|  |
| --- |
| Woodling-Metzger Database |
| CS 325 Database Wells College April 24, 2009 |
| Kellie Stiadle |

Table of Contents

[1 Requirements Analysis 1](#_Toc228533147)

[1.1 Enterprise Overview 1](#_Toc228533148)

[1.2 Problem Summary 1](#_Toc228533149)

[1.3 Users 1](#_Toc228533150)

[1.4 Data 2](#_Toc228533151)

[1.4.1 Family Name 2](#_Toc228533152)

[1.4.2 Current Member 2](#_Toc228533153)

[1.4.3 Ancestor 2](#_Toc228533154)

[1.4.4 Job 2](#_Toc228533155)

[1.4.5 Address, phone numbers, e-mail addresses 2](#_Toc228533156)

[1.4.6 Military service 2](#_Toc228533157)

[1.4.7 Parent/Child Role 2](#_Toc228533158)

[1.4.8 Married 3](#_Toc228533159)

[1.4.9 Stories 3](#_Toc228533160)

[1.4.10 Photos 3](#_Toc228533161)

[1.5 Implementation Environment 3](#_Toc228533162)

[2 Database Design 4](#_Toc228533163)

[2.1 Semantic Model 4](#_Toc228533164)

[2.2 Relational Model 6](#_Toc228533165)

[3 Implementation 8](#_Toc228533166)

[3.1 Table Relationships 8](#_Toc228533167)

[3.2 Integrity Constraints 8](#_Toc228533168)

[3.3 SQL Script 8](#_Toc228533169)

[3.4 Queries and Views 9](#_Toc228533170)

[4 Conclusions and Recommendations 10](#_Toc228533171)

[5 Appendix 11](#_Toc228533172)

[5.1 MySQL Script for Creation of the Woodling-Metzger Database 11](#_Toc228533173)

[5.2 Additional Queries 14](#_Toc228533174)

[6 User Manual 15](#_Toc228533175)

# 1 Requirements Analysis

The descendents of the Woodling-Metzger family tree may be scattered across the country, but they keep in touch with each other via e-mail, cell phones, reunions, cards, and stories. Because the first generation descendents are now in their seventies and eighties and the subsequent generations do not live near each other, the history of this large and interesting family is in danger of being eradicated as the first generation descendants’ memories are being erased by Alzheimers and their stories are lost through death. It is important to retain this information and to make sure that the next generations have access to their ancestry.

## 1.1 Enterprise Overview

The Woodling-Metzger tree is comprised of nine of the original thirteen first generation descendents, 32 of 35 second generation descendents, 51 third generation descendents, and an unknown number of fourth generation children. While most of the original nine live near each other in central Pennsylvania, the rest of the family lives all over the United States. Even so, it is a close group; only a few members have lost contact with the rest. Certain members who still live in the original area collect different pieces of information. One aunt knows everyone’s birthday dates; another has an extensive collection of photographs. This uncle organizes family reunions, while that one writes down stories about his childhood and mails them out on holidays. The most organized of the nine has recently developed several health issues both physical and mental and has passed on her roles as the clearing house for any family news and the keeper of all of the family books to her children.

## 1.2 Problem Summary

The Woodling-Metzger history is stored in a mish-mash of out of date family tree books, photo albums, computer files, saved e-mails, VCR tapes of get-togethers, preserved documents, deteriorating gravestones, and first generation minds. A photo of an ancestor is even hanging in a carousel museum in an amusement park. Because the Woodling-Metzger descendents are no longer concentrated in central Pennsylvania, access to these various papers is denied to many who are truly interested in this history.

In 1997, there was an attempt to collate and update the family trees, but that book is now over 11 years old and has mistakes. In addition, it is extremely difficult to use family tree books. They are much like a singularly linked list: information at either end is easy to obtain, but if the information needed is in the middle, it is tedious and difficult to acquire. Trying to analyze any of the data is frustrating.

## 1.3 Users

There are three types of users for this database: historians, current update managers, and general users.

The historian is able to add information about ancestors of the Woodling-Metzger trees to the database. He can add, delete, or change information about past generations up to and including the current first generation. This information will be based on first generation accounts, written documents, research, and photographs.

The update manager has access to information of the present generations of the Woodling-Metzgers. He will be able to update or insert information concerning births, deaths, weddings, accomplishments, addresses, phone numbers, e-mails, etc.

The general user will have complete access to the information in the database, but will not be able to change anything in the database. He will be able to make queries, look up information about another relative, or find his direct ancestry line. If he has an update or change, he must contact the current update manager.

## 1.4 Data

###  1.4.1 Family Name

 Overall Family information: surname, county of origin, date arrived in America

###  1.4.2 Current Member

 Each person in the family has a first, middle, and last name, an ID number to separate those who have same names, a birth date, town and state or country where born.

###  1.4.3 Ancestor

 Each ancestor has the same information as the current members plus a death date, cause, cemetery, and location of burial.

###  1.4.4 Job

 Different jobs a person may have had

###  1.4.5 Address, phone numbers, e-mail addresses

 Places where the person was born, lived, and died, as well as addresses, phone numbers, and e-mail addresses if living

###  1.4.6 Military service

 When served, what branch, what rank

###  1.4.7 Parent/Child Role

 Who is related to whom, how

###  1.4.8 Married

 Who is married to whom, start, end, why ended

###  1.4.9 Stories

 Interesting stories or facts about the person

###  1.4.10 Photos

 Pictures of that person

## 1.5 Implementation Environment

Equipment needed to access this database is a PC. Pages will be read-only for general users and accessed using MySQL Query Browser.

# 2 Database Design

## 2.1 Semantic Model

**Family Tree Database First Part (not including ISA relations):**

****

**Figure 1-ER diagram part 1**

**Family Tree Database Second Part, ISA relations:**



**Figure 2-ER diagram part 2**

Since this diagram is complicated, it may help to describe the entity sets, relationships, and attributes shown in a little more detail.

**Entity Sets:**

*Family\_name* includes the primary key attribute surname, plus the “place of origin” and the “date that the first known ancestor arrived in America.” This entity has a strong relationship to the main entity set in this model, *Person.*

*Person* has numerous attributes to define each person including “name,” and its primary key ID number, which is extremely important when dealing with many people having similar or exactly the same names. *Person* is then split into the more specified entities *Ancestor* and *Current* which are further defined later. *Person* has two separate roles “marry” and “parent” to explain the relationships between certain members of the family.

In addition *Person* has several relationships including a strong one with the entity *Military*, whose attributes are” rank,” “dates\_served,” and the combined primary key attributes branch and name of war. *Person* is related to four entity sets: *jobs, stories, photos, and town\_county.* *Jobs* is a weak entity set because numerous people can have the same job, such as farmer, and multiple generations have worked in the same factory. Since information for one or the other attribute is often missing, this makes even the combined attributes of “job title” and “where worked” a discriminating key at best. *Stories* and *photos* are strong entities related to *person*. They both have a primary key labeled as ID. *Stories* also includes the attributes “what\_about,” and “when\_occurred.” *Photos*’ attributes are “date” and “location.” *Person* is connected to the entity *town\_county* to establish the birth place of that individual.

The specialized entities *ancestor* and current have different attributes and relationships. *Ancestor* refers to people who have died so it includes the attributes cause of “death” and “date died” and has a two different yet strong relationships with the entity *town\_county.* One of these relationships refers to the places the person has lived including dates. The other relationship is to specify where the person is buried.

*Current* refers to people who are still alive. It is connected to two weak entities *Phone Numbers* and *Email*. These two entities are really multi-valued attributes to *Current*, so they cannot stand on their own. Owing to the fact that most people have multiple phone numbers and email addresses, it makes sense to show them as weak entities in the ER model in order to break them down and show more information. *Current* is also related to *town\_county* through the relationship *address*. With the attribute “street” included, this will provide the current address for the person.

*State\_country* is a strong entity related to *town\_county*. Since so many of the Metzgers and Woodlings live in Pennsylvania, having this entity as its own table will save space in the database.

## 2.2 Relational Model

The tables for the Woodling-Metzger family database are as follows:

*family\_name*(surname, origin, date\_arrived)

*belong*(surname, IDnumber)

*person*(IDnumber, f\_name, mid\_name, l\_name, b\_date)

*marry*(IDnumber, IDnumber, start\_date, end\_date, reason\_ended)

*parent*(IDnumber, IDnumber)

*military*(name of war, branch, dates­\_served, rank)

*served*(IDnumber, name of war, branch)

*jobs*(IDnumber*, job\_title, where\_worked*)

*stories*(storyID, what\_about, when\_occurred)

*has*(IDnumber, storyID)

*photos*(photoID, date, location)

*is\_in*(IDnumber, photoID)

*ancestor*(IDnumber, date­\_died, cause)

*town\_county*(townID, name)

*state\_country*(placeID, name)

*born*(townID, IDnumber)

*buried*(townID, IDnumber, cemetery)

*lived*(townID, IDnumber, start\_date, end\_date)

*current*(IDnumber)

*address*(IDnumber, townID, street)

*phone\_numbers*(IDnumber, *home*, *cell*, work\_num)

*email*(IDnumber, *home, w\_email*)

# 3 Implementation

## 3.1 Table Relationships

The Woodling-Metzger database is comprised of twenty-two tables. Since this is a genealogy database, the most important entities are *person* and the subentities *current* and *ancestor*. The other entities exist to provide more information about the particular person.

## 3.2 Integrity Constraints

The tables for the Woodling-Metzger database were created in MySQL using an SQL script. ID numbers are integers. The rest of the attributes are string attributes. There are no NULL values since information not known will be given as “UNKNOWN.” In this way, weak entity sets can still function even with little information for some people. Primary keys cannot, of course, be NULL and must be unique attributes. For many of the tables, primary keys include foreign keys from other tables. IDnumber from the *person* entity is used often, which makes sense since all of the other entities revolve around *person.*

The use of some form of ID number as a primary key for many of the entities is a useful way to make sure that the integrity of the key remains intact even if little information has been collected for a certain individual. For each of these ID numbers, a constraint is placed to ensure that it is greater than zero. When these ID numbers are used in role relationships, an additional constraint of nonequality is placed. For example, a childID cannot equal a parentID or a person would be his own parent!

## 3.3 SQL Script

A script was written and executed to create the tables in the Woodling-Metzger database.

CREATE TABLE person

 (IDnumber INTEGER(30),

 f\_name VARCHAR(30),

 mid\_name VARCHAR(30),

 l\_name VARCHAR(40),

 b\_date VARCHAR(20),

 PRIMARY KEY(IDnumber)

 );

An example of the *person* table script shows five attributes in the table with “IDnumber” listed as the primary key. An integrity constraint mandating that this number has to be greater than zero strengthens the primary key. Also, “l\_name” can not be “Unknown” or “Null.” Or will be shown once I figure out how to do it.

The following screen shot of the SQL Query Browser shows a Schemata box on the right with the *person* table open. Listed below it are the five attributes, with a key next to “IDnumber” to indicate that it is the primary key. In the main screen, part of the script inserting the actual data into *person* is shown. Integer values are shown in purple, while character or string values show up in orange. Notice that the birthdates are entered as character values in order to include slash marks or to write “Unknown” as a value.





This is how the data appears once the information is saved in the database. Across the top of the “Resultset 2” box are the attributes with the corresponding data in the columns below. This table was produced with a simple query asking for all data in the *person* table to be listed. Idnumbers start at 01001, with each generation starting at a new thousands digit. For example the second generation starts at 02001.

## 3.4 Queries and Views

There are many queries that could be built for the casual user of this database. One example would be a view of known jobs for various people in the database. This query is created by combining data from the *person* table and the *jobs* table and then selecting the first names, last names, and job titles. Other queries could select only farmers or people who worked in Milton, PA. If a person wanted to investigate only jobs that ancestors held, he could ask for ID numbers that are between 01000 and 07000.



More queries coming…I would like to show a *marry* query and a *parent* query, but need to talk to you first about primary keys, foreign keys, and constraints. I also am not sure what queries an administrator would have that would be really any different from the general user, so I thought I would try to focus on some of the more interesting queries (like the roles) instead.

# 4 Conclusions and Recommendations

Now that the genealogy database exists, other family trees can, in time, be added to provide even more historical data. This database is set up in such a manner that whole new family lines can be added without too much change to the current tables. In addition, whenever new information is discovered about a person in the database, it should be relatively simple to add, delete, or update the correct tables.

It is extremely important, however, that the update manager maintains accurate and current information for living family members so that the data being stored does not quickly become out of date. The historian should remain on top of his part of the database as well, so that as data comes in or changes, he can keep the history of this family as precise as possible.

# 5 Appendix

## 5.1 MySQL Script for Creation of the Woodling-Metzger Database

CREATE DATABASE IF NOT EXISTS woodling\_metzger;

use woodling\_metzger;

CREATE TABLE family\_name

 (surname VARCHAR(40),

 origin VARCHAR(40),

 date\_arrived VARCHAR(20),

 PRIMARY KEY(surname)

 );

CREATE TABLE person

 (IDnumber INTEGER(30),

 f\_name VARCHAR(30),

 mid\_name VARCHAR(30),

 l\_name VARCHAR(40),

 b\_date VARCHAR(20),

 PRIMARY KEY(IDnumber)

 );

CREATE TABLE belong

 (surname VARCHAR(40),

 IDnumber INTEGER(30),

 PRIMARY KEY(surname, IDnumber)

 );

CREATE TABLE marry

 (husbandID INTEGER(30),

 wifeID INTEGER(30),

 start\_date VARCHAR(20),

 end\_date VARCHAR(20),

 reason\_ended VARCHAR(50),

 PRIMARY KEY(husbandID, wifeID)

 );

CREATE TABLE parent

 (childID INTEGER(30),

 parentID INTEGER(30),

 PRIMARY KEY(childID, parentID)

);

CREATE TABLE military

 (name\_of\_war VARCHAR(50),

 dates\_served VARCHAR(50),

 branch VARCHAR(40),

 rank VARCHAR(40),

 PRIMARY KEY(name\_of\_war, branch)

 );

CREATE TABLE served

 (IDnumber INTEGER(30),

 name\_of\_war VARCHAR(50),

 branch VARCHAR(40),

 PRIMARY KEY(IDnumber, name\_of\_war, branch)

 );

CREATE TABLE jobs

 (IDnumber INTEGER(30),

 job\_title VARCHAR(50),

 where\_worked VARCHAR(50),

 PRIMARY KEY(IDnumber, job\_title, where\_worked)

 );

CREATE TABLE stories

 (storyID INTEGER(30),

 what\_about VARCHAR(60),

 when\_occurred VARCHAR(40),

 PRIMARY KEY(storyID)

 );

CREATE TABLE has

 (IDnumber INTEGER(30),

 storyID INTEGER(30),

 PRIMARY KEY(IDnumber, storyID)

 );

CREATE TABLE photos

 (photoID INTEGER(30),

 photo\_date VARCHAR(20),

 location VARCHAR(50),

 PRIMARY KEY(photoID)

 );

CREATE TABLE is\_in

 (IDnumber INTEGER(30),

 photoID INTEGER(30),

 PRIMARY KEY(IDnumber, photoID)

 );

CREATE TABLE ancestor

 (IDnumber INTEGER(30),

 date\_died VARCHAR(20),

 cause VARCHAR(60),

 PRIMARY KEY(IDnumber)

 );

CREATE TABLE town\_county

 (townID INTEGER(30),

 name VARCHAR(40),

 PRIMARY KEY(townID)

 );

CREATE TABLE state\_country

 (placeID INTEGER(30),

 name VARCHAR(45),

 PRIMARY KEY(placeID)

 );

CREATE TABLE lived

 (IDnumber INTEGER(30),

 townID INTEGER(30),

 start\_date VARCHAR(20),

 end\_date VARCHAR(20),

 PRIMARY KEY(IDnumber, townID)

 );

CREATE TABLE buried

 (IDnumber INTEGER(30),

 townID INTEGER(30),

 cemetery VARCHAR(50),

 PRIMARY KEY(IDnumber, townID)

 );

CREATE TABLE current

 (IDnumber INTEGER(30),

 PRIMARY KEY(IDnumber)

 );

CREATE TABLE address

 (IDnumber INTEGER(30),

 street VARCHAR(40),

 townID INTEGER(30),

 PRIMARY KEY(IDnumber, townID)

 );

CREATE TABLE phone

 (IDnumber INTEGER(30),

 home VARCHAR(20),

 cell VARCHAR(20),

 work\_num VARCHAR(20),

 PRIMARY KEY(IDnumber, home, cell)

 );

CREATE TABLE email

 (IDnumber INTEGER(30),

 home VARCHAR(45),

 w\_email VARCHAR(45),

 PRIMARY KEY(IDnumber, home, w\_email)

 );

## 5.2 Additional Queries

# 6 User Manual