Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Nikolai Steen Kjosnes

Relasjonell database for analyse av historisk demografisk data

En løsning utviklet i MySQL og PHP

Bacheloroppgave i Informatikk med spesialisering i informasjonsbehandling
Veileder: Tore Mallaug
Mai 2020

NTNU
Kunnskap for en bedre verden

Nikolai Steen Kjosnes

Relasjonell database for analyse av historisk demografisk data

En løsning utviklet i MySQL og PHP

Bacheloroppgave i Informatikk med spesialisering i informasjonsbehandling
Veileder: Tore Mallaug
Mai 2020

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU

Kunnskap for en bedre verden

Innholdsfortegnelse

1. Abstract / Sammendrag

1.1 English

Large amounts of historical demographic data from church records and censuses have been collected and stored in Excel-spreadsheets by Oddgeir Fossli. There is now a need to develop a database such that these data can be stored in a way that makes further analysis easier.
The database developed for this task is a relational database in MySQL. A PHP-based solution has also been developed to use this database for analysis. The PHP-solution fetches identifying data about each of the people from the source tables and helps the user in linking the sources where the same person appears in different sources. An unambiguous ID is made for each person, and you can see their life from baptism to death through the sources they have appeared in.
In addition, the solution aids in analysis by finding connections between the registered persons by looking at the sources, i.e. connecting parents and their child.

1.2 Norsk

Store mengder demografisk data fra kirkebøker og folketellinger har blitt lagret i Excel-filer av oppgavestiller Oddgeir Fossli. Det er nå nødvendig å utvikle en databaseløsning slik at dataene kan bli mer hensiktsmessig lagret og kan brukes til analyse.
Databaseløsningen som har blitt utviklet er en relasjonell database i MySQL. Det har blitt laget en PHP-basert løsning for å samhandle med databasen. PHP-løsningen henter ut konkrete data om personer fra kildene, og hjelper brukeren med å linke kildene sammen der samme person dukker opp i flere kilder. Det blir laget en entydig ID for hver person slik at man kan se en persons liv fra dåp til død gjennom kildene de har dukket opp i.
I tillegg til dette hjelper løsningen med analyse av dataene ved at den finner familieforhold ved å se på personer fra samme kilde, f. eks. foreldre og barn.

2. Introduksjon

Dette er sluttrapporten til en bacheloroppgave som ble gjennomført i forbindelse med en bachelorgrad i informatikk med spesialisering i informasjonsbehandling våren 2020. Arbeidet ble utført av Nikolai Steen Kjosnes, under veiledning av Tore Mallaug. I denne rapporten vil jeg gå gjennom oppgaven, hvordan jeg valgte å løse den, hva jeg gjorde for å løse den og resultatet av arbeidet.
Oppgavestiller er Oddgeir Fossli, som jobber med å ta en grad i historie med fokus på slektsforskning og flytting. Han har samlet store mengder demografisk data fra kirkebøker og folketellinger, og ønsker at det utvikles en databaseløsning som kan hjelpe med videre analyse av dataene, slik at disse kan brukes i en avhandling.

Oppgaven går dermed ut på å vurdere hvilken databaseløsning som passer problemstillingen best, implementere den og utvikle en løsning som bruker databasen til analyse av dataene. Analysen går mer spesifikt ut på å finne koblinger der samme person dukker opp i flere forskjellige kilder. En utfordring med kildene er at navn ikke har noen standardisert skrivemåte, så fornavnet til samme person kan dukke opp med annerledes skrivemåte i de forskjellige kildene. Derfor må man også utvikle en løsning for navnstandardisering for å kunne finne disse koblingene der samme person går igjen i forskjellige kilder.

Jeg har valgt å strukturere sluttrapporten slik: Først går jeg gjennom datagrunnlaget og forklarer hvilke utfordringer det bringer med seg. Deretter forklarer jeg hvilke løsninger som var under vurdering, hvilke jeg valgte og hvorfor. Så går jeg gjennom hvordan arbeidsprosessen var, for så å forklare resultatet av prosjektarbeidet.
Til slutt drøfter jeg hvordan jeg har opplevd arbeidet, kommer med anbefalinger for videre arbeid, og kommer med en kort konklusjon.

3. Teori

I dette kapittelet vil jeg gå gjennom teori som er relevant for gjennomføringen av prosjektet. Det største fokuset vil være på å gå gjennom datagrunnlaget til oppgaven, og hvilke utfordringer dette fører med seg.

3.1 Forklaring av kildetabellene

Før jeg forklarer utfordringene med datagrunnlaget bør jeg gå gjennom hvilket datagrunnlag denne oppgaven har. Oppgavestiller har 10 forskjellige kildetabeller, hvor hver kildetabell tilsvarer data fra én kilde. Jeg vil nå liste dem etter tabellnavnet de har i databasen, og gi en kort beskrivelse av hva slags kilde de beskriver:

1. **kb_dopte**: Inneholder informasjon om alle som ble registrert døpt i kirkebøkene.
   Hver rad inneholder informasjon om tre personer: barnet, far og mor.
2. **kb_konfirmerte**: Inneholder informasjon om konfirmerte registrert i kirkebøkene.
   Hver rad inneholder informasjon om tre personer: konfirmanten, far og mor.
3. **kb_gifte**: Giftemål som ble registrert i kirkebøkene.
   Hver rad inneholder informasjon om fire personer: brud, brudgom, faren til bruden og faren til brudgommen.
4. **kb_dode**: Inneholder informasjon om personer som ble registrert døde i kirkebøkene.
   Hver rad inneholder informasjon om den døde, og noen rader har også med informasjon om en pårørende.
5. **kb_emigranter**: Inneholder informasjon om emigranter til USA.
   Hver rad inneholder informasjon om én person som emigrerte
6. **no_t1865, no_t1875, no_t1891, no_t1900, no_t1910**: Folketellinger fra året gitt i tabellnavnet. Hver rad inneholder informasjon om én person.

For mer informasjon om tabellene i databasen, se kapittel 4. ER-diagram i kravspesifikasjonen.

3.2 Utfordringene ved datagrunnlaget

En utfordring med denne oppgaven er det omfattende datagrunnlaget. Kildetabellene, spesielt de fra kirkebøkene, inneholder svært mange felt. For eksempel består tabellen for gifte av 64 forskjellige felt. Se Vedlegg 4: Tabelloversikt for eksempler på kolonnenavnene i hver av kildetabellene.
Heldigvis er det mange felt som er mindre interessante i denne oppgaven. For å linke én person til mange kilder trenger vi bare feltene som de kildene har til felles. Eksempler på felter vi ikke er interessert i er dødsdato, dødsårsak, yrke m. fl. Disse er ikke like nyttige til linking siden de kun dukker opp i noen få kilder (f. eks. dødsdato), eller kan endre seg fra kilde til kilde (f. eks. yrke).
Disse feltene kan naturligvis være nyttige til mange andre oppgaver, men for å finne mulige koblinger mellom personene i kildene er de ikke så nyttige. Derfor bør det første vi gjør være å hente ut de mest relevante feltene fra hver av kildene inn i en samlet tabell.
De verdiene som er felles for alle tabellene er følgende:

- Fornavn
- Etternavn
- Farsnavn
- Kjønn
- Fødselsdato (noen kilder har ikke datoen, bare fødelsår)

Ved å hente ut de nevnte verdiene til én tabell har vi forenklet datagrunnlaget, som gjør arbeidet videre lettere.

En annen utfordring med datagrunnlaget er det faktum at disse dataene er fra 1800- og 1900-tallet. På den tiden var det ikke så mange som kunne lese og skrive. Det førte til at når prestene ikke kunne vite hvordan en person skrev navnet sitt da de skrev dem inn i kirkebøkene. Som konsekvens av det har vi nå flere skrivemåter på samme navn, og mange personer dukker opp i tabellene med navnet sitt skrevet på forskjellige måter avhengig av hvilken kilde du ser på. Det gjør det vanskeligere å

finne folk som går igjen i kildene, siden man da ikke kan gjøre et enkelt søk etter eksakte matchende navn.

3.3 Hvordan finner man samme person i flere kilder?

Fødselsnummeret ble først innført i 1964 (Furseth, J. og Ljones, O.), så det er ingen unik ID vi kan bruke til å identifisere en person i flere kilder. For å klare å se sammenhenger mellom kilder må vi vite hvilke felt som ikke endrer seg fra kilde til kilde. Det er tre verdier som ikke endrer seg fra kilde til kilde:

- Fornavn: Fornavnet endrer seg ikke, men skrivemåten kan endre seg. For eksempel kan “Karl Johan” endre seg til “Karl” eller “Johan” avhengig av hva personen brukte resten av livet deres.
- Farsnavn: Farsnavnet er tradisjonelt farens fornavn med “-sen” på enden om barnet er en gutt og “-datter/dtr.” på enden om barnet er en jente. For eksempel vil barnet til Knut hete Knutsen til farsnavn dersom det er en gutt, og Knutsdtr. dersom barnet er en jente. Farsnavnet er det nærmeste vi har våre moderne etternavn, da etternavnet kan endre seg fra kilde til kilde ettersom personene flytter.
- Fødselsdato/fødselsår: Mange kilder mangler selve datoen, men de fleste kildene har fødselsår. Fødselsåret er det feltet som endrer seg minst, siden det ikke er noen alternative skrivemåter man må forholde seg til.

Siden koblingene mellom personer er såpass uklare er det mest gjennomførbart å lage en løsning som finner kilder som kan være samme person, og heller lar sluttbrukeren verifisere om den faktisk er samme person eller ikke.
Når løsningen skal finne kilder som er samme person må den ha en person å starte med, og så finne potensielle personer fra andre kilder som kan være samme som den vi startet med.
Dette kan gjøres ved å lage en løsning som tar inn et navn og returnerer alle gangene det navnet eller en alternativ skrivemåte av det navnet dukker opp. Jeg kommer tilbake til hvordan en slik løsning praktisk kan fungere i kapittel 4. Når vi har en slik metode kan vi bruke den og fødselsåret til å finne et utvalg personer som er

sannsynlig at inneholder den personen vi søker etter. Etter det er det opp til sluttbrukeren å bruke menneskehjernens utmerkede mønstergjenkjenning og logisk tenking til å vurdere linken mellom person og kilde. I tillegg til de felles feltene kan man også bruke de originale kildetabellene til å vurdere kobling mellom person og kilde. For eksempel om man skal linke en kilde fra kildetabellen for døde. Her er det av og til registrert en pårørende, og ofte er det en person i familien. Hvis den pårørende er en forelder til eller gift med personen vi vil linke kilden til kan vi se på de andre kildene den personen har fra før. Hvis personen har vært gift, eller vi har funnet linken til tabellen for døpte, er det en sjanse for at pårørende dukker opp der. Hvis det skjer er det trygt å si at de er samme person.
Dette var et grunnleggende eksempel på hvordan kildedata kan brukes til å vurdere koblinger, og det er mange andre mulige koblinger man kan finne mellom tabellene, som f. eks. samme foreldre i kb_dopte og kb_konfirmerte, eller om personen er gift og har fått et barn kan man se en sammenheng mellom den foreldrene til barnet og personene som giftet seg.

4. Metode

I dette kapittelet vil jeg gå over de metodene jeg valgte å bruke for å løse oppgaven. Jeg vil også gå gjennom andre løsninger som var under vurdering, og hvorfor jeg endte opp med å velge den løsningen jeg valgte.

4.1 Valg av databaseløsning

Den første oppgaven vil være å få dataene fra Excel-arkene inn i en database. Det finnes mange databaseløsninger der ute, og hvilken som er best kommer an på behovet man har.

4.1.1 MySQL

MySQL er et relasjonelt databasesystem, som betyr at det består av tabeller som kobles sammen gjennom relasjoner. Et grunnleggende eksempel på et slikt system kan være om man har en tabell med kunder, og en tabell med bestillinger. Koblingen (relasjonen) mellom de to tabellene vil være det faktum at hver bestilling vil inneholde informasjon om hvilken kunde som utførte bestillingen.
I min mening passer datagrunnlaget her godt til en relasjonell struktur. Vi ønsker å hente ut de viktigste dataene om hver person i kildene, men vi ønsker fortsatt å kunne finne tilbake til resten av dataene vi ikke hentet. Dette er en relasjon mellom tabellen “personerfrakilder” og hver av kildetabellene, og den er svært nyttig å ha tilgang til.
Tabellstrukturen til tabellene i datagrunnlaget er også statisk, som er en fordel i en relasjonsdatabase. Vi kommer ikke til å finne nye parametre for folketellingen i 1865, den er ferdig og dataene den resulterte i vil ikke endre seg.
Det er kun én person som skal bruke denne løsningen, så responstid er ikke en vesentlig faktor. Det gjør at andre databasesystemer som i utgangspunktet håndterer store datamengder raskere vil ha en mindre fordel mot MySQL.
En stor grunn til at valget falt på MySQL som databasesystem for dette prosjektet er at det er MySQL jeg har blitt lært opp til å bruke i flere fag på NTNU. Jeg har derfor mye erfaring i systemet, både MySQL-spesifikke oppgaver og hvordan man

konstruerer en god relasjonell database. Erfaringen jeg har fått gjennom bachelorgraden vil bidra til mindre tid brukt til å lære et nytt system, som igjen vil føre til mer tid brukt til utvikling.

4.1.2 NoSQL og Big Data

NoSQL er et ikke-relasjonelt databasesystem som blir mye brukt i big data.
En fordel med en NoSQL-basert løsning i dette prosjektet vil være at det er lettere å føre inn helt nye kildetabeller, siden NoSQL ikke krever en like statisk tabellstruktur som en relasjonell database krever. Denne fordelen er likevel ikke så stor, siden kirkebøkene og folketellingene omtalt i kapittel 3.1 er omtrent alle tabellene man trenger.
En ulempe med NoSQL er at jeg ikke har noen erfaring med den typen database, så mye tid ville ha gått til å lære meg det i stedet for å utvikle et svar på oppgaven.

4.1.3 Løsning i Excel

Kildedataene er allerede i Excel-ark, og Excel er et svært kraftig verktøy som tillater kobling av data mellom to filer. I teorien er det en mulighet for at man kunne ha laget en løsning som fungerer innad i Excel.
Grunnen til at denne løsningen ikke ble brukt i utviklingen er at en slik løsning vil nå en grense på funksjonalitet. Arbeidet som gjøres her vil ikke kunne videreføres til videre arbeid i samme grad som hvis dataene ble ført inn i en database.

4.1.4 Databasetype brukt i løsningen

Etter vurdering av de nevnte alternativene bestemte jeg meg for å bruke MySQL i løsningen. Kombinasjonen av erfaringen jeg har i MySQL og godene man får av å bruke MySQL gjorde at jeg tok den avgjørelsen.

4.2 Valg av programmeringsspråk

MySQL støttes av en hel rekke språk, f. eks. C, C++, Java, Python, PHP osv., så tilgjengelighet var ikke en faktor i denne avgjørelsen.
Det språket jeg endte opp med var PHP, og det er av flere grunner:

1. Jeg var allerede godt kjent med det fra flere tidligere fag, så jeg trengte ikke å bruke noe unødvendig tid på å lære meg språket eller forstå hvordan man skriver SQL-spørringer i PHP. Det er også svært lett å jobbe med MySQL i PHP.
2. Brukergrensesnitt i PHP er lett å lage, siden det bare er HTML. Med et rammeverk som Bootstrap (https://getbootstrap.com/) er det lett å få et brukergrensesnitt som ser bra ut uten mye arbeid.
3. En løsning utviklet i PHP kjører like bra på et lokalt system som på en dedikert server. Dersom det skulle være ønskelig å sette løsningen på en egen server er dette ikke bare mulig, men også ganske lett å gjøre med en løsning utviklet i PHP.
4. Hvis man kun vil kjøre løsningen lokalt trenger man å installere MySQL. For en bruker som ikke nødvendigvis kan så mye MySQL er det greit å ha et brukergrensesnitt, så phpMyAdmin bør også installeres.
   Når man først må installere MySQL og phpMyAdmin er det like greit å installere dem via en pakke som WampServer. WampServer er en pakke med en lokal Apache-server, MySQL og PHP, kort sagt alt man trenger for å utvikle i PHP. For mer informasjon om WampServer, se vedlegg 3.

4.3 Utviklingsstrategi

Strategien jeg har valgt til utviklingen av produktet i denne oppgaven er en iterativ strategi der jeg, veileder og oppgavestiller blir enige om delmål under prosjektmøtene. Arbeidet går delmål for delmål fra møte til møte. På den måten er strategien mer lik Scrum enn en tradisjonell waterfall-metode, der prosjektets løp planlegges i mye større grad i starten av prosjektet.
Denne strategien er mer hensiktsmessig i et prosjekt som dette, siden det var vanskelig å forutse hvilke deler av prosjektet som ville være mest tidkrevende i starten av prosjektarbeidet. Oppgaven var også så omfattende at det ikke var lett å si hvilke oppgaver man burde fokusere på, og i hvilken rekkefølge.
Derfor fungerte det mye bedre å ta utviklingen delmål for delmål, og heller få gradvis tilbakemelding. Denne metoden minsker også sannsynligheten for at mye arbeid går

tapt om man skulle oppdage at et mål man hadde satt seg ikke var mulig eller hensiktsmessig å oppnå.

4.4 Navnstandardisering

Som nevnt er navn en av de største utfordringene som må løses i denne oppgaven. Kildene er fra en tid der svært få kunne lese og skrive, så om forskjellige prester skrev navnet til samme person på forskjellige måter var det ingen måte å vite det. Derfor kan samme person dukke opp med navnet sitt skrevet på forskjellige måter gjennom kildetabellene. Det gjør at vi må utvikle en løsning som håndterer navn med variasjoner i skrivemåte. De to måtene som ble vurdert og testet i utviklingen av dette produktet er Levenshtein-distanse og en manuell navnstandardisering:

4.4.1 Levenshtein-distanse

Levensthein-distanse (Levenshtein distance, 2020) er et mål som viser "avstanden" mellom to biter tekst som et tall, der tallet er lavere jo likere tekstene er. Tallet regnes ut ved at man teller hvor mange endringer man må utføre på den ene teksten for at den skal bli lik den andre teksten. Slike algoritmer faller under kategorien "edit distance", der Levenshtein-distansen spesifikt tillater å fjerne tegn, å bytte ut tegn og å sette inn tegn.

For eksempel: vi har to personer fra kildene, den ene heter "Kristian" til fornavn, og den andre heter "Christian" til fornavn. Hvis vi regner ut Levenshtein-distansen mellom disse to navnene vil vi ende opp med at avstanden er 2. Det er fordi man må endre "Kristian" to ganger for å ende opp med "Christian":

Operasjon 1: "Kristian" → "Cristian" (Bytter ut "K" med "C")

Operasjon 1: "Cristian" → "Christian" (Setter inn en "h" etter "C")

I løsningen har jeg brukt en implementasjon av levenshtein-distanse som en MySQL-funksjon. Koden til den ble skrevet av Kevin Woblick og lastet opp på hans github i 2017 (Woblick, 2017).

Fordeler

- Fordeler med denne løsningen er at den ikke krever noe ekstra arbeid av brukeren. Så fort de har importert kildene kan de starte å linke.

Ulemper

- Blir mer upålitelig med store omskrivinger av navn.

4.4.2 Manuell navnstandardisering

En annen mulighet for å håndtere navnene er en slags manuell navnstandardisering. Dette er en metode som oppgavestiller hadde tenkt ut, men som jeg også kom opp med under arbeidet med prosjektet. Det går ut på å ha en tabell med oversikt over navn og den "standardiserte" versjonen av navnet, slik:

Variasjon	**Standardisert**
Kristian	Kristian
Christian	Kristian
Ole	Ole
Ola	Ole
Jakob	Jakob
Jacob	Jakob

Systemet ville så ha brukt den standardiserte versjonen når den sammenligner navn, slik at man da kan gjøre direkte søk etter matchende navn.

Navnene i tabellen må riktignok komme fra et sted, så de ustandardiserte navnene kommer fra kildetabellene, mens brukeren er nødt til å fylle inn de et standardisert navn for hvert av de ustandardiserte. Det kan gjøres i et skjema som ser slik ut:

Variasjon	Standardisert
(*Tilfeldig navn fra databasen*)	

Et eksempel på et slikt skjema finnes i filen fornavnStandardisering.php i løsningen.

Fordeler

- Man kan garantere at alle variasjoner av navn er like godt dekket, uansett hvor forskjellige de er fra originalen.

Ulemper

- Krever mer arbeid av brukeren, siden vedkommende må se gjennom alle unike navn i kildetabellene. Med mange kilder importert kan mengden navn man må gjennom bli stor.
  Mengden navn brukeren må gjennom blir enda større når man ser at man også må standardisere farsnavnet, siden både fornavn og farsnavn brukes for å identifisere personer i kildedataene.

4.4.3 Metode brukt i løsningen

I løsningen har jeg implementert begge to til en viss grad, men den eneste som brukes i linking av personer er levenshtein-distanse. Det er mulig å bruke navnstandardiseringen til å standardisere alle fornavn fra alle kildene, men disse standardiserte navnene blir ikke brukt i noen andre prosesser. Dette er fordi under testing av begge løsningene viste levenshtein-distanse seg å være både lettere å implementere og lettere å bruke. Når man tar utgangspunkt i fødselsår og levenshtein-distansen mellom både fornavn og farsnavn blir resultatet et overraskende nøyaktig søk.
Hvis søket fortsatt er for bredt er det også mulig å kun importere data i mindre mengder. Hvis man f. eks. importerer kun data for én kommune vil det være lettere å finne matchende kilder, siden sjansen for at folk holdt seg i samme kommune mesteparten av livet på den tiden var relativt stor.

4.5 Datasikkerhet

Datasikkerhet er et svært viktig aspekt når man jobber med databaser. Ofte skal databasene være koblet til en nettside som er offentlig tilgjengelig, og da er det viktig å hvitvaske input fra brukeren. Hvis man er dårlig på det kan en ondsinnet bruker få uønsket tilgang til data i databasen med SQL-injeksjon.
I dette prosjektet er ikke det så farlig, siden den eneste personen som skal bruke løsningen også har direkte tilgang til databasen gjennom phpMyAdmin. Hvis dette skulle blitt en offentlig tjeneste måtte man ha gått gjennom løsningen og hvitvasket input fra brukeren alle steder der det skjer. Jeg har altså latt være å gjøre det, da det ikke ble vurdert nødvendig i kontekst av oppgaven.

5. Prosess

5.1 Input fra oppgavestiller

I tillegg til at oppgavestiller var til stede på flere møter for å gi sin input arrangerte jeg også et møte nærmere slutten av utviklingen der han kunne se over løsningen og komme med ønsker og tilbakemelding. De punktene han kom med på det møtet, som jeg senere implementerte, er:

- Standardisér tabellnavnene. Tidligere i utviklingsprosessen het tabellen for døde bare “dode”, men for å gjøre forskjellen mellom kilder fra kirkebøker og folketellinger tydeligere kan man legge på en prefiks, slik at “dode” blir til “kb_dode”. Folketellingene gikk fra “folketelling1875” til “no_f1875”. Denne prefiksen gjør det lettere å implementere det neste kravet:
- Oversikt over familie i folketellinger. Først litt bakgrunnsinformasjon: Kildetabellene til folketellingene har tre felt som heter “Kommune”, “Tellekrets” og “Lopenr”. Disse tre feltene forteller noe om rekkefølgen folk ble telt i, og personer som har matchende verdier i alle tre feltene er som regel i familie med hverandre. Ønsket var derfor å gi brukeren en oversikt over familien under linking, slik at det skal være lettere å finne koblinger mellom personens andre kilder og familien i folketellingen. Det ble gjort mye lettere å innføre med prefiksene, siden man lett kan sjekke om tabellnavnet starter på “no_” eller “kb_”.
- Originalt var det bare en løsning for linking av kilder til personer som var implementert. Det betyr at man kun kan linke én kilde av gangen. Ønsket var å kunne linke flere kilder om gangen til samme person. Derfor ble det laget en funksjon for linking med utgangspunkt i person, i stedet for utgangspunkt i kilder. Den tar dataen om personen du gir den og søker etter ulinkede kilder som matcher den personen. Deretter kan du huke av for de kildene som matcher, og trykke på én knapp for å linke alle.
- Delvis import av data. Det var et ønske fra oppgavestilleren å kunne importere kildene stegvis. Dvs. å importere noen kilder, linke dem, og så importere flere

kilder og linke de nye kildene. Det var ikke mulig før, men etter ønske fra oppgavestiller ble det implementert.

- Det siste punktet er en oversikt over antall personer som har blitt linket mot det totale antallet personer i tabellen “personerfrakilder”. Dette var lett å innføre, og jeg hadde allerede planlagt å gjøre det før møtet.

5.2 Møter

Møter ble holdt hver andre uke eller oftere. På slutten av hvert møte ble neste møte avtalt, med noen unntak der det ikke passet. Møtene ble holdt jevnlig for å sikre god fremgang.
Totalt ble det holdt 12 møter. Det første møtet var 21. januar 2020, og det siste var 18. mai 2020.
De første møtene ble brukt på å diskutere og avgrense oppgaven, og å definere brukerkrav og prosjektmål. De senere møtene gikk ut på nødvendig dokumentasjon, gjennomgang av løsningen og tilbakemelding på arbeid.

5.3 Tidsbruk

Jeg har delt aktivitetene inn i 4 forskjellige oppgaver:

1. Møte: Tid brukt på prosjektmøter med veileder og/eller oppgavestiller.
2. Samle info: Tid brukt på å lese dokumentasjon eller på andre måter tilegne seg kunnskap.
3. Dokumentasjon: Tid brukt på å skrive dokumentasjon. Dokumentasjon betyr all teksten som er i dette dokumentet og de andre vedleggene.
4. Utvikling: Tid brukt på å skape produktet. Dette inkluderer skriving av kode, bugfixing, feilsøking på nett og lignende.

Timeliste er inkludert i .zip-filen som vedlegg 7.

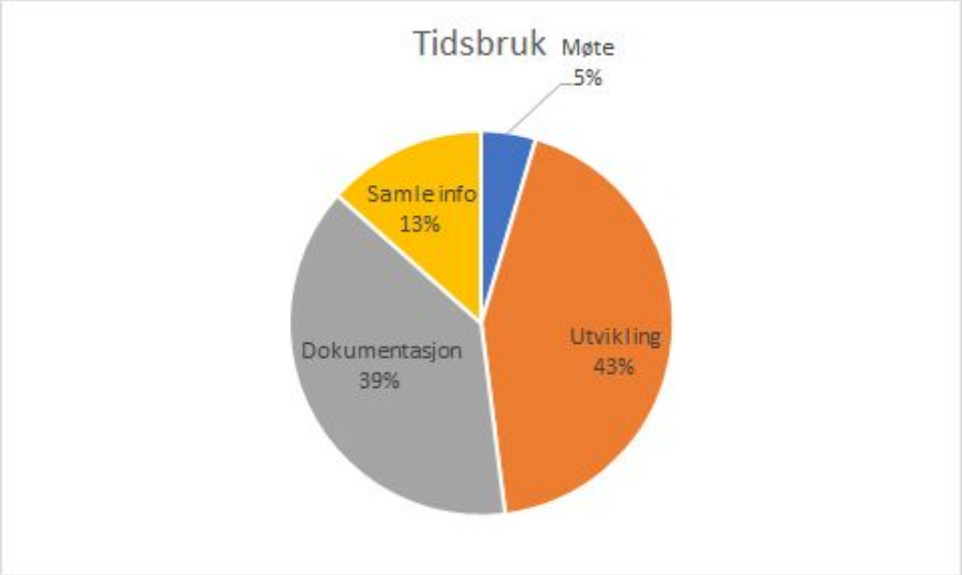


Fig. 1: Tidsbruk fordeling

Aktivitet	**Tid (t)**
Møte	18.5
Utvikling	174.5
Dokumentasjon	155
Samle info	54

6. Resultater

I dette kapittelet vil jeg gå gjennom resultatet av arbeidet, nemlig produktet. Først vil jeg gå gjennom funksjonalitet i løsningen og relatere det til funksjonaliteten som var spesifisert i kravspesifikasjonen. I denne seksjonen vil jeg bruke skjermbilder av løsningen for å illustrere funksjonaliteten, og for å gi leseren et tydeligere bilde av hvordan det er å bruke løsningen.
Deretter vil jeg drøfte hvorvidt løsningen er lett å bruke, og hva jeg har gjort for å sikre dette. Til slutt vil jeg fortelle litt om hva jeg har gjort for skalerbarheten, og å nevne litt om hvordan utviklingen har tatt hensyn til videre arbeid.

6.1 Funksjonalitet i løsningen

Forsiden linker til de fleste funksjonene, så det gir mest mening å begynne her:

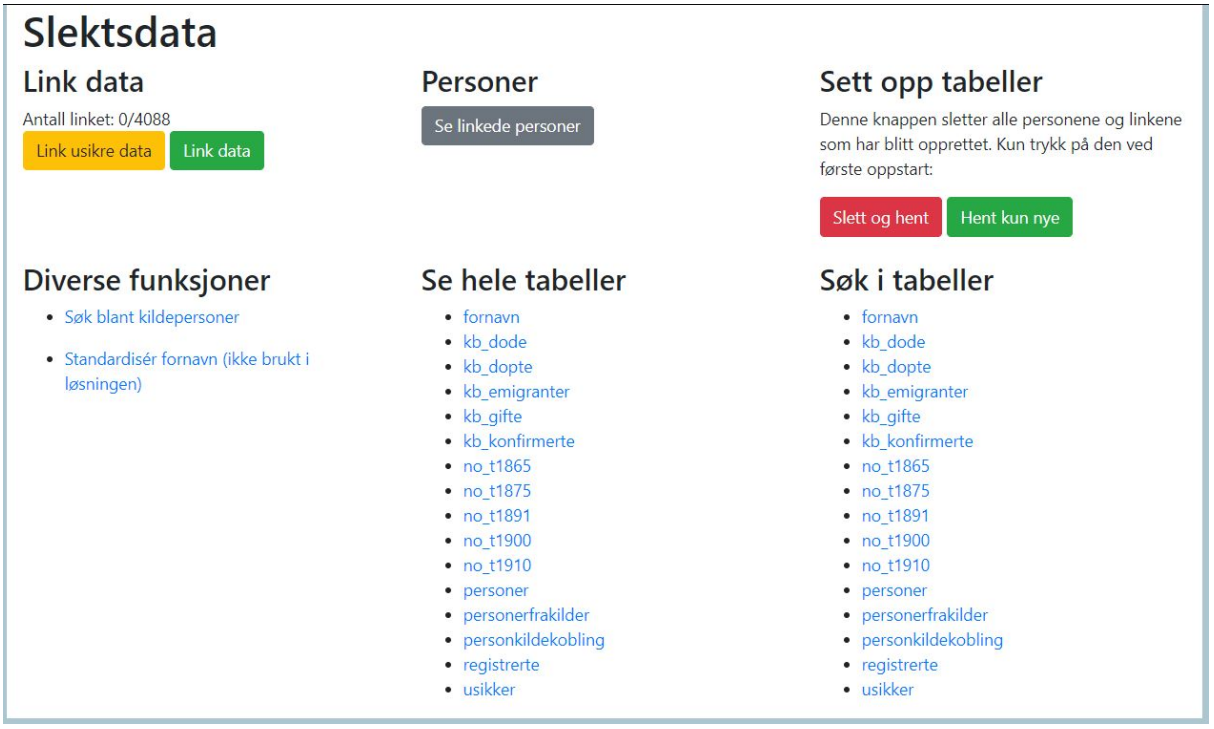


Fig. 2: Forside

Forsiden er delt inn i 6 deler, slik vist på figuren. Øverst til venstre er knapper for linking av data med utgangspunkt i kilder. Den gule knappen lar brukeren linke data de tidligere har hoppet over og markert som usikre, mens den grønne lar brukeren

linke data de ikke har sett på før.
Til høyre for den seksjonen er en knapp som åpner listen over linkede personer. Til høyre der igjen er knappene for import av data. Disse knappene fyller personerfrakilder med personer fra kilder.
Nederst til venstre er diverse funksjoner som brukeren kan få bruk for. Den øverste er et søk i tabellen personerfrakilder som bruker samme kriteriene som funksjonene for linking av kilder. Den andre er den ubrukte manuelle navnstandardiseringen.
Nederst i midten er en oversikt over alle tabellene i databasen. Når man trykker på en av dem blir man vist hele tabellen på en ny side. Det er så det skal være lettere for brukeren å se nøyaktig hva som ligger i databasen uten å måtte gå inn på phpMyAdmin.
Nederst til høyre er også en oversikt over alle tabellene, men disse lenkene lar brukeren søke etter rader i tabellene som matcher de kriteriene brukeren gir den. Den søker kun etter de verdiene brukeren gir dem, men har også en mulighet for å søke etter tomme felt hvis brukeren skriver “-1” i det aktuelle feltet.

6.1.1 Automatisk henting av personer fra kildetabellene

Den første funksjonen en bruker vil benytte seg av er sannsynligvis henting av personer fra kildetabellen. Ved å trykke på knappen på hovedsiden markert “Slett og hent” kan brukeren få serveren til å gå gjennom alle kildetabellene og hente ut data om hver person som nevnes. Dataene blir samlet i én tabell, “personerfrakilder”. I tillegg blir tabellene “personer”, “personerfrakilder”, “usikker”, “registrerte” og “personkildekobling” tømt for å gjøre rom for linking av ny kildedata.
Når serveren henter ut personene fra kildetabellene blir alle radene som ble hentet lagt inn i en oversikt kalt “registrerte”. Det lar oss holde styr på hvilke kilder som er importert, slik at vi kan legge til kilder i databasen og kun importere de nye kildene. For å importere kun nye kilder trykker man på knappen markert “Hent kun nye”. Dette ble implementert etter ønske fra oppgavestiller, se kap. 5.1 - Input fra oppgavestiller.

6.1.2 Linke kilde til person

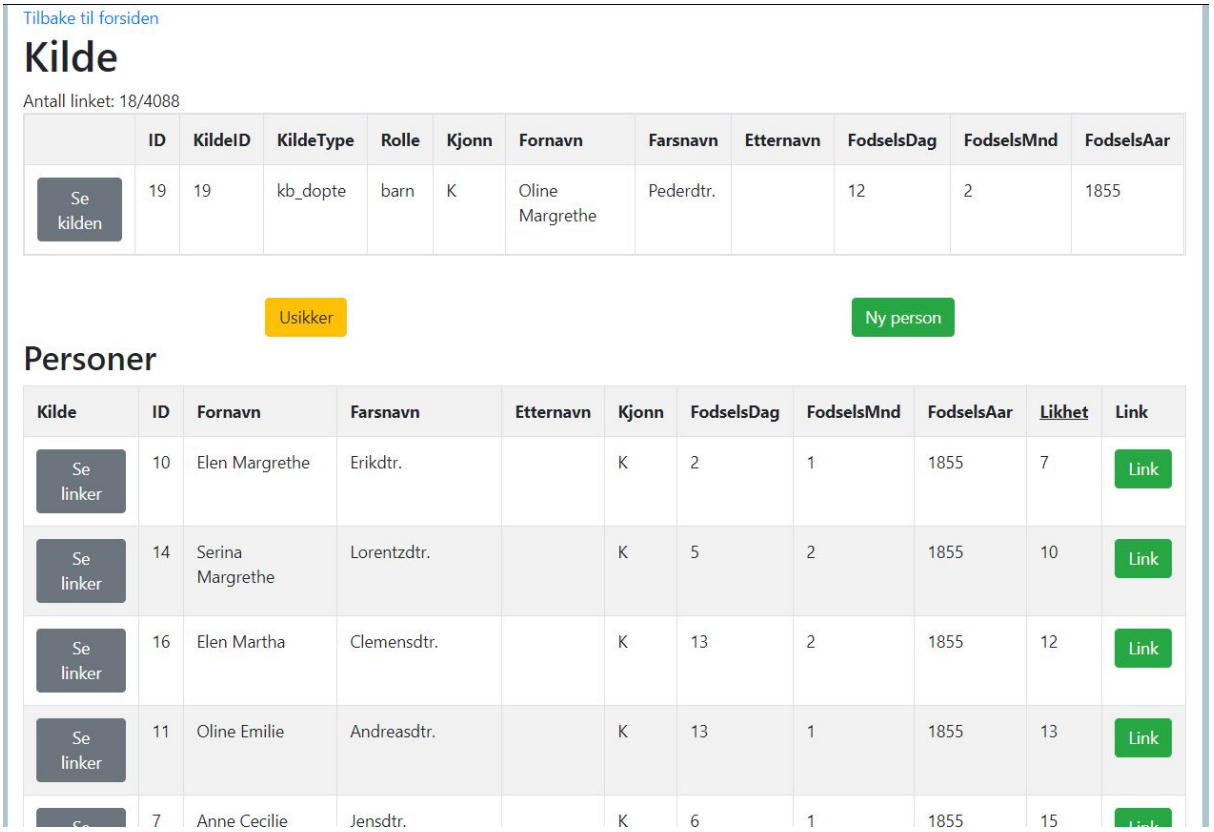


Fig. 3: Link kilde til person

Løsningen henter automatisk en kilde (fra "personerfrakilder") som ikke har blitt linket, og søker etter personer fra tabellen "personer" som matcher personen i den kilden. Som nevnt i kapittel 3 må vi ta utgangspunkt i fødselsår, fornavn og farsnavn. Det første den gjør er å søke etter personer med matchende fødselsår eller manglende fødselsår. Deretter sorterer den disse personene etter summen av levenshtein-distansen mellom fornavnet til kilden og fornavnet til personen, og levenshtein-distansen mellom farsnavnet til kilden og farsnavnet til personen.
En omskriving av SQL-spørringen til pseudokode for forståelighet vil se slik ut:

Vi søker etter personer som matcher denne kilden:

Fornavn: Jonas | Farsnavn: Jensen | Fødselsår: 1855

Gi meg alle personer fra tabellen "personer" hvor fødselsåret deres = 1855, sorter etter levenshtein(fornavnet deres og "Jonas") + levenshtein(farsnavnet deres og "Jensen")

6.1.3 Linke person til kilde

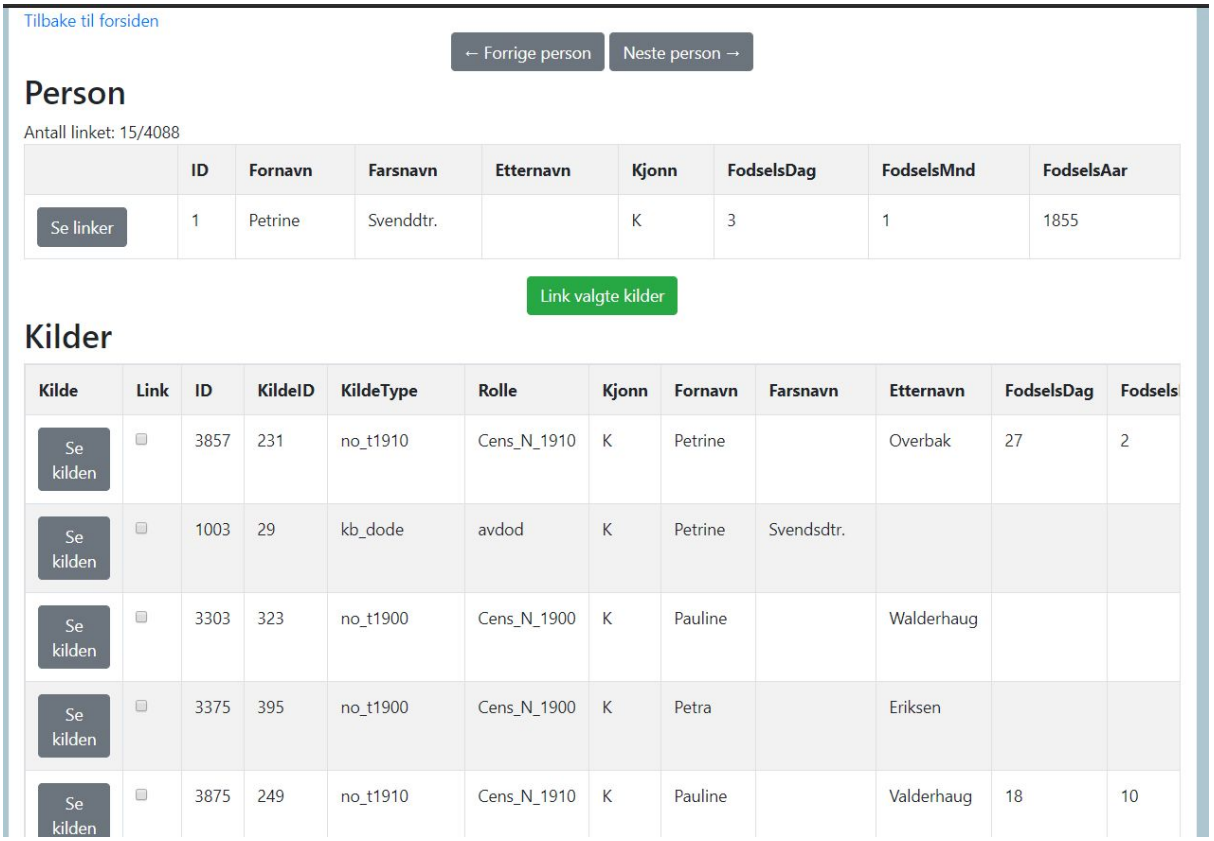


Fig. 4: Linke person til kilde

Denne funksjonen gjør effektivt det motsatte av den forrige funksjonen. Brukeren gir serveren en person, så søker serveren etter kilder som matcher den personen, og som ikke allerede har blitt linket. Øverst i figuren er personen brukeren har valgt, og nederst i figuren er en liste over kildene som matcher navnet til personen. Brukeren kan huke av boksene i feltet “Link” til de kildene som inneholder den personen, og trykke “Link valgte kilder” for å linke alle de markerte kildene på én gang.

6.1.4 Personoversikt

Tilbake til forsiden

Personer

	ID	Fornavn	Farsnavn	Etternavn	Kjonn	FodselsDag	FodselsMnd	FodselsAar	
Se linker	1	Petrine	Svenddtr.		K	3	1	1855	Link kilder
Se linker	2	Mikkael (død)	Pettersen		M	28	1	1855	Link kilder
Se linker	3	John	Torgersen		M	9	2	1855	Link kilder
Se linker	4	Mortinus Severin	Sørensen		M	22	2	1855	Link kilder
Se linker	5	Jørgen Martin	Johnsen		M	28	1	1855	Link kilder
Se linker	6	Anne Kristine	Jacob Losius???dtr.		K	22	1	1855	Link kilder
Se linker	7	Anne Cecilie	Jensdtr.		K	6	1	1855	Link kilder

Fig. 5: Liste over personer

Figuren over viser oversikten over entydige personer. Det er disse som er resultatet av linkingen. Denne listen vises gjennom knappen øverst i midten på forsiden. Herfra har man to muligheter. Man kan trykke på en av knappene til høyre, som tar deg til siden nevnt i kapittel 6.1.3, eller så kan man trykke på knappen til venstre, som tar deg til denne siden:

Tilbake til forsiden

Person

ID	Fornavn	Farsnavn	Etternavn	Kjonn	FodselsDag	FodselsMnd	FodselsAar
1	Petrine	Svenddtr.		K	3	1	1855

Kilder

	ID	KildeID	KildeType	Rolle	Kjonn	Fornavn	Farsnavn	Etternavn	FodselsDag	FodselsMnd	FodselsAar
Se kilden	1	1	kb_dopte	barn	K	Petrine	Svenddtr.		3	1	1855

Relaterte personer

Ingen personer linket til relaterte kilder. Se Relaterte kilder under.

Relaterte kilder

	ID	KildeID	KildeType	Rolle	Kjonn	Fornavn	Farsnavn	Etternavn	FodselsDag	FodselsMnd	FodselsAar
Se kilden	144	1	kb_dopte	mor	K	Bergithe	Jacobsdtr.				
Se kilden	287	1	kb_dopte	far	M	Svend	Pedersen	Werdahl			

Fig. 6: Personside

Hver person har en personside som viser en oversikt over kildene som er linket til dem, i tillegg til relaterte personer og relaterte kilder. De relaterte kildene er kilder som kommer fra samme rad som en av kildene som er linket til personen. F. eks. ser vi at personen på bildet har blitt linket med KildeID = 1 i kb_dopte, dvs. rad 1 i tabellen for døpte. Fra den raden hentet vi to andre personer, nemlig foreldrene. Disse vil også ha KildeID = 1, så disse dukker opp som relaterte kilder. Relaterte personer er personer som er linket til disse kildene. Foreldrene til denne personen er ikke linket enda, så de dukker ikke opp her.

6.1.5 Dynamisk visning av tabeller og søk i tabeller


Tilbake til forsiden

Viser tabell: no_t1865

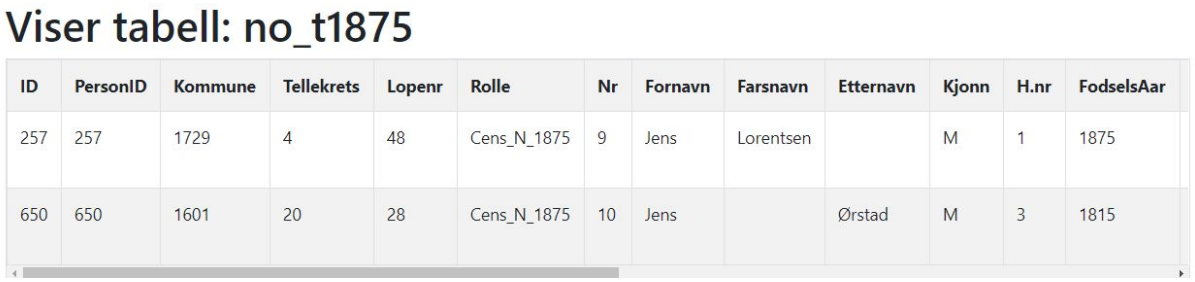
ID	PersonID	Kommune	Tellekrets	Lopenr	Rolle	Nr	Fornavn	Farsnavn	Etternavn	Kjonn	H.nr
1	1	1729	1	20	Cens_N_1865	1	Johannes	Petersen		M	1
2	2	1729	1	20	Cens_N_1865	2	Anne Martha	Pedersdtr.		K	1
3	3	1729	1	20	Cens_N_1865	3	Peter Andreas	Johanessen		M	1
4	4	1729	1	20	Cens_N_1865	4	Iver	Johanessen		M	1
5	5	1729	1	20	Cens_N_1865	5	Iver	Thoresen		M	1
6	6	1729	1	20	Cens_N_1865	6	Ingeborg Anna	Iversdtr.		K	1
7	7	1729	1	20	Cens_N_1865	7	Elen Anna	Hansdtr.		K	1
8	8	1729	1	20	Cens_N_1865	8	Jocumine	Petersdtr.		K	1

Fig. 7: Dynamisk visning av tabeller

Løsningen lar også brukeren se alle dataene som er lagret i databasen ved å dynamisk hente alle tabellene fra databasen. Figuren over viser folketellingen fra 1865.

Fig. 8: Dynamisk søk i tabeller

På samme måte som løsningen lar brukeren se alle tabellene i databasen lar den også brukeren søke i alle tabellene. Den genererer automatisk et skjema med en tekstboks for hvert felt. Kun de feltene som er fylt inn vil bli brukt til å søke. Dersom brukeren ønsker å søke etter et tomt felt kan de skrive “-1” i det aktuelle feltet.



ID	PersonID	Kommune	Tellekrets	Lopenr	Rolle	Nr	Fornavn	Farsnavn	Etternavn	Kjonn	H.nr	FodselsAar
257	257	1729	4	48	Cens_N_1875	9	Jens	Lorentsen		M	1	1875
650	650	1601	20	28	Cens_N_1875	10	Jens		Ørstad	M	3	1815

Fig. 9: Søkeresultat, søk etter “Jens”

Figuren over viser søkeresultat i folketellingen fra 1875 etter personer med fornavnet “Jens”.

ID	PersonID	Kommune	Tellekrets	Lopenr	Rolle	Nr	Fornavn	Farsnavn	Etternavn	Kjonn	H.nr	FodselsA
7	7	1729	1	53	Cens_N_1875	1	Haagen		Bang	M	1	1823
8	8	1729	1	53	Cens_N_1875	2	Sara		Bang	K	1	1823
9	9	1729	1	53	Cens_N_1875	3	Herman Sigvart		Bang	M	1	1855
10	10	1729	1	53	Cens_N_1875	4	Albertine		Bang	K	1	1853
11	11	1729	1	53	Cens_N_1875	5	Eline		Johnson	K	1	1871
67	67	1729	1	88	Cens_N_1875	2	Elen Louise		Muus	K	1	1816

Fig. 10: Søkeresultat, søk etter tomt farsnavn

Figuren over viser søkeresultatet dersom en skriver “-1” i feltet for farsnavn. Det resulterer i et søk etter rader med tomt farsnavn.

6.1.6 Navnstandardisering

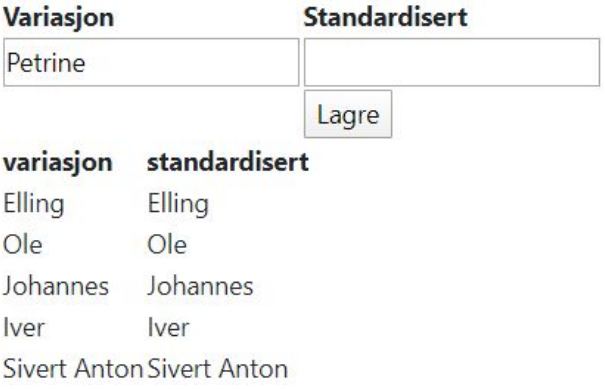


Fig. 11: Manuell navnstandardisering

Denne siden inneholder den manuelle navnstandardiseringen. Denne løsningen blir ikke brukt til linking, men den henter fortsatt fornavn fra tabellen “personerfrakilder”. Grunnen til at jeg lot den være med i den endelige løsningen er at den kan være en start for videre arbeid. Hvis dette grunnlaget skal brukes for en ny bacheloroppgave et senere år kan det være mer effektivt for en ny student å ikke starte fra scratch.

6.2 ER-diagram

Entity Relationship diagram (ER-diagram) viser forholdet mellom entitetene (tabellene) i databasen. ER-diagrammet for denne løsningen er inkludert på neste side, men en selvstendig PDF er også inkludert i .zip-filen. En mer detaljert forklaring av diagrammet finnes i siste kapittelet i kravspesifikasjonen (kapittel 3, vedlegg 2).

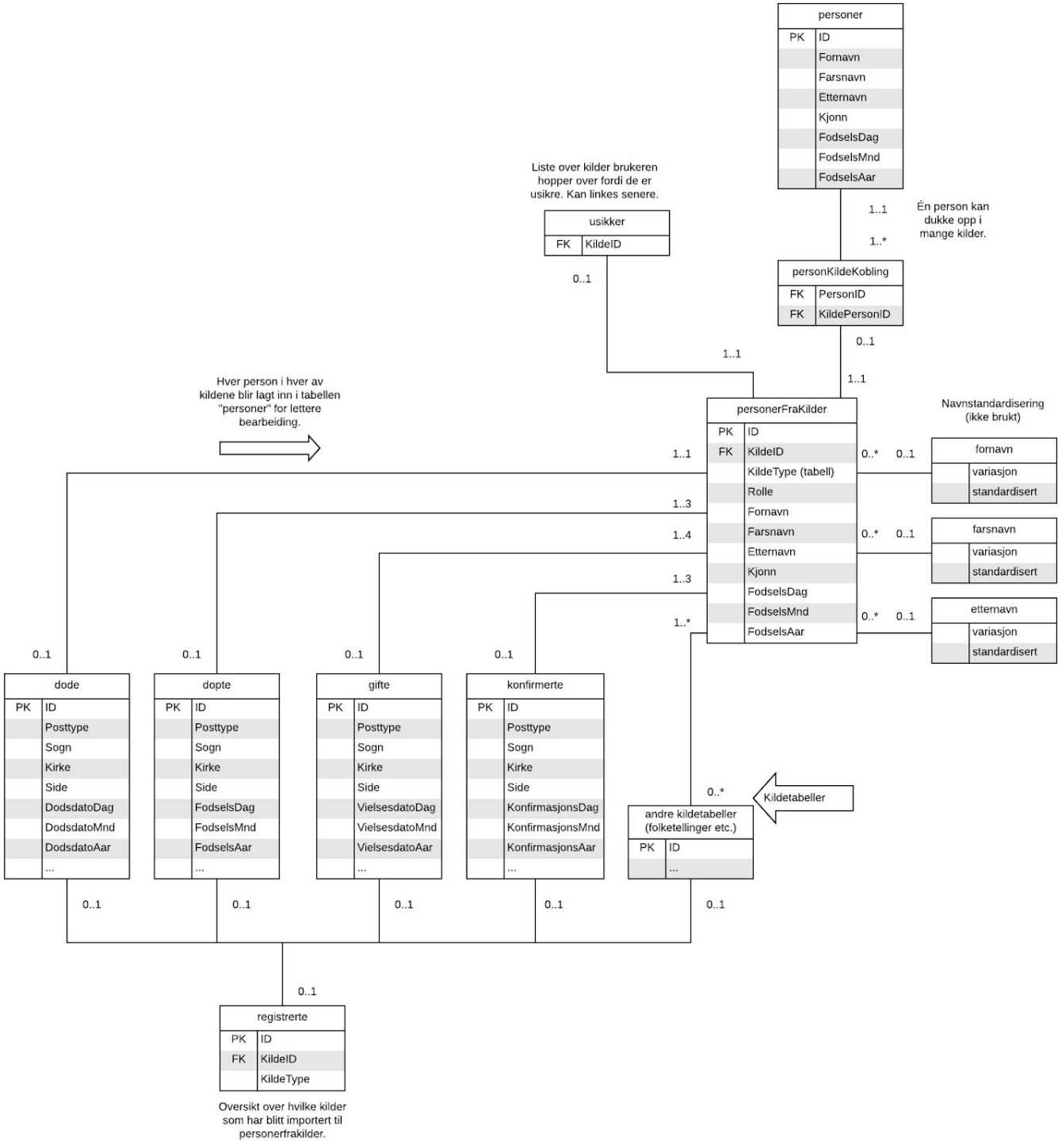


Fig 12. ER-Diagram

6.3 Hvor lett er løsningen å bruke?

Vedlegg 3 er en brukerveiledning som er skrevet med det formål å hjelpe oppgavestiller med å sette opp løsningen på sin egen maskin. Den forklarer oppsett av lokal server, eksport av dataene fra Excel til MySQL og instrukser på hvordan man legger til nye kildetabeller.

Selve løsningen har blitt programmert med brukervennlighet i tankene. Knappenes funksjon er markert med farger, der grønne knapper utfører en handling (f. eks. linke data), røde knapper sletter data og ber brukeren om bekreftelse før noe blir slettet, og grå knapper leder til en ny side (f. eks. se kilden en person kommer fra). De to gule knappene omhandler funksjonen der man kan hoppe over en person hvis man er usikker.

6.4 Skalerbarhet og videre arbeid

Løsningen er laget med videre arbeid i tankene, da den originale oppgaven er svært omfattende. Det har vært viktig å legge opp til at meg selv eller andre skal kunne jobbe videre med produktet, siden det har blitt diskutert på flere møter at denne problemstillingen kan tas opp igjen som en senere bacheloroppgave.
Med tanke på videre arbeid og skalerbarhet har jeg valgt å hardkode så lite som mulig. Løsningen henter informasjon fra databasen der det er praktisk, som f. eks. løsningen for å vise tabellene i nettleseren eller løsningen for søk i tabeller. De to unntakene der jeg var nødt til å hardkode er:

1. Tilkoblingen til databasen, som er satt til localhost siden løsningen er tenkt å kjøres på en lokal server.
2. Koden som henter ut personer fra kildetabellene, siden de forskjellige tabellene har varierende kolonnenavn på de relevante feltene fordi de kan ha flere enn én person per rad. F. eks. heter kolonnene som inneholder navnet til barnet i kilden “no_dopte”: “Fornavn”, “Farsnavn” og “Etternavn”, mens farens navn blir lagret under “FornavnFar”, “FarsnavnFar” og “EtternavnFar”.

Lite hardkoding fører til at løsningen blir mer dynamisk, og håndterer endring i langt større grad. For eksempel er det lett for brukeren å legge en helt ny tabell inn i systemet, det eneste de trenger å endre er 3 linjer kode i hentFraKilder.php (se vedlegg 3 - brukerveiledning for hvordan dette utføres).

6.5 Prosjektmål

I dette delkapittelet vil jeg gå gjennom de målene som ble satt i forstudierapporten, og se hvorvidt de ble nådd under dette prosjektet.

6.5.1 Effektmål

- Raskere og enklere søk i data.
    - Dette målet er definitivt nådd. Løsningen som henter alle personene fra kildene og lagrer dem i én tabell, sammen med løsningen som tillater søk etter kildepersoner med levenshtein-avstand lar brukeren søke i dataene både raskere og enklere.

6.5.2 Resultatmål

- Lage en løsning der man kan importere rådataene fra Excel til en relasjonsdatabase for videre behandling.
    - Brukerveiledningen forklarer i detalj hvordan en slik operasjon utføres.
- Kunne analysere og arbeide med dataene uten å endre dem. Dette prosjektet omhandler historiske data, så det er viktig å beholde integriteten til dataene av historiske grunner.
    - Kildetabellene endres ikke. Det er en av fordelene med å hente ut personene til en felles tabell, man kan endre den nye tabellen uten å endre kildetabellene.
- Utvikle et system i databasen for standardisering av fornavn, gårdsnavn/etternavn og patrikonnavn.
    - Dette punktet ble skrevet under antakelse av at standardisering var den eneste måten å finne samme person i flere kilder. Det viste seg at

det fantes andre måter (levenshtein-distanse), men likevel har dette målet blitt nådd.

- Utvikle en løsning som kan hente data fra databasen og skrive ut et personkort i henhold til gitt mal.
    - Dette målet har delvis blitt nådd. Personkort skal inneholde en rekke verdier som forteller om personen, f. eks. fødselsdato, navn, dødsdato m. fl. Den har også et felt for kilder som personen dukker opp i. Personsiden i denne løsningen dekker en del av disse kriteriene, og resten er anbefalt som videre arbeid.
- Utvikle en løsning som kan hente data fra databasen og skrive ut et familiekort i henhold til gitt mal.
    - Dette målet har ikke blitt nådd, men løsningen legger opp til at dette skal være mulig å utføre som videre arbeid.

6.5.3 Prosessmål

- Utføre en vellykket bacheloroppgave.
    - Personlig mener jeg at dette er en vellykket bacheloroppgave.
- Få mer erfaring i arbeid med databaser og behandling av data.
    - Dette prosjektet har gitt meg mye mer praktisk erfaring i arbeid med både MySQL og PHP.
- Bli mer erfaren i utføringen av et større prosjekt.
    - Dette prosjektet er det største prosjektet jeg har gjennomført. Jeg har definitivt lært mye av gjennomføringen.
- Oppdragsgiver får en bedre forståelse for bruk av databaser.
    - Under møter har jeg, veileder og oppdragsgiver drøftet hvordan denne oppgaven kan løses i en relasjonell database. Jeg vil tro at disse diskusjonene har bidratt til en større forståelse av bruksområdene og begrensningene av databaser hos oppdragsgiver.

7. Refleksjon

I dette kapittelet vil jeg reflektere over min egen innsats og hvordan jeg føler arbeidet har gått. Jeg vil også drøfte mine meninger om hvordan produktet endte opp. Dette er ment som å fungere som et slags refleksjonsdokument, men jeg har valgt å inkludere det i sluttrapporten siden jeg er den eneste som har jobbet på dette prosjektet.

7.1 Hvordan gikk arbeidet?

I starten var det utfordrende å se for seg hvordan sluttproduktet kom til å se ut, ettersom oppgaven var svært omfattende. Etter møter med oppgavestiller og veileder ble oppgaven tydeligere, og arbeidet begynte å gå bedre.
Som de fleste andre i samfunnet på denne tiden ble jeg påvirket av effekten til koronaviruset. Hele Norge stengte litt mer enn halvveis inn i arbeidet, og det var en periode der det var utfordrende å drive prosjektet fremover. Å drive prosjektet fremover var også en generell utfordring i hele prosjektets løp, da dette både var et prosjekt jeg måtte drive fremover på egen hånd, og det største prosjektet jeg har utført. Jeg føler jeg har lært mye om det jeg har jobbet med, men også mye om meg selv og hvordan jeg jobber best. Jeg er glad jeg gjennomførte dette prosjektet.

7.2 Hvordan ble produktet?

Jeg er svært fornøyd med produktet. Det ble tidlig nevnt at dette er en oppgave som er mulig å stille som bacheloroppgave igjen ved en senere anledning, med det arbeidet jeg gjør nå som utgangspunkt. Jeg føler produktet nå er en god start som setter opp til svært mange forskjellige oppgaver. Med personene linket til kildene kan man bygge inn all slags videre funksjonalitet, som jeg vil diskutere i neste kapittel.

8. Videre arbeid

Oppgaven er, som nevnt, svært omfattende. Det gjør at det er mange retninger man kan gå for å videreutvikle løsningen. I dette kapittelet vil jeg gå gjennom de punktene jeg mener er de mest relevante for videre arbeid, og gi noen idéer som har kommet opp under arbeidet for hvordan man kan starte å implementere disse nye funksjonene.
Under diskusjoner med oppgavestiller har følgende områder vist seg å være de viktigste punktene for videre arbeid:

8.1 Emigranter og amerikanske kilder

Området oppgavestiller setter mest fokus på er flytting, da spesielt mellom Norge og USA. På 1800-tallet var det svært mange som emigrerte til USA, og disse vil dukke opp i kildene som denne løsningen tar for seg, spesielt i kilden for emigranter.
Etter at de hadde flyttet til USA vil også disse menneskene dukke opp i amerikanske folketellinger fra den tiden. Man kan så spørre seg: Hvorfor tar ikke denne løsningen med de amerikanske kildene på samme måte som de norske? Jeg har tross alt laget et kapittel i brukerveiledningen om hvordan man legger inn nye kilder, og det bevisst gjort slik at den prosessen skal være så enkel som mulig.
Svaret på det spørsmålet er et område jeg har nevnt i detalj tidligere i denne rapporten, nemlig navn. Da folk kom til USA var det mange som valgte å "amerikanisere" navnet sitt. Overgangen fra norsk navn til amerikansk navn er langt større enn variasjonen mellom de norske navnene. Dette gjør at løsningen slik den er nå ikke vil fungere mot de forandrede amerikanske navnene. Det er to måter jeg har sett for meg at man kan utføre et slikt tillegg på:

1. Lage en løsning som standardiserer navn, eller på en annen måte håndterer dem slik at de amerikanske kildene kan passe inn i denne løsningen. Dette vil kreve en utvidelse av løsningen slik den fungerer i dag til å håndtere denne nye typen navn.

2. Lag en løsning lik denne, men som kjører på en separat database for den norske. Dermed kan man ha en amerikansk database for amerikanske kilder, og en norsk database for norske kilder. Koblingspunktet mellom dem vil være de norske oversiktene for emigranter og de amerikanske oversiktene for immigranter. Dette vil kreve at man lager en løsning som kan koble dataene i disse to databasene sammen.

8.2 Mer komplett personside/familieside

Et annet relevant punkt for videre arbeid som jeg vil ta opp i denne rapporten er personside og familieside. Slik løsningen står i dag viser den en oversikt over relaterte kilder og relaterte personer til den personen du ser på. Her kan man se alle personene som er linket til kilder fra samme rad som den personen du ser på. Det gjør at man kan bruke feltet “Rolle” til å finne f. eks. personer med “Rolle = Far” eller “Rolle = Brud”. Dvs. finne mer spesifikt hvem som er foreldre til eller gift med en gitt person. Vedlagt i kildekoden er et tekstdokument med SQL-spørringen som returnerer personer fra personerfrakilder som har felles KildeID som kildene til personen (fra personer) du gir den. Den er delvis ment å brukes til dette området for videre arbeid.
For å finne data som dåpsdato, konfirmasjonsdato og dødsdato kan man se om personen er linket til en kilde med KildeType lik henholdsvis “kb_dopte”, “kb_konfirmerte” og “kb_dode”. På den måten kan man finne mer komplett informasjon som man kan vise på personsidene.
Dette forslaget går dermed ut på å lage en løsning som finner mer data ut ifra disse koblingene og viser dem på en oversiktlig måte.
Siden man kan finne personer som er i familie med hverandre bør man også kunne lage en løsning som viser en familieside for hver familie.

8.3 GPS-koordinater og steder

Kirkebøkene og folketellingene inneholder henholdsvis sogn og kommune, så det bør er mulig å utvikle en løsning som ser gjennom kildene til en person og returnerer en liste med GPS-koordinater over hvor de har blitt registrert. Hvis man får

koordinater for hver person i databasen kan man utføre mange spennende analyser om flytting, f. eks. å lage et kart som viser flytting innenfor en tidsperiode oppgitt av brukeren. Den største utfordringen her vil kanskje være å få tak i en liste over sogn og kommuner med GPS-koordinater, siden løsningen jeg har utviklet lar deg finne sogn og kommune for hver av kildene til hver av personene.

9. Konklusjon

Jeg har utviklet en relasjonell databaseløsning som lar deg importere kilder og linke personene i disse kildene til entydige personer i en egen tabell. Den har også mulighet for videre analyse av dataene ved at den finner familien til hver person gjennom kildene de er linket til.
Jeg har også utviklet en brukerveiledning som instruerer brukeren i oppsett av løsningen, slik at det skal være enkelt å komme i gang.
Det er en god start på en svært omfattende oppgave, og legger bra opp til videre arbeid som kan utføres av andre bachelorstudenter senere. De områdene for videre arbeid jeg anbefaler er: Å integrere den amerikanske delen av kildene, videre arbeid med personside/familieside og å bruke stedsdataene til å beregne GPS-koordinater.
Prosjektmålene fra forstudierapporten ble i stor grad nådd, og jeg vurderer dette som et godt gjennomført prosjekt.
Takk til oppgavestiller Oddgeir Fossli og veileder Tore Mallaug for god hjelp underveis.

10. Vedlegg

Vedleggene ligger på slutten av dette dokumentet der mulig. De av vedleggene det ikke var hensiktsmessig å inkludere i PDFen er inkludert i .zip-filen.
Under følger en oversikt over vedleggene, samt hvor de finnes:

1. Forstudierapport (På slutten av denne PDFen)
2. Kravspesifikasjon (På slutten av denne PDFen)
3. Brukerveiledning (På slutten av denne PDFen)
4. Tabelloversikt (Tabelloversikt.xlsx i .zip-filen)
5. ER-diagram (PDF i .zip-filen)
6. Oversikt over kode (På slutten av denne PDFen)
7. Timeliste (Timeliste.xlsx i .zip-filen)

I tillegg til de nevnte vedleggene vil følgende filer også være inkludert i .zip-filen:

- Kildekode
- .sql-fil for oppsett av database

11. Figurer

12. Referanser

1. Levenshtein distance. (2020) *Wikipedia*. Tilgjengelig fra: https://en.wikipedia.org/wiki/Levenshtein_distance (Hentet 13. mai 2020)
2. MySQL-levenshtein.sql. (2017) *Kevin Woblick*. Tilgjengelig fra: https://gist.github.com/Kovah/df90d336478a47d869b9683766cff718
3. Furseth, J. og Ljones, O. (2015) 50-årsjubilant med behov for oppgradering, *Samfunnsspeilet* 1/2015, s. 23. Tilgjengelig fra https://www.ssb.no/befolkning/artikler-og-publikasjoner/_attachment/224332?_ts=14cb7c21460 (Hentet 15.05.2020)

Vedlegg 1 - Forstudierapport

Innholdsfortegnelse

1. Introduksjon

Dette dokumentet er forstudierapporten til en bacheloroppgave gjennomført i sammenheng med et studium i informatikk med spesialisering i informasjonsbehandling ved NTNU våren 2020. Oppgaven går ut på å utvikle en implementasjon av en database for lagring og analyse av historisk demografisk data. I dette dokumentet vil du finne en beskrivelse av prosjektet, mål for prosjektet, interessentene til prosjektet og retningslinjer og standarder som prosjektet utføres ut ifra.

2. Bakgrunn for prosjektet

Dette prosjektet utføres som en bacheloroppgave i informasjonsbehandling. Oppgavestiller Oddgeir Fossli har lenge arbeidet med å digitalisere demografiske data fra kirkebøker. Gjennom dette arbeidet har han endt opp med store mengder data i form av tabeller i regneark, men for å bruke dataene til videre arbeid trenger han en løsning som er mer tilpasset den typen data han jobber med.

Formålet med dette prosjektet er da å utvikle en løsning som han kan bruke for mer hensiktsmessig lagring og behandling av dataene.

2.1 Beskrivelse av problemer og behov

Oppgavebeskrivelsen går som følger: “Oppgaven blir å opprette en database hvor enkeltindividet blir hovedperson og lenke de forskjellige personhistoriske hendelsene til denne.”.

Oppgaven vil dermed gå ut på:

- Å vurdere hvilke databaseløsninger som er hensiktsmessige.
- Å finne ut av hvordan løsningen skal implementeres.
- Å implementere løsningen og utvikle tilleggsfunksjonalitet.

2.2 Kort om dagens systemer og rutiner

Dagens systemer består av at kildedataene er ført inn i flere Excel-ark. Filene er svært store og hver fil har ingen kobling til de andre filene. Det er dermed ingen prosesser i dag som ligner på de dette prosjektet skal utvikle.

Oddgeir Fossli er bruker og eier av dagens systemer, og han har ansvar for forvaltning og drift.

Problemet med dagens løsning er at store mengder data er lagret utover flere Excel-ark. Det gjør det vanskelig å etablere koblinger mellom dataene i de

forskjellige Excel-arkene. En database vil gjøre dataene lettere å behandle og mer oversiktlige.

3. Prosjektmål

3.1 Effektmål

- Raskere og enklere søk i data.

3.2 Resultatmål

- Lage en løsning der man kan importere rådataene fra Excel til en relasjonsdatabase for videre behandling.
- Kunne analysere og arbeide med dataene uten å endre dem. Dette prosjektet omhandler historiske data, så det er viktig å beholde integriteten til dataene av historiske grunner.
- Utvikle et system i databasen for standardisering av fornavn, gårdsnavn/etternavn og patrikonnavn.
- Utvikle en løsning som kan hente data fra databasen og skrive ut et personkort i henhold til gitt mal.
- Utvikle en løsning som kan hente data fra databasen og skrive ut et familiekort i henhold til gitt mal.

3.3 Prosessmål

Da dette prosjektet er en bacheloroppgave utført av en student er det også noen prosessmål:

- Utføre en vellykket bacheloroppgave.
- Få mer erfaring i arbeid med databaser og behandling av data.
- Bli mer erfaren i utføringen av et større prosjekt.

- Oppdragsgiver får en bedre forståelse for bruk av databaser.

3.4 Prosjektets omfang

Datagrunnlaget er, som tidligere nevnt, svært omfattende. For å gjøre prosjektet mer gjennomførbart har vi avgrenset datamengden til kun kildene fra kommunen Inderøy. I teorien vil ikke dette føre til et dårligere sluttprodukt, siden produktet skal utvikles med skalerbarhet i tankene. Dataen skal altså ikke hardkodes, og løsningen skal fungere med et annet datagrunnlag, gitt at dataen er formatert likt.

Brukerkrav vil bli avklart i første halvdel av prosjektet i samarbeid med oppdragsgiver. Endelige krav til funksjonalitet er ikke gitt ved prosjektstart, men utarbeides interaktivt gjennom prosjektmøter.

3.5 Prosjektets milepæler og hovedaktiviteter

Prosjektet vil bli gjennomført med jevnlige møter mellom student og veileder. Møtene vil, i utgangspunktet, ta sted hver 2. uke. Oppgavestiller vil delta på møtene der det er hensiktsmessig.
På møtene vil prosjektets status bli gjennomgått og planen videre bli vurdert.

4. Interessenter og rammebetingelser

4.1 Interessentanalyse

Interessent	**Suksesskriterier**	**Bidrag til prosjektet**
Ekstern: Oddgeir Fossli (Oppgavestiller)	Arbeidslettelse Et brukbart produkt Et produkt som kan utvikles videre	Testdata Hjelp med beslutninger Kunnskap om demografi og historie
Intern: Tore Mallaug (Veileder)	Riktig dokumentasjon (sluttrapport, timeliste osv.)	Kunnskap, veiledning
Intern: Nikolai Steen Kjosnes (Bachelorstudent)	Et godt gjennomført prosjekt Vellykket produkt og prosjektrapport	Arbeid som driver prosjektet fremover

4.2 Rammebetingelser

Endelig leveringsfrist på produkt og sluttrapport er 20.05.2020 i forbindelse med bacheloroppgaven.

5. Retningslinjer og standarder

5.1 Krav til dokumentasjon

Da dette er en bacheloroppgave vil det kreves en sluttrapport og en timeliste. Dette dokumentet (forstudierapporten) er også en del av kravet til dokumentasjon.

5.2 Krav til kvalitetsgjennomganger

Kvalitetsgjennomganger vil bli gjennomført på møtene. Disse holdes jevnlig slik at kvaliteten sikres .

5.3 Krav til standarder og metoder

Det eneste kravet til standarder og metoder er at tabellene formateres etter standarden “Kyrre”[1]. Dette er ønsket for å holde standardene fra kildedataene, i tillegg til å sikre oversiktlig tabellstruktur.

[1] http://xml.arkivverket.no/artikler/kyrre.pdf

Vedlegg 2: Kravspesifikasjon

Innholdsfortegnelse

1. Introduksjon

Dette dokumentet er skrevet i forbindelse med en bacheloroppgave i informasjonsbehandling ved NTNU. Oppgaven går ut på å lage en relasjonsdatabase som skal brukes til å håndtere data fra folketellinger og kirkebøker fra 1800- og 1900-tallet.

Dette er dokumentet for kravspesifikasjonen til prosjektet. Hensikten til dette dokumentet er å dokumentere hvilke funksjoner produktet skal ha, og hvordan de relaterer til brukerens behov.
Det vil også gå gjennom ER-diagrammet og forklare formålet med hver av tabellene og koblingen mellom dem.

2. Use caser

Use caser for oppstart

Navn	UC-1 – Oppsett av databasen
Mål	Hensikten med dette use caset er å sette opp databasen, og å legge inn kildene fra Excel-arkene til tabellene i databasen. Med dette kan vi videre behandle dataene.
Aktør	Bruker initierer prosessen.
Trigger	Når brukeren starter opp med systemet er tabellene tomme. Det som starter use caset er altså behovet for å føre inn dataen for å kunne bruke systemet.
Pre-betingelse	Ingenting er satt opp. Brukeren har Excel-ark med kildetabeller.
Post-betingelse	Databasen er oppe og kilder er ført inn. Den lokale serveren er satt opp og kan brukes til å kjøre php-filene.
Hovedløp	Dette hovedløpet krever at brukeren må benytte seg av brukerveiledningen, som går i detalj om hvordan denne prosessen utføres.

Navn	UC-2 – Hente personer fra kilder, og klargjør dem til linking
Mål	Hensikten med dette use-caset er å importere alle personene fra alle kildetabellene inn i én tabell slik at de kan brukes til videre arbeid.
Aktør	Bruker initierer prosessen.
Trigger	Kildetabellene er ført inn, men er ikke importert til tabellen “personerfrakilder”. Brukeren trenger at de importeres for å kunne fortsette med arbeidet.
Pre-betingelse	Kildetabellene er ført inn i databasen.
Post-betingelse	Alle personene fra alle kildetabellene er ført inn i én tabell.
Hovedløp	Steg 1: Bruker trykker på knappen markert “Slett og hent”. * Systemet henter automatisk ut data om alle personene og fyller dem inn i én tabell.
Sideløp	Hvis brukeren allerede har importert en del av personene i kildetabellene, men har lagt til mer senere, må de følge dette handlingsforløpet: Steg 1: Bruker trykker på knappen markert “Hent kun nye”. * Systemet henter automatisk ut data om alle personene som ikke allerede er importert og fyller dem inn i én tabell.

Use caser for bruk

Navn	UC-3 – Linke kilder til personer
Mål	Hensikten med dette use caset er å etablere en kobling mellom en person fra kildene og en person fra tabellen “personer”.
Aktør	Bruker initierer prosessen.
Trigger	Brukeren har behov for å linke de kildene de har importert med personer i databasen. Bruker starter use-caset ved å trykke seg inn på siden fra hovedmenyen.
Pre-betingelse	Databasen er satt opp og kildene er importert
Post-betingelse	Tilstanden til system og aktør etter at use caset er utført, både ved hovedløp og sideløp.
Hovedløp	Steg 1: Brukeren trykker seg inn på siden for linking av data fra hovedmenyen. * Systemet finner frem en kilde som ikke er linket, og søker etter personer som matcher den. Steg 2: Brukeren vurderer personene opp mot kilden Steg 3: Brukeren utfører valget ut ifra det de kom fram til av steg 2. Dvs. registrere som ny person, link kilde til person eller hopp over kilden og registrer som usikker.
Spesielle krav	Listen med personer bør matche kilden så godt som mulig, slik at brukeren ikke trenger å lete langt ned i listen.

Navn	UC-4 – Linke personer til kilder
Mål	Hensikten med dette use caset er å etablere en kobling mellom en person fra tabellen “personer” og en eller flere kilder.
Aktør	Bruker initierer prosessen.
Trigger	Brukeren har behov for å linke de kildene de har importert med personer i databasen. Bruker starter use-caset ved å trykke seg inn på siden fra hovedmenyen.
Pre-betingelse	Tabellen “personer” har blitt delvis fylt opp.
Post-betingelse	Personen som brukeren valgte har blitt vurdert opp mot kildene. Mest sannsynlig har noen kilder blitt linket til personen, men det kommer an på brukerens avgjørelse i steg 3.
Hovedløp	Steg 1: Brukeren trykker seg inn på siden for linking av data fra personoversikten. * Systemet finner frem personen som brukeren trykket på i forrige steg, og søker etter kilder som matcher den. Steg 2: Brukeren vurderer kildene opp mot personen. Steg 3: Brukeren huker av alle kildene de mener skal linkes til personen, og trykker deretter på en knapp for å linke de valgte kildene til personen.
Spesielle krav	Listen med kilder bør matche kilden så godt som mulig, slik at brukeren ikke trenger å lete langt ned i listen.

Navn	UC-5 – Finne familie til en person
Mål	Hensikten med dette use caset er å finne oversikt over familien til en gitt person.
Aktør	Bruker initierer prosessen.
Trigger	Brukeren trenger å finne familien til en person. Dette trigges av en ekstern faktor.
Pre-betingelse	Nok personer er linket til kilder til at personen man ønsker å se familie til har kilder der familie nevnes.
Post-betingelse	Ingenting endres i systemet. Brukeren har en liste over kilder/personer som er i familie med personen de ønsket å finne mer informasjon om.
Hovedløp	Steg 1: Brukeren trykker seg inn på listen over personer. Steg 2: Brukeren trykker seg inn på siden til personen de vil se oversikten over. Steg 3: Brukeren får opp oversikt over personer som er i familie med den personen. Her kan de trykke seg inn på sidene for hver av personene, for å se hvem de er i familie med, samt å trykke seg inn på kildene disse personene kommer fra.
Spesielle krav	Nok personer må være linket til at en person som er linket har andre personer i familien deres som også er linket.

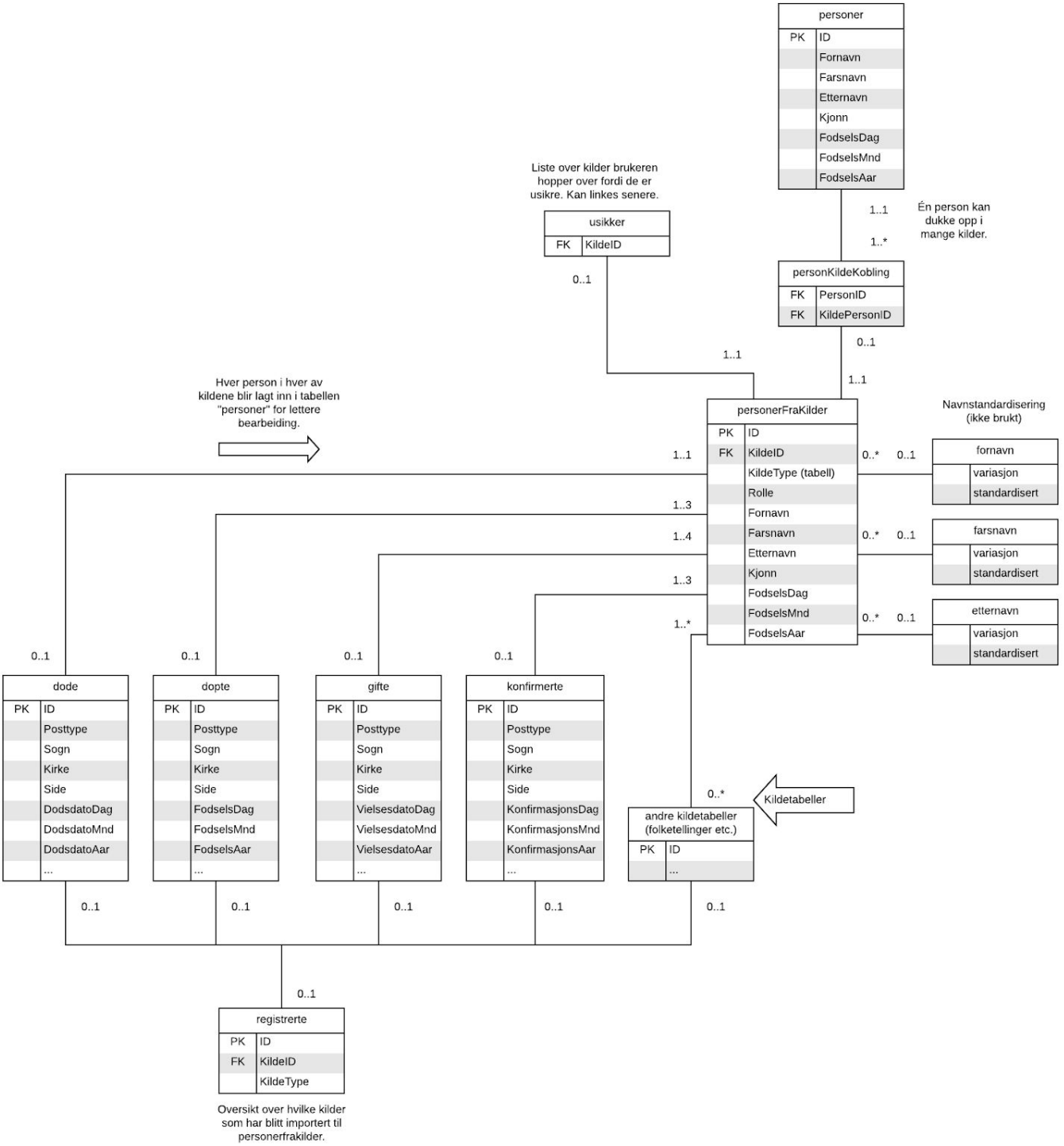
3. ER-diagram

ER-diagrammet er lagt under tabelloversikten. Dersom et større bilde er ønskelig, se vedlagt PDF i .zip-filen for en større versjon av ER-diagrammet.

Tabelloversikt:

- **dode, dopte, gifte, konfirmerte, andre kildetabeller:**
  Disse er alle kildetabellene. Hver rad er én kilde fra en kirkebok eller folketelling. Hvor mange personer man finner på hver rad kommer an på hva slags kilde det er, for eksempel består én rad fra kilden for gifte av fire forskjellige personer: brud, brudgom, brudgomsfar og brudsfar. Folketellingene, derimot, består av kun én person per rad.
- **personerfrakilder:**
  Denne tabellen blir fylt med hver person fra kildetabellene, slik at det skal være lettere å behandle dataene. Man henter kun ut de mest relevante og universale feltene, så ting som dåpsdato blir ikke hentet ut. En historisk person kan dukke opp flere ganger i denne tabellen hvis de dukker opp i flere kilder. F. eks. vil en person som ble registrert i kilden for døde og kilden for døpte dukke opp to ganger, én for hver kilde.
- **personkildekobling:**
  Under linking vil denne fylles opp med koblinger mellom kildepersoner (personerfrakilder) og entydige personer (personer).
- **personer:**
  Når en ny person registreres vil den bli ført inn i denne tabellen. Når den personen etter hvert dukker opp i senere kilder vil kilden kunne linkes til denne entydige personen. Her vil hver person kun dukke opp én gang. Dette er en oversikt over entydige personer. Det er derfor personene i denne tabellen linkes til personene fra personerfrakilder.
- **usikker:**
  Det er ikke alltid det er så tydelig hvem en person skal linkes med. Da kan det lønne seg å hoppe over den, og ta den igjen senere når man har mer info. Derfor har brukeren muligheten til å trykke “usikker” for å hoppe over den kilden de linker for øyeblikket. Kilden blir lagt inn i denne tabellen, som senere blir brukt til å kun linke de kildene man var usikre på.
- **fornavn, farsnavn, etternavn (ikke brukt i løsningen):**
  Samme person kan ha navnet sitt skrevet forskjellig i ulike kilder. Dette er en løsning for standardisering av navnvariasjoner. Mer om navnstandardisering finner du i sluttrapporten.

- **registrerte:**
  Denne tabellen holder styr på hvilke personer fra kildetabellene som har blitt ført inn i personerFraKilder. Når kildepersonene blir ført inn i personerFraKilder tar den bare med de kildene som ikke står i denne listen, og legger deretter de den førte inn inn i denne tabellen. Det er sånn at man kan føre inn data litt og litt, slik at brukeren skal kunne linke kilder fra f. eks. kun 1 kommune av gangen. Hvordan denne tabellen blir brukt er opp til brukerens behov.


personer
PK ID
Fornavn
Farsnavn
Etternavn
Kjonn
FodselsDag
FodselsMnd
FodselsAar
1..1
1..*
Én person kan dukke opp i mange kilder.
Liste over kilder brukeren hopper over fordi de er usikre. Kan linkes senere.
usikker
FK KildeID
0..1
personKildeKobling
FK PersonID
FK KildePersonID
0..1
1..1
1..1
Hver person i hver av kildene blir lagt inn i tabellen "personer" for lettere bearbeiding.
personerFraKilder
PK ID
FK KildeID
KildeType (tabell)
Rolle
Fornavn
Farsnavn
Etternavn
Kjonn
FodselsDag
FodselsMnd
FodselsAar
Navnstandardisering (ikke brukt)
fornavn
variasjon
standardisert
farsnavn
variasjon
standardisert
etternavn
variasjon
standardisert
1..1
1..3
1..4
1..3
1..*
0..*
0..1
0..1
dode
dopte
gifte
konfirmerte
PK ID
Posttype
Sogn
Kirke
Side
DodsdatoDag
DodsdatoMnd
DodsdatoAar
FodselsDag
FodselsMnd
FodselsAar
VielsesdatoDag
VielsesdatoMnd
VielsesdatoAar
KonfirmasjonsDag
KonfirmasjonsMnd
KonfirmasjonsAar
...
andre kildetabeller (folketellinger etc.)
PK ID
...
0..*
Kildetabeller
registrerte
PK ID
FK KildeID
KildeType
Oversikt over hvilke kilder som har blitt importert til personerfrakilder.

Vedlegg 3: Brukerveiledning

Innholdsfortegnelse

Introduksjon

Dette dokumentet er ment å gi brukeren en oversiktlig guide på hvordan man setter opp løsningen. Den er delt i tre deler:

- Hvordan man setter opp den lokale serveren og databasen.
- Hvordan man importerer data fra Excel til databasen.
- Hvordan man legger til nye tabeller.

I tillegg følger det en liste over lenker på slutten med de programmene man må laste ned.

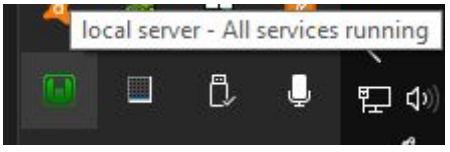
Oppsett

Lokal server

Løsningen er skrevet i språket php, som betyr at nettleseren ikke klarer å åpne filene av seg selv. Php består av instrukser som en server utfører før den viser nettsiden til brukeren, så for å kunne bruke php-filene må vi ha en server.
Heldigvis finnes det mange alternativer for programmer som kan kjøre servere lokalt på PCen, så vi slipper å sette opp en ekstern server eller i verste fall betale for en. Det serverprogrammet jeg valgte, og anbefaler, er Wampserver. Grunnen til at jeg anbefaler den er todelt: Den første grunnen er at den kommer med MySQL database og et grafisk brukergrensesnitt inkludert, som gjør både arbeid og bruk av databaser lettere.

Oppsett av databasen

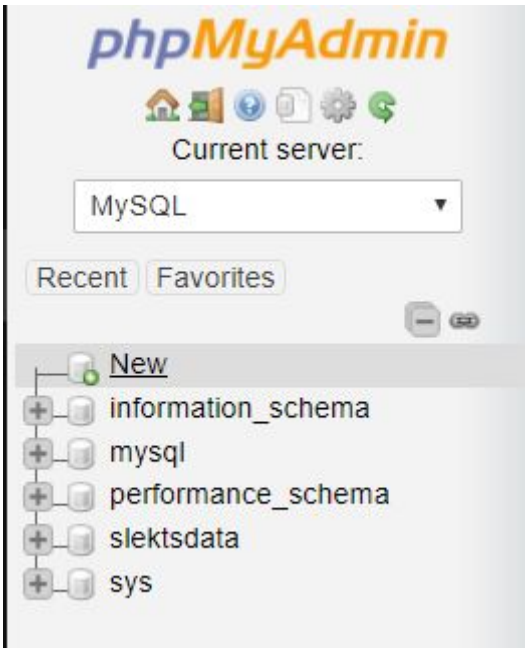
Når du har satt opp Wampserver må du opprette databasen. Dette kan enkelt gjøres ved å gå inn i phpMyAdmin, som skal ligge på lenken http://localhost/phpmyadmin hvis du har Wampserver kjørende. Hvis den lenken ikke fungerer, sjekk om Wampserver har et grønt ikon i menyen i oppgavelinjen, slik vist på skjermbildet:



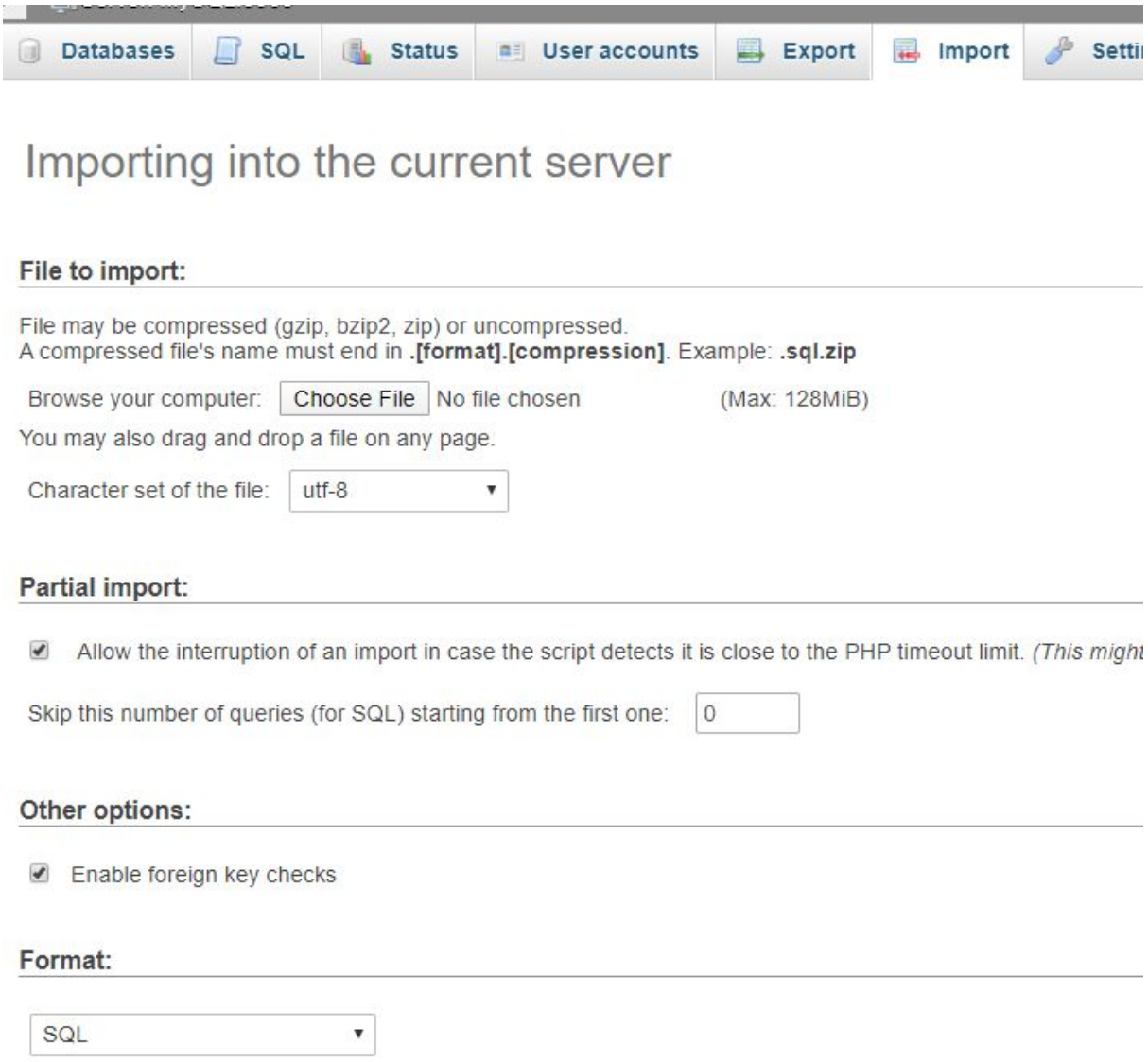
Hvis alt er riktig satt opp skal du bli tatt til en side der du kan logge inn, slik vist på skjermbildet:

Her har Wampserver satt brukernavn og passord til henholdsvis “root” og ingenting. Dersom du fyller inn akkurat slik som på bildet skal det fungere.

Nå må du opprette en ny database. Det gjør du ved å trykke på knappen markert “New” i menyen til venstre, slik vist på skjermbildet:



Deretter navigerer du deg inn på menyen for “import”, det skal være en knapp merket det i linjen øverst på skjermen.



Her trykker du på “Choose File”, og legger inn den vedlagte filen som heter “database.MySQL”. Etter det trenger du ikke å endre noe med i den menyen, så du kan bare trykke på knappen merket “Go” nederst på skjermen.

Med det skal databasen settes opp for deg, og om alt går som det skal vil du ha en database full av tomme tabeller.

Neste steg blir da å fylle dem med dataene dine.

Import av data

Når det kommer til import av data har jeg funnet to måter å gjøre det på. Den ene er å bruke phpMyAdmin sin integrerte importfunksjon, og den andre er å bruke MySQL sin utvidelse til Excel. Begge har noen fordeler og ulemper, så du får gjøre din egen vurdering om hvilken som er best.

phpMyAdmin og OpenDocument Spreadsheets

Prosess

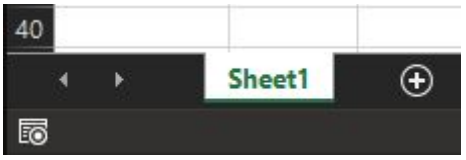
For å importere data fra Excel (.xlsx-filer) til MySQL via phpMyAdmin må man endre filtypen. Det er fordi phpMyAdmin ikke støtter direkte import av .xlsx-filer. De alternativene vi har å velge mellom her er:

- Kommaseparerte verdier (.csv). Disse fungerer dårlig siden komma dukker opp i mange av kildetabellene. Det er mulig å endre separatoren i Excel, men for å gjøre det må du inn i windows-innstillingene.
- OpenDocument Spreadsheet (.ods). Dette er et regneark akkurat som .xlsx, men forskjellen er at phpMyAdmin klarer å importere dataene.

Jeg anbefaler å bruke .ods, så det er den jeg kommer til å dekker her. Hvis du får .CSV til å fungere er det selvfølgelig også en mulighet.

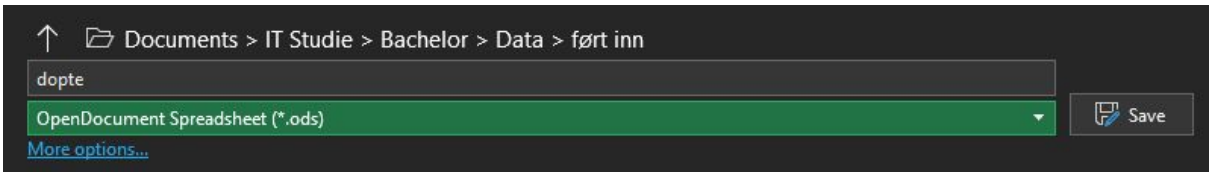
For å konvertere til .ods er det enklest å åpne et nytt regneark ved siden av der du har kildedataene, og lime inn den tabellen du ønsker å importere.
Neste steg blir å endre kolonnenavnene slik at de samsvarer med de navnene som ligger inne i databasen. Dette er viktig, siden det er kolonnenavnene som lar databasen vite hvilken kolonne i kildedataene som hører til hvilken kolonne i databasen. Det følger med et regneark med oversikt over kolonnenavnene i hver av tabellene i databasen. Sørg for at den tabellen du importerer sine kolonner matcher kolonnenavnene til den tabellen du ønsker å importere til.

Det siste du må gjøre før du kan lagre som .ods er å endre navnet på regnearket nederst på skjermen til nøyaktig det tabellen du vil importere til heter. Den endringen kan se slik ut:

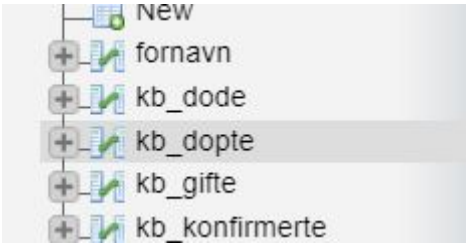


endres til

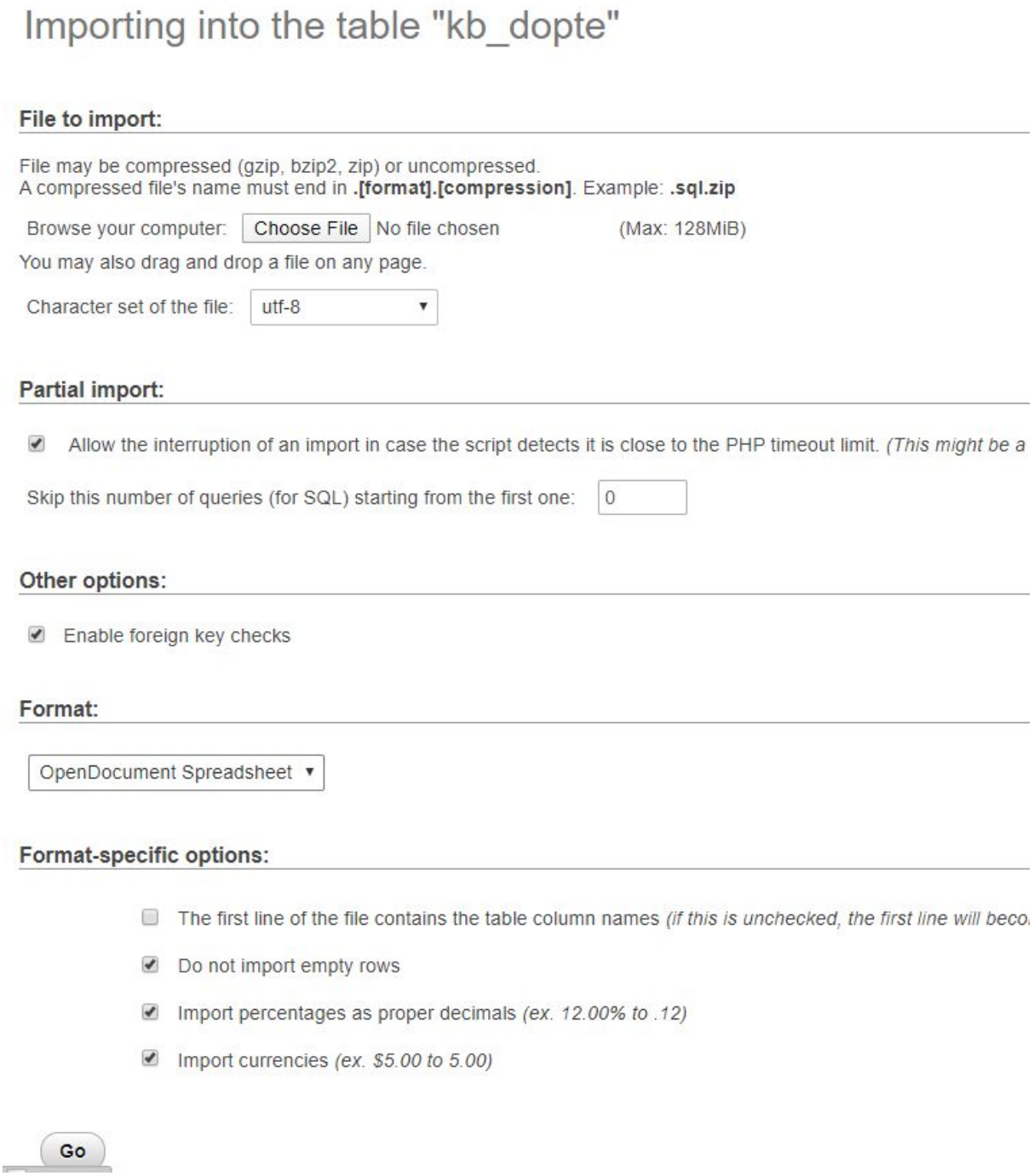
Deretter er det bare å trykke seg inn på Fil -> Lagre som, og lagre filen som et OpenDocument Spreadsheet (*.ods), slik vist på bildet:



Da gjenstår det bare å bruke phpMyAdmin til å legge .ods-filen inn i databasen. For å gjøre det trykker du deg inn på den tabellen du ønsker å importere inn i fra listen på venstre side i phpMyAdmin:



Deretter velger du “Import” fra menyen på toppen av skjermen, akkurat slik som da du satte opp tabellene med .sql-filen. Siden for import ser slik ut:



1. Trykk på “Choose File”, og velg .ods-filen fra der du har lagret den.
2. Velg OpenDocument Spreadsheet fra menyen under “Format:” dersom den ikke har blitt valgt for deg.
3. Kryss av boksen for “The first line of the file contains the table column name”. Det forteller systemet at den kan bruke den første linjen til å matche kolonnene i filen med de i databasen.
4. Trykk på knappen markert “Go” nederst på skjermen.

Med det skal tabellen være lagt inn hvis alt går som det skal. Hvis ikke skal det dukke opp en feilmelding som forteller hva som gikk galt.

Fordeler

- Tydelige feilmeldinger fra phpMyAdmin dersom noe skulle gå galt.
- Man trenger ikke å installere ny programvare for å bruke phpMyAdmin.

Ulemper

- Man må lagre filene i en ny filtype med spesifikke overskrifter i kolonnene.

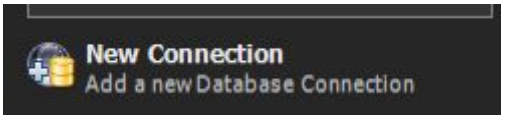
MySQL for Excel

Prosess

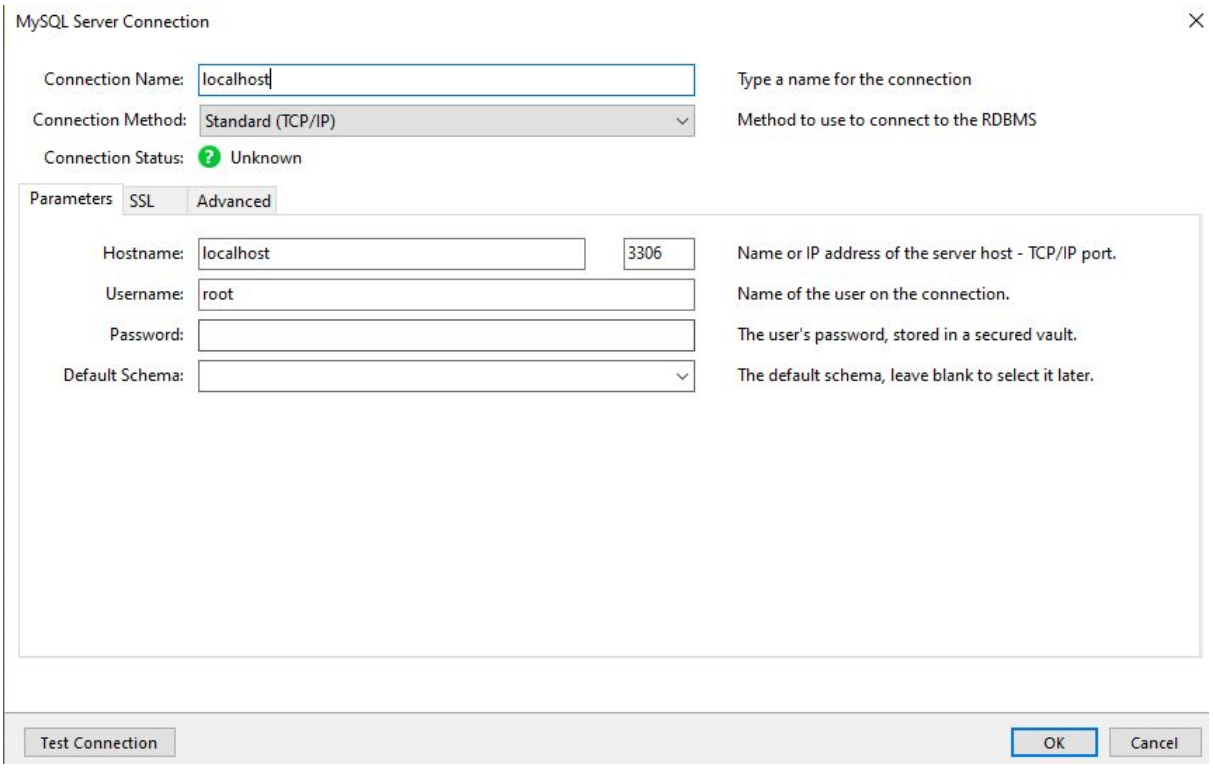
For å importere data med MySQL for Excel starter du med å markere alle feltene du vil importere, og overskriften. Hvis overskriften består av flere felt kan det hende du må

Ikonet for MySQL for Excel finner du under siden for “Data”, helt til høyre:

Første gang du setter det opp må du etablere en kobling til databasen. Det gjør du ved å trykke på knappen markert “New Connection”, nederst i sidemenyen:



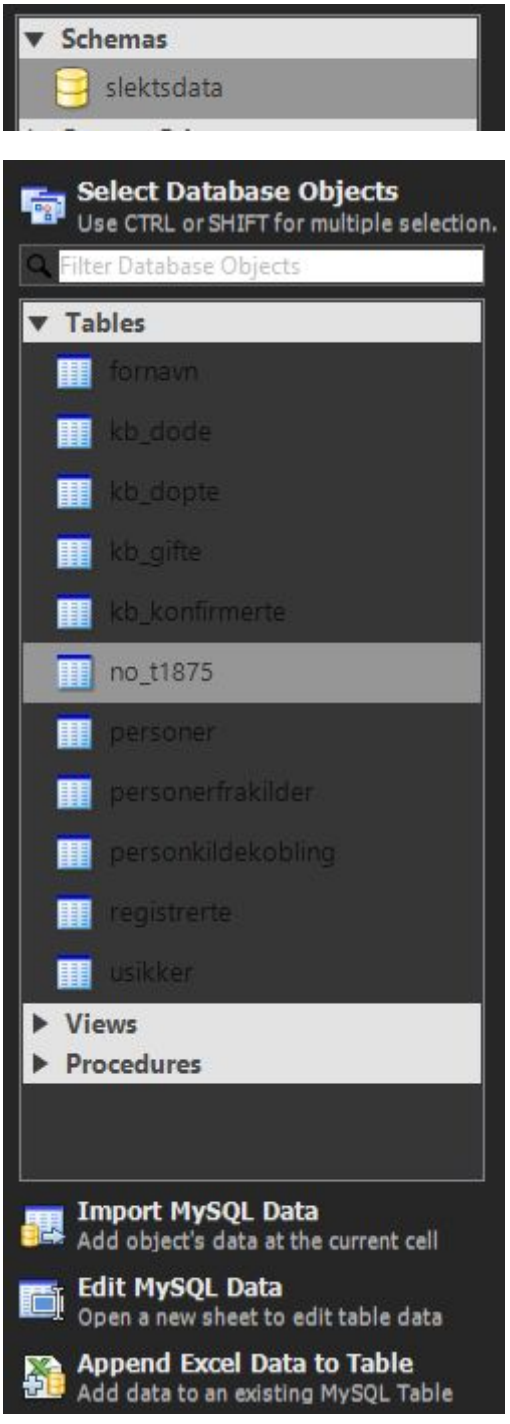
Deretter fyller du inn skjemaet slik:



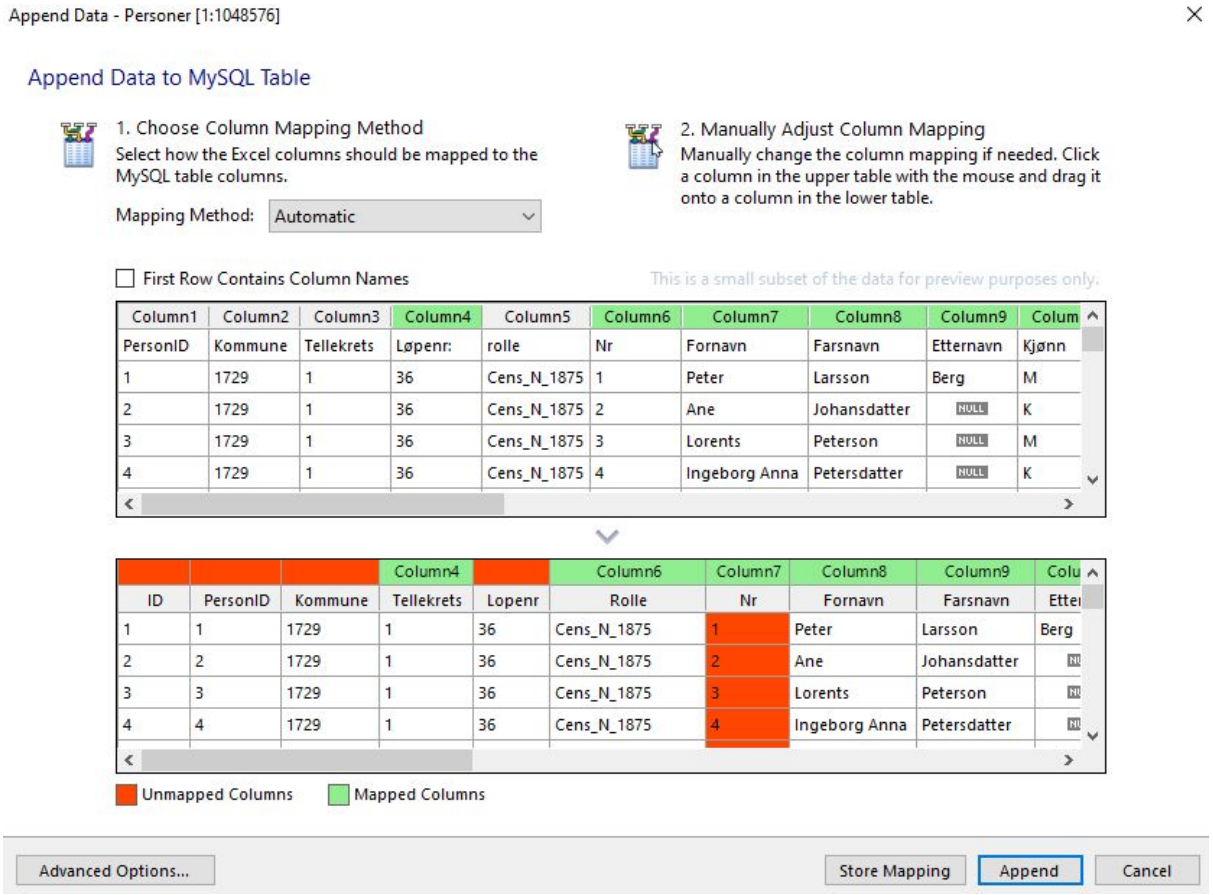
Når du har gjort det er tilkoblingen etablert, så det steget trengs ikke å gjøres for senere importer.

Da kan du dobbelklikke på den tilkoblingen du etablerte i forrige steg. Den skal ha dukket opp i sidemenyen.

Dobbelklikk på databasen som dukker opp, og velg den relevante tabellen:



Når du har markert feltene i Excel som du vil legge inn, inkludert kolonnenavnene, og trykket på tabellen i databasen du ønsker å føre inn i, så trykker du på “Append Excel Data to Table” nederst i menyen. Da skal du få opp et skjema som ser slik ut:



Her må du sørge for at feltene i Excel-arket blir ført inn under de riktige feltene i databasen. Det gjør du ved å først trykke på “First Row Contains Column Names”, så gjør programmet det den kan ut ifra de kolonnenavnene du gir den. Det er ikke en perfekt løsning, så du må gå gjennom selv og sørge for at de matches riktig.
Måten du gjør det på er å dra kolonnenavnet fra den øverste tabellen ned til den riktige kolonnen i den nederste tabellen.
Kolonnen helt til venstre skal være rød, siden det er IDen som databasen gir hver rad. Alle de andre feltene skal være grønne.
Når det kravet er oppfyllt kan du bare trykke på knappen markert “Append”. Da oppdateres tabellen i databasen med de nye feltene.

Fordeler

- Mer brukervennlig.

- Alt kan gjøres i Excel-vinduet.
- Lar deg matche kolonnene i Excel med kolonnene i databasen, slik at endring av kolonnenavnene ikke er nødvendig.

Ulemper

- Utydelige feilmeldinger kan gjøre det vanskelig å løse problemer hvis noe går galt underveis.

Nye tabeller

Løsningen er laget slik at det skal være lett å legge til nye kildetabeller. Det er to aktiviteter man må gjennom for å legge til en ny kildetabell: Man må legge den inn i databasen, og man må legge til litt php-kode som henter personene fra denne nye tabellen.

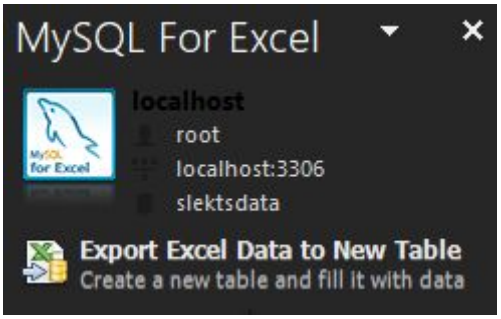
Endringer i databasen

phpMyAdmin og OpenDocument Spreadsheets

For å legge til tabellen i databasen med phpMyAdmin og OpenDocument Spreadsheet endrer du bare navnet på regnearket til det ønskede tabellnavnet på samme måte som du endret den til den tabellen.

MySQL for Excel

For å legge inn en ny tabell via MySQL for Excel markerer du tabellen, inkludert kolonnenavnene. Deretter kobler du deg til databasen med MySQL for Excel slik vist i forrige kapittel. Når du har fått opp oversikten over tabellene velger du “Export Excel Data to New Table”.



Endringer i php-koden

Når tabellen er lagt til i databasen gjenstår det bare å endre filen som henter personene fra kildetabellene sånn at den inkluderer den nye tabellen.
Filen som henter personer fra kildetabellene heter hentFraKilder.php. For å redigere den kan du høyreklikke på den og trykke “Åpne i notisblokk”. Instruksjoner for hvordan man legger til en tabell er inkludert nederst i filen, men jeg går litt dypere inn i det i dette dokumentet.
Under har jeg inkludert de linjene med kode som trengs for å importere personer fra en tabell. For å legge til en ny tabell bytter du ut de feltene markert med grå bakgrunn med de relevante kolonnenavnene fra tabellen du førte inn i databasen i forrige steg.
Dersom noen av feltene mangler, f. eks. fødselsdag og fødselsmåned i en folketelling, er det bare å fjerne dem fra koden. I tillegg må du fjerne de samme overskriftene fra den første delen av koden, som lister hvilke verdier av “personerfrakilder” du ønsker å fylle.

Mal

```
$tabell = "*tabellnavn*";
$sql = "INSERT INTO $table (KildeID, KildeType, Rolle, Kjonn,
Fornavn, Farsnavn, Etternavn, FodselsDag, FodselsMnd, FodselsAar)
SELECT
ID, '$tabell', *Rolle*, *Kjonn*, *Fornavn*, *Farsnavn*,
*Etternavn*, *FodselsDag*, *FodselsMnd*, *FodselsAar* FROM $tabell
WHERE ID NOT IN (SELECT KildeID FROM registrerte WHERE KildeType =
'$tabell');";
$connection-> query($sql);
registrer($tabell);
```

Eksempel

```
$tabell = "kb_dode";
$sql = "INSERT INTO $table (KildeID, KildeType, Rolle, Kjonn,
Fornavn, Farsnavn, Etternavn, FodselsDag, FodselsMnd, FodselsAar)
SELECT
ID, '$tabell', Rolle, Kjonn, Fornavn, Farsnavn, Etternavn,
FodselsDag, FodselsMnd, FodselsAar FROM $tabell
WHERE ID NOT IN (SELECT KildeID FROM registrerte WHERE KildeType =
'$tabell');";
$connection-> query($sql);
registrer($tabell);
```

Lenker

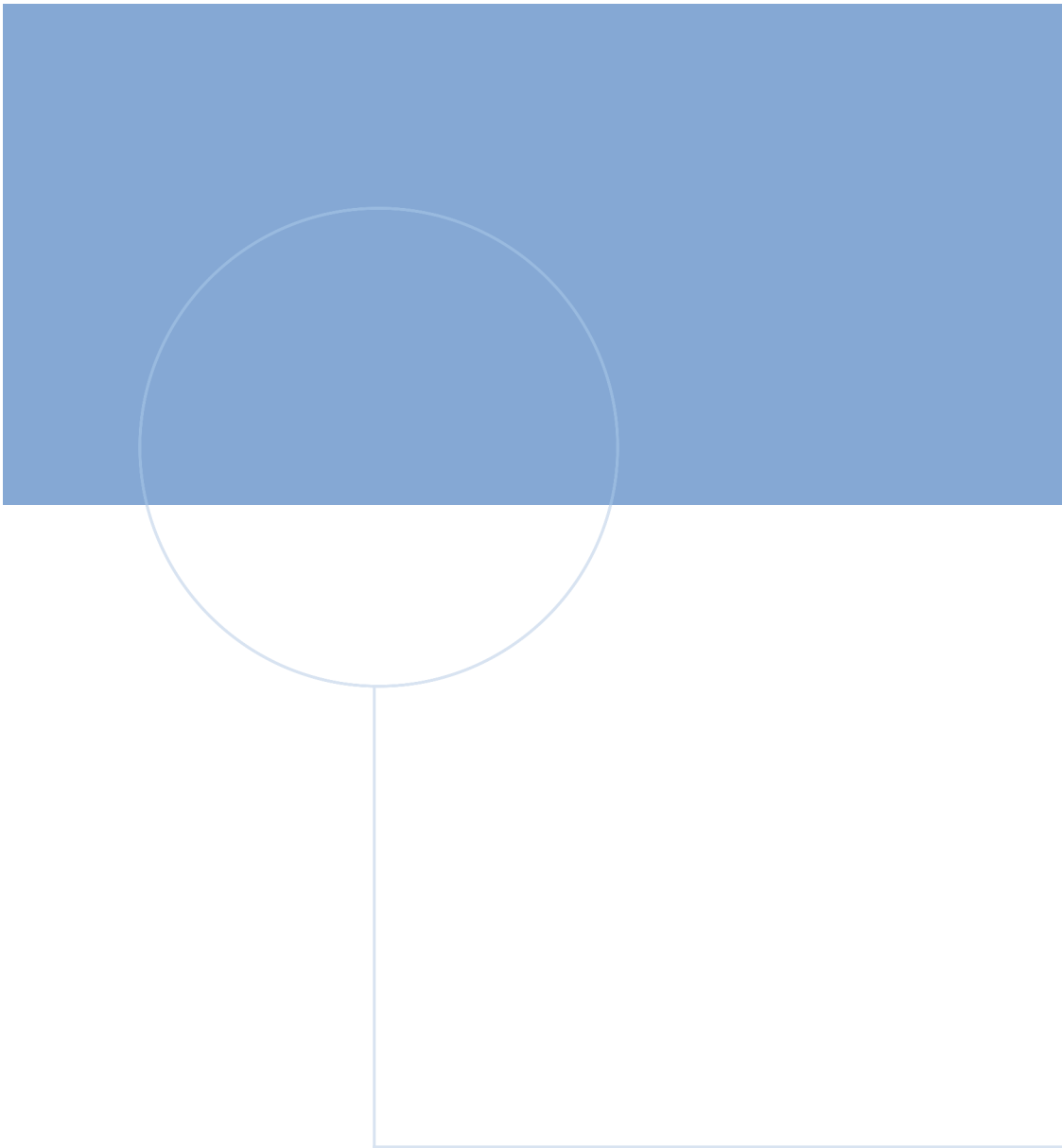
1. Wampserver: https://sourceforge.net/projects/wampserver/
2. MySQL for Excel: https://dev.MySQL.com/downloads/windows/Excel/

Vedlegg 6: Oversikt over kode

Dette vedlegget har som formål å gi en liten forklaring over hvilke filer i kildekoden som gjør hvilke ting, selv om filnavnene i teorien burde være nok. Denne er ment å brukes til videre arbeid, eller ved vurdering dersom noen skulle ha behov til å gå gjennom koden. Jeg har også kommentert koden der jeg følte det var hensiktsmessig, så det bør gå an å lese koden for å forstå hva hver fil gjør.

- fornavnStandardisering.php
    - Her er løsningen for manuell navnstandardisering. Som nevnt i sluttrapporten blir ikke denne brukt. Den er ment som grunnlag for videre arbeid dersom det skulle være nødvendig å ta i bruk en slik løsning for navnstandardisering
- functions.inc.php
    - Funksjoner som brukes i andre filer finnes i denne filen, som blir inkludert i de andre filene slik at de også har tilgang til dem. Her lagres databasetilkoblingen, slik at om man skulle måtte endre databasetilkoblingen kan det gjøres ved å endre én linje. Andre funksjoner i denne filen er funksjonen relatert til visningen av tabellene, funksjonen som viser antall personer linket mot det totale antallet personer i personerfrakilder, og funksjonen som registrerer en kilde som importert.
- hentFraKilder.php
    - Denne filen henter personer fra kildetabellene og fører dem inn i personerfrakilder. Det er denne man redigerer for å legge til nye kildetabeller.
- index.php
    - Forsiden med knapper til de andre sidene.
- link.php
    - Siden for linking med utgangspunkt i kildene. Man får oppgitt en ulinket kilde, og må velge en person å linke den med.
- linkPerson.php

    - Siden for linking med utgangspunkt i personene. Man gir den en person, og må velge én eller flere kilder å linke den med.
- nyPerson.php
    - Registrerer en kilde som en ny person.
- personer.php
    - Oversikt over personer i tabellen personer.
- result.php
    - Søkeresultatet fra siden for søk i database. Blir også brukt til å vise kilder når man trykker på “Vis kilde”-knappen i andre undersider.
- resultPerson.php
    - Søkeresultatet fra siden for søk etter personer fra personerfrakilder. Denne bruker levenshtein-avstand til å sortere resultatet.
- searchPerson.php
    - Skjemaet som gir deg søkeresultatet nevnt i punktet over.
- showLinks.php
    - Personsiden der man ser hvilke kilder personen er linket med, i tillegg til relaterte personer og kilder.
- sokTabell.php
    - Siden for søk i tabell. Lager automatisk et skjema som reflekterer tabellen, og lar deg søke etter både tomme og fylte felt.
- style.css
    - CSS-kode som bestemmer hvordan siden skal se ut. Mye av CSSen i løsningen kommer fra rammeverket Bootstrap, så denne filen inneholder ikke så mye.
- visTabell.php
    - Side for visning av gitt tabell. Man oppgir tabellnavn og den håndterer resten.



NTNU

Kunnskap for en bedre verden