UNIVERSITETET I TRONDHEIM
NORGES TEKNISKE HØGSKOLE
FAKULTET FOR ELEKTRO- OG DATATEKNIKK

HOVEDOPPGAVE

Kandidatens navn: Hans Peter Lindemann

Fag: Datateknikk

Oppgavens tittel (norsk) : Bruk av MS160 i bildearkivet Bauta

Oppgavens tittel (engelsk) : Use of MS160 in the Image Archive Bauta

Oppgavens tekst:

Målsettingen i Bauta-prosjektet er å gjøre historiske fotografier tilgjengelig ved hjelp av nett- og datateknologi. Til nå er det etablert en database med fotografier fra Norsk Folkemuseums bildesamling. Hvert fotografi er (mer eller mindre fullstendig) beskrevet i en felles datamodell, og databasen et tilgjengelig over datanett via et World Wide Web grensesnitt utviklet av Stein Langørgen og Hans Peter Lindemann i forbindelse med vårens prosjektarbeid i databehandling.

Fotografier og beskrivelser er så langt lagret i en standard Oracle relasjonsdatabase. MS160 er et generelt datafilter som er konstruert for å oppnå robust ("fuzzy") informasjonsgjenfinning basert på innhold framfor bruk av indekser. Målsettingen med denne oppgaven er å vurdere bruk av spesialisert maskinvare (MS160) som verktøy for å kunne lage raskere og mer spennende applikasjoner enn det som er mulig når man baserer seg på ordinær utførelse av SQL-spørringer. Oppgaven bør lede fram til konkrete demonstrasjoner.

Oppgaven gitt:	22. august 1994
Besvarelsen leveres innen:	22. desember 1994
Besvarelsen levert:	22. desember 1994
Utført ved (institusjon, bedrift):	Institutt for Datateknikk og Telematikk, NTH
Kandidatens veiledere:	Øystein Grøvlen og Roger Midtstraum

Trondheim, 22. desember 1994

Roger Midtstraum
faglærer

Summary

I have developed a MS160-interface to the Bauta database with historical photographs. A program has been implemented to search in stored information about the photographs. MS160 is a VLSI-chip with the capability of searching through data at 160 MB/s. By using special purpose hardware I have tried to improve the functionality of the system.

The Bauta database will consist of photographs from Norwegian museums and cultural institutions. At the moment pictures taken by the Norwegian photographer Anders B. Wilse are stored in the database. The collection of photographs taken by Wilse consists of about 100 000 photographs, and is owned by Norsk Folkemuseum in Oslo.

Platforms

I have used a 486 PC for searching in the database. The machine is extended with a RAMSCAN board for the ISA-bus. The board held the MS160 coprocessor and 32 MB dedicated memory called the reservoir. The reservoir was loaded with a copy of the reference information of the database.

I have used a WWW server program as interface to the search application. The search interface-programs are developed in C++ and in HTML documents. The toolkit-program MS160api Beta 2.1 have been used to simplify the interface against the RAMSCAN-board and the MS160. I have used the Borland C++ 4.0 compiler.

For the moment the database is stored on a DEC/Alpha with 96 MB RAM. The Bauta database is using Oracle Database Management System version 7. For retrieving all the information from this database I have written a program in Pro*C. SQL statements is included in the Pro*C documents. A Pro*C precompiler is used to generate standard C-code.

Functionality

The user interface provides many opportunities for interaction with the Bauta database:

- The search menu is made out of several fields which can be filled in for searching in the database. Motive, Location, Time, Portrayed Persons end Photographer can be used in querying the database. There is also a field for searching in all the information in the database. A search button invokes the search program. The collection of searchable fields were based on recommendations from Norsk Folkemuseum.
- The search interface allows use of logical operators. The Boolean operators AND, OR and NOT can be used inside each field. Up to seven different criterias can be specified at the same time.

- By changing the menu parameter "Search Type" the user can specify different ways of displaying the search results:

  - The hitlist presents the search results with basic information about each photograph. The elements of the list contain hypertext links to more information about each picture and to showing the photograph by an external viewer program. Depending on the search parameters the small icons are presented as well.
  - The user can also specify to show lists of the distribution of the search results on either county or decade. These types of frequency lists can be generated for all combinations of the search-criterias.

- Different parameters can also be altered by the user in order to change the presentation of the hitlist. Small copies of the photographs called icons can alternatively be presented with the search result. The user can decide maximal number of hits in the database which are to be transferred.

Solution

The user interface was made of documents written in the hypertext markup language HTML. Hypertext links were the connections between the various documents. Dataproducing processes were also specified in the links. The search program produced the results written in HTML code.

A program was made to transfer the information in the database to the reservoir of MS160. The Pro*C-program generated a file containing all relevant reference information. The database was read by using nested relations. The file can be transferred to the MS160 PC by FTP. Another program wrote the file to the MS160 reservoir.

The programs for searching were implemented using the library functions of MS160api. The programs were called by the WWW-server. The server also returned the search-results to the user.

Results

The response time in my solution was low. By comparing the response time with the existing Oracle database I found the MS160 superior.

The possibilities for extending the functionality were good. The experiences with implementing the frequency lists and logical operators were satisfying.

In my study of possibilities for searching with MS160 in structured information, I also found some weaknesses. Errors might occur when searching in records of variable length. Corrections of these might influence negatively on the response time.

The possibilities for searching with left truncations in specific field were also limited. This is unfortunate because this was specifically wanted by participants of the Bauta project.

Many suggestions on improvements of the functionality of the Bauta-base have been made in the report. Most of them were not implemented due to time shortage. In the future these might be considered realised. I also made comments on how to improve my program. These might solve some of the weaknesses I found.

I conclude with that it is not clear whether the advantages of fast searching justify the possible problems. It is up to the participants of the Bauta-project which factors should be emphasized. This report is however a framework for further considerations whether MS160 could be used for searching in the Bauta database.

Innhold

Figurer

Tabeller

Kapittel 1

Innledning

Denne rapporten oppsummerer arbeidet utført av Hans Peter Lindemann med hovedoppgaven "Bruk av MS160 i bildearkivet Bauta" ved Norges Tekniske Høgskole. Oppgaven ble gitt høsten 1994 av Institutt for Datateknikk og Telematikk.

Hovedmålsettingen med oppgaven er å vurdere bruk av søkebrikken MS160 i bildearkivet Bauta. Bauta er en database som inneholder historiske fotografier sammen med en del opplysninger som beskriver dem. Basen vil i første omgang innholde bilder samlet av den norske fotografen Anders Beer Wilse, og hoveddelen av dem er fra den første delen av 1900-tallet.

Ved å utvikle en applikasjon som muliggjør søking i databasen vil jeg vurdere hvor egnet MS160 er til søking i databaser med en informasjonsstruktur og bruk som Bautabasen.

De viktigste spørsmålene jeg forsøker å svare på i rapporten er både hvilke muligheter og hvilke begrensinger søkebrikken MS160 gir. Jeg vil spesielt se på hvilken funksjonalitet brikken åpner for ved muligheten til rask søking gjennom store datamengder. Brikken gir liten støtte til søking i strukturerte datamengder og jeg vil se på hvilke begrensninger og vanskeligheter dette kan føre til.

MS160 konkurrerer med andre systemer for informasjonsgjenfinning. Jeg vil derfor sammenligne hvor godt og effektivt brikken kan ta i bruk den mest alminnelige funksjonaliteten.

1.1 Kontaktpersoner

Bauta-prosjektet er et samarbeid mellom Norsk Folkemuseum, Nasjonalbiblioteksavdelinga i Rana (NBR), BIBSYS, Norsk museumsutvikling, NTH/IDT, UiO/Dokumentasjonsprosjektet, Riksarkivet og Uninett. Min oppdragsgiver var IDT, og jeg samarbeidet med Norsk Folkemuseum og NBR.

Faglærer for oppgaven på IDT var Roger Midtstraum (`roger@idt.unit.no`), og veileder var Øystein Grøvlen (`oysteing@idt.unit.no`).

Min kontaktperson i Mo i Rana var Svein Arne Brygfjeld(`sveinb@nbr.no`).

Mine kontaktpersoner ved Norsk Folkemuseum var Trond Bjorli (`tb@norskfolke.museum.no`) og Steinar Bjørneset (`sb@norskfolke.museum.no`).

1.2 Organisering av denne rapporten

Denne prosjektrapporten er organisert i følgende deler:

- Kapittel 1: Innledning.
- Kapittel 2: Bakgrunn. Innholder grunnleggende bakgrunnsinformasjon om elektronisk informasjonsbehandling og en beskrivelse av MS160 og Bauta-prosjektet.
- Kapittel 3: Problembeskrivelse. Inneholder en beskrivelse av de problemene som jeg vil vurdere ulike løsninger av i oppgaven.
- Kapittel 4: Datalagring. Inneholder en beskrivelse av mine valg for lagring av dataene ved bruk av en MS160 applikasjon.
- Kapittel 5: Brukergrensesnitt. Inneholder valg av brukergrensesnitt, og en beskrivelse av dette.
- Kapittel 6: Implementasjonsbeskrivelse. En beskrivelse av realiseringen av den valgte løsningen.
- Kapittel 7: Erfaringer. En beskrivelse av de erfaringene jeg har gjort med det ferdige resultatet.
- Kapittel 8: Konklusjon.

Kapittel 2

Bakgrunn

I dette kapittelet vil jeg beskrive en del grunnleggende bakgrunnsinformasjon om elektronisk informasjonsbehandling generelt og om systemer for informasjonsgjenfinning. Til slutt vil jeg gi en beskrivelse av MS160 og Bauta-prosjektet.

2.1 Informasjonsbehandling

Lagring av store informasjonsmengder er en av de viktigste bruksområder til moderne datamaskiner[Salton 89]. Data i maskinlesbar form lagres i databaser. Mengden av tilgjengelig informasjon i verden i dag er større enn noensinne. Ved datamaskiner er det laget et medium som er i stand til å lagre store datamengder og i tillegg enkelt hente dataene ut på kort tid.

En database er en samling av relaterte data. Med data menes fakta som kan opptegnes og i tillegg har en implisitt mening. Ifølge [Elmasri 89] er alminnelig bruk av ordet database mer begrenset:

- En database er en logisk koherent samling av data med en innebygget mening.
- En database er designet, bygget og fylt med data med et spesielt formål. Den har en tilsiktet gruppe med brukere. På forhånd er det tenkt ut applikasjoner som denne gruppen er interessert i.
- En database representerer et aspekt ved den virkelige verden. Aspektet kalles ofte en mini-verden. Forandringer i denne mini-verdenen blir reflektert i databasen.

En database har med dette en kilde som dataene er avledet fra, en interaksjon med den virkelige verden og en tilsiktet brukergruppe som er interessert i innholdet.

Informasjonen i en database kan ha ulike former. Den kan bestå av film, bilder, lyd og tekstlig beskrivelse. Det finnes idag metoder for å lagre representasjoner av alle disse på datamaskiner. Av de ulike informasjonsformene er det den tekstlige beskrivelsen som brukes mest til datalagring. Den er enklere og blir lettere forstått enn de andre formene. Det meste av arbeidet med informasjonsprosessering har derfor vært foretatt rundt problemer med tekstbehandling.

For alle lagringsformene er det imidlertid et problem å uttrykke en tilsiktet mening nøyaktig nok. Informasjonen må på en best mulig måte avspeile den mini-verden den skal representere. Brukere må mest mulig effektivt oppfatte innholdet i informasjonen slik det var

tiltenkt. I tillegg til de rent tekniske problemene er det derfor en rekke psykologiske problemstillinger forbundet med informasjonslagring i datamaskiner.

Ulike fagretninger konsentrerer seg om ulike aspekter ved behandlingen av elektronisk informasjon. Forskning på databasesystemer dreier som om hvordan dataene best kan lagres. Informasjonsgjenfinning dreier seg mer om hvordan en best kan finne frem i informasjonen i datamengden.

2.1.1 Databasesystemer

Et databasesystem er en samling av programmer som gjør det mulig å opprette og vedlikeholde en database. Den er konstruert for å representere og manipulere objekter fra den virkelige verden og relasjoner mellom disse. Den skal ha en en mest mulig sentralisert styring av operasjoner på dataene.

Det er derfor vanlig å tildele databasesystemet en rekke oppgaver. Den skal definere databasen, konstruere den ved å lagre data, og manipulere ved hente ut data og oppdatere informasjonen. I litteraturen stilles i tillegg en rekke krav til funksjonaliteten. I [Saxton 90] stilles følgende krav til styringen av databasen.

- Redusere redundans av lagrete data
- Unngå inkonsistens av lagrete data
- Dele data mellom flere brukere
- Få frem standarder
- Anvende sikkerhetsrestriksjoner.
- Beholde data integritet

Ved oppfyllelse av disse kravene vil en få en enhetlig struktur av dataene. Målet er at brukeren skal få en stabil og enkel tilgang til opplysningene. Den faste strukturen kan imidlertid gjøre at representasjonen av informasjon fra den virkelige verden blir vanskeligere. Data må tilpasses skjemaer med en stiv struktur. Databasesystemer vil derfor lett sette begrensninger for hvor godt en mini-verden kan modelleres.

Vanligst av dagens systemer for databasestyring er såkalte relasjonsdatabasesystemer. Disse gir støtte til lagring av strukturerte data i todimensjonale tabeller. Mellom tabellene eksisterer det relasjoner som skal avspeile en virkelig avhengighet mellom dataene. Lagring og henting av data skjer vanligvis gjennom et enhetlig data-manipulerings språk. Mye brukt i ulike systemer er spørrespråket SQL.

Eldre hovedsystemer er hierarkiske databaser og nettverksdatabaser. Hierarkiske databaser krever en mer rigid datastruktur og gir ikke samme støtte til alle typer relasjoner mellom data. Funksjonaliteten til begge er mer avhengig av den fysiske lagringen. De ikke like fleksible til å med forespørsler som relasjonsdatabaser. Det kan være vanskelig å formulere forespørsler som ikke ble planlagt da databasen ble designet.

I de siste årene har også såkalte objektorienterte databaser vært tatt i bruk. Disse åpner for å definere objekter. I objektene kan både dataene og metoder som kan utføres på disse dataene tas i bruk. Arv av egenskaper mellom ulike objekter kan også defineres. Bruk av objektorienterte databaser kan ha flere fordeler fremfor andre systemer:

- De gir støtte støtte for en fleksibel modellering av objekter. Sammensatte objekter kan defineres med egenskaper fra tidligere definerte objekter i en hierarkisk struktur. Gjennom arv av attributter og metoder gis mulighet til spesialisering av egenskaper.
- Endringer får bare innflytelse på det enkelte objekt som modifiseres
- Kun ønskete operasjoner kan utføres direkte på dataene.
- Dataintegriteten sikres bedre ved et utvidbart typeapparat.
- Det vil dessuten gis et bedre samsvar og støtte til programmering med objektorienterte programmeringsspråk.

Mye data har imidlertid en struktur som gjør bruk av tradisjonelle databasesystemer vanskelig. Eksempler på dette er de store mengdene av fritekst som i dag er lagret elektronisk. Selv om teksten også representerer deler av verden er denne sammenhengen som oftest så kompleks at dette ikke lar seg definere av en database. Noen databasesystemer gir støtte til lagring av fritekst i spesielle felter. Disse feltene vil imidlertid ikke utgjøre en integrert del av databasen. Dataenes mening er ikke representert.

For behandling av denne typen data er det utviklet spesialtilpassete programmer [Sieverts 93]. En del av informasjonen vil ofte ha en struktur. Ved lagring av ulike dokumenter vil for eksempel starten av dokumentet ofte ha faste felter for beskrivelse av innholdet. Disse feltene kan underlegges alminnelig databasekontroll. Den ustrukturerte delen av dataene vil det ikke være en tilsvarende kontroll over. Ulike metoder kan imidlertid tas i bruk for å finne frem i en slik datamengde (se avsnitt 2.2.1. Enkelte typer dokumenter vil være delvis strukturerte. Det forskes idag på databaser for spesielle dokumentformater som SGML[Blake 94].

2.1.2 Informasjonsgjenfinning

Informasjonsgjenfinning defineres som metoder og prosedyrer for å gjenfinne lagrete data om et emne. Den omhandler hvordan informasjonens organisering, struktur, henting og visning best kan tilrettelegges for effektiv gjenfinning.

Gjenfinningen kan deles i to ulike former: **fakta-gjenfinning** og **begrepsgjenfinning** [Sift 85]. **Fakta-gjenfinning** kjennetegnes ved at brukeren har et veldefinert informasjonsbehov. Brukeren ser etter spesielle fakta på grunnlag av en konkret forespørsel. Typisk for denne situasjonen er at det er at det også er også er et veldefinert kriterium for om data er relevante eller ikke. Det er heller ikke vanskelig for brukeren å avgjøre om responsen fra systemet er relevant eller ikke.

Motstykket til faktagjenfinning er **begrepsgjenfinning**. Behovet til brukeren er vagt og tvetydig. Kriteriene for relevans er uklare. De kan være subjektive og avhengig av brukerens egen kunnskap. Dette kan vanskelig la seg spesifisere i et søkekriterium. Brukeren må kanskje selv studere dataene lenge for å finne ut om de er relevante. Forskning rundt informasjonsgjenfinning har mye vært fokusert rundt hvordan upresise søkekriterier på en best mulig måte kan returnere relevant informasjon [Saxton 90].

Hvorvidt søkingen har karakter av faktagjenfinning eller begrepsgjenfinning avhenger mye av hva slags data som er lagret eller i hvilke deler av dataene det søkes. Dersom dataene har et innhold og struktur som gir en klar mening for brukeren vil søkingen som oftest tilsvare faktagjenfinning.

Fakta-gjenfinning brukes gjerne i databaser for administrasjon. Et eksempel på dette kan være opplysninger om alle studenter i databasen til et universitet. En klar og utvety-

dig forespørsel vil være henting av all informasjon som er lagret om en bestemt student. Eksempel på begrepsgjenfinning kan være søking i databaser som består av dokumenter i naturlig språk. En ønsket forespørsel vil typisk være henting av dokumenter som omhandler et spesielt emne. Relevanskriteriet vil i en slik sammenheng lett bli uklart. I den samme databasen kan imidlertid både fakta og begrepsgjenfinning forekomme. Selv i den samme forespørselen kan det være elementer av begge deler. Det kan søkes etter dokumenter av en spesiell forfatter eller i en spesiell tidsperiode. Denne søkingen vil være klar og utvetydig. Slike kriterier kan kombineres med en mer uklar søking etter et tekst-dokumenter om et emne.

Som nevnt i innledningen vil det imidlertid ofte være vanskelig å konstruere databaser med informasjon som er tilstrekkelig meningsbærende for en bruker. Tekstlig informasjon vil for eksempel være tvetydig og vil ikke nødvendigvis bli oppfattet slik den var tiltenkt. Flere ord kan beskrive det samme forholdet. Et søkekriterium med samme mening vil ikke alltid gi treff dersom det kreves at ordene skal være syntaktisk identiske. Naturlig språk sin tvetydighet gjør at det ikke er sikkert at søkekriteriene blir oppfattet korrekt.

Det er også et problem at brukeren selv formulerer uklare kriterier for søking i dataene. På forhånd vil brukeren ha uklare forestillinger om hva slags data han søker etter. Vanskeligheten med å orientere seg i store datamengder bidrar til at søkingen blir upresis. Brukerens kjennskap til emnet det søkes i og til selve dataene vil avgjøre hvor klart kriteriene kan formuleres. Søkekriteriene kan dessuten være så komplekse at de vanskelig lar seg formulere i valg av enkeltord.

Tilsammen fører dette til at det blir svært vanskelig å hente ut alle relevante data fra en større database. Det vil etter all sannsynlighet være informasjon som brukeren aldri får. Blant informasjonen som hentes ut vil det også ofte være data som ikke er relevante. For å måle effektiviteten av gjenfinningen brukes begrepene Recall og Presisjon [Salton 89].

1. Recall(R) defineres som andelen av det relevante materialet som er hentet:
   R = Antall relevante elementer hentet / Totalt antall relevante elementer.

2. Presisjon(P) er andel av de hentete elementene som er er relevante:
   P = Antall relevante elementer hentet / Totalt antall elementer hentet.

Målet med informasjonsgjenfinning er å maksimerere begge disse størrelsene. Samtidig optimalisering er imidlertid vanskelig slik at det må finnes et kompromiss. Helt klare kriterier kan få problemer med å finne alle alle elementer med samme mening. Dersom kriteriene settes mer uklare kan de inkludere flere relevante elementer. Det vil imidlertid lett også gis treff for en større andel irrelevante elementer. Presisjonen vil synke. For den enkelte bruker vil det være et dilemma hvor klart søkekriteriene skal defineres.

2.2 Informasjonsgjenfinningssystemer

For å muliggjøre gjenfinning av elekronisk lagret informasjon trenges programmer som kan utføre en ønsket funksjonalitet. Tradisjonelle databasesystemer har tidligere gitt liten støtte til de uklare søkene. Det viktigste ved funksjonaliteten har vært å sikre optimal lagring av dataene. Dette har sammenheng med at det viktigste bruksområdet har vært til de administrative databasene. Dette er mindre databaser der det er viktig med en sikker og konsistent lagring. Disse støtter i hovedsak faktagjenfinning. Et typisk bruksområde er bedrifters lagring av intern informasjon. Noen egentlig søking er ikke vanlig i disse. Det vanlige er at brukeren på forhånd vet hvilke data som skal hentes ut.

Spesiell programvare beregnet for effektiv informasjonsgjenfinning har vært utviklet. Disse er i stand til å ta i bruk mye funksjonalitet som gir støtte til uklare søk. Informasjonsgjenfinning kan foretas i data med varierende grad av struktur. Som regel mangler de imidlertid mye av den grunnleggende funksjonaliteten til en database. De gir liten styring over datalagringen. Enkeltdata kan ikke oppdateres. Typisk for funksjonaliteten er at hele datamengden hentes fra ferdige formaterte filer. Ved statiske datamengder som sjelden oppdateres vil dette ikke være noe problem. Hele databasen kan oppdateres når det er nødvendig ved at filene med informasjonen leses på nytt. Datafilene kan genereres genereres av et vanlig databasesystem og dermed gi mulighet for enkel oppdatering og sikre kontroll over lagringen.

De ulike produsentene av tradisjonelle databasesystemene har de senere årene utvidet funksjonaliteten til mye av det samme som de spesialutviklete programmene. Men ifølge [Sieverts 93] har de fremdeles et stykke igjen til funksjonaliteten og effektiviteten er like god. Kravene til effektiv lagring og oppdatering kan lett gå utover søkingen. Dette er spesielt uheldig ved ved statiske datamengder. Her er effektiv oppdatering av dataene ikke så viktig.

Ulike strategier har vært forsket på for å perfeksjonere virkemåten til systemer for informasjonsgjenfinning. Disse søker å forbedre søkehastighet, funksjonalitet, presisjon og recall. Ifølge [Visschedijk 93] har det vært et markant skille mellom forsking utført på maskinvare og programvare. Programvare-utviklere har sett på problemer rundt recall og presisjon som det viktigste problemet. De har søkt å finne nye metoder for søking med uklare kriterier. Målet har vært å forbedre informasjonsgjenfinningen på to ulike nivåer. Datalagringen skal bedres og på søkenivået skal mulighetene til formulering av forespørsler styrkes.

Maskinvare-utviklere derimot har fokusert på søkehastigheten. De vil redusere responstiden for eksisterende og tradisjonelle metoder som brukes i programvare. Med hjelp av spesialisert maskinvare søkes det å gi raskere aksess til dataene.

2.2.1 Datalagring

Elektronisk informasjon kan ha ulikt format. Dataene kan være mer eller mindre strukturerte og ha forskjellige lagringsenheter. I store utstrukturerte datamengder er det aktuelt i tillegg å opprette deskriptorer for å representere meningen. Eksempler på dette er noen få nøkkelord som beskriver hovedinnholdet i store mengder fritekst. Nøkkelordene kan beskrive hvilket emne og fagområde teksten ligger under. I større eller mindre grad vil dette si noe om menings-innholdet i teksten. Ved hjelp av deskriptorene søkes det å forbedre gjenfinningen av dataene. Med søking i orginaldataene kan ofte bli vanskelig finne data med samme mening som et søkekriterium. Med spesialtilpassete deskriptorer kan dette bli lettere.

Mye forskning har vært utført på hvordan deskriptorer best kan opprettes. Tradisjonelt har deskriptorer vært laget manuelt. Idag er det imidlertid laget programmer for automatisk generering av deskriptorer fra tekst-dokumenter. En kunnskapsbase brukes til å klassifisere og analysere dataene. Fra andre lagringsformer enn tekst er automatisk generering av deskriptorer vanskelig. Det vanlige er manuelt å lage tekstlige beskrivelser.

Fordi deskriptorer vanligvis er ord fra naturlig språk står en overfor semantiske problemer. Synonymer er forskjellige ord med samme mening mens homonymer er samme ord med forskjellig mening. Problematisk er også ord med ulik grad av spesifisering. Båt og sjark kan referere til det samme objektet. Flere systemer for automatisk deskriptorgenerering forsøker å løse disse problemene ved hjelp av omfattende oppslagsverk og ordbøker. Ved

bruk av manuelle metoder vil en tilsvarende generering være svært tidkrevende.

I [Visschedijk 93] beskrives flere slike systemer. Et eksempel er CLARIT som automatisk kan danne deskriptorer ut fra fritekst. I en kunnskapsbase ligger de fleste engelske ord med de 1 000 000 vanligste tolkningene. Alle ord som det forventes er meningsbærende har et innslag i basen. Alle ord bli representert av et eller flere substantiver. Tvetydige ord vil da utgjøre ca 1 % av ordene. Tolkning av disse skjer ut fra statistikker for fordeling av ord i selve teksten og i vanlig engelsk. I den ferdige indeksen vil ordene være organisert etter deres mening og grad av spesifisering. Selv om tolkningen ikke alltid blir korrekt åpner det for søking etter begreper og ikke bare etter enkeltord. Det er ikke utviklet tilsvarende systemer for norsk tekst. Konstruksjonen av en tilsvarende kunnskapsbase vil kreve store ressurser.

2.2.2 Funksjonalitet

Søking i databaser foregår ved at brukeren formulerer en forespørsel. Avhengig av forespørselens innhold returnerer databasen informasjon på grunnlag av en innebygget funksjonalitet.

[Saxton 90] inndeler forespørsler i følgende ulike former etter deres grad av direkte henvisning til lagrete data.

1. Forespørsler rettet direkte mot poster. Disse vil henvise direkte til dataene.
2. Forespørsler rettet mot dokument-representasjon lagret som poster. Disse vil ikke henvise til selve dataene, men til deskriptorer som beskriver dem.
3. Forespørsler som på grunnlag av en dokument-representasjon og kriteriet krever at systemet gir grader av relevans for aktuelle dokumenter.
4. Forespørsler som på grunnlag av tilgjengelige fakta krever at systemet skal gi en et vurdering på et høyere nivå for brukeren. Dette kan for eksempel være å finne de mest egnete kandidatene til å få en jobb ut fra enkeltdata om alle søkere.

Av disse er det de to første som er bruk i tradisjonelle systemer for informasjonsgjenfinning. I databasesystemer består gjenfinningen som regel i å hente ut informasjonen ut fra kriterier som henviser direkte innholdet i enkelte felter. [Sieverts 93] har i en omfattende undersøkelse evaluert de vanligste kommersielle systemene for informasjonsgjenfinningen. Mulighetene til å formulere forespørsler ble studert. I en oversikt presenteres de mest alminnelige formene:

- Mulig å søke ved hjelp av indeks.
- Mulig å søke i ikke indeksert informasjon.
- Mulig å vise indeksere ord sammen med antall og synonymer.
- Mulig å søke direkte i indekser.
- Mulig å spesifisere hvilke felt det skal søkes i.
- Mulige å bruke boolske operatorer.
- Mulig å kombinere boolske operatorer ved hjelp av parenteser.
- Mulig å bruke tidligere søkeresultat direkte i boolske uttrykk.

- Mulig å søke etter nærliggende kriterier i datamengden.
- Trunkering mulig, høyre, venstre eller intern.
- Søking i intervaller mulig (Større enn, Mindre enn).
- Mulig å lagre søkestrategi.

Felles for alle disse forespørslene er at de kun henviser direkte til dataene og deskriptorer av dem. Informasjonen som hentes ut er poster eller dokumenter som har et innhold som entydig tilfredsstiller søkekriteriene

Som tidligere nevnt forskes det på hvordan det best kan tilretteles for søking med uklare kriterier. De tradisjonelle systemene kritiseres for ikke å gi tilstrekkelig støtte slik at alle relevante dokumenter kan hentes. Karakteristisk for de fleste brukte systemer idag er at de kun gir støtte til bruk av Boolsk logikk. Sannhetsverdien til en forespørsel er enten 0 eller 1 for ulike data. Ved bruk av logiske operatorer regnes det ut en entydig sannhetsverdi for hele uttrykket. Operatorene er den fremste støtten til formulering av avanserte forespørsler.

[Cooper 88] kritiserer bruken av boolske operatorer av flere årsaker. De gir ikke den tilstrekkelige støtten når kriteriene er uklare:

- Lite brukervennlige formler: Alminnelige brukere vil ha problemer med å skjønne funksjonaliteten. Uten kjennskap til boolsk logikk kommer bruken av operatorene lett i konflikt med vanlig språkbruk.
- Problem ved for få eller for mange returnerte treff: Det gis ingen indikasjon for hvordan forespørselen kan endres til å gi et passende antall treff.
- Varierende viktighet: Det gis ikke støtte til å definere deler av søkekriteriet som mer viktig enn andre deler.

Den tredje formen for forespørsel som ble gitt var systemer som gir grader av relevans for aktuelle dokumenter. Med uklare søkekriterier vil en slik rangering være ønskelig. Dette vil løse problemet med for få eller for mange treff. Brukeren vil også få returnert dokumenter som delvis oppfyller søkekriteriet. Det forskes på ulike metoder for hvordan dette kan gjøres.

Den enkleste metoden er å gi en rangering etter antall oppfylte søkekriterier i ELLER-uttrykk. Metoden er enkel, men gir likevel en indikasjon for hvor godt ulike data tilfredsstiller søkekriteriet.

Med bruk av fuzzy logic introduseres sannhetsverdier mellom 0 og 1. Grader av likhet mellom en forespørsel og et dokument kan brukes. Oppfyllelse av enkeltkriterier kan gis en verdi fra 0 til 1. Ved hjelp av spesielle beregningsformer kan sannhetsverdien til hele uttrykket regnes ut. Et eksempel på bruk er at det åpnes det for at ulike kriterier gis ulike vekter. Dersom det settes like verdier for oppfyllelse av alle kriterier vil rangeringen likne mye på den som teller antall oppfylte i et ELLER-uttrykk. Settes ulike verdier vil enkeltkriterier ha større innvirkning på den totale sannhetsverdien enn andre. Deler av søkekriteriet defineres til å være mer viktig enn andre. Sannhetsverdien kan også settes til å være avhengig av andre faktorer. Ved søking i fritekst kan verdien settes til å avhenge av antall ganger et ord forekommer i et dokument. Hvor nærme søkeordene er i et enkeltdokument kan også settes til å påvirke verdien. Det kan defineres at hvis flere søkeord er i nærheten av hverandre gir det større verdi enn om de er lenger fra. For eksempel kan det være en fordel dersom ordene information og retrieval er like ved siden av hverandre enn

om de er i hver ende av dokumentet. Sannsynligheten vil være større for at et dokumentet omhandler informasjonsgjenfinning

En annen teoretisk innfallsvinkel er å fokusere på sannsynlighets-betrakninger. Denne forskingen ser mer på sannsynligheten for at et dokument er relevant enn på graden av relevans. Denne teknikken vil imidlertid også kunne danne grunnlaget for en rangering. Det kan rangeres etter sannsynlighetsverdien til de hentete elementene.

Som den siste formen for forespørsel nevnte jeg de som gir en vurdering på et høyere nivå for brukeren. For å muliggjøre dette er det utviklet såkalte ekspertsystemer. Disse er et forsøk på å bevege seg fra databaser til kunnskapsbaser. Kunnskap kjennetegnes med et høyere abstraksjonsnivå enn alminnelige data. Det er innlagt en rekke semantiske sammenhenger. Under utføringen av søkingen og under presentasjonen av resultatet tas det hensyn til disse. Til nå har imidlertid funksjonaliteten vært begrenset. Naturlig språk er for uttrykksfullt og tvetydig til at søkekriterier på noen måte kan gis en fullstendig tolkning. Ifølge [Salton 89] vil bruk av ekspertsystemer kun være aktuelt under disse forutsetningene.

- Når miljøet er svært begrenset og kan representeres med få entiteter og deres relasjoner.
- Når dataene oppfyller en spesiell funksjon som automatisk plasserer dataene i en spesiell sammenheng der de fleste vanlige tvetydige tolkningene er fraværende.

Brukeren kan på andre måter assisteres i formuleringen av søkekriterier. Systemet kan gi forslag til hvordan søkekriterier kan utvides eller begrenses. Tvetydigheter eller mulige feil kan det gjøres spesielt oppmerksom på. Som et alternativ til formuleringen av avanserte deskriptorer kan forespørslene sikre at meningen blir bevart. Brukeren kan gjøres oppmerksom på hvilke aktuelle synonymer som finnes. Med bruk av ulike kunnskapsbaser kan det gis opplysninger om hvordan kriteriene bør velges.

Selv om en rekke systemer for informasjonsgjenfinning åpner for avanserte søkeformer er det et problem at få er optimalisert i forhold til dem. Ved for stort tidsforbruk faller mange av fordelene bort.

2.2.3 Optimalisering av søketid

Som nevnt er det i første rekke gjennom introduksjon av forbedret maskinvare at en idag forsker på å redusere søketiden i informasjonsgjenfinningsystemer. Også på programvaresiden brukes imidlertid ulike strategier for å gi raskere responstid. Dette gjelder spesielt i tradisjonelle databasesystemer. Den mer fastlagte funksjonaliteten gjør optimalisering av bestemte metoder mer aktuelt.

Forbedret Programvare

I tradisjonelle databasesystemer er det vanlig å optimalisere forespørslene. Data-manipuleringsspråk som SQL gir mulighet til komplekse setninger i et høynivåspråk. De fleste systemer har algoritmer for hvordan en vilkårlig forespørsel best kan utføres.

Viktig er bruk av indekser. Operativsystemer gir støtte til raskere aksessering av data ved bruk av indeksering av filposisjoner i lagringsmediet. Databasesystemet gir også mulighet for indeksering av ofte aksesserte data. Denne indekseringen foretas oftest ved bruk av inverterte lister. Lister over grupper av søkeord lages bygget opp av selve ordet sammen med pekere til posisjonene til ordet i datalageret. Ulike trestrukturer som B-trær kan brukes

for raskt å aksessere et ord i indeksen. Dersom deskriptorer brukes som beskriver en større datamengde er det vanlig å indeksere disse. I relasjonsdatabasesystemer er indeksering av nøkkelverdiene vanlig.

Mye arbeid og lager kan gå med til oppbygging av indekser. Flere strategier brukes for å optimalisere indekseringen. Det er mulig å organisere indekser slik at enkelte ord aksesseres raskere enn andre. Det kan lages statistikker over hvilke ord som brukes ofte i en dokumentsamling. De ordene som det er mange av og i tillegg er meningsbærende vil sannsynligvis bli oftere aksessert en de som sjelden brukes. Ved å gi ulike ord vekter etter hyppigheten kan mye brukte data aksesseres raskere enn de sjeldent brukte. Dette er spesielt aktuelt i naturlig språk der kun en liten del av ordene brukes mesteparten av tiden. Ved hjelp av vektene kan en optimalisere søkingen etter ord som ofte aksesseres.

Det regnes med at en fullstendig indeks gjennomsnittlig tar opp like mye plass som ordene som indekseres. Undersøkelser har vist at en ved fullstendig indeksering av alle ord får en indeks på mellom 50 og 300 % av den opprinnelige datamengden [Hollaar 92]. Indeksmengden kan reduseres på ulike måter. Kun deler av ordet indekseres, relaterte ord kan da få samme indeks. Alternativt kan en hashtabell brukes. Ved bruk av en egnet hash-funksjon kan fordelingen av ord i indeksene bli jevn.

Systemer gir støtte gir støtte til fritekstsøk i lagret tekstinformasjon. Den enkleste systemet for fritekstsøk er å sekvensielt sammenligne søkeordet tegn for tegn med databasen. Ulike algoritmer finnes for mer optimaliserte tekstsøk. I [Baeza-Yates 92] er det beskrevet en del av de mest brukte algoritmene.

Forbedret Maskinvare

Mye forskning har vært gjort på optimalisering av søketiden ved hjelp av spesialisert maskinvare. En hel rekke forslag har vært lagt frem til forbedrete prosessorer og koprosessorer fot utføring av databaseoperasjoner. Eksempler på dette er egne koprosessorer for strengsøking eller JOIN-operasjoner.

Den raske utviklingen av generell maskinvare har imidlertid gjort at det finnes få eksempler på spesialutviklete database-maskiner. Utviklingstiden er vanligvis for lang til at den greier å følge med den generelle utviklingen [Kambayashi 94]. Isteden benyttes ofte allerede eksisterende maskinvare til å sette sammen maskiner som utfører databaseoperasjonene raskere. Ved hjelp av arkitektur som optimaliserer bestemte operasjoner er det oppnådd betydelige resultatforbedringer [Hollaar 92], [Mitkas 94].

Lagringsmediet kan endres til et som er raskere aksesserbart. I Memorydatabaser legges hele eller deler av databasen i hovedlager. Innsetting av store mengder halvlederlager gjør dette mulig. Dette sørger for minimal aksesstid på dataene. Det har i de senere år vært forsket mye på denne arkitekturen. Spesielt har søkelyset vært rettet mot problemer med oppdatering av dataene [Garcia-Molina 92]. Store krav stilles til kontroll og sikkerhet i et flyktig lager.

Ved bruk av flere prosessorer muliggjøres parallellisering av søkingen. Flere alminnelige CPUer benyttes. En rekke algoritmer finnes for parallellisering av ulike databaseoperasjoner. Disse kan anvendes med en slik arkitektur.

2.3 MS160

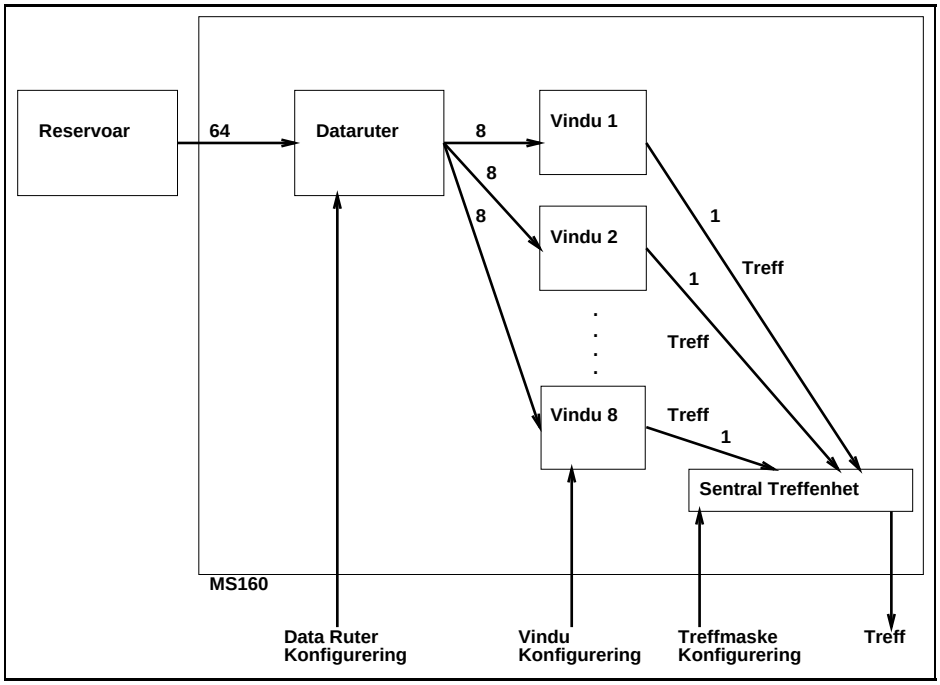
MS160-søkebrikke er et eksempel på en realisert bruk av spesialisert maskinvare arkitektur til hurtig informasjonsprosessering [Stephansen 93]. MS160 er en VLSI brikke med mulighet til parallellsøking gjennom data i hastigheter opp til 160 MB/s.

Som en del av PCkortet RAMSCAN kan den brukes til hurtig søking i databaser. I tillegg til søkebrikken inneholder kortet et reservoar av opptil 320 MB RAM. Søkingen foretas ved at alle data i hele eller deler av reservoaret sendes gjennom brikken. Avhengig av brikkens konfigurasjon genereres treff-posisjoner som siden kan leses fra reservoaret. Ved hjelp av den spesielle arkitekturen blir søkingen rask uten bruk av spesielle optimaliseringsmetoder. Dataene leses sekvensielt uten bruk av indekser. De mer optimale algoritmene for strengsøk tas heller ikke i bruk.

2.3.1 Arkitektur

Vinduer

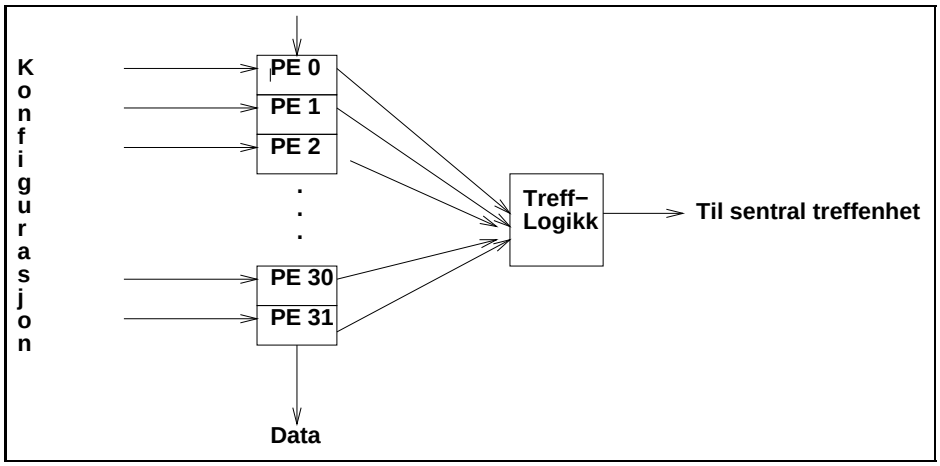
MS160 består av 8 vinduer for søkekriterier (se fig. 2.1). Hvert vindu består av 32 prosesseringselementer (PE). I hver PE ligger 2 komparatorer som innholder øvre og nedre grense for byteverdier i søkekriteriet. Hver komparator kan sammenligne en byte med en innkommende datastrøm. Ved bruk av alminnelige 7 eller 8 bits tegnsett tilsvarer dette et tegn. Det er mulig å konfigurere brikken til å kombinere flere PE til sammenligning av dataelementer på mer enn en byte. Spesielt er dette aktuelt ved sammenligning av tall. For testing av intervaller kan slike felter bestå av 8 byte og ved testing av likhet av 256 byte.



Figur 2.1: MS160-arkitektur

Hvert vindu kan dermed inneholde et søkekriterium på opp til 32 tegn. Kortere kriterier oppnås ved at intervallene i resten av elementene settes til hele tegnintervallet. Ved et 8

bits tegn er dette 0 til 255.



Figur 2.2: Et vindu

Under søkingen sendes datastrømmen gjennom vinduet. Dataene skiftes gjennom elementene slik at alle data er innom alle komparatorene. Vinduene har tilkoblet en trefflogikk-enhet. Dersom en verdi i datamengden er innenfor søkeintervallet rapporteres treffet til trefflogikken. Når alle de 32 elementene inneholder tegn som tilfredsstiller kriteriet rapporteres treff for hele vinduet til en sentral resultatlogikk-enhet.

Ruter

Data sendes inn i MS160 på en 64 bits synkron databuss. Dataene sendes først inn i en enhet for dataruting (se fig. 2.3) . Ruteren kan fordele datastrømmen til hvilken som helst av de 8 vinduene på 8 bit brede busser. Denne funksjonaliteten oppnås ved bruk av 3 nivåer av multipleksere. Hver multiplekser kan sende en 8 bits datastrøm til en av to mulige utganger. 8 av de 64 bitene kan med dette godt sendes til alle vinduene. Alle vinduer søker dermed i samme datamengde. For at hele datamengden skal sendes gjennom vinduene, må det da foretas 8 søk. Ruteren kan også lenke sammen flere vinduer ved å sende dataene etter tur. Det blir dermed mulig å definere søkekriterier på mer enn 32 byte.

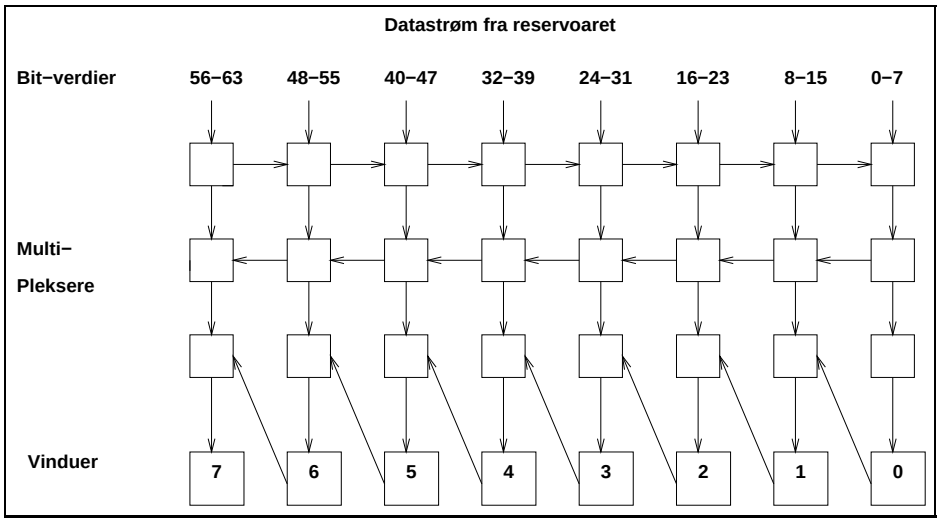
Sentral treff-enhet

Den sentrale treffenheten mottar treff fra individuelle vinduer. Hvilke som helst kombinasjon av treff i vinduene kan defineres til i gi treff for hele søket. Vinduene med treff blir definert som en adresse i en 256 bit programmerbar RAM. Ved å sette 1 eller 0 i forskjellige posisjoner i den såkalte treffmasken kan det bestemmes hvilke kombinasjoner som gir treff.

MS160 har to ulike søkemodus. Ved Count-søk teller treff-enheten treffene og returnerer til slutt det totale antallet. Ved Report-søk returneres adressene til alle treffene.

For hvert vindu kan det defineres en såkalt huskelengde(MatchLatency). Dette er antall posisjoner som vinduet fortsetter å generere treff etter at det egentlige treffet inntraff. Dette kan benyttes for å søke etter flere ord samtidig med en betingelse for hvor stor avstanden mellom dem skal være. Dersom ruteren sender samme datastrøm til flere vinduer, kan oppfyllelse av nærliggende søkekriterier defineres til å gi treff for søket.

Huskelengden og definering av treffmasken gir til sammen muligheten til alle former for



Figur 2.3: Ruter-nettverk

logisk søking. Ved at samme data sendes til flere vinduer kan alle ulike logiske kombinasjoner av treff for enkeltkriterier defineres til å gi treff. Det logiske søket skjer innenfor intervallet definert av huskelengden.

Konfigurering

All konfigurering av brikken skjer ved hjelp av en egen konfigureringsenhet. Denne skriver til ruteren vinduene og treff-enheten. En 8 bits databuss er forbindelsen til brukeren Brukeren skriver til 8 bits adresser på brikken.

I alt 828 byte kan konfigureres på brikken. Disse bestemmer søkekriterier, feltlengde, huskelengde, søkemodus, treffmasken og rutingen.

2.3.2 RAMSCAN

RAMSCAN er et PC-kort for en ISA-buss. I tillegg til MS160 inneholder den et reservoar av data på opp til 320 MB RAM. Dette muliggjør at data kan sendes til MS160 i store hastigheter grunnet den lave aksesstiden. Det er vanlig å organisere reservoaret i 8 like store kolonner. Innen kolonnene er dataene organisert i blokker på 1 kB. Som nevnt overføres dataene fra lageret til MS160 på en 64 bits databuss. Det hentes samtidig 8 bit fra alle kolonner. Dette muliggjør parallell søking i de 8 kolonnene.

2.3.3 Anvendelse

Brikken åpner for mange muligheter til søking i tekst. De mange formene for konfigurering gjør ulike anvendelser i forbindelse med informasjonsgjenfinning mulig.

Spesielt egnet har brikken vist seg til generelt fritekstsøk. Ved søking etter enkeltord kan det søkes parallelt i alle åtte vinduer. Dermed oppnås den maksimale søkehastigheten på 160 MB/s.

Ved bruk av flere søkekriterier kan kriteriene legges i ulike vinduer. Det søkes dermed samtidig etter alle kriterier plassert i ulike vinduer.

Dette kan være spesielt gunstig sammenlignet med logisk søk i vanlige databaseløsninger. Søketiden vil ikke endre seg i vesentlig grad i forhold til søking etter enkle kriterier. I tradisjonelle databasesystemer kan flere faktorer gjøre at søketiden blir lang dersom det søkes etter flere kriterier i et logisk søk. Spesielt gjelder dette hvis de enkelte søkeordene har høy selektivitet, mens hele uttrykket til sammen har en begrenset mengde treff. Kriteriene kan henviser til felter i flere tabeller i en relasjonsdatabase. Fordi søkingen etter elementene i det logiske uttrykket må foretas etter tur kan søketiden bli lang. JOIN-operasjoner må utføres mellom tabellene. Dette kan bli tidkrevende dersom reduksjonen av poster utfra kriteriet er liten før tabellene slås sammen. Med MS160 kan flere kriterier med høy selektivitet likevel få kort responstid.

Før å redusere søketiden kan en i tradisjonelle databaser brukes indekser for enkeltfelter. For søking etter flere kriterier kan det opprettes indekser for kombinasjoner av felter. For å muliggjøre effektiv søking etter enver kombinasjon av kriterier vil imidlertid dette kreve opprettelsen av en stor mengde indekser.

Brikken gir spesiell støtte til søking i strukturerte datamengder med konstant postlengde. Det kan defineres en postlengde slik at treff kun kommer ved slutten av hver post. Postlengden kan settes til en verdi mellom 1 og 256. Huskelengden vil bli regnet i hele poster. Dette vil sikre at treffmasker har treff fra vinduer i samme post.

2.4 Programmeringsgrensesnitt mot MS160

Det eksisterer programmeringsgrensesnitt mot MS160 og RAMSCAN. Flere av programmene finnes i ulike versjoner med mindre ulikheter. Grensesnitt er utviklet for ulike operativsystemer. Funksjonaliteten er også noe ulik. De mest komplette grensesnitt-programmene utvikles idag av distributøren av MS160 MicroWay MRT. Utviklingen av flere av dem har imidlertid startet i forbindelse med en hovedoppgave på NTH. Følgende systemer er idag tilgjengelige:

- **RAMSCAN Toolkit 1.0**
  Utviklet av MicroWay MRT som et forholdsvis lavnivå grensesnitt til brikken. Utviklet for MS-DOS og MS-Windows operativsystem.
- **MS160api Beta 2.1**
  Videreutvikling av RAMSCAN Toolkit 1.0 som ennå ikke er ferdig testet. Gir mulighet til søking med noe mer avanserte søkefunksjoner.
- **SCORE**
  Utviklet av Fredrik Kvamme i forbindelse med hovedoppgave på NTH og videreutviklet for Microway MRT. Gir mulighet til kontakt med MS160 via en klient-tjener arkitektur. Dette gjør brikken tilgjengelig via et datanettverk. Tjenerprogram er utviklet for en OS/2-plattform. Klientprogrammer er tilgjengelig både for unix og OS/2 Disse gir mulighet for noe mer høynivå-søking enn de to foregående. SCORE er for tiden under stadig utvikling og kan ikke regnes som ferdig testet.

2.5 Bautaprosjektet

Nasjonalbibliotekavdelinga i Rana har i samarbeid med Norsk Folkemuseum, Norsk museumsutvikling, NTH/IDT, UiO/Dokumentasjonsprosjektet og Riksarkivet inngått en avtale om å etablere en database for fotohistorisk materiale. Samarbeidet har fått navnet Bautaprosjektet. Hovedintensjonene med denne databasen er:

- Den skal inneholde digitaliserte fotografier med lav kvalitet. Dette skal gjøre det mulig for en bruker å få bildene frem på en skjerm. Bildene har lav kvalitet av hensyn til overføringshastigheten over datanettet og for å sikre seg mot ulovlig bruk av hentede bilder. Det vil også redusere det nødvendige lagringsvolumet.
- Den skal inneholde en rimelig mengde referanseopplysninger for fotografiene. Det skal være mulig å søke i disse opplysningene etter bilder.
- Den skal kunne nås via Uninett[1] og senere via ISDN. Dette vil øke tilgjengeligheten av fotografiene radikalt.
- Den skal kunne brukes av institusjoner omkring i Norge for lagring av denne type informasjon. Det skal dermed være mulig for brukere av basen å supplere samlingen med bilder og opplysninger om bildene.

Selve databasen skal etter planen utvikles og settes i drift ved NBR i Mo i Rana. I første omgang vil opplysninger fra Wilsesamlingen bli lagt inn. I det såkalte Wilse-prosjektet har NBR og Norsk Folkemuseum bestemt seg for at hele Wilsesamlingen skal overføres for lagring ved Nasjonalbiblioteket. Dette omfatter ca 110 000 negativer på glassplater og 30 000 papirkopier i album. Negativene skal avfotograferes og lagres på videoplate. Brukskopier skal sendes tilbake til Oslo. Som en del av Bauta-prosjektet vil filmene bli digitalisert og opplysninger om fotografiene bli registrert elektronisk.

Pr. desember 1994 er data om ca. 58 000 fotografier registrert og lagt inn i databasen. Det arbeides med den videre registreringen av Folkemuseets samlinger og innen påsken 1995 regnes det med at alle bildene i Wilse-samlingen er ferdig registrert.

Digitaliseringen av Wilse-samlingen er også i gang. Ca. 18000 bilder er nå digitalisert og lagt inn i databasen.

2.5.1 Oppbygging av databasen

Dagens database består utelukkende av bilder fra Wilse-samlingen. Ved hjelp av databasesystemet Oracle er både bilder og referanse-opplysninger lagret. Datamodellen er imidlertid generell og vil kunne brukes for lagring av tilsvarende bildesamlinger. Den er basert på et langvarig arbeid med å standardisere lagring av referanseopplysninger til norske bildesamlinger.

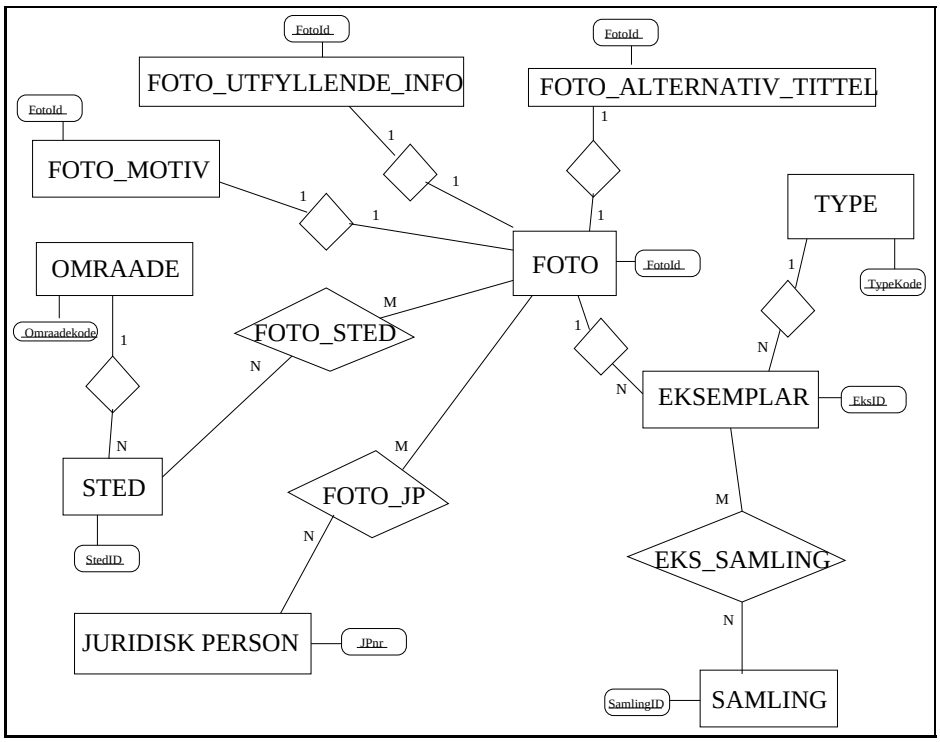
Med opprettelsen av Sekretariatet for fotoregistrering SFFR i 1977 startet arbeidet med å utvikle en standard for registrering av fotografi [Erlandsen 92]. Resultatet ble definisjonen av det såkalte fotokortet. Den ble basert på eksisterende registreringssystem ved norske og utenlandske kultur-institusjoner. Fotokortet inneholdt en standard for hvilke opplysninger som skulle registreres ved norske bildesamlinger.

I 1985 startet arbeidet med å utvikle et system for registrering av gjenstander og fotografier ved hjelp av datamaskiner. I et prosjekt initiert av NKKM og finansiert av NAVF utviklet NAVF's edb-senter for humanistisk forskning i Bergen et slikt program. Systemet var basert på databasesystemet Dataflex for tegnterminaler. Både registrering og søking var imidlertid relativt komplisert. Kun fagfolk med opplæring kunne bruke systemet.

Datamodellen for fotografidelen av prosjektet ble utviklet i samarbeid med SFFR. I all vesentlighet ble den basert på det eksisterende fotokortet. I dag er registreringsystemet tatt i bruk av mer 90 norske kulturinstitusjoner for registrering av fotografier og gjenstander. Mye arbeid er lagt ned i å skrive inn registreringsopplysninger.

[1] Det norske akademiske datanettet som er knyttet til det internasjonale Internettet.

I Bauta-prosjektets database vil det bli åpnet for at allerede registrerte referanseopplysninger i Dataflex på en enkel måte skal kunne legges inn. Datamodellen vil derfor i all vesentlighet være basert på den tidligere brukte standarden. Alle informasjonsfelter vil henvise til lignende felter i NKKMs datamodell. Ny teknologi åpner imidlertid for en utvidelse. At digitaliserte fotografier også vil bli lagret gjør at bildene også må sees som en del av datamodellen. Fordi Oracle er en relasjonsdatabase blir dessuten datamodellen mer fleksibel. Også i fotokortet og i Dataflex-systemet var det mulig med repeterende felter. Begrensninger i Dataflex reduserte imidlertid mulighetene. I registreringsskjemaet var det for eksempel ikke mulig å registrere mer enn 3 personer relatert til hvert bilde. Slike problemer løses ved bruk av Oracle. Av figuren sees en ER-modell av den eksisterende datamodellen. Den endelige datamodellen er imidlertid ikke fastsatt. Flere faktorer gjør at en vil vente med denne avgjørelsen.



Figur 2.4: "ER-modell" for deler av Bauta-basen

Datamodellen må ikke bare sees i forhold til den tidligere standarden, men også i forhold til hvordan registreringen faktisk har vært foretatt. Gjennom flere års registrering har det vist seg at den opprinnelige modellen bør modifiseres. Enkelte felter er nesten tomme mens det i andre er liten plass til den tilgjengelige informasjonen. Mye har forplantet seg videre i database-tabellene. Uheldigvis har disse problemene ført til at en del registrering av plassmangel er lagt i gale felter.

Selv om all registrering med Dataflex i teorien skulle følge de samme retningslinjene har det etter all sannsynlighet vært ulik lokal praksis. Ved innlegging av nye bildesamlinger må en derfor se om det blir nødvendig å tilpasse datamodellen til alternative registreringsmåter.

Noen enkeltfelter og tabeller vil spesielt bli sett på i forbindelse med en revurdering av lagringsstrukturen. Et eget motivfelt er blitt opprettet. I forbindelse avfotograferingen av negativene er ord som beskriver bildets innhold er lagt inn. En slik beskrivelse er imidlertid svært vanskelig. Det blir vurdert hvordan standardbetegnelser kan lages for denne registreringen. Til nå er registreringen vært foretatt av ufaglært personale. Presisjonen

av registreringen kan derfor ikke bli veldig stor. Det vil kreve store ressurser å la høyt kvalifisert personale foreta en grundig kategorisering av alle bildene. En tenker seg seg imidlertid at en oppgradering av registreringskvaliteten kan foretas gradvis gjennom bruk av systemet. Brukere vil over tid supplere databasen med opplysninger.

Motivfeltet består idag stort sett av en oppramsing av avbildete objekter og handlinger . Etter NKKM sin instruks skal det kun bestå av substantiv og verbalhandlinger. Eksempel på en motivbeskrivelse er: `mann, gutt, sildoljefabrikk`. Det har vært foreslått å også inndele bildene i mer generelle grupper. Ulike genre-betegnelser som kan brukes er f.eks. landskap, reklame eller portrett.

Registreringen av fotografiets sted vil det også bli sett på. Tabellene for stedregistrering er ikke tilpasset de innlagte dataene. Fastsettelse av et fotografis sted er dessuten forbundet med mange problemer. Administrative grenser kan være endret flere ganger siden fotografiet ble tatt. Hvilke stedopplysninger som er registrert er dessuten svært forskjellig. Presise adresser, gårdsnavn og upresise områdenavn kan alle forekomme. Det må bestemmes hvilke felter som skal opprettes.

Personregistreringen er lagt i en tabell kalt **juridisk_person**. Denne inkluderer reelle personer og juridiske personer som selskaper med relasjon til bildet. En egen rollekode avgjør om relasjonen er fotograf, avbildet eier osv.

Feltet **utfyllende_info** har vært brukt til all informasjon som det enten ikke er plass til eller som av andre årsaker ikke passer inn i de andre feltene. Ved bruk av en relasjonsdatabase er denne praksisen litt uheldig. Data i felter bør i størst mulig grad referere direkte til entiteter. Det vil bli vurdert om deler av denne informasjonen kan legges inn i mer egnete felter.

I norske kulturinstitusjoner foretas mange andre former for registreringer enn av fotografier. Mange av samarbeidsdeltakerne i Bauta-prosjektet arbeider også med å utvikle standarder for denne. Systemet for registrering av gjenstander utviklet av NKKM har en datamodell som ligner mye på den for fotografier. Norsk Folkemuseum er for tiden i gang med å registrere alle sine gjenstander etter denne standarden. En vil her stå over for mange av de samme problemstillingene som ved foto-registreringen. Det stilles de samme spørsmålene ved hvordan registrering av sted og beskrivelse av gjenstanden skal foretas. Det vil bli forsøkt å samordne den videre fastsettelsen av datamodellen for ulike typer registreringer.

2.5.2 Former for søking

Informasjonsgjenfinningen i Bautabasen foregår ved å søke i referanseinformasjonen. Mye av meningsinnholdet i denne er representert gjennom strukturen i datamodellen. Det vil være mulig å søke etter bestemte egenskaper ved bildet ved å søke i spesielle felter. Unntaket for dette er informasjonen som ligger i feltet **utfyllende_info**. Denne vil kun bestå av fritekst. Meningen av innholdet vil ikke være representert i datamodellen. Feltet kan innholde alle former for informasjon som beskriver bildet. Kun generelt fritekstsøk i hele feltet vil gi brukeren mulighet til å finne alle relevante data.

Søkingen kan ha elementer av både fakta og begrepsgjenfinning (se avsnitt 2.1.2). Enkelte typer data som avbildete personer kan gi grunnlag for faktagjenfinning. Brukeren kan entydig være interessert i portretter av en spesiell person. Andre typer søk som søking etter spesielle motiv er mer uklare. Dette er eksempler på begrepsgjenfinning. Det vil være vanskelig å avgjøre hvilke bilder som kjennetegnes ved et spesielt motiv. Mangelfull registrering vil komplisere dette ytterligere.

Uten programvare som er istand til å tolke selve bildeinformasjonen vil dette måtte foretas

av brukeren. Etter et søk må brukeren selv måtte avgjøre hvilke av de hentete bildene som er relevante. Denne prosessen vil som oftest kunne kalles begrepsgjenfinning. Selv etter et konkret søk etter en avbildet person kan den videre utvelgelsen være uklar. Utvelgelsen vil bestemmes av subjektive spørsmål som hvilke bilder som er gode eller hvilke bilder som passer som illustrasjon i en bestemt sammenheng. Også for brukeren selv vil disse kriteriene være uklare og vage.

2.5.3 Grensesnitt mot databasen

Det er idag utviklet to ulike grensesnitt mot bauta-basen.

Ved Nasjonalbiblioteket er det utviklet et grensesnitt mot databasen basert på Oracle-applikasjonen Oracle Forms 4.0. Dette gir et helhetlig grensesnitt både for registrering og søking i databasen. Til nå har all registrering vært foretatt med det tegnbaserte programmet Dataflex. Et mer brukervennlig grensesnitt vil lette registreringen. Oracle Forms har også muligheten til å hente data via nett. En ulempe med dette er imidlertid at folk som skal bruke det må ha Oracle-programvare installert på sin lokale maskin.

I forbindelse med et vårprosjekt på NTH utviklet Stein Langørgen og jeg selv et World-Wide Web-grensesnitt for søking i Bauta-basen [Bauta 94]. Det har et enklere grensesnitt enn Oracle Forms, men har den fordelen at det kan nås av mange flere. Alle med internett-tilkobling vil potensielt kunne nå databasen. Ulike klientprogrammer til World-Wide Web er allerede mye i bruk. Programmene kan dessuten lastes ned gratis fra allment tilgjengelige maskiner via FTP.

2.5.4 Erfaringer med grensesnitt

Både Oracle Forms og WWW-grensesnittet har gjort at databasen potensielt kan nås for et stort antall brukere. Til nå har imidlertid ikke databasen vært offentlig tilgjengelig. I en prøveperiode har det sommeren 1994 vært mulig å nå databasen via WWW-grensesnittet uten at dette har vært annonsert. Av hensyn til eventuelt sensitive opplysninger i databasen er imidlertid dette grensesnittet sperret. Norsk Folkemuseum har som eier av Wilse-samlingen valgt å stoppe tilgangen fordi man er usikker på om all informasjon som ligger i databasen kan gjøres allment tilgjengelig. Det vil bli foretatt en gjennomgang av bildematerialet og referanse-opplysninger nå med tanke på tilgjengeliggjøring over nett.

Responstiden er delvis avgjørende for brukervennligheten. I og med at det er mulig med netttilkobling vil både overføringshastigheten og selve søketiden i databasen bestemme denne. I WWW-prosjektet ble det diskutert hvor mye disse spilte inn. Med en idag vanlig båndbredde på 256 kbit/s og lavere vil for de fleste søk overføringstiden være størst. Spesielt gjelder dette når bilder overføres. For en del komplekse søk vil imidlertid søketiden være fremtredende. Dette gjelder særlig når mange join-operasjoner må foretas. Databasen inneholder mange tabeller slik at dette lett blir resultatet.

Oracle Forms-grensesnittet har hatt en del problemer med selve programvaren. Dette gjelder særlig en dårlig støtte til visning av bilder. Den gir dessuten dårlig støtte til søk i flere tabeller.

Kapittel 3

Problembeskrivelse

Som nevnt i innledningen er de viktigste spørsmålene jeg forsøker å svare på i rapporten både hvilke muligheter og hvilke begrensninger søkebrikken MS160 gir ved søking i databaser. Gjennom å implementere et program for søking i Bauta-basen vil jeg forsøke å svare på dette.

3.1 Funksjonalitet

For de fleste informasjonsystemer legger søketiden begrensninger på funksjonaliteten. Spesielle typer søk kan gi uakseptabelt lang søketid. Ved Stein Langørgen og mitt WWW-prosjekt viste vi hvordan dette også er et problem for dagens søking i Bautabasen ved hjelp av en relasjonsdatabase [Bauta 94].

Ved sin korte søketid kan bruk av MS160 åpne for en utvidet funksjonalitet. Dette gjelder både fordi den generelle søkingen skjer raskere, men også fordi søkealgoritmen kan optimaliseres bedre ved å utnytte søkebrikkens spesielle arkitektur. I oppgaven skal jeg utforske hvilke søkeformer som er realiserbare grunnet den raske søketiden. Utformingen av brukergrensesnittet vil avhenge av tiden ulike former for søk tar. Selv med bruk av MS160 vil enkelte former for søk kreve mye dataprosessering. Funksjonaliteten må tilpasses dette.

De tidligere grensesnittene til Bauta-basen har kun benyttet seg av tradisjonelle metoder for informasjonsgjenfinning. Bilder har blitt returnert dersom felter i referanseinformasjonen har vært like søkekriteriene. Brukeren vil imidlertid ofte ha uklare kriterier i søk etter de historiske fotografiene. De tradisjonelle metodene har nettopp vært kritisert for at de ikke gjør det mulig å finne alle relevante elementer uten klare kriterier. Jeg vil derfor spesielt se på hvilke metoder som finnes for å muliggjøre henting av flest mulig relevante bilder. Andre systemer for informasjonsgjenfinning gir utvidete muligheter og assistanse til formulering av søkekriterier. Ved å studere disse vil jeg komme med forslag til funksjonalitet som kan forbedre informasjonsgjenfinningen i Bauta-basen.

Søking med uklare kriterier vil i seg selv stille ekstra krav til lav responstid. Søkeprosessen vil ofte ha en iterativ karakter. Det vil måtte søkes mange ganger før brukeren er tilfreds med resultatet. Dette vil gjøre det spesielt viktig at en ikke må vente for lenge mellom hvert søk. Ved flere søk etter hverandre i Bauta-basen vil sannsynligvis både presisjonen og mengden relevante bilder øke. Sjansen er større for at en bruker vil søke videre dersom tidsforbruket ved de ekstra søkene er lavt.

Hvilke funksjonalitet som bør realiseres vil være avhengig av hvem som skal bruke databasen. I utgangspunktet vil dette være alle typer brukere. Ved at det legges opp til netttilkobling

vil et stort antall personer få en enkel tilgang. En tenker seg også at dataterminaler skal erstatte dagens manuelle bruk av bildesamlingene på de ulike institusjonene. Samlingene med historiske fotografier vil sannsynligvis ha en så stor allmen interesse at mange også vil ta den i bruk. Dette gjør at både meget kvalifiserte og helt uerfarene brukere vil være interessert i databasen. De uerfarne vil sannsynligvis ønske en så enkel funksjonalitet som mulig. De vil heller ikke ikke være så opptatt av at alle relevante bilder blir funnet. Mer faste brukere av samlingen vil derimot være interessert i all funksjonalitet som kan forbedre gjenfinningen. De vil bruke samlingen så ofte at de vil lære seg kompliserte søkeformer. I oppgaven velger jeg i første rekke å se på hvilke muligheter som finnes. En stor del uerfarne brukere vil muligens ikke være interessert i avanserte søkeformer. Et grensesnitt bør imidlertid utformes slik at helt enkel søking også er mulig. En del utvidet funksjonalitet vil dessuten sannsynligvis være til størst hjelp for de med manglende kjennskap til bildesamlingen. Den vil gi assistanse slik at det ikke er nødvendig med en god oversikt over innholdet i databasen.

3.2 Realisering

For at søke-funksjonalitet skal være mulig, må den kunne realiseres. MS160 gir muligheten til meget rask søking i store mengder av fritekst. I oppgaven vil jeg vurdere om det er mulig å overføre dette til rask søking i strukturerte datamengder. Som nevnt i avsnitt 2.3.3 gir MS160 støtte til søking i bestemte former for datastrukturer. Dersom alle poster er like lange er det mulig å konfigurere brikken til å benytte seg av det. Både ved definisjon av postlengde og huskelengde gir brikken støtte til dette. Det er imidlertid et spørsmål om hvor godt dette lar seg overføre til poster med variabel lengde.

En eventuelt utvidet funksjonalitet må også kunne implementeres. Det vil også settes krav til responstid. For stort tidsforbruk vil gjøre nytten av nye funksjoner mer begrenset. Mye av poenget med denne oppgaven var å vise hvilke søkeformer som ble muliggjort med MS160. Mange forslag til nye søkeformer vil sannsynligvis kunne utføres like effektivt i alminnelig programvare. I oppgaven vil det derfor bli studert i hvor stor grad nye søkeformer vil bli forenklet ved hjelp av spesialisert maskinvare som MS160.

Det vil settes begrensninger for hvor mye ny funksjonalitet som kan realiseres i min demonstrasjon. Jeg har begrenset tid til rådighet og en del utvidelser kan kreve store ressurser. En oppgradering av kvaliteten på referanseopplysningene ville kunne muliggjøre nye søkeformer. Et slikt arbeid vil imidlertid gå utenfor både mine tidsrammer og kvalifikasjoner. Jeg vil likevel påpeke på hvilke områder utvidelser kan gjøres og antyde hvorvidt bruk av MS160 ville være en fordel.

Jeg vil avgjøre hvordan grensesnittet mot selve brikken skal realiseres. Som beskrevet i forrige kapittel finnes det ulike applikasjoner for programmering mot brikken. Valget mellom disse er blant annet avhengig av hvilke operativsystem som jeg vil at programmet skal kjøre under. De ulike toolkitene har også litt ulik funksjonalitet. Søkebrikken er idag kun tilgjengelig for installasjon på PC.

Søkebrikken MS160 har kun eksistert i noen få år. Likevel har en rekke prosjekter, hovedoppgaver og annen forskning vært gjort på ulike anvendelser av brikken. Flere av disse har fokusert på hvordan søking i tekstlig informasjon best kan utføres [Bjåstad 93],[Grøvlen 93], [Løkken 94], [Løkken 93], [Bøen 93], [Eide 93] og [Stephansen 92]. Mye forskjellig programvare har også vært utviklet for å lette tekstsøking. Disse har gjerne et programmeringsmessig mer høynivå grensesnitt enn de grunnleggende toolkitene direkte mot brikken. Jeg vil derfor studere andres arbeid på området.

I oppgaven vil jeg se på ulike løsninger av grensesnittapplikasjoner mot brukeren. Dette

kan være med på å avgjøre hva slags funksjonalitet som kan realiseres. En mulighet er å forsatt basere seg på WWW eller Oracle Forms. For min demonstrasjon vil imidlertid Oracle Forms ikke være tilgjengelig. Løsninger kan også baseres på rene vindussystemer som Microsoft Windows. Det vil bli samarbeidet med oppdragsgiverne for å klargjøre hvilke ønsker de har. Det må for eksempel avgjøres hvor ønskelig det er at MS160-applikasjonen skal kunne nåes over nett. Dette vil bestemme om det er nødvendig med klient og tjener programvare.

Konsekvensen av nett-tilknytning vil bli diskutert. Som tidligere skrevet ble det i WWW-prosjektet konstatert at for de aller fleste søk var overføringstiden for standard nett-forbindelser langt større enn selve søketiden i databasen. Mesteparten av tiden gikk med til overføring av bilder. Dette stiller et stort spørsmål med verdien av rask søking med MS160 i det hele tatt. Dersom bedring av søketiden kun har liten innvirkning på den totale responstiden er bruk av spesialisert søke-maskinvare av mindre interesse. Jeg vil se på hvordan forholdene rundt disse spørsmålene er ved bruk av min realiserte løsning for søking i Bautabasen.

3.3 Andre databaseløsninger

MS160 konkurrerer med en rekke andre medier og applikasjoner for lagring og henting av data og informasjon. Som nevnt i avsnitt 2.2 finnes det en rekke løsninger for informasjons-prosessering. Flere av alternativene er mye mer spesialisert for hurtig søking enn et alminnelig relasjonsdatabasesystem. Dagens Oracle database med Bautabasen kan muligens også optimaliseres mer for søking. I min oppgave vil jeg likevel i første rekke sammenligne MS160-grensesnittet med dagens database. Jeg vil ikke ha mulighet til å foreta en reell vurdering av virkemåten til MS160 sammenlignet med andre systemer enn denne.

Kapittel 4

Datalagring

Med bruk av MS160 og RAMSCAN til søking i Bauta-basen er det nødvendig å legge deler av informasjonen i databasen ut på reservoaret. I dette kapittelet vil jeg beskrive de løsningsalternativer jeg stod overfor, og de løsninger jeg har valgt for hvordan datalagringen skal gjøres.

4.1 Løsningsalternativer

De er ønskelig å minimalisere datamengden i reservoaret. All søking er sekvensiell slik at en mindre datamengde vil gi lavere søkehastighet. Samtidig er det ønskelig at mest mulig av dataene er søkbare.

4.1.1 Informasjon

Den primære datakilden i Bautabasen er de digitaliserte fotografiene. En del av referanseopplysningene må også sies å utgjøre den originale datamengden. Opplysninger om sted, avbildete personer og dato er registrert samtidig med fotografiet. Disse lar seg ikke på noen enkel måte avlede av selve bildet. Administrative opplysninger om bildets ulike registreringer er også grunnleggende data.

Under konstruksjonen relasjonsdatabasen ble det opprettet en del deskriptorer. Den viktigste er motivfeltet. Manuelt blir alle bilder gjennomsett og forsøkt beskrevet med noen nøkkelord. Hovedårsaken til dette er at det ikke finnes noen automatisert og effektiv metode å søke i selve bildeinformasjonen. Med dagens teknologi er dette vanskelig. Det er utviklet systemer for enkel bildegjenkjenning. Bildene i Bauta-basen er imidlertid for uensartet og alminnelige søkekriterier er for spesielle til at til at søking i bildene er praktisk mulig. Med bruk av MS160 vil dette ikke bli endret. Isteden søkes det i motiv-feltet. Som beskrevet i avsnitt 2.2.1 er det en rekke problemer forbundet med en slik representasjon.

For å forenkle søkingen kan eventuelt andre deskriptorer også opprettes og legges i reservoaret. Som nevnt i avsnitt 2.5 vurderes det hvordan stedfeltet og motivfeltet kan utvides. En mulighet er å gjøre dette ved bruk av ulike oppslagsverk. Det finnes oppslagsverk som kan gi utfyllende informasjon om geografisk plassering ut fra data om sted eller gårdsnavn. Det kan gis opplysninger om nåværende og tidligere administrative inndeling. Dersom disse gjøres elektronisk tilgjengelige vil deskriptorene kunne genereres automatisk. Med synonymordbøker kan motivfeltet utvides til mer å inkludere meningen til ordet. I avsnitt 2.2.1 beskrev jeg hvordan utenlandske systemer vil kunne foreta dette automatisk. For norsk språk er dette ennå ikke tilgjengelig.

Det må avgjøres hvilke data som skal legges i reservoaret. Ved at grensesnittet er beregnet for søking vil det ikke nødvendig å legge inn alle dataene. For en alminnelig bruker vil en del registeropplysninger ikke være interessant. Opplysninger om plassering o.l. vil det av sikkerhetshensyn ikke være aktuelt å presentere. Personopplysninger kan innholde sensitiv informasjon som av personvernhensyn ikke kan offentliggjøres. Det vil da heller ikke være behov for å lagre disse. En del opplysninger er det muligens ikke så interessant å søke i, men kan være passende å få presentert sammen med bildet. For søkingen vil det ikke være nødvendig at disse dataene ligger i reservoaret. Aksessen blir likevel enklere ved at de logisk ligger sammen med de andre foto-opplysningene. Den totale datamengden blir imidlertid større.

Selve bildet må også lagres. Dersom det legges sammen med referanseopplysningene i reservoaret vil dette vesentlig øke datamengden og søketiden. Med dagens størrelse på reservoaret vil det heller ikke være plass. Alternativt kan det legges i et bakgrunnslager. Aksesstiden blir større, men likevel liten sammenlignet med tid for dekomprimering. Erfaringene fra vårprosjektet sier at tiden for dekomprimering av JPEG-bildet vil dominere. Bildene kan godt hentes ut fra den primære lagringen i relasjonsdatabasen. Dette vil redusere diskbehovet.

4.1.2 Struktur

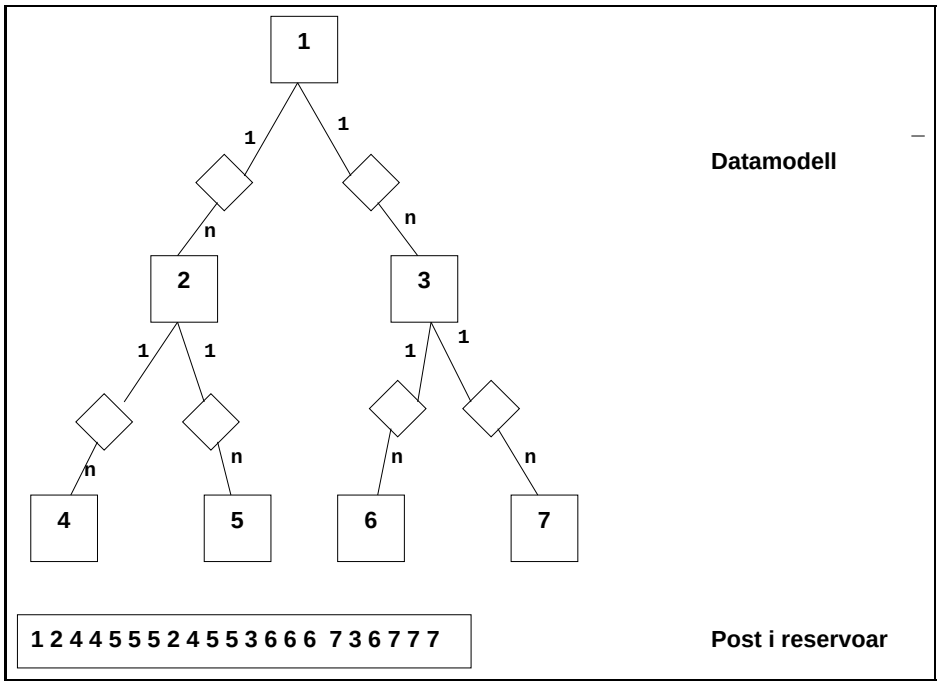
I dagens relasjonsdatabase er dataene lagt i flere tabeller. Mellom tabellene er det relasjoner. Relasjonene kan være en til en, en til mange eller mange til mange. Det vil teoretisk sett være mulig å beholde en slik struktur på reservoaret. Søking kan begrenses til bestemte deler. JOIN operasjoner vil kunne utføres ved å simulere funksjonaliteten til et relasjonsdatabasesystem. Dette kan imidlertid bli tidkrevende. En vil ikke utnytte utnytte den raske sekvensielle søkehastigheten.

Tidligere forsøk på å utlegge lignende datastrukturer har alltid lagt alle data i en stor tabell[Løkken 94]og [Bjåstad 93]. Hver post vil inneholde alle data relatert til en grunnenhet. f.eks fotografi. Alle data fra alle tabeller vil da kunne finnes ved å gå gjennom hele datamengden en gang. Dette gir en hierarkisk struktur som vist i figur 4.1. Alle relasjonene er nøstet ut. En ulempe med en slik representasjon er at en vil få en viss redundans. Data kan vil bli duplisert ved utlegging av mange til en og mange til mange relasjoner. Det totale datavolumet øker. En eventuell oppdatering av data i en slik struktur ville bli vanskelig. Men en egen oppdateringsfunksjon vil neppe være aktuelt i forbindelse med Bauta-basen. Alternativt kunne dataene legges i en tabell ved alminnelige JOIN-operasjoner. Dette ville imidlertid føre til redundans også ved en til mange relasjoner. Alle opplysninger om et fotografi ville heller ikke lenger ligge sammen.

4.1.3 Postformat

Det meste av informasjonen i Bautabasen utenom bildene består av tekst. En del opplysninger om årstall o.l. er tall, men disse utgjør bare en liten del. Hvert tegn i teksten kan representeres ved en byte i reservoaret. Tall kan enten representeres med et tegn pr. siffer eller i binær form. Ved binær form spares det plass. For hver byte kan 256 ulike kombinasjoner representeres og ikke 10 som ved tekst. Konverteringen blir imidlertid mer komplisert.

For å muliggjøre søking som ikke skiller mellom store eller små bokstaver kan tegnrekkefølgen endres fra det vanlige ASCII-formatet. Store og små bokstaver settes ved siden av hverandre. Øvre og nedre grense i kriterievinduet settes til hver av de to tegnene. Enkelte programmeringsgrensesnitt gir støtte til automatisk omlegging til en slik



Figur 4.1: Eksempel på hierarkisk struktur

tegnrekkefølge.

Ulike metoder kan brukes for å skille mellom feltene i postene. Dersom feltene har en konstant lengde trenger en ikke spesielle merker. I Bautabasen er det imidlertid både felter av fast og variabel lengde. Alle felter kan settes til den maksimale feltlengden, men dette vil føre til svært stor ekstra plassbruk.

Tidligere søking med MS160 i ustrukturerte data har alltid benyttet seg av spesielle tegn for å markere starten av felter. Slike tagger kan defineres blant de mange ubrukte posisjonene i standard tegnsett. I Bautabasen benyttes det internasjonale 8-bits tegnsettet ISO8859-1 (Latin-1). ISO8889-1 har en rekke byteverdier som ikke er definert fordi de ofte brukes som kontrolltegn. Dette kan gi unike starter på poster og felter i postene. Med bruk av binær representasjon av tallene kan det imidlertid bli vanskelig å skille ut taggene. [Løkken 94] har imidlertid foreslått en metode for å minske sannsynligheten for slike feil.

Taggene kan også brukes til å muliggjøre søk fra starten i enkeltfelt. Ved at taggen plasseres først i kriterieordet vil det kun gis treff i bestemte felt. Taggen kan også plasseres foran alle ord i felt for å muliggjøre søk etter alle mulige ord. For å muliggjøre dette må det også opprettes en slutt-tag som markerer slutten på feltet.

Begrensningen med bruk av taggene til søking er at de kun er mulig å søke fra starten av ord. Sturle Dahl har i forbindelse med en sommerjobb utført for Microway MRT, foreslått å bruke egne tegnsett for spesielle felter. Ubrukte byteverdier kan få lagt inn bokstavalfabeter for bestemte tegn. Med store og små bokstaver utgjør alfabetet nesten 60 tegn. Det er knapt nok plass til et ekstra alfabet dersom tagger også brukes. Eventuelt kunne all tekst-informasjon reduseres til kun å inneholde små bokstaver. Dette vil gi plass til en rekke alfabeter. Søkingen ville bli like effektiv, men dataene ville ha mistet mye informasjon for senere utskrift. Eventuelt kunne en kun bruke standard ASCII tegnsett. Dette ville frigjøre de 128 største verdiene. I Bautabasen idag brukes det imidlertid tegn som ikke er en del av standard ASCII.

Rekkefølgen av datafeltene kan bli avgjørende for hvor vellykket søket er. Det er kun ved å definere huskelengden til passende verdier at det er mulig å søke parallelt etter flere ord i samme post. Det er imidlertid et problem at fordi postlengden er variabel kan dette gi treff for ord i forskjellige poster. For i størst mulig grad å unngå dette har det tidligere vist seg gunstig å plassere korte konstante felter i slutten av posten og de lange variable i begynnelsen. Huskelengden kan da settes lav for de korte og konstante feltene og sikre at disse ikke gir treff for gale poster.

4.2 Valg av løsning

Jeg velger å legge alle interessante referanseopplysninger i reservoaret. Eventuelle opplysninger som det ikke vil være interessant å søke i vil kun utgjøre en liten del av den totale datamengden. Selve bildet lar jeg fortsatt ligge i relasjonsdatabasen. Det er naturlig nok ikke plass i reservoaret. Under en demonstrasjon kan bildene eventuelt hentes fra Mo i Rana.

Jeg vil ikke ha mulighet til å opprette helt nye deskriptorer. Jeg vil kun påpeke at det kan opprettes. Stedfeltet vil jeg modifisere litt. Ved hjelp av kommunenummeret kan alle fylkene finnes. Jeg legger derfor også fylket inn i reservoaret for å muliggjøre søking etter dette. En del av bildene har andre land registrert i stedfeltet. Jeg legger derfor inn Norge på de andre for å sikre konsistens.

Etter samtaler med NBR og Norsk Folkemuseum har jeg lagt følgende datafelter inn i reservoaret:

- EksemplarID
- Bilder1ID
- Bilder16ID
- Repronummer
- Datokommentar
- Tittel
- Alternativ
- Utfyllende info
- Rollekode
- Navn
- Motiv
- Stedkode
- Adresse
- Områdepresisering
- Kommunekode
- Fylke
- Land
- Typebetegnelse
- System
- Klassifiseringskode
- Invariansnummer
- Aksesjonsnummer
- Datering, fra
- Datering, til
- Registreringdato
- Innkomstår

Jeg har valgt å bruke en hierarkisk struktur for postene. Alle data som hører til et fotografi ligger sammen med bildet. Den reelle datamodellen har ikke en til mange eller mange til mange relasjoner på mere enn ett nivå. Strukturen blir derfor enkel. Hierarkiet består i at enkelte av feltene i hver post er repeterende.

Jeg velger å representere både tall og tekst som tegn. Tallene vil uansett utgjøre en liten del av teksten. Kun noen få byte spares dersom de representeres binært.

Jeg bruker tagger for å markere starten og slutten av felter. Dette sikrer en minimal lagring av de variable feltene. q Jeg bruker også taggene til å muliggjøre søking i enkeltfelter. Dette betyr at jeg velger bort muligheten til å ha egne tegnsett for enkeltfelter. Med bruk av ISO8859-1 er det liten plass til ekstra tegnsett. Taggene bruker en stor del av de ubrukte plassene. I enkeltfelter må det da alltid søkes fra begynnelsen av ordet. Fritekstsøk i hele posten blir likevel mulig. Den samme taggen legges i begynnelsen av hvert ord i felter som

består av tekst med flere ord. Etter norske rettskrivningsregler skal det være mellomrom foran alle ord. Jeg bytter derfor ut mellomrom med en tag. For å markere slutten av felter med flere ord og repeterende felter bruker jeg en egen slutt-tag. Denne er felles for alle felter. For å lette lesingen av hele posten har jeg også en slutt-tag for denne. Starttaggen er identisk med taggen til EksemplarID som alltid er der.

Enkelte relaterte felter gir jeg tagger i rekkefølge for å muliggjøre søking i flere felter. Dette er motivrelaterte felter og stedrelaterte felter.

Jeg legger alle de korte feltene til slutt. De variable og lange legger jeg i begynnelsen. EksemplarID legger jeg likevel først da denne ikke er interessant å søke i sammen med andre felter.

Av tabell 4.1 fremgår navnet på alle feltene og hvor i databasen det er hentet. Tagnummeret og om feltet er repeterende er markert. Avsatt maksimal feltlengde er den plassen som er satt av til feltet i Oracle-databasen. Har multiplisert denne med n for de repeterende feltene. Faktisk maksimal feltlengde er den maksimale lengden for feltene i referanseinformasjonen som i dag er lagt inn i databasen. For de repeterende feltene inkluderer lengden alle repetisjonene.

Rekkefølgen av taggene er identisk med rekkefølgen av feltene i posten. Bakgrunnen for rekkefølgen er en kombinasjon av avsatt maksimal feltlengde i database og faktisk maksimal feltlengde blant de innlagte dataene. Har ikke merket lengde på bildeidentifikatorene da disse ikke er nødvendig å legge inn. De vil alltid være identisk med EksemplarID. Taggene brukes for å markere om bildene er lagt inn eller ikke. En del tallfelt har ikke egentlig avsatt lengde De refererere da bare til den innlagte maksimale lengden.

Sist i tabellen er taggene som markerer slutten på posten og slutten på feltet. ASCII-tegn 10 er forøvrig identisk med det vanlige kontrolltegnet for linjeskift.

Av alle feltene er det egentlig bare EksemplarID som er obligatorisk. Legger likevel dato i alle felter. Dersom en egen datering ikke er registrert settes intervallet 0000-9999.

Feltnavn	Databasefelt	Tagnr	Repe-terende	Avsatt maks. feltlengde	Faktisk maks. feltlengde
EksemplarID	FOTO.FotoID	132		-	6
Bilder1ID	BILDER_1.bildeid	133		-	-
Bilder16ID	BILDER_16.bildeid	134		-	-
Repronummer	EKS_REPRONR.Repronr	135		-	6
Datokommentar	FOTO.Datering_kommentar	136		100	0
Tittel	FOTO.Tittel	137		100	0
Alternativ tittel	FOTO_ALTERNATIV_TITTEL. Alt_Tittel	138		100	0
Utfyllende info	FOTO_UTFYLLENDE_INFO. Utfyllende_info	139		1000	311
Rollekode	FOTO_JP.Rollekode	140	X	2*n	8
Navn	JURIDISK_PERSON.Etternavn	158	X	100*n	111
Motiv	FOTO_MOTIV.Motiv	141		100	64
Stedkode	FOTO.relasjonskode	142			2
Adresse	STED.Adresse	143		40	26
Områdepresisering	STED.Omraade_presisering	144		40	10
Kommunekode	OMRAADE.Omraade_kode	145		4	4
Kommune	OMRAADE.Omraade_navn	146		100	15
Fylke	OMRAADE.Omraade_navn	147		100	17
Land	OMRAADE.Omraade_navn	148		100	25
Typebetegnelse	TYPE.Typebetegnelse	149		100	21
System	FOTO_KLASS.System	150	X	2*n	8
Klassifiseringskode	FOTO_KLASS.Klasskode	151	X	10*n	40
Invariansnummer	FOTO.Invnr	152		40	18
Aksesjonsnummer	FOTO.Aksesjonsnr	153		20	0
Datering, fra	FOTO.Datering_fra	154		4	4
Datering, til	FOTO.Datering_til	155		4	4
Registreringdato	FOTO.Registrerings_dato	156		11	11
Innkomstår	FOTO.Innkomst_aar	157		4	4
Slutt Post		10		-	-
Slutt Felt		130		-	-

Tabell 4.1: Felter i reservoaret

Kapittel 5

Brukergrensesnitt

I dette kapittelet vil jeg beskrive de løsningsalternativer jeg stod overfor, og de løsninger jeg har valgt for brukergrensesnittet. Deretter beskriver jeg skjermbildene som inngår i et MS160-grensesnitt mot Bauta-basen.

Ved brukergrensesnittet bestemmes funksjonaliteten til søkingen. Funksjonaliteten i min løsning er helt avhengig av det som er realiserbart med bruk av MS160. Kun bestemte former for interaksjon med databasen vil være mulig.

Jeg vil derfor i valget av min løsning for brukergrensesnitt også studere dette med henblikk på hvordan dette kan implementeres. Søkeformer som på en eller annen måte drar fordel av MS160 sin spesielle arkitektur vil bli foretrukket. Det vil imidlertid også bli lagt vekt på de ønsker en bruker til funksjonalitet. Dersom det senere viser seg at dette ikke lar seg implementere på en god måte vil det være en god måte å få avklart hvilke begrensninger brikken har.

5.1 Løsningsalternativer

5.1.1 Funksjonalitet

Formålet med brukergrensesnittet er på en best mulig måte å gjøre det mulig for en bruker å finne frem til ønskete fotografier. Ved hjelp av all tilgjengelig informasjon bør systemet tilrettelegge denne informasjonssøkingen. Et søkeprogram vil på grunnlag av data som beskriver de ønskete fotografiene finner frem til bildene. I kapittel 4.2 har jeg beskrevet hvordan jeg vil legg dataene ut på reservoaret. Søkingen vil derfor være begrenset av hvor godt denne informasjonen representerer bildene.

I avsnitt 2.2.2 er det beskrevet en del av den vanligste funksjonaliteten til eksisterende systemer for informasjonsgjenfinning. Det er også beskrevet hvordan nyere forskning prøver å danne grunnlaget for mer avanserte søkemetoder. Jeg vil gå gjennom disse punktene og se hvordan MS160 kan tilpasses denne funksjonaliteten og se hvilke muligheter den åpner for.

Indeksering

Viktig i mange systemer er muligheten til å indeksere. Dette vil gi raskere aksess til dataene. En av fordelene med MS160 er at en slik indeksering ikke er nødvendig. Selve dataene kan raskt aksesseres uten at det er nødvendig å gå via en indeks. Søking av ikke

indeksert informasjon er ikke noe problem. Diskkapasitet til indekser spares.

Bruk av indekser har imidlertid også funksjonelle fordeler. Synonymer kan henvise til de samme dataene. Med søking i selve datamengden blir dette vanskelig. Lister over synonymene og antall forekomster av et ord kan genereres sammen med selve indekseringen. Dette kan være til hjelp under søkeprosessen.

Det er ikke noe i veien for å bruke en slik funksjonalitet med MS160. Noen spesiell støtte gis det ikke til selve indekseringen. Lister over forekomsten av ord i spesielle felt kan imidlertid genereres uavhengig av en egentlig indeksering. Disse kan sorteres alfabetisk eller etter forekomst.

Søking i enkeltfelter

Med opprettelsen av tagger i datalagringen kan det søkes i enkeltfeltene. Begrensningen er ar det vanskelig kan søkes fra annet enn fra starten av ordene. En slik søking i de enkelte feltene er en grunnleggende funksjonalitet i databasesystemer. De tidligere grensesnittene til bautabasen har utelukkende basert seg på dette. For en slik søking må flere valg foretas. Hvilke felter skal det være mulig å søke i og hvordan skal skal søkingen spesifiseres av brukeren.

Til nå har søkingen foregått ved at brukeren kan fylle ut flere felter i en søkemeny. Hvert felt i søkemenyen vil søke i et eller flere databasefelter etter søkekriterier. Fylles flere felter ut kreves det at fotografiet skal oppfylle alle kriteriene. Dersom kriteriene søker i databasefelter fra flere tabeller har dette ofte resultert i lang responstid. Av hensyn til søketiden har det defor vært uheldig å søke i mange felter på en gang. Ved mulighetene til parallellisering av søkingen med MS160 kan dette bli bedret.

Grensesnittene mot Bautabasen benytter seg idag av følgende søkekategorier.

- Motiv
- Sted
- Tidsdrom fra og til
- Avbildet Person
- Fotograf

Søkingen i hver kategori foregår i et eller flere felter i databasen. Som et alternativ til disse kategoriene kan en tenke seg søking i alle felter på en gang.

Boolske operatorer

Andre MS160-applikasjoner har gjerne benyttet seg av muligheten til logisk søking. Gjennom muligheten til parallellisering har brikken gitt god støtte til dette. Dette vil også være aktuelt for bautabasen. Også for en relasjonsdatabase er dette mulig, men kan fort bli tidkrevende. Den logiske operatoren OG er vanlig å bruke. Den benyttes ved at det kreves at alle søkekriterier skal oppfylles. Operatoren ELLER vil imidlertid også være nyttig. Flere ord kan beskrive den samme typen bilder. Ved ELLER kreves det at kun ett av kriteriene oppfylles. Samme funksjonalitet kan oppnås ved å søke flere ganger, men da vil bildene ikke opptre samlet. De samme bildene kan dessuten opptre i flere søk. Operatoren IKKE er spesielt aktuell dersom få aktuelle bilder opptrer blant mange uinteressante. Dersom

det eksisterer noe felles ved de uinteressante kan disse utelukkes. En bruker vil ofte først bli oppmerksom på gode kriterier for IKKE operatoren etter å ha studert trefflisten etter et innledende søk.

Ved hjelp av ulike konfigureringer av treffmasken er det mulig å definere en hvilken som helst logisk kombinasjon av søkekriteriene til å gi treff. Dette gjør en hvilken som helst bruk av parenteser mulig. Begrensningen er at kriteriene kun kan legges i 8 vinduer. Flere enn 8 kriterier er mulig dersom flere legges i samme vindu. Avstanden mellom kriterier i samme vindu må da være liten og konstant. Begge kriteriene må dessuten være oppfylt samtidig for å gi treff.

Resultat tidligere søk

Resultat av tidligere søk kan defineres til å utgjøre en del av et nytt logisk søk. Med alminnelig bruk av reservoaret kan en slik funksjonalitet implementeres ved at de nye søkekriteriene legges sammen de fra forrige søk. Noen egentlig søking i de tidligere søkeresultatene er dette egentlig ikke. Hele datamengden søkes igjennom. Noen tidbesparelse vil ikke oppnås.

Alternativt kan det neste søket foretas i programvaren. Informasjonen i postene som gav treff må da hentes ut. Ved et lite antall treff vil dette sannsynligvis være mer effektivt enn bruk av MS160 og reservoaret. Det er imidlertid i første rekke ved mange treff at en videre søking er interessant. Store mengder plass må allokeres til dataene i minnet. Søkehastigheten i CPU kan under all omstendighet ikke kunne sammenlignes med hastigheten til MS160. Ved mange treff vil sannsynligvis bruk av MS160 være mest effektivt.

Nærliggende data

Ved sin bruk av huskelengden gir MS160 god støtte til søk etter nærliggende ord. En slik funksjonalitet er vanligvis ikke interessant i en strukturert datamengde. At avstanden er kort betyr ikke nødvendigvis at dataene er relaterte. I Bautabasen vil imidlertid bildene være sortert etter negativnummer. Bilder med kort avstand kan derfor ha et lignende motiv. Muligheten til andre avanserte søkeformer svekker imidlertid nødvendigheten av en slik funksjonalitet.

Trunkering

MS160 åpner også for trunkering, men setter også en del begrensninger. Alltegn kan markeres ved å sette 0 og 255 i søkeintervallet til enkelttegn. Intern trunkering er vanskelig å få til dersom ikke antall tegn i det interne alltegnet kjennes. Eks. fisk*båt vil ikke kunne benyttes dersom antall tegn i * ikke kjennes. Eventuelt kan fisk og båt brukes i flere vinduer, men dette gir ikke sikkerhet for at de er samme ord. Dette må eventuelt kontrolleres siden siden.

Med bruk av tagger vil som nevnt søking i enkeltfelter være fra starten av feltet. Kun høyretrunkering blir da tilgjengelig. Venstretrunkering blir ikke mulig. Søkes det i hele posten uavhengig av felt blir også venstretrunkering mulig.

Søking i intervaller

Gjennom bruk av øvre og nedre grense for søkekriterier er det mulig å søke i intervaller. Spesielt aktuelt vil dette være for tall. Det viktigste behovet for dette i Bautabasen er søking på tidsrom for fotografier.

Lagring av søkestrategi

En MS160 applikasjon kan godt lagre tidligere søkestrategier. Dette vil imidlertid ikke bruke noe spesielt av MS160 sin funksjonalitet. Søkekriteriene kan lagres for siden å hentes ved nye søk.

Utvidet boolsk logikk

Fordi søkekriteriene i Bautabasen ofte er uklare kan det være en fordel å få en rangering over hvor godt treffene tilfredsstiller søkekriteriene. [Radecki 88] beskriver en hel rekke forslag til hvordan denne rangeringen kan gjøres. Dette vil som nevnt utvide den tradisjonelle boolske logikken ved at en ikke lenger har sannhetsverdier for treff på bare 0 eller 1.

Den enkleste måten for utvidet boolsk logikk er å rangere treffet etter antall oppfylte kriterier i ELLER-uttrykk. MS160 ville kunne støtte en slik funksjonalitet. Treffmasken kan konfigureres til kun gi treff dersom et bestemt antall kriterier er oppfylt. Det kan dermed søkes flere ganger for å få en rangering. Alternativt kan det settes et minimum for antall kriterier som må være oppfylt og gjøre den videre rangeringen i programvaren.

Med ulik bruk av trunkering kan det gis grader av treff for et enkelt søkekriterium. Ord med lignende stamme vil ofte være relaterte. Treff kan dermed rangeres etter hvor like ordet og søkekriteriet er. En slik form for mønstergjenkjenning kan også brukes dersom det er tvil om hvordan et ord staves. Nærliggende ord kan gi delvis treff. Søking i intervaller kan også brukes til en å gi resultatene en rangering. Treff kan rangeres etter hvor nærme en tallverdi i et felt er en optimalverdi. Alt dette kan imidlertid bli kostbart med MS160. Metodene vil være begrenset av at MS160 har et øvre og et nedre intervall. Dersom en del av rangeringen foretas i programvaren kan imidlertid brikken kunne brukes til en grovsortering av aktuelle kandidater.

Alt dette er imidlertid enkle former for utvidet boolsk logikk. En kan godt tenke seg mer avanserte metoder til bruk med MS160. Oppfyllelse av enkeltkriterier kan godt gis andre sannhetsverdier enn 0 eller 1. Det kan brukes ulike vekter for kriteriene. Ulempen med en slik funksjon at den ikke kan benytte seg av MS160 sine muligheter til parallell logisk søking. Denne kan kun gi støtte til vanlig boolsk logikk. Det kan ikke sendes andre verdier enn 0 eller 1 fra et vindu til treffenheten. EN må derfor med bruk av fuzzy logic søke etter alle kriterier hver for seg. Sannhetsverdiene må beregnes ved hjelp av trefflistene til enkeltkriteriene.

Assistanse til formulering av kriterier

En rekke former for funksjonalitet kan tenkes for å assistere brukeren i formuleringen av søkekriterier. Ved å supplere brukeren med egnet informasjon kan en bli mer i stand til å finne bildene

MS160 har muligheten til hurtig å telle elementer som tilfredsstiller et sett av søkekriterier. Dette har mange tidligere applikasjoner også benyttet seg av. I programmet MPiRE gis det

øyeblikkelig tilbakemelding om antall treff for hver tast som trykkes ned [MPiRE 94]. For en bruker har antallet treff mye å si for hvor trolig det er at bildene er interessante. Veldig mange treff kan indikere at også elementer med ikke relevant informasjon er med. Mange ord referere til mange ulike objekter og det er vanskelig på forhånd å være oppmerksom på dette. For at en slik funksjonalitet skal være praktisk anvendbar kreves det en meget lav responstid. En bruker vil bare bli forvirret dersom antall treff først kommer etter at flere andre taster er trykket ned. Alternativt kan da funksjonaliteten være at antall treff først blir returnert etter at brukeren selv starter søkingen.

Antall treff kan også være interessant i forbindelse med hvordan de fordeler seg med bruk av forskjellige søkekriterier. Søkeprogrammet returnerer fordelingen på et sett av forhåndsdefinerte kriterier. Slike frekvenstabeller kan gi informasjon om hvordan interessante treff best kan isoleres i et neste søk.

En mulighet er at denne frekvensfordelingen dannes av eksisterende ord i spesielle felter i databasen. For eksempel kan gis en oversikt over antall bilder med ulike former for motiv. Motivene det gis oversikt over er de som finnes i motivfeltet og dermed gir treff for minst et bilde. Dette kan imidlertid føre til en komplisert søkealgoritme med MS160. Ordene må legges i vinduene før søkingen starter. Det bør derfor på forhånd plukkes ut hvilke ord som det skal søkes etter. Det kan dannes en liste over ulike ord i forskjellige felter. En slik liste har mye til felles med en indeks.

Dersom ulike ord i et felt telles på forhånd , vil det kunne genereres en frekvenstabell over hele databasen. Noen spesiell fordel ville en imidlertid ikke få ved bruk av MS160. Det stilles få tidskrav til en søking som bare trenges å utføres ved oppdatering av databasen.

Hurtig søking vil imidlertid være viktig dersom det søkes etter frekvensen av ord i felter sammen med andre søkekriterier. Et problem er det at dette vil kreve svært mange søk i databasen da de fleste felter inneholder mange ulike ord. Eventuelt kan kun en del av ordene telles hvilke må imidlertid bestemmes på forhånd. En liste over ord med flest treff er da ikke mulig da det krever at alle ord telles.

Andre løsninger er mulig for å likevel få frem interessant informasjon. For noen enkeltefelter vil antall mulige ord være begrenset. Eksempel på dette er sted-feltene. Både fylke og eventuelt land vil innholde et begrenset antall ulike kriterier. I enkelte tall-felt vil det dessuten være naturlig å slå sammen flere verdier til et søkekriterium. Det kan søkes etter perioder som tiår. MS160 gir god støtte til slike intervallsøk. Det vil også være mulig å telle ord fra en liste over alle ulike ord dersom selve søkekriteriet begrenser ordet. BIBSYS har realisert en slik funksjonalitet i sitt WWW-grensesnitt. Ved søkeformen nabosøk returneres en oversikt over antall treff for alle ord alfabetisk nær søkekriteriet. En kan tenke seg tilsvarende funksjonalitet i Bauta-basen. Dersom fisk er fylt ut i motivfeltet, kan det returneres frekvensen til alle ord som starter på fisk.

Dersom søket ikke gir for mange treff vil en del av beregningen av frekvensfordelingen kunne gjøres i programvaren. Ved få treff vil det sannsynligvis være vel så effektivt å foreta tellingen direkte av de innleste postene fremfor å foreta et helt nytt søk for hvert ulike kriterium.

I avsnitt 4.1.1 beskrev jeg hvordan ulike oppslagsverk kunne brukes for å lage egnete deskriptorer. Tilsvarende oppslagsverk kan alternativt brukes under formuleringen av søkekriterier. Systemet kan gi brukeren forslag om relaterte ord som alternative kriterier. Eventuelt kan systemet etter spesifikasjon av et kriterium. automatisk søke etter flere synonymer med logisk ELLER i mellom. MS160 vil utføre et slikt logisk søk effektivt. En indeks over felter kan også brukes til direkte å gi forslag til kriterier. Indeksen har oversikt over alle innlagte ord som starter med et søkekriterium. Ved spesifisering av "fisk" kan det direkte gis forslag om at dette kan avgrenses til ord som "fiskere" og "fiskebåt"

5.1.2 Plattform

Avgjørende for hva som er mulig av funksjonalitet for grensesnittet, er hvilken programplattform som benyttes. Valget av dette er avhengig av hvilket operativsystem som benyttes. I og med at MS160 kun er tilgjengelig på PC setter dette noen begrensninger. For min presentasjon vil det også være avgjørende hva som er tilgjengelig av programvare på NTH.

Hovedalternativene står mellom ulike de vindussystemer. De forskjellige operativsystemene har alle sine egne vindussystemer: X Window for unix, Presentation Manager for OS/2 og Microsoft Windows for DOS. Gjennom SCORE toolkitet vil det være mulig å bruke alle maskinplattformer for klient-programmet. Funksjonaliteten til vindus-systemene er ikke ulik. Felles for dem er at det er forholdsvis komplisert å utvikle applikasjoner.

Et alternativ til disse er bruk av en World-Wide Web-tjener. World-Wide Web (WWW) er et globalt hypermedia-system der man kan navigere rundt det verdensomspennende datanettet Internet. WWW er en klient-tjener arkitektur. Tjener-programmer kan tilby tilby tjenester via et nettverk til klient-programmer. Både WWW- tjener og klient-programmer er idag tilgjengelige for de vanligste operativsystemene.

Bruk av et slikt tjenerprogram har den fordelen at den med sin enkle arkitektur forenkler programmeringen. Det muliggjør dessuten netttilkobling. Selve bildene fra en maskin i Mo i Rana kan da også utgjøre en del av presentasjonen. Den enkle programvaren gjør en del funksjonalitet vanskelig å realisere. Overføringstiden legger også begrensninger. En funksjon med øyeblikkelig telling for hver tast som trykkes ned vil ikke være aktuelt. WWW-tjenere er idag tilgjengelige for de vanligste operativsystemene.

5.1.3 Presentasjon av resultat

Gjennom valg av plattform bestemmes også mulighetene for presentasjon av resultatet av søket. De ulike vindussystemene gir mulighete for presentasjon i flere vinduer. Bildene kan integreres som en del av grensesnittet. I min presentasjon vil som sagt bildene ikke kunne utgjøre en del av et slikt grensesnitt.

World-Wide Web-grensesnittet vil kun kunne presentere resultatet av søket i et vindu av gangen. De grafiske mulighetene er dessuten begrenset.

De eksisterende grensesnittene returnerer begge etter søket en liste over de første treffene. Hver element i listen inneholder noe referanseinformasjon og et ikon av bildet. En kan bla seg fremover for å se nye treff. I WWW-grensesnittet kan en dessuten få frem lengre lister uten ikoner av bildet. Mer informasjon om hvert bilde fås frem ved å aktivisere hypertekstlenker i listen.

5.2 Valg av løsning

Jeg velger å bruke WWW som plattform i demonstrasjonen. Funksjonaliteten vil være begrenset, men implementasjonen blir enklere. Den grunnleggende funksjonaliteten i søkingen blir fremdeles mulig å realisere, selv om presentasjonen blir enkel. En fordel er det at bildene kan utgjøre en del av presentasjonen.

Jeg velger å beholde funksjonaliteten med søking i enkeltfelter. Feltvalgene er tidligere valgt ut fra hva som passer brukerne Norsk Folkemuseum og NBR så jeg beholder kategoriene i feltinndelingen.

I søkemenyen vil jeg i tillegg opprette et innlesningsfelt for søking i alle databasefelter på

en gang. All informasjon vil da bli gjennomsøkt. Med mitt valg av søking i enkeltfelter ved hjelp av tagger blir venstretrunkering vanskelig. Søking i alle felter uten hensyn til tagger muliggjør dette.

I tabell 5.1 vises de ulike kategoriene av innlesningsfelter sammen feltene som det søkes i i reservoaret. I kolonnen Kriterium vises spesielle kriterier som må oppfylles for å gi treff. Datafeltnavnene er de samme som brukes i tabell 4.1. Jeg har inkludert tagverdien til de enkelte feltene. Som beskrevet i kapittelet om datalagring er felter som det søkes i samtidig alltid tag-verdier ved siden av hverandre.

Søkekategori	Datafelt	Tagverdi	Kriterium
Alle felter			
Motiv	Motiv	141	
Sted	Adresse	143	
	Områdepresisering	144	
	Kommunekode	145	
	Kommune	146	
	Fylke	147	
	Land	148	
Datering,fra	Datering,til	155	< Datering,til
Datering,til	Datering,fra	154	> Datering,fra
Avbildet Person	Navn	158	Rollekode 71 – 73
Fotograf	Navn	158	Rollekode 11 – 19

Tabell 5.1: Søkekategorier

Jeg velger å implementere bruk av boolske operatorer da MS160 gir god støtte for det. Boolsk søking blir bare mulig innen hver søkekategori. Spesifiseres kriterier i flere søkefelter, brukes det som tidligere logisk OG mellom kriteriene. Innen hvert innlesningsfelt markeres bruk av de logiske operatorene med bruk av spesiellle tegn: * for OG, + for ELLER og - for IKKE. Jeg lar OG ha presedens foran ELLER. I de eksisterende grensesnittene tolkes mellomrom mellom flere ord som logisk OG. Velger å også beholde denne funksjonaliteten ut fra brukerenes ønsker.

For å forenkle implementasjonen av parsingen av feltet åpner jeg ikke for bruk av parenteser. Alle logiske uttrykk kan i prinsippet utrykkes som en sum av mintermer. Uten bruk av parenteser vil enkelte bli mer kompliserte å utrykke.

Jeg lar dessuten logisk søk kun være mulig i sted og motivfeltet. For dateringen vil det ikke være så aktuelt. For feltene avbildet og fotograf vil min form for logisk søk komme i konflikt med representasjonen av personnavn. Det er lagt inn mellomrom mellom fornavn og etternavn. Dette blir vanskelig å kombinere med at mellomrom tolkes som logisk OG.

Jeg lar det ikke være mulig å søke direkte i tidligere søk. Søkingen i hele reservoaret bør være så effektiv at dette ikke er nødvendig. Tidligere søkekriterier kan eventuelt lagres, men den tilstandsløse protokollen til WWW gjør slik bruk av tidligere kriterier vanskelig. Søkekriteriene vil imidlertid ikke bli slettet når en går tilbake til søkemenyen. Dersom nye kriterier fylles ut sammen med de gamle vil dette være identisk med søket rett før.

Bruker høyretrunkering for alle søk. Venstretrunkering skjer kun ved søking i alle felter. Intern trunkering lar jeg pga den kompliserte implementasjonen ikke være mulig. Dette ville ha krevet bruk av flere vinduer med ekstra kontroll for å kontrollere at treffet var riktig. Etter ønske fra oppdragsgiverene har jeg gjort bruk av trunkering implisitt. For å

forenkle implementasjonen velger jeg å ikke gi mulighet til å spesifisere alltegn eventuelt unngå å bruke det.

Gjennom søkingen i tidsrom for bildet utnyttes muligheten for søking i intervaller. Gir ikke brukeren selv muligheten til å spesifisere hvilke felt det skal søkes i bestemte intervaller. Det har ikke vært uttrykket ønske om en slik mulighet.

Jeg velger å la være å implementere noen form for rangering av treff ved en utvidet boolsk logikk. Referanseinformasjonen er for enkel til at en avansert rangering skal være mulig. De enkle forslagene jeg la frem vil det være mulig å realisere, men jeg velger det likevel bort.

Jeg velger å gi brukeren assistanse til formuleringen av kriterier og vil spesielt utnytte muligheten til hurtig telling av treff. Ved at jeg valgte WWW-grensesnitt faller muligheten til øyeblikkelig respons ved hvert tastetrykk bort. Velger isteden å gi en oversikt over frekvensen på fylker og tiår for de innleste kriteriene.

5.3 Søkemeny

På grunnlag av den valgte funksjonaliteten konstruerer jeg en søkemeny. Velger å utforme menyen mest mulig lik det eksisterende WWW-grensesnittet. I figur 5.1 har jeg et eksempel på en utfylt søkemeny. Mellom kriteriene i sted og motivfeltet har jeg brukt logiske operatorer. I eksempelet vil det søkes etter bilder med både mann og gutt i motivfeltet. I stedfeltene kreves det at bildet inneholder enten Asker eller Blommenholm. Askerbilder tas ikke med dersom de inneholder Skaugum i et stedfelt.

Kommenterer funksjonaliteten til de enkelte feltene. (Datafeltnavnene som det henvises til refererer til navnene i tabell 4.1).

- Alle felter: Kan fylles ut et logiske uttrykk av hvilken som helst informasjon om bildet. Det vil søkes i alle felter på en gang. Både høyretrunkering og venstretrunkering brukes automatisk.
- Motiv: Kan fylles ut med et logisk uttrykk som beskriver bildet. Det vil søkes etter uttrykket i Motiv.
- Sted: Fylles ut med et logisk uttrykk som angir stedet der fotografiet er tatt. Det vil søkes etter uttrykket i disse feltene: Adresse, Områdepresisering, Kommunekode, Kommune, Fylke og Land.
- Datering, fra: Fylles ut med et årstall som angir den tidligeste dateringen et bilde skal ha. For bilder som returneres må den innleste verdien må være mindre enn verdien i feltet Datering,til.
- Datering, til: Fylles ut med et årstall som angir den seneste dateringen et bilde skal ha. For bilder som returneres må den innleste verdien må være større enn verdien i feltet Datering, fra.
- Avbildet person: Fylles ut med navnet på en avbildet person. Det vil søkes etter navnet i feltet Navn. Rollekoder i feltet Rollekode som godtas for avbildet person er 71–73.
- Fotograf: Fylles ut med navnet på fotografen. Det vil søkes etter navnet i feltet Navn. Rollekoder i feltet Rollekode som godtas for fotograf er 11–19.

- Start søk: Søker i databasen med de innleste verdiene, og presenterer resultatet avhengig av type søk.
- Nullstill: Sletter alle tekstfelt og setter alle parametrene til standardverdiene.
- Type søk: Bestemmer hva søket returnerer. Feltet har disse alternativene:
  - Treffliste: Returnerer en liste over bildene som blir funnet.
  - Frekvens fylke: Oversikt over fordelingen av treffene på fylker.
  - Frekvens tiår: Oversikt over fordelingen av treffene på tiår.
- Max antall ikoner: Det maksimale antall ikoner (småbilder) som returneres i ett søk. Standardverdi er 10.
- Tolking: Beskriver hva som skjer hvis antall treff i databasen overstiger "Max antall ikoner" (forrige parameter.) Det kan velges mellom disse alternativene (n er "Maks antall ikoner"):
  - Ikoner droppes hvis flere: Hvis søket gir flere enn n treff, returneres bare beskrivelsen av bildene uten ikoner.
  - Maks antall pr. skjermside: Maksimalt n beskrivelser og ikoner hentes over pr. søk. I slutten av trefflista finnes en lenke til de n neste bildene.
- Max antall innslag: Det maksimale antall bilder (beskrivelser) som skal returneres etter et søk i databasen. Standardverdi er 500.
- Tegnsett: Tegnsettet som brukes på maskinen det søkes fra. Dette må settes riktig for at norske tegn skal overføres riktig. Valgene er ISO-8859-1 (vanlig for UNIX-maskiner), CODEPAGE-850 (brukes for PC), og MAC-8BIT (brukes for Macintosh).

5.4 Treffliste

Trefflisten inneholder en liste over bilder som tilfredsstiller søkekriteriene. Den returneres dersom *Type Søk* settes til *Treffliste*. Dersom antall treff overstiger det maksimale antallet som ble spesifisert i søkemenyen vil kun det maksimale antall bilder bli returnert.

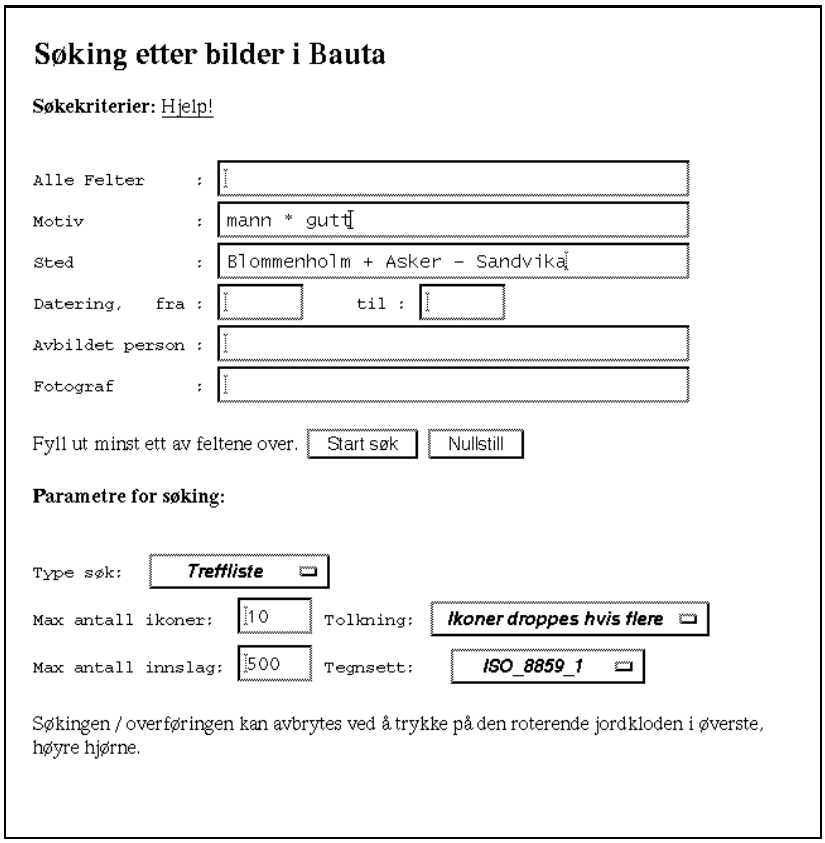
Avhengig av hva som ble spesifisert i søket kan en forminsket utgave av hvert bilde være med for hvert element i listen. Hvis ikonet ikke finnes i databasen, returneres istedet et standardbilde for å indikere at det egentlige bildet ikke finnes.

Hvis ikoner ikke er med i trefflisten, vil det isteden stå teksten "[BILDE]" på dette stedet. Se beskrivelsen av feltene "Max antall ikoner" og "Tolkning" i avsnitt 5.3. Et eksempel på en treffliste der ikoner ikke er med, er vist i figur 5.2.

Annen informasjon i trefflisten er:

- EksemplarID, en unik identifikator til hvert bilde.
- Kommune, Fylke angir i hvilken kommune og fylke fotografiet er tatt.
- Datering fra Datering_fra og Datering_til, angir i hvilken periode fotografiet er tatt.
- Motiv fra Motiv, gir en beskrivelse av fotografiet.

Hvert listeelement inneholder i tillegg to hyperlenker:



Figur 5.1: Hovedmeny for søking i Bauta

Søkeresultat

[BILDE] 1 Mer info. (id:11287) Verdal, Nord−Trøndelag, 1936
Sandvika turisthotell, bygninger, steingjerde, jorde, skog

[BILDE] 2 Mer info. (id:14748) Bærum, Akershus, 1932
Sandvika, kart for plassering av kirken

[BILDE] 3 Mer info. (id:22047) Bærum, Akershus, 1936
Sandvika herredshus, bru, trær

[BILDE] 4 Mer info. (id:22048) Bærum, Akershus, 1936
Sandvika herredshus, bru, trær

[BILDE] 5 Mer info. (id:22049) Bærum, Akershus, 1936
Sandvika herredshus, bru, trær

[BILDE] 6 Mer info. (id:22050) Bærum, Akershus, 1936
Sandvika herredshus, bru, trær

[BILDE] 7 Mer info. (id:22052) Bærum, Akershus, 1936
Sandvika herredshus, bru, trær, bil

[BILDE] 8 Mer info. (id:37815) Bærum, Akershus, 1930
fotografi: Sandvika samvirkelag, utstillingsvindu, trapp, skilt

[BILDE] 9 Mer info. (id:45347) Bærum, Akershus, 1880−1890
Sandvika, vann, robåt, hus

[BILDE] 10 Mer info. (id:45348) Bærum, Akershus, 1880−1890
Sandvika, vann, robåt, hus

[BILDE] 11 Mer info. (id:45350) Bærum, Akershus, 1880−1890
Sandvika, vann, hus

11 bilder ble funnet. Ikoner er ikke overført.

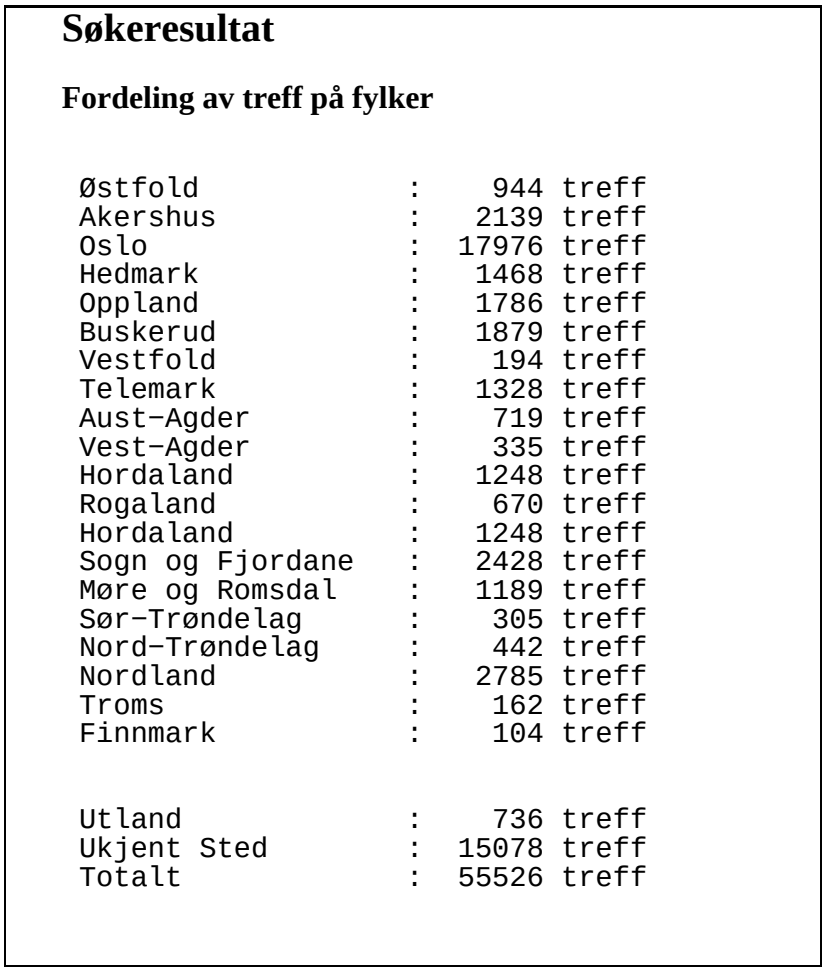
Du har nå kommet til slutten av listen.

Figur 5.2: Treffliste uten ikoner (Sted = Sandvika)

- Bilde: Ved trykking av mustasten på det forminskete bildet (evt. teksten "[BILDE]") vil en få frem selve bildet i full størrelse.[1]
- Mer info: Ved trykking på teksten "Mer info" vil en få frem mer informasjon om bildet.

5.5 Frekvens fylke

Dersom *Type Søk* settes til *Frekvens fylke* returneres en oversikt over fordelingen av treffene på fylkene. I tillegg blir antall treff registrert i utlandet og det totale antall treff vist. Et eksempel på skjermbildet av slik liste er vist i figur 5.3.



Søkeresultat

Fordeling av treff på fylker

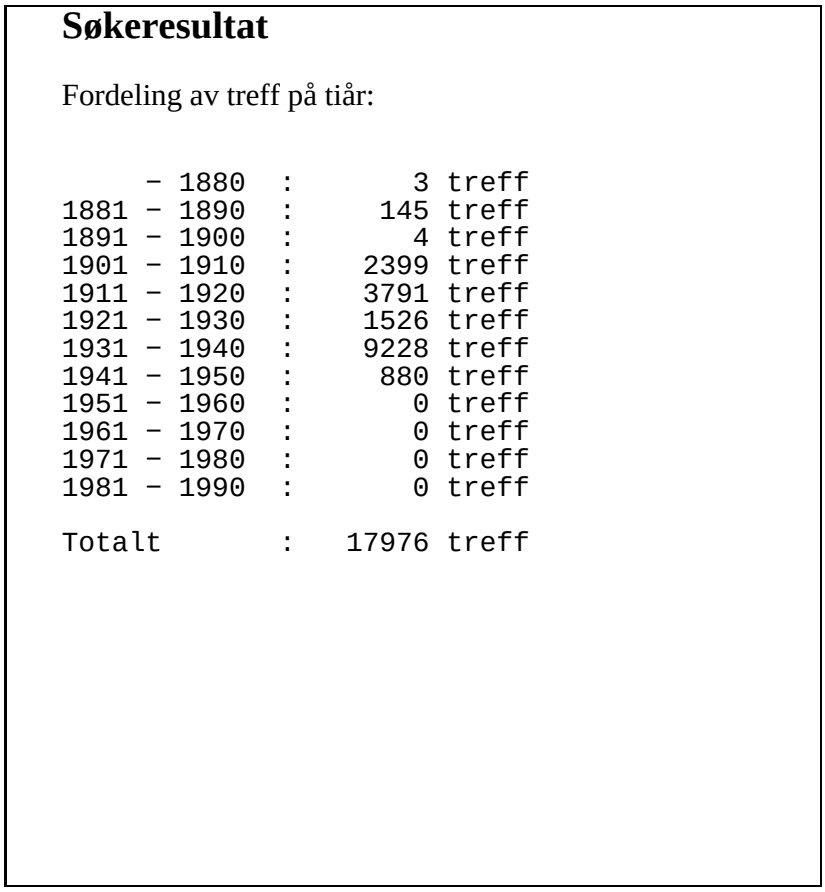
Østfold	:	944 treff
Akershus	:	2139 treff
Oslo	:	17976 treff
Hedmark	:	1468 treff
Oppland	:	1786 treff
Buskerud	:	1879 treff
Vestfold	:	194 treff
Telemark	:	1328 treff
Aust-Agder	:	719 treff
Vest-Agder	:	335 treff
Hordaland	:	1248 treff
Rogaland	:	670 treff
Hordaland	:	1248 treff
Sogn og Fjordane	:	2428 treff
Møre og Romsdal	:	1189 treff
Sør-Trøndelag	:	305 treff
Nord-Trøndelag	:	442 treff
Nordland	:	2785 treff
Troms	:	162 treff
Finnmark	:	104 treff
Utland	:	736 treff
Ukjent Sted	:	15078 treff
Totalt	:	55526 treff

Figur 5.3: Frekvensfordeling for fordeling på fylker, hele databasen

[1] Hvis bildet i full størrelse ikke finnes i databasen, blir det heller ikke laget noen slik lenke.

5.6 Frekvens tiår

Dersom *Type Søk* settes til *Frekvens tiår* returneres en oversikt over fordelingen av treffene på tiår. Periodene er regnet fra 1880 –1990. Et eksempel på skjermbildet av slik liste er vist i figur 5.4.



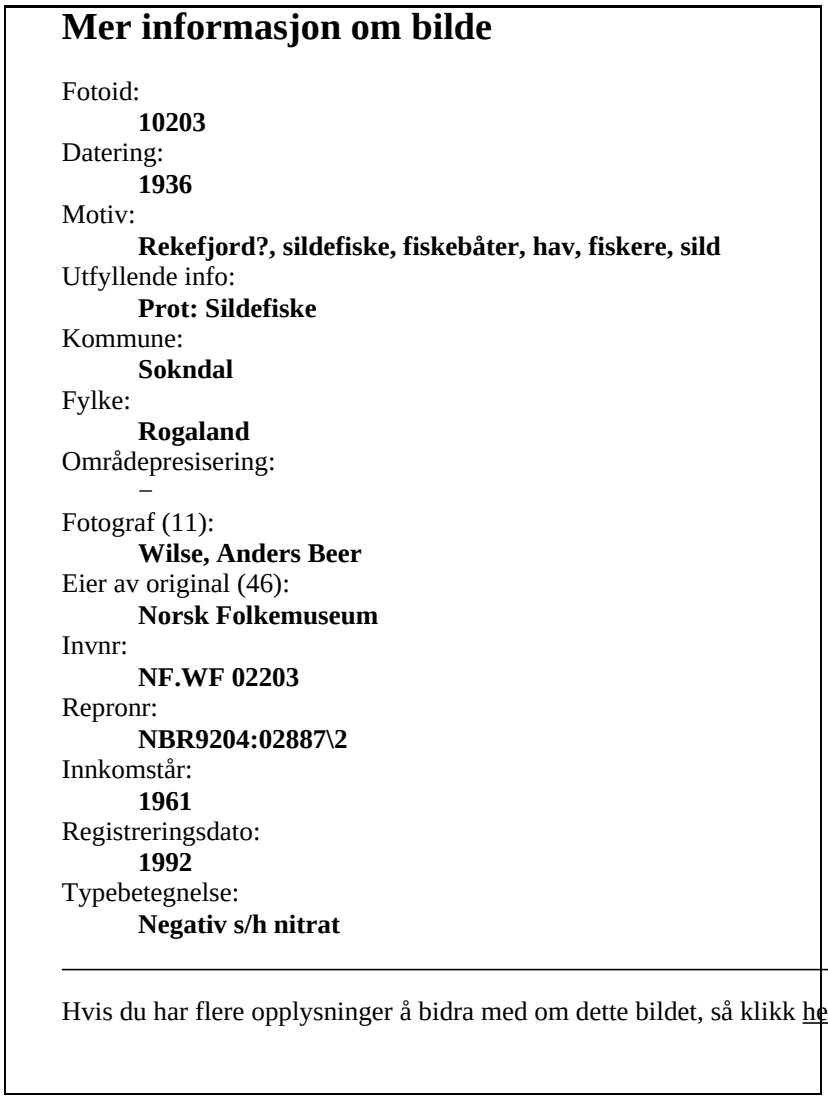
Figur 5.4: Frekvensfordeling for fordeling på tiår, (Sted = Oslo)

5.7 Mer informasjon

Når det klikkes på "Mer info" i trefflisten, vil flere opplysninger om bildet bli hentet. Et eksempel på dette er vist på figur 5.5.

Feltene som hentes, er:

- EksID, en unik identifikator til hvert bilde.
- Datering, angir hvilke periode fotografiet er tatt i. I tillegg skrives teksten fra feltet Datering_kommentar ut i parentes hvis feltet inneholder tekst.
- Motiv, gir en beskrivelse av fotografiet.
- Tittel
- Alternativ tittel



Figur 5.5: Mer informasjon om bilder

- Utfyllende info
- Kommune
- Fylke
- Land
- Områdepresisering
- Adresse
- Fotograf, eier o.l. fra Navn Alle juridiske personer som er registrert i forbindelse med bildet skrives ut sammen med rollen som vedkommende har. Juridiske "personer" refererer av og til også til stedsnavn. Forskjellige rollekoder som er i bruk inkluderer Fotograf (sikker/antatt), Produksjonsstempel, Giver/siste eier, Tidligere eier, Eier av original, Tidligere bruker, Avbildet sted/person (sikker/antatt/feilaktig), Utsikt over og Informant.
- Klassifikasjonskode
- Invariansnummer
- Aksesjonsnummer
- Innkomstår
- Registreringsdato

Kapittel 6

Implementasjonsbeskrivelse

I dette kapittelet beskrives først de løsningsalternativer jeg har stått overfor ved implementasjonen, og deretter beskrives nærmere hvordan implementasjonen av den valgte løsningen er gjort.

Gjennom valget av datalagring og brukergrensesnitt bestemmes det meste av funksjonaliteten. Det er likevel alternativer for hvordan dataene skal legges ut på reservoaret og hvordan selve søkingen skal foretas.

6.1 Alternativer for utlegging på reservoar

Alle data ligger idag i en relasjonsdatabase ved Nasjonalibibliotekavdelingen i Rana . Ved hjelp av ulike programmer skal dataene legges inn på reservoaret på RAMSCAN-kortet. I avsnitt 4.2 beskrives formatet på postene slik de skal ligge i reservoaret. Programmet som leser dataene må derfor også formatere dataene som leses fra databasen slik.

For uthenting av dataene fra en Oracle databasen finnes flere alternativer. Det er flere applikasjoner for Oracle som gir aksess til dataene. Avanserte grafiske grensesnitt vil imidlertid ikke være aktuelt. De krever mye programmeringsarbeid for en funksjonalitet som ikke er nødvendig.

En mye brukt form for grensesnitt er å legge inn SQL-setninger i et annet programmeringsspråk, såkalt Embedded SQL. Et slikt program må først kompileres i en prekompilator før kompilatoren for det aktuelle programmeringsspråket kan brukes. Oracle har utviklet prekompilatorer for de fleste alminnelige programmeringsspråkene. Installert i Mo i Rana er imidlertid bare prekompilatoren for standard C, Pro*C. Ved WWW-grensesnittet mot databasen ble dette grensesnittet benyttet med godt resultat.

I Pro*C vil man programmere SQL-setninger rett inn i vanlige C-program, og får resultatet av søk returnert i variabler deklarert i C-programmet. Man bruker først prekompilatoren som oversetter SQL-setningene til vanlig C-kode, deretter kompileres programmet på vanlig måte. Fordelen med en slik fremgangsmåte er at den gir gode muligheter til å formatere dataene direkte. Ved bruk av ulike C-funksjoner kan dataene legges direkte over på det ønskete lagringsformatet. Resultatet kan skrives til en fil.

Oracle har også utviklet grensesnittet `sqlplus`. Man kan skrive en SQL-setning direkte på en fil og starte programmet `sqlplus` som utfører SQL-setningen. Utskriften fra søket havner på en fil som deretter kan parses for å hente ut den relevante informasjonen. Under en slik parsing med bruk av et tilgjengelig programmeringsspråk av kan dataene for formateres til et ønsket lagringsformat. I forhold til bruk av Pro*C eller et annen form

for Embedded SQL er dette mer komplisert. Dataene kan ikke bli direkte formatert i programmet som leser fra databasen.

For å legge ut dataene på reservoaret må dataene overføres til samme maskin som som MS160. Dette kan skje ved standard filoverføringsprogram som FTP.

Hvordan dataene legges ut på RAMSCAN-kortet vil avhenge av hva slags programmeringssgrensesnitt som benyttes. Enkelte lar reservoaret utgjøre et eget drev i et filsystem. Dataene kan da bare legges ut ved bruk av filsystemets vanlige kopieringskommando. Ulempen med dette er at det er liten kontroll over hvor postene legges. Under søking er det uheldig dersom poster splittes i flere kolonner. Søkekriteriene i søkevinduet vil ikke få treff dersom kriteriet splittes i flere poster.

Andre programmer gir støtte til å skrive data direkte til blokker i reservoaret. Et eget program må da lages som leser data fra en fil og skriver til kortet. En fordel med et slikt grensenitt er at en kan fordele dataene jevnt på alle kolonnene. Dataene vil ofte kun utgjøre en del av det tilgjengelige lageret. Ved å fordele dataene jevnt vil søkingen kunne begrenses til kun de radene i hver kolonne som innholder relevant informasjon. Samtidig vil søkingen kunne skje parallelt i alle kolonnene.

Problemet med deling av poster kan løses ved direkte aksess til reservoaret. Når data skrives til en ny kolonne kan det sørges for at dette ikke skjer midt i en post. Fordi de tilgjengelige programmene kun gir støtte til skriving av hele blokker fra filer kan imidlertid dette bli komplisert. En måte å forenkle hele utleggingen på er å dele datamengden opp i 8 like store filer. Hver fil kan da skrives til en kolonne. I oppdelingen er det sørget for at filen aldri ender midt i en post.

6.2 Alternativer for Søking

Gjennom valget av WWW-grensesnitt bestemmes mye av implementasjonen. En WWW-tjener sørger for å returnere returnere menyer og søkeresultater til en WWW-klient. WWW-tjenere er allment tilgjengelige for de fleste operativsystemer.

Selve søkemenyen kan implementeres ved å skrive et dokument i hypertekstspråket HTML. Innlesningsfeltene skrives ved hjelp av HTML-taggen FORM. Søkingen må bli utført eksekverbare enkeltprogrammer. Disse kan returnere søkeresultatene som HTML-kode på bakgrunn av de innleste parametrene.

Søkingen vil enklest bli utført av et av grensesnittprogrammene til kortet. Det vil være mulig å implementere en egen applikasjon med lavnivå aksess til kortet, men dette vil kreve mer programmering. I avsnitt 2.4 har jeg beskrevet de tre tilgjengelige toolkitene: RAMSCAN Toolkit 1.0, MS160api Beta 2.1 og SCORE. De inneholder alle omtrent den samme funksjonaliteten. En vesensforskjell er at de to første kjører under en DOS/Windows-plattform, mens SCORE kjører under OS/2. En annen ulikhet er at RAMSCAN Toolkit 1.0 og MS160api Beta 2.1 begge gir støtte til bruk av reservoaret ved hjelp av filsystemet. MS160api Beta 2.1 og SCORE gir mulighet til å skrive direkte til blokker i reservoaret.

6.3 Valg av løsninger

Velger å bruke et program skrevet i Pro*C til uthenting av data fra databasen. Et slikt program vil være fleksibelt til direkte formatering av dataene. Det skriver alle data til en fil som kan over føres til MS160 maskinen.

Velger å skrive data direkte til blokker på reservoaret. Dette reduserer datamengden som må gjennomsøkes og sikrer mot at poster deles. For å forenkle utleggingen lages et eget program som oppdeler den totale datamengden i 8 jevnstore filer.

De kraftigste tilgjengelige PCene på NTH med MS160 bruker alle OS/2 operativsystem. Teoretisk vil disse kunne kjøre både DOS Windows og OS/2 programmer. Jeg har imidlertid ikke greid å få WWW-tjeneren for Windows til å fungere tilfredsstillende under OS/2. Jeg har også forsøkt å benytte WWW-tjeneren på maskiner under Windows. Den tilgjengelige versjonen av nettverksprogrammet PC-NFS 5.0 gir dessverre ikke skikkelig støtte til å kjøre tjeneren. Jeg vil defor måtte velge WWW-serveren for OS/2.

Jeg velger å bruke MS160api Beta 2.1 som programvare mot brikken. SCORE gir også anledning til å skrive direkte på reservoaret. Den eneste skikkelig fungerende kompilatoren som er tilgjengelig er Borland C++ 4.0 for Windows. Valg av MS160api vil gjøre det mulig å lage søkeprogrammer under DOS/Windows.

En ulempe med mine valg er at WWW-tjeneren for OS/2 vil måtte starte et DOS/Windows-program. For å muliggjøre dette lot jeg tjeneren først kalle et program skrevet i OS/2. Dette kaller igjen et DOS-program. Kommunikasjonen mellom disse foregår ved at parametre og resultater blir skrevet til fil. Dette er ingen ideell løsning. En konsekvens er at all søking tar ekstra tid da det er tidkrevende å starte et DOS-program fra et OS/program. Noe egentlig alternativ greide jeg imidlertid ikke å finne. Med den tilgjengelige programvaren ble forlenget søketid en konsekvens av WWW-grensesnittet.

6.4 Filer og Moduler

Min implementasjon består av følgende filer:

6.4.1 Utlegging på Reservoar

- `hentbase.pc` - Program som henter ut data fra relasjonsdatabasen og legger dem i hierarkiske poster i filen base.
- `parse.c` - Rutiner som parser input og returnerer parameterverdier. Skrevet av Roar Foshaug. Brukes i programmet hentbase.
- `charsets.c` - Rutiner som oversetter mellom ulike tegnsett. Skrevet av Roar Foshaug. Brukes i programmet hentbase.
- `parse.h` - Headerfil for `parse.c`.
- `charsets.h` - Headerfil for `charsets.c`.
- `common.h` - Headerfil som inkluderes fra `parse.c` og `charsets.c`.
- `delbase.c` - Program som deler filen base i 8 jevnstore filer base0 – base7. Sørger for at det ikke deles midt i poster.
- `skrivbas.c` Program som legger filene base0 – base7 ut på reservoaret til MS160 i kolonne 0–7. Legger filene fra starten i hver kolonne returnerer den største raden som den skrives til.

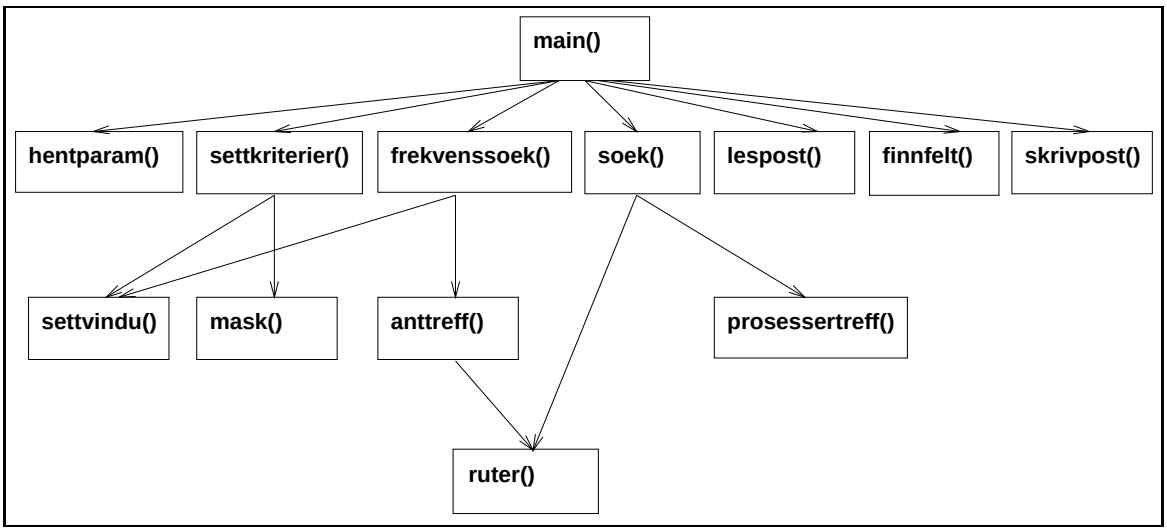
6.4.2 Søking

- `hovedmeny.html` - hovedmeny for søking i databasen.
- `hjelp.html` - hjelp til søking i databasen. Lenke til dette dokumentet fra `hovedmeny.html`.
- `soek2.c` - OS/2-program som kaller det egentlige søkeprogrammet soek.cpp.
- `merinfo2.c` - OS/2-program som kaller det egentlige programmet for mer informasjon om hvert bilde `merinfo.cpp`.
- `soek.cpp` - Hovedprogrammet som utfører søk i databasen. Kalles når brukere trykker "Start søk" i hovedmenyen. Avhengig av type søk som er spesifisert returneres en treffliste en frekvens over fylker eller frekvens over tiår.
- `merinfo.cpp` - Programmet som genererer "Mer informasjon" om bilder, se skjermbilde på figur 5.5.
- `kall.cpp` - Rutiner for lesing fra reservoaret som både brukes av soek.cpp og merinfo.cpp.
- `kall.h` - Headerfil for `kall.cpp`.
- `fylker.txt` Innholder alle fylker som telles ved frekvenssøk. Lese av soek.cpp

6.4.3 Biblioteksfiler for MS160api

- `tdosbird.lib` Biblioteksfil for MS160api. Kompileres sammen med skrivbase, soek og merinfo.
- `ms160api.h` Headerfil for `tdosbird.lib`.
- `ms160def.h` Headerfil med definisjoner for `tdosbird.lib`. Inkluderes av `ms160api.h`.

Filene med endelse `.pc` er skrevet i Pro*C, og må først kompileres med en Pro*C prekompilator som oversetter alle databaseforespørsler i programmet til standard C. All kildekode er i vedlegget.



Figur 6.1: Kallgraf for soek.cpp

6.5 Oppbygging av soek.cpp

Beskriver oppbyggingen av hovedprogrammet for søking `soek.cpp` mer i detalj. Programmet innholder en del hovedfunksjoner som tar seg av søkingen. Av figur 6.1 sees en kallgraf for programmet. Virkemåten til de enkelte funksjonene:

- `hentparam()` Henter parameter og dekoder til format som kan brukes i en URL. Brukes for å lage lenker i trefflisten.
- `settkriterier()` Parser kriteriene fra de innleste feltene og legger dem i vinduene. Setter treffmasken.
- `frekvenssoek()` Kalles ved type søk = frekvens fylke eller tiaar. Skriver ut frekvensfordelingen.
- `soek()` Søker i reservoaret ved type søk = treffliste.
- `lespost()` Leser en post fra reservoaret.
- `finnfelt()` Leser et felt fra en post som starter med en bestemt tag.
- `skrivpost()` Skriver et element i trefflisten til fil.
- `settvindu()` Konfigurerer et vindu ut fra parametre.
- `mask()` Beregner treffmasken ut fra antall kriterier og hvilke logiske operatorer som er brukt.
- `anttreff()` Søker i reservoaret i count-modus og returnerer antall treff.
- `ruter()` Konfigurerer ruteren.
- `prosessertreff()` Beregner treffposisjoner utfra offset i kolonne. og legger resultatet i en tabell.

6.6 Valg av huskelengde

For å muliggjøre parallell søking etter flere kriterier er det nødvendig å sette en passende huskelengde som gjør at nærliggende kriterier genererer treff samtidig. For å unngå at kriteriene genererer flere etterpåfølgende treff har jeg valgt alltid å legge slutten av posten som et eget kriterium i et eget vindu. Jeg setter huskelengden for denne til 0. Jeg unngår dermed at det genereres flere treff for hver post.

For at det skal genereres treff for et søk må huskelengden for et vindu settes så stort at det fremdeles genereres treff for vinduet når slutten av posten nås. Samtidig bør det unngås at den settes så stor at den fremdeles genererer treff når slutten av posten etter nås. Ved at det søkes i enkeltfelt eller felter ved siden av hverandre vil det være varierende avstand til slutten av posten. Felter mot slutten av posten trenger en liten huskelengde, mens feltene i starten trenger en stor.

Det er imidlertid et problem at enkelte felter har svært variabel lengde. For postene med de lengste feltene kan det være nødvendig med en stor huskelengde for fremdeles å generere treff når slutten nås. For søking i det samme feltet med kortere post kan det bli generert treff for også posten etter.

Problemet er delvis løst ved at de korteste feltene er lagt mot slutten av posten. Med søking i de korte postene blir sannsynligheten svært liten for ar det på en liten huskelengde skal nås frem til slutten over en post etter. Flere av dem har dessuten konstant lengde slik at det uansett ikke blir noe problem. For de lange i begynnelsen er det imidlertid et problem. Spesielt er problemet stort for feltet som søker i alle felter. For å sikre at alle treff blir registrert må huskelengden settes til postens maksimallengde.

Tidligere forsøk på å løse dette problemet har vært gjort ved å gå igjennom alle postene som gir treff og kontrollere at den ikke bare er registrert ved at huskelengden var for lang. Ved muligheten til logisk søk med operatoren IKKE vil en slik metode ikke være helt sikker. En kan da også risikere at felter som skulle gi treff ikke kommer. Treff for kriterier i posten foran kan resultere i at et kriterium med IKKE foran ikke oppfylles.

Ideelt sett burde huskelengden settes uavhengig av de enkelte dataene. Den bør settes på grunnlag av plassen som er satt av i databasen. Av tabell 4.1 sees imidlertid at den avsatte plassen er langt større enn det den nåværende Bautabasen innholder. Ved for store huskelengder øker sannsynligheten for feil. Velger derfor å minimalisere huskelengden etter de innlagte dataene.

Ved av et hjelpeprogram har jeg målt avstanden fra starten av felter til slutten av posten for alle postene. I tabell 6.1 har jeg lagt inn maksimal og minimalverdier for de enkelte søkekategorier for avstand til slutten av posten og slutten av neste post. Ved å sammenligne maksimalverdien til slutten av posten og minimalverdien til slutten av neste post ser jeg til hvilke verdier huskelengden trygt kan settes. Dersom maksimal verdien er større enn minimalverdien vil det kunne oppstå gale treff. I den siste kolonnen er det markert andelen av postene som har slutten av neste post nærmere enn maksimalverden avstand til slutten av posten.

Ut fra tabellen velger jeg følgende huskelengder for kriteriene.

- Alle felter: Svært vanskelig å gi en god verdi. For å sikre at det søkes i alle felter settes den til 566 Dette vil imidlertid gi ekstra treff for 84 % av postene.
- Motiv: Med verdi 225 er det ingen ekstra treff.
- Sted: Setter til 168. Et minimalt antall ekstra treff.

Søkekategori	Avstand slutt post		Avstand slutt neste post		Andel feil
	Maks	Min.	Maks.	Min.	
Alle felter	566	103	1094	206	0.840
Motiv	225	79	815	274	-
Sted	168	59	699	162	0.000
Person / Fotograf	296	115	815	274	0.000
Datering	25	25	591	128	-

Tabell 6.1: Avstand til slutten av posten

- Person/Fotograf: Setter til 296. Et minimalt antall ekstra treff.
- Datering: Med verdi 25 er det ingen ekstra treff.

Med mine valg av huskelengden vil feilmarginen bli minimal bortsett fra ved søking i alle felter. Årsaken til dette er at det inkluderer søking i feltet Utfyllende Info. Dette feltet har svært variabel lengde. Ser av tabell 4.1 at den maksimale lengden er 312 tegn. Dette er lengre enn de korteste postene. Søking etter personer i feltet rett etter skjer imidlertid uten problemer. Kun en mikroskopisk del av søkene vil gi feil.

Søking i alle felter vil gi ekstra treff for hele 84 % av postene. For å muliggjøre søking i all informasjon inkludert utfyllende info bør derfor treffene etterbehandles for å fjerne ekstra treff. Dette blir imidlertid problematisk ved svært mange treff. Etterbehandlingen kan bli tidkrevende. For å gi et korrekt antall treff må alle postene leses og kontrolleres ikke bare de første som vises. Det blir på samme måte vanskelig å få en korrekt frekvensfordeling. Her kan normalt COUNT-søk benyttes slik at posisjonen til posten ikke returneres. Det vil da ikke være mulig å kontrollere treffene. Endres søket til REPORT-søk blir dette med tidkrevende. Etterbehandlingen vil dessuten ikke kunne gjøre noe med at enkelte treff ikke blir registrert ved bruk av IKKE-operatoren.

Velger av tidshensyn ikke å implementere en kontroll for ekstra treff ved søk i alle felter. Funksjonaliteten ville under alle omstendigheter ikke blitt helt tilfredsstillende. Responstiden kunne blitt langt høyere. Sannsynligvis vil det ikke gi så mange som 84% ekstra treff. Sorteringen på negativnummer gjør at relaterte bilder er ofte etter hverandre og gir treff allikevel.

6.7 Parallellisering

Jeg har valgt å bruke et ekstra vindu for søking etter slutten av posten. Med bruk av færre enn 4 søkekriterier vil det likevel være mulig å parallellisere søkingen. Med et søkekriterium vil kriteriet sammen med slutt-tagen kunne dupliseres 4 ganger i de 8 vinduene. To eller tre kriterier kan dupliseres 2 ganger. 4 til 7 kriterier kan ikke dupliseres. Flere enn 7 ulike kriterier vil det ikke være mulig å søke etter på en gang.

Ulempen med en slik parallellisering er at det vil være en skygge etter hvert treff på 12 byte dersom posisjonene til treff returneres (Report modus). Dersom det ikke parallelliseres vil dette ikke være noe problem. Ingen poster er så korte. Ved parallellisering er imidlertid dette uheldig. Slutten av en post kan befinne seg mindre enn 12 byte etter slutten av en annen regnet fra kolonnestarten. Dersom det søkes etter disse parallelt vil den siste posten aldri bli funnet sammen med den første. Dette er uheldig i en statisk datamengde. Det

vil være mulig å sikre seg mot dette ved å kontrollere alle mulige skyggeposisjoner etter søket. Dette kan imidlertid bli tidkrevende ved mange treff. Uheldig er det også at treffene ikke vil bli rapportert i rekkefølge, men etter avstand fra starten av kolonnen. Treffene bør presenteres etter negativnummer. Ved få treff vil en sortering være enkel. Ved mange treff kan det imidlertid bli nødvendig å sortere mange treffposisjoner blant dem også de som ikke skal hentes.

Den teoretisk minimale søketiden i den eksisterende datamengden uten parallellisering er 0.7 sekunder (se avsnitt 7.4.1) Ved parallellisering vil dette kunne reduseres til 1/2 eller 1/4. Ved mange treff vil sannsynligvis den totale tiden for begge bli større fordi tid vil gå med til å stoppe søket ved lagring av treffposisjoner. I tillegg kommer den ekstra tiden til kontroll av skyggeposisjoner og til sortering treffene.

Jeg velger bort å parallellisere søkingen ved Report-søk etter trefflisten. Tidsbesparelsen blir sannsynligvis ikke stor og implementasjonen blir enklere. Ved Frekvenssøkingen brukes Count-modus ved at treffene bare telles. Her er det ingen skygge slik at jeg ved disse søkene vil duplisere søkekriteriene i flere vinduer.

6.8 Overføring av bilder

Et av formålene med implementasjonen var å gjøre det mulig å hente bilder direkte fra databasen i Mo i Rana. Jeg planla å bruke den eksisterende WWW-tjeneren til å overføre bildene. Dette var noe av bakgrunnen for at jeg valgte et WWW-grensesnitt. I løpet av oppgaven har imidlertid hele databasen blitt sperret for offentlig innsyn. Materialet vil bli gjennomgått og kontrollert for eventuelle sensitive opplysninger. Jeg har fått tillatelse til å hente selve referanseopplysningene. Jeg vil imidlertid ikke kunne hente bildene fra en allment tilgjengelig WWW-tjener.

Jeg implementerte likevel funksjonaliteten som gjør det mulig å hente bildene. Mye av arbeidet var basert på WWW-prosjektet slik at det ikke ble mye ekstraarbeid. Bildene vil imidlertid ikke bli overført. Fotografier kommer ikke kommer frem fordi tjeneren ved NBR ikke vil sende dem.

Kapittel 7

Erfaringer

Jeg vil dette kapittelet diskutere hvilke erfaringer jeg har gjort med bruken av den realiserte løsningen. Først vil jeg se på erfaringer med utleggingen på av Wilsesamlingen på reservoaret. Deretter vil det bli sett på brukervennligheten og responstiden. Jeg vil spesielt studere hvordan denne er sammenlignet med de allerede eksisterende grensesnittene. Med bakgrunn i disse erfaringene vil jeg se på hvilke fremtidige endringer og utvidelser som kan gjøres for å bedre virkemåten til MS160-grensesnittet. Jeg vil også se på hvilke endringer som kunne gjøres med selve brikkens funksjonalitet for å gjøre den mer i stand til å søke i en datamengde som Bauta-basen.

7.1 Utlegging på reservoaret

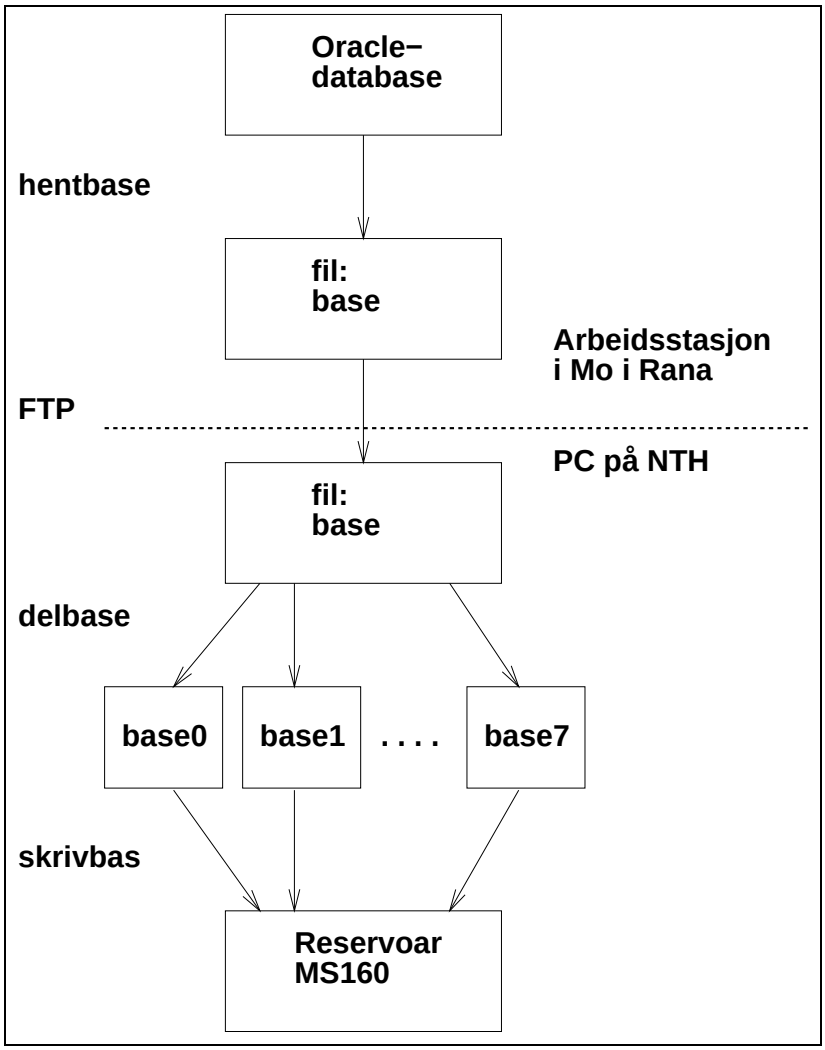
Jeg har kontrollert tidsforbruket for innlegging av dataene på reservoaret. Prosessen som skal til kommer frem av figur 7.1. Ved kjøring av programmet `hentbase` på maskinen med Wilse-samlingen ble det generert en fil `base` på ca 14 MB. Uthentingen tok ca 45 minutter. Filen ble overført via FTP til maskinen med MS160. Med en effektiv overføringshastighet på 10 kB/s tok overføringen 23 minutter. Programmet `delbase` delte filen `base` i 8 jevnstore filer `base0` - `base7` på ca 2 minutter. Programmet `skrivbas` la disse filene ut på reservoaret på ca 30 sekunder.

Et eksempel på formatet på filene som legges ut på reservoaret er vist i figur 7.2. Det er tatt et utsnitt fra starten av `base0`. Det henvises til tabell 4.1 for rekkefølgen av feltene. Merk at taggene mangler da disse er kontrolltegn som ikke kan skrives ut.

Med de gitte rammevilkårene blir oppdateringen av databasen forholdsvis tidkrevende. Denne prosessen vil imidlertid ikke foretas så ofte. Tiden det tar å lese de ferdig-genererte filene inn i reservoaret er dessuten lav. Hver gang maskinen med MS160 startes på nytt må dette programmet kjøres. Ved innlegging av nye data vil dette være tiden det ikke vil være mulig å søke på grunn av oppdatering.

7.2 Brukergrensesnitt

Ved at World-Wide Web ble valgt som plattform ble det satt en del begrensninger for utformingen av brukergrensesnittet og funksjonaliteten. Brukergrensesnittet er likevel utformet slik at en ikke datakyndig person kan ta det i bruk ved hjelp av enkel veiledning. Grunnleggende søking og henting av informasjon og bilder bør være lett og intuitivt å forstå.



Figur 7.1: Prosess for oppdatering av reservoar

```
1NBR9001:0000311Raabe, Einar von Hannofiskere,garnbøting.7218
65VåganNordlandNorgeNegativ s/h nitratOU284.0000LS 1985/19:00
03194219501990 1985
2NBR9001:000020Raabe, Einar von HannoRobåt,fiskebåteribakgr72
Vestfjorden1865VåganNordlandNorgeNegativ s/h nitratOU501.0000
LS 1985/19:00021942195019901985
3NBR9001:0000111Raabe, Einar von Hannofisktrekkesoppfrahavet.
72Vestfjorden1865VåganNordlandNorgeNegativ s/h nitratOU136.00
00OU226.0000LS 1985/19:00011942195019901985
4NBR9001:0000411Raabe, Einar von Hannofiskereombordifiskebåt,
721865VåganNordlandNorgeNegativ s/h nitratOU501.0000LS 1985/1
9:00041942195019901985
5NBR9001:0000511Raabe, Einar von Hannofiskebåterpåveiutfra Svo
lvær71Svolvær havn1865VåganNordlandNorgeNegativ s/h nitratOU5
01.0000LS 1985/19:00051942195019901985
6NBR9001:0000611Raabe, Einar von Hannofiskebåter,fiskeflåtenl
iggeråventer.72Vestfjorden1865VåganNordlandNorgeNegativ s/h n
itratOU501.0000LS 1985/19:00061942195019901985
7NBR9001:0000711Raabe, Einar von Hannofiskebåter,fiskeflåtenp
åveitilfiskefeltet71Vestfjorden1865VåganNordlandNorgeNegativ
s/h nitratOU501.0000LS 1985/19:00071942195019901985
8NBR9001:0000811Raabe, Einar von Hannofiskebåter,fiskeflåtenp
åveitilfiskefeltet71Vestfjorden1865VåganNordlandNorgeNegativ
s/h nitratOU501.0000LS 1985/19:00081942195019901985
9NBR9001:0000911Raabe, Einar von Hannofiskebåter,fiskeflåtenp
åveitilfiskefeltet71Vestfjorden1865VåganNordlandNorgeNegativ
s/h nitratOU501.0000LS 1985/19:00091942195019901985
10NBR9001:0001011Raabe, Einar von Hannotomennistyrhuset721865
VåganNordlandNorgeNegativ s/h nitratOU501.0000LS 1985/19:0010
1942195019901985
```

Figur 7.2: Utsnitt av filen som legges i reservoaret

Den utvidete funksjonaliteten vil muligens ikke være like enkel å ta i bruk for en uerfaren bruker. Bruk av logiske operatorer er ikke selvinnlysende. For en mer erfaren bruker vil imidlertid de utvidelsene som er gjort neppe by på problemer. Det er også denne gruppen som vil ha mest å hente ved økte muligheter og støtte til å formulere dekkende søkekriterier.

Gjennom min egen bruk av databasen har jeg funnet utvidelsene nyttige. Datamengden er stor og uoversiktlig slik at det kan bli vanskelig å finne frem til ønsket informasjon. Spesielt anvendbar har jeg derfor funnet mulighetene til å avgrense bilder i en stor hentet datamengde. Frekvenslistene for tiår og fylker gir opplysninger om hvilke steder og tidsrom som det kan søkes i for å gi en passende lang treffliste. Bruk av IKKE-operatoren fjerner de uinteressante bildene.

Modifiseringen av sted-representasjonen vil være nyttig for de uten spesifikke krav til sted. Ved å åpne for søking i fylker vil det bli lettere å spesifisere sted for de med mer uklare krav til plassering.

Som tidligere beskrevet har jeg ikke hatt anledning til å hente selve bildene i databasen. All testing har derfor foregått uten bruk av dem. Den innlagte funksjonaliteten er imidlertid identisk med det tidligere WWW-grensesnittet [Bauta 94]. De samme erfaringene som da ble gjort ville derfor kunne gjøres med visning av bilder i MS160-grensesnittetet.

Under valget av huskelengden for kriteriene ble det beskrevet hvordan dette kunne føre til at gale bilder ble returnert (se avsnitt 6.6). For de fleste kriterier var imidlertid ikke dette noe problem. Bare for søking i alle felter kunne feilene bli mange. Her kunne til gjengjeld 84% av bildene resultere i ekstra treff. Disse antagelsene ble bekreftet under uttestingen. Resultatet av søkingen i alle felter var ikke tilfredsstillende. I figur 7.3 vises trefflisten ved søking etter `Blommenholm` i `Bærum` i alle felter. I reservoaret er bildene sortert etter `id` og disse ligger stort sett fortløpende. Figuren viser at treffene ligger i grupper etter hverandre slik at en del av de ekstra treffene gir treff allikevel. Etter hver gruppe er det imidlertid ekstra bilder.

En svakhet ved grensesnittet er at det ikke gir støtte til venstretrunkering bortsett fra for

Søkeresultat

[BILDE] 1 Mer info. (id:23097) Bærum, Akershus, 1919
gruppe, kvinner, brodering, husvegg, vindu

[BILDE] 2 Mer info. (id:23098) Bærum, Akershus, 1919
gruppe, kvinner, mann, trær

[BILDE] 3 Mer info. (id:23099) Nesodden, Akershus, 1919
Nesodden feriekoloni, gruppe, jenter, kvinner, menn, hus, skog

[BILDE] 4 Mer info. (id:34839) Bærum, Akershus, 1919
gruppe, kvinner, brodering, husvegg, vindu

[BILDE] 5 Mer info. (id:34840) Bærum, Akershus, 1919
gruppe, kvinner, mann, trær

[BILDE] 6 Mer info. (id:34841) Nesodden, Akershus, 1919
Nesodden feriekoloni, gruppe, jenter, kvinner, menn, hus, skog

[BILDE] 7 Mer info. (id:43175) Bærum, Akershus, 1919
gruppe, kvinner, mann, trær

[BILDE] 8 Mer info. (id:43176) Bærum, Akershus, 1919
gruppe, kvinner, mann, trær

[BILDE] 9 Mer info. (id:43177) Bærum, Akershus, 1919
mann, stol, stokk, busker

[BILDE] 10 Mer info. (id:43178) Nesodden, Akershus, 1919
Nesodden feriekoloni, gruppe, jenter, kvinner, flagg, greiner

10 bilder ble funnet. Ikoner er ikke overført.

Du har nå kommet til slutten av listen.

Figur 7.3: Resultat ved søking etter Blommenholm og Bærum i alle felter

søk i alle felter. Oppdragsgiverne har gitt uttrykk for at de ønsker seg en slik funksjonalitet. Sammen med problemene jeg har hatt med søkingen i alle felter setter dette et spørsmål med bruk av MS160. Venstretrunkering lar seg ikke realisere på en enkel måte.

7.3 Programvare

Jeg tok i bruk hjelpeprogrammer for å lette realiseringen av min løsning. Mine erfaringer med bruken av dem har vært noe blandet. Jeg valgte bruke en WWW-tjener for å formidle all kontakt med dataene. Som nevnt i avsnitt 6.3 valgte jeg bort å bruke WWW-tjeneren for Windows på grunn av problemer med samspillet mellom denne og tilgjengelige nettverksprogrammer. Bruken av en OS/2 tjener for å kjøre et DOS-program var imidlertid ikke ideelt. Det gikk mye ekstra tid med til oppstart av søkeprogrammene.

MS160api Beta 2.1 bærer også preg av å ikke være noen offisiell versjon. Gjennom bruken av den oppdaget jeg flere svakheter. Funksjonen `setGetResults()` som returnerer adressen i reservoaret til treffene innholder feil. Byteverdiene i adressen er byttet om. Mer alvorlig er det at funksjonen er så ustabil at den ofte returnerer helt gale verdier. I den første versjonen jeg brukte, MS160api Beta 2.0, virket ikke denne funksjonen i det hele tatt. Fremdeles er det et stykke igjen til programvaren er stabil.

Mitt inntrykk av alle toolkitene jeg har studert er at de krever programmering på et unødvendig lavt nivå. Det er tungvint at søkeprogrammer må innholde mange komplisert metoder når disse stort sett er felles for alle. Grensesnittapplikasjonene bør videreutvikles slik at programmering mot brikken i fremtiden kan gjøres enklere.

7.4 Tidsforbruk

Kort responstid er avgjørende for at grensesnittet skal være brukervennlig. Noe av bakgrunnen for forsøk med bruk av MS160 er nettopp muligheten for hurtig søking. Jeg har regnet på hvilke responstid en teoretisk kan forvente ut fra brikkens spesifikasjoner. Etterpå har jeg holdt disse tallene opp mot testresultatene. Jeg har også sammenlignet responstiden for ulike typer søk i Oracle-Databasen og i reservoaret til MS160.

Under testingen har jeg valgt å helt se bort fra overføringstid ved søking via datanett. Den vil uansett være uavhengig av søkesystemet. Det gis til slutt en vurdering av hva overføringstiden kan ha å si for vurderingen av testresultatet.

7.4.1 Treffliste

Avgjørende for responstiden er søkehastigheten og datamengden som må gjennomsøkes. I søkingen etter trefflisten valgte jeg å ikke søke parallelt (se avsnitt 6.7. Dette gjør at alle de 8 kolonnene i reservoaret må søkes etter tur. Søkehastigheten blir da 160 MB/s / 8 = 20 MB/s. Den totale datamengden er ca 14 MB. Tiden for et søk blir da: 14 MB / 20 MB/s = 0.7s. Fordi det uansett ikke søkes parallelt i flere kolonner vil antall kriterier ikke ha noen innvirkning på søketiden. Dette forutsetter at antall kriterier ikke overstiger det maksimale antallet på 7. I søkingen etter trefflisten er brikken i Report-modus. Dette betyr at søkingen avbrytes for hvert treff for at adressen skal legges i et buffer. Ved mange treff vil dette sannsynligvis innvirke på responstiden. Under all testingen er det de 500 første treffene som leses fra reservoaret. Prosesseringen av opp til 500 poster vil derfor også ta tid.

Kriterier	Antall treff	Responstid (s)
Motiv = Munkholmen	9	0.9
Motiv = sildoljefabrikk	17	1.0
Motiv = sildefiske	396	2.7
Motiv = mann	9847	4.1
Sted = Blommenholm	7	0.9
Sted = Trondheim	131	1.5
Sted = Blommenholm + Asker	930	3.2
Sted = Blommenholm + Asker - Skaugum	163	1.6
Sted = Oslo	17976	5.4
Datering = 1879–1879	2	0.9
Datering = 1939–1940	10291	4.4
Avbildet = Nansen	18	1.0
Avbildet = Hamsun	108	1.3
Fotograf = Skjevlo	1	0.9
Fotograf = Lumholtz	731	3.0
Fotograf = Wilse	48375	9.6

Tabell 7.1: Responstid for ulike søk etter treffliste

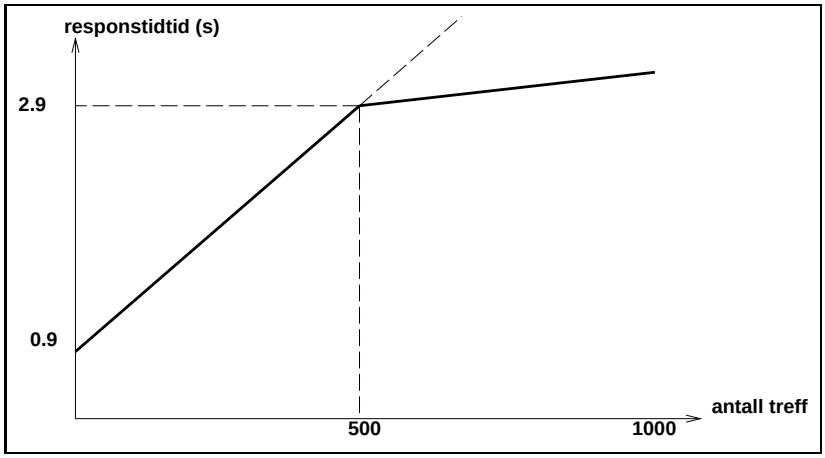
Av tabell 7.1 fremgår testresultatene for søking etter treffliste med ulike søkekriterier. Resultatene stemmer med antagelsene. Søketiden virker uavhengig av antall søkekriterier og søkekategori. Antall treff har imidlertid stor innvirkning på resultatet. Ved svært mange treff som ved søking etter fotograf Wilse blir responstiden svært mye større enn det teoretiske minimum på 0.7 s.

Ved å studere testresultatene kan det lages et uttrykk for søketiden som en funksjon av datamengden og antall treff. Ved få treff er responstiden 0.9 sekunder. Dette er i nærheten av søketiden på 0.7 s i en datamengden på 14 MB. Antar et konstantledd på ca 0.2 sekunder. Frem til 500 treff øker søketiden raskt til 2.9 sekunder. Dette gir et tillegg pr treff på 0.0044 sekunder. Fra 500 treff og oppover øker søketiden saktere. Postene leses ikke. Det er et tillegg på ca 0.00014 sekunder pr treff. Søketiden kan dermed uttrykkes:

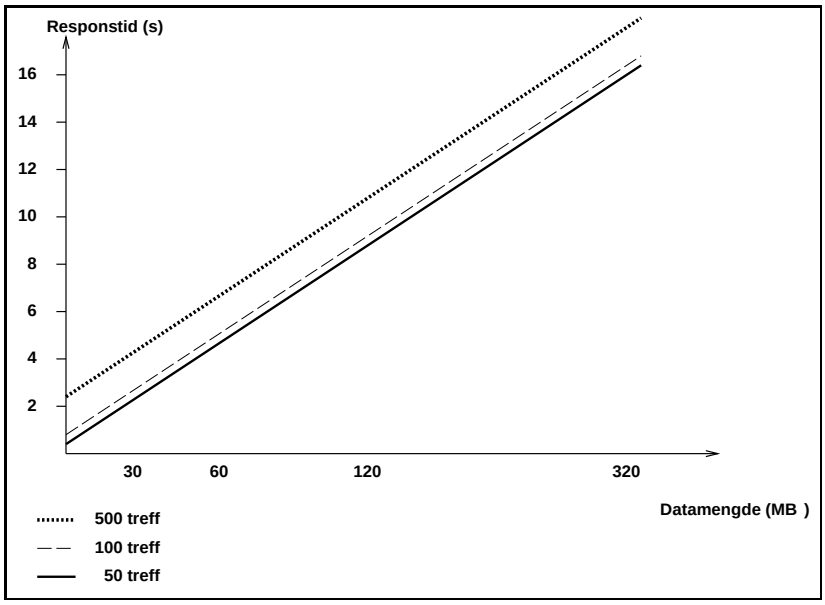
```
Responstid (s)=
0.2 + Datamengde(MB) / 20 + 0.0044 * Antall leste treff + 0.00014 * Antall treff.
```

Under min testing var responstiden mye avhengig av antall treff. Figur 7.4 viser responstiden som funksjon av antall treff Linjen har en knekk ved 500 treff der postene ikke lenger leses fra reservoaret. Dette kan imidlertid endre seg ved nye betingelser. Datamengden i Wilsesamlingen vil øke til det dobbelte når alle bildene er lagt inn. En tenker seg senere at flere samlinger skal legges inn og at det skal være mulig å søke i alle bildene samtidig. Figur 7.5 viser viser responstiden som funksjon av datamengden ved ulike antall treff. Det er vist tider helt frem til reservoarets maksimale lagringkapasitet på 320 MB. Det kommer frem at for større datamengder vil responstiden være mest avhengig av søketiden i dataene.

I de eksisterende løsningene har det ikke vært standard å lese så mange som 500 treff på en gang. Spesielt dersom ikoner overføres sammen med trefflisten blir dette ikke aktuelt. Det har vært vanlig å kun lese 5 eller 10 bilder av gangen. Tiden for å lese postene til treffene i reservoaret ville da blitt minimal. Det er unødvendig at søket skal stoppes for



Figur 7.4: Responstid som funksjon av antall treff



Figur 7.5: Responstid som funksjon av datamengde

treff som ikke leses. Hadde dette vært ungått, kunne det siste leddet vært sløyfet. Dette kan tenkes gjort ved å modifisere søkeprogrammet.

WWW-tjeneren for OS/2 bruker ca 4 sekunder ekstra på å starte søkeprogrammet under DOS. Under testingen har jeg tatt tiden på kjøring av programmet fra DOS-kommandolinjen. For å få den totale tiden ved bruk av et WWW-klient-program må det derfor legges til 4 sekunder.

7.4.2 Frekvensliste

Ved søking etter frekvenslisten for fylker og tiår kan søkingen skje parallelt i flere kolonner. Søkehastigheten vil da avhenge av antall kriterier. Tiår eller fylke sammen med slutttaggen tar opp 2 vinduer. Dette gir en maksimal søkehastighet på 80 MB/s. Ved 1 eller 2 ekstra kriterier synker den til 40 MB/s. Ved 3 til 6 ekstra kriterier er den nede på 20 MB/s. I søkingen etter frekvenslistene er brikken i Count-modus. Dette betyr at søkingen ikke avbrytes for hvert treff. Det går ikke tid til behandling av disse avbruddene slik at responstiden blir uavhengig av antall treff. I alt foretar frekvenssøket for fylke 21 søk og frekvenssøket for tiår 12 søk. Tiden for et enkeltsøk må derfor ganges med disse tallene for å finne den totale responstiden.

Type Søk	Antall ekstra kriterier	Søkehastighet (MB/s)	Tid pr søk(s)	Antall søk	Beregnet søketid(s)	Målt responstid(s)
Frekvens fylke	0	80	0.175	21	3.7	4.0
	1 - 2	40	0.35	21	7.3	7.8
	3 - 6	20	0.7	21	14.7	15.6
Frekvens tiår	0	80	0.175	12	2.1	2.3
	1 - 2	40	0.35	12	4.2	4.5
	3 - 6	20	0.7	12	8.4	9.0

Tabell 7.2: Beregnete og målte responstider for Frekvenssøk

Av tabell 7.2 fremgår den teoretiske og den faktisk målte responstiden for frekvenssøk med et ulikt antall ekstra kriterier. Tallene stemmer meget godt med de beregnete optimalverdiene. Det går liten tid ekstra med til prosessering av data utover selve søkingen.

7.4.3 Sammenligning med Oracle-databasen

Det er vanskelig å foreta en god sammenligning på tidsforbruket mellom Oracle-databasen og MS160. Oracle ligger på en DEC/Alpha 96 MB RAM og 10 GB Disk. MS160 er installert på en 486 PC med 66 MHz klokkefrekvens 16MB RAM og 500 MB Disk. Reservoaret til MS160 har i tillegg 32MB RAM tilgjengelig. Maskinen med Oracle er altså adskillig kraftigere. Rent prismessig er den også mye dyrere; med pris på ca 500 000 koster den over 10 ganger mer enn en PC med tilleggsutstyr. Avgjørende for testresultatet vil det også være at selve Oracle-databasen ligger på harddisk. Aksesstiden vil være langt større enn for MS160 der reservoaret er et halvlederlager. Enkelte database-operasjoner kan kreve mye lesing fra disk. Ved JOIN-operasjoner kan hele tabeller måtte leses inn minnet. En mest mulig lik test ville foregått på samme type maskin der relasjonsdatabasen legger hele datamengden i minnet.

Testingen vil likevel si noe om hvordan databaseløsningene er i forhold til hverandre. Variasjonene for ulike typer søk for den enkelte maskin vil gi et sammenligningsgrunnlag. En

vil kunne se på tiden også da ingen av løsningene har entydig bedre maskinvare enn den andre. Det vil derfor være mulig å antyde hvor god en MS160-applikasjon er sammenlignet med dagens database.

Testingen har også verdi fordi den sier noe om forskjeller i responstid for maskiner som faktisk er i bruk. Deltakerene i Bauta-prosjektet må i valget om de skal ta i bruk MS160 sammenligne med maskinen de allerede har til rådighet.

For Oracle databasen er det ingen tilgjengelig WWW-tjener. En del av den nye funksjonaliteten er dessuten ikke tilgjengelig i det gamle WWW-grensesnittet. SQL gir imidlertid støtte til bruk av både boolske operatorer og telling slik at den samme funksjonaliteten kan testes i begge systemene.

Jeg har laget testprogrammer. For å måle responstiden for søking etter trefflisten har jeg basert meg på bruk av søkeprogrammet til WWW-grensesnittet. Jeg har modifisert det noe for blant annet å få til logisk søking. Som ved MS160-grensesnittet skrives de 500 første treffene ut. For å lage frekvenslistene har jeg laget et program av SQLsetninger i `sqlplus`. Tellingen foregår ved hjelp av SQL-kommandoen COUNT(). Tiden for utførelsen av programmene måler jeg ved programmet `time`.

Type Søk	Antall treff	Responstid MS160 (s)	Responstid Oracle(s)
Motiv = Munkholmen	10	0.9	1.9
Motiv = sildoljefabrikk	17	1.0	2.0
Motiv = sildefiske	396	2,7	3.8
Motiv = mann	9847	4.1	16.1
Sted = Blommenholm	7	0.9	0.5
Sted = Trondheim	131	1.5	1.0
Sted = Asker	923	3.2	3.3
Sted = Oslo	17976	5.4	27.2
Sted = Oslo Motiv = portrett	10373	4.3	23.2
Sted = Oslo Motiv = Ibsen	19	1.0	9.3
Sted = Blommenholm + Asker	930	3.2	4.3
Sted = Blommenholm + Asker - Skaugum	163	1.6	2.0
Datering = 1879–1879	2	0.9	3.4
Datering = 1939–1940	10291	4.4	18.8
Avbildet = Nansen	18	1.0	0.6
Avbildet = Hamsun	108	1.3	1.1
Fotograf = Skjevlo	1	0.9	0.5
Fotograf = Lumholtz	731	3.0	3.6
Fotograf = Wilse	48375	9.6	1:24.3
Fotograf = Wilse Sted = Blommenholm + Asker - Skaugum	163	1.6	45.7
Alle felter = Munkholmen	10	0.9	37.6
Alle felter = Sildefiske	568	2.9	39.2

Tabell 7.3: Sammenligning av responstid for søk etter treffliste

Av tabell 7.3 fremgår responstiden for både MS160 og for Oracle databasen for ulike type

søk. For de fleste typer søk med få treff er responstiden lav for begge. Oracle utfører også logisk-søking raskt dersom ingen av enkeltkriteriene har mange treff. I enkelte tilfeller er Oracle raskere enn MS160.

Ved mange treff bruker Oracle meget lang tid. Søking etter fotograf Wilse som omfatter mesteparten av databasen tar uakseptabelt lang tid. Årsaken ligger nok mye i at svært mange JOIN-operasjoner må utføres. Mesteparten av tiden går nok med til å lese data fra disk, men en del tid går sannsynligvis med til å selve utføringen i minnet.

For enkelte typer søk med få treff bruker Oracle også mye tid. Dette er typisk søk der enkeltkriteriene har lav selektivitet. Eksempel på dette er søking etter Wilse og Blommenholm. Selv om det til sammen bare utgjør 7 treff tar søkingen lang tid. At Wilse har nesten 50000 treff er nok ansvarlig for det . Sannsynligvis utfører Oracle JOIN-operasjoner med alle 50000 postene. Fordelene med MS160 sin parallellisering av det logiske søket kommer her frem. De 7 treffene hentes direkte. Ikke på noe tidpunkt er det nødvendig å håndtere alle treffene til enkeltkriteriene.

Noe innlesningsfelt for søking i alle felter fantes ikke det tidligere WWW-grensesnittet. Responstiden for søking i alle felter med Oracle har jeg fått frem ved å søke etter ordet i alle de vanlige tekstfeltene. I WWW-prosjektet ble det beskrevet hvordan det var særlig tidkrevende for Oracle-databasen å søke i flere felter ved bruk av SQL-operatoren OUTER JOIN. Ved søkingen i alle felter tas denne i bruk. Det kommer klart frem at MS160 er langt raskere. Som tidligere beskrevet er imidlertid funksjonaliteten til denne typen søk med MS160 ikke helt god.

Type Søk	Kriterier	Antall treff	Responstid MS160	Responstid Oracle
Frekvens tiår		55526	2.3	45.2
	Sted = Blommenholm	7	4.5	11.1
	Sted = Asker	923	4.5	11.3
	Sted = Oslo	17976	4.5	58.4
	Sted = Asker Motiv = mann	38	4.5	66.5
	Sted = Oslo Motiv = mann	382	4.5	66.0
	Sted = Oslo + Asker Motiv = mann	420	9.0	67.0
Frekvens fylker		55526	4.0	9.7
	Datering= 1939-1940	10290	7.8	12.4
	Datering= 1939-1940 Motiv = mann	1427	7.8	13.6
	Datering= 1939-1940 Motiv = mann Avbildet = N	8	15.6	14.9

Tabell 7.4: Sammenligning av responstid for frekvenssøk

I tabell 7.4 er det oversikt over tidsforbruket for ulike typer frekvenssøk. Som det kommer frem greier Oracle best å optimalisere tellingen av fylkene. Den er mye raskere enn tellingen av tiår til tross for at det telles flere ganger. Dette kan ha sammenheng med at SQL-setningen utføres enklere. Ved fylker reduseres det først fra STED-tabellen og ikke fra FOTO-tabellen. Tabellen STED innholder langt færre poster. Avgjørende kan det også

være at ved "Frekvens fylker" sammenlignes bare de to første sifrene i kommunekoden med et tall. Ved "Frekvens tiår" gjøres to sammenligninger for et intervall. Datoen må konverteres til Oracles interne datoformat.

Sammenlignet med Oracle er MS160 klart raskest for de fleste typer søk. For søking med mange kriterier med frekvens fylker er de imidlertid ganske like. Det er imidlertid vanskelig å si om dette hadde vært tilfellet med mere like testvilkår. I de fleste tilfeller der tidsforbruket til Oracle er høyt er det fordi JOIN-operasjoner på tabeller med mange poster må utføres. Mye data må leses fra disk under utføringen av operasjonen. Søkingen med MS160 virker likevel mer stabil. Den virker kun avhengig av antall treff. Responstiden med Oracle er mer variabel. Det er vanskelig å si på forhånd hvor krevende søket er og hvor godt det optimaliseres.

7.4.4 Konsekvens av nett-tilknytning

En av målsettingene med Bauta-prosjektet er å gjøre bildeinformasjonen tilgjengelig via datanett. En stor del av brukerene vil da ikke befinne seg på samme sted som databasen. Tiden det tar å overføre bilder og referanseinformasjon vil da måtte legges til søketiden. I WWW-prosjektet ble det vist at for de aller fleste søk var overføringstiden langt større enn søketiden. Under uttestingen med MS160 er det konstatert at søkingen vil skje enda raskere. Den relative nedgangen i total responstiden blir da likevel lav. For de søkene som Oracle optimaliserer dårlig vil det også bli en stor resultatforbedring.

Det regnes med at i fremtiden vil overføringskapasiteten for de fleste nettforbindelser bli langt større. Fordelene med MS160 kan da øke. Mellom universitetene i Norge er det allerede i dag et supernett med en båndbredde på 32 Mbit/s. NBR vurderer å knytte seg til dette nettet. Bauta-basen kan i så fall bli tilgjengelig i det norske universitetsmiljøet med langt større overføringshastighet.

Maskinen som brukes med MS160 er langt billigere enn arbeidstasjonen. Dette kan løse noe av problemet med lang overføringstid. PCer med MS160 kan plasseres ut på flere steder. Det vil da lokalt være mulig å søke raskt uten å være begrenset av båndbredden på datanettene.

7.5 Fremtidige endringer og utvidelser

Under diskusjonen om valg av løsning for funksjonaliteten kom jeg med mange forslag til mer avansert funksjonalitet. De fleste av disse ble imidlertid ikke tatt i bruk av praktiske årsaker. Med en viderutvikling av grensesnittet til Bautabasen bør flere av disse kunne tas i bruk. Dette kan skje med eller uten bruk av en MS160-søkebrikke.

Jeg foreslo å lage flere deskriptorer for å lette gjenfinningen av bilder. Disse kan lages ved ulike oppslagsverk. Det studeres allerede idag hvordan dette kan gjøres. Mer strukturert og utfyllende referanseinformasjon vil ikke være avhengig av MS160 for at det skal komme til nytte. Brikken vil heller ikke sette begrensninger. Ved hjelp av informasjonen vil det kunne gis bedre støtte til søk med uklare kriterier. Det kan gis forlag til relaterte søkekriterier. Alternativt kan det automatisk også søkes etter andre ord med en tilsvarende mening.

En funksjon jeg foreslo var å også søke etter antall treff for ord alfabetisk nær søkekriteriet. Dette vil tilsvare det BIBSYS kaller nabosøk. Det vil ikke være avhengig av innlegging av nye data om semantiske sammenhenger. Mange av ordene som er alfabetisk nærme vil ofte ha en lignende mening.

De løsningene jeg valgte tok stort sett i bruk MS160. Som beskrevet kan imidlertid mange former for funksjonalitet kunne gjøres like rast i programvare. Frekvensfordeling for en liten del av datamengden vil kunne finnes like effektivt i programvaren. MS160 er best egnet for å finne frekvensfordelingen i store datamengder der kriteriene er kjent på forhånd. Med innlegging av nye funksjoner kan det fokuseres mer på grensene for når søking gjennom hele datamengden med MS160 er en fordel. Med denne kunnskapen kan det finnes optimale verdier for når søkebrikken bør tas i bruk.

En del modifiseringer kan gjøres for å forbedre virkemåten til min løsning. Det foretas ingen kontroll av treffene ved søking i alle felter. Denne har en så stor feilprosent at dette bør gjøres. Som sagt vil dette føre til problemer ved mange treff. Frekvens-fordelingen vil dessuten ikke kunne bli korrekt uten at alle felter kontrolleres. Kontroll av treffene kan realiseres for å avklare hvor godt disse problemene lar seg løse.

I min demonstrasjon stoppes søket uansett for hvert treff. Dette gav lang responstid dersom søkekriteriet gav treff mange for bilder til tross for at bare de 500 første ble returnert. Som tidligere skrevet kan det ved mange treff være interessant å kun skrive treffposisjonen maksimalt et fastsatt antall ganger. For resten av søket leses kun antall treff. Dette vil gi færre stopp og raskere søking. På grunn av problemer med funksjonen som returnerer antall treff og treff posisjoner fikk jeg ikke testet dette, men dette vil være en naturlig utvidelse.

Jeg valgte å ikke parallellisere søkingen etter trefflisten. Tidstapet var lavt med de dataene jeg søkte i. Med større datamengder vil imidlertid tidstapet bli større ved søking med få kriterier. Selve parallelliseringen vil være enkel, men den vil kreve en sortering av treffene etter søket. Av hensyn til presentasjonen bør trefflisten være sortert etter ID. En innføring av en slik funksjon kan avklare konsekvensene.

7.6 Endringer av MS160 søkebrikke

Konstruksjonen til MS160 er ikke fastlagt. Det foretas stadig forskning på hvordan virkemåten kan bedres. Gjennom min bruk av brikken til søking i Bauta-basen har jeg lagt merke til enkelte ting som gjorde realiseringen vanskelig. En endret funksjonalitet ville bedre mulighetene til søking i strukturerte datamengder.

Uheldig er det at det ikke er noen funksjon for å nulle ut treff fra enkeltvinduer. Med en slik funksjon i brikken kunne problemene med store huskelengder som gir ekstra treff vært unngått. Ved slutten av hver post burde vinduer settes til ikke lenger å rapportere treff.

Skyggen som oppstår etter hvert treff ved søking i report-modus burde også vært unngått. I MS160-grensesnittet til Bauta-basen vanskeliggjorde dette parallelliseringen av søkingen. Skyggen kan gjøre at treff i poster som søkes parallelt ikke blir registrert. Utviklere av MS160 bør se på mulighetene for å hindre dette fenomenet

Bruken av dedisert RAM vanskeliggjør og fordyrer bruken av MS160. Det ville vært en fordel om dataene kunne leses direkte fra det ordinære minnet. I dagens løsning er en avhengig av at store mengder RAM kjøpes inn og kun brukes til reservoar for MS160. Hadde reservoaret utgjørt en del av minnet kunne det ta i bruk så stor minneplass som det trengte. Resten kunne frigjøres til bruk av andre applikasjoner. Programmer ville også kunne ta i bruk MS160 mer direkte. Søkebrikken kunne søke rett i data lagret som variabler.

Kapittel 8

Konklusjon

Jeg har i denne rapporten vist hvordan et MS160 grensesnitt kan realiseres mot bildearkivet Bauta. Gjennom dette grensesnittet er det mulig å søke hurtig i en av Norges største databaser med kulturhistoriske bilder. Den første fotosamlingen som er lagt inn, er Norsk Folkemuseums Wilse-samling, som i alt består ca. 100 000 fotografier. Senere vil også andre samlinger kunne registreres på denne måten

I oppgaven har jeg studert metoder som kan gjøre et søkegrensesnitt mer brukervennlig. Jeg har forsøkt å legge inn funksjonalitet i min demonstrasjon som gjør brukeren bedre i stand til å finne relevante bilder.

Fordelene med dette systemet er:

- Hurtig søking: Bruk av MS160 gir en lav responstid. Dette er avgjørende for brukervennligheten. Sammenlignet med dagens Oracle-database er tidsforbruket lavt.
- Stabil responstid: Responstiden er stabil ved at hvilke søkekriterier som brukes har liten innvirkning.
- Støtte til utvidet funksjonalitet: Søking med boolske operatorer og generering av frekvenslister utføres effektivt av brikken.
- Lav pris: Idag brukes en kostbar arbeidstasjon til søking i databasen. Sammenlignet med denne vil kostnadene til innkjøp av PC med MS160 være lav. Overføringstiden over datanettet kan idag ofte bli stor. Bruk av MS160 kan åpne for at det blir økonomi til å plasseres databasen flere steder. Søking kan da foretas lokalt med lav responstid.

Søkesystemet har imidlertid også ulemper:

- Brikken gir ikke en god støtte til søking i strukturerte datamengder der postene har variabel lengde. For å gi en korrekt funksjonalitet kan det foretas en videre kontroll i programvaren. Jeg har ikke implementert dette, men regner det for sannsynlig at kontrollen kan bli kostbar.
- Venstretrunkering for søking i enkeltfelter blir vanskelig å realisere. Deltakerne i Bauta-prosjektet har tidligere ønsket en slik funksjonalitet.
- De eksisterende grensesnittapplikasjonene har enkelte mangler. De bærer preg av å ikke være helt ferdig utviklet. Biblioteket jeg valgte har en feil som gjør at det kan bli returnert gale treffposisjoner. Programmeringen skjer dessuten på et så lavt nivå at en videre utvikling kan bli tidkrevende.

I rapporten har jeg kommet med en rekke forslag til hvordan funksjonaliteten kan utvides. Av tidsmessige ble imidlertid det meste av dette ikke implementert. Det kan vurderes hvorvidt noe av dette dette skal tas i bruk ved en videreutvikling av brukergrensesnittet. Funksjonaliteten kan både legges inn som en del av MS160-grensesnittet eller som en del av dagens relasjonsdatabase.

En av Bauta-prosjektets målsetninger er å la det bli mulig å søke i referanseinformasjonen til kulturhistoriske bildesamlinger. MS160 kan gjør dette på en effektiv måte. En står imidlertid overfor problemer som må løses før en et MS160-grensesnitt kan tas i bruk. Gjennom denne rapporten har deltakerne i Bauta-prosjektet fått et grunnlag for å vurdere bruk av MS160 til søking i databasen.

Vedlegg A

Utdrag av tabelldefinisjoner i Bauta-basen

```
CREATE TABLE FOTO (
        FotoID                  INTEGER         PRIMARY KEY,
        Invnr                   VARCHAR2(25)    NOT NULL,
        Aksesjonsnr             VARCHAR2(18),
        Innkomst_aar            DATE,
        Innkomst_kommentar      VARCHAR2(3),
        Registrerings_dato      DATE,
        Registrerings_sign      VARCHAR2(4),
        Klausul                 VARCHAR2(50),
        Produktnr               VARCHAR2(15),
        Datering_fra            DATE,
        Datering_til            DATE,
        Datering_kommentar      VARCHAR2(80),
        Tittel                  VARCHAR2(100)
)

CREATE TABLE TYPE (
        Typekode                NUMBER(3)       PRIMARY KEY,
        Typebetegnelse          VARCHAR2(50),
        UNIQUE(Typekode,Typebetegnelse)
)

CREATE TABLE KLASS (
        System                  VARCHAR2(2),
        Klasskode               VARCHAR2(10),
                PRIMARY KEY (System, Klasskode),
        Forklaring              VARCHAR2(100)
)

CREATE TABLE OMRAADE (
        Omraade_kode            VARCHAR2(4)             PRIMARY KEY,
        Omraade_navn            VARCHAR2(40),
        UNIQUE (Omraade_kode,Omraade_navn)
)
CREATE TABLE STED (
        StedID                  INTEGER         PRIMARY KEY,
        Omraade_kode            VARCHAR2(4),
        Omraade_presisering     VARCHAR2(40),
        Adresse                 VARCHAR2(40),
        Gnr_Bnr                 NUMBER(7),
                FOREIGN KEY (Omraade_kode) REFERENCES OMRAADE,
                UNIQUE (Omraade_kode, Omraade_presisering,
                        Adresse, Gnr_Bnr)
)
```

```
CREATE TABLE JURIDISK_PERSON (
        JPnr                    INTEGER         PRIMARY KEY,
        Etternavn               VARCHAR2(80),
        Fornavn                 VARCHAR2(80),
        Yrke                    VARCHAR2(80),
        Foedt_etablert_aar      DATE,
        Foedt_etablert_komm     VARCHAR2(15),
        Doed_nedlagt_aar        DATE,
        Doed_nedlagt_komm       VARCHAR2(15),
        StedID                  INTEGER,
        JPtype                  CHAR(1),
                FOREIGN KEY (StedID) REFERENCES STED,
                UNIQUE (Etternavn, Fornavn, Yrke, Foedt_etablert_aar,
                        Doed_nedlagt_aar)
)


# * Utfyllende_info er lagt ut i egen tabell pga. lagringshensyn
# * og siden en del foto trolig ikke benytter dette feltet.

CREATE TABLE FOTO_UTFYLLENDE_INFO (
        FotoID                  INTEGER         PRIMARY KEY,
        Utfyllende_info         VARCHAR2(1000),
                FOREIGN KEY (FotoID) REFERENCES FOTO
)

# * Sign_Paaskrift er lagt ut i egen tabell pga. lagringshensyn
# * og siden en del foto trolig ikke benytter dette feltet.

CREATE TABLE FOTO_SIGN_PAASKRIFT (
        FotoID                  INTEGER         PRIMARY KEY,
        Sign_Paaskrift          VARCHAR2(80),
                FOREIGN KEY (FotoID) REFERENCES FOTO
)

CREATE TABLE EKSEMPLAR (
        EksID                   INTEGER         PRIMARY KEY,
        FotoID                  INTEGER         NOT NULL,
        Typekode                NUMBER(3)       NOT NULL,
        P_bygning               VARCHAR2(12),
        P_rom                   VARCHAR2(8),
        P_vegg_reol             VARCHAR2(10),
        P_Dato                  DATE,
        P_Sign                  VARCHAR2(20),
        F_Hoeyde                NUMBER(4,2),
        F_Bredde                NUMBER(4,2),
        Bev_tilstand            VARCHAR2(100),
                FOREIGN KEY (FotoID) REFERENCES FOTO,
                FOREIGN KEY (Typekode) REFERENCES TYPE
)

# * De feltene som antas å være minst benyttet for EKSEMPLAR er
# * pga. lagringshensyn lagt ut i tabellen EKSEMPLAR_ANNET

CREATE TABLE EKSEMPLAR_ANNET (
        EksID                   INTEGER         PRIMARY KEY,
        MidlP_Plass             VARCHAR2(80),
        MidlP_Dato              DATE,
        F_Annet                 VARCHAR2(100),
        Montering               VARCHAR2(100),
                FOREIGN KEY (EksID) REFERENCES EKSEMPLAR
)

# * Motiv er repeterende felt
```

```
CREATE TABLE FOTO_MOTIV (
        FotoID                  INTEGER         PRIMARY KEY,
        Motiv                   VARCHAR2(100)   NOT NULL,
                FOREIGN KEY (FotoID) REFERENCES FOTO
)

# * Repronr er repeterende felt

CREATE TABLE EKS_REPRONR (
        EksID                   INTEGER         PRIMARY KEY,
        Repronr                 VARCHAR2(15)    NOT NULL,
                FOREIGN KEY (EksID) REFERENCES EKSEMPLAR
)

# * Restaurering er repeterende felt

CREATE TABLE EKS_RESTAURERING (
        EksID                   INTEGER         PRIMARY KEY,
        Restaurering            VARCHAR2(80)    NOT NULL,
                FOREIGN KEY (EksID) REFERENCES EKSEMPLAR
)

# * Alternativ tittel (og kode) er repeterende felt

CREATE TABLE FOTO_ALTERNATIV_TITTEL (
        FotoID                  INTEGER         PRIMARY KEY,
        Alt_Tittel_Kode         VARCHAR2(2)     NOT NULL,
        Alt_Tittel              VARCHAR2(100)   NOT NULL,
                FOREIGN KEY (FotoID) REFERENCES FOTO
)

# * M:N-relasjon

CREATE TABLE FOTO_KLASS (
        FotoID                  INTEGER,
        System                  VARCHAR2(2),
        Klasskode               VARCHAR2(10),
                PRIMARY KEY ( FotoID, System, Klasskode),
                FOREIGN KEY (FotoID) REFERENCES FOTO,
                FOREIGN KEY (System, Klasskode) REFERENCES KLASS
)

CREATE TABLE SAMLING (
        SamlingID               NUMBER(4)       PRIMARY KEY,
        Eier_JPnr               INTEGER         NOT NULL,
        Samling_navn            VARCHAR2(80),
        Samling_katalognavn     VARCHAR2(80),
        Samling_beskrivelse     VARCHAR2(200),
                FOREIGN KEY (Eier_JPnr) REFERENCES JURIDISK_PERSON
)
# * M:N-relasjon

CREATE TABLE EKS_SAMLING (
        EksID                   INTEGER,
        SamlingID               NUMBER(4),
                PRIMARY KEY (EksID, SamlingID),
                FOREIGN KEY (EksID) REFERENCES EKSEMPLAR,
                FOREIGN KEY (SamlingID) REFERENCES SAMLING
)

# * Tlf er repeterende felt

CREATE TABLE JP_TELEFON (
        JPnr                    INTEGER         PRIMARY KEY,
```

```
        Tlf                     VARCHAR2(16)     NOT NULL,
                FOREIGN KEY (JPnr) REFERENCES JURIDISK_PERSON
)

CREATE TABLE JP_TELEFAX (
        JPnr                    INTEGER          PRIMARY KEY,
        Fax                     VARCHAR2(16)     NOT NULL,
                FOREIGN KEY (JPnr) REFERENCES JURIDISK_PERSON
)

CREATE TABLE JP_EMAIL (
        JPnr                    INTEGER          PRIMARY KEY,
        Email                   VARCHAR2(30)     NOT NULL,
                FOREIGN KEY (JPnr) REFERENCES JURIDISK_PERSON
)

# * Alternative navn er repeterende felt.

CREATE TABLE JP_ALTNAVN (
        JPnr                    INTEGER          PRIMARY KEY,
        Alt_Etternavn           VARCHAR2(80),
        Alt_Fornavn             VARCHAR2(80),
                FOREIGN KEY (JPnr) REFERENCES JURIDISK_PERSON
)


# * M:N-relasjon

CREATE TABLE FOTO_JP (
        FotoID                  INTEGER,
        JPnr                    INTEGER,
        Rollekode               NUMBER(2),
                PRIMARY KEY (FotoID, JPnr, Rollekode),
                FOREIGN KEY (FotoID) REFERENCES FOTO,
                FOREIGN KEY (JPnr) REFERENCES JURIDISK_PERSON
)

# * M:N-relasjon

CREATE TABLE FOTO_STED (
        FotoID                  INTEGER,
        StedID                  INTEGER,
                PRIMARY KEY (FotoID, StedID),
                FOREIGN KEY (FotoID) REFERENCES FOTO,
                FOREIGN KEY (StedID) REFERENCES STED,
        Relasjonskode           NUMBER(2)        NOT NULL
)
```

Bibliografi

[Baeza-Yates 92] Ricardo A. Baeza-Yates: **Text Retrieval: Theory and Practice.** IPIF Transactions A Vol: A-12 1992 s. 465-76.

[Bauta 92] Lars N. Aune, Mette R. Raabel, Johan Seim, Morten N. Sjølyst, Odd Are Svensen, Thomas B. Svensen: *Bauta, Bilder i automatisk arkiv*, Prosjektoppgave (IS), IDT, NTH, November 1992.

[Bauta 94] Stein Langørgen, Hans Peter Lindemann: *World-Wide Web som grensesnitt mot bildearkivet Bauta*, Prosjektoppgave, IDT, NTH, Mai 1994.

[Bjåstad 93] Håvard Bjåstad, Geir Bruskeland: *SPRINT, Search Phone Records In No Time* Prosjektoppgave, IDT, NTH, April 1993.

[Blake 94] Blake, Consens, Kilpelainen, Larson, Snider,Tompa: **Text/Relational Database Systems: Harmonizing SQL and SGML** *Proceedings of the first international Conference ADB-94* 1994.

[Brown] Brown, Callan, Croft, Moss: **Supporting Full-Text Information Retrieval with a Persistant Object Store** *4th International Conference on Extended Database Technology* 1994.

[Bøen 93] Øystein Bøen, Erik Rondeel: *SQL for the MS160, Integrating RDB technology with special-purpose hardware*, Hovedoppgave, IDT, NTH, Desember 93.

[Cooper 88] William S. Cooper: **Getting beyond Boole** *Information Processing and Management* Vol. 24 No. 3 1988 s. 243–48.

[Eide 93] Tor Eide, Torbjørn Rognes: *Search Application Framework*, Prosjektoppgave, IDT, NTH, April 93.

[Elmasri 89] Ramez Elmasri, Shamkant B. Navathi: *Fundamentals of Database Systems*, 1989

[Erlandsen 92] Roger Erlandsen: *Registrering av norsk fotohistorisk materiale*, Notat, Senter for fotoregistrering 1992.

[Garcia-Molina 92] Hector Garcia-Molina, Kenneth Salem: **Main Memory Databasesystems: An Overview.** *IEEE transactions on Knowledge and Data Engineering.* Vol: 4 Iss:6 Desember 1992 s. 509-16.

[Glassco 93] Richard A. Glassco: **Evaluating Commercial Text Search and Retrieval Packages** *Information Technology and Libraries* Vol: 12 Iss:4 Desember 1993 s. 413–21.

[Grøvlen 93] Øystein Grøvlen: **MS160 Meteorological Database Application** Notat, IDT ,NTH, 1993.

[Hofstad 93] Knut Hofstad, Ståle Løland, Per Scott: *Norsk Dataordbok* Universitetsforlaget 1993.

[Hollaar 92] Lee A. Hollaar: **Implementation and Evaluation of a Parallel Text Searcher for very Large Text Databases** *IEEE transactions on Knowledge and Data Engineering.* 1992 s. 300- 307

[Kambayashi 94] Yahiko Kambayashi, Hiroki Takakura, Shintaro Meki: **Data Structure and Algorithms for a new Hardware Technology** *Proceedings of FODO 93* s 164-96.

[Kristensen 93] Jaana Kristensen: **Expanding End-Users Query Statements For Query for Free Text Searching with a Search-aid Thesaurus** *Information Processing and Management* Vol. 29 No. 6 1993 s. 733-44.

[Løkken 93] Eystein Løkken, Frode Høgseth: *Søkesystem for foretaksindeksen med MS160 Search Engine*, Prosjektoppgave, IDT, NTH, April 1993.

[Løkken 94] Eystein Løkken, Jens Åge Ellingsrud: *Søk i strukturerte data med MS160 Search Engine*, Hovedoppgave, IDT, NTH, Februar 1994.

[Mathisen 94] Roger Mathisen: *Et Bildebehandlingssystem. Innhentning, lagring og overføring av store mengder bildedata*, Hovedoppgave, IDT, NTH, Januar 1994.

[Mitkas 94] Pericles A. Mitkas, Surya P. Sastry: **High-speed Text Search Based omn a Programmable Signal Processor** *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences* 1994.

[MPiRE 94] MicroWay MRT: **MPire, Multi Purpose information Retrieval Environment Release 1.0, User Manual**, Mars 1994.

[MS160API 94] MicroWay MRT: **MS160API Beta 2.1, Programmers Guide, Reference Manual**, November 1994.

[Radecki 88] Tadeusz Radecki: **Trends in Research in Information Retrieval - The Potential for Improvements in Conventional Boolean Logic** *Information Processing and Management* Vol. 24 No. 3 1988 s. 219–27.

[Salton 89] Gerard Salton: *Automatic text processing : the transformation, analysis, and retrieval of information by computer*, 1989

[Saxton 90] Lawrence V. Saxton, Vijay V Raghavan: **Design of an Integrated Information retrieval/ Database Management System** *IEEE transactions on Knowledge and Data Engineering.* Vol. 2 No. 2 Juni 1990 s. 210–19.

[Sieverts 93] Eric G. Sieverts, Marten B. Hofstedte**Software for information storage retrieval testet evaluated and compared Part I - VII** *The Electronic Library* Vol.9 No. 3 s. 145–154. , Vol.9 No. 6 s. 301–317. , Vol.10 No. 1 s. 5–19. , Vol.10 No. 4 s. 195–208. , Vol.10 No. 6 s. 339–357. , Vol.11 No. 2 s. 73–91. , Vol.12 No. 1 s. 21–27.

[Sift 85] *User Guide To Searching in Sift*, Februar 1985

[Stephansen 92] Eirik Stephansen, Kjell Sverre Jerijaervi: *Fundamental Software for Fuzzy Information Retrieval Based on New Hardware* Hovedoppgave, IDT, NTH, Desember 1992.

[Stephansen 93] Eirik Stephansen: **MS160 ultra fast non-numeric coprocessor**, *Microprocessors and Microsystems*, Vol. 17, No. 9, November 1993, s. 561–564.

[Visschedijk 93] Ankie Visschedijk, Forbes Gibb: **Unconventional Text Retrieval Systems** , *Online and CDRom review*, Vol. 17, No. 1, 1993, s. 11–23.

[Østbye 93] Jon Birger Østbye: *Utkast til feltkatalog for NKKMs Edb-prosjekt.* Notat 1.4.1993.

Vedlegg B

Kildekode

Følgende filer er vedlagt:

- Utlegging på Reservoar
  - `hentbase.pc`
  - `parse.c`
  - `charsets.c`
  - `parse.h`
  - `charsets.h`
  - `common.h`
  - `delbase.c`
  - `skrivbas.c`
- Søking
  - `hovedmeny.html`
  - `hjelp.html`
  - `soek2.c`
  - `merinfo2.c`
  - `soek.cpp`
  - `merinfo.cpp`
  - `kall.cpp`
  - `kall.h`
- Testprogrammer
  - `tid.pc` -testing av tid for treffliste på Oracle.
  - `fylke.sql` -testing av tid for frekvens fylke på Oracle.
  - `tiaar.sql` -testing av tid for frekvens tiaar på Oracle.