

MicroSCADA project documentation database

Karolina Kolam

Bachelor's thesis
Electrical Engineering
Vaasa 2014



BACHELOR'S THESIS

Author: Karolina Kolam

Degree Programme: Electrical Engineering

Specialization: Electrical Power Engineering

Supervisor: Susanne Österholm

Title: MicroSCADA project documentation database

Date 22.04.2014

Number of pages 54

Appendices 3

Summary

This bachelor's thesis was commissioned by ABB Power Systems, Network Management. The purpose of the thesis was to create a database for storing MicroSCADA project information. Also a usable tool for writing data to the database and creating reports was to be made. Before this thesis work, all the information about the department's projects was stored as single text documents. A database would collect all information in one place and also store the information for a long time. Also it would provide security so that no data is lost.

The theoretical section of this thesis is about database history and relational databases. Also the programming language SQL is treated. The database was done in Access and the user interface in Excel. The communication between the two programs was handled by the database programming language SQL and the programming language VBA.

The outcome was a hidden database and a user interface where the user can write new data to the database and also edit the data in the database. The user can search the database and as a result of the search print a report. The report has a standard layout to standardize all the project reports.

Language: English

Key words: database, SQL, relational databases, DBMS

EXAMENSARBETE

Författare: Karolina Kolam

Utbildningsprogram och ort: Elektroteknik, Vasa

Inriktningsalternativ/Fördjupning: Elkraft

Handledare: Susanne Österholm

Titel: Databas för dokumentation av MicroSCADA projekt

Datum

Sidantal

Bilagor 3

Abstrakt

Detta ingenjörarbete var beställt av ABB Power Systems, Network Management. Syftet med detta ingenjörarbete var att skapa en databas för dokumentering av information om MicroSCADA projekt. Ett lämpligt verktyg för att skapa rapporter och skriva ny data till databasen skulle också ingå. Före detta ingenjörarbete sparades all information som skilda textdokument. Med en databas kunde man samla all information på ett ställe för att arkiveras under en längre tid. Det förenklade framförallt sökprocessen men bidrog också med säkerhet så att ingen viktig information försvann.

Teoridelen i detta arbete handlar om databas historik och relationsdatabaser. Programspråket SQL behandlas också. Databasen gjordes i Access och ett användargränssnitt i Excel. Kommunikationen mellan dessa program sköttes med programspråket SQL och även programmeringsspråket vba.

Resultatet av detta arbete var ett system var användaren kan skriva ny data till databasen men också editera data i databasen. Användaren kan också söka i databasen och som resultat av sökningen kan användaren också välja att skriva ut en rapport på sitt sökresultat. Rapporten har en bestämd layout med rubriker och fonter för att skapa enhetliga rapporter.

Språk: Engelska

Nyckelord: databas, SQL, relations databas, DBMS

Table of Contents

1	INTRODUCTION	1
1.1	The Company	1
1.2	The Purpose of the Thesis	2
2	ASSIGNMENT	3
2.1	Scope	3
2.2	Choosing a Suitable User Interface.....	4
2.2.1	Excel-based User Interface	5
2.3	Tools.....	5
2.3.1	Access	6
2.3.2	Excel	6
2.3.3	Visual Basic for Applications	7
3	DATABASES.....	7
3.1	Storing Information.....	7
3.1.1	Database Systems	8
3.2	Database Environment	10
3.2.1	Client-Server Environment	11
3.2.2	Multi-Tier Environment.....	12
3.3	Database Architectures.....	14
3.3.1	Flat File Database.....	14
3.3.2	Hierarchical Database.....	15
3.3.3	Network Database	16
4	RELATIONAL DATABASE SYSTEM	17
4.1	Elements of the E/R Model	18
4.2	Basic Elements of a Relational Database	19
4.2.1	Primary Key	20
4.2.2	Index	21
4.3	Relationships	21
5	THE DATABASE LANGUAGE SQL.....	23
5.1	Simple Queries.....	25
5.2	Wildcards in SQL	25
5.3	Joins in SQL	26

6	DATA STORAGE	28
6.1	The Memory Hierarchy	28
6.1.1	The Cache	29
6.1.2	Main Memory.....	29
6.1.3	Virtual Memory and Secondary Storage	30
6.1.4	Tertiary Storage.....	30
6.1.5	Modifying Data	31
7	COURSE OF ACTION	32
7.1	Planning the System.....	32
7.2	Database Design	33
7.2.1	Relations and Relationships	33
7.2.2	Table Design	34
7.3	User Interface Design.....	35
7.3.1	Communication between Database and User Interface	36
7.3.2	Creating Reports and Searching the Database	37
8	FINAL RESULT.....	40
9	DISCUSSION	41
10	SOURCES.....	1
11	APPENDIX	3

1 Introduction

This is a thesis work done for the ABB, Power Systems, Network Management unit in Vaasa. The purpose was to create a database that would ease the making of hardware and software documentation for projects. A key factor was that it would be easy to search for components used in the projects. Therefore a database was to be created for storing the data and also an easy way to write the data to the database was needed.

1.1 The Company

ABB is the result of the merger of two companies, the Swedish ASEA and the Swiss Brown Boveri & Cie. The two companies merged in 1988 to form ABB. The company is now a global leader in power and automation technologies. Over 150,000 people work for ABB and the company operates in about 100 countries. It is based in Zurich, Switzerland and the company's shares are traded on the stock exchanges of Zurich, Stockholm and New York. (The ABB Group 2013)

The company is divided into five divisions that are organized by the customers and industries that they serve. The divisions are

- Power Products
- Power Systems
- Discrete Automation and Motion
- Low Voltage Products
- Process Automation

ABB operates in about 30 places around Finland and in Vasa you can find ABB in Stromberg Park. The name of the park comes from the company that used to operate on this location before it was bought by ASEA in 1987 and later became a part of the ABB Corporation. This thesis was written for ABB Power Systems, Network Management. (The ABB Group 2013)

1.2 The Purpose of the Thesis

All components of a project are documented while testing a project. The most important piece of information to be documented is the serial number of each component. Additional information such as customer, RAID system type and voltage level is also documented. This is done to speed up the fixing of errors since all the information about the components is to be found in the same place. If a component breaks, the customer simply needs to inform the serial number to the ABB project department to retrieve a new component that suits the purpose. By knowing as much as possible about the project, preparations can be made in Finland instead of spending several weeks on site.

The purpose of this thesis work was to create a database for storing information about hardware and software used in projects, such as serial numbers, product description etc. No database was used before this and all data was stored as single text documents on a hard drive and this made searching for specific components very difficult. For example if an intermittent fault for servers type X from manufacturer Y is discovered and all the servers of this type need to be replaced. The first step would be to locate all servers and in which projects they are in use. With the system that was in use before the database, searching for all the servers would be very troublesome and would take a lot of time. It would also be easy to miss servers since the system wasn't well organized and the number of project folders is very high. With a database the search for the servers would be much easier since it is possible to create a function that searches the database.

The database should work so that a report for the project can be created with a wizard in Excel and at the same time the data is exported to an Access database. For the user the report is the main outcome but the most important outcome is that the data is stored and easy to find later on. It is very easy to search a database compared to searching manually in folders and if the thesis meets its purpose, a simple button click will search the database for desired components.

2 Assignment

The database needed to be logically structured and easy to modify. The data stored in the database had to be easy to access and the search criteria flexible. A nice layout for every report had to be created. The flexibility of searching the database was one key aspect and another was always to generate good reports.

2.1 Scope

The scope of the thesis is illustrated in Figure 1. The yellow part of the figure is where the most important features of the thesis can be found and, after discussing the project and taking time and previous knowledge into consideration, it was decided to focus on the core features. The hardware and software specification reports were also a main focus. Other features could of course be added later on.

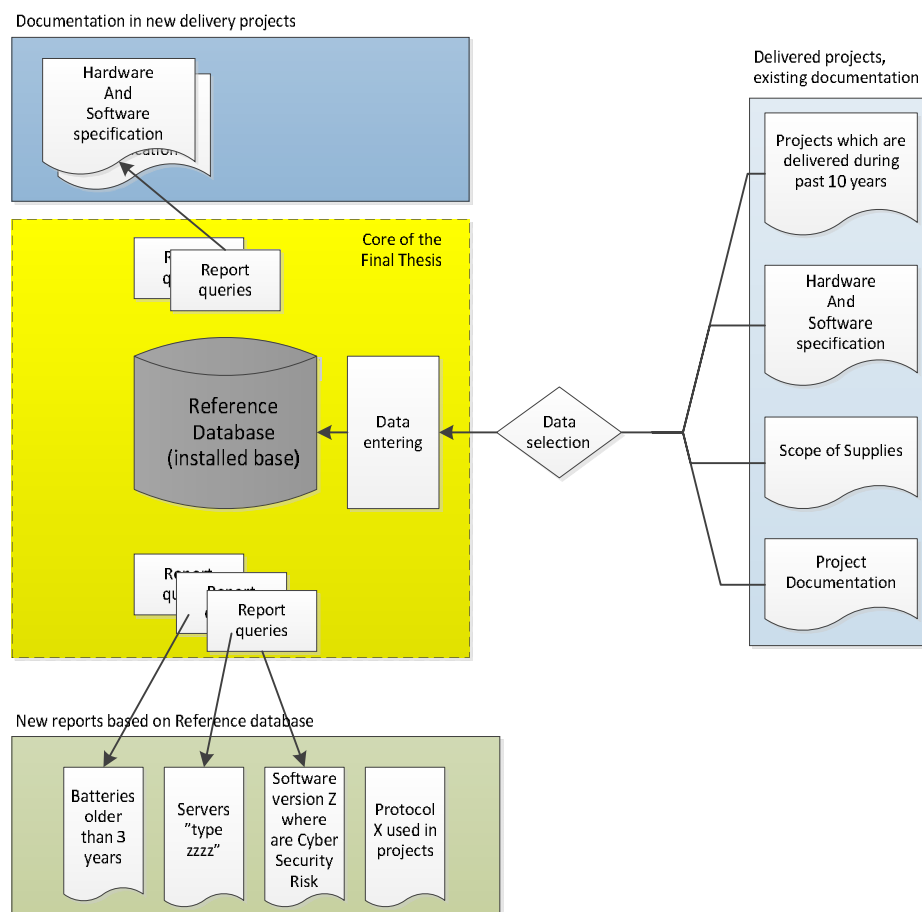


Figure 1. Scope of the thesis

2.2 Choosing a Suitable User Interface

There are several ways to create a user interface for this database. The easiest solution would be to continue using the text documents and then manually write the data to the database. But this would, assuming that people are lazy, lead to an almost empty database. A key aspect of this assignment was that the solution needed to be so good so that people would start using it, instead of generating the reports in the old way. If someone chooses to create a text document as the report, the data of the project will not be included in the database and when searching the database the outcome will be incomplete.

The simplest solution would be to create the reports in Access. The reports would have a nice layout and the report design would also be easy to edit. Also, inserting or deleting tables would be easy. When discussing different solutions with the people that will be the end users it turned out that Access was not a very familiar program and it was not even installed on all computers. Another key aspect was that it would be more effective if the users don't have access to the database, since this might lead to unwanted changes in the database. Therefore a program that communicates well with Access would suit well as user interface.

Excel was the natural choice since the two programs are designed to communicate and there are built in functions for this. Since both programs are Microsoft products, the programming language Visual Basic for Applications (VBA) was well suited for the assignment. Excel is also familiar to all users and the program is installed on all computers. Assignments that are similar to the thesis assignment have been done several times and almost all are done in Excel. This system though was the first system that had to communicate with an Access database.

2.2.1 Excel-based User Interface

A booking system for FAT-area tables was in use, which was created with the programming language VBA in an Excel sheet. This system works very well and is in use by the whole department. It is easy to use and since the user has very few alternatives there is no unnecessary saving of data. This system was used as inspiration for the database user interface. In order to create a usable user interface we decided that it would be best to create it in an Excel spreadsheet as a simple wizard, as done in the booking system. The user will have a few alternatives in every display and will proceed by clicking “Next”. The user is also able to edit an existing project by simply choosing the button for this purpose. For the user the report is the result, because when working on a project the report is all the user needs. It is later on the database could come in handy. The planning of the user interface was done very carefully to manage and solve problems before starting to create the user interface. While planning it was good to have a close dialog with the ones who are going to use the database because they pointed out issues and also came up with some solutions. The first visualisation of what the user interface should look like can be seen in Appendix 2 “Database user interface”.

2.3 Tools

For this thesis work two programs were mainly used. The database was created in Access and the user interface in Excel. The communication between the two programs was done with Visual Basic. It is possible to communicate between the two program’s predefined applications within the programs but since the operations are limited it was best to create the communication with code. By writing the code, one could decide exactly how the communication should work.

2.3.1 Access

Microsoft Access is a database management system. Access makes it very easy to create and manage databases. The databases created with Access are relational databases which is the most popular type of databases today. Also, creating the relationships is very handy since it is done by drag and drop functions. The first version of the program was released in 1992 as an evolved product of earlier programs Omega and Cirrus. When Office 95 was released Access became a member of the Office Professional Suite. A screenshot of Access can be seen in Figure 2. (Arvidsson 1999, pp 7-23)

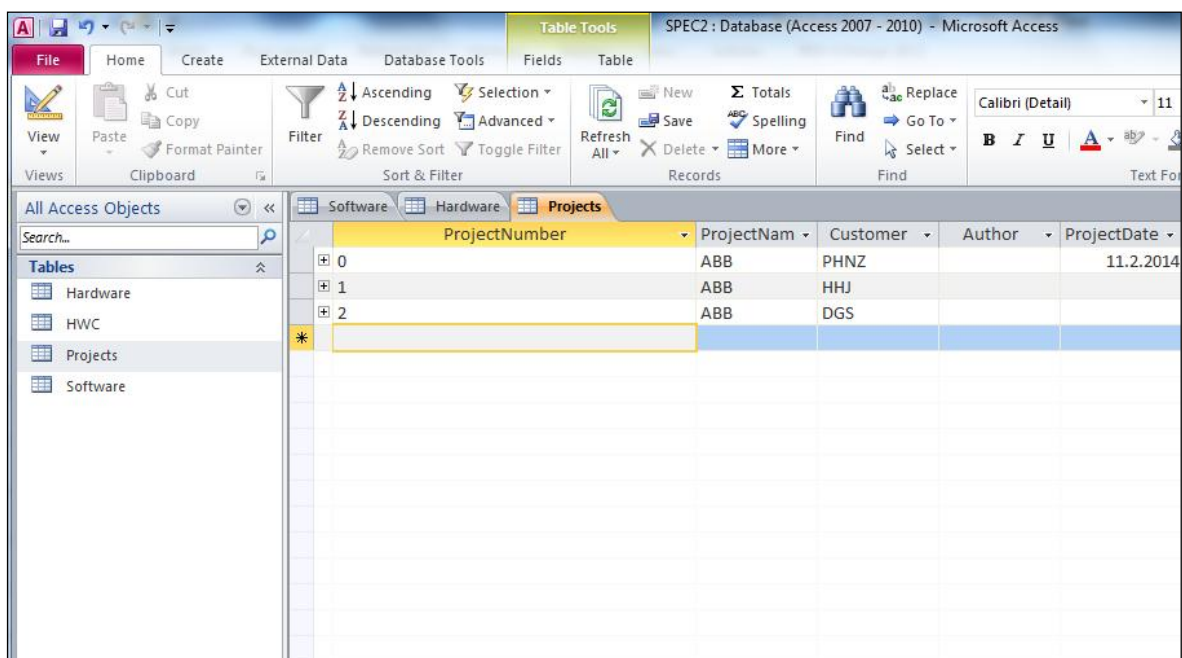


Figure 2. Screenshot of Access

2.3.2 Excel

Microsoft Excel is also a member of Microsoft Office. The program is a so called spreadsheet application that the user can do calculations with. Other features are graphing tools, pivot tables etc. The programming language is the same as in Access, Visual Basic for Applications, but Excel was the first application to include VBA. (Microsoft 2014)

2.3.3 Visual Basic for Applications

Visual Basic for Applications, VBA, is a dialect of the Microsoft programming language Visual Basic. VBA first appeared with Excel version 5 and, starting from Excel 97 VBA was shared by all Microsoft Office applications.

3 Databases

3.1 Storing Information

Today databases are used in the whole world and are important to every business. They are used to store and present data, both to customers and clients. Databases are used by almost everyone even if the user is not aware that he or she is accessing a database. The average user can be a person shopping online, browsing through a company's items or a scientist storing data from an experiment.

To understand databases one has to understand data first. Data is a gathering of information, one piece or more. Common examples of data are names, values and numbers. Sometimes information differs from data, so if one says that it is 23 degrees outside today, then the number 23 is the data and the whole sentence is information. (Stephens & Plew 2003, p 6; Databasteknik 2013)

Databases have been used since the late 1960s and have developed into a powerful tool. For creating and managing large amounts of data a software called *database management system*, DBMS, or shorter, a database system is used. The software is the link between the user and the database and provides the user with persistent storage. The DBMS is the program that stores and handles databases. An example of a DBMS is Microsoft Access that was used in this thesis work. As the DBMS is the link between the user and the database it also makes it possible for the user to modify data, and this is done with a query language. DBMS also supports many users to access the data at the same time, called transaction management. Transactions are executed one-at-a-time, either completely or not

at all to avoid unwanted changes if two users modify the data at the same time. (Garcia-Molina, Ullman & Widom 2002, pp. 1-2; Databasteknik 2013)

3.1.1 Database Systems

Basically a database is just an organized collection of stored data. An example of a simple database is a phone book. It is a collection of names, numbers and addresses that are related to each other in an organized way. The database is managed by a DBMS. One task for the DBMS is to enable the users to create new databases and organize the data as the users sees fit. This can be done with a specialized language called data-definition language that is a part of SQL. (Stephens & Plew 2003, p. 10; Garcia-Molina et al. 2002, pp 4-5)

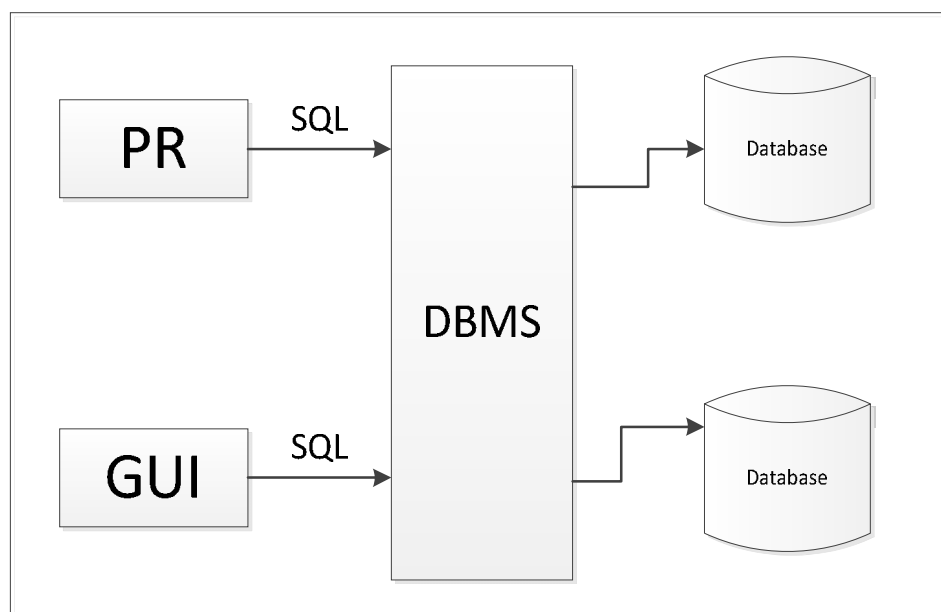


Figure 3. Database system

The DBMS also allows the user to ask questions about the data, which is called to query the data. One could write a query that asks the database how many records are stored in the database and it will answer. A query can also modify the data, if a suitable query language is used. More about queries in chapter 5. (Stephens & Plew 2003, p. 10; Garcia-Molina et al. 2002, pp 4-5)

Since a database often contains very large amounts of data it is important that the DBMS supports the storage in a secure way so no data is lost. No unauthorized users should access the database and no information should be lost while modifying the database. In addition the modifications that one user makes should not affect other users. DBMS should support many users to access the database at once without unwanted consequences. (Stephens & Plew 2003, p. 10; Garcia-Molina et al. 2002, pp 4-5)

Figure 3 demonstrates the DBMS that comprise modules that can both access and control data. The DBMS in the figure also includes a database that contains data.

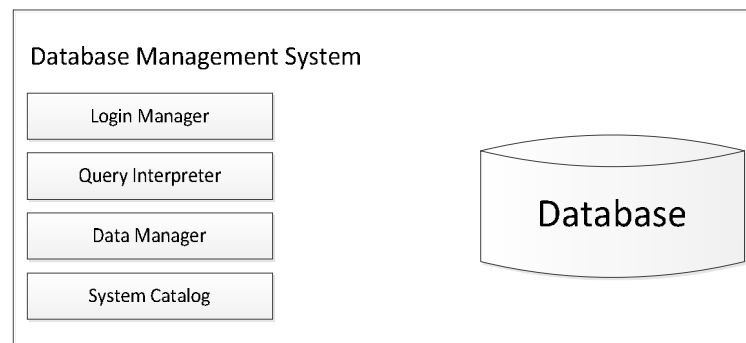


Figure 4 DBMS components (Stephens & Plew 2003, p 10)

Early database management systems evolved from file systems. The file systems are good for storing large amounts of data, just as DBMS, but are much more primitive. A file system is not suitable for many users simultaneously, which is a necessary function today. If two, or more, users modify the same data at the same time there is no way to be certain that both users' changes will be made. Also a file system doesn't support a query language. (Garcia-Molina et al. 2002, pp 4-5)

An example of an early application of a DBMS was corporate records. Information about the sales and the employees were stored. For the employees their names, addresses and salary were important information when a paycheck was to be made. Queries to retrieve this information were needed. Also printing of reports for sales, bills, receipt was done with queries. Changes in the database were made all the time by simple queries. (Garcia-Molina et al. 2002, pp 4-5)

These early systems did not support high-level query languages and the DBMS was very large and expensive. At that time a large computer was needed to store a gigabyte of data compared to today when we can store several terabytes on one disk. When the DBMS became smaller it also got cheaper. And, as a result of that, people started to store more and larger amounts of data. Today we can easily store many terabytes of data and, in more advanced systems, petabytes of data. (Garcia-Molina et al. 2002, pp. 5-6)

3.2 Database Environment

The database environment is the environment that the database exists in. It is a combination of hardware, software and networking. All the query processes and applications used to access and modify the data are also included in the database environment. Some applications don't let the user access the data from within the database environment, the user accesses the data from outside the database environment. The web-based application is an example of an application that lets the user access data from outside the database environment. When deciding what kind of database one should use and how it should be working, the database environment is important to acknowledge. (Stephens & Plew 2003, pp. 15-17)

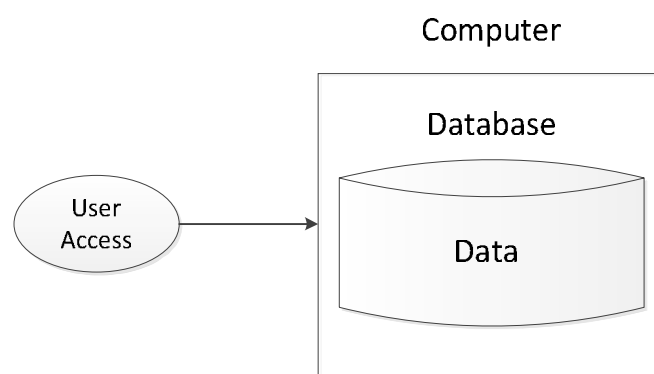


Figure 5. Basic database environment (Stephens & Plew 2003, p. 17)

3.2.1 Client-Server Environment

It has become very common to divide the work of a DBMS into client-server architecture. The database architecture refers to the way the data is organized. It might be important to note that there can be more than one client or server. The client sends a request to the server and then the server executes the request. A database would exist on the server and the user could access the database through the client. The client and the server communicate and pass information over a network. (Garcia-Molina et al. 2002, p. 7; Stephens & Plew 2003, pp. 18-20)

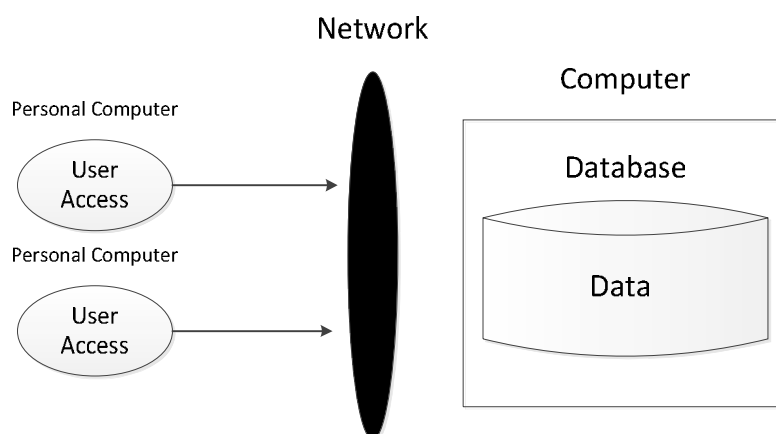


Figure 6. Basic client-server environment (Stephens & Plew 2003, p 19)

Advantages of the client-server architecture are that it is easy to increase the number of users. It is also easy to add both clients and servers to the architecture. If a new server is added to a system the tasks of the servers can be divided so that the more demanding tasks can be performed by the servers with faster processors. The servers with slower processors can perform tasks that demand less processing resources. Figure 6 illustrates a complex client-server environment with several servers and clients. This could be a realistic example of the client-server architecture for a small company. (Garcia-Molina et al. 2002, p. 7; Stephens & Plew 2003, pp. 18-20)

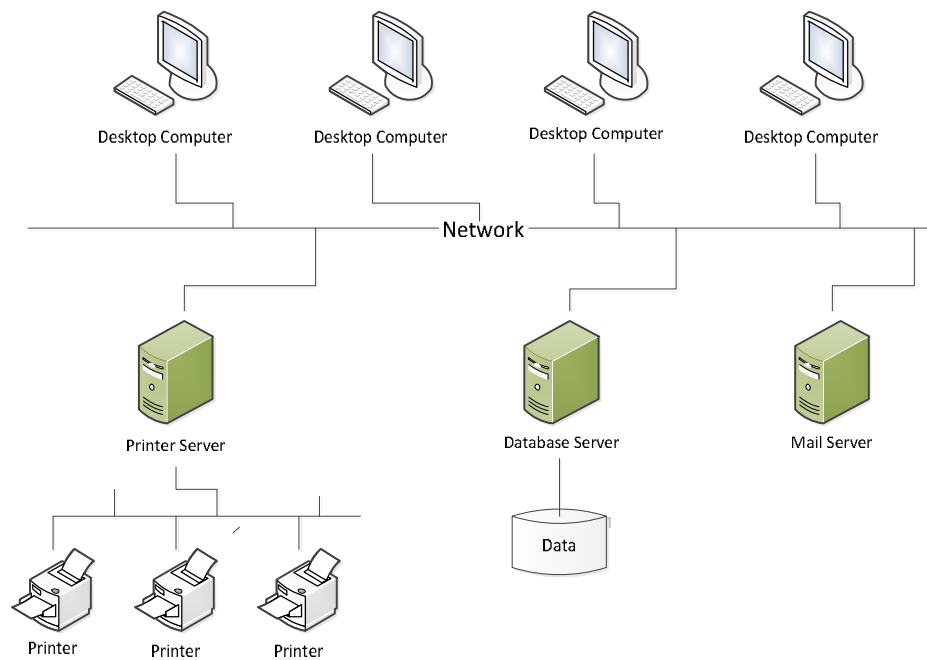


Figure 7. Client-server environment (Stephens & Plew 2003, p. 21)

3.2.2 Multi-Tier Environment

The connection to the server can be done in two ways in a client-server environment. In a two-tier architecture the client is directly connected to the server. This works well for a small system. In a bigger system with several servers and clients, a computer can be placed between the client and the server. Then the architecture is called three-tier architecture. It is possible to have several computers between the server and the client. However, the most common architecture is the three-tier architecture. The idea of having a computer placed between the client and the server is to avoid overpowering the server if the amount of users is very high. (Stephens & Plew 2003, pp. 21-23)

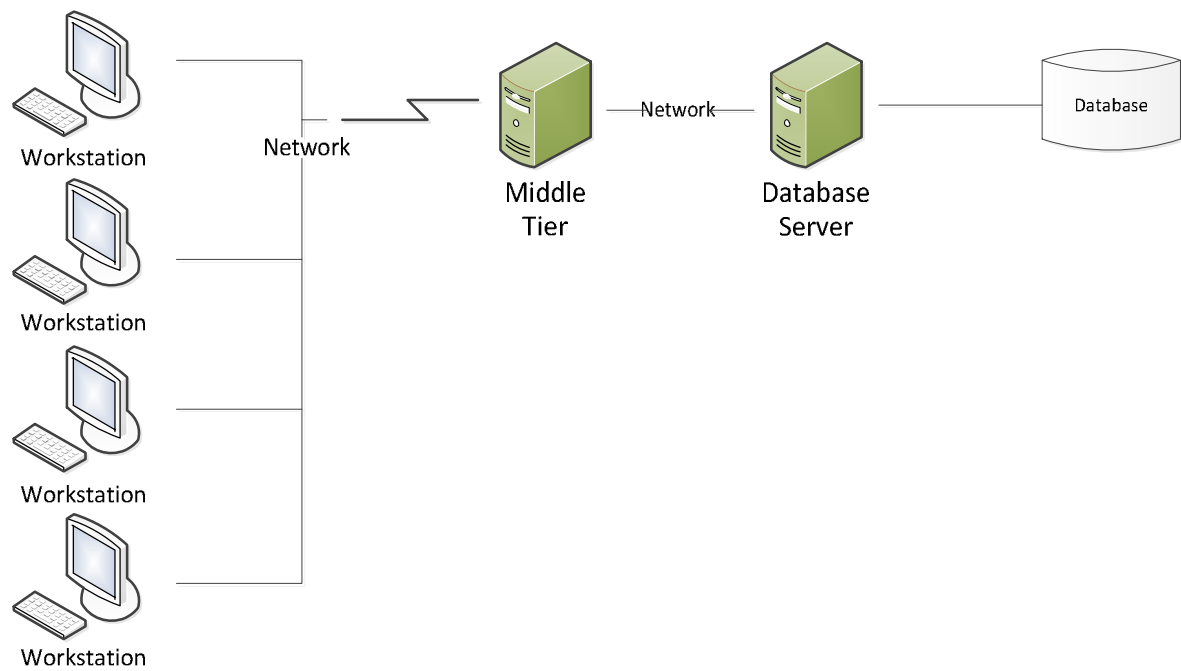


Figure 8. Multiple tier database architecture (Stephens & Plew 2003, p 22)

The computer that works between the client and the server can balance the workload for the server by controlling the number of users that is connecting to the server. It can also control the number of requests that is sent simultaneously to the server. (Stephens & Plew 2003, pp. 21-23)

3.3 Database Architectures

The database architecture is the way that the data is organized to work properly for an organization. The architecture's systems differ from each other and many different systems exist today.

3.3.1 Flat File Database

In the 1950s computer programs stored their data as flat files. These files are organized in the same way as a filing system. It is an assemblage of documents that are organized in a specific way, such as alphabetically. It is troublesome to search in a large flat file database since you have to search in a consequent order to find the row you are looking for. For this reason it is also very inefficient to search a flat file database and large amounts of data should be stored in a DBMS. But for smaller amounts of data that are not expected to grow the flat file database is acceptable. Other problems with storing data in a flat file database is that only one user at a time can access a file. (Stephens & Plew 2003, pp. 66-67)

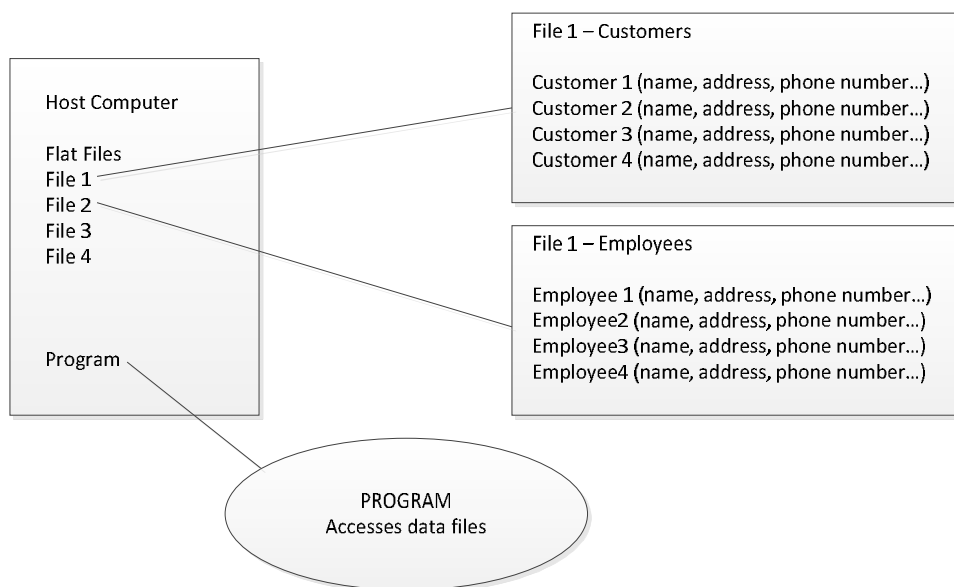


Figure 9. Flat-file database architecture (Stephens & Plew 2003, p. 67)

As an example of the problems with the flat file database we could imagine that a company documents and saves information about the sales every day. The database will grow quickly since if one customer buys more than one item, the information about the customer will be stored separately for every item. Table 1 illustrates the data redundancy problem that the flat file database causes. (Stephens & Plew 2003, pp. 66-67)

Table 1. Flat file with sales data

NAME	NUMBER	ADDRESS	ITEM
Sheldon Cooper	1234	Pasadena	Chair
Sheldon Cooper	1234	Pasadena	Table
Sheldon Cooper	1234	Pasadena	Plant
Penny Howard	2345	Pasadena	Chair
Leonard Raj	4567	Pasadena	Table

To make the database more efficient the information about customer 1234 could be stored only once and the information about his purchases stored in a separate relation. (Stephens & Plew 2003, pp. 66-67)

3.3.2 Hierarchical Database

In the late 1960s the file storage systems evolved into the hierarchical database. The name comes from the way the database stores data. The architecture looks like a family tree with the first record called the root. The root has one or more child records and the child records have child records of their own. It differs from a human family tree in a key aspect, since all records have only one parent. (Stephens & Plew 2003, pp. 68-70)

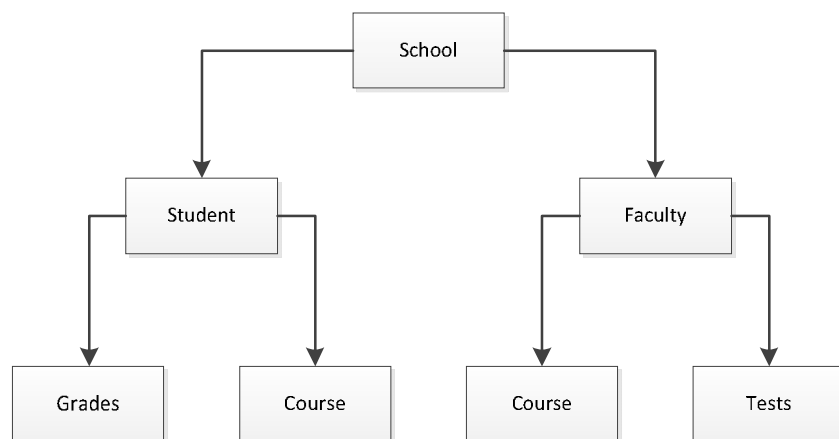


Figure 10. Hierarchical database structure (Stephens & Plew 2003, p 69)

The hierarchical database was the first to have relationships between the records. This solved the redundancy problem from the flat file database. The customers would be listed

in a separate record and therefore they only appeared once. Figure 10 illustrates the advantage of the relationship system. (Stephens & Plew 2003, pp. 68-70)

Table 2. Decreasing redundancy with relationships

NAME	NUMBER	ADDRESS
Sheldon Cooper	1234	Pasadena
Penny Howard	2345	Pasadena
Leonard Raj	4567	Pasadena

ITEM
Chair
Table
Plant

One of the problems with the hierarchical database is that there can only be one tree per database. As seen in Figure 10. Hierarchical database structure (Stephens & Plew 2003, p 69) the record named “Course” needs to appear twice since there can only be one parent per record. The two records need to be identical for the system to work properly so, if one of the records is modified, the other record needs to be modified in the same way. Data redundancy is not completely gone in the hierarchical database system. (Stephens & Plew 2003, pp. 68-70)

To access any record, the user has to start at the root and then go through every parent record to access the right record. The user also has to know how the records were structured to find the right record. (Stephens & Plew 2003, pp. 68-70)

3.3.3 Network Database

Most of the problems with the hierarchical database were solved with the network database system. The network database can have several trees and it also made many-to-many relationships possible as seen in Figure 11 Network database model (Stephens & Plew 2003, p. 71) which illustrates the same system as in Figure 10. Hierarchical database structure (Stephens & Plew 2003, p 69). Now the “Course” record only appears once. (Stephens & Plew 2003, pp. 70-71)

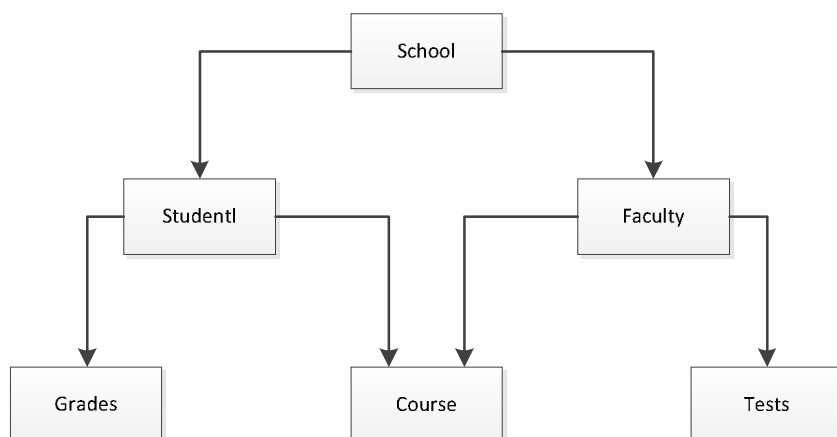


Figure 11 Network database model (Stephens & Plew 2003, p. 71)

The user doesn't have to go via the root to access one of the child records, which was also an improvement. However the user still needed to know how the records were structured to find the right record. The problem with the network database was maintenance and implementation. Also, a person who wasn't familiar with the database structure needed to be able to use the database and therefore a new architecture was needed. (Stephens & Plew 2003, pp. 70-71)

4 Relational Database System

The relational database is the most common database model today. Database systems changed following a paper written by mathematician Edward Frank Codd from the IBM Research Laboratory in 1970. Dr. Codd suggested that the data would be stored as relations to encourage data independency. This made the physical location of the rows unimportant. This was more user-friendly since the earlier database models required the user to be familiar with the database structure to be able to search the database. The relational database system also eliminated the data redundancy and data could then be modified without making changes in the application program. (Garcia-Molina et al. 2002, p. 4)

As can be seen in Figure 12 all relations are stored only once since they have relationships that connect them. There is no hierarchy between the relations and a new user doesn't have to be familiar with the database to find the right relation.

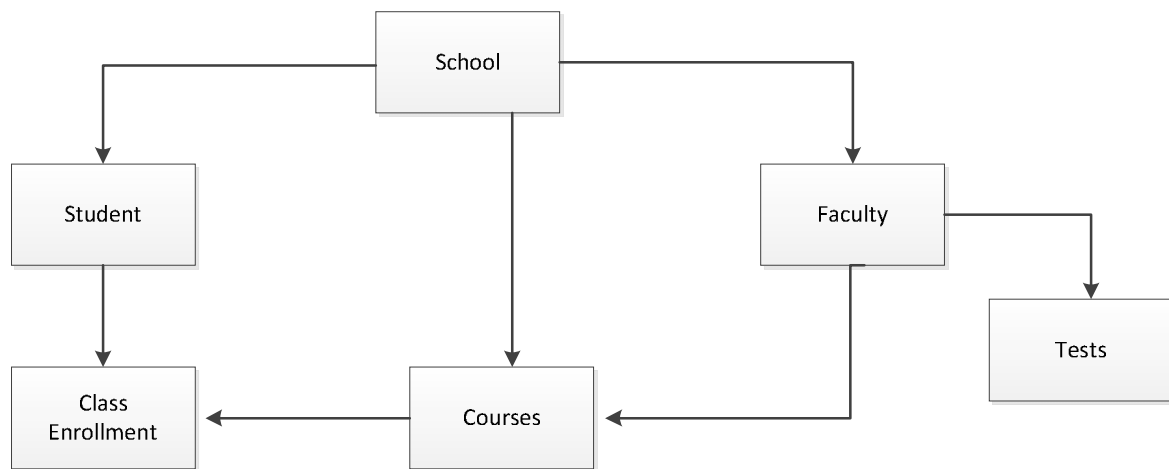


Figure 12. Relational database model (Stephens & Plew 2003, p. 72)

4.1 Elements of the E/R Model

When designing a database it is essential to first consider what the database is going to be used for. Then the developer can decide what information the database should contain and what relationships should be established. When designing a schema for a relational database, it is common to start with an entity-relationship model and then convert it to the relational model. (Garcia-Molina et al. 2002, pp. 23-25)



Figure 13. Database modeling (Garcia-Molina et al. 2002 p 24)

Figure 13. Database modeling (Garcia-Molina et al. 2002 p 24) shows how the E/R model could be used when designing a database. The model is later modified to a concrete relational design, which is called relational database schema. (Garcia-Molina et al. 2002, pp. 23-25)

The objects which are used in the E/R model consist of three main elements. The first one is the entity set, which is a collection of several entities that are similar to each other. An entity is an abstract object and in Table 3, every row is an entity and the set of all rows forms an entity set. The second element is attributes that describe the entities in the set. In

Table 3 the given attributes are Name, Type, Breed and Color. The last main element is the relationships, which is the connections between the entity sets. If we have two entity sets like Animal and Owner, we could have the relationship “owner of” that connects owner with the right animal. The idea is that an Animal entity is related to an Owner entity. (Garcia-Molina et al. 2002, pp. 23-25)

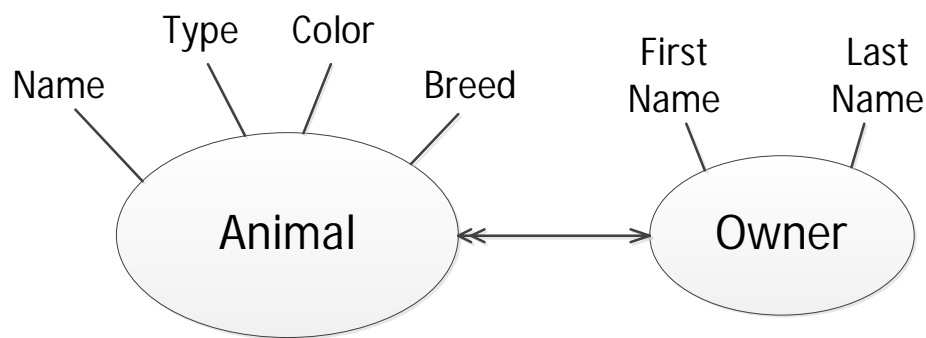


Figure 14. E/R design

4.2 Basic Elements of a Relational Database

In the relational model data can be represented as a two-dimensional table and is called relation. The relational model builds on the E/R model and has therefore the same basis. The relation Animals is the same as the entity set Animals. The attributes are the names of the columns and usually describe the entries in the columns under them. It is a common step to turn an entity set into a relation and have the same sets of attributes. (Garcia-Molina et al. 2002, pp. 61-63)

Table 3. The relation Animals

Name	Type	Breed	Color
Fluffy	Dog	Poodle	Black
Stella	Dog	Pug	Beige
Minnie	Cat	Ragdoll	White

The name of the relation and the set of attributes is the schema for the relation. The schema for the relation in Table 3 is Animals (Name, Type, Breed, Color). The order of the attributes is also the standard order for the relation Animals.

Tuples are the names of the rows in a relation. One tuple has one constituent of all the attributes of the relation. In Table 3 the first tuple looks like Fluffy, Dog, Poodle and Black for attributes Name, Type, Breed and Color. It is important to use the same order as in the attributes in the relation schema. (Garcia-Molina et al. 2002, pp. 61-63)

To be able to search for one or several tuples in the database, a unique value for every row is needed. This is done by having attributes where the entered value is different than all other values in the table. This value will be the primary key and, by using a primary key, the user can search for a specific tuple. (Garcia-Molina et al. 2002, pp. 61-63)

4.2.1 Primary Key

The primary key often consists of numbers since numbers go to infinity. Everyday examples of so-called primary keys are license plates and phone numbers. One primary key that is used in many systems is the social security number since it is a unique combination of numbers and letters. Since the social security number is an ideal primary key one person can also be identified in many different databases. This means that a person's medical records, police records, bank records and so on can easily be combined. It is noteworthy that by the Finnish law only some have the rights to store this kind of data. (Stephen and Plew 2003, pp. 89-94)

In a relation, the primary key can never be a null value. In a database where no unique values are to be stored an ID column can be introduced so that a tuple can be uniquely identified. The key might be referred to in other relations and therefore it is also important that the key is never changed or duplicated. Customer numbers are perfect examples of primary keys since several persons can have the same name, and addresses and phone numbers can change. By creating an extra column for each customer and giving it a unique number a primary key is created. See Table 4. (Stephen and Plew 2003, pp. 89-94)

Table 4. Relation with primary key

ID <i>Primary Key</i>	NAME	CITY	COMPANY
123	Sheldon	Texas	ABB
124	Sheldon	Pasadena	UOP
125	Howard	Vasa	ABB

4.2.2 Index

To make accessing the database more efficient a database object called index is used. The index is a pointer to specific table data. By giving the primary key column an index, the computer can search the database.

The index system works like the index list that can be found at the back of a book. To help the reader find certain information fast, all the keywords and page numbers are listed alphabetically. The reader can then sort through the list to find the right keyword, turn to the right page and then retrieve the information. If there wasn't an index, the reader would have to read through the whole book to find the information. In a database the index works the same way and the servers are the books. (Stephen and Plew 2003, pp. 94-96)

4.3 Relationships

When establishing a relationship between two tables it might be good to know that there are different kinds of relationships. To establish a relationship, an attribute called foreign key is used. The foreign key is a normal attribute in one table that is a primary key in another table. An example of this could be a table with schools and one table with students. All students could go to one school but the school could have several students. (Stephen and Plew 2003, pp. 109-110)

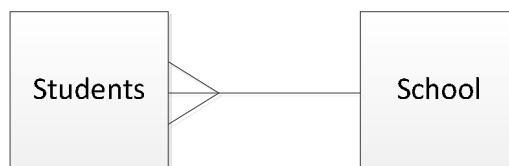


Figure 15. One-to-many relationship diagram

The primary key in the school table could be the name of the school since it should be unique. In the students table an attribute named school should be included since it can serve as the foreign key between the tables. With this information a relationship can be established since the same information appears in both tables. (Stephen and Plew 2003, pp. 109-110)

If the students could only have one school and the schools could only have one student the relationship would be called a one-to-one relationship. For example in a database with one relation for persons and one for DNA the relationship between the two relations would be one-to-one since one person can only have one DNA and the DNA can only belong to one person. (Stephen and Plew 2003, pp. 109-110)



Figure 16. One-to-one relationship

The third kind of relationship is the many-to-many relationship. In the same database as in the example about one-to-many relationships, many-to-many relationships could exist. One could be the relationship between the students and the teachers. One student could have several teachers and one teacher could have several students. (Stephen and Plew 2003, pp. 110-111)

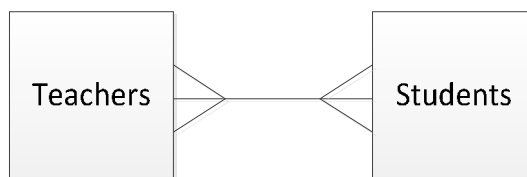


Figure 17. Many-to-many relationship diagram

Many-to-many relationships use an associative table between the two main tables. By doing this, two one-to-many relationships are created instead. The associative table will include foreign keys for both tables and the combined set of them will work as the primary key for the associative table. (Stephen and Plew 2003, pp. 110-111)

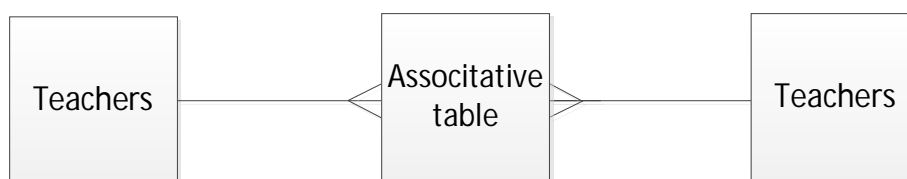


Figure 18. Associative table to resolve many-to-many relationship

5 The Database Language SQL

To form a relationship, communication must occur. To communicate with a relational database you can use a language called Structured Query Language (SQL). SQL standards are provided by the ANSI and ISO standards and the commands are very clear. If you want to select something from the database, the SQL command is simply SELECT. It is noteworthy that although SQL commands are often written in capitals, SQL is case insensitive. Both SELECT and select are equally right. SQL commands can be grouped into four main categories. (Stephen and Plew 2003, pp. 98-99,247-249)

Data Definition Language (DDL) is the part of SQL that is used to create, alter or drop something in the database. DDL communicates with the database design and the DBMS.

CREATE is the most common word to start a DDL command. Other popular commands are ALTER and DROP. ALTER modifies the structure of database objects and DROP deletes database objects. (Stephen and Plew 2003, pp. 98-99,247-249)

For example:

```
CREATE TABLE Animals (Animal_ID NUMBER PRIMARY KEY, ...)
```

Data Manipulation Language, DML, is the part of SQL that manipulates the data in the database table. Common DML commands include the words UPDATE, INSERT, DELETE. (Stephen and Plew 2003, pp. 98-99,247-249)

For example:

```
INSERT INTO Animals (Animal_ID, Name, Type, Breed, Color) VALUES
(1, 'Fluffy', 'Dog', 'Poodle', 'Black')
```

Database Query Language, DQL, is the part of SQL that asks questions of the database and receives an answer. If the user wants to know how many poodles the database contains, a command that selects all the poodles can be formed. (Stephen and Plew 2003, pp. 98-99,247-249)

For example:

```
SELECT * FROM Animals WHERE Breed='Poodle'
```

The SQL is now going to return all the tuples of the Animals table where the breed is poodle. (Stephen and Plew 2003, pp. 98-99,247-249)

For controlling access to the database the Data Control Language, DCL, is used. Some users can be privileged with more rights than others and some users might have no access

at all. Some control commands are `ALTER` to grant access and `REVOKE` to remove granted access. (Stephen and Plew 2003, p. 250)

5.1 Simple Queries

SQL queries are well structured and easy to create and an example of how to use the select command can be seen below. The `SELECT` query is, as already mentioned, a way to ask for data. By also using `FROM` and `WHERE` the answer is the tuples that satisfy the given conditions. The command looks like:

```
SELECT column_name FROM table WHERE condition
```

After the `SELECT` no specific column name has to be defined. The column name can be replaced by a `*` for retrieving all columns in a table. It is also possible to write many column names after `SELECT`. The condition is the requirement that the column needs to fulfill, such as a value that the column contains. To avoid duplicates `SELECT DISTINCT` can be used. (Garcia-Molina et al. 2002, pp. 239-243)

5.2 Wildcards in SQL

By putting the `%` sign both in front of and after the letter `s`, the query asks for all the entities that contains `s`. If an entity can be spelled in several different ways, like a name, the `_` sign can be used instead of a letter. So by writing `'_elma'` the query would return both Selma and Celma since both names fulfill the requirements. (Refsnes Data 2014)

If we want to search for several entities but they can start with different letters, the query can be written like:

```
SELECT customer
FROM customer_table
```

```
WHERE customer_name LIKE '[bsp]%'
```

The customer names can start with b, s and p in the example above. To search for names in alphabetical order it is not necessary to write all the letters separately in the query. Instead it can be written like '[a-c]!'. Now all names that start with a, b and c fulfill the requirements. Also if the search should not include names that start with a, b and c the ! can be added '[!a-c]!'. Now the query will return all names that don't start with a, b and c. (Refsnes Data 2014)

5.3 Joins in SQL

In a relational database, relationships between the tables have been established. This is done by having the same information in a foreign key as in a primary key. SQL can use this information to understand the relationship between the tables. The clause `INNER JOIN` is used to combine two or more tables by using the common field between them. (Refsnes Data 2014)

Customers

Customer_Number	Customer_Name	City
1	Selma	Vasa
2	Elsa	Helsinki
3	Maja	Stockholm

Orders

Customer_Number	Order_number	Date
1	1111	24.09.2013
2	2222	30.01.2013
3	3333	04.09.2013

To combine the two tables Customer and Orders, a query that joins the two tables can be written. The combining column is the customer number since it contains the same data

```
SELECT  Customers.Customer_Number, Customers.Customer_Name,
Orders.Order_Number
FROM Customers
```

INNER JOIN Orders

ON Customers.Customer_Number= Orders.Customer_Number

The outcome of the query above would be a combination of the two tables and look like the table below.

Customer_Number	Customer_Name	Order_Number
1	Selma	1111
2	Elsa	2222
3	Maja	3333

To retrieve all tuples when there is a match in one of the tables FULL JOIN can be used. If we want to retrieve all the tuples from the left table, and the matched tuples from the right table LEFT JOIN can be used. Vice versa RIGHT JOIN can be used. (Lahtonen 2002, pp. 100-103; Refsnes Data 6.2.2014)

6 Data Storage

When creating a database it is important to consider how the data will be stored. If the database becomes very large, managing all the data also becomes more difficult. If the database is relatively small, a single disk can be enough for storing the database. If it is a larger database, for example a bank, a tertiary storage might be needed. (Garcia-Molina et al. 2002, p. 507)

6.1 The Memory Hierarchy

A computer system often has several different components for storing data. A schematic of the memory hierarchy can be found in Figure 19.

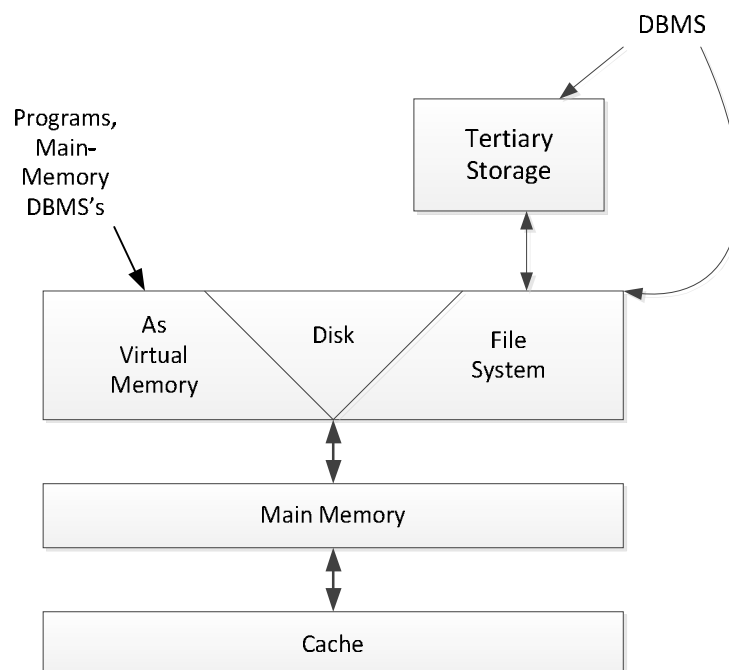


Figure 19 The Memory Hierarchy (Garcia-Molina et al. 2002, p. 507)

6.1.1 The Cache

On the lowest level, the cache is found. The cache is a place for temporary storage. The system places a copy of often used information to have fast access to it. It is faster to retrieve information from the cache, than from the place where the information is originally stored, like the main memory. The only difference to the user is that it goes faster to retrieve the information with a cache than without.

The size of the cache is limited because it is built with more efficient and more expensive hardware. And it is faster to search a small amount of data than a large amount. If a machine executes an operation, it will first look in the cache for the necessary information. If it can't find it in the cache, it moves to the main memory. After the machine finds the information it copies it to the cache. Since the size of the cache is limited, the system often needs to remove something, before it can place the new copies in the cache. If the data that is removed from the cache has been modified, the system first needs to copy the new values to the main memory. (Datatermgruppen 2013)

Reading data from a cache only takes a few nanoseconds, which is very efficient for the user. Moving data between the main memory and the cache takes about 100 nanoseconds. (Garcia-Molina et al. 2002, p. 508)

6.1.2 Main Memory

The next level, after the cache, is the main memory. The main memory is the center of everything that happens in a computer. From data modification to various executions, all the data is resident in the main memory. Random access is implicated in the main memories, which means that any byte is moved in the same amount of time. Accessing data from the main memory is slow, compared to the cache, and takes from 10 to 100 nanoseconds. (Garcia-Molina et al. 2002, p. 508)

6.1.3 Virtual Memory and Secondary Storage

The virtual memory is an advanced form of memory that is built in the operating system. By using an external memory, an internal memory is simulated that is bigger than the physical internal memory. The files of the program occupy a virtual memory address space and most of the content of a fully occupied virtual memory is stored on the disk. (Datatermgruppen 2013)

Computers today use some sort of disk for secondary storage. The secondary storage is much slower and also more capacious than the main memory. As seen in Figure 19 the disk is supposed to support both the file system and the secondary storage. The files that are stored on the secondary storage are moved between the disk and the main memory, controlled by the operating system. Moving from the disk to the main memory is called disk read, while moving files from the main memory to the disk is called disk write. (Datatermgruppen 2013; Garcia-Molina et al. 2002, pp. 510-511)

6.1.4 Tertiary Storage

Databases can be very large, several terabytes. To serve that need, tertiary storage has been developed. The tertiary storage offers much larger capacities and smaller costs than secondary storage. Also, tertiary storage has higher read/write times. There are several kinds of tertiary storage devices. (Garcia-Molina et al. 2002, p. 512)

Ad-hoc Tape Storage is the simplest form of tertiary storage. The data is put on tape reels and stored in racks. When data is wanted, a human operator locates the tape and mounts it to a reader. Then the information is copied to the secondary storage or to the main memory. Writing to the tape works in the same way: a suitable tape is localized and then the data is copied from the disk to the tape. Several terabytes, up to several petabytes can be stored in a tape library. (Garcia-Molina et al. 2002, p. 512) (HP 9.2.2014)

Another principle is the Optical-Disk Juke Box. The so-called juke box can load and unload optical discs and up to several petabytes can be stored. A robotic arm extracts a disk and moves it to the reader. Then the data in the disk can be read to the secondary storage. (Garcia-Molina et al. 2002, p. 512)

6.1.5 Modifying Data

No computation takes place on the disk. The blocks are moved from the disk to the main memory or the cache. The disk only needs to move the data main memory. Writing data to the disk works in the same way as reading data from the disk. The right sector needs to be positioned under the disk head and the head writes the data. (Garcia-Molina et al. 2002, p. 523)

To modify data on the disk it first needs to be modified in the main memory. The first step is to read the original data from the disk to the main memory. Then the changes can be made in the main memory copy of the data. Finally, the modified data need to be written to the disk. (Garcia-Molina et al. 2002, p. 523)

7 Course of Action

7.1 Planning the System

When planning the system it was important to consider everything that the database needed to contain to make the design right from the beginning. By discussing the matter with project engineers, looking at old hardware and software documentation and taking the old model into consideration, the planning of the database could begin.

The next question to be answered was how the data should be written to the database. The project engineers need a simple way to modify the data in the database. This particular part didn't take long to figure out since, as already mentioned, other systems similar to this were already in use. Also, the program needed to be accessed by multiple users and needed to be installed on the users' computers.

The database needed to be stored somewhere and after discussing the matter with my supervisor, it was decided that the database should be stored on the same network drive as the project folders since all the project engineers already had access to the hard drive. It is also logical since it was familiar to the user to access the hard drive for information about the projects. The network drive itself is located in a central ABB server and the system was created as a client-server environment.

The relations in the database needed to be linked to one another to avoid redundancy in data. By not storing the same data several times, the database is smaller and the communication to the database is faster. All the data that was to be stored contained some form of unique information, which made it suitable for primary key and for dividing the data into groups.

7.2 Database Design

When examining the old hardware and software documentation files I recognized a pattern. In every project the data that was stored could be divided into three groups, hardware, hardware components and software. Also information about the project itself, like project number and customer was stored. By using these four groups as guidelines I created four relations to store data.

7.2.1 Relations and Relationships

The database was designed as a relational database and Figure 20 illustrates the relations and their relationships. The main relation is the Projects relation and in this table all basic information about the project is stored. The projects have a unique project number and this number is a suitable primary key. All projects consist of different hardware like computers, servers etc. The hardware is only used once and the serial number is unique. The hardware can have different components and software can also be installed on the hardware.

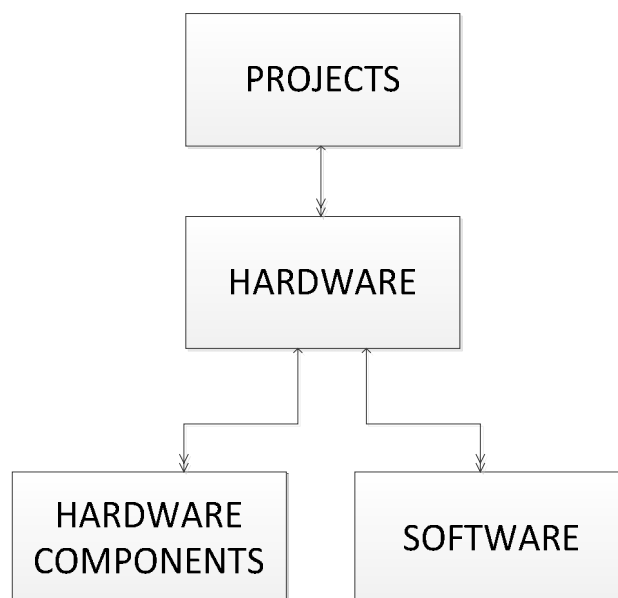


Figure 20. Database design

All the relationships are one-to-many relationships and since every component is only used once and only used in one project, no other kind of relationships was needed. As mentioned earlier in the chapter about one-to-many relationships, some form of data

redundancy is needed to establish the relationship. An example of this in this system is the project number that is stored both in the Project table and in the Hardware table. In the Hardware table, all hardware for all projects will be stored and, to specify which hardware belongs to which project, the project number functions as both primary and foreign key. The system is illustrated in Figure 21. The primary keys are unique in their respective table but the project number can occur several times as the foreign key in the second table.

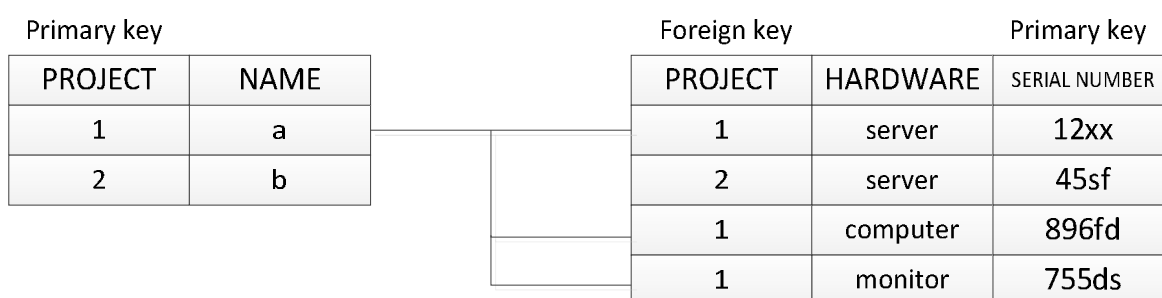


Figure 21. Relationships

7.2.2 Table Design

The different relations consist of rows and columns and the design looks like tables. A screenshot from the empty database can be seen in Figure 22. The figure is from the relation Projects. The data for a project will be listed in a row and organized according to the columns.

ProjectNumber	ProjectNam	Customer	Author

Figure 22. Screenshot of empty database

All tables, except the Projects table, have both a primary and a foreign key. This is illustrated in Appendix 1, HW & SW database SPEC.

7.3 User Interface Design

The main part of this thesis was creating the user interface. New data needed to be written to the database and old data modified. In addition, a tool for writing reports was needed.

As mentioned, it was decided that the user interface would function as a wizard. A wizard is a setup assistant that guides the user through defined dialogs. Not many options should be given to the user. The basic idea of this can be seen in Appendix 2, Database user interface. The wizard is built up as a system where one user form leads to the next user form based on the options chosen in the previous user form. When using the system, the user can choose between a couple of buttons depending on what he/she would like to do. If either “Create new” or “Edit existing” is chosen, a user form will open. By clicking “Create New” the user form seen in Figure 23 will open.

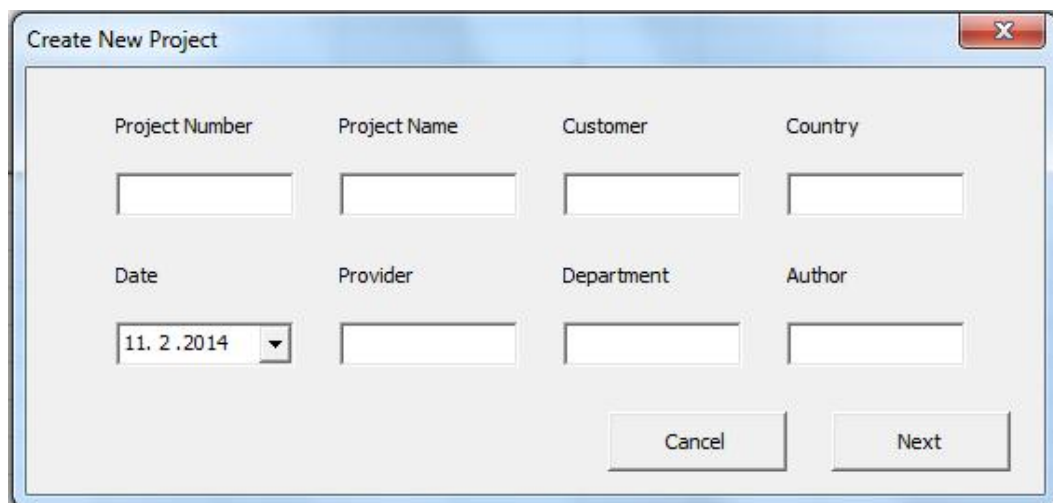


Figure 23. User form Create New Project

After all the required information has been filled in, the user proceeds by clicking next and the following user form will open. In which order the user forms appear can be seen in Appendix 3, User forms in Excel. To help the user understand how the system works, instructions were also written.

7.3.1 Communication between Database and User Interface

The communication had to work so not only new data could be written to the database, but also so that old data could be modified and fetched from the database. The code for this was basically the same for all operations, only the SQL statement was different. Figure 24 is a screenshot from the code that writes data to the database. The SQL statement writes the text from “Textbox1” and the value from “DTPicker1” to the columns ProjectNumber and ProjectDate in the table Projects. The code for retrieving data is a bit more complicated but still quite similar. As mentioned in chapter 5.1 the INSERT would be replaced by SELECT.

```
Private Sub CommandButton1_Click()

    Dim connector As ADODB.Connection
    Dim strConnection As String
    Dim strStatement As String
    Dim emptyRow As Long

    'SQL statement that inserts data for Project Number as a text and the date as a date value
    strStatement = "INSERT INTO Projects (ProjectNumber, ProjectDate) VALUES ('" & TextBox1.Text & "', '" & DTPicker1.Value & "'"

    Set connector = New ADODB.Connection
    strConnection = "Provider=Microsoft.ACE.OLEDB.12.0;"
    strConnection = strConnection & "Data Source='C:\db\SPEC2.accdb';" 'path to database

    connector.Open strConnection 'open the connection
    connector.Execute strStatement

    connector.Close 'close the connection
    Set connector = Nothing

    new_hwndsw.Show 'show next user form
End Sub
```

Figure 24. Insert code

7.3.2 Creating Reports and Searching the Database

The report is created by retrieving the information from the database and then printing it to a sheet in the excel file. In the beginning of the project, as can be seen in Appendix 2, the idea was to preview the report while creating a new project. This was not very functional if modifications to a project were made later on and a new report was needed. The system now works so that all the data is first written to the database by creating a new project. After filling in all information the user clicks the “Create Report” button and a user form will show. The user writes the project number in the textbox and the report will be printed.

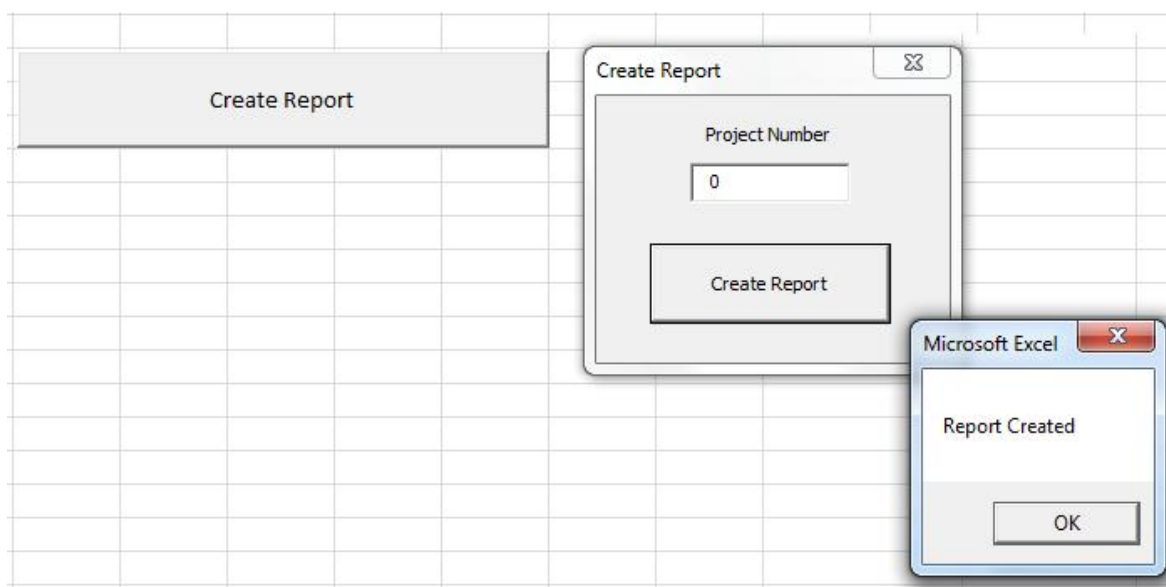


Figure 25. Create report

The layout of the report was the most challenging part of this function. The report will have all the project information in the report header and then list all hardware. Under every piece of hardware, the software and the hardware components belonging to this particular hardware should be listed.

The solution to get the report to list all the right components under the right hardware was to store the hardware primary as a variable. Then the variable can be used to retrieve the right hardware components and software since the hardware primary key was stored as a foreign key in both the software and hardware components relations. The system will first loop through all hardware and retrieve all rows with the right project number. Then the system will loop through the other tables and retrieve all rows with the right foreign key. Also, the code needed to print the right heading and print the information belonging to that

heading below. In Figure 26 the code for printing the headings with the right information is displayed. The first column only prints the heading with a bold font and the second column prints the information stored in the record set fields “rst2”.

```

row = row + 1
Report.Range("A" & row) = "Type"
Report.Range("A" & row).Font.Bold = True
Report.Range("B" & row) = "Serial Number"
Report.Range("B" & row).Font.Bold = True
Report.Range("C" & row) = "Version"
Report.Range("C" & row).Font.Bold = True
Report.Range("D" & row) = "Number of discs"
Report.Range("D" & row).Font.Bold = True
Report.Range("E" & row) = "Information"
Report.Range("E" & row).Font.Bold = True
row = row + 1

Do

Report.Range("A" & row) = rst2.Fields("Type")
Report.Range("B" & row) = rst2.Fields("SerialNumber")
Report.Range("C" & row) = rst2.Fields("Version")
Report.Range("D" & row) = rst2.Fields("NumberOfDiscs")
Report.Range("E" & row) = rst2.Fields("Information")
rst2.MoveNext
    row = row + 1
Loop Until rst2.EOF
rst2.Close

```

Figure 26. Headings

To search the database is similar to printing the report. The result of the search will also be printed in a separate worksheet and the code is very close to the report code. The SQL differs, though, since the user form for the search function makes the user choose from which table he/she wants to search. The tables are not obvious to the user since the user form presents the options seen in Figure 27.

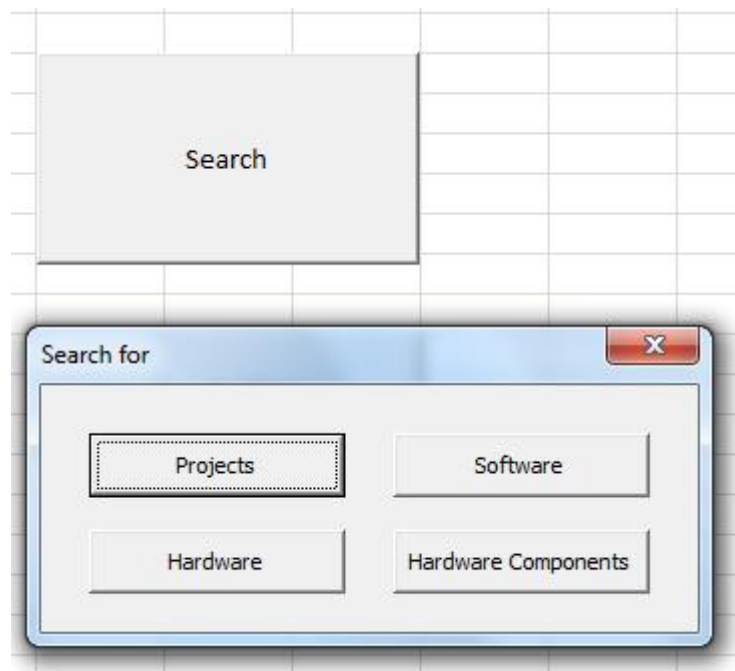


Figure 27. Search options

When deciding what the user would like to search for, several or only one requirement can be made. The search form for software can be seen in Figure 28. The drop down list under “Type” retrieves its information straight from the database. This means that only software types that are already saved in the database can be searched for. When writing data to the database, a number of types are predefined to prevent the users from spelling the same thing in many different ways. It is noteworthy that Operating System and operating system are different words to the database.

A 'Search by' dialog box with a title bar and a close button (X). It contains four input fields: 'Serial Number for Hardware', 'Serial Number', and 'Type'. The 'Type' field is a dropdown menu with 'operating system' selected and highlighted with a dotted border. A 'Search' button is at the bottom.

Figure 28. Search form

8 Final Result

The result was a database that stores the information in four different relations and can be seen in Appendix 1. The database is hidden to the user and the only thing the user can see is the user forms that appear in the Excel form. The user can choose if he, or she, wants to create a report, search the database, modify data in the database or add data to the database.

The system works so that, when collecting data from new projects, the user directly fills in the project information in the database. First the user has to create a new project and then the user can choose if he wants to add hardware, software or hardware components. The user will also be asked, when filling in software or hardware components, to fill in the serial number for the hardware that the component is included in to connect them in the database. After filling in all the components of the project the user can choose to click the “Create Report” button to print a report of the project. There all the information about the project is included. An upgrade from the system that was in use before the database is that the report doesn’t need to be stored as a text document. Anyone with access to the Excel file and that knows the project number can print a project report. This can of course be done several years after the project was completed.

Another requirement set on the thesis was that a tool for searching for specific components needed to be made. This can also be done by simply clicking the “Search” button. The user can now choose if he wants to search for projects, hardware, software or hardware components.

9 Discussion

The database will ease the storing of project data in the future. The functionality and flexibility of the relational database can be read in chapter 4. The database was an improvement for the department. The design is simple and the data is logically ordered in the database. When I understood the basic elements of the relational database, chapter 4.2, it was easy to structure the column and tuples. I am satisfied with the database and I think it works well. It is well planned and therefore unnecessary columns were avoided and it doesn't grow quickly since only important information is to be stored.

The communication between the database and the user interface was in the beginning a bit challenging. After understanding SQL, chapter 5, writing the code became easier. All communication is based on simple SQL queries, and this makes it easy for others to understand the code. The user interface became a problem since the more I learned about programming, the more flaws I saw in my code. This made it hard for me to know when the system was finished. The result is well functioning and corresponds with the scope, but I have several ideas of how to develop the system.

One function that could be improved is the search function. It could be more flexible and also provide more options for the user. Also, the layout would be nicer if the search button opened only one user form and, from there, the user could search the whole database or limit the search, for example only search for software.

This has been a very challenging and demanding task. Especially since I am studying electrical engineering and had no previous experience of databases and very little of programming. It was also very difficult to estimate how much time the system would demand and it took much longer than I estimated in the beginning. That being said I have learned a lot about databases and programming. Most importantly I have learned even more about being an engineer and being able to solve the problem presented to me. When entering working life I know more about solving problems than I have no previous experience of.

I think this thesis work is valuable for me when applying for jobs. I have the education of an electrical engineer but I have also managed a task that is suitable for IT engineers. Often electrical engineers don't know how to or want to do any kind of programming and IT engineers don't know electricity the same way as an electrical engineer. This makes my knowledge valuable since I have a wider perspective.

10 Sources

Arvidsson Stefan (1999) *Access 2000*

Sundbyberg: Pagina

Databasteknik (2013)

<http://www.databasteknik.se/> (read 30.4.2014)

Garcia-Molina, H., Ullman, J. & Widom J. (2002). *Database Systems: The complete book.*

New Jersey: Prentice Hall

Hansson Roger (2000) *Cache – teknisk förklaring*

<http://www.datatermgruppen.se> (read 8.2.2014)

HP (2013)

<http://www.hp.com/us/en/products/tape-automation/product-detail.html?oid=3936307#!tab=models> (read 9.2.2014)

Lahtonen Tommi (2002) *SQL*

Jyväskylä: Docendo Finland OY

Microsoft (2014) *Excel*

<http://office.microsoft.com/en-us/excel/> (read 7.2.2014)

Refsnes Data (1999-2014) *SQL Tutorial*

<http://www.w3schools.com/sql/default.asp> (read 6.2.2014)

Stephens Ryan & Plew Ron (2003). *Sams Teach Yourself Beginning Databases in 24 hours.* Indiana: Perpetual Technologies, Inc

The ABB Group (2014)

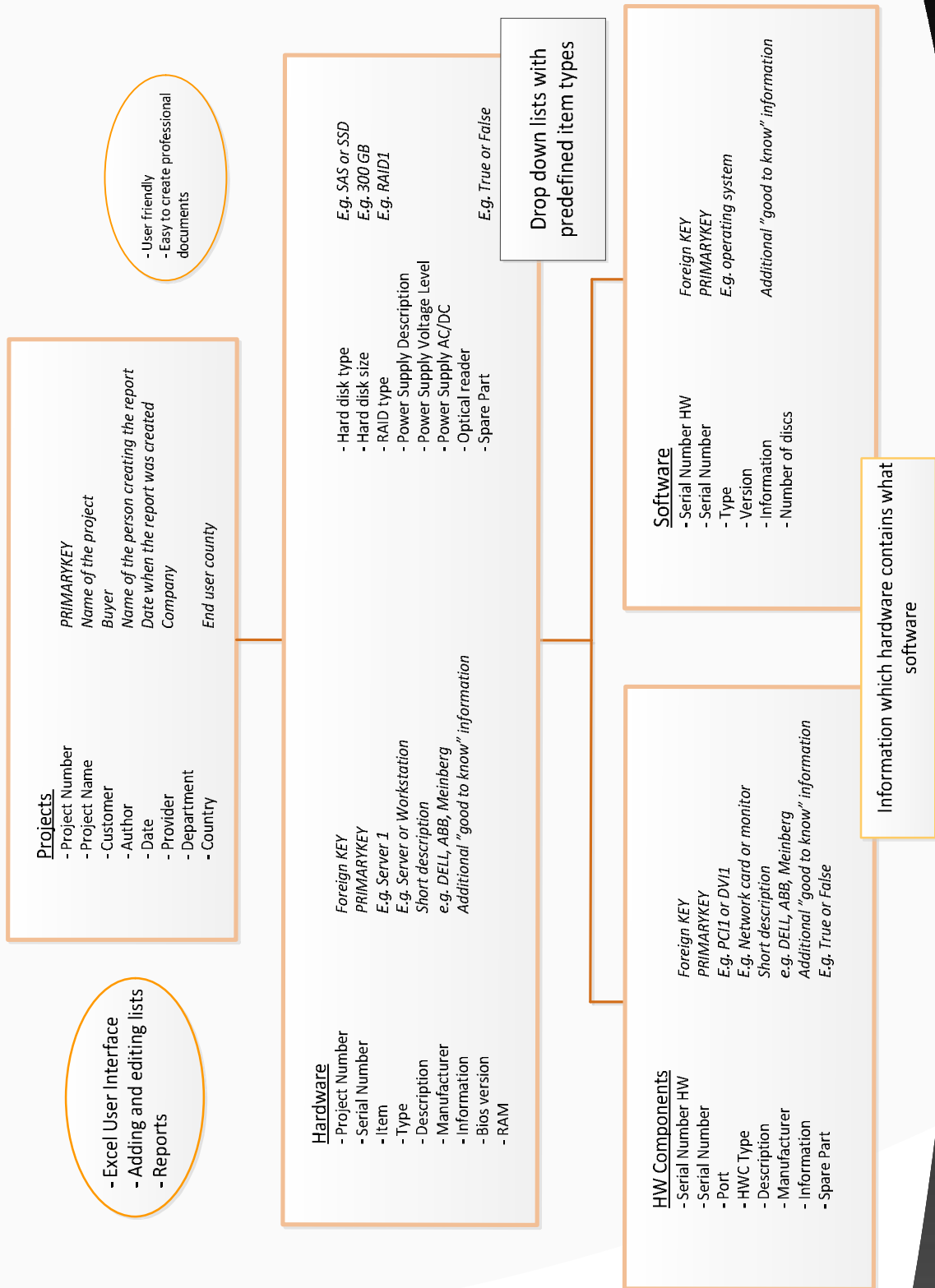
<http://www.abb.com> (read 13.12.2014)

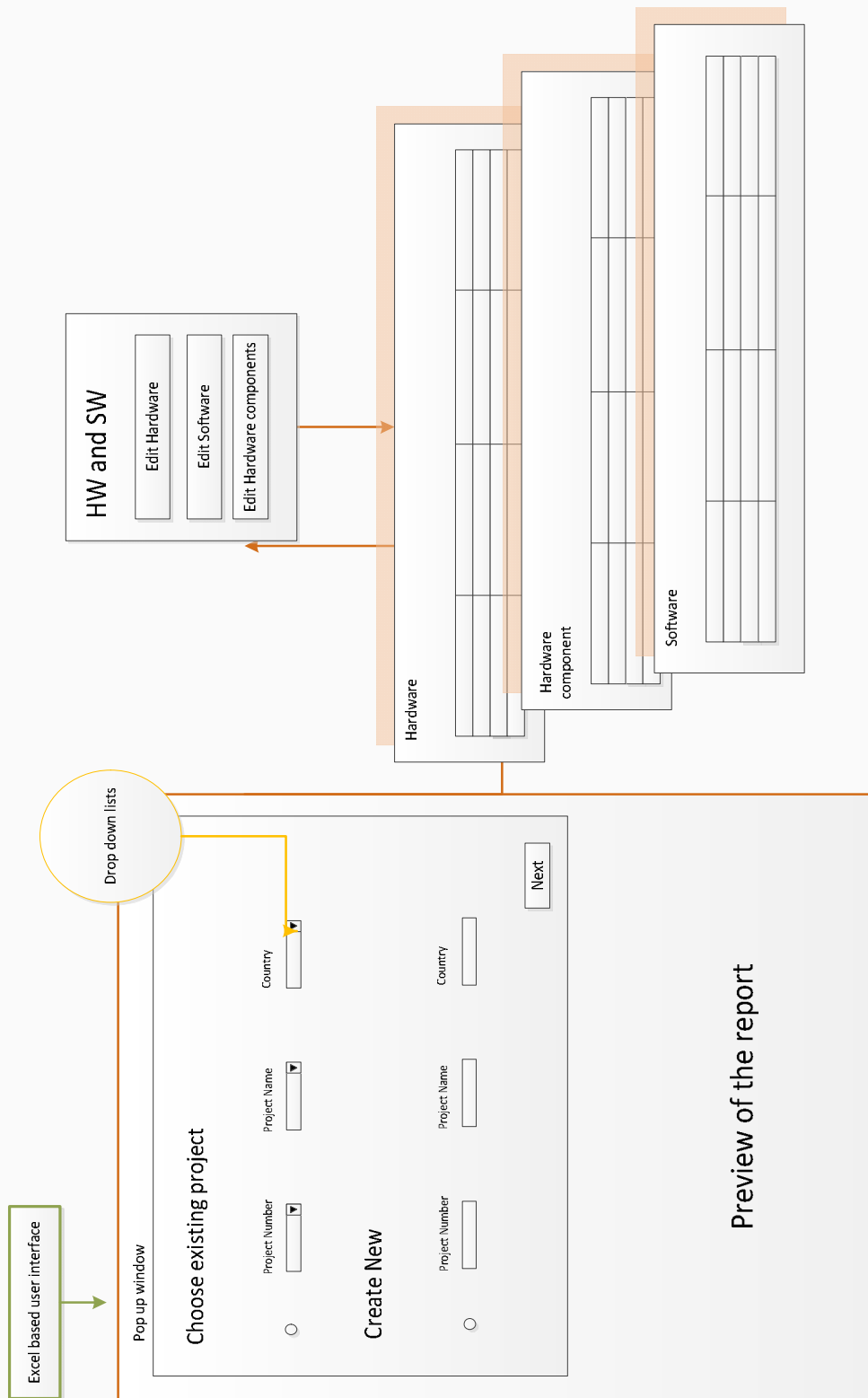
11 Appendix

Appendix 1 – HW & SW database Spec

Appendix 2 – Database user interface

Appendix 3 – User forms in Excel





User forms in Excel

