deployment. ClickOnce is a part of the .Net 2.0 Framework and will be further enhanced in the next version of Windows, code-named "Longhorn" [9] [10].

Another problem with a standalone application is the platform dependency. A thick client requires a local runtime environment. For example a Windows Form application will only run on a windows platform with the .Net framework installed.

These drawbacks are compensated with the ability to work offline. It is possible for the application to run offline but it can only work with local data due to the non existing internet connection. Because all computation is done on the computer that the application is running on, the amount of data transmitted over the internet is reduced. The client retrieves data from a data source, makes some computation on it and then sends it back to the server. In the case of web based application the data is passed back and forth between the client and the server each time a new calculation is to be done. If many clients are connected to the server at the same time this leads to allot of processing on the server and the power of the clients is not used [9].

2.3.2 Web based application

The World Wide Web came to life in the early 1990s when CERN laboratory in Switzerland needed to distribute documents and graphics via the Internet [11].
The WWW brought to the basic computing two very important concepts:

1. Hyperlinks
2. One common user interface for all applications [7]

To run resources in the form of executable programs the CGI (Common Gateway Interface) was invented. The CGI allowed a web browser to execute resources on a web server. This took web sites to another level, what we could call web applications and made it possible to use far more logic than HTML could accomplish. The developers were able to accomplish standard data processing functionality such as database access and they could distribute it across the world [11].

As the World Wide Web grew users became more comfortable with using various applications on the web. The users did not have to run different programs for each function that they wanted to perform [7].

Before the web became dynamic the only thing that could be requested from a web server was static pages. Every line of the HTML page was written by the designer before it was placed on the web server. When a client requests a static page the server reads the request and finds the right page. The server then sends the requested page back to the client.

The development of building dynamic web pages led to a software program called application server. When the web server receives a request for a dynamic page, it passes this page to the application server. The application server reads the code on the page and finishes the page according to the instructions in the code. The page that is returned from the application server to the web server is static. The web server then sends this page back to the requesting browser.

An application server lets you work with resources on the server, such as databases. It does not communicate directly with the database. It uses a database driver that acts as an interpreter to receive data from the database [12].

Figure 2. This is how it works today
Because the web browser requests data from a server that then returns the result, the traffic over the net can become large. This can lead to congestion which then leads to long transfer time. And the fact that you get the result from the server means that you always have to be connected to the internet. Another performance problem lies on the server. When requesting a page, the server needs to make some calculation. If you have a lot of users making requests at the same time the server can get heavily loaded.

A thick client provides a lot of functionality which is not possible on a web application. If you decide to implement a web application you will have to consider that you will lose some functionality because of the limitations of the web.

The main advantage of a web based solution is that it is centralized. This has its advantages in easy update and deployment. The only requirements on the clients are that a web browser is installed and that the clients have an internet connection. The hardware on the server is often more powerful than the average client.

Another advantage is that a web application is platform independent. The same software can be accessed through a web browser regardless of the client’s operative system.

2.3.3 Web or not from a Business Intelligence perspective

2.3.3.1 Standalone BI application

A standalone or thick BI application uses the client/server model. The client works as a standalone computer and works with the obtained result sets of data.

A thick client system contains the client with the application and a database. The client obtains the data from the database and can then work with it locally. The client calculates and forms the data and sends it back to the database. Because the data is manipulated locally the traffic over the network is reduced. The model is illustrated in figure 3 [13].

![Thick client model](image)

Figure 3. Thick client model

An example of how an interaction between the server and a client works in a standalone BI application:

1. The user logs into the application on the client.
2. The user executes a request. The application submits the SQL statement to the database.
3. The database receives the user request and passes it to the RDBMS.
4. The RDBMS analyses the SQL statement and performs the task.
5. The result set is sent back to the client.
6. The application receives the data and can format and analyze it.

2.3.3.2 Web based BI application

If you choose to implement your BI application on the web, it also uses the client/server model. But in this case the server does the computation and then presents the data to the client. The server must be more robust and have more computational power than in the case of a standalone application. The client on the other hand does
not have to be as robust as the client in a thick client solution. To illustrate how the web implementation works, see figure 4 [13].

![Diagram of web-based BI application](image)

**Figure 4.** Web based BI application

There are more steps in this model than in the previous to get the data. To get the data to the client, these steps are performed:

1. The user logs onto the system and starts a web browser.
2. The user navigates to a BI web site.
3. The web server displays the user interface for the BI application.
4. When a request is send from the web browser to the web server, the web server submit this request to the application server.
5. The application server submits the SQL statement to the database.
6. The database receives the request and passes it to the RDBMS application.
7. The RDBMS application analyses the SQL statement and performs the task.
8. The result set is sent back to the application server.
9. The application server formats the result and generates a HTML page.
10. The web server displays the HTML page to the user via the web browser.
3 METHOD
To see if a web application is better, or worse than a standalone application two different solutions had to be created. The application will be used for restructuring of data in a Data Warehouse in appendix C. The two different solutions were then compared to each other according to:

- Implementation (implementation analysis)
- Performance (performance testing)
- Security (literature study)
- Usability & Layout (interviews)

3.1 Implementation
To compare a web based and a standalone application according to size and the time it took to implement we will study code from the two different implementations and compare the amount of time it took to implement them. We will look at a number of classes the two solutions and compare these to each other according to size.

3.2 Performance
If you want the application to have good performance should you use the web based or the standalone implementation? In order to see this we will look at the different solutions according to response time. To test the performance Visual Studio 2005 Team Systems performance test will be used. This test tool works on both solutions which make it easy to compare the test results. To test an application you have to make a test project for that application. The application that is to be tested is started via the test project and the function calls in the test scenario are recorded. The test scenario is as follows:

- Start the application.
- Log on to the application.
- The application starts in the treewindow.
- Choose a dimension.
- Move a node within the tree structure (cut and paste).
- Move a node from the tree structure to the list (cut and paste).
- Move a node from the list to the tree structure (cut and paste).
- Create a new dimension…
  - …with a new root.
  - …with an existing root.
  - …from an existing dimension (copy).
- Delete a dimension.
- Create a new member.
- Update a member.
- Delete a member.

The result will only show the calculation time. This means that the transfer time over the network isn’t included in the result. This can make the web application seem faster than it is in reality.

3.3 Security
If the customer wants to have a secure application should you choose a standalone or a web based solution? Is it harder to secure either of the solutions or is it
approximately the same steps you have to take in order to secure them. The security is hard to test therefore some security vulnerabilities will be discussed and solutions will be given to help securing the applications.

3.4 Usability & Layout

When it comes to comparing the layout and usability of a program it becomes a little bit tricky. It is very individual if you like a layout or if you think that a program is user friendly. Therefore we will let some people work with both solutions and then answer some question regarding design and usability. The first layout that will be developed is the standalone applications layout. The development of the web based layout is supposed to be similar to the standalone layout.

The question form consisted of six questions with five options on each question, Useless, very bad, bad, good and very good where useless is 1 and very good is 5. Each question was to be answered for both the standalone and web application. The form had these questions:

1. Look – What do you think about the appearance of the application?
2. Feel – How is the feeling when working with the application?
3. Learnability – Is it easy to learn how to use the application?
4. Stability – How does the application handles errors?
5. Performance – How was the performance of the application?

The form can be found in appendix D.
4 RESULTS

4.1 Implementation

The solutions are a part in a Business Intelligence solution and are used for restructuring of dimensional data in the storage area. The data is stored according to the database diagram in appendix E which is a part of the whole Data Warehouse database structure (Appendix C). The two different solutions are a web based and a standalone application which where supposed to have the same functionality. But due to the lack of functionality in the web based solution there are some differences.

The applications are built upon the same logic. For a detailed description of how the logic works see appendix F. The database handler is designed to work on SQL-server, Oracle and DB2. This makes the applications very flexible and easy to expand. Another property of the solutions is that all SQL statements are not hard coded, but saved in an XML file (Appendix G). This makes it easy to change the SQL statements as long as you do not change the database structure.

When we started designing the application we agreed on making it a layered solution. The code that handled the logic, such as database and file handling, was made as an assembly file (.dll) to make it accessible from both solutions. It was only the GUI (Graphical User Interface) and its functionality that differed between the two solutions. From the beginning we thought we could implement the same functionality in both the web and standalone solution.

We began implementing the logic, such as database connection and file handling and made it applicable on any GUI. As we proceeded with our implementation we noticed that the user interface on the web had its limitations, this is because it is a so called thin client where it’s hard to implement a lot of functionality. In our case we could not create a solution with drag and drop. We found a solution to that problem by making the standalone application available on a web page. This led to full functionality but it is not a great solution for the web. When accessed the whole program was downloaded to the client and then executed as a thick application inside the web browser. The idea behind a web based solution was lost. The fast and easy accessibility disappeared and the functionality was moved from the server to the client. If you choose to use this type of implementation you will have to look at the code access security in the .NET framework and get an SSL certificate [16].

The only real measurement that was made on the two applications regarding implementation was to count lines of code and the time it took to implement. The result is presented in table 1 on the next page.
The time it took to implement the two solutions is presented in table 2. We will only compare the implementation of the two GUI's. This is the only thing that was implemented separately because they use the same logic. Lead time is longer in the standalone implementation. This is because the logic was developed in parallel with the standalone GUI.

<table>
<thead>
<tr>
<th>Class</th>
<th>LOC</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatabaseHandler</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>Dimension</td>
<td>131</td>
<td>131</td>
</tr>
<tr>
<td>DimensionHandler</td>
<td>155</td>
<td>155</td>
</tr>
<tr>
<td>FileHandler</td>
<td>132</td>
<td>132</td>
</tr>
<tr>
<td>InputHandler</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>ListHandler</td>
<td>497</td>
<td>497</td>
</tr>
<tr>
<td>Node</td>
<td>214</td>
<td>214</td>
</tr>
<tr>
<td>TreeHandler</td>
<td>218</td>
<td>218</td>
</tr>
<tr>
<td>Typ</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>User</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>UserHandler</td>
<td>44</td>
<td>44</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GUI</th>
<th>LOC</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoginWindow</td>
<td>151</td>
<td>87</td>
</tr>
<tr>
<td>Mainform</td>
<td>528</td>
<td></td>
</tr>
<tr>
<td>NewDimensionWindow</td>
<td>460</td>
<td>322</td>
</tr>
<tr>
<td>NewMemberWindow</td>
<td>220</td>
<td>112</td>
</tr>
<tr>
<td>ObjectWindow</td>
<td>487</td>
<td>307</td>
</tr>
<tr>
<td>TreeWindow</td>
<td>838</td>
<td>699</td>
</tr>
<tr>
<td>error</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>4331</td>
<td>3205</td>
</tr>
</tbody>
</table>

**Table 1.** Lines of code

**Table 2.** Implementation time
4.2 Performance

The discussion in chapter 2 says that the performance is better in a standalone application. Is this true or not? To prove this the two applications was tested in relation to performance. Because the two solutions do not have the exact same functionality the test could not be exact. But it still gives a hint to which solution that is best regarding performance. The applications were tested according to the scenario in the method and the summary of the test is shown in figure 5 and 6.

**Performance Report Summary**

**Most Called Functions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Calls</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>app.Node.get_node.kld</td>
<td>216152</td>
<td>24.500</td>
</tr>
<tr>
<td>System.Collections.IEnumerator.MoveNext</td>
<td>215241</td>
<td>24.396</td>
</tr>
<tr>
<td>System.Collections.IEnumerator.get_Current</td>
<td>213911</td>
<td>24.246</td>
</tr>
</tbody>
</table>

**Functions With Most Individual Work**

<table>
<thead>
<tr>
<th>Name</th>
<th>Time</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>app.TreeWindow.insertNode</td>
<td>192.66759</td>
<td>38.625</td>
</tr>
<tr>
<td>System.Collections.IEnumerator.get_Current</td>
<td>70.144898</td>
<td>14.041</td>
</tr>
</tbody>
</table>

**Functions Taking Longest**

<table>
<thead>
<tr>
<th>Name</th>
<th>Time</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>app.TreeHandler.saveCopiedDimension</td>
<td>1146.592635</td>
<td>2.799</td>
</tr>
</tbody>
</table>

*Figure 5. Standalone performance report summery*

**Performance Report Summary**

**Most Called Functions**

<table>
<thead>
<tr>
<th>Name</th>
<th>Number of Calls</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>app.Node.get_node.kld</td>
<td>643498</td>
<td>24.480</td>
</tr>
<tr>
<td>System.Collections.IEnumerator.get_Current</td>
<td>644935</td>
<td>24.420</td>
</tr>
</tbody>
</table>

**Functions With Most Individual Work**

<table>
<thead>
<tr>
<th>Name</th>
<th>Time</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>treeWindow.insertNode</td>
<td>448.499265</td>
<td>40.715</td>
</tr>
<tr>
<td>System.Collections.IEnumerator.get_Current</td>
<td>161.549565</td>
<td>14.665</td>
</tr>
</tbody>
</table>

**Functions Taking Longest**

<table>
<thead>
<tr>
<th>Name</th>
<th>Time</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>app.TreeHandler.saveCopiedDimension</td>
<td>1075.808804</td>
<td>17.178</td>
</tr>
<tr>
<td>treeWindow.insertNode</td>
<td>600.226596</td>
<td>9.584</td>
</tr>
<tr>
<td>app.ListHandler.getlist</td>
<td>485.123212</td>
<td>7.746</td>
</tr>
</tbody>
</table>

*Figure 6. Web performance report summery*

These summaries show which functions that are most used in the applications and how many times they are called. It also shows which of the functions that made most individual work and which functions that takes the longest time to execute.

As seen in figure 5 and 6 the number of calls on the methods that is called the most is approximately three times greater on the web application than on the standalone. From the report summary we also see that it is the same functions in both implementations that are called the most.
The test result from the test scenario is presented in the following table. It shows the different times of the steps in the scenario. The result is also plotted as a graph in figure 7. Figure 8 shows the time for all calls made by each function and as seen in the figure the web application has more calls for almost every function which leads to longer execution time.

<table>
<thead>
<tr>
<th>Event</th>
<th>Function name</th>
<th>Standalone Time (ms)</th>
<th>Web Time (ms)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log on</td>
<td>loginButton_Click</td>
<td>1571,666</td>
<td>292,854</td>
<td>The standalone application takes longer time to login the user because it loads the whole application. On the web, it's just the page that it's redirected to that is loaded.</td>
</tr>
<tr>
<td>Change dimension</td>
<td>SelectedIndexed_Changed</td>
<td>562,404</td>
<td>482,277</td>
<td>This occurs 7 times on the web but only 2 times in the standalone application.</td>
</tr>
<tr>
<td>Move inside tree structure</td>
<td>cut + paste</td>
<td>41,385</td>
<td>482,786</td>
<td>The total time for both cutting and pasting a node.</td>
</tr>
<tr>
<td>Move from tree structure to list</td>
<td>cut + paste</td>
<td>43,825</td>
<td>256,914</td>
<td>The total time for both cutting and pasting a node.</td>
</tr>
<tr>
<td>New dimension</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New root</td>
<td>saveButton_Click</td>
<td>162,205</td>
<td>275,561</td>
<td></td>
</tr>
<tr>
<td>Existing root</td>
<td>saveButton_Click</td>
<td>39,009</td>
<td>43,585</td>
<td></td>
</tr>
<tr>
<td>Copy entire tree</td>
<td>saveButton_Click</td>
<td>1425,826</td>
<td>1123,951</td>
<td></td>
</tr>
<tr>
<td>Delete a dimension</td>
<td>removeDimension</td>
<td>486,276</td>
<td>49,790</td>
<td>When deleting a dimension the standalone version shows a messagebox which takes little more than 200 ms to load and it takes another 200 ms to clear the treeview.</td>
</tr>
<tr>
<td>Create a new member</td>
<td>addNode</td>
<td>24,825</td>
<td>136,157</td>
<td></td>
</tr>
<tr>
<td>Update a member</td>
<td>updateNode</td>
<td>57,179</td>
<td>164,186</td>
<td></td>
</tr>
<tr>
<td>Delete a member</td>
<td>removeNode</td>
<td>507,651</td>
<td>185,664</td>
<td>When deleting a member the standalone version shows a messagebox which takes little more than 200 ms to load</td>
</tr>
</tbody>
</table>

**Table 3.** The test scenario report
**Figure 7.** Chart of the test result for single calls

**Figure 8.** Chart of the time for all calls
4.3 Security

An easy accessible application such as a web based application leads to more security problems. As a result hackers can use the web based application to penetrate corporate systems and access restricted data.

Many of the most dangerous security problems today are not worms and viruses, and not vulnerabilities on the server side but vulnerabilities in the web based applications themselves.

There are some fundamental differences between a standalone and a web based application. In a thick client there is no way to alter the messages it sends to its database. But a web browser is very easy to manipulate.

On a standalone application a lot of the validation is done on the client side to reduce network traffic and improve server performance. Web applications try to do the same with help of JavaScript and HTML, but HTML can be changed and JavaScript can be disabled. This places all the input validation on the server side and it is hard to check every input for malicious values.

In client-server (standalone) environments, a continuous session between the clients and the server is maintained. Once the user logs into the application an unbroken connection feeds the user with information. In web applications there is no session; a user requests a page and then loses the connection to the server until a new page is requested. The way a web server keeps track of its user is by session cookies. The cookie resides on the client side and can be changed to hack the application [14].

4.4 Usability & Layout

The design of the standalone and web application is made to look similar. This helps the user to recognize and use both solutions. If a user has worked with one of the solutions it is easy for this person to start using the other. We started with the implementation of the standalone GUI and then proceeded with the web GUI and made it look similar to the standalone application.

The main difference between the two solutions is the drag-and-drop functionality. The solution for the web was to make a “cut and paste” functionality, which is available in both solutions. The application has two main windows, objectwindow and treewindow. The objectwindow works almost similar in both applications (figure 9).
The treewindow is where the two solutions differ. The standalone application is a thick client. This makes it possible to have heavy functionality, such as drag and drop. With this feature the usability is increased for the standalone application. It is possible to make a tree structure draggable on a webpage but it would not work on our solution because the items should be dragged between two objects (tree and list). Even if the standalone version has more functionality the two solutions are almost similar in layout (figure 10).
The result of the survey about usability and layout is presented in table 4 and has been answered by eight persons. The test people consisted of the customers, developers and people without technical background.

<table>
<thead>
<tr>
<th></th>
<th>Standalone</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look</td>
<td>4,9</td>
<td>3,6</td>
</tr>
<tr>
<td>Feel</td>
<td>4,8</td>
<td>3,8</td>
</tr>
<tr>
<td>Learnability</td>
<td>5,0</td>
<td>3,8</td>
</tr>
<tr>
<td>Stability</td>
<td>4,7</td>
<td>2,8</td>
</tr>
<tr>
<td>Performance</td>
<td>4,5</td>
<td>3,8</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>4,8</td>
<td>3,8</td>
</tr>
<tr>
<td>Overall</td>
<td>4,8</td>
<td>3,6</td>
</tr>
</tbody>
</table>

Table 4. Usability & Layout results
5 ANALYSIS

5.1 Implementation

It is not an easy task to compare the two solutions. The architecture differs and the functionality as well. But if we start to look at the lines of code in each solution we can see that the web application has about 1000 lines of code less than the standalone. The logic is the same in both solutions so the lines of code don’t differ in these classes.

The treewindow class is one of the classes that differ most. The thing with the treewindow is that it has more functionality in the standalone application. Most of the extra lines of code comes from the drag and drop functionality which makes it possible for the user to drag nodes between the tree structure and list instead of using cut and paste.

The standalone application also has a MainForm which is a container for everything in the application. All panels and windows are opened from here and all menus and toolbars are created here. In the web application it is the web browser that is the container. This is the main reason to why the lines of code differ so much.

If we look at the time it took to implement the two GUI’s we can see that the standalone version took less time than the one on the web. This might look strange because it has more lines of code. But from our experience, both from this and other projects, it’s easier to get a functional look on a windows application than on a web page. You can also see that the lead time of the standalone GUI was longer. This is because we implemented some logic at the same time. If we had started with the web interface we would probably have longer lead time on that part than on the standalone.

From this we can say that it takes approximately the same time to implement both solutions. But you can implement more functionality in a standalone application during the same amount of time.

5.2 Performance

The performance test shows the time spent in different functions. It doesn’t show the time it takes for a web page to be sent over the internet and presented to the user. Therefore the test result lets us know how fast the computation is done on the server, in the case of the web application and how fast it is on the client when it comes to the standalone application.

If we compare the result in table 3 we can see that the events that works more with the database than with the GUI tends to be faster on the standalone than on the web application. For example, the event that creates a new member only passes the information about the new member to the database and it is faster on the standalone version.

In the event of removing a dimension the standalone application is much slower. It took almost 490 ms for the standalone application to do this but 200 ms of these went by when it created the message box that confirms your delete command. If you subtract these 200 ms you still see that it is 200 ms slower. This is because of the time it takes to add and remove the graphical representation of the nodes in the standalone application. To speed up the graphical work you can use a method called beginupdate in c#. This disables the object to draw graphic while it’s being updated. This method was implemented when the test was conducted but it wasn’t enough.

In some cases the web is faster but in reality it isn’t. If you add the time it takes for the web page to become visible to the user the total time will always be longer when comparing with a standalone application. On the web everything is reloaded more
often which leads to even more execution time as you can see in the performance summary reports in figure 5 and 6. In these figures you also see that it is the same functions that are used most but they are called roughly three times more in the web application. As seen in figure 8 the total time for the operations in the web application is longer for almost every function.

Most of the time the user will move nodes between the tree structure and the list, this makes the cut and paste functions more commonly used than the other functions. The cut and paste functions are faster in the standalone application as seen in figure 7 and 8.

5.3 Security

The first and most common security risk is the user input. This is the users’ way to interact with the application and therefore the most common way for a hacker to infiltrate a server/company. The securing of input strings also ensures that SQL injection is not possible. It is not enough to validate the user input on the client side of the application. The most important thing is to secure the inputs on the server side. This is because a web client can never be trusted; the user can turn off scripts or change them. The only thing with the client side validation that is good is the fact that it can handle simple input mistakes to reduce the number of times the server has to validate input. There are a lot of solutions on how to solve this problem, but the best way is to only allow a list of valid characters. It is also easier to check input against a list of valid characters then to strip unwanted characters. This is because it is difficult to determine all possible malicious characters. In appendix H you can see how we solved this.

Another big dilemma is the authentication. People who want access to an application can steal passwords and gain access to the application by sniffing up the user information. Using the sniffing technique an attacker can get the username and password of other users. To solve this problem the application should send the login id and the hashed password to the database server. At the server stored procedure should be used to authenticate the user. To make it more secure the traffic between client and server should be encrypted.

Both web and standalone applications share some security vulnerabilities. The main difference is that a web application can be reached from the internet which allows almost anyone to access and attack it.

A big security risk within a web application is Cross Site Scripting (XSS). It is perhaps the most common web application vulnerability. XSS refers to an attack where the user submits malicious HTML, which can include script code, to a dynamic web application. With this type of attack a hacker can “steal” another user’s session and use it before the session cookie expires. The attacker can create a malicious script that other user unknowingly executes when viewing dynamically generated pages which generates content provided by the attacker. To limit this security risk you can encode HTML meta characters explicitly. For example: replace < with &lt; and # with &#35;. You must see to that the application performs validation on all headers, cookies, query strings, form fields and hidden fields [17] [18].

The conclusion that can be drawn from this is that a web application suffers from more security vulnerabilities than a standalone application. However, securing the web application is worthless if the web server has not been secured. To secure a web application the environment that it works in has to be secure. The security begins with having secure servers and networks. It is important to have the latest security patches for the operating system and software on the web server. The web server has to be behind a well maintained firewall and monitored for hacking intrusion.
5.4 Usability & Layout

As seen in the results it is the standalone version that has the best usability and layout. The standalone application is easier to learn and the layout is more appealing to the user. It also feels more stable and has better performance than the web application. The standalone implementation is better in usability mainly because it has more functionality which makes it easier to learn and the user can recognize the functionality and features from other applications.

5.5 Validity

To prove that our result is correct in our research questions the result will be discussed regarding validity. To begin with, we used the same development procedure for both implementations (we used the waterfall model). This ensures that the two different applications were developed in the same way. They were also developed on the same framework, the .NET framework, and the same development environment was used (Visual Studio .NET). If you compare with other types of frameworks like J2EE, you will get the same result as long as you use the same development model and environment.

To strengthen our performance test results we used the same test procedure/scenario and the same test tool (Visual studio 2005 Performance test) for both applications. To compare two applications the same test tool must be used to ensure validity. But even if the same test tool is used it’s hard to ensure that the validity of the results because it’s difficult to compare two different types of application architectures.

When it comes to the comparison of security it’s not easy to see if the result is valid or not. There are always new security threats emerging. Even if you secure your application today, it may not be secure in the future. The discussion about application security is held as general as possible to ensure the external validity.

Because the people who tested usability and layout had different backgrounds the test wasn’t directed towards this type of applications and it is possible to perform the test on any application.

The tests have been performed to be as general as possible to ensure that the result is applicable when it comes to comparing web and standalone applications. The result is applicable not just on BI solutions like the one used here, but on all kinds of applications that have similar functionality.
6 CONCLUSION

If you are looking for a fast and secure application with a lot of functionality you should choose the standalone implementation. This implementation lets the program execute with the full power of the client machine. It’s faster than a web application but the thing that really makes it better than a web solution is the fact that it has more functionality which increases the usability.

If your main purpose is to reach many users, and have an easy maintained and easy deployed application that can be accessed from remote computers with different operating systems you should choose the web based solution. It takes longer to load/execute a web page but in return the application is easy to access and maintain. This makes the web based solution suitable if you have many users working from different locations.

It’s often harder to secure a web application because it’s not just the application that has to be secure but the server as well. It’s also easier for hackers to access a web page than a program on a computer. This makes it more important to consider security issues when creating a web application.

There is no answer to the question: web or standalone application? It’s up to the developers to look at the requirements and then decide which of the solutions that suits the customer best. To be able to decide which solution to use it is important for the developers to discuss with the customers what they want to be able to do with their application.
7 REFERENCES

[1] Manjarekar C, To BI or not to BI, White paper, DMReview


8 APPENDIX

8.1 Appendix A – The different steps in BI

Step 1

The first step is the company’s source systems. A company can have one or more source systems. As described in the figure above the source system can be of different kind. It is possible to make a BI solution for any kind of source system.

The source system only contains raw data. When the customer makes its requirements on the BI solution the source systems data must be validated to see if the requirements can be fulfilled.

Step 2

Extraction, Transformation and Loading (ETL) is performed to get correct data to the Data Warehouse. For the data to fit the Data Warehouse it may have to be reformatted. As a simple example, the source system has stored date as YY/MM/DD and the DW stores it as DD/MM/YY. The data that does not have any meaning for the DW is removed and faulty data is cleansed. At the end of this step the correct and quality assured data is loaded into the Data Warehouse [5].

Step 3

This is where the correct and quality assured data is stored, the so called Data Warehouse. The stored data is the foundation for the rest of the BI solution and it is in this step our application will be used.

Step 4

The access area is sometimes called Data Marts. Data Marts are usually smaller than the Data Warehouse and focus on a particular subject or department. The database design of the data mart is typically built around the star structure that is optimal for the needs found in the department. To be able to shape the star structure, the requirements of the department must be gathered and analyzed. The conclusion of this is that the data mart holds the data from the Data Warehouse that is interesting for a department.

Step 5

Online analytical processing (OLAP) is what comes into mind when talking about the functionality in Business Intelligence. OLAP is a piece of software technology that enables the users to get insight into data thought fast and consistent access to a variety
of possible views of the data in the data marts/Data Warehouse. OLAP is characterized by these different functionalities:

- Calculation and modeling applied across dimensions.
- Trend analysis over sequential time periods.
- Slicing subsets for on-screen viewing.
- Drill-down to deeper levels of consolidation.
- Reach-through to underlying detail data.
- Rotation to new dimensional comparisons in the viewing area.

OLAP helps the user analyze historical data in various “what if” scenarios. OLAP also helps the user to combine the company’s information through personalized viewing. This is achieved through use of an OLAP Server [15].

**Step 6**

The data that is analyzed in the previous step can be presented to the user in different ways. The result can for example be viewed as a graph or a table. It can be published over the internet via a web site or sent to you by mail.
8.2 Appendix B – Star schema
8.3 Appendix C – The Data Warehouse model
8.4 Appendix D – Usability and Layout form

Usability and Layout form
X – Useless
1 – Very bad
2 – Bad
3 – Good
4 – Very good

Please answer these questions according to your own thoughts. If you have any comment please write them down in the comment box. A description of each question is found on the last page.

<table>
<thead>
<tr>
<th>Look</th>
<th>Standalone</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feel</th>
<th>Standalone</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Learnability</th>
<th>Standalone</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stability</th>
<th>Standalone</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance</th>
<th>Standalone</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Satisfaction</th>
<th>Standalone</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question descriptions

Look – What do you think about the appearance of the application?

Feel – How is the feeling when working with the application?

Learnability – Is it easy to learn how to use the program?

Stability – How does the application handle errors (application and input errors)?

Performance – How was the performance of the application?

Satisfaction – Your overall opinion.
8.5 Appendix E – Database diagram
Appendix G – Example of a XML-file

```xml
<?xml version="1.0" encoding="utf-8" ?>
<MSSQL>
    <SQL>
        <statement id="0">SQLserver</statement>
        <comment>Database</comment>
    </SQL>
    <SQL>
        <statement id="1">SERVER=(local); database=prod_dw_boverket; Trusted_Connection=True;</statement>
        <comment>Connection string</comment>
    </SQL>
    <SQL>
        <statement id="2">select s.lkid, s.lkanfaderid, a.beskrivning, s.lkstruktypid, s.regtid, s.regId from ba_lk a, ba_lkstruktur s
        where a.lkid = s.lkid and s.lkstruktypid = STRUKTURTYPIDVALUE ORDER BY a.beskrivning</statement>
        <comment>Treedstructure</comment>
    </SQL>
    <SQL>
        <statement id="3">DELETE from ba_lkstruktur where lkstruktypid = STRUKTURTYPIDVALUE</statement>
        <comment>delete from the tree structure</comment>
    </SQL>
    <SQL>
        <statement id="4">select a.regid, a.regtid, a.lkid, a.lkkey, a.beskrivning, a.from_datum, a.tom_datum, a.lktypid, a.map_id,
        d.ext_beskrivning from ba_lk a, ba_lk_attribut d where a.lkid = d.lkid ORDER BY a.beskrivning</statement>
        <comment>the list in the objectwindow</comment>
    </SQL>
    <SQL>
        <statement id="5">select a.lkid, a.beskrivning from ba_lk a, ba_lk_attribut d where a.lkid = d.lkid and not exists (select *
        from ba_lkstrukturtyp b, ba_lkstruktur c where a.lkid = c.lkid and b.lkstruktypid = STRUKTURTYPIDVALUE and
        b.lkstruktypid = c.lkstruktypid) ORDER BY a.beskrivning</statement>
        <comment>the list in the treeview</comment>
    </SQL>
    <SQL>
        <statement id="6">select * from userTable where userName='USERNAMEVALUE'</statement>
        <comment>Authenticate user</comment>
    </SQL>
    <SQL>
        <statement id="7">select * from ba_lktyp</statement>
        <comment>load the types</comment>
    </SQL>
    <SQL>
        <statement id="8">select * from ba_lkstrukturtyp</statement>
        <comment>Loads the dimensions</comment>
    </SQL>
    <SQL>
        <statement id="9">insert into ba_lkstruktur values('REGIDVALUE', REGTIDVALUE, UIDVALUE,
        ANFADERIDVALUE, STRUKTURTYPIDVALUE)</statement>
        <comment>Save the tree</comment>
    </SQL>
    <SQL>
        <statement id="10">INSERT INTO BA_lkSTRUKTURTYP (regid, regtid, beskrivning, from_datum, tom_datum, map_id)
        VALUES ('REGIDVALUE', REGTIDVALUE, 'BESKRIVNINGVALUE', REGTIDVALUE, TOMDATUMVALUE,
        MAPIDVALUE)</statement>
        <comment>Add dimension</comment>
    </SQL>
    <SQL>
        <statement id="11">select max(lkstruktypid) from BA_lkSTRUKTURTYP</statement>
        <comment>Get last dimension</comment>
    </SQL>
    <SQL>
        <statement id="12">INSERT INTO BA_lkSTRUKTUR VALUES ('REGIDVALUE', REGTIDVALUE, UIDVALUE,
        ANFADERIDVALUE, STRUKTYPIDVALUE)</statement>
        <comment>Add node to the tree</comment>
    </SQL>
    <SQL>
        <statement id="13">DELETE FROM BA_lkSTRUKTURTYP WHERE lkSTRUKTYPID =
        STRUKTYPIDVALUE</statement>
        <comment>delete dimension</comment>
    </SQL>
    <SQL>
        <statement id="14">DELETE FROM BA_lkSTRUKTUR WHERE lkSTRUKTYPID =
        STRUKTYPIDVALUE</statement>
        <comment>delete all nodes in one dimension</comment>
    </SQL>
</MSSQL>
```
<statement id="15">INSERT INTO BA_lk (regid, regtid, lkkey, beskrivning, from_datum, tom_datum, lktypid, map_id)
VALUES ('REGIDVALUE', REGTIDVALUE, 'KEYVALUE', 'BESKRIVNING', REGTIDVALUE, TOMDATUMVALUE, TYPIDVALUE, MAPIDVALUE)</statement>

<comment>Adds a new member</comment>

<statement id="16">INSERT INTO BA_lk_ATTRIBUT (regid, regtid, lkid, ext_beskrivning) VALUES ('REGIDVALUE', REGTIDVALUE, UIDVALUE, 'EXTERNBESKRIVNING')</statement>

<comment>Adds a member attribute</comment>

<statement id="17">UPDATE BA_lk SET regid = 'REGIDVALUE', regtid = REGTIDVALUE, beskrivning = 'BESKRIVNINGVALUE', lkkey = 'KEYVALUE', lktypid = TYPIDVALUE WHERE lkid = UIDVALUE</statement>

<comment>update a member</comment>

<statement id="18">UPDATE BA_lk_ATTRIBUT SET regid = 'REGIDVALUE', regtid = REGTIDVALUE, ext_beskrivning = 'EXTERNBESKRIVNINGVALUE' WHERE lkid = UIDVALUE</statement>

<comment>update the members attribute</comment>

<statement id="19">SELECT * FROM BA_lkSTRUKTUR WHERE lkid = UIDVALUE and lkanfaderid = UIDVALUE</statement>

<comment>Check if a node is a root</comment>

<statement id="20">DELETE FROM BA_lk_ATTRIBUT WHERE lkid = UIDVALUE</statement>

<comment>delete a members attribute</comment>

<statement id="21">DELETE FROM BA_lkSTRUKTUR WHERE lkid = UIDVALUE</statement>

<comment>remove a node</comment>

<statement id="22">DELETE FROM BA_lk WHERE lkid = UIDVALUE</statement>

<comment>remove a node</comment>

<statement id="23">select a.regid, a.regtid, a.lkid, a.lkkey, a.beskrivning, a.from_datum, a.tom_datum, a.lktypid, a.map_id, d.ext_beskrivning from BA_lk a, BA_lk_ATTRIBUT d where a.lkid = (select max(lkid) from BA_lk)</statement>

<comment>get the last member</comment>

<statement id="24">select * from BA_lkSTRUKTUR</statement>

<comment>gets the structure</comment>

<statement id="25">DELETE FROM BA_lkSTRUKTUR WHERE lkanfaderid = ANFADERIDVALUE</statement>

<comment>delete children of a node</comment>

<MSSQL>
8.8 Appendix H – InputHandler

using System;

namespace dimBI
{
    /// <summary>
    /// InputHandler secures a string, in order to secure the user inputs.
    /// </summary>
    public class InputHandler
    {
        public InputHandler()
        {
        }

        /// <summary>
        /// trimString removes white spaces in the beginning and the end of a string.
        /// </summary>
        /// <param name="inputString">the string that should be trimmed</param>
        /// <returns>the trimmed string</returns>
        public string trimString(string inputString)
        {
            string trimString = inputString;
            trimString = trimString.Trim();
            return trimString;
        }

        /// <summary>
        /// checkChar checks if the string contains unwanted characters
        /// </summary>
        /// <param name="inputString">the string that should be checked</param>
        /// <returns>a string with the unwanted characters</returns>
        public string checkChar(string inputString)
        {
            string notAllowed = "";
            foreach(Char c in inputString)
            {
                if((c >= 40 && c <= 41) || (c >= 43 && c <= 59) || (c >= 63 && c <= 90) || (c >= 97 && c <= 122) || c == 'å' || c == 'ä' || c == 'ö' || c == 'Å' || c == 'Ä' || c == 'Ö' || (c >= 32 && c <= 33) || c == 38 || c == 95)
                    {
                        //Allow: A-Z, a-z, 0-9, & ( ) _ + / SPACE . , ! : ; ? @
                        //å ä ö Å Ä Ö
                    }
                else
                {
                    if(!containsChar(notAllowed, c))
                    {
                        notAllowed += c;
                    }
                }
            }
            return notAllowed;
        }
    }
}
/// <summary>
/// containsChar checks if the char is already in the notAllowed string, in order to only get one copy of each unwanted character.
/// </summary>
/// <param name="notAllowed">the string that should be checked</param>
/// <param name="c">the char that should be checked</param>
/// <returns>true if notAllowed contains the character</returns>
private bool containsChar(string notAllowed, Char c)
{
    bool result = false;
    foreach(Char d in notAllowed)
    {
        if(c.Equals(d))
        {
            result = true;
        }
    }
    return result;
}
8.9 Appendix I – Glossary

**Business Intelligence (BI)**–a collection of technologies that allows for gathering, storing, accessing and analyzing data to help business users make better decisions.

**Data Warehouse (DW)**–a collection of databases, data tables and mechanisms to access the data on a single subject.

**Standalone application**–an application that operates on its own in a desktop environment.

**Web application**–a software program that uses HTTP for its core communication protocol and delivers web-based information to the user in the HTML-language.

**Thick client**–see standalone application.

**Thin client**–see web application.

**RDBMS**–Rational Database Management System is a structured computer information storage and retrieval system where the basic unit is a Table with Rows and Columns.

**Data Mart**–A subset of a Data Warehouse, for use by a single department or function.

**OLAP**–On-line Analytical Processing

**Star Schema**–One of several models for a Data Warehouse. A star schema has a central fact table, connected to a set of dimension tables.