SCA database and search engine

Vignir Barði Einarsson

Supervisor: Nik Whitehead

Faculty of Information Technology,

University of Akureyri

Submitted April 2006, in partial fulfilment of

the conditions of the award of the degree Bsc.

I hereby declare that this dissertation is all my own work,

except as indicated in the text :

Signature ________________________________

Date ______/______/_______

Abstract

This document contains background information about my final year project. It contains information about what it was that motivated me to do this project as well as what the gain of doing this project was. It contains detailed description of my work, what the project was supposed to achieve, and also how the system is suppose to function. This document also includes some information about the existing system and other related systems. Included are descriptions of my designs and implementations as well.

Table of Contents

Table of Figures

Chapter 1, Introduction

This chapter talks about the background of the project, the reasons for taking on this project, and also what motivated me to do this project.

1.1 Background

The Society for Creative Anachronism also known as SCA is dedicated to research and recreate the European Middle Ages in the present. The society was established in 1966 in Berkeley California and has grown to include over 30,000 members. Every person in the society picks a unique name to use within the society. Members have also the option to register a heraldic device or coat of arms which is essentially a logo and identifies the owner in the same way a company's logo identifies its company. Since the coat of arm identifies the member who displays it, it is very important that no two members have the same coat of arm to prevent confusion.

The purpose of this project is to create a database to store information about the members of the SCA. This database has to store names, arms and other information, for example when the name or the device was first registered. The existing data has to be written into the database and user friendly interface and mechanism to search the data has to be constructed.

1.2 Motivations

When choosing a final year project my first thought was about what it was I wanted to learn from it. From my point of view the final year project is supposed to be a way to add to my existing knowledge from studying computer science. I also saw it as a way to add more practical experience to what I have learned so far. One of my initial goal was to learn a new programming language and to add a practical experience to my existing knowledge of databases. Then choosing a supervisor for my final year project I choose to work with Nik Whitehead since I had a very good experience working with her in the second year group project.

At a meeting with Nik she told me about this project and we figured out that it was well suited for me to learn those things I wanted to learn from doing the final year project. When I started doing my research for the project I did not know anything about the SCA. After examining the existing system to search the SCA records I saw that I had an opportunity to improve it and make it more user friendly which again motivated me to take on this assignment.

1.3 Personal gain

What I gained from doing this project was that I got a hand on experience designing a major programming and database system. I ran into multiple problems because many of my initial plans did not add up when it came to implementation. This taught me a lot about planing a project of this size. When it came to implementing the system I had to learn a number of things in order to implement the project successfully. I learnt two new programming languages, PHP and Perl. I got a good experience importing raw data into a database and learned a lot about the use of regular expressions. I got an experience implementing and designing a user interface for web applications. I also got a hand on experience designing and implementing a database from scratch.

1.4 Other gain

I have recently been informed by my supervisor that the program and the database I constructed is going to be used by the SCA society. As the search engine I constructed while doing this project is user friendlier than the existing system and in some cases more powerful, it is my believe that this project will be a good asset for the SCA.

Chapter 2, Description of the work

This chapter describes what the project is suppose to achieve and contains description of the system functionality.

2.1 What the project is supposed to achieve

The primary objective of this project is to make a database and search engine that is able to search for various information within the SCA records. The records are kept in a flat datafile that is publicly available at the SCA web page which can be located at www.oanda.sca.org. This flat file is roughly 8.5 megabytes in size. The information the systems has to store and be able to return include information about the names of members, the date when the names were first registered and in which kingdom the names were registered. The database also has to contain information about blazons that are registered by the society, the names and the kingdom the blazons were registered in, and also the description of the blazons. The system has to be able to search within the database and effectively return the results. Blazons in the database are registered using old heraldic French which not everybody is familiar with. One of the objective of the project is that a user can search for plain English words when searching for a blazons in the database rather than the heraldic French words.

2.2 How the project is suppose to function

The functionality of the system should be that a user can visit the web page where the system is stored and choose a search that the they would like to conduct. The system should then return a value that matches the search criteria. The secondary objective was to create an administration system where an administrator can log into and add new keywords which users can search for.

Chapter 3, Related work

This chapter describes the existing system.

3.1 Existing system

Existing system is available that can search the records of the SCA is on the web page www.oanda.sca.org. The system consist of six separated web pages which each keeps one search form. The search forms are the following:

1. A Name Search Form which allows the user to search the SCA Armorial database for items associated with a particular name.

2. An Armoury Description Search Form which allows the user to search for registered items that appear under a particular heading in the SCA Ordinary.

3. A Name Pattern Search Form which allows the user to search for items associated with a name.

4. A Blazon Pattern Search Form which allows the user to search for blazons containing particular words or text patterns.

5. A Date/Kingdom Search Form which allows the user to search for registrations during a particular time-period or via a particular kingdom.

6. A Complex Search Form which allows the user to do sophisticated searches on the database by combining the results of multiple searches.

Chapter 4, Design

This chapter covers the design of all the system components and the format of the data files that are used.

4.1 Data files

There are two flat text files available at the SCA search form page. One being the actual datafile containing all registrations within the SCA society which is going to be referred to as registration file from now on in this report. Another file is available containing all the keywords that a heraldry can have, which will be called category file from now on.

4.2 Category file

The category file has two kind of records, one describing attributes and the other describing objects that can have the attributes. An example of these two kinds of records would look like the following:

Example 1, attribute:
"|tincture:sable<dark" here does "tincture:" mean that this record is a colour. The word "sable" means black in the heraldic French and "dark" is a description for people reading the category file

Example 2, object:
"beast, deer|BEAST-DEER AND STAG" the word beast at the beginning of the file means that this record is a part of the main group beast. The word "deer" means that this beast is a deer "BEAST-DEER AND STAG" is the string that is used as keyword when describing the heraldry in the registration file.

The string "BEAST-DEER AND STAG:sable" in the registration file would mean that the heraldry in the datafile contains a black deer.

Information I needed to import from the category file were the words that are most likely to be used by an individual while searching the system. I had to identifying the words that are most likely to be searched for when acquiring heraldry information and write them into the database.

Example 1: |tincture:sable<dark: The word sable is a well known word in heraldry. An indivitual looking for heraldry that contains black object or background would most likely look up the word sable. The word tincture is the group and tells nothing about this being string representing the word black and is therefore not needed. The word <dark does not say much about this being black so just like word tincture it's not needed.

Example 2 : beast, deer|BEAST-DEER AND STAG. In this example the user would look up the word deer. In the registration file the record is registered using the last upper case string BEAST-DEER AND STAG this string is however not something that a user would try to search for when searching for a heraldry containing the word deer.

4.3 Registration file

The registration file contains all informations about registrations within the SCA. The structure of registrations in the file are always the same. Like presented by Iulstan Sigewealding in the document "F*ormat of the SCA Armorial Database"*

"Each record occupies a single line in the file, and each line contains one record. A record has at least five fields, separated by stiles (`|'). The five required fields are: name, source, type, text, and notes, in that order."[1]

The easiest way to describe the records in the registration file is by making an example of two of the records in the file.

Example 1: Antonio di Rienzo|0207D|N||

Looking at the first example we can see that the name associated to this record is Antonio di Rienzo, which in this case is the name that the member chose to use within the society. Next string, 0207D, is the date that the record was first registered and the kingdom that the record was registered in. The number 0207 means that this record was registered in July of 2002 and the D references to that it was registered in the kingdom of Drachenwald. The N between the stiles means that this record describes a Name.

Example 2: Antonio di Rienzo|0210D|d|Gules, a fleur-de-lys within a double tressure Or.||FDL:1:or:spna|FIELD:gules|GU|ORLE AND TRESSURE:2:or:pl:surrounding 1 only

Looking at example 2 we can see that the name associated with this record is the same as in the previous example or Antonio di Rienzo. The date associated with this record which is 0210 means that this record was registered in October of 2002 and the D means that this record was registered in the kingdom of Drachenwald. The d between the stiles means that this record is a coat of arm. Following comes a string that describes the coat of arm which in this case is: “Gules, a fleur-de-lys within a double tressure Or”. “Gules” meaning the colour red, a “fleur-de-lys” is a ornament that is somewhat shaped as a lily, and within “douple tressure Or” means within two golden lines. Following the string comes a description of the heraldry in keywords. “FDL:1:or:spna” In this string “FDL” means a fleur-de-lys, “1“ saying that there is only one of it, “or” meaning that it is gold and “spna” meaning that it is the sole or primary identification of the heraldry. “FIELD:gules” “FIELD” means a field and “gules” standing for the colour red again “GU” also means a red field. Looking at the next string “ORLE AND TRESSURE:2:or:pl:surrounding 1 only” we can see that in this string the words “ORLE AND TRESSURE” means inner boarder not touching the sides of the shield, the number “2” meaning that there are two of them, “pl” means plain lines, and “surrounding 1 only” means that one is surrounding the other. Full description of the datafile can be found in Appendix C.

4.4 Designing the database

An important stage of the design phase was to design a database which could store information about the records in the registration file and in the category file. The first approach to this was a database that contained a specific column for all data in both the category file and in the registration file as well. This approach to the database design looked very appealing on paper but when it came to implementing the queries into the program to search for heraldry in the database this approach became to complicated to be efficient. In order to search for one word the system had to search for the keywords in four columns in two different tables. Even though I had counted for this in the designing of the search system, this became very confusing and complicated to implement. I then went back to the design phase and simplified the database at a later stage in the project. Next design of the database was simpler than the first one and in this design the system needed only to query one column in one of the tables in the database to search for a keyword in a heraldry. Figure 4.1 contains diagram of the database.

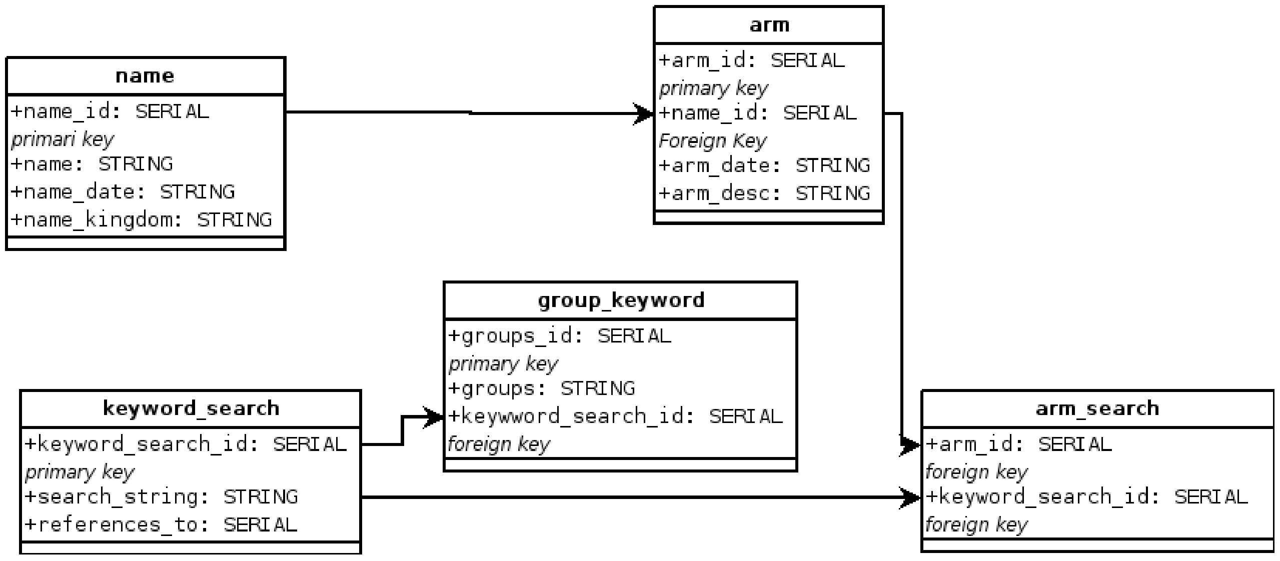


Figure 4.1

As it can been seen in figure 4.1 the final version of the database consists of five tables which are the following.

A table that is called 'name'. This table contains the complete information about the names of all members of the SCA. The table “name” contains the following column:

1. The column “name_id” which is an integer and a primary key for the table “name”.
2. The column “name” which is a string and contains the name that the member has chosen to use within the society.
3. The column “name_date” which is an integer and stores informations about the date the record was first registered.
4. The column “name_kingdom” where each row in the database stores one letter that represents the kingdom the record was registered in.

A table called “arm”. This table stores all information about the heraldic devices the members of the SCA have registered. This table contains the five following columns.

1. The column “arm_id” This column is a integer and primary key for the table “arm”
2. The column “name_id”, Is an integer and a foreign key that maps to the table “name”. As every member in the SCA that registers a heraldic device needs to register his or her name it is not necessary to store the name that the member has chosen twice in both the “arm” table and also in the table “name”. To avoid keeping duplicated value in the database I chose to use foreign key that maps to the “name” table.
3. The column “arm_date” which is an integer that stores the date the arm was registered
4. The column “arm_desc” which is a string that contains the full description of the blazon in the old heraldic French, like it is written by the member.
5. The column “arm_kingdom” This column stores one letter that represents the kingdom the blazon was registered in.

When reading from the category file the value goes into a table called “keyword_search”. This table contains all keywords a user can search for when looking for a heraldry. When an administrator of the system adds a keyword into the system that a user can search for it goes into this table and with a reference to the old heraldic French keyword. This table contains three column

1. The column “keyword_search_id” which is an integer and a primary key for the table.
2. The column “search_string”. This column contains all keywords a user can search for when looking for heraldic devices.
3. The column “references_to” The value of this column identifies if the search string is an old heraldic French word or a plain English word that the administrator of the system has added to the database. If the string in the column “search_string” is one of the keywords imported from the category file then the value of this column is null. If a string in the column “search_string” has been added to the system by the administrator of the system then the value of this column is the “keyword_search_id” of the old heraldic French word that it is supposed to represent.

The table “group_keyword” is only used when writing a data about heraldic devices into the database. The upper case keywords from the category file are read into this table and mapped to the corresponding keyword in the “keyword_search” table. For instance, looking at the string “beast, deer|BEAST-DEER AND STAG”. Here the string “BEAST-DEER AND STAG” is kept in the “group_keyword” table and is mapped to the keyword “deer” in the “keyword_search” table. When reading into the database from the registration file the program queries the “group_keyword” table for the keyword_search_id of the string it is looking for and writes it into the table “arm_search” which we will be looking at next. The table “group_keyword” contains three columns.

1. The column “groups_id” is an integer and primary key for the table.
2. The column “groups” is a string from the category datafile.
3. “keyword_search_id” is a foreign key and maps the corresponding keyword in the “keyword_search” table.

The table “arm_search” maps keywords to heraldic devices. The ID of a heraldic device is written into this table along with the ID of the keywords id contains.

1. “arm_id” which is a foreign key and maps to “arm_id” in the “arm” table
2. “keyword_search_id” Is a foreign key and maps to the “keyword_search” table

4.6 Designing the front end

When it came to designing the user interface my main criteria was to keep it as simple and as user friendly as possible. Also I wanted to keep all the graphics at minimum as it increases the time it takes to load the web page. I decided to design the user interface so that the user could do all of the searches within the same web page. When designing the look of the web page I started of by using paper and pencil to draw it. After doing a number of designs I ended up choosing to use eight check boxes, three text fields and one drop down list. The check boxes are used to decide what the individual wants to search for and what gets returned. The text field is used to input the search words the user wants to search for and the drop down list is used to select which kingdom to search for. The results of a search then appears below the search fields allowing the user to stay at the same web page, which again makes it easy to conduct another search.

4.6 Designing the search engine

Designing the system which searches the database was relatively straight forward process. The whole search system is kept in the same file so no class diagram had to be constructed. The system is designed so that the user can search for a name or a part of a name he or she wants displayed. The user can also wish to have the date the record was registered, the kingdom it was registered in, or text string that represents the coat of arm. The user can select if he or she wants to search for all of the information some of them or just one. It is also possible for the user to choose what information about the record he or she wants returned. For an example if a user would like to search for every person called Aaron in the database and who has the string “sable” in their heraldry, and return heraldry that matches that search criteria, he or she would mark the check box “search for name” as well as the check box “search for device”. Next the user would have to fill in the text fields for both the name and device and mark the check box “return device” and hit the submit button In return the user will obtain information about heraldry that matches the criteria . Altogether there are 225 searches that are possible to construct from the 8 check boxes. Therefore the system needs to be able to construct 225 SQL queries depending on the users search criteria. Number of these SQL queries where constructed and tested on the design phase of this project before implementing them into the program that searches the data. By testing these SQL queries I found a pattern that I was able to use in order to construct the rest of the queries.

4.7 Designing program to import the data

Designs for the program which import the data into the database where insufficient. Which resulted in an increased time it took to construct it. The only designs I did for this part of the project were to construct the SQL queries beforehand. When it came to implementing this part of the project it became apparent to me that it would require more work than I anticipated to construct this program. Due to the lack of designs this process took up to much time allowed for this project.

Chapter 5, Implementation

In this chapter all stages of the implementing of the system are described

5.1 Database

I used PostgreSQL 8.0.4 database in my project. The PostgreSQL is best described at the PostgreSQL web page:

“PostgreSQL is a powerful, open source relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages).”[2]

Installing postgreSQL was an easy process on the Gentoo Linux machine which was used to develop this system. When creating tables and column in the database first thing I did was to use external tool called phpPgAdmin which is a web based administration tool for PostgreSQL. phpPgAdmin came in handy when it came to creating a demo version of the database and debuging the program to import the data. It is a user friendly tool and the graphical user interface makes it easy to browse and alter tables. The down side of the phpPgAdmin tool is that it does have the ability to create primary keys and foreign keys. That is why when it came to implementing the final version of the database I choose to do it in terminal which gave me better control over the whole process. I did not want to make the program that writes into the database create the tables and columns. Therefore the database has to be created before hand and all tables, columns and restrictions have to be created as well. This process took relatively short time when the database had been designed. I created a view called “deviceview” in the database to use when searching for heraldry as the process of searching for heraldry is more complicated than the other searches. By using views I was able to make the part of the search program that constructs the query which searches for heraldry simpler and more efficient. A database dump which contains the structure of the database can be found in Appendix B of this report.

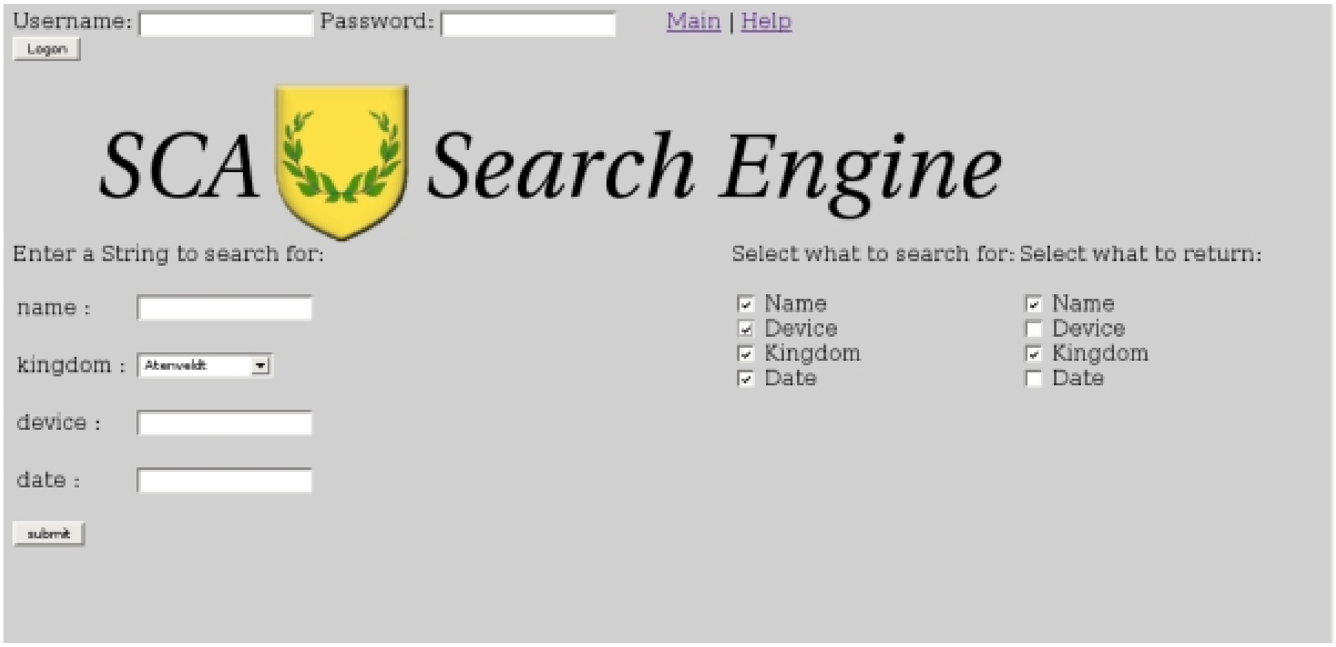
5.2 Importing the data

The program that imports the data is constructed in the Perl programming language. The program can be found in Appendix A of this report under the name data.pl. My reason for choosing the Perl programming language is that it has excellent regular expression functionality. The Perl program breaks down the strings in the two data files, the registration file and the category file. Then it writes it into the database using multiple and complex rules. When writing into the database the Perl script first checks the letter identifying if the record is a heraldry or a name. In other words it checks the single letter within the stiles if it is a N or d, if the letter is a N it breaks down the record and writes it into the name table of the database. First the record gets an unique ID, then the name is written into the name column and then the date is separated from the kingdom. Next the date and the kingdom is written into corresponding column in the database.

If the record is a heraldry it is written into the “arm” table in the database. The record then gets a unique ID and the name ID in the “name” table is registered as a foreign key in the “arms” table and by that linking the heraldry to the name. The arm description is written into the database, and the date and kingdom is separated and written into corresponding columns in the database.

5.3 User interface

The user interface was constructed using HTML web format. The code for the user interface can be found in Appendix A under the name sca.php. The only graphic used was one banner that I constructed to make the web page that keeps the system looking better. The user interface also uses javascript to hide the text fields for the searches that are not being used. When a user marks that he or she wants to search for name the text field name appears, if the user clears the check box the text field disappears again, the same happens when marking the other check boxes. The user interface also contains a help page containing text that explains how to conduct a search within the system. A screen shot of the main page of the system can be seen in figure 5.1


Username: Password: Main | Help

Logon

SCA Search Engine

Enter a String to search for:

name :

kingdom : Atenveldt

device :

date :

submit

Select what to search for:

- ✓ Name
- ✓ Device
- ✓ Kingdom
- ✓ Date

Select what to return:

- ✓ Name
- ☐ Device
- ✓ Kingdom
- ☐ Date

Figure 5.1

5.4 Searching the data

The program that searches the database was constructed using the PHP scripting language which is used to create the dynamic web page. Or like the PHP programming language is described in the official PHP manual.

“PHP, which stands for "PHP: Hypertext Preprocessor" is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated webpages quickly, but you can do much more with PHP.”[3]

The PHP program that I constructed can be found in Appendix A under the name “namesearch.php”. The primary function of this part of the system is to construct SQL queries and execute them based on the users search criteria. The program starts of with three strings: “SELECT” “FROM” & “WHERE”. The program then adds to the strings depending on the users search criteria and search strings. After the program has construct the SQL queries the query is executed and the results are returned.

Let's look at an example used in a previous chapter and pretend that the user searches for every person called Aaron in the database who has the string “sable” in their heraldry. The user then marks the check box “search for name” as well as the check box “search for device”, then fills in the text fields for both the name and device and marks the check box “return device”.

After hitting the submit button the search system checks what the user is searching for and recognizes that the user is searching for a string in a heraldry and adds the string “deviceview” to the string “FROM” which then looks like: “FROM deviceview”. The system then adds the string to look for to the heraldry. The “WHERE” string then looks like “WHERE search_string = 'sable'” the system then checks to see what else to look for and sees that the user wants to search also for the name and adds to the “WHERE” string which then looks like

“WHERE search_string = 'sable' AND names ~~* '%$name%” The system then checks what to return to the user and recognizes that it is supposed to return a string that contains the heraldry that belongs to members that match the search criteria plus adds to the “SELECT” string which then looks like: “SELECT arms_desc”. The system adds all of the strings together and executes the SQL query that looks like this: “SELECT arms_desc FROM deviceview WHERE search_string = 'sable' AND names ~~* '%$name%”. Of course this example is just one of the 255 searches available to a user but should give a rough idea of how the system works. The search for name functionality of the system is constructed so a user can search for a part of name and without it being case sensitive.

The original plan was to construct this program using the C++ programming language and use common gateway interface to interact with a web page. After having looked closer at that possibility I decided against it. One of the reasons for changing from C++ to PHP was that PHP is better suited to work with web pages and I was interested in learning PHP as it is being used in many areas of the IT business.

5.5 Administration System

Creating a specified administration system was not a part of the original plan and was an idea that came up in the process while implementing the search system. The functionality of this part of the system is that it allows an administrator to log into an administration section of the system and choose option to add to the database. The administrator then inputs the old heraldic word into one of the text fields and the new plain English word into another. The system then queries the “keyword_search” table to find the old word and writes the new word into the same table adding the keyword search ID to the “reference_to” column in the database. This system uses the MD5 algorithm to encrypt the administrator password and make sure that no unauthorised persons can write into the database, The MD5 algorithm is a built in function in the PHP programming language and is sufficient encryption for a system like this. Figure 5.2 contains screen shot of the web page containing the administration system. The code for the administrator system can be found in Appendix A under the name admin.php.

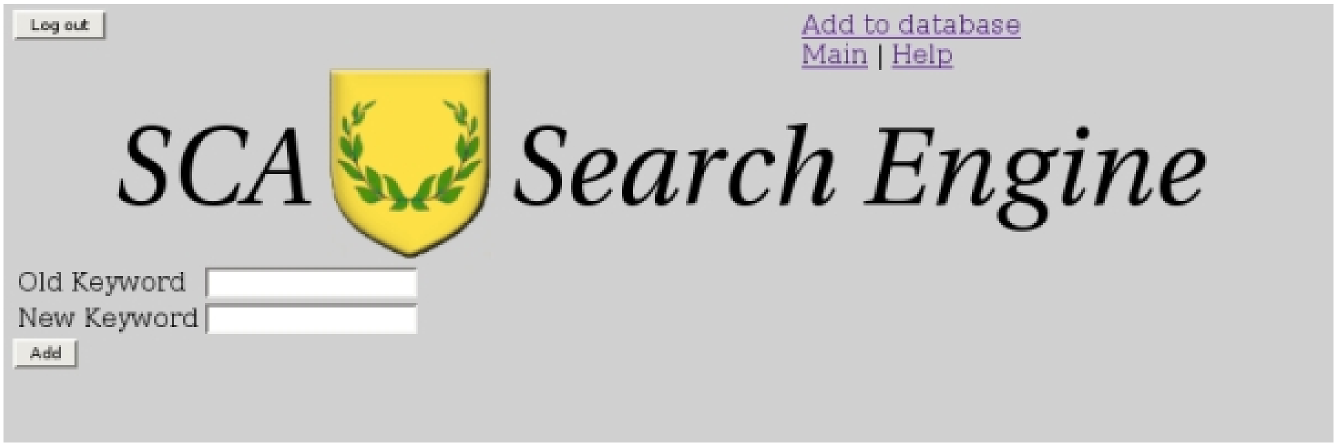


Figure 5.2

Chapter 6, Evaluation

My goals for this project were to create a database for the SCA data, upload the data, create a user friendly interface and adding to its functionality so it is able to search for plain English words within the database rather than the old heraldic words. In the beginning of this project I did not plan on having users able to combine searches the way I ended up doing and it was not a part of my initial plans as well to create an administration system which allowed new keywords to be searched for. As far as I see I have met all of my goals, plus added some extra functionality to the system. The system I have created is fully functional in doing all of the above. While working on this project I have learned new programming languages, I have learned how to work with raw data files, and I have also learned how to design and implement databases. I believe that the experience and knowledge I have gained from doing this project will be a great asset for me in the future.

References

[1] Iulstan Sigewealding, Format of the SCA Armorial Database, viewed 20 November, 2005, <http://oanda.sca.org/data_format.html>*.*

[2] The PHP group, PHP manual, viewed 14 Mars, 2006, <http://www.php.net/manual/en/preface.php>

[3] PostgreSQL Global Development Group, About, viewed 10 April, 2006 <http://www.postgresql.org/about/>

Bibliography

Bennet S , *Farmer R &* McRobb S, 2002, *Object-Oriented System Analysis And Design*, McGraw-Hill, Maidenhead, Berkshire.

Miklós Sándorfia, Heraldic Display, viewed 22 November, 2005, <http://andrew.draskoy.net/sca/eilean/display/>.

Ramakrishnan, R & Gehrke J, 2003, Database management Systems, McGraw-Hill, New York, NY

Siever, E, 1999, Perl in a Nutshell, O'Reilly, Sebastopol, CA

The Society for Creative Anachronism*, What is an S.C.A. Heraldic Device, viewed 24 November, 2005,* <http://www.sca.org/heraldry/laurel/whatis/device.html>*.*

Appendix A (Source code)

admin.php

```
<?php
if (!isset($_POST['submit']))
{
}

if ($_POST['where'] == '1')
{

$adminSearchString = $_POST['opt1'];
$adminInputQuery = $_POST['opt2'];

$dbconn = pg_connect("host=localhost dbname=oandadb user=vignir password=casio99")
or die('Could not connect: ' . pg_last_error());

$adminSelect = "SELECT keyword_search_id FROM keyword_search WHERE search_string =
'$adminSearchString'";
$result = pg_query($adminSelect) or die('The old keyword does not match to keywords in the
database');
$keyword_search_id = pg_fetch_result($result, 0, 0);

$maxSelect = "SELECT MAX (keyword_search_id) AS maxID FROM keyword_search";
$result = pg_query($maxSelect) or die('Query failed: ' . pg_last_error());
$MaxID = pg_fetch_result($result, 0, 0);
$MaxID ++;

$adminInputQuery = "INSERT INTO keyword_search (keyword_search_id, search_string,
references_to)  VALUES ('$MaxID' , '$adminInputQuery', $keyword_search_id)";
$result = pg_query($adminInputQuery) or die('Query failed: ' . pg_last_error());
}
?>
```

namesearch.php

```
<?PHP


if ($searchName != 'on' && $searchDevice != 'on' && $searchKingdom != 'on' && $searchDate !=
'on')

{ echo "Must select what to search for  <META HTTP-EQUIV='REFRESH' CONTENT='5'; URL=' .
basename($PHP_SELF) . '> <br>";};


if ($returnName != 'on' && $returnDevice != 'on' && $returnKingdom != 'on' && $returnDate !=
'on')

{ echo "Must select what return  <META HTTP-EQUIV='REFRESH' CONTENT='5'; URL=' .
basename($PHP_SELF) . '> <br> ";};


        $dbconn = pg_connect("host=localhost dbname=oandadb user=vignir password=casio99")
        or die('Could not connect: ' . pg_last_error());



$selectQuery = ' SELECT DISTINCT ';
$fromQuery  = 'FROM';
$whereQuery = 'WHERE';

       if ($searchDevice == 'on'){
             $whereQuery .=" ( ";
             $fromQuery .= " deviceview ";
             $searchDeviceArray = explode(" ",$device);
             foreach ($searchDeviceArray as $searchString){
                   if ($gaur == 1)  $whereQuery .= " OR ";
                         $res = pg_query($dbconn, "SELECT references_to FROM keyword_search
WHERE search_string = '$searchString'");
                          if ($res) {
                              if (pg_field_is_null($res, 0, "references_to") == 1) {
                                $whereQuery .=" search_string = '$searchString' ";
                                 }
                              if (pg_field_is_null($res, 0, "references_to") == 0) {
                                $val = pg_fetch_result($res, 0, 0);
                                $query1 = pg_query($dbconn, "SELECT search_string FROM
keyword_search WHERE keyword_search_id = '$val'");
                                $val = pg_fetch_result($query1, 0, 0);
                                    $whereQuery .=" search_string = '$val' ";
                         }
                         }
                   $gaur = 1;

             }

                   $whereQuery .=" ) ";

                   if ($returnName =='on'){
                   $selectQuery .= " names ";
             }
                   if ($returnName =='on' && ($returnDevice =='on' || $returnKingdom =='on'
|| $returnDate =='on')) $selectQuery .= " , ";
                   if ($returnDevice =='on'){
                       $selectQuery .= " arm_desc ";
                }
```

```
                    if ($returnDevice =='on' && ($returnKingdom =='on' || $returnDate
=='on')) $selectQuery .= " , ";
                    if ($returnKingdom =='on'){
                        $selectQuery .= " name_kingdom ";
                }

                    if ($returnKingdom =='on' && $returnDate =='on') $selectQuery .= " , ";
                     if ($returnDate =='on'){
                        $selectQuery .= " name_date ";
                }
             if ($searchName == 'on'){
                    $whereQuery .="AND names ~~* '%$name%' ";
             }
             if ($searchKingdom == 'on'){
                    $whereQuery .="AND name_Kingdom ='$kingdom' ";
             }
             if ($searchDate == 'on'){
                    $whereQuery .="AND name_date ='$date' ";
             }
}

if ($searchName == 'on' && $searchDevice != 'on'){
      $fromQuery .= ' names ';
      $whereQuery .= ' names.names ';
             if ($returnName == 'on'){
                    $selectQuery .= ' names.names ';
                    }

                    if (($returnDevice == 'on' || $returnKingdom == 'on' || $returnDate ==
'on') && $returnName == 'on') $selectQuery .= ', ';

             if ($returnDevice == 'on'){
                    $selectQuery .= ' arm.arm_desc ';
                    $fromQuery .= ' LEFT OUTER JOIN arm ON (names.name_id = arm.name_id) ';
                     }

                     if (($returnKingdom == 'on' || $returnDate == 'on') && $returnDevice
=='on') $selectQuery .= ', ';

             if ($returnKingdom == 'on'){
                    $selectQuery .= ' names.name_kingdom ';
             }
                     if ($returnDate == 'on' && $returnKingdom =='on') $selectQuery .= ', ';
      //Fix this in the db
             if ($returnDate == 'on'){
                    $selectQuery .= ' names.name_date ';
      };
      $whereQuery .= " ~~* '%$name%' ";
}

if (($searchName =='on' && $searchDevice != 'on') &&  ($searchKingdom == 'on' || $searchDate
=='on')) { $whereQuery .= " AND "; $fromQuery  = 'FROM'; $selectQuery = 'SELECT';};

if ($searchKingdom == 'on' && $searchDevice != 'on'){
       $fromQuery .= ' names ';
       $whereQuery .= ' names.name_kingdom ';

                if ($returnName == 'on'){
                        $selectQuery .= ' names.names ';
                        }
                    if (($returnDevice == 'on' || $returnKingdom == 'on' || $returnDate ==
'on') && $returnName == 'on') $selectQuery .= ', ';
```

```
                if ($returnDevice == 'on'){
                        $selectQuery .= ' arm.arm_desc ';
                         $fromQuery .= ' LEFT OUTER JOIN arm ON (names.name_id =
arm.name_id) ';
                         }
                    if (($returnKingdom == 'on' || $returnDate == 'on') && $returnDevice
=='on') $selectQuery .= ', ';

                if ($returnKingdom == 'on'){
                        $selectQuery .= ' names.name_kingdom ';
                }
                    if ($returnDate == 'on' && $returnKingdom =='on') $selectQuery .= ', ';
             if ($returnDate == 'on'){
                        $selectQuery .= ' names.name_date ';
        };
        $whereQuery .= " = '$kingdom' ";
}


if ($searchKingdom == 'on' && $searchDate =='on' && $searchDevice != 'on') { $whereQuery .=
" AND "; $fromQuery  = 'FROM'; $selectQuery = 'SELECT';};

if ($searchDate == 'on' && $searchDevice != 'on'){
        $fromQuery .= ' names ';
        $whereQuery .= ' names.name_date ';

                if ($returnName == 'on'){
                        $selectQuery .= ' names.names ';
                        }

                     if (($returnDevice == 'on' || $returnKingdom == 'on' || $returnDate ==
'on') && $returnName == 'on') $selectQuery .= ', ';

                if ($returnDevice == 'on'){
                        $selectQuery .= ' arm.arm_desc ';
                        $fromQuery .= ' LEFT OUTER JOIN arm ON (names.name_id = arm.name_id)
';
                         }
                     if (($returnKingdom == 'on' || $returnDate == 'on') && $returnDevice
=='on') $selectQuery .= ', ';

              if ($returnKingdom == 'on'){
                        $selectQuery .= ' names.name_kingdom ';
                }
                     if ($returnDate == 'on' && $returnKingdom =='on') $selectQuery .= ', ';

             if ($returnDate == 'on'){
                        $selectQuery .= ' names.name_date ';
        };
        $whereQuery .= " = '$date' ";
}


$query = $selectQuery . $fromQuery . $whereQuery;

$result = pg_query($query) or die('No results matching the search string');
$modPrint = 0;
if ($returnName == 'on') {$nameCount = $modPrint; $modPrint ++;}
if ($returnDevice == 'on') {$devCount= $modPrint; $modPrint ++;}
if ($returnKingdom == 'on') {$kingCount = $modPrint; $modPrint ++;}
if ($returnDate == 'on') {$dateCount = $modPrint; $modPrint ++;}
  $output = '<br> <br> <div id="output">';
          while ($line = pg_fetch_array($result, NULL , PGSQL_ASSOC)) {
```

```
			$resultCounter ++;
			$output .= " Result nr: $resultCounter <br> ";
				foreach ($line as $col_value) {
					$counterinn = $i % $modPrint;
					if ($returnName == 'on' && $counterinn == $nameCount) $output .= "
Name :";
					if ($returnDevice == 'on' && $counterinn == $devCount) $output .=
" Device : ";
					if ($returnKingdom == 'on' && $counterinn == $kingCount) {
						$output .= " Kingdom : ";
						switch ($col_value) {
							case 'A': $col_value = "Atenveldt "; break;
									case 'C': $col_value = "Caid "; break;
									case 'D': $col_value = "Drachenwald ";
break;
									case 'E': $col_value = "East "; break;
									case 'G': $col_value = "Gleann Abhann ";
break;
									case 'H': $col_value = "AEthelmearc ";
break;
									case 'K': $col_value = "Calontir "; break;
									case 'L': $col_value = "Laurel or SCA ";
break;
									case 'M': $col_value = "Middle "; break;
									case 'm': $col_value = "Ealdomere "; break;
									case 'n': $col_value = "Northshield ";
break;
									case 'N': $col_value = "An Tir "; break;
									case 'O': $col_value = "Outlands "; break;
									case 'Q': $col_value = "Atlantia "; break;
									case 'R': $col_value = "Artemisia "; break;
									case 'S': $col_value = "Meridies "; break;
									case 'T': $col_value = "Trimaris "; break;
									case 'W': $col_value = "West "; break;
									case 'w': $col_value = "Lochac "; break;
									case 'X': $col_value = "Ansteorr "; break;
						}
					}
					if ($returnDate == 'on' && $counterinn == $dateCount) $output .= "
Date : ";
					$i++;
					$output .= "$col_value\n\n";
					$output .= '<br>';
				}
			$output .= '<br>';
			$output .= '<hr>';
			$output .= '<br>';
		}
pg_free_result($result);
pg_close($dbconn);
$output .= '</div>';
echo $output;
?>
```

sca.php

```
<?
$user = 'vignir';
$pass = '6a204bd89f3c8348afd5c77c717a097a';
if ($_POST['logon'] == '1')
{
setcookie('logon['.$_POST['username'].']', md5($_POST['password']));
echo '<script> document.location = "sca.php" </script>';
}
if($_POST['logon'] == '2')
{
setcookie('logon['.$user.']', $pass, time() - 3600);
echo '<script> document.location = "sca.php" </script>';
}
if (isset($_COOKIE['logon']))
{
	foreach ($_COOKIE['logon'] as $username => $password)
	{
	if ($user == $username && $pass = $password)
		{
		$loggedin = 1;
		}
	else
		{
		setcookie($username, $password, time() - 3600);
		}
	}
}
if ($_GET['e'] != NULL && $loggedin != 1)
{
die ('kemur');
}
if ($_GET['f'] != NULL && $loggedin != 1)
{
die ('kemur');
}
if ($_GET['f'] != NULL)
{
include 'admin.php';
}
?><html>
<head>
<title>SCA Search page</head>
<body bgcolor="#d0d0d0">

<div style="float:left;width:50%">
<form method="POST">
<?php
if ($loggedin != 1)
{
?>
Username: <input type="text" name="username">
Password: <input type="password" name="password">
<input type="hidden" name="logon" value="1">
<input type="submit" value="Logon">
<?php
}
else
{
echo '<input type="hidden" name="logon" value="2">';
echo '<input type="submit" value="Log out">';
```

```
}
?>
</form>
</div>
<?php
if($loggedin == 1)
{
?>
<div style="float:left;width:50%">
<a href="sca.php?e=add">Add to database</a>
</div>
<?php } ?>
<img src="banner.png" alt ="Banner" align="left" width="801" height="122">
<div id="nav" style="height:122;"><a href="sca.php?s=">Main</a> | <a
href="sca.php?s=help">Help</a></div>
<br><br>
<?php
if ($_GET['s'] == NULL && $_GET['e'] == NULL || $_GET['s'] == 'main')
{
?>
<div id="Layer3" style="float:left;width:55%">

Enter a String to search for:
<br>
<br>


<?php
function skila()
{
if (isset($_POST['submit']))
       {
        $name = $_POST['name'];
        $searchName = $_POST['searchName'];
       $device = $_POST['device'];
       $searchDevice = $_POST['searchDevice'];
        $kingdom = $_POST['kingdom'];
        $searchKingdom = $_POST['searchKingdom'];
        $date = $_POST['date'];
        $searchDate = $_POST['searchDate'];
        $returnName = $_POST['returnName'];
        $returnDevice = $_POST['returnDevice'];
        $returnKingdom = $_POST['returnKingdom'];
        $returnDate = $_POST['returnDate'];
       include 'namesearch.php';
       }
}
?>

<SCRIPT language="JavaScript" SRC="js_toggleVis.js" TYPE="text/javascript"></SCRIPT>

<form name="input" action="" method="POST">
<span id="nameField" name="nameField" style="DISPLAY: none">
<table border ="0">
<colgroup span="1">
<col width="90">
<td>name :</td>
</col>
</colgroup>
<td><input type="text" name="name"></td>
</table>
<br>
</span>
```

```
<span id="kingdomField" name="kingdomField" style="DISPLAY: none">
<table border ="0">
<colgroup span="1">
<col width="90">
<td>kingdom :
</col>
</colgroup>
<!--</td><td><input type="text" name="kingdom"></td>-->
</td><td>
<select name="kingdom">
<option  value ="A">Atenveldt
<option value  ="C">Caid
<option value ="D">Drachenwald
<option value ="E">East
<option value ="G">Gleann Abhann
<option value ="H">AEthelmearc
<option value ="K">Calontir
<option value ="L">Laurel or SCA
<option value ="M">Middle
<option value ="m">Ealdomere
<option value ="n">Northshield
<option value ="N">An Tir
<option value ="O">Outlands
<option value ="Q">Atlantia
<option value ="R">Artemisia
<option value ="S">Meridies
<option value ="T">Trimaris
<option value ="W">West
<option value ="w">Lochac
<option value ="X">Ansteorra
</td>
</select>

</table>
<br>
</span>
<span id="deviceField" name="deviceField" style="DISPLAY: none">
<table border ="0">
<colgroup span="1">
<col width="90">
<td>device :</td>
</col>
</colgroup>
<td><input type="text" name="device"></td>
</table>
</br>
</span>
<span id="dateField" name="dateField" style="DISPLAY: none">
<table border ="0">
<colgroup span="1">
<col width="90">
<td>date :</td>
</col>
</colgroup>
<td><input type="text" name="date"></td>
</table>
<br>
</span>

<input type="submit" value="submit" name="submit">

</div>
```

```
<div id="Layer4" style="float:left; width:22%" >
Select what to search for:
<br>
<br>
<input type="checkbox" name="searchName" onClick="toggleVis('nameField')">
Name
<br>
<input type="checkbox" name="searchDevice" onClick="toggleVis('deviceField')">
Device
<br>
<input type="checkbox" name="searchKingdom" onClick="toggleVis('kingdomField')">
Kingdom
<br>
<input type="checkbox" name="searchDate" onClick="toggleVis('dateField')">
Date
</div>

<div id="Layer5" style="float:left; width:23%">
Select what to return:
<br>
<br>
<input type="checkbox" name="returnName">
Name
<br>
<input type="checkbox" name="returnDevice">
Device
<br>
<input type="checkbox" name="returnKingdom">
Kingdom
<br>
<input type="checkbox" name="returnDate">
Date
</form>
</div>

<div id="Layer6" style="float: left; width:80%; clear: both;" align="left">
<?php skila(); ?>
</div>
<?php
}
elseif ($_GET['s'] == 'help')
{
?>
<div id="help" style="float:center;width:80%">
<?php
echo    'There are eight check boxes that can be marked when conducting a search within this
system.
        Four of the check boxes are used to choose what to search for and the other four are
used to
        select what values to return. When marking the \'search for\' check boxes \'name\',
\'date\' or \'device\' a text field
        appears where the search string is typed in. When marking the check box \'kingdom\'
a
        drop down list appears containing all kingdoms registered within the SCA. There are
also
        check boxes to choose what values to be returned. To conduct a search at least one
of the check box from each
        group must be marked. The system allows searches to be combined in any way a user
wants.
        After selecting the search criteria and fill in the text fields, hit the submit
button and the system returns
        all records that match the search criteria';
```

```
?>
</div>
<?php
}
elseif ($_GET['e'] == 'add')
{
echo '<form action="sca.php?f=add"  method="post">';
echo '<table border="0">';
$options = array('Old Keyword'=>'opt1', 'New Keyword'=>'opt2');
foreach ($options as $op => $tions)
{
echo '<tr>';
echo '<td>'.$op.'</td><td><input type="text" name="'.$tions.'"></td>';
echo '</tr>';
}
echo '</table>';
echo '<input type="hidden" name="where" value="1">';
echo '<input type="submit" value="Add">';
echo '</form>';
}
?>


</body>
</html>
```

data.pl

```
#!/usr/bin/perl
use DBI;

$dbh = DBI->connect ( "dbi:Pg:dbname=oandadb", "", "");
if ( !defined $dbh ) {
die "Cannot connect to database!\n";
}

myCat();

$id_name = 0;

open (FILE_HANDLE, oanda) || die "The file does not exist\n";

while(<FILE_HANDLE>){

       $str =$_;
       chomp ($str);
              if ($str =~ m/\|d\|/) {arms ($str); arm_search($str)};
              if ($str =~ m/\|N\|/) {name ($str)};
       }

close (FILE_HANDLE);
sub name{
       #Prepare data for table Name
       $name_id ++;
       $names = $str;
       $names =~ s/\|.*//;
         $names =~ s/'/\\'/g;

       $name_date = $str;
       $name_date =~ s/\|N\|.*//;
       $name_date =~ s/.*\|//;
       $name_date =~ s/\D//;
       if ($name_date =~ m/-/) {
              $name_date =~ s/.*-//;
              $name_date =~ s/\D//;
              $name_kingdom = $str;
                $name_kingdom =~ s/\|N\|.*//;
              $name_kingdom =~ s/.*\|//;
                     $name_kingdom =~ s/.*\d//;
              name_table ($name_id, $name, $name_date, $name_kingdom);   ##Uncomment to add
to table name
              $name_id ++;
              $name_date = $str;
              $name_date =~ s/\|N\|.*//;
              $name_date =~ s/.*\|//;
              $name_date =~ s/-.*//;
              $name_date =~ s/\D//;
       }

       $name_kingdom = $str;
         $name_kingdom =~ s/\|N\|.*//;
         $name_kingdom =~ s/.*\|//;
         $name_kingdom =~ s/.*\d//;
       name_table ($name_id, $names, $name_date, $name_kingdom);  ##Uncomment to add to
table name
}

sub name_table{
        $sth = $dbh->do("
        INSERT INTO names (name_id, names, name_date, name_kingdom) VALUES ($name_id,
'$names', $name_date, '$name_kingdom')
        ");
        if ( !defined $sth ) {
        die "name_table Cannot prepare statement: $DBI::errstr\n";
        }
}

sub arms{
        #Table Arms
        $arm_id ++;
        $device_foreign_key = $id_name;
        $arm_date = $str;
        $arm_date =~ s/\|d\|.*//;
        $arm_date =~ s/.*\|//;
          if ($arm_date =~ m/-/) {
                $arm_date =~ s/.*-//;
                $arm_date =~ s/\D//;
                arms_table ($arm_id, $arm_date, $arm_kingdom);        ##Uncomment to add to
table name
                $arm_id ++;
                $arm_date = $str;
                $arm_date =~ s/\|d\|.*//;
                $arm_date =~ s/.*\|//;
                $arm_date =~ s/-.*//;
        }
          if ($arm_date eq ''){$arm_date = '0000'; $arm_kingdom = 'A';}

        $arm_date =~ s/\D//;
        $arm_kingdom = $str;
          $arm_kingdom =~ s/\|d\|.*//;
          $arm_kingdom =~ s/.*\|//;
          $arm_kingdom =~ s/.*\d//;


        $arm_desc = $str;
        $arm_desc =~ s/.*\|d\|//;
        $arm_desc =~ s/\|.*//;
        $arm_desc =~ s/'/\\'/g;
        $arm_desc =~ s/,/\\,/g;
        $dev_field1 = $str;
        $dev_field1 =~ s/.*\.\|//;
        arms_table ($arm_id, $arm_date, $arm_kingdom, $arms_desc); ##Uncomment to add to
table arms
}
sub arm_search{
        $compare_string = $str;
        $compare_string =~ s/.*\.\|//;
        $compare_string =~ s/.*\]\|//;
        $compare_string =~ s/.*\?\|//;
        $compare_string =~ s/'/\\'/g;
        $i = 0;
        $i_2 = 0;
        @some_arr = split(/\|/, $compare_string);
        splice (@some_arr, 0, 1);
                foreach (@some_arr) {
                        @some_arr_2 = split (/:/, $some_arr[$i]);
                        $i++;
                        $check =  $some_arr_2[0];
                        splice (@some_arr_2, 0, 1);
                        table_arms_group();
```

```
                              foreach (@some_arr_2){

                                     $check_attr =  @some_arr_2[$i_2];

                                     $i_2 ++;
                                     table_group_attribute();
                              }
                       $i_2 =0;
       }
}
sub table_group_attribute{
              #Insert into table Arms
              $sth = $dbh->prepare("
              SELECT keyword_search_id FROM keyword_search WHERE search_string =
'$check_attr';
              ");
              if ( !defined $sth ) {
              die "table group attribute SELECT Cannot prepare statement: $DBI::errstr\n";
              }

              $sth->execute;
              #insert into table arms search
              $sth = $dbh->do("
              INSERT INTO arm_search (arm_id, keyword_search_id) VALUES ($arm_id,
$keyword_search_id)
              ");
              if ( !defined $sth ) {
              die "table group attribute INSERT Cannot prepare statement: $DBI::errstr\n";

              }
}
sub table_arms_group{

              #Insert into table Arms
              $sth = $dbh->prepare("
              SELECT keyword_search_id FROM group_keyword WHERE groups = '$check';
              ");
              if ( !defined $sth ) {
              die "table arms group SELECT Cannot prepare statement: $DBI::errstr\n";
              }


              $sth->execute;
              #while ($row = $sth->fetchrow_array()) {
              # }
              $row =  $sth->fetchrow_array();
              $arms_group_id++;
              if ($check eq '(changed/released)'){$row = 10000};  ##Hax til að redda
(changed/released)
              $sth = $dbh->do("
                INSERT INTO arm_search (arm_id, keyword_search_id) VALUES ( $arm_id, $row)
                ");
                if ( !defined $sth ) {
                die "table arms grouop INSERT Cannot prepare statement 1: $DBI::errstr\n";
                                                          }
}

sub arms_table{
       #Insert into table Arms
       $sth = $dbh->do("
       INSERT INTO arm (arm_id, name_id, arm_date, arm_desc, arm_kingdom) VALUES ($arm_id,
$name_id, $arm_date, '$arm_desc', '$arm_kingdom' )
       ");
```

```
        if ( !defined $sth ) {
        die "arms table INSERT Cannot prepare statement: $DBI::errstr\n";
        }
}

sub myCat #Reads from mycat and splits it up
{

open (CAT_FILE,mycat)|| die "the file does not existi\n";

$groups_id = 0;
$keyword_search_id = 0;
$references_to = 'NULL';


        while(<CAT_FILE>) {

                $cat_str = $_;
                chomp ($cat_str);
                $search_string = $cat_str;
                $cat_arg_group = $cat_str;
#Table Attribute attr_name

        $groups = $cat_str;
                if ($groups =~ m/.+\|/){
                        $groups =~ s/'/\\'/g;
                        $groups =~ s/.*\|//;
                        group_keyword($groups_id, $groups , $keyword_search_id);
##Uncomment to add to table a_group
                        $groups_id ++;
                }


                if ($search_string =~ m/^\|/){   ####add function to read in new search words

                        $search_string =~ s/\;.*//;
                        $search_string =~ s/\|.*\://;
                        $search_string =~ s/\<.*//;
#                       print "keyword_search_id   : $keyword_search_id,   search_string   :
$search_string \n ";
                        ###@args_array [$attr_ID] = $attr_Name;
                        keyword_search ($keyword_search_id, $search_string); ### Uncomment to
add into attr_name

                        $keyword_search_id ++;
                }


#Table group  g_Group
                if ($search_string =~ m/.+\|/){
                        $search_string =~ s/\;.*//;
                        $search_string =~ s/'/\\'/g;
                        $search_string =~ s/\,.*//;
                        $search_string =~ s/\|.*//;
                                keyword_search ($search_string, $keyword_search_id); ##Uncomment
to add to table g_group
                                $keyword_search_id ++;

                }

                   $search_string = $cat_str;
                   if ($search_string =~ m/\;/){
                           $new_keyword = $search_string;
                        $new_keyword =~ s/.*\;//;
                        $new_keyword =~ s/\|.*//;
                        @keyword_arr = split(/\+/, $new_keyword);
                        $references_to = $keyword_search_id - 1;
                                foreach (@keyword_arr) {
                                $search_string = @keyword_arr[$i];
                                keyword_search ($search_string, $keyword_search_id,
$references_to);    ##Uncomment to add to table g_group

                                $keyword_search_id ++;
                                $i++;
                          $group_error ++;
                          }
                                $references_to = 'NULL';

                    $i = 0;
                    }
      }
}
sub keyword_search{ #inserts values into table attribute
      #Insert into table attribute
      $sth = $dbh->do("
      INSERT INTO keyword_search (keyword_search_id, search_string, references_to) VALUES
($keyword_search_id, '$search_string', $references_to)
      ");
      if ( !defined $sth ) {
      die "attribute table Cannot prepare statement: $DBI::errstr\n";
      }
}

sub group_keyword{ #inserts values into table attribute
      #Insert into table attribute_group
      $sth = $dbh->do("
      INSERT INTO group_keyword (groups_id, groups, keyword_search_id ) VALUES ($groups_id,
'$groups', $keyword_search_id)
      ");
      if ( !defined $sth ) {
      die "Cannot prepare statement: $DBI::errstr\n";
      }
}
```

Appendix B (Database dump)

```
--
-- PostgreSQL database dump
--

SET client_encoding = 'SQL_ASCII';
SET check_function_bodies = false;
SET client_min_messages = warning;

--
-- Name: SCHEMA public; Type: COMMENT; Schema: -; Owner: postgres
--

COMMENT ON SCHEMA public IS 'Standard public schema';


SET search_path = public, pg_catalog;

SET default_tablespace = '';

SET default_with_oids = true;

--
-- Name: arm; Type: TABLE; Schema: public; Owner: vignir; Tablespace:
--

CREATE TABLE arm (
    arm_id integer NOT NULL,
    name_id integer NOT NULL,
    arm_date integer,
    arm_kingdom character varying(1),
    arm_desc character varying(500)
);


ALTER TABLE public.arm OWNER TO vignir;

--
-- Name: arm_search; Type: TABLE; Schema: public; Owner: vignir; Tablespace:
--

CREATE TABLE arm_search (
    arm_id integer NOT NULL,
    keyword_search_id integer NOT NULL
);


ALTER TABLE public.arm_search OWNER TO vignir;

--
-- Name: keyword_search; Type: TABLE; Schema: public; Owner: vignir; Tablespace:
--

CREATE TABLE keyword_search (
    keyword_search_id integer NOT NULL,
    search_string character varying(80) NOT NULL,
    references_to integer
);


ALTER TABLE public.keyword_search OWNER TO vignir;
```

```
--
-- Name: names; Type: TABLE; Schema: public; Owner: vignir; Tablespace:
--

CREATE TABLE "names" (
    name_id integer NOT NULL,
    "names" character varying(200) NOT NULL,
    name_date integer NOT NULL,
    name_kingdom character varying(1) NOT NULL
);


ALTER TABLE public."names" OWNER TO vignir;

--
-- Name: deviceview; Type: VIEW; Schema: public; Owner: vignir
--

CREATE VIEW deviceview AS
    SELECT keyword_search.search_string, "names"."names", "names".name_date,
"names".name_kingdom, arm.arm_date, arm.arm_desc, arm.arm_id,
keyword_search.keyword_search_id FROM "names", arm, keyword_search WHERE ((("names".name_id
= arm.name_id) AND (arm.arm_id = arm_search.arm_id)) AND (arm_search.keyword_search_id =
keyword_search.keyword_search_id));


ALTER TABLE public.deviceview OWNER TO vignir;

--
-- Name: group_keyword; Type: TABLE; Schema: public; Owner: vignir; Tablespace:
--

CREATE TABLE group_keyword (
    groups_id integer NOT NULL,
    groups character varying(80) NOT NULL,
    keyword_search_id integer NOT NULL
);


ALTER TABLE public.group_keyword OWNER TO vignir;

--
-- Name: arm_pkey; Type: CONSTRAINT; Schema: public; Owner: vignir; Tablespace:
--

ALTER TABLE ONLY arm
    ADD CONSTRAINT arm_pkey PRIMARY KEY (arm_id);


ALTER INDEX public.arm_pkey OWNER TO vignir;

--
-- Name: group_keyword_pkey; Type: CONSTRAINT; Schema: public; Owner: vignir; Tablespace:
--

ALTER TABLE ONLY group_keyword
    ADD CONSTRAINT group_keyword_pkey PRIMARY KEY (groups_id);


ALTER INDEX public.group_keyword_pkey OWNER TO vignir;

--
```

```
-- Name: keyword_search_pkey; Type: CONSTRAINT; Schema: public; Owner: vignir; Tablespace:
--

ALTER TABLE ONLY keyword_search
    ADD CONSTRAINT keyword_search_pkey PRIMARY KEY (keyword_search_id);


ALTER INDEX public.keyword_search_pkey OWNER TO vignir;

--
-- Name: names_pkey; Type: CONSTRAINT; Schema: public; Owner: vignir; Tablespace:
--

ALTER TABLE ONLY "names"
    ADD CONSTRAINT names_pkey PRIMARY KEY (name_id);


ALTER INDEX public.names_pkey OWNER TO vignir;

--
-- Name: arm_name_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: vignir
--

ALTER TABLE ONLY arm
    ADD CONSTRAINT arm_name_id_fkey FOREIGN KEY (name_id) REFERENCES "names"(name_id);


--
-- Name: public; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;


--
-- PostgreSQL database dump complete
--
```

Appendix C (Format of the SCA Armorial Database)

Format of the SCA Armorial Database

by Iulstan Sigewealding
9 December 1998

This document describes the format used to record names and armory in the SCA Armorial Database. The database is actually a flat text file. The text is mostly ASCII, with a few Latin-1 encodings. (Latin-1 is an extension of ASCII and is an international standard. Unfortunately, its 8-bit codes are NOT compatible with the "437" code page normally active on PCs running MS-DOS in the United States.)

The file consists of a copyright notice followed by records.

NOTICE

The copyright notice begins with a line that reads "NOTICE:" and ends with a line that reads "END OF NOTICE." The records begin immediately after the end-of-notice.

RECORDS

Each record occupies a single line in the file, and each line contains one record. A record has at least five fields, separated by stiles (`|'). The five required fields are: name, source, type, text, and notes, in that order.

- (1) a name associated with the item
  - If the type is one of `B' `D' `N' `a' `b' `d' `g' `j' `s' `BD' `BN' or `D?', this is the registrant's name.
  - If the type is `AN', this is the alternate name.
  - If the type is `HN', this is the household name.
  - If the type is `O', this is the name of the award or order.
  - If the type is `t', this is the heraldic title.
  - If the type is one of `ANC' `BNC' `BNc' `NC' `Nc' `HNC' or `OC', this is the earlier name.
  - If the type is one of `Bv' `Bvc' `v' or `vc', this is the incorrect spelling of the name.
  - If the type is `C', this is a blank.
  - If the type is `R', this indicates where the cross-reference appears in the Armorial.

  There is no official limit on the length of a name.

- (2) reference to the source of the information:
  - (a) empty, for items whose source is unknown
  - (b) a four-digit LoAR date (in the form YYMM) optionally followed by a kingdom ID letter

    Examples:

    - \`7010' (= the October 1970 LoAR or errata letter)
    - \`9001E' (= the January 1990 LoAR or errata letter under "EAST")
    - \`5912W' (= the December 2059 LoAR or errata letter under "WEST")

    In this case, the LoAR date indicates the date of the Laurel action that created the record. For combined records (record types \`B' \`BD' and \`D') this is not necessarily the date the name was registered, nor is it necessarily the date the armory was registered.
  - (c) a dash, followed by an LoAR date, optionally followed by a kingdom ID letter

    In this case, the item has been released, transfered, redesignated, corrected, reblazoned, or is otherwise obsolete. The date is the date of the obsoleting action.
  - (d) a pair of LoAR dates, each optionally followed by a kingdom ID, the pair separated by a dash

    In this case, the item is obsolete. The second date is the obsoleting action, and the first date is the last action before the obsoleting action.

  The kingdom ID letters correspond closely to codes used by the S.C.A. Registry:

  - A = Atenveldt
  - C = Caid
  - D = Drachenwald
  - E = East
  - G = Gleann Abhann
  - H = AEthelmearc
  - K = Calontir
  - L = Laurel or SCA (not submitted through a specific kingdom)
  - M = Middle
  - m = Ealdormere
  - n = Northshield
  - N = An Tir
  - O = Outlands
  - Q = Atlantia
  - R = Artemisia

    - S = Meridies
    - T = Trimaris
    - W = West
    - w = Lochac
    - X = Ansteorra
- (3) record type (required):
    - a = Augmentation of arms
    - AN = Alternate Name
    - ANC = Alternate Name Change
    - b = Badge (also used for misc. armory -- see "notes" field)
    - B = personal name and Badge (combined record)
    - BN = Branch Name
    - BD = Branch name and Device (combined record)
    - BNC = Branch Name Change
    - BNc = Branch Name Correction
    - Bv = Branch name Variant without correction (probable error)
    - Bvc = Branch name Variant with Correction
    - C = Comment (database identification and version control data)
    - d = Device
    - D = personal name and Device (combined record)
    - D? = uncertain type of armory (could be a, b, d, or s)
    - g = reGalia
    - HN = Household Name
    - HNC = Household Name Change
    - j = Joint badge cross-reference
    - N = primary personal Name
    - NC = personal Name Change
    - Nc = personal Name Correction
    - O = award name or Order name
    - OC = award name change or Order name Change
    - R = cross-Reference
    - s = Seal
    - t = heraldic Title
    - v = personal name Variant without correction (probable error)
    - vc = personal name Variant with Correction
- (4) text:
    - If the type is one of \`B' \`D' \`a' \`b' \`d' \`g' \`s' \`BD' or \`D?', this is the blazon.
    - If the type is \`BN' or \`N', this must be empty.
    - If the type is \`j', this is the name of other registrant of the joint badge. (The other registrant's record will have the complete blazon.)
    - If the type is \`AN', this is \`For' followed by the primary name of the registrant.
    - If the type is \`HN', this is the name of the registrant, or a list of

    registrants, enclosed in double-quotes and separated by `and'.
    - If the type is `O', this is the name of the branch that registered the award or order.
    - If the type is `t', this is the name of the branch which registered the title.
    - If the type is `BNC' or `NC', this is the word `See' followed by the new (branch or personal) name.
    - If the type is `ANC' `HNC' or 'OC' this is the new (alternate, household, or award/order) name.
    - If the type is `BNc', `Bv', `Bvc', `Nc', `v' or 'vc', this is the corrected form of the (branch or personal) name.
    - If the type is `C', this is the text of the comment.
    - If the type is `R', this is the word `See' followed by the name under which the referenced item appears, or a list of names enclosed in double-quotes and separated by `or'.

  There is no official limit on the length of the text field.

- (5) notes, individually enclosed in parentheses:
    - To clarify the blazon: `A Cross of Samildanach is four Menorahs in cross'
    - To indicate the status the item: `Closed' `Deceased' `Disbanded' `-released' `-corrected blazon'
    - To indicate a special class of armory: `Civil Ensign' `Ensign' `Flag' `Naval Ensign' `War Banner'
    - To indicate the intended use of the item: `For House of Duckford' `For the populace'
    - To indicate the other holder of a joint badge: `JB: Elinor Aurora of Rosewood'

  There is no official limit on the number of notes or their length.

- (6) other fields:

  For name items, there are always exactly four stiles (and thus five fields).

  For armory items (types `B' `D' `a' `b' `d' `g' `s' `BD' and `D?') additional fields are used to describe the headings under which the armory is indexed in the SCA Ordinary. Obsolete armory is not indexed, and there is no official limit on the number of headings.