Design av en webbaserad person- och släktskapsdatabas

- Utveckling av en fri kommunikationskanal för släktforskning på Internet.

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

Examensarbete:
Olof Wingren

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds Universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds Universitet
Lund 2006

Sammanfattning

Design av en webbaserad person- och släktskapsdatabas
- utveckling av en fri kommunikationskanal för släktforskning på Internet.

Denna examensrapport redogör för en utvecklingsprocess av en webbasserad släktforskningsapplikation, där resultat från släktforskning kan delas av flera aktiva släktforskare. Genom att grundligt undersöka olika datamodeller och befintliga applikationer försöker arbetet säkerställa grunden till en funktionell webbapplikation. Arbetet ledde till en betaversion med begränsad funktionalitet som skall vidareutvecklas i framtiden till ett GNU/GPL projekt med öppen källkod för gratis användning och nerladdning. Målet med arbetet uppfylldes i stora drag, det frångicks dock lite på grund av tidsbrist som uppstod i samband med prototypprogrammeringen.

Abstract

Designing a database person and relations database for the webb.
- development of a free communication channel for genealogy research on the Internet.

This essay describes a developmentprocess for a webb-based genealogy research application, where results från genealogy research can be shared by other active researchers. By thoroughly research different datamodells and existing applikations this essay tries to secure the foundation for a functional application. The result of this project will lead to a beta version that in the near future will be developed into a GNU/GPL project with open source for free use and download. The goals with this essay was to a large deal fulfilled, but with some minor glitches caused by lack of time after dealing with problems derived from prototype programming.

Keywords: Genealogy research, Multiuser webbapplikation, GPL/GNU.

Förord

De som idag bedriver släktforskning bedriver det ofta i det dolda, det verkar svårt att få tag på bra och tydligt släktforskningsmaterial, i alla fall för de oinvigda nybörjare som inte har energi nog att sätta sig in i varje liten aspekt. De som har gått långt i sin kunskap sitter oftast med högar av dokument i någon isolerad lokal och söker understödja sina teorier för att klottra ner ett släktträd. Sådan forskning tenderar att försvinna med personen som uträttat den.

Jag hoppas att med detta arbete bidra till att mer god släktforskning blir tillgängligt för allmänheten. Jag hoppas även att fler vågar prova på att släktforska under konsultering av mer erfarna släktforskare.

Tack till min handledare *Mats Rüder* för konstruktiv kritik trots vanligtvis sent inskickade rapporter innan handledningstillfällen, och tack till min farbror *Mikael Wingren* för det väldokumenterade släktforskningsmaterialet som jag har haft mycket nytta av under arbetets gång.

Innehållsförteckning:

1. Inledning

Släktforskning, eller ”*Genealogi*”, går ut på att forska i människors släktförhållanden. Tillsammans med *Biografin* brukar *Genealogi* räknas samman under begreppet *personhistoria* enligt Wikipedia [7]. Man syftar alltså till att dokumentera en persons härstamning eller avkomlingar eller både och. Man kompletterar också oftast med händelser och detaljer om personerna man söker, såsom bosättning, viktiga årtal och andra intressanta detaljer. Allt sådant kan också leda till att nya spår upptäcks, som t ex ifall någon emigrerad släktgren. Släktforskning omfattar egentligen bara blodsband och inte moderna relationer såsom adoption eller ingifta syskon.

Motivet till detta examensarbete har uppstått då jag under en tid har administrerat en webbsida där mina släktingar har försökt samordna sin omfattande släktforskning. På webbsidan är trädet dokumenterat med hjälp av Microsoft Excel, vilket har lett till tekniska problem. Vissa har helt enkelt inte programmet Excel och andra använder Macintosh och de är svåra att ge bra support till som PC-användare. Dessutom tar ett så pass omfattande släktträd - samlat i en enda fil - en hel del utrymme och flertalet aktiva släktforskare orkade inte göra sig besväret att uppdatera informationen på webbsidan någon längre tid. Webbsidan har alltså plågats av att den inte är designad för ändamålet – släktforskning – utan är egentligen ett forum med dess funktioner. De tekniska problemen tillsammans med ett krångligt gränssnitt har idag lett till att sidans utveckling stagnerat.

För att få en nystart i släktforskningen, och få medlemmar att bli mer aktiva, behövs en bättre webbapplikation där arbetet byggs upp mer runt släktträdet och där detta är överskådligt och lättillgängligt så att alla kan ta del av – och ha åsikter om – de olika detaljerna i släktforskningen.

Idag finns redan många hjälpmedel för släktforskare på marknaden. Dessa tar oftast betalt för mer avancerade – och mer användbara – funktioner och har i många fall ingen möjlighet till samarbete mellan användare utan man står beroende av utvecklarna som mellanhand. Detta leder å ena sidan till en kvalitetssäkring, å andra sidan till censur och kostnader för mellanhändernas arbete. Jag vill istället skapa en kostnadsfri resurs som stimulerar släktforskare att samarbeta och dokumentera sina resultat på ett sådant sätt att fler kan ta del av det.

1.1 Kravspecifikation

Detta är ingen kravspecifikation i ordets egentliga bemärkelse. Det är egentligen en sammanfattning av premisser snarare än en i arbetet framtagen

kravspecifikation. När jag nämner kravspecifikation i arbetet är det dock dessa premisser jag åsyftar.

Det jag vill åstadkomma är alltså en lösning som består av en *databas* och en *webbapplikation* med möjligheten att visa databasen i form av *ett släktträd* som följer de för genealogi redan etablerade konventionerna[1]. Databasen ska i sin tur vara utformad för att kunna lagra släktforskningsdata, tillhörande flera släktträd, men anpassad för att användas med en webbapplikation. Webbapplikationen ska vara *modulbaserad*, så att den går att vidareutveckla och konfigurera efter behov. Den ska ha stöd för *teman* och för *flera språk*. Applikationen ska också ha stöd för *flera användare*, så att de ska kunna hjälpas åt att släktforska. Applikationen ska också vara kompatibel med *Open Source* och bör därför använda sig av PHP och MySQL.

1.2 Problemformulering

Målet med examensarbetet är att utforma och implementera en webbapplikation för släktforskning där flera användare kan hjälpa varandra. Applikationen ska utgöras av öppen källkod och bestå av dels en *person- och släktskapsdatabas* och dels av ett *webbaserat släktträd*. Vad arbetet syftar till är alltså att:

- Utveckla en webbapplikation och databas för släktforskning med öppen källkod för fri distribution.

Vilket sönderfaller i två delproblem:

- Designa och implementera en *person- och släktskapsdatabas*, där släktskapsrelationer[2] mellan individer kan härledas utifrån logiska samband från inmatad data.
- Utveckla och implementera en prototyp av ett *webbaserat släktträd*, härlett ur databasen, där släktskap tydligt representeras.

1.3 Framgångskriterier

Webbapplikationen kan anses uppnå sin funktion ifall:

- ”Avancerade” släktskap kan lagras i databasen.
- Släktträdet i webbapplikationen följer praxis[3] i sin utformning.
- Databasen innehåller minimalt med redundans och null-värden.

---

[1] Det vill säga grafiskt korrekt representerat, såtillvida att personer i trädet länkas samman av linjer som representerar släktskap och där en riktning indikerar vem som är barn och vem som är förälder, vem har fått barn med vem och vilka är då syskon, kusiner etc.

[2] Med släktskapsrelationer menas blodsband mellan två valfria individer i släktträdet. Förutom kusin/kusinbarn/far/mor även ”farfars brors barnbarn”/”morfars farfars kusins dotter” och dylika ”avancerade” släktskap.

[3] Se fotnot 1.

- Användaren kan välja avgränsad visning av olika delar genom sökning av släktträdet.

1.4 Syfte

Syftet är att projektet vidareutvecklas till en webbapplikation där arbetet för flera användare centreras runt uppbyggnaden av ett släktträd. Systemet hjälper användaren att fastslå och hålla reda på relationer. Det hjälper också användare att utväxla och distribuera släktdata på ett effektivt och entydigt sätt. Lösningen bygger på öppen källkod så att projektet förhoppningsvis leder till distribution under GPL/GNU-licens [8] för gratis användning av hobbyforskare över hela världen. Den slutliga produkten skall vara utformad med stöd för flera användare och med diskussionsgrupper, kommentarer etc.

1.5 Avgränsning

Jag gör ingen utredning i mitt arbete av vilka program som är bäst lämpade för mitt syfte. Jag har valt att använda MySQL och PHP för databas respektive programmering av webbapplikation för dessa verktyg är defactostandard på ”den fria webben” och de uppfyller alla krav jag har på mina verktyg.

2. Metod

Övergripande kan sägas att metoden består av fyra steg. (1) Först samlar jag in material genom att undersöka liknande lösningar. (2) Med materialet som grund utformar jag databasen. (3) Därefter undersöker jag olika tekniker för att rita upp ett släktträd i webbapplikationen och beaktandes denna, så skapar jag en logisk struktur för applikationen (filstruktur, kodstruktur, m.m.). (4) Jag avslutar projektet med att testa vald metod genom att mata in ett befintligt släktträd och konstruera kod för visning av släktträd.

Mer detaljerat så inleder jag med att undersöka nödvändiga funktioner i databasen genom att undersöka befintliga släktforskningsresurser och analysera program och format som finns tillgängliga på Internet. Detta för att ge större styrka till och komplettera kravspecifikationen (punkt 1.1) som beskriver de premisser som gäller för webbapplikationen. Utifrån resultatet av undersökningen sammanställer jag en lista med viktiga funktioner som måste beaktas. Sedan utreder jag hur man på bästa sätt ska lagra släktskapsrelationer i databasen och även hur databasen ska designas för att vid vidareutveckling implementera bilder knutna till relationer, händelser och personer. Varefter design av databasen påbörjas i form av ett entitets-/relations diagram. ER-diagrammet färdigställs sedan genom en effektivisering, som innebär att tabeller delas upp och trimmas för bättre prestanda. Slutligen implementeras databasen i MySQL.

När design av databasen är färdig inleds efterforskning om olika möjligheter till att grafiskt framställa släktträdet, då detta måste vara bestämt innan design av själva applikationsgrunden kan inledas. När en strategi om släktträdets grafiska struktur och teknik bestämts inleder jag en grundläggande utformning av webbapplikationen med övergripande delar och filsystem, vilken jag implementerar på server för vidare utveckling. Sedan utvecklas ett preliminärt gränssnitt för inskrivning av nya individer till databasen och för att skapa nya släktträd i databasen.

Därefter testar jag databas och applikation genom att försöka implementera tekniken bakom det grafiska släktträdet, genom att följa metoden jag i tidigare steg har tagit fram, eventuella problem tas upp. Efter detta avser jag att testa och ytterligare förbättra applikation och databas i mån av tid.

2.1 Insamling och analys av bakgrundsmaterial

Genom insamling och analys av bakgrundsmaterial avser jag bygga en bredare kunskapsgrund för vidare design av databas och webbapplikation. Detta steg

är viktigt då jag själv på detta stadium i arbetet inte vet mycket om de program för släktforskning som finns tillgängliga.

Jag gör detta genom att först göra en *översiktlig kartläggning* över vilka resurser som finns tillgängliga för släktforskare. Resultatet av kartläggningen delar jag in i olika *kategorier* av släktforskningsresurser. För varje kategori gör jag sedan ett *urval* där ett av programmen/resurserna, som passar in i kategorin, väljs ut och sedan *analyseras* i en sammanställning.

Analysen gör jag genom att kondensera de grundläggande funktioner/lagringsmöjligheter ett program av det här slaget bör tillhandahålla. Med detta kondensat (punkt 2.1.2) avser jag sedan komplettera de generaliserade krav jag sedan innan examensarbetet sammanställt i kravspecifikationen. I slutändan syftar alltså analysen till att ge en *konkret lista* (punkt 2.1.3) över viktig funktionalitet för databas och webbapplikation.

Ovanstående metod är egentligen en iterativ process då analysen av resurserna kan leda till att nya kategorier skapas eller att programmet byter kategori eller att kartläggningen måste revideras. Den ger ändå tillräcklig insikt i förutsättningarna för examensarbetet.

2.1.1 Informationsinsamling

Genom att samla in information om andra hjälpmedel för släktforskare - tillgängliga på nätet - försöker jag att komplettera den kravspecifikation jag sammanställt i inledningen. Steget omfattar *översiktlig kartläggning*, *kategorisering* och *urval* av resurser som beskrivet i styckets inledning. Detta steg kommer ej att redovisas för i själva genomförandet då resultatet framkommer under nästa analyserande metodsteg. De Insamlingen kommer att utföras som följer.

För att få ett brett underlag av släktforskningsresurser i insamlingen undersöker jag några populära släktforskningsresurser. Jag försöker bedöma källans trovärdighet genom att undersöka hur många som använder resursen (och finner den relevant) och genom att undersöka Googles pageranking[4] (som påvisar t ex om det är många andra sidor som länkar till resursen). Jag gör sedan ett urval av program som representerar olika kategorier av släktforskningsresurser – kategorier såsom klientbaserade släktforksningsprogram, föreningar/”communities” för släktforskning, databaser, med mera. De utvalda programmen analyseras sedan utförligt i nästa metodiksteg.

[4] Går att få reda på genom Google toolbar för IE eller Firefox. En högre siffra tyder på fler besökare och fler sidor som i sin tur har bra pageranking som länkar dit. Med pagerank kan man alltså få reda på om en sida är välansedd eller inte.

2.1.2 Analys av befintliga släktforskningsresurser
I detta steg jämför jag de resurser jag har hittat och deras funktioner med min kravspecifikation för att undersöka ifall jag har utelämnat någon relevant funktion. De resurser jag anser mest relevanta i föregående steg undersöks och analyseras efter användningsområden och hur pass relevanta de är att beakta vid design av databas och webbapplikation i aktuellt projekt. En sammanfattning över dessa resurser dokumenteras i rapporten.

2.1.3 Listning av funktioner för databasen
Genom att grundligt undersöka de resurser som jag har valt ut, så avser jag att sammanställa en lista över viktig funktionalitet för databasen. Detta är också ett sätt att sålla bort onödiga funktioner. Funktionerna måste också ta hänsyn till det nya användningsområdet och kan därför inte vara exakta gentemot ursprunget. Jag efterstävar däremot att de ska vara kompatibla med släktforskning i stort. Med kompatibelt menar jag att databasen efter vidareutveckling ska kunna hantera importering och exportering av data till andra dataformat utan att viktig information går till spillo.

2.2 Design av databas
För att få en databas som går att implementera i flera olika databashanterare (i detta projekt kommer MySQL användas), och för att kommunicera databasens funktion på ett tydligt sätt så har jag valt att designa den med entitet/relationsdiagram. Ett ER-diagram gör databasen lättöverskådlig och lätt att implementera även på andra databashanterare än MySQL. Det gör även databasen lättare att förbättra vid senare tillfälle och är ett bra hjälpmedel att jobba med vid första utformningen av databasen. En annan fördel är att ER-diagram även är ett illustrationssätt som många har erfarenhet av och förstår, och behöver därför inte förklaras i lika stor utsträckning som eventuella andra illustrationssätt.

2.2.1 Utredning om släktskaplagring
Det i särklass största problemet med en person- och släktskapsdatabas, är att hitta ett bra sätt att implementera och i databasen lagra de olika typerna av släktskap så att en sökning i databasen går snabbt. Samtidigt ska inte databasens snabbhet gå ut över funktionalitet för användarna. Genom att undersöka för- och nackdelar med olika metoder för släktlagring såsom användning av släktkod, försöker jag komma fram till ett bra och för webbapplikationen praktiskt sätt att göra detta.

2.2.2 Behandling av personuppgifter
För att inte bryta mot personuppgiftslagen (PUL), undersöker jag riktlinjerna för PUL. Att detta steg inte utförs innan översiktlig design av ER-diagram är

för att PUL kontrollerar innehållet och alltså mest kan påverka de tilltänkta attributen på entiteterna, entiteterna förses inte med fullständiga attribut förr än i nästa metodsteg. Behandling av personuppgifter resulterar i att jag diskuterar om vad som är tillåtet eller otillåtet att lagra i databasen av relevanta typer av information för släktforskningsapplikationen.

2.2.3 Bildlagring och koppling till databasen

I vidareutveckling av applikationen, efter tiden avsatt för examensarbetet, ämnar jag ha med möjligheten att lagra bilder, kopplade till olika delar av databasen. För att inte behöva återgå till designstadiet av databasen i vidareutveckling av webbapplikationen så försäkrar detta metodsteg att databasen är kompatibel med en bildhanteringsfunktion.

Bilderna lagras inte direkt i databasen då detta tar mycket plats. Ofta har webbhotell en ganska låg begränsning på databasens storlek men inte på webbutrymmet. Istället uppladdas bilderna genom formulär till hemsidans avsatta webbutrymme. För att sedan koppla bilderna till poster i databasen på ett effektivt och användbart sätt bör databasen ha ett genomtänkt stöd för att koppla fysiskt lagrade bilder på webbservern till rätt person eller händelse.

Jag undersöker var i databasen det är lämpligt att införa lagring av sökvägar till uppladdade bilder på webbutrymmet och med vilken metod detta bör göras, alltså var de ska lagras på servern med mera.

2.2.4 Översiktlig design av ER-diagram

Jag producerar en grov design av databasen och med grundläggande funktionalitet från min lista över viktiga funktioner från bakgrundsmaterialet (steg 2.1.3). Utmärkande attribut definieras på entiteter, men inte slutgiltigt då attribut kan tillkomma under effektiviseringsfasen och/eller i ”Slutgiltig specificering av tabeller” (steg 2.2.6). Att designen av ER-diagrammet på detta sätt delas upp möjliggör att varje steg för sig valideras innan nästa steg påbörjas.

2.2.5 Effektivisering av ER-diagram

Databasens upplägg beaktande relationer och nycklar utgår från strävan efter så lite null-värden och redundans som möjligt. Därför delar jag upp entiteter i svaga och starka för att databasen ska bli kompakt och effektiv. Nycklar definieras och kompletta listor av attribut för varje entitet definieras. Relationerna namnges efter dess funktion. ER-diagrammet vidareutvecklas.

2.2.6 Slutlig specificering av tabeller

Som ett sista steg i design av databasen så inför jag alla attribut på alla tabeller i databasen tillsammans med ytterligare förklaring över tabellernas funktioner,

efter specifikationer från steg 2.1.3 där viktiga funktioner listas, med eventuella tillägg för grundläggande funktionalitet.

2.2.7 Implementering av databasen

Databasen implementeras i MySQL. Jag väljer MySQL eftersom det är ett gratis verktyg under GPL-licens och går därför i enlighet med denna webbapplikations syfte. Implementationen görs med hjälp av webbapplikationen PhpMyAdmin, som också är ett Open Source program, precis som MySQL. Programmet ger en bra översikt och smidigt skapande av tabeller och relationer. Implementering genomförs för att det ska vara möjligt att börja utveckla webbapplikationen. Implementeringen innebär också en kvalitetssäkring av DB-designen – då databasen motsätter sig en grovt inkorrekt design genom felmeddelanden.

2.3 Design av webbapplikation

För att kunna utveckla en funktionell webbapplikation gör jag en undersökning där jag orienterar mig i vilken metodik jag kan använda mig av vid skapandet av trädet, genom att undersöka ett släktträds komponenter och försöka härleda ett system jag kan använda mig av vid ställningstagande om teknisk bas. Detta gör jag genom att använda en pappersdesign där jag försöker se samband i ett redan kvalitetssäkrat släktträd och undersöker vilka problem som kan uppstå. Varefter jag försöker komma fram till en teknisk lösning där de problemen är lättast överkomna och där resultatet kommer så nära som möjligt ett grafiskt släktträds funktioner. Detta resonemang mynnar ut i ett konkret val av vilken teknik som ska användas för det grafiska släktträdet.

Efter att ha fastställt den tekniska lösningen bakom det grafiska släktträdet designar jag den grundläggande designen för webbapplikationen i form av filstruktur och kodstruktur. Denna design implementerar jag sedan på servern för att på så sätt kunna ta reda på ifall den logiska strukturen fungerar praktiskt. När detta är implementerat så designar jag ett preliminärt gränssnitt för informationsinmatning till person- och släktskapsdatabasen för att sedan kunna testa databasens funktionalitet genom att lägga in ett avancerat släktträd.

2.3.1 Efterforskning av teknik för grafiskt släktträd

Jag skapar en övergripande pappersdesign av ett släktträd och försöker komma på samband som går att utnyttja vid skapande av släktträdet men också för att försöka förekomma de problem som måste lösas. Jag använder Internet för att undersöka vilka tekniker som går att använda för att skapa det dynamiska grafiska släktträdet. Detta genom att läsa om html och undersöka hur olika

sidor byggs upp, jag kommer att kritiskt diskutera dessa lösningar och vad som krävs för att kunna skapa släktträdet. Jag kommer också att göra mindre praktiska test för att bättre kunna bedöma vissa detaljer i tekniken. Utifrån processen väljer jag vilken teknik som passar sig bäst för ändamålet. Resultatet blir ett motiverat val av den teknik som ska användas.

2.3.2 Skapa logisk struktur för webbapplikation

Utifrån vald teknik utformar jag applikationens grundläggande logiska struktur. Med struktur menas filstruktur på servern, kodstruktur och relaterade beslut om applikationen funktionalitet. Dessa olika strukturer sammanfaller i en enda övergripande metod för hur applikationen ska utformas. Besluten tar jag genom teoretisk utredning av för och nackdelar med olika möjliga strukturer på servern. Utredningen avslutas med att jag implementerar strukturen på servern för att testa ifall resultatet av den teoretiska utredningen går att implementera i praktiken. Implementationen testar strukturvalet då inte en browser används i kodningen och testmoduler/testteman, osv också implementeras. Strukturen är felaktig t.ex. ifall inte en modul laddas in eller ifall temat inte appliceras på exempeltext.

2.3.3 Preliminärt gränssnitt för informationsinmatning

Jag designar och implementerar ett gränssnitt i webbapplikationen för inmatning av släktforskningsdata till databasen. Med preliminärt menar jag att inte all funktionalitet i systemet, såsom dubbletthantering och informationsvalidering är implementerad, utan bara själva formuläret för att mata in redan kvalitetskontrollerad släktdata för vidare utveckling och testning. I utformningen tar jag däremot hänsyn till användbarhet genom att följa en logisk inmatningssekvens där grundläggande uppgifter såsom namn först krävs och mer detaljerad information *skiktas*[5] och är valfri att fylla i.

2.4 Testning

När gränssnittet för inmatning är färdigt så matar jag in kvalitetskontrollerad släktdata i databasen för att testa databasens kapabilitet. Sedan implementerar jag en testdesign för det grafiska släktträdet för att noggrannare undersöka ifall metoden fungerar i praktiken.

2.4.1 Inmatning av släktträd för vidare utveckling

För att testa inmatningen och för att ha information att använda vid testdesign av grafiskt släktträd så matar jag in ett släktträd med hjälp av applikationens gränssnitt. För att anses lyckat bör ett femtiotal med varandra besläktade

[5] Med ”skiktas” menar jag att dessa inmatningsfält inte syns direkt utan är valbara t ex genom att dölja dessa fält tills användaren trycker på en knapp (eller byter flik). En skiktning innebär att ett gränssnitt inte blir för mycket på en gång utan användaren förstår lättare sammanhanget i vilket fälten står.

individer vara inmatade. Vissa individer med mer detaljerad information införd, för att bättre testa funktionalitet i släktträdet. Jag kommer att använda mig av kvalitetssäkrad släktforskning av Mikael Wingren[6], den innehåller både avancerade släktskap såsom personer med flera partners och ojämna generationer, såtillvida att vissa ”grenar” innehåller fler generationer än andra.

2.4.2 Testdesign av grafiskt släktträd

För att testa den valda tekniken (se steg 2.3.1) om teknik för det grafiska släktträdet vidareutvecklar jag den. Koden för släktträdet implementeras på servern. Om modellen är funktionell så leder detta steg till att hela det inmatade släktträdet på minst 50 individer kan visas på ett korrekt sätt. Alltså med intakta släktband och avancerade släktskap som t ex flergifte, när en person har fått barn med fler än en person.

[6] Mikael är min farbror och har släktforskat aktivt sedan en längre tid. Han har sammanställt ett släktträd med över 500 individer som sträcker sig tillbaka sju generationer.

3. Genomförande

3.1 Analys av befintliga släktforskningsresurser

Första steget mot förverkligande av webbapplikationen är att göra en djupdykning i redan befintliga resurser för släktforskning. Genom att undersöka det aktuella utbudet så ämnar jag få en bättre orientering i vad databasen bör innehålla för information och ifall det finns några funktioner jag måste ha i åtanke vid design av databasen.

Jag redovisar här ett urval av en stor mängd resurser som alla mer eller mindre fyller samma funktion. Urvalet är baserat på kategorisering, språk och ranking. Detta beskrivs utförligare i metodsteget ”Insamling och analys av bakgrundsmaterial” (metodiksteg 2.1). Kategoriseringen har jag gjort efter att ha undersökt ett större antal olika resurser. Jag valde då en resurs per kategori då det skulle vara överflödigt att visa flera som erbjöd i stort sett samma funktioner. Beträffande språk så finns det troligen applikationer på andra språk, kanske med andra funktioner, men jag är begränsad till svenska och engelska. Valet av vilken resurs som får utgöra varje kategori har jag gjort på popularitet, länkar och goda omdömen. Resurserna jag har valt att beakta och deras genres är dessa.

1. *Genline* är en *databas med släktforskarmaterial*. En mängd webbsidor erbjuder liknande tjänster – där man kan granska olika sorts dokument för att utöka sin släktforskning.
2. ”*Föreningen släktdata*” har dels en databas med redan utförd släktforskning - har man tur har någon annan redan utfört den forskning man själv tänkte genomgöra – dels är föreningen ett *”community”* där man kan utbyta erfarenheter men också söka hjälp med sin forskning.
3. ”*Swedish Genealogy Page*” är bara ett *koncept* på en webbapplikation för släktforskning, och liknar min idé men verkar inte ha kommit längre än till konceptstadiet.
4. ”*Min släkt*” är ett *klientbaserat släktforskningsprogram*, som i mycket påminner om min idé i funktionalitet. Med den stora skillnaden att den är helt klientbaserat och därmed har begränsade möjligheter till interaktion med andra släktforskare.
5. *GEDCOM*, ett *standardformat* för släktforskningsdata (ett bland många, men enligt utsago det mest utbredda), dock designat för lagring av släktforskningsdata i en fil och inte för en arbetande databas för flera användare.

3.1.1 Genline – databas med släktforskarmaterial

Ett av de kommersiella hjälpmedlen till släktforskning är *Genline* [1]. Genline erbjuder ett datorprogram man kan ladda hem för att sedan vid utförligare

användning betala en abonnering på. Programmet ger däremot inte användare någon möjlighet till att organisera sin egen släktforskning, det tillhandahåller bara möjligheten att söka igenom förteckningar såsom kyrkböcker. Det mesta är fotografier, så fritextsökning finns inte. En användare måste i stort sett veta namn och år och sedan bläddra sig fram för att förhoppningsvis träffa på vad man söker efter. Det finns många liknande sidor som tar betalt för att tillhandahålla väldigt specifika tjänster som kan vara släktforskare till gagn. En sådan tjänst ligger långt ifrån vad detta arbete eftersträvar, Genline är en resurs för informationsinsamling och inte ett hjälpmedel för dokumentation av resultat.

3.1.2 Föreningen släktdata – databas och ”community”

Förrutom Genline, som funktionellt ligger långt ifrån mitt examensarbete, hittas *Föreningen Släktdata* [2] som har en omfattande databas men också den intressanta funktionen att besökare kan efterlysa vissa, ur släktforskningssynpunkt, intressanta personer. De publicerar också resultat från lyckade släktforskningsprojekt runt om i Sverige och bygger på så vis vidare på sin databas. Liknande funktionalitet ligger närmre det som min webbapplikation kommer att tillhandahålla när den är färdigställd, då min webbapplikation syftar till att släktforskare ska kunna ta del av varandras forskning.

3.1.3 Konceptet ”Swedish Genealogy Page”

Det närmaste jag har kommit min egna idé om en släktforsknings webbapplikation är Peter Sjölunds ”Swedish Genealogy Page”:

”Idén är att möjliggöra kontakt mellan släktforskare med samma släktforskarmål. Man kan alltså lägga in egna släktnamnsuppgifter, men också söka bland i databasen av andra släktforskare inlagda uppgifter, och hoppas på att få napp.” [3]

Tyvärr tillhandahåller *senITel* [3], som är en förening för pensionerade Telia tjänstemän och har en gedigen guide om släktforskning, bara en bruten länk till projektet och jag har därför inte kunnat ta reda på mer om denna sida. Det är möjligt att den blivit uppköpt av något av de kommersiella företagen eller bytt adress men jag får inte upp några relevanta spår efter sökning på *Google* [4].

3.1.4 Min släkt – ett klientbaserat program

MinSläkt [5] är ett svenskt program för släktforskning som tillhandahåller vissa funktioner som även min webbapplikation bör erbjuda. Funktioner som programmet tillhandahåller är bl.a. ett grafiskt släktträd, beskrivning/sökning på relationer, exportering till olika konventionella format till andra

släktforskningsprogram, utskrifter av olika slags rapporter, historiska händelser och bilder kopplade till personer i registret, logisk rimlighetskontroll på inmatade värden. Programmet erbjuder också en funktion för konfidentiella noteringar, sådana som inte vem som helst ska ha tillgång till, vid t ex exportering till HTML.

MinSläkt står i funktionalitet nära min webbapplikation med den skillnaden att min webbapplikation ska vara till för användning på Internet. Men informationen som lagras och till viss del också upplägget och den grafiska utformningen är värd att beakta vid design av både databas och webbapplikation.

3.1.5 Standardformatet GEDCOM

Ett av de format som MinSläkt kan exportera till är GEDCOM [6] som är ett standardformat för exporterad släktforskning mellan olika program. I standarden finns en datastruktursmodell och extensiv dokumentation om formatet. Ytterligare efterforskningar visar att formatet är en utbredd standard och bör därför beaktas i vidare utformning av webbapplikationens struktur.

Formatet är gjort för lagring av släktdata i en eller flera filer på hårddisk och är inte gjort för att användas direkt i en databas och inte heller till en databas med koppling till webbapplikation. Därför kan jag inte använda modellen rakt av men kan finna inspiration och smarta lösningar av upplägget. En bra funktion i GEDCOM är att de delar på individ och familj. Relationerna definieras alltså genom tillhörigheten i en familj, familjen innehåller far, mor, och barn med referenser till dessa som personer. Varje person kan ha flera familjer och på detta sätt kan t ex en person vara barn i en familj och far i en familj.

3.1.6 Generella krav på funktionalitet

Med utgång i programmet MinSläkt [5], standardformatet för exportering av släktforskningsdata GEDCOM [6] och av egen erfarenhet och kunskaper om släktforskning, kan jag sluta mig till dessa uppenbara generella krav på funktionalitet hos databasen, och på den färdiga webbapplikationen, kring vilka det rådet konsensus:

- Personer knyts till personer genom parrelation, släktband eller händelser (de två första är synliga i släktträd).
- Händelser är knutna till personer.
- Visning av anfäder eller ättlingar som möjlig separat vy av släktträdet. (Till exempel att visa ett träd med anfäder till individ, eller alla ättlingar till en annan.). Denna punkt ställer krav på släktträdets dynamiska design.

- Källhänvisning vid införd information.
- Bilder knutna till personer och händelser.

Följande funktioner är sekundära och krävare vidare utveckling av applikationen, och jag avser alltså inte att implementera detta inom tiden avsatt för examensarbetet:

- Exportering till GEDCOM-fil.
- Fleranvändningsstöd.
- Diskussionsforum kopplade till händelser och personer.
- Allmänt diskussionsforum.
- Utskrift av rapporter.
- Blanketter man kan skriva ut och fylla i vid informationsinsamling utan tillgång till dator.

3.1.7 Sammanfattning befintliga släktforskningsresurser

Programmet *Genline* tillhandahåller en annan typ av tjänst än den min applikation har som målsättning. Genline är en databas med detaljinformation, inte sammanställd fakta styrkt av flera källor och satt i sitt sammanhang. *Föreningen släktdata* tillhandahåller däremot flera tjänster däribland delar de med sig av redan gjort släktforskning i form av sökbar databas, de är även ett sorts ”community” för släktforskare där användare kan efterlysa viss information och hoppas på svar. Föreningen släktdata är däremot en betaltjänst, vilket inte går ihop med mina aspirationer.

Konceptet ”*Swedish Genealogy Page*”, som jag tyvärr verkar ha runnit ut i sanden, har en strävan som liknar den som motiverat mitt arbete det vill säga att få släktforskare med samma mål att kunna samarbeta över en Internetplattform. Programmet *MinSläkt* är till stor del vad min applikation ska tillhandahålla sina användare, med den skillnaden att MinSläkt är klientbaserat och med begränsade möjligheter att utföra gemensam släktforskning, det går dock att exportera till HTML och till GEDCOM-format – vilket är ett steg i min riktning. Ett klientbaserat program har däremot sina begränsningar när det gäller distribution med förbehåll för t ex olika operativsystem och tillgänglighet från olika datorer.

Modellen för släktforskningslagring – GEDCOM – är inte gjord för användning i en webbapplikation med databaskoppling, men kan fungera som inspiration gällande vad som ska lagras och kanske på vilket sätt entiteter kan samverka i databasen.

3.2 Design av databas

För att lättare kommunicera tanken bakom ett ER-diagram och dess funktionalitet börjar jag med att förklara ett ER-diagrams olika byggklossar: svaga-/starka entiteter, relationer, nycklar och attribut, och förklara deras sammanhang och betydelse. Mitt designarbete börjar som följer.

Jag lägger upp en genomtänkt strategi för att lagra släktskapen i databasen. Det visar sig att bästa sättet att göra detta är att använda sig av en egen tabell för lagring av individers släktskap till varandra. Vidare tar jag upp applikationens möjligheter till att binda bilder till olika delar av informationen. Efter resonemang slutar jag mig till att lagra sökvägar till bilder i en databastabell, med en referens till vart i informationen bilden hör hemma, är det bästa sättet eftersom samma bild kan höra till flera händelser/relationer.

Efter att jag har kommit fram till släktskapslagring och bildhantering designas databasen översiktligt med generell funktionalitet. I designmomentet tas beslut om att separera personer från deras relationer, för att kunna skapa flera släktträd med samma personer, och att knyta händelser till personer och inte till själva släktträdet också detta för att ”återvinna data”. När den generella databasdesignen är färdig effektiviserar jag tabellerna Relation och Person. Dessa blir till flera svaga och starka entiteter och blir mer effektiva gällande redundans och null-värden. Efter effektiviseringen diskuterar jag Personuppgiftslagen (PUL) och hur denna påverkar val av *attribut* i databasen. I nästa steg där jag färdigställer attributen i ER-diagrammet tar jag hänsyn till önskad funktionalitet och till PUL och gör en lista över vilka attribut vilka entiteter ska ha.

3.2.1 Terminologi för ER-diagram

Vid design av databas har jag valt att använda mig av ett entitet/relationsdiagram. Nedan följer en förklaring av hur ett sådant fungerar.

Ett ER-diagam består av *entiteter*, dvs enheter (ruta med information om tabellen), som binds samman av *relationer* som är streck. På varje relation finns en fyrkant, roterad 45 grader, som definierar relationens typ. Varje entitet innehåller attribut, ett attribut är en tabellkolumn i databasen. Det finns två typer av entiteter; starka entiteter innehåller egna *nycklar*, som är ett attribut som gör tabellraden unik, svaga entiteter måste låna sina nycklar som då kallas *främmande nycklar.*

Ex. En entitet ”Person” kan ha nyckeln ”personnummer” och attributen ”förnamn” och ”efternamn”, på det viset blir varje rad i tabellen person en identitet. De kan ha samma för- och efternamn men personnumret gör raden unik.

En stark entitet
Stor text är entitetns namn
Liten grå text är attribut
Understrukna attribut är nycklar

En svag entitet
Visas med dubbel kantlinje
Understrukna attribut är främmande nycklar.

Vad gäller relationer entiteter emellan, så finns det både *starka relationer* och *svaga relationer*. De starka relationerna binder samman starka entiteter, medan de svaga relationerna markerar att en svag entitet ”lånar” sin nyckel från en stark entitet. På detta sätt får man en mer kompakt databas och inte massor av rader som innehåller tomma fält eller fält med samma innehåll.

Ex. Tabellen ”Handlingslista” innehåller listans ”Namn” och inhandlingsställe. Den innehåller också varans namn. Varje rad, innehållandes varan, består då av ”Listnamn”, ”Inhandlingsställe” och ”Varunamn”. Behöver man fler produkter av samma vara blir det flera rader i databasen som ser precis likadana ut. Detta kallas för redundans. Om man istället delar man upp tabellen i två entiteter en stark ”Handlingslista” och en svag ”Listrad”. Handlingslista innehåller attributen ”Namn” och ”Inhandlingsställe”. Detta står bara en gång på inhandlingslista. I tabellen ”Listrad” upprepas bara namnet på inhandlingslistan (som blir en främmande nyckel) och varunamn och kvantitet av varan. På det sättet behövs bara en rad för flera varor av samma slag, och inhandlingsstället upprepas inte, vi har fått en effektiv databas.

Svag relation: Stark relation:

Pilar på relationer markerar beroendet. Är pilen riktad från t ex från en tabell ”person” till en tabell ”skolklass” betyder det att en ”skolklass” kan ha flera personer, men en person kan bara gå i en skolklass. Finns inga riktningspilar så kan många personer gå i många skolklasser. Är det pilar åt båda hållen på en relation så gäller sambandet ”en person till en skolklass och en skolklass till en person” d.v.s. bara en person kan vara i skolklassen, vilket inte stämmer rent logiskt för en riktig skolklass.

3.2.2 Släktskapslagring i databasen

I en person- och släktskapsdatabas ställs man inför det största problemet, att ha ett system för att lagra släktskap mellan individerna. Släktskapslagring är i allra högsta grad viktigt eftersom komplexa sökningar kommer att utföras för att få fram komplicerade släktträd. Sökningen ska gå snabbt men utan att påverka funktionalitet för användaren – såsom extraarbete vid inskrivning och matematiska funktioner användaren måste räkna ut själv.

Som jag ser det finns det tre möjliga tekniker som kan fungera för detta ändamål. Det första är en *släktkod* som beskriver släktskapet till en speciell individ genom att numrera barnföljden som 1, 2, 3 osv. Den andra metoden är att, som i GEDCOM, definiera *familjeobjekt*. Ett familjeobjekt består av en far, en mor och deras barn. Varje person i familjen lagras för sig i en egen tabell och binds därefter till flera familjer. Den familj personen står i som ”barn” är den familj individen har växt upp i och innehåller alltså personens föräldrar. Familjer där personen står som far eller mor är familjer där han har producerat barn i som vuxen. Det tredje sättet är att definiera relationer rakt av i en egen tabell och göra webbapplikationen smart på så sätt att den efter att ha angivit en relation kan räkna ut fler relationer och föreslå dessa. Den största fördelen med den tredje modellen är att långväga släktskap räcker för att föra in en person i ett släktträd. Utnyttjar man modell ett och två så går inte detta utan vidare utan ”dummies” i form av okända personer måste utgöra mellansteg då sådana släktskap är mer beroende av varandra.

3.2.2.1 Släktkod

Ursprungligen laborerade jag med tanken på någon form av ”släktkod”, en sifferkombination som representerar släktskapet mellan individer från den äldsta medlemmen i släktträdet till den yngsta där en etta representerar ”första barnet”, en två ”andra barnet” och så vidare. Det skulle i praktiken innebära att t ex jag med utgång från min farfar skulle bli ”32” - farfars tredje barns andra barn. Ska man sedan lägga till farfars far hade koden byggts ut till ”232”, eftersom min farfar har en äldre bror. Släktkoden utgår hela tiden ifrån den äldsta släktingen i släktträdet, så när en ännu äldre släkting läggs till så måste en extra siffra in före den befintliga släktkoden (t ex 245 blir då kanske 3245 ifall förälder till ”2” var sin fars tredje barn).

Fördelen med en ”släktkod” på det här sättet är att släktskap är väldigt enkla att räkna ut. Jämför man två personers släktkoder, t ex ”232” och ”221”, så ser man direkt att om det finns samma antal siffror så tillhör de samma generation. Första gemensamma siffran från höger sett innebär släktskap, i detta fall tre, generationer ”upp”. En skiljande siffra är syskon, två skiljande siffror betyder kusin, tre skiljande siffror hade varit tremänning osv.

Problem kan uppstå om två släktträd i webbapplikationen ska slås ihop. T ex om det ena släktträdet sträcker sig längre tillbaka i tiden. Problemet då blir att

den äldsta anfadern ligger ännu längre tillbaka i tiden. Antingen byter man då ut alla koder i det yngre släktträdet, eller låter man släktkoden vara lokalt orienterad i varje delsläktträd, på det sättet behöver inte programmet räkna om släktkoden och man sparar serverprestanda.

Jag övergav däremot mina planer på släktkod då koden skulle bli dynamisk vid påbyggnad av släktträdet och hela tiden ändra ”form”. Vid ett stort släktträd - med flera hundra individer, vilket inte är ovanligt vid omfattande släktforskning – så hade applikationen behövt gå igenom alla individer och lagt till information ifall en till generation ”uppåt” lades till vilket hade varit väldigt belastande för servern.

3.2.2.2 Familjeobjekt

I GEDCOM-standarden [6] för lagring av släktforskningsdata förklaras släktskapen på så sätt att varje person binds till flera familjeobjekt. Varje familjeobjekt innehåller en far, en mor och eventuellt ett eller flera barn. Varje individ i familjen lagras i separat tabell och tillhör fler familjeobjekt. För att ta reda på släktskapen söker man genom familjeobjekten, i den familj en person är barn så måste alltså far och mor i samma familj vara hans föräldrar, är personen förälder i ett familjeobjekt måste alltså barnen vara hans barn. På så sätt kan man följa en blodslinje från person till person till ett helt släktträd är genomgått.

Fördelen med GEDCOM är att allt dokumenteras på ett synbarligen mycket logiskt sätt då modellen har en struktur som i mycket påminner om hur vi som människor ser på familjer och släktkonstellationer. Såsom kärnfamiljen som definierar personen - parrelationen som ett eget objekt med möjligt resultat i avkomma. Detta är mycket riktigt kärnan i släktskap, med nackdelen i att släktskapsökningar kan bli väldigt komplicerade och bestå av många nästlade sökfrågor då en ”ren” GEDCOM-lösning skulle leda till en väldigt komplex databas där en mängd logiska sökningar måste utföras för att få rätsida på ett helt släktträd.

3.2.2.3 Lagrande av relationer i tabell

I programmet MinSläkt [5] så ombedes användaren att vid inlägg av person i släktträd välja typ av släktskap och till vilken person. Faller användaren in på rätt plats – genom en funktion som kollar om ålder och årtal verkar stämma överens (en person född 1914 kan logiskt sätt inte vara barn till någon född 1840) – så söks möjliga resterande släktskap automatiskt upp ifall de existerar.

Ex. En mor och en far finns redan i databasen, en bror till fadern finns också och även broderns barn. Vid inskrivning av personen anges bara modern, modern har redan en relation inskriven till fadern:

”make/sambo”, vilket programmet söker reda på och skriver in fadern som far och faderns brorsbarn som kusiner.

Detta förfarande ger också ny funktionalitet vid inmatning av personer med okänt släktskap. Att istället för att bara linjärt lägga till far, mor eller barn istället kunna lägga till en kusin och sedan kusinens tremänning. Detta kan vara till hjälp t ex om man vet att de har umgåtts och bott i samma stad men saknar uppgifter om anknytande släktingar.

3.2.3 Personuppgifter

Enligt PUL (Personuppgiftslagen) [9] så ska den personuppgiftsansvariga - i detta fall jag som utvecklare av programmet - se till min webbapplikation inte bryter mot lagen, som konkretiseras av följande punkter vilka jag tar hänsyn till i avseende om innehåll och behandling:

a) Personuppgifter alltid behandlas på ett korrekt sätt och i enlighet med god sed (Exakt vad som menas med god sed är otydligt i PUL).
b) Personuppgifter samlas in bara för särskilda, uttryckligt angivna och berättigade ändamål.
c) Personuppgifter inte behandlas för något ändamål som är oförenligt med det för vilket uppgifterna samlades in.
d) De personuppgifter som behandlas är adekvata och relevanta i förhållande till ändamålen med behandlingen.
e) Inte fler personuppgifter behandlas än som är nödvändigt med hänsyn till ändamålen med behandlingen.
f) De personuppgifter som behandlas är riktiga och, om det är nödvändigt, aktuella, alla rimliga åtgärder vidtas för att rätta, blockera eller utplåna sådana personuppgifter som är felaktiga eller ofullständiga med hänsyn till ändamålen med behandlingen.

Eftersom ändamålet för applikationen och dess innehåll är för släktforskning bör alltså endast historiskt relevanta uppgifter behandlas. Bland viktiga historiskt relevanta uppgifter finns födelsedatum, härkomst, ockupation, avkomma, osv. Medan personnummer, inkomst, telefonnummer och givetvis koder och namnteckningar är av diskuterbar art. Man kan väl generellt sett säga att de senare behöver ha mycket god motivering och personen i fråga bör vara avliden sedan länge. Är det känsliga uppgifter som ändå har stor relevans bör man främst få personens, eller närmast anhörigs, godkännande.

Enligt PUL [9] så är även fotografier en känslig personuppgift. I släktforskningssammanhang kan man hävda att fotografier är en stor tillgång, då t ex vissa släktskap kan gå på utseende (kan vara vanligt vid okänd

avkomma t ex utanför äktenskap) och då vara del av de indicier som krävs för en större diskussion.

Sammanfattningsvis är det viktigt att ha kvar möjligheten att ange källa för uppgifter och en möjlighet för användarna att i kommentar skriva in ifall de fått godkännande från berörda personer. En annan funktionalitet är möjligheten att märka vissa uppgifter som känsliga och dölja dessa för besökare som inte är registrerade medlemmar.

3.2.4 Bildhantering

För att koppla ihop bilder med information i en databas finns det två olika relevanta sätt. Förutom dessa två så går också att lagra bilder i själva databasen som hel fil, vilket hade varit det allra bästa men detta tar alltför stor plats för fotografier och oftast har webbhotell storleksgränser på databasen vilket gör en sådan bildhantering orealistisk, dessutom skulle det bli enormt stor belastning på databasen som vore tvungen att kasta ur sig åtskilliga tusentals sidor med tecken för en enda bild.

De två återstående realistiska sätten är att antingen att ändra filnamnen efter databasposten med vilken bildfilen ska höra ihop med - exempelvis skulle alla bilder som hör till ”Per” heta per001.jpg, per002.jpg (många varianter på detta skulle gå att framställa), det andra alternativet är att lagra filnamn och sökväg till bildfilen i en egen databastabell.

Den största fördelen med att döpa om filnamnet efter databasens innehåll är att ingen inmatning i databasen behövs. Nackdelen är att det kan ta lång tid att söka efter bilderna. Det kan också vara svårt att vid felsökning ta reda på vad bilden man söker heter, eftersom det troligen skulle vara en avart på personens ID-nummer i databasen. T ex om personen har fått ID ”2453” så måste en administratör gå in i databasen och leta reda på personens id-nummer innan rätt bild kan hittas på filservern. En fördel däremot är att bilder kan tas bort manuellt, direkt med en ftp-klient, utan att behöva gå in i databasen.

Att använda sig av databasen för att lagra bildens sökväg har sina för- och nackdelar. En fördel är att en databas är sökoptimerad och det går snabbt att få fram vilken bild som ska användas, det är också lätt att extrahera denna information och direkt sätta den i en sökväg i applikationen, endast en lätt kontroll ifall filen ligger kvar på filservern bör finnas med. Hade man använt sig av omdöpning hade en hel listning av alla filer på servern fått genomgås vilket är belastande för servern och ganska långsamt, sedan hade rätt bild fått matchas mot databasen för att få reda på om filnamnet överrensstämmer med personens ID. Ett sådant tillvägagångssätt är krävande för servern, men man hade givetvis kunnat ordna detta med någon form av ”cache”-lösning, så att

listningen av filer på servern sparas undan och ändras bara ifall antingen en fil saknas, eller ifall en ny fil tillförs.

Jag väljer att använda mig av lagring av sökväg i databas. Främst för att detta är ett mindre komplicerat sätt och det finns också möjlighet till att införa kommentarer till olika bilder och statistik på t ex nedladdning av bilden osv. Det är även på detta sätt möjligt att koppla en bild till flera personer och händelser, vilket inte hade varit möjligt med filnamnsmodellen.

Tabellen som lagrar bilder har jag valt att kalla *Pictures* och binds till person-ID eller person- och familjeträds-ID. Detta för att vissa bilder kan vara relaterade till familjeträdet, medan vissa bilder bör höra endast till personen som porträtt av personen i fråga. Ifall bilden tillhör person- och familjeträds-ID, så kan t ex bilder förknippas med personen i just det sammanhanget, som t ex gruppfoton eller personinspirerad konst.

3.2.5 Översiktlig design

I slutändan är den färdiga applikationen tänkt för flera användare och för flera släktträd. Varje användare ska kunna skriva in egna personer att sätta in i sitt släktträd, för att dessa ska gå att hantera som någon slags objekt som sedan andra användare ska kunna dra nytta av genom att t ex bygga på med ytterligare information och kopiera till sitt eget släktträd. Det är därför logiskt att lagra inskrivna personer i en egen tabell. Tabellen får värden som förnamn, efternamn, titel, yrke/sysselsättning och ett personligt löpnummer. Löpnumret används sedan för att koppla ihop olika tabeller till personen.

Ett problem med en särskild persontabell kan däremot uppstå ifall flera användare nyttjar samma person och någon av dessa går in och ändrar information om personen, som de andra användarna, vilka också vill utnyttja personobjektet, förlitade sig på i sitt avgörande. Detta anser jag är ett sekundärt problem och bör lösas genom gränssnittet. T ex genom att ursprungliga skaparen har äganderätt till personobjektet och andra måste be om tillåtelse för att kunna ändra på det.

Ett annat problem är eventuella dubbletter som kan uppstå när flera användare skriver in samma person. Det går att skydda sig mot dubbletter i databasen genom att ha personens namn som nyckel.

Ex. Per Persson matas in i databasen och får sitt namn som nyckel. En annan användare försöker skriva in ”Per Persson” igen och försöker spara men databasen har redan en Per Persson och skickar ett felmeddelande tillbaka till användaren. Användaren tittar då

själv igenom persondatabasen och hittar då den redan inskrivna Per Persson.

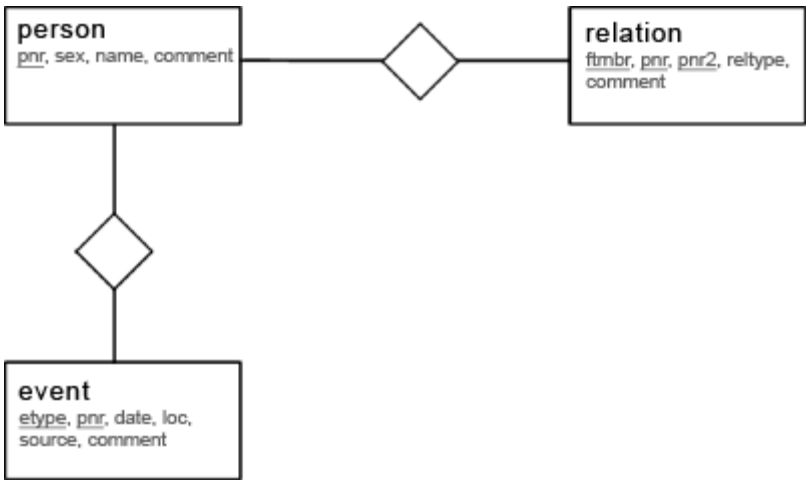
Nackdelen med detta förfarande, och det som gör att denna princip inte går att tillämpa i praktiken, är att det alltför ofta förekommer att namn går igen i släkter. En son till Per Persson är rätt trolig att heta t ex Per Persson junior eller dylikt. Ibland hoppar namn över en eller flera generationer, många döper sina barn till sina avlidna släktingars namn för att hedra och uppmärksamma dessa. Eftersom namndubbletter förekommer frekvent och naturligt i släkter är det alltså inte hållbart att tvinga databasen på detta sätt. En tänkbar lösning på detta skulle kunna vara att liksom konungafamiljer döpa namndubbletter till ”Per Persson I”, ” Per P II”, ”Per P III”, ”Per P IV”, osv. Men detta kan i sin tur bli krångligt när släktträdet byggs på med ytterligare äldre generationer av släktingar och i ett riktigt utförligt släktträd kan namngivningen bli väldigt invecklad.

Enklare men inte hundra procentigt är att minimera namndubbletter genom ett smart inmatningsformulär när användaren skriver in en person. Stämmer t ex förnamn, efternamn och ungefärligt födelsedatum så ser systemet till att varna användaren och föreslå den redan inskrivna personen, vet användaren med sig att det inte är samma person det handlar om så ignorerar han helt enkelt denna varning och fortsätter sin inmatning.

Förutom tabellen där varje rad är en person, så behövs en tabell där släktskapen, personerna emellan, dokumenteras. Det är denna tabell som - abstrakt sett - innehåller själva släktträdet i den mån varje rad i tabellen länkar samman en person med dess närmaste släktingar.

Ytterligare information som är till stor nytta och intresse vid dokumentation av släktforskning är händelser som är knutna till personer. I programmet MinSläkt [5] kan man skriva in flera händelser knutna till en eller flera personer. Händelserna klassificeras i olika kategorier, som födelse, dop, adoption, konfirmation, utbildning, examen, bosatt, emigration, immigration, pensionering, död, begravning, testamente och boupptecknin. Men det finns givetvis många andra klassifikationer man skulle kunna tänka sig. En fördel med klassifikationen – förutom dokumenteringen - är givetvis att man kan göra avancerade sökningar och lista t ex alla som ligger begravda på en speciell kyrkogård eller dylikt.

Ex.



Tabellerna ges engelska namn (se bild) då databasen ska vara förståelig även för personer med andra modersmål. Vissa databaser har inte heller tillräckligt stöd för specialtecken som å, ä och ö och är man inte försiktig med detta kan det innebära extra arbete efter implementering. Tabellen ”person” är den tabell som innehåller individer, tabellen ”relation” innehåller relationer mellan individer, och bildar därmed själva släktträdet. Tabellen ”event” innehåller händelser bundna till personer i persontabellen.

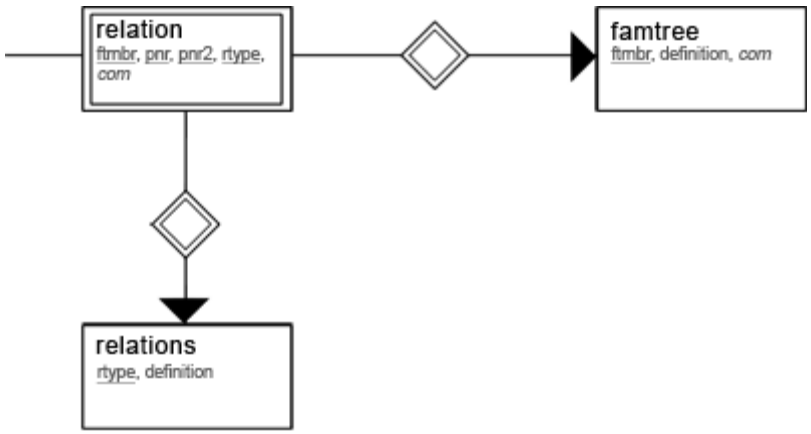
3.2.6 Effektivisering av tabellen Relationer

Tabellen innehållande relationer, sedermera definierad som den tabellen som utgör stommen till varje släktträd, blir till flera tabeller då null-värden och redundans ska undvikas i en effektiv databas.

Problematiken ligger i att tabellen behöver innehålla ett övergripande släktnamn – som mer konkret blir släktträdets ”namn” i slutliga applikationen – men även information om släktträdet, alltså en kommentar som förklarar för oinvigda vad just den släktforskningen har eftersträvat. Tabellen bör även innehålla personers släktskap och vad för relation de har gentemot varandra, eventuell kommentar över deras släktskap och eventuellt någon mer information som t ex källan som denna information är hämtad från, hur och när.

För att undvika redundans och mycket null-värden har jag beslutat att dela upp tabellen relation i tre tabeller. Två av entiteterna är starka och en är svag. De starka entiteterna kallar jag *famtree* och *relations*. Den svaga entiteten har jag valt att kalla *relation*.

Ex.

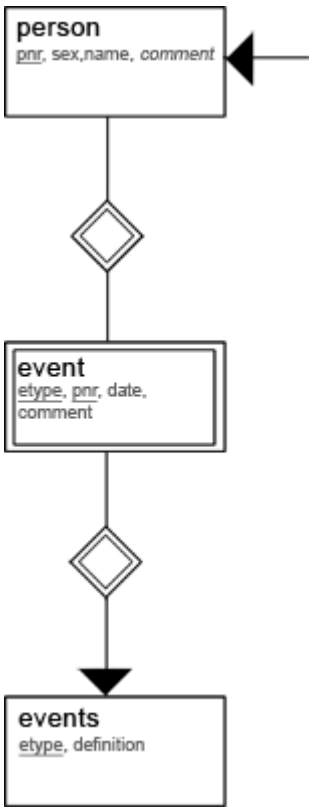


Famtree innehåller övergripande information om släktträdet, dvs släktträdets namn, definition och eventuell kommentar. *Relation* är beroende av dels huvudnyckeln från *famtree* och dels av ”personnumret” från tabellen *person* (se 4.1.5). Funktionen hos *relation* är att binda ihop en person med ett släktträd och då samtidigt relation till en annan person. Tabellen *relations* funktion är att beskriva vilken relation personen i *relation* har till en annan person och till det bifoga en kommentar till denna relation. Sista entiteten beskrivande relationer är *relations* som innehåller en lista på olika sorters relationer som individer i tabellen person kan ha till varandra.

Valet att lägga relationstyper i databasen och inte ”hårdkoda” dem i webbapplikationen är främst för flerspråksstöd men även då administratörer av den slutliga webbapplikationen möjligtvis vill lägga till eller ta bort vissa relationer av någon anledning.

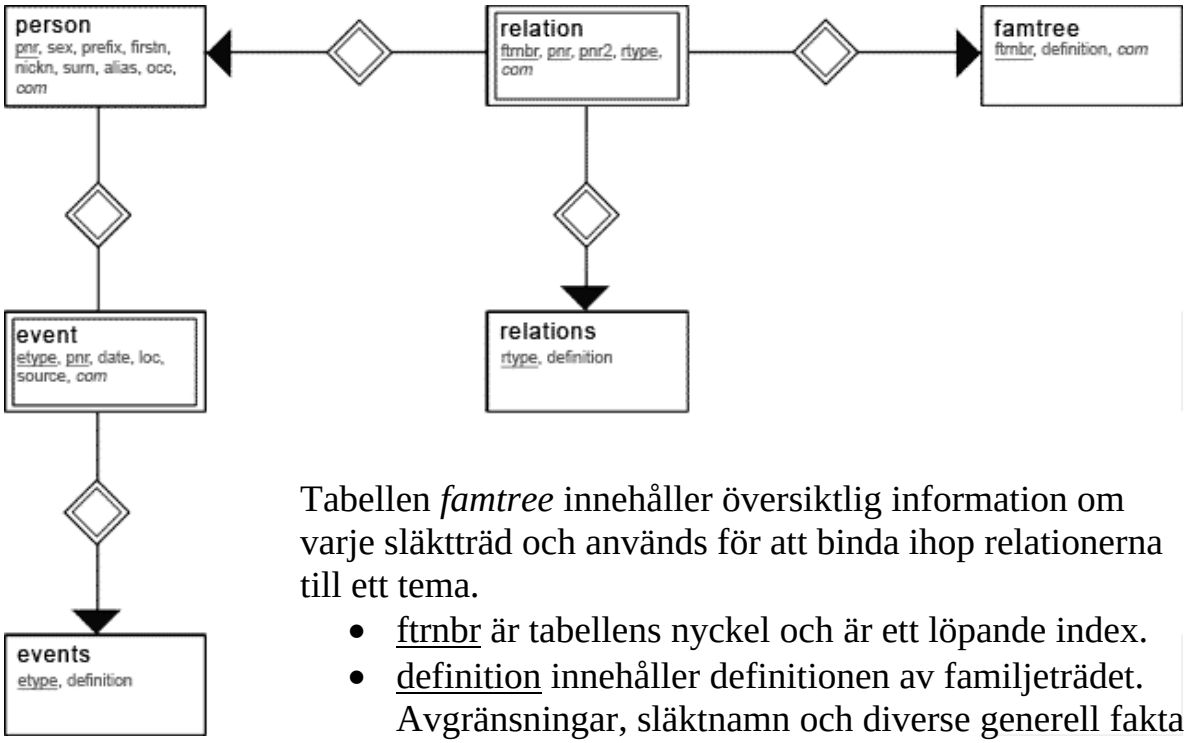
3.2.7 Effektivisering av tabellen Person

Persontabellen ska innehålla data om personer vilka sedan knyts samman genom olika släktträd. Tabellen *Person* är skiljd från själva släktträden och varje inskriven person är ett slags eget objekt som kan användas i flera släktträd. Till varje person ska flera olika händelser (Eng: ”*Events*”) kunna knytas (se stycke 3.2.1). För att undvika redundans i persontabellen införs en separat tabell *Events*, som innehåller de olika händelserna. Varje händelse kan knytas till flera personer, som tvillingfödsel, skolgång, m.fl. där flera individer kan vara inblandade.



Tabellen *Events* i sin tur delas upp i en stark entitet *Events* och en svag entitet *Event*, där *Events* innehåller de olika typer av händelser som finns (dop, vigsel, begravning, student, etcetera). *Event* i sin tur innehåller de specifika händelserna med referens till *Events*, datum för händelsen, plats, källa för uppgiften och ytterligare kommentarer. Varje *Event* binds till en eller flera personer med ett tabellfält där personernas databasspecifika personnummer radas upp. Med databasspecifika personnummer menar jag inte personernas egentliga personnummer, utan ett tilldelat identifikationsnummer som personen får vid inskrivning i databasen (se 3.2.5). Den svaga tabellen *Event* är beroende av nyckel från *Events* och främmande nyckel från *Person.*

3.2.8 Färdigställning av attribut och nycklar



Tabellen *famtree* innehåller översiktlig information om varje släktträd och används för att binda ihop relationerna till ett tema.

- ftrnbr är tabellens nyckel och är ett löpande index.
- definition innehåller definitionen av familjeträdet. Avgränsningar, släktnamn och diverse generell fakta – efter användarens tycke.
- com lämnar utrymme för övriga kommentarer gällande släktträdet.

Tabellen *person* är en förteckning över personer som är inskrivna i programmet, varje rad innehåller information om personen i fråga:

- pnr är tabellens nyckel och är ett löpande index, den tilldelade siffran blir personens identitet i övriga tabeller.
- sex är personens könstillhörighet och kan anta formerna: [”man”, ”kvinna”, ”ej angivet”].
- prefix är eventuellt tilltal av variant: [”dr”,”herr”,”fru”,…].
- firstn är personens namn, inklusive eventuella mellannamn.
- nickn är eventuellt smeknamn.
- surn är personens efternamn.
- alias är en referens till fler personnummer, ifall personen också är känd som någonting annat. Vilket är bra vid t ex samma person på olika orter eller som bytt namn vid flyttning.
- occ är personens huvudsakliga yrke, går också att ange flera yrken.
- com är för eventuell kommentar till personuppgifterna.

Tabellen *relation* definierar relationen en person har till en annan i ett specifikt släktträd. Tabellen är en svag entitet och är beroende av tre främmande nycklar.

- ftrnbr är främmande nyckel från *famtree*.
- pnr är främmande nyckel från *person*.
- pnr2 är en referens till den person den först angivna personen har en relation till.
- rtype är främmande nyckel från *relations* och pekar på den typ av relation pnr och pnr2 har till varandra.
- com är för eventuell kommentar till relationen.

Tabellen *relations* innehåller definitioner av olika släktskap som går att välja mellan när man skapar en relation i tabellen *relation*:

- rtype är nyckel och innehåller de olika relationerna [”mor”, ”far”, ”syskon”, ”kusin”, ”tremänning”].
- definition innehåller en definition av relationen angiven i rtype.

Tabellen *event* är en svag identitet som binder ihop de starka entiteterna *person* och *events*. Den har som uppgift att dokumentera olika händelser personer i *person* har varit med om:

- etype är främmande nyckel från *events* och innehåller typen av händelse.
- pnr är främmande nyckel från *person*.
- date är datumet då händelsen ägde rum.
- loc är den plats händelsen utspelade sig på.
- source definerar källan varifrån informationen är hämtad.
- com är för eventuell kommentar till händelsen.

Tabellen *events* innehåller de olika typer av händelser som kan ha inträffat och som kan användas i tabellen *event*:

- etype är nyckel och innehåller de olika händelserna: [”födelse”, ”dop”, ”adoption”, ”konfirmation”, ”utbildning”, ”examen”, ”bosatt”, ”emigration”, ”immigration”, ”pensionering”, ”död”, ”begravning”, ”testamente”, ”boupptеckning”, ”annat/egen”].
- definition innehåller definitionen av de olika händelserna.

3.3 Utveckling av webbapplikation

För att utveckla en stabil applikation som är så pass komplex som en webbapplikation för släktforskning börjar jag med att utreda vilken teknik jag bör använda vid utveckling av släktträdet: CSS eller tabeller. Sedan undersöker jag och utformar en grundläggande design och filstruktur för webbapplikationen.

För att sedan kunna fortsätta att utveckla applikationen skapar jag ett preliminärt gränssnitt för inmatning av släktdata till databasen, varefter jag

matar in ett släktträd på runt 50 individer för fortsatt utveckling. Detta steg har också en testningsfunktion för att dels se hur formuläret fungerar för användare och dels om databasen beter sig som den bör.

Efter informationsinmatningen designar jag och implementerar det grafiska släktträdet på servern, vilket är ett stort och viktigt steg som består av mycket programmering i PHP. När detta är färdigt utvecklar jag en funktion för det jag kallar ”avgränsad visning av släktträd”, vilket betyder att vissa släktled eller restriktioner kan göras vid visning av släktträdet, t ex att välja att bara visa släktskapet mellan två specifika individer och de personer som binder dem tillsammans.

3.3.1 Teknik för grafiskt släktträd

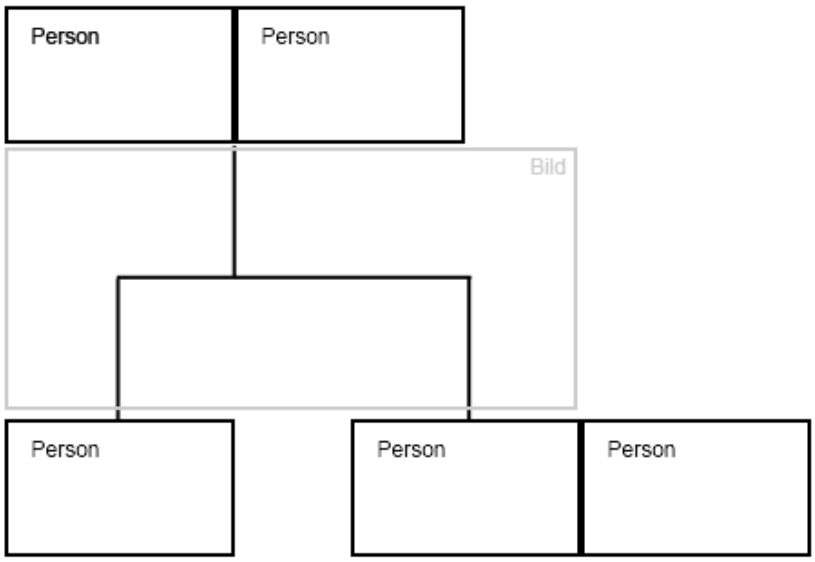
För att kunna ta beslut om teknik för utveckling av grafiskt släktträd för webbapplikationen så måste jag utreda vad som ger ett släktträd dess karakteristik genom att undersöka samband i ett släktträds utformning. Jag har kommit fram till att ett släktträd generellt sett består av två element; entiteter (eller konkret: rutor) med information i form av namn, bilder, årtal med mera och linjer som länkar samman dessa till en logisk enhet. Varje linje representerar ett släktband från förälder till barn. Andra logiska enheter som parrelation representeras genom att individerna placeras nära varandra. En familj är nära placerade entiteter med släktband.

För att kunna representera en individ, eller egentligen vilken information som helst, har jag uppdagat två relevanta tekniska lösningar till webbapplikationen. Den ena lösningen är att använda sig av nästlade tabeller där varje cell anpassas till att antingen vara en individ eller ett mellanrum, den andra är att använda sig av lager i form av div-taggar där varje div-tagg, innehållandes en individ, kan placeras ut med hjälp av angivelser i pixlar. De flesta browsers stöder idag lager som är en möjlighet integrerad med CSS-tekniken.

3.3.1.1 Relationslinjer

CSS används också för att komma åt speciella egenskaper hos HTML-objekt. Man kan med CSS och tabeller relativt enkelt lösa linjerna som representerar släktband i släktträdet. Med hjälp av CSS kan man ange att t ex den övre delen av ramen runt cellen ska bli synlig, man kan också ange exakt bredd och färg på linjen. Vill man använda sig av layer-tekniken sitter inte div-taggarna ihop i någon övergripande tabell, så där skulle istället linjer få dras med ytterligare lager innehållandes t ex bilder med linjer som binder ihop de olika individerna.

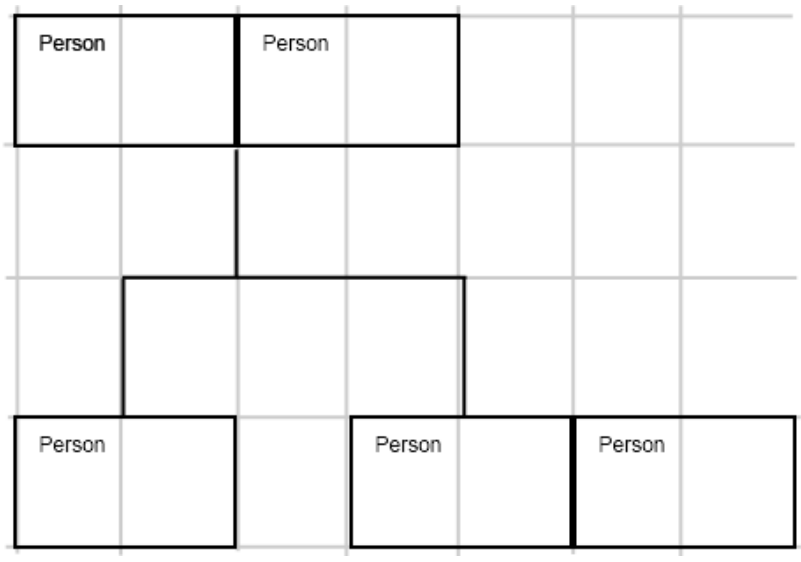
Ex.



Bilden föreställer lösning med relationslinjer som bild vid användning av lager. Rutorna med Person är alltså lager, varje ruta innehåller information om en individ.

När man använder sig av tabeller istället för lager kan man bygga upp ett rutnät med koordinater. En person kan t ex ta upp två horisontella *celler*, detta gör man genom att i den första cellens tagg sätta attributet ”colspan” till 2. Det innebär att man slår ihop två celler till en större cell (se nästa sida).

Ex.



3.3.1.2 Placering av entiteter i tabeller

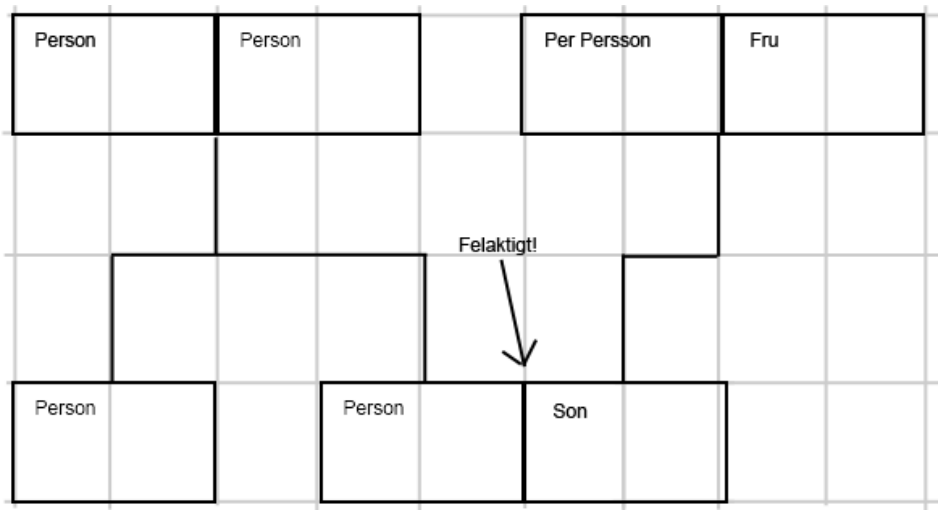
Med både tabeller och lager så är det ”lätt” att placera ut entiteter. När det gäller tabeller så använder man det rutnät som man skapar med celler. Genom

att veta vilken följd dessa ligger i så kan jag placera personerna rätt i förhållande till varandra.

Ex. Ifall personen Per Persson har en fru så vet jag att hon måste komma i nästa cellpar efter Per Persson. Alltså skriver applikationen in en fru direkt efter en make.

Det kanske största problemet när det gäller att placera ut entiteter/personer är när det gäller generationer. T ex i exemplet ovan, om Per och hans fru har barn så kommer dessa tre rader längre ner i tabellen vilket fungerar bra än så länge. Men ifall det finns fler individer i släktträdet som också har barn så riskerar generation att bli förskjuten och/eller krocka (se bild).

Ex.

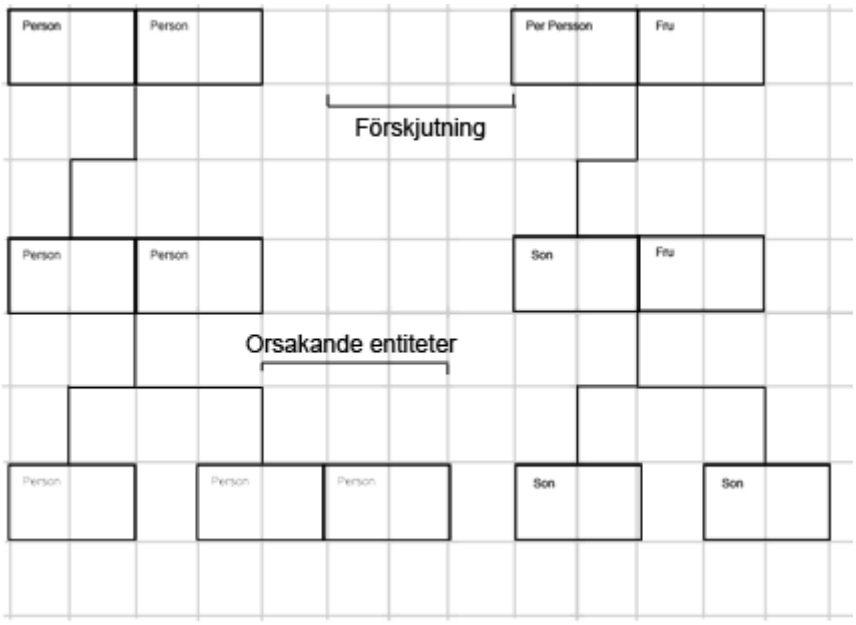


För att undvika detta måste vi alltså redan när vi ritar ut generation ett (I bilden: Per Persson, hans fru och de andra på samma rad) förutspå hur många barn dessa har och ifall Person & Person har fler än ett barn - eller ifall deras barn i sin tur har fått barn med mer än en partner – så måste hela grenen Per Persson flyttas minst en cell åt höger för att skapa mellanrum. Generell kan man säga att:

- Ett barn med en partner ger INGEN förflyttning av nästa familj.
- Ett barn med en partner ger INGEN förflyttning.
- Ett barn med två partners ger en förflyttning på två rutor.
- Två barn ger en förflyttning med en cell.
- Två barn och ett barn har en partner, ger förflyttning med tre celler.
- Osv.

Sedan ökas detta på ytterligare eftersom barnen i sin tur kan ha egna barn som i sin tur har egna barn. Detta ger en ännu större förflyttning (se bild).

Ex.



Förskjutningen i bilden beror i den tredje generationen finns det två syskon varav den ena har en fru. Det blir en förskjutning på en enhet för syskonet och ytterligare två enheter för dennes fru. Hade även det andra syskonet haft en fru hade en ytterligare förskjutning på två enheter inträffat. Inga begränsningar verkar finnas gällande antalet celler i ett och samma HTML-dokument. Jag har provat med 20 000 uppritade tabeller med en cell i varje tabell. Jag har också provat att rita upp en tabell med 1000 rader och 100 celler i varje rad. Alltså totalt 100 000 celler, det var belastande för webbläsaren men det gick att göra på ett korrekt sätt. Jag räknar med att, om trädet följer mönstret givet i ovanstående bild, cirka 30 procent av cellerna kommer att användas till individer. Eftersom en individ består av två sammansatta celler leder detta till att minst 15 000 individer kommer att kunna visas i ett och samma släktträd. Denna siffra är dock väl tilltagen då ett sådant stort släktträd skulle kräva otaligt med datorkraft för att räkna ut alla släktskap och hämta namn och annan information om 15 000 individer.

3.3.1.3 Placering av entiteter med lager/CSS

När vi arbetar i lager har vi inte samma logik att ta användning av eftersom lager använder sig av pixelbunden placering. Vi har alltså inte samma *enhet* att arbeta med som tabellen har med sina celler, därför får vi skapa en egen logisk enhet som man kan specificera löst t ex som 70 pixlar. Ska vi följa tabelldesignen kan vi då utgå ifrån att en entitet blir 140 pixlar bred och ett mellanrum blir då 70 pixlar.

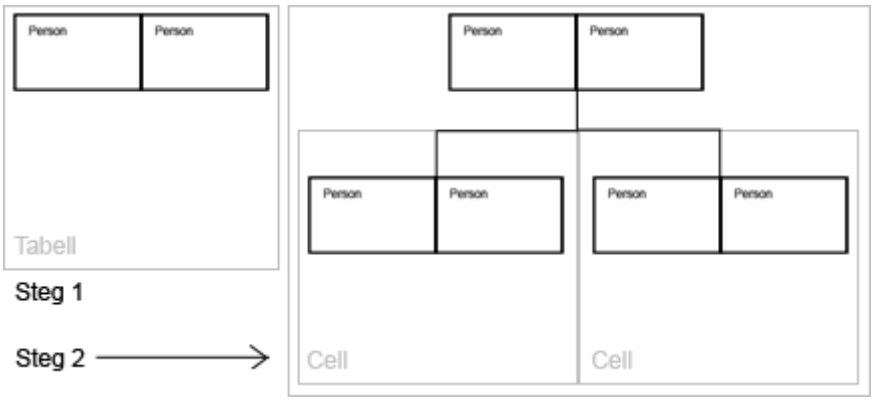
En fördel med lager istället för tabeller är att man själv kan bestämma ordningen av de entiteter man vill placera ut. I tabellen är du alltid tvungen att gå rad för rad uppifrån och ner, vänster till höger. Med lager kan du välja att

arbeta vertikalt; kanske börja från botten med de yngsta släktingarna och arbeta uppåt – på detta sätt hade du kanske kunnat i större grad förutspå förskjutningar. En nackdel med lager är att vissa browsers inte stöder detta utan alla lager kan hamna i en ”hög” eller under varandra.

3.3.1.4 Alternativ tabellanvändning

För att slippa räkna med förskjutningar vid tabellanvändning laborerade jag med nästlade tabeller. Principen var att varje bildat par får en ny ”tabell i tabellen”, d.v.s. en nästlad tabell. Principen har den stora fördelen att eftersom tabellerna automatiskt anpassas till sitt innehåll ”glider” de andra tabellerna undan och relationsstreck förlängs. Utformningen skulle bli av denna modell:

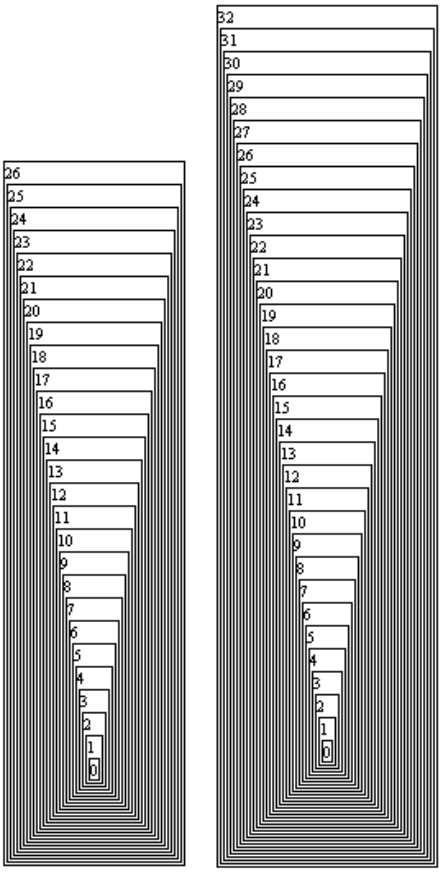
Ex.



I steg 1 ser vi ett par och deras tilldelade tabell (ljusgrå ruta). I nästa steg – om det första paret fick två barn vilka båda två i sin tur skaffar varsin partner (och egentligen barn för att få vara med i ett släktträd) – så blir också de paren tilldelade varsin cell. Skaffar de i sin tur barn får dessa varsin cell. När cellen blir bredare tvingar den undan de angränsande cellerna och den luftighet jag eftersträvas infinnes ”automatiskt”.

Efter testning av nästlade tabeller har jag kommit fram till att webbläsaren Internet Explorer inte klarar mer än 27 nästlade tabeller, medan Firefox klarar motsvarande 33 nästlade tabeller. I båda fallen leder detta till att varje person i släktträdet – om metoden skulle praktiseras – inte kan ha ättlingar i mer än 27 led. Givetvis leder detta till ganska många entiteter i ett släktträd, räknar vi att varje par skaffar två barn som i sin tur skaffar två barn var blir det drygt 134 miljoner individer (!). För att sätta detta i perspektiv så sträcker sig min släktforskning tillbaka cirka 7 generationer. Den äldsta släktingen är född 1807.

Ex.



I norden kan man relativt lätt komma tillbaka till början av 1700-talet genom kyrkoböcker, följer man sedan rättegångspapper och jordeböcker och har tur kan man komma tillbaka till 1530-talet, har man adliga rötter är det möjligt att kanske forska sig ner till strax före år 1000. Räknar man att en generation är omfattar 20 år, kommer man tillbaka 540 år från dagens datum. Men förr i tiden var det vanligare att skaffa barn tidigt i livet, det innebär ett kortare generationsintervall än 20 år.

Sammanfattningsvis betyder detta att denna teknik inte räcker till vid VÄLDIGT omfattande och krävande forskning speciellt om man är ute efter en antavla (som bara omfattar rakt stigande blodsband: far, farfar, farfars far, osv.). Det finns däremot stora fördelar med denna teknik. Främst då ingen pixelplacering måste användas vid utplacering av individer i släktträdet. Med andra ord så måste ingen sådan matematik implementeras, men däremot måste den ordning som HTML och browsers renderar tabeller följas. Det vill säga att en inutiliggande tabell måste avslutas innan den ytterliggande tabellen avslutas etc. För övrigt så är begränsningarna egentligen irrelevanta då man ändå inte i realiteten kan visa 27 generationer samtidigt på en normalstor datorskärm. Det är därför i varje fall aktuellt att på något sätt begränsa antalet samtidigt visade generationer med en funktion där man kan ”klicka vidare” i trädet om man vill se mer.

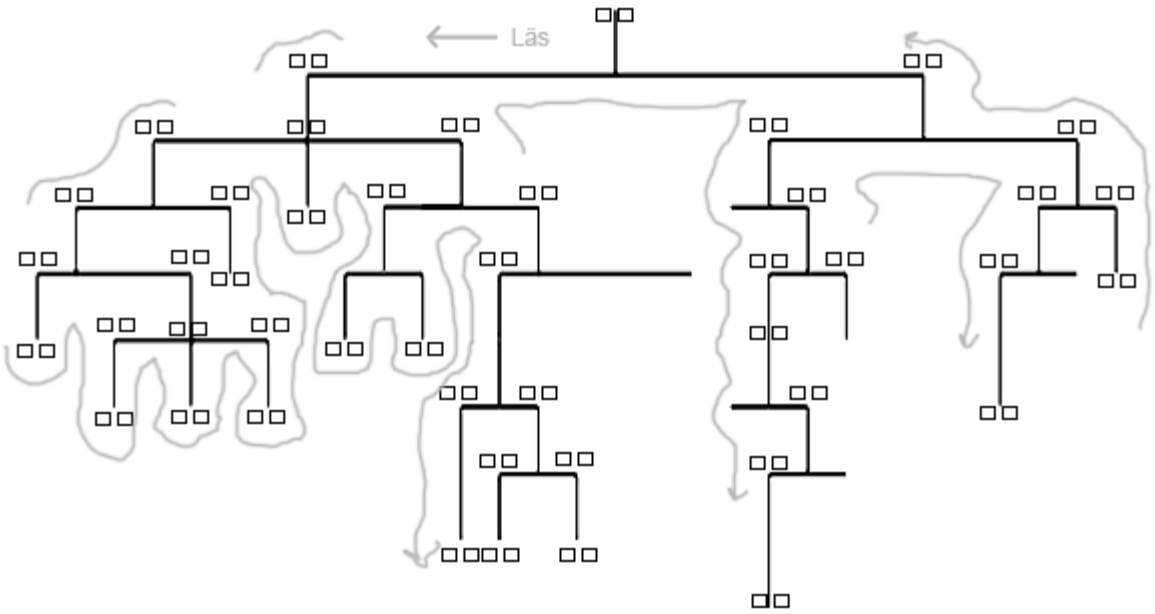
Ex. Du har en begräsad visning på t ex fem generationer. Ditt släktträd innehåller 9 generationer. När du först visar släktträdet är stamfadern högst upp i trädet, vi kallar honom för P1. Han i sin tur har ett par barn, som har barn osv. Programmet ”bryter av” och klipper bort alla yngre släktingar till P1’s 5’e generation barn. Vill du sedan se dessa, så trycker du på ett av P1’s 5’e generation barn, då visas dessa valda barn som stamfader (högst upp i trädet) och hans/hennes barn i ytterligare fem generationer.

3.3.1.5 Att läsa släktskap ur databasen

För att kunna välja mellan lager/CSS eller tabeller - som är rätt lika i sin tillämpning som teknik till släktträdet – så måste jag utreda ytterligare hur man på bästa sätt läser ett släktträd från databasen, för att på det viset kanske uppdaga om någon av teknikerna då passar ändamålet bättre.

Ifall jag utgår från att släktträdet utgår från en person – vilket gör det till en *stamtavla* – och visar alla hans släktingar. Då har jag alternativet att antingen börja direkt att placera ut den personen och sedan fortsätta i horisontell led steg för steg. Men för att då få reda på förskjutningen måste jag ta reda på hur många barn personen i fråga har och även ifall dessa i sin tur barn och så vidare, tills den yngsta generationen har stötts på. För att detta ska vara ett effektivt sätt så måste dessa data sparas på något sätt, annars får man göra om samma sökning för varje individ i trädet, fast utan en generation ”på toppen”. Möjligtvis kan man lagra varje förskjutning i en lista för att sedan applicera förskjutningarna efterhand individerna placeras ut. Denna teknik borde fungera lika bra oavsett om lager/CSS eller tabeller används.

En annan modell av släktskapsläsning från databasen skulle kunna vara att detta ”kravlande” sätt att följa varje ”gren” moturs hela varvet runt. Praktiskt hade det sett ut som följer.



När man läser släktträdet på detta sätt kommer man alltid att känna till förskjutningen innan man behandlar nästa ”gren” eftersom allt till vänster om kommande gren redan är behandlat, därför kan man med denna läsning börja rita upp släktträdet direkt utan att känna till antal barn eller annan grundläggande fakta. Detta kan vara såväl resurssparande som förenklande av utvecklingsprocessen.

Eftersom den ”enkla” tabell-tekniken (inte nästlade tabeller) inte kan rita annat än celler från vänster till höger - och cellrader uppifrån och ner - så är det alltså inte möjligt att använda fördelarna med detta kravlande lässätt tillsammans med tabell-tekniken. Använder man sig däremot av nästlade tabeller kan man mycket väl tänkas implementera denna teknik, eftersom det är i den ordningen man ändå ritar upp tabellerna i varandra (se illustration under punkt 3.3.1.4).

3.3.1.6 Buffring av resultat innan utskrift

Slutligen kan man alltså konstatera att svårigheten när man använder tabeller ligger i att dessa måste skapas en entitet i taget från höger till vänster och en rad i taget uppifrån och ner. För att kunna göra detta måste all information om trädet först mellanlagras för att sedan ritas ut i ”rätt ordning”.

Använder man sig av lager/CSS kan man placera ut entiteter godtyckligt – i den ordning man föredrar – och med hjälp av detta kan man rita upp trädet ”samtidigt” som man söker efter samband, genom att läsa relationerna ”moturs”. Svårigheten med lager är att sedan länka ihop utplacerade entiteter med linjer. Ifall man skulle välja att använda sig av bilder för relationslinjerna (eller andra entiteter med tabeller som blir till streck) så måste man också klura ut vilka olika linjer som behövs och var dessa ska placeras. För att göra detta behövs de utplacerade entiteternas positioner också lagras.

Detta resulterar i att för att slippa mellanlagring/buffring så måste jag välja nästlade tabeller, eftersom att med både enkla tabeller och CSS/lager så måste vissa koordinater i släktträdet mellanlagras/sparas undan innan utskrift.

3.3.1.7 Hur kan man lagra ett släktträd innan utskrift?
Både med tabeller och med CSS måste man lagra släktträdets uppbyggnad, vilken man härleder från informationen i databasen till en konkret datamodell vilken sedan kan skrivas ut post för post. Till min hjälp har jag i PHP möjligheten att arbeta med *arrayer* (på svenska: listor). Arrayer i PHP kan man göra nästlade, så varje post i ett *array* innehåller ännu ett array, som i sin tur kan innehålla ännu ett array. Detta kan jag troligen utnyttja då även vårt släktträd är uppbyggt på liknande sätt om man använder sig av moturs kravling av databasen, att varje nytt array är en ny ”gren” på släktträdet.

Man skulle - om man använder sig av tabeller när man ritar släktträdet - t ex kunna använda arrayer genom att huvudarrayet, i vilket de andra arrayerna är samlade, kan representera cellrader. Det är då lätt att använda sig av koordinater för de olika cellerna som korresponderar med arrayerna. T ex om huvudarrayets ruta ett och arrayet i denna ruta har lika många poster som raden har celler. På det sättet kan man ange en cells placering som: array[y][x]. Vid utskrift kontrolleras varje ny cell mot arrayet för att se vad den ska innehålla, är arrayet tomt på den platsen blir cellen tom och alltså en mellanrumscell.

För att undersöka om detta var möjligt i praktiken så gjorde jag en mindre testdesign. Vid testdesign, för att undersöka buffert och sedan utskrift med hjälp av enkla tabeller (inte nästlade), stötte jag på stora problem. Problemen låg mest i programmeringsmetodiken, att omsätta teorin till praktiskt kod visade sig vara svårt. Efter att ha konstruerat en rekursiv funktion som tog sig igenom hela släktträdet - för att dokumentera vilka personer som tillhör vilken generation - visade det sig att programmet feltolkade information såtillvida att förskjutningar i generationerna ställde till problem. Att använda den här tekniken blev alltså för komplext, men teoretiskt möjligt för någon med mer erfarenhet i strukturerad programmering än jag.

3.3.1.8 Beslutstagande i frågan om teknik för grafisk släktträd
Enligt mina resonemang, kring teknik för grafiskt släktträd, kan jag sluta mig till att det finns några fördelar och några nackdelar för varje teknik. Tekniken med tabeller där celler fungerar som koordinater är en gedigen teknik med plats för tillräckligt många individer för att bygga upp ett utförligt släktträd med anor tillbaka till innan tusentalet – och förmodligen bra mycket längre än så. Att rita ut relationsstreck blir förmodligen inga problem då mellanliggande

tomma celler kan användas med CSS för att markera dess kanter. Problem med tekniken är att varje ruta måste ritas i en bestämd följd – från vänster till höger, rader uppifrån och ner. Detta resulterar i att släktträdet måste lagras i ett array innan det ritas ut.

Med lager/CSS är man oberoende av den ordning som tabeller innebär, utan valfri ruta kan placeras ut på valfri plats i valfri ordning. För att hitta placering på rutorna kan man skapa en *egen enhet* och använda sig av samma teknik som för tabeller och celler. Problemet med CSS är att för att rita ut relationer i trädet måste du ändå ha buffrat (sparat undan) trädstrukturen innan du ritar ut det. Vissa äldre browsers klarar inte av lager/CSS utan dessa kan hamna i en hög någonstans eller uppradade under varandra.

Vad gäller komplexitet så är både CSS och enkla tabeller beroende av en hel del matematik för att lösa problemen med positionering. Efter ett försök med buffring av koordinater och utplacering av individer i ett släktträd, kom jag fram till att detta är väldigt komplext med både tabeller och CSS.

Vad gäller nästlade tabeller, så finns det egentligen bara en negativ aspekt och det är begränsningen hur många tabeller som kan ligga i varandra. IE klarade 27 och Firefox klarade 33. Det är dock rätt irrelevant eftersom det egentligen finns mycket liten nytta i att visa alla individer i ett så stort släktträd samtidigt. Bättre (och mindre belastande för t ex server och klientsidan) är att begränsa släktträdet till t ex 15 generationer och att användaren sedan kan navigera sig vidare om han är intresserad av att se mer. Nästlade tabeller har fördelarna att ingen matematik måste tillämpas och trädet kan ritas ut utan att behöva mellanlagras och processeras i en buffert. Tabellkanterna blir mer eller mindre automatiskt till relationslinjer, och kan säkert ganska lätt användas till detta.

Med detta i beaktande är nästlade tabeller lättast att arbeta med och uppnår samma mål med mindre ansträngning. Tekniken leder också till mindre serverbelastning, vilket är eftersträvansvärt speciellt vid omfattande användning. Vald teknik för det grafiska släktträdet blir alltså med stora fördelar *nästlade tabeller*.

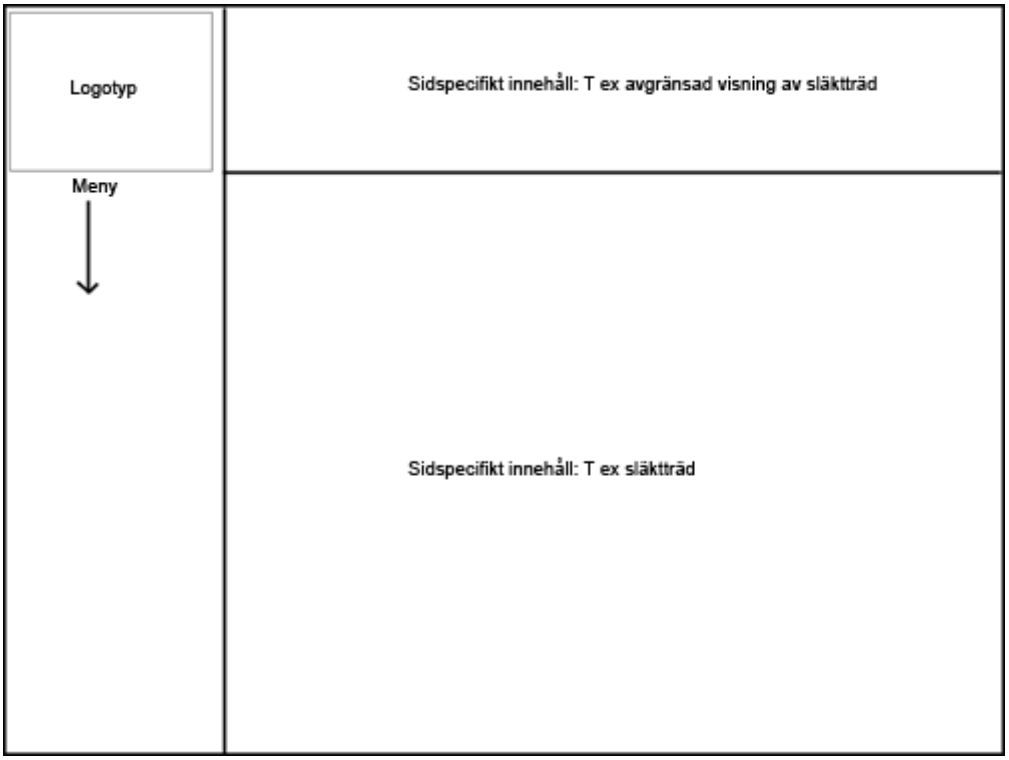
3.3.2 Grundläggande design av webbapplikation

Beträffande layout för en webbsida finns det några vanliga alternativ:

- Att bygga lager med CSS.
- Att använda tabeller för innehållet.
- Att använda sig av frames.

Eftersom det grafiska släktträdet kan bli – teoretiskt – hur stort som helst så är varken CSS-lager eller tabeller särskilt bra grund till en sådan layout. Är trädet väldigt stort och användaren vill ta sig tillbaka till menyn eller ändra premisserna för släktträdets visning så måste användaren skrolla väldigt långt. Att användaren ser sina alternativ – både meny och specifika funktioner - är viktigt i ett användarvänligt perspektiv. Frames däremot fungerar så att browser-fönstret delas upp i delar, vart och ett får sina egna scrolling-kontroller. Detta gör att menyn kan finnas i en frame, släktträdet i en frame och kontroller för släktträdet i ytterligare en frame. Detta leder till ett betydligt behändigare sätt att hantera släktträdet.

Ex.



Ovanför är en s.k. trepanelslayout med frames, där fördelarna är att varje ”frame” (bilden: inramad rektangel) har egna handtag för scrollning och där bara en frame behöver uppdateras åt gången, vilket leder till mindre serverbelasning vid sidbyte och kräver dessutom mindre bandbredd hos klienten. Sedan bredband har blivit mer tillgängligt de senaste åren är det senare inget tungt vägande argument för frames.

3.3.2.1 Sidhuvud och sidfot

Varje fil på servern med större syfte kommer att behöva tillgång till vissa funktioner, såsom uppkoppling till databas, start av session för inloggning och andra vanliga funktioner. Därför anser jag att det är klokt att skapa en fil innehållandes denna information, som sedan varje fil länkar in i början av dokumentet. Detta förfaringssätt ger systemet flera viktiga fördelar; varje fil

tar mindre plats då informationen inte behöver upprepas i flera filer och att viktiga inställningar för många filer går att ändra genom att bara ändra i filen med sidhuvudet. Det behövs även en fil för att stänga databaskoppling och andra rutinprocedurer. Den inkluderas då i slutet av varje fil där sidhuvudet har inkluderats i början.

3.3.2.2 Modulhantering

Den färdiga applikationen är tänkt som ett verktyg för släktforskning, och där inte bara möjligheten att skapa ett släktträd kommer att ingå. Därför kommer jag att designa applikationsgrunden för att kunna hantera moduler, dessa kan sedan programmeras vid senare tillfälle och på ett enkelt sätt implementeras direkt till släktforskningsprogrammet genom en enkel procedur.

För att uppnå denna funktionaliteten tänker jag att applikationen ska söka igenom en katalog efter moduler, när den hittar en speciell fil i katalogen så inkluderar den informationen i denna fil. Filen innehåller information om modulens namn och även information om hur den ska behandlas av systemet.

3.3.2.3 Språkstöd

Applikationen, i sitt färdiga tillstånd, ska kunna översättas. För att kunna göra detta måste ett stöd för fler språk införas. Detta gör jag genom att byta ut applikationstypiska uttryck – som annars skulle hårdkodas i varje html-fil – istället byts ut till konstanter. Dessa konstanter definieras i en separat fil; en *språkfil*. Genom att länka in denna språkfil i sidhuvudet på varje sida så får alla sidor tillgång till dessa uttryck genom samma variabler. För att sedan byta språk så ändrar användaren vilken fil som inkluderas.

För att moduler med egna språkfiler ska kunna implementeras och översättas så måste en standard på språkfilens namngivning definieras. Språkfilen döps efter det engelska namnet på språket med små bokstäver. Svenska blir till exempel: ”swedish.php”, engelska blir ”english.php” och franska blir ”french.php”. Ifall någon språkfil mot förmodan skulle saknas så anges istället engelska som språk. Engelska blir då *standardspråk* i applikationen, detta för att engelska är det språket som mest används på Internet och som flest har tillgång till att översätta själva. Svenska som standardspråk hade förmodligen gjort att utländska användare av programmet inte ens skulle förstå hur de ska navigera sig till inställningarna och ändra språk på applikationen.

3.3.2.4 Färgteman och stilmall

För att enkelt ändra utseende på en webbapplikation finns det idag många sidor som utnyttjar så kallade *teman*. Teman är ett sätt för administratören att enkelt kunna ändra utseende efter egen smak, så att applikationen får en egen stil. Rent praktiskt är ett tema en alternativ konstruktion av hemsidan, oftast

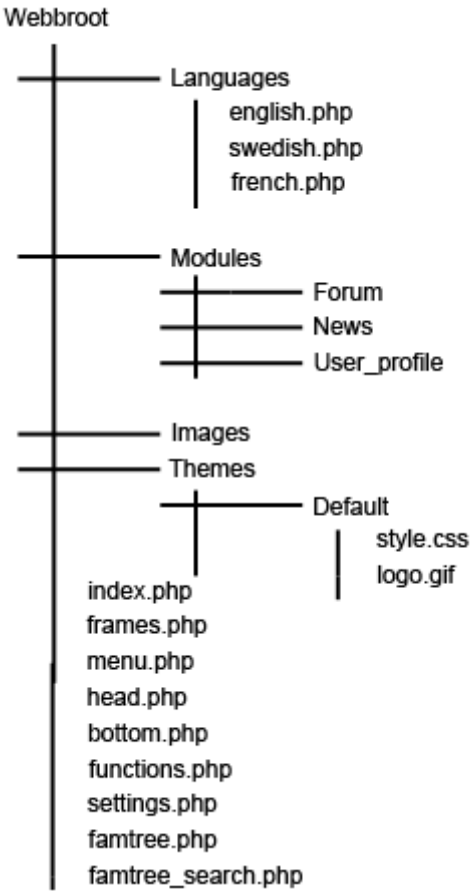
ändras bara små detaljer på hemsidan i de olika teman som finns att välja på. Färg på bakgrund, text, typsnitt, logotyp och knappar i form av bilder är några sådana saker. För att kunna använda olika teman skapar jag en katalog där varje underkatalog innehåller en stilmall och alla bilder som systemet använder. Underkatalogens namn är också namnet på *temat*. Katalogen som heter ”default” är det temat som automatiskt väljs om inget annat tema anges. Saknas det information i ett annat tema och detta är förvalt av administratör eller användare, så används den saknade informationen istället från temat ”default”.

3.3.2.5 Inställningar

Specifika inställningar som det INTE finns praktiskt nytta av att ändras under drift - såsom lösenord till databas, sidans titel, metainformation om sidans innehåll, sidans webbadress, applikationens tema, applikationens språk och eventuellt andra inställningar – lagras i en inställningsfil. Detta för att förenkla implementation på en ny server, i praktiken ska den installationsansvarig bara behöva ändra i denna fil för att applikationen ska fungera på den nya servern. Då med visst förbehåll för att databasen ska sedan tidigare ha implementerats och att servern i övrigt är kompatibel med applikationens specifikationer.

3.3.2.6 Filsystem

Applikationsfiler, sådana filer som följer med grunden till applikationen som t ex filer för släktträdet, sidhuvud och sidfot, läggs direkt under *webbroten* (dvs den katalog som utgör grunddomän på servern). Teman placeras i katalogen ”themes”, varje tema ligger sedan i varsin underkatalog. Tilläggsmoduler placeras i katalogen ”modules”, varje modul ligger sedan i varsin underkatalog. Bilder tillhörande de olika släktträden placeras i en katalog ”images”. Språkfilerna ligger i katalogen ”languages”.



Filträdet ovanför är ungefärligt och filer kan tillkomma under utveckling av applikationen. Se specifikation på filernas funktionalitet under nästa rubrik; ”Applikationsfiler”.

3.3.2.7 Applikationsfiler

Applikationsfiler är de som bygger upp applikationens grund. Dessa ligger i webbroten. Här följer en specifikation av filernas namn och deras funktion:

- index.php : Är den fil som exekveras först vid inträde i applikationen. Filen ska, i vidareutveckling av applikationen, innehålla inloggningsfunktionalitet för användare.
  Här ska också finnas information om webbsidan och dess syfte som administratören kan ändra efter ett sidspecifikt syfte.
- frames.php : Styr de frames som applikationen är uppbyggd av. Efter att användaren har loggat in genom index.php så ritas frames upp av frames.php och fylls med meny och sidspecifikt innehåll.
- menu.php : Innehåller menyn och är även den fil som binder ihop de olika modulerna genom att den genomsöker katalogen modules och gör modulerna åtkomliga genom länkar i en meny.
- head.php : Är filen som innehåller sidhuvudet. Dvs metainformation om sidans innehåll, databaskoppling, <head> och <body>-taggar med sidan titel. Denna fil inkluderas sedan i varje fil med html-innehåll i

applikationen för att styra utseende och tillhandahålla databaskoppling samt tillgång till gemensamma funktioner.

- bottom.php : Innehåller avslut på processer som påbörjats av head.php (såsom databaskoppling) och implementeras sist i varje fil där head.php har implementerats först.
- functions.php : Innehåller funktioner och länkas in i head.php för att göras tillgängliga i alla filer där head.php är inlänkad.
- settings.php : Innehåller inställningar för applikationen som ställs in vid installation av applikationen; såsom lösenord till databas, sidans titel, metainformation osv.
- famtree.php : Innehåller kod för uppritning av grafiskt släktträd.
- famtree_search.php : Innehåller kod för avgränsad visning av grafiskt släktträd dvs. ett formulär där val för hur släktträdet ska visas kan göras.

3.3.3 Preliminärt gränssnitt för informationsinmatning

Mitt gränssnitt för informationsinmatning har jag valt att dela upp i fyra olika delar och en övergripande sammanfattning. Delarna är: Skapa ett släktträd, Skapa person, Lägg till händelse till Person och Skapa relation. Dessa olika moment är logiskt rangordnade från ett till fyra. Tanken är att man först skapar ett släktträd, sedan skapar man ett antal personer med grundläggande information såsom namn och kön. Dessa kompletterar man sedan med såkallade händelser, såsom födelse, död, utbildning med mera för att få en mer karakteriserad person. När personen är klar länkar man ihop flera personer med relationer och knyter dessa till varandra och till det släktträd man skapade i steg ett.

Rent praktiskt har jag utnyttjat applikationens möjlighet till modulhantering och lagt det preliminära inmatningssystemet som en modul, detta för att det just är primärt och ska kunna bytas ut eller kompletteras. Varje ”moment” ovan har fått en egen fil placerad i modulen kallad ”addform”. Filen ”addform.php” innehåller dels en förklaring till hur man ska använda de olika momenten, dels en lista på alla personer i persontabellen i databasen. I listan står det om varje person har fått en mor och en far (relationer) tilldelade, genom att klicka i listan kan man få upp ett delvis förifyllt formulär, för att underlätta komplettering av information. Det finns även en direktlänk för att lägga till en händelse till personen.

Hantering av formulärsdata och insättning/uppdatering av databasen sköts av filen ”postcode.php” som efter anrop visar ett statusmeddelande för användaren efter tillagd information i databasen. Vissa felmeddelanden är också inkopplade efter uppenbara fel som måste undvikas i databasen, bland annat möjligheten att skriva in två mödrar till samma person undveks genom

en enkel kontroll i databasen – ”postcode.php” uppdaterar då istället databasen.

3.4 Testning

Genom att mata in ett stort och redan kvalitetskontrollerat släktträd kontrollerar jag dels databasens stabilitet och dels ifall det finns något utrymme för att ytterligare effektivisera inmatningsformuläret och i så fall hur detta ska gå till.

Efter inmatning av släktträdet gör jag en testdesign av släktträdet, detta för att testa den teoretiska metod jag har kommit fram till i metodsteget ”Efterforskning av teknik för grafiskt släktträd”.

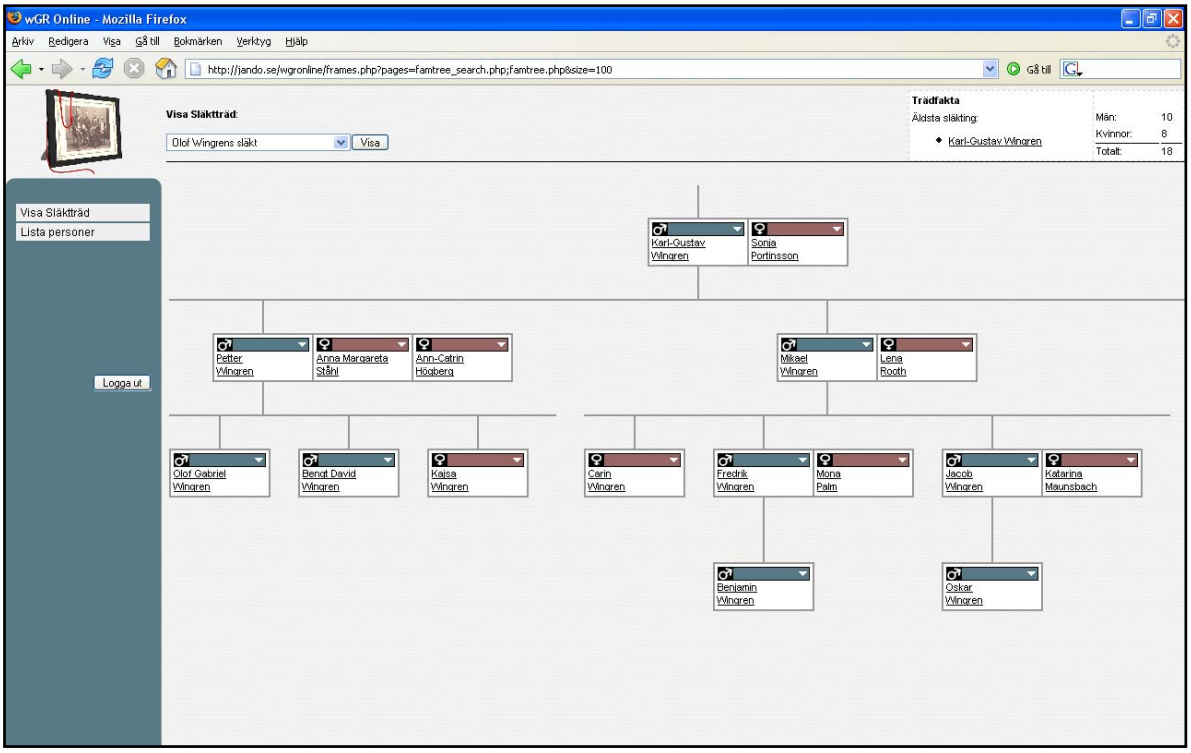
3.4.1 Inmatning av släktträd

Vid inmatning av 51 individer, från en redan kontrollerad släktforskning, har jag upptäckt vissa mindre problem i det preliminära gränssnittet för informationsinmatning. Problemen rör framförallt överskådlighet och enkelhet i inmatningen, detta då det inte finns någon ”sammanfattning” över varje person och dennes personuppgifter.

En möjlig förbättring är att lägga till möjligheten till att välja far och mor redan vid inskrivning av grundläggande personfakta. Detta förfarande skulle spara tid då ett tiotal klick hade undvikits och dessutom en totalt sex sekunders väntetid vid databaskoppling. Fler förbättringar som är nödvändiga för en effektiv inmatning är t ex intelligent kontroll av rimlighet där förslag till föräldrar och släktskap kan göras av applikationen. Sortering av personer på – av användaren – valfritt sätt, t ex efter namn, efternamn, pnr eller av annan karaktär, för att hjälpa användaren att snabbare hitta rätt person.

3.4.2 Testdesign av grafisk släktträd

Jag valde att använda mig av nästlade tabeller, efter att ha konstaterat att tekniken belastar servern mindre än tabeller och CSS. Den skulle också lösa problematiken med relationslinjer och behöver inte buffras eller och pixelplaceras som tabeller/CSS. De senare teknikerna riskerade även att bli för komplicerade för någon med min begränsade erfarenhet av strukturerad programmering i större sammanhang. Nästlade tabeller innebar en mycket lättare lösning, som var mer rakt på sak än de andra teknikerna. Jag kom också fram till att den tekniken underlättar för servern eftersom det bara handlar om en genomsökning av databasen och ingen egentlig efterbehandling av resultatet därifrån.



Att programmera metoden med nästlade tabeller gick ganska bra. Det svåraste var att få en rekursiv funktion att fungera bra, det var också tidvis krångligt att få undantagsfall, som månggifte, att visualiseras på ett övertygande sätt. Jag är dock mycket nöjd med den slutliga versionen och kan konstatera att tekniken fungerar bra och går att finjustera till att passa kraven. I prototypen uppfylls dock inte kravet på att ”avancerade släktskap” (t ex kusiner) kan visas.

I övrigt fungerade applikationen utmärkt med modulhantering och *frames* som tillåter stora släktträd att visas. Språkstöd, fil med inställningar och användning av ”utseende teman” fungerar också alldeles utmärkt också i kombination med släktträd och moduler.

4. Resultat

Det finns en stor mängd resurser för släktforskning på Internet. Programmet *MinSläkt* är mycket likt min webbapplikation i funktionalitet förutom att det är klientbaserat och har inte stöd för flera användare. Det finns också en väl utbredd lagringsstandard för släktforskningsdata som heter *GEDCOM*. Denna är däremot gjord för lagring av släktforskningsdata i filformat och inte för en aktiv webbapplikation för flera användare. Följande funktioner fann jag viktiga att implementera i webbapplikationen:

- Personer knyts till personer genom parrelation, släktband eller händelser (de två första är synliga i släktträd).
- Händelser är knutna till personer.
- Visning av anfäder eller ättlingar som möjlig separat vy av släktträdet. (Till exempel att visa ett träd med anfäder till individ, eller alla ättlingar till en annan.). Denna punkt ställer krav på släktträdets dynamiska design.
- Källhänvisning vid införd information.
- Bilder knutna till personer och händelser.

Vad gäller lagrande av släktskap i databasen så kom jag fram till att använda mig av en separat tabell där avancerade släktskap kan lagras, istället för att som t ex i GEDCOM skapa familjer. En separat tabell är ett snabbare och mer databasvänligt sätt att lagra släktskapen på. Databasen i sig delade jag upp i fyra starka och två svaga entiteter. Principen är att en relation mellan två personer utgör länk mellan person-tabellen och familjeträdstabellen. På så sätt kan varje person i person-tabellen återanvändas i flera sammanhang. För att ytterligare beskriva personer kan man lägga till ”händelser” till varje person, en händelse kan t ex vara födsel, dop, examen, död, begravning osv.

Efter att ha färdigställt databasen och implementerat den på server utan några problem, så påbörjade jag efterforskningar på hur jag skulle utforma tekniken för det *grafiska släktträdet*. Det var svårt att välja vilken teknik (mellan *tabeller, nästlade tabeller* eller *CSS/lager*) jag skulle använda mig av, framförallt då det var mycket spekulativt. Avgörandet gavs av att vissa browsers inte har fullt stöd för CSS/Lager och att därför tabeller var mer allmängiltigt. I övrigt hade de två olika förfaringssätten inga större fördelar eller nackdelar gentemot varandra som hjälpte till i beslutet. Nästlade tabeller däremot visade sig ha betydliga fördelar i praktiken, mycket tack vara att metoden är rätt så uppenbar i sin utformning och lätt att arbeta med då den inte innebar någon buffring av släktträdet.

Vid design av applikationens *grundlayout* kom jag fram till att använda mig av *frames* – för att lätt kunna hantera stora släktträd i applikationen – och av en intelligent modulhantering, så att nya funktioner i applikationen lätt kan designas och läggas till allteftersom utveckling av applikationen fortskrider. Både i databas och i grundlayout lämnade jag plats för möjligheten att *implementera bilder* knutna till databasen, men att implementera detta var inte prioriterat inom ramarna för detta examensarbete.

Jag implementerade fortsättningsvis också ett *gränssnitt för informationsinmatning* för att i testsyfte mata in redan gjort släktdata i databasen och har på detta sätt lärt mig mer om hur inmatningen kan göras och hur inmatningsformuläret kan förbättras. ”Släktdatan” använde jag mig sedan av när jag gjorde en testdesign av det grafiska släktträdet på en server, baserat på den metod jag kommit fram till i efterforskningssteget av detsamma. Testdesignen gick utan större missöden och tekniken visade sig fullt tillämpbar i praktiken, vissa förbättringar kan dock göras innan trädet är stabilt. Detta gäller främst undantagsfall i avancerade släktskapsband (t ex vad gäller kusiner).

I övrigt fungerar applikationen bra, med språkstöd, inställningar, databaskoppling och med modulhantering, såsom inmatningsformuläret. Det grafiska släktträdet har lite eller ingen inverkan på systemets funktionalitet i övrigt.

5. Slutsatser

Designen och implementation av databasen har fram till slutet av detta examensarbete inte inneburit några problem. Problem kan däremot komma att uppstå vid vidareutveckling av applikationen då stöd för flera användare, multimedia i form av bilder, inloggning, sekretess med mera kommer införa en större komplexitet. Databassökningar kan t ex behöva buffras om databaserna blir väldigt stora och många användare kan ge högre krav på säkerhet genom hela applikationen. Viss kod får då lov att anpassas, i stort innebär det att ”låsa” varje fil från illegal åtkomst genom att sätta in kontroller i den fil som utgör sidhuvudet.

Några problem uppstod dock med programmeringen av webbapplikationen. De mindre problemen kunde lösas antingen genom att läsa dokumentationen om PHP, eller genom att felsöka koden noggrant. Inget problem har dock varit av sådan komplex art som det som uppstod under implementeringen av det grafiska släktträdet. Svårigheterna där var de största i projektet men inte av oöverkomlig art. Problemen låg programmeringsmetodiken, som bitvis blivit komplicerad med t ex rekursiva funktioner och undantagsfall av placeringar.

Vad gäller den avgränsade visningen av släktträdet - möjligheten att påverka visningen av släktträdet på olika sätt med filter och olika inställningar - så sträcker den sig endast så långt som till att användaren kan välja vem som ska vara stamfader i släktträdet. Det vill säga vem som ska stå högst upp i detta. Ytterligare avgränsningar är dock fullt möjliga att implementera men så är inte gjort i prototypen.

På grund av de problem jag stött på (se ovan) så uppfyller inte resultatet alla de *framgångskriterier* som ställdes i inledningen till arbetet:

- *Avancerade släktskap kan lagras i databasen.*
  Kriteriet är uppnått.
- *Databasen innehåller minimalt med redundans och null-värden.*
  Kriteriet är uppnått. Men kan vid vidareutveckling komma att behöva bufferttabeller för att snabba på sökningar i databasen, då dessa med aktuell teknik kan bli ganska omfattande och påfrestande.
- *Släktträdet i webbapplikationen följer praxis i sin utformning.*
  Kriteriet är uppnått. Men kan kompletteras med ytterligare funktionalitet för att i högre grad göra detta för t ex avancerade släktskap.
- *Användaren kan i större utsträckning välja avgränsad visning av olika delar genom sökning av släktträdet.*
  Kriteriet är uppnått till viss del, då man kan välja vem som ska vara

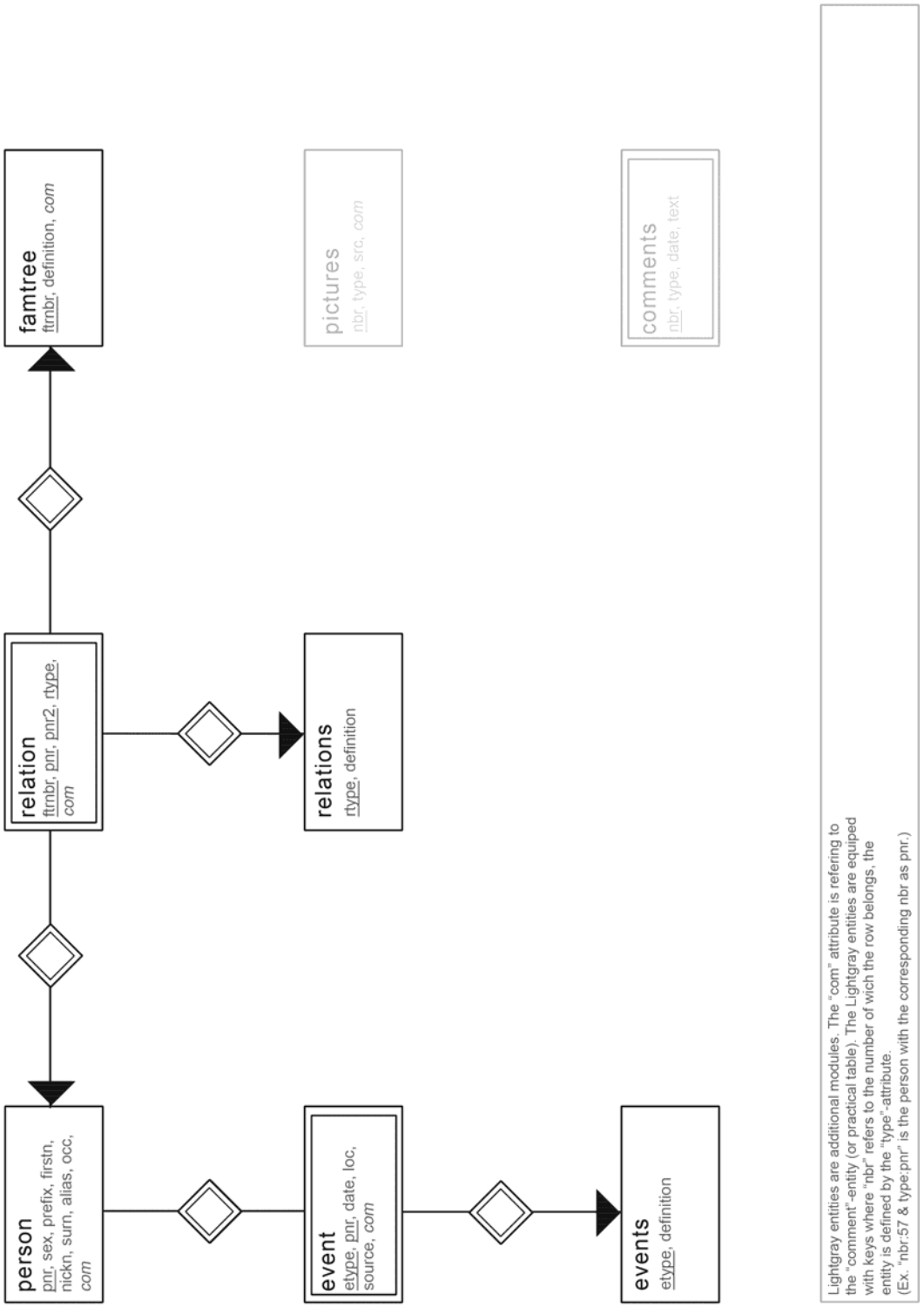
stamfader i trädet och på det sättet avgränsa visningen. Övriga avgränsningar går att implementera, men är inte implementerade i prototypen på grund av tidsbrist.

I det hela tycker jag att arbetet har uppfyllt målet och syftet. Den har blivit en stabil bas att vidareutveckla med nya funktioner och förbättringar. Vissa av problemen var av den art att de var svåra att förutsäga, men uppdagades genom testdesign och detta kommer jag att bära med mig i nästa fas av utvecklingen.

6. Referenser

1. http://www.genline.se/ (Mars 2006)
   ”Genline – Släktforskning”
2. http://www.slaktdata.org/ (Mars 2006)
   ”Föreningen Släktdata”
3. http://www.senitel.org/geneal.htm (Mars 2006)
   ”senITel – Släktforskning”
4. http://www.google.se (Mars 2006)
   ”Google sverige”
5. http://www.dannbergsdata.se/ (Mars 2006)
   ”MinSläkt – Släktforskningsprogram för amatörer”
6. http://homepages.rootsweb.com/~pmcbride/gedcom/55gctoc.htm
   (April 2006) “The GEDCOM Standard Release 5.5”
7. http://sv.wikipedia.org/wiki/Sl%C3%A4ktforskning
   (April 2006) “Genealogi – Wikipedia, den fria encyklopedin”
8. http://www.gnu.org/copyleft/gpl.html
   (April 2006) “GNU – General Public License”
9. http://www.notisum.se/rnp/sls/lag/19980204.HTM
   (Maj 2006) “Personuppgiftslag (1998:204)”

Bilaga 1 – ER diagram



Bilaga 2 - GEDCOM datamodell