Sammendrag

I denne rapporten blir det sett nærmere på datavarehus-teknologi. Et datavarehus er en database hvor man samler data fra flere systemer og legger dem best mulig til rette for analyse. I dette ligger det at man tar vare på alle data, kobler dem til et tidspunkt og organiserer de slik at de er lette å søke i. Ledelsen får på denne måten tilgang til historiske data og bedre anledning til å gjennomføre avanserte analyser av dataene. I tillegg vil genereringen av rapporter og søk i datamengden kunne utføres separat fra de vanlige systemene, og vil dermed ikke påvirke responstiden til disse.

Rapporten går først inn på en del relevant fagterminologi. En forklaring på OLTP (On-Line Transaction Processing) og OLAP (On-Line Analytical Processing) etterfølges av en mer detaljert gjennomgang av datavarehuset, teorien bak det og hvordan det kan brukes. Deretter rettes søkelyset mot de problemene man har på Peterson Ranheim AS og andre lignende norske bedrifter. Det vurderes om et datavarehus vil kunne løse disse problemene, og hvordan. Som et svar på hvordan man kan løse problemene uten å overskride eventuelle kostnadsrammer introduseres «stjerneskjema». Dette er en av de mest vanlige teknikkene brukt til å konstruere datavarehus med relasjonsdatabaser som utgangspunkt. Bruk av stjerneskjema og fremgangsmåten for å konstruere et datavarehus på denne måten blir derfor gjennomgått. Til slutt fokuseres det på noen forskjellige sluttbrukerverktøy som kan benyttes i forbindelse med et datavarehus. De deles inn i flere kategorier og noen eksempler fra hver kategori blir presentert.

Forord

Denne rapporten er resultat av et litteraturstudium gjennomført i forbindelse med faget 45073 Databehandling Prosjekt. Oppgaven som ligger til grunn for litteraturstudiet er gitt av bedriften Peterson Ranheim AS etter samtaler med veileder Torgeir Kruke og faglærer Roger Midtstraum. Målet med oppgaven har vært å gi en oversikt over et fagområde som er relativt nytt, og å sette dette i sammenheng med konkrete problemer som eksisterer i Peterson Ranheim og mange andre norske bedrifter.

Rapporten bærer litt preg av at dette er et relativt nytt fagområde. Mye av grunnlaget kommer fra såkalte «whitepapers» som er funnet på Internett. Disse rapportene er av blandet kvalitet, og mange omhandler de samme tingene. Store deler av forståelsen kommer derfor fra to bøker og en større rapport som måtte bestilles fra USA. Det var i USA trenden først startet, og det er også her de fleste produsentene kommer fra. Ellers har det vært vanskelig å finne andre lett tilgjengelige informasjonskilder. Det har vist seg at til tross for den varierende kvaliteten så er Internett den største informasjonskilden på nye fagområder. Gjennom World Wide Web er artikler fra diverse datamagasiner tilgjengelig, og mange av hjemmesidene til produsentene inneholder relevant informasjon. I tillegg har diskusjonsgruppene vært nyttige, da spesielt diskusjonsgruppa comp.databases.olap, der flere anerkjente personer innen fagområdet jevnlig skriver inn.

Jeg vil med dette rette en takk til den veiledning og støtte jeg har fått fra Peterson Ranheim AS ved Torgeir Kruke, og til faglærer Roger Midtstraum som velvillig har svart på spørsmål fra meg.

Trondheim, 30. april 1996.

_____

Vegar Kruke

Innhold

Figurer

Tabeller

1. Innledning

De fleste bedrifter og organisasjoner er på utkikk etter nye og bedre måter å utnytte sine tilgjengelige ressurser på. Dette har nærmest blitt et krav for å kunne overleve i dagens konkurransepregede miljø. Et av de heteste temaene i den forbindelse er «data tilrettelagt for beslutningsstøtte», et tema som går på å utnytte de data som ligger i bedriftenes databaser til støtte for ledelsens beslutninger.

I dag er det vanlig for alle bedrifter å ha en eller flere databaser hvor man lagrer data. Dette innebærer at mye nyttig informasjon blir lagret, informasjon som kan være til stor hjelp for ledelsen i bedriften. Det er nemlig ikke bare informasjon om kundeadresser, ordrebestillinger o.l. som lagres, men også informasjon om utviklingen i markedet, hvor de forskjellige produktene selger best osv. Dessverre er det vanskelig å nyttiggjøre seg av denne informasjonen fordi det innebærer en analyse av de grunnleggende dataene. Strukturen på databasene fører til at det hovedsakelig er folk ved IT-avdelingen som har kunnskap nok til å gjøre dette. Dersom man tillater vanlige brukere å utføre avanserte søk mot databasene risikerer man at tjenermaskinen må slite med feilaktige oppgaver som tar dagevis å fullføre. Dette vil samtidig ha en negativ innvirkning på responstiden til alle andre som jobber mot databasen, en situasjon som i aller høyeste grad er uønsket. En annen del av problemet er at ledelsen ikke alltid vet hvilken informasjon de er ute etter, noe som gjør det enda vanskeligere for IT-avdelingen. Den beste måten å løse dette på er å la lederne få prøve ut forskjellige beregninger og analyser på egen hånd, uten at dette påvirker databasen ellers.

En måte å oppnå dette på er å flytte data over til en egen database som er bedre egnet for å støtte slike søk etter informasjon. Dette er en løsning som flere og flere interesserer seg for. En slik database kaller man da for et datavarehus, en database der man har tilrettelagt dataene for beslutningsstøtte. Datavarehuset vil gjøre det mulig for lederne å få svar på sine spørsmål på typisk under ett halvt minutt, slik at de raskt kan føre tankene sine videre i nye analyser. En forutsetning for at dette skal være mulig er at gode sluttbrukerverktøy med brukervennlige grensesnitt er lett tilgjengelig. Verktøyene forenkler adgangen til datavarehuset slik at ledelsen kan utnytte dets muligheter maksimalt. Samtidig vil man ved å kjøre datavarehuset på en egen maskin unngå at søkene påvirker bruken av de andre databasene. Med et datavarehus og gode verktøy vil altså sluttbrukerne selv kunne lage rapporter, analysere data, finne svar på detaljerte spørsmål og følge egne tankerekker uten å måtte gå via IT-avdelingen først.

For å legge dataene til rette for denne typen behandling anbefales det at de lagres med en annerledes struktur enn i de vanlige databasene. Et eksempel på dette er at data lagres redundant (samme data lagres flere steder), noe som ikke er normalt i vanlige databaser. Dette fører til at korrigeringer og forandringer i datastrukturen er mer komplisert enn i andre databaser, og skikkelig dokumentasjon og vedlikehold av datavarehuset er nødvendig for å unngå problemer. Denne måten å konstruere databaser på er uvant for de fleste IT-avdelinger, og vil derfor kreve en del innsats for å gjennomføres.

Det er fortsatt for tidlig å si om datavarehus kommer til å være en suksess eller ikke, men ting tyder på at bedrifter rundt om liker tanken bak konseptet. I en undersøkelse gjennomført av «The Meta Group», et konsulentfirma i USA, kommer det frem at 95% av 250 IT-ansvarlige i forskjellige bedrifter har konstruert eller planlegger å konstruere et datavarehus i løpet av 1996. Det tilsvarende tallet for en den samme undersøkelsen i 1995 var 15% [DEPO96]. Amerikanske bedrifter er gjennomgående mye større enn norske, men konseptet

med datavarehus er like fullt aktuelt også i Norge. Hvor nyttig denne nye teknologien er for en middels stor norsk bedrift er et av de punktene denne rapporten vil prøve å belyse.

Som en naturlig følge av den økende populariteten til datavarehus har det dukket opp flere artikler og rapporter på fagområdet. Mange av disse er dessverre «bestillingsverk» fra bedrifter som lager datavarehusapplikasjoner, er derfor lite teknisk detaljerte og bringer sjelden nye ideer til fagområdet. Utviklingen skjer også så raskt at de begrensninger og problemer som eksisterer i dag gjerne er løst noen uker senere. Dette gjør at påstander i en del artikler kan bli feilaktige og mindre relevante i løpet av kort tid, noe man må passe på dersom man ønsker å studere fagområdet nærmere. Men den raske utviklingen skaper samtidig spenning rundt emnet og alle bedrifter gjør klokt i å holde et lite øye på situasjonen.

1.1 Oppgaven

Denne oppgaven er gitt av Peterson Ranheim AS, en treforedlingsbedrift like nord for Trondheim. De vurderer å forbedre sine rutiner for behandling av data og informasjon innen bedriften, og ønsker i den forbindelse en bedre forståelse for fagområdet datavarehus. Oppgaveteksten lyder som følger:

> *«Datavarehus (Data Warehouse) er et begrep som har begynt å samle stor interesse både hos brukere og leverandører. Dette er et konsept som tar sikte på å tilfredsstille behovet for sammenstilling, rapportering og analyse av data.*
>
> *I likhet med mange større bedrifter ser Peterson-konsernet muligheter for å utnytte datavarehus til sin fordel. I første omgang ønskes en oversikt over fagfeltet med dets muligheter og metoder. Det er også aktuelt med en vurdering av mulige verktøy og forslag til hvordan et pilotprosjekt kan igangsettes. Pilotprosjektet bør omfatte det som allerede nå er skilt ut i en separat rapporteringsdatabase.»*

I dette ligger det at jeg skal forsøke å finne frem til hvordan et datavarehus fungerer og hvordan dette kan utnyttes av Peterson Ranheim. Begrepet er relativt nytt og det er aktuelt med en vurdering av hvor nyttig det kan være i middels store norske bedrifter. Jeg skal også se litt på hvilken type verktøy som kan passe til bruk i forbindelse med et slikt system.

2. Teori og teknologi

2.1 On-Line Transaction Processing (OLTP)

Siden midten av 1970-tallet har det blitt mer og mer vanlig å registrere ulike transaksjoner i dertil egnede databaser, en prosess som omtales som OLTP – On-Line Transaction Processing. Mye av æren for denne utviklingen må tilfalle dagens relasjonsdatabaser som er optimalisert for denne typen prosesser. Eksempler på data som lagres i slike transaksjonssystemer er salgsdata, lagerdata, produksjonsdata, kundedata og data om ansatte. I mange sammenhenger omtales disse systemene også som operasjonelle databaser, da de benyttes i det operasjonelle miljøet i bedriften.

Datastrukturen og programvaren i slike systemer er gjerne optimalisert for å lette korrigering av innskrevne data, bl.a. ved å ha lite redundante data og normaliserte tabeller. Det benyttes også kompliserte korreksjonssystemer for å forsikre at data ikke forsvinner eller blir forandret utilsiktet. Spesielt viktig er det med effektive låsemekanismer som muliggjør at flere personer kan redigere på samme database samtidig. Uten slike låsemekanismer kan man risikere at forskjellige personer oppdaterer samme data samtidig, og viktig informasjon går tapt. Andre ting som kjennetegner disse systemene er [OLAP95]:

- En bruker aksesserer og bearbeider data en post av gangen
- Responstiden bør ikke overskride 3 sekunder
- Maskinvaren utnyttes konstant (se Figur 2.1)
- Data oppdateres ofte og har relativt kort varighet
- Høy ytelse og høy tilgjengelighet prioriteres

For de tilfellene der man bare skal lese dataene i databasen for så å behandle de videre med andre programmer stilles det andre krav. Slike operasjoner har ikke behov for kompliserte korreksjonssystemer og låsemekanismer, men krever rask tilgang til de ønskede dataene og en effektiv utnyttelse av ressursene. Det er i denne forbindelse man snakker om OLAP.

2.2 On-Line Analytical Processing (OLAP)

OLAP har etter hvert blitt navnet på den analyse som gjennomføres for å trekke ut informasjon fra store datamengder. Dagens marked er preget av sterk konkurranse og stadig skiftende forhold, og som en følge av dette må bedrifter utnytte sine tilgjengelige data bedre. Samtidig øker mengden av data som skal lagres, og kvalitetskravene til informasjonen blir stadig strengere. I takt med dette øker også behovet for mer avansert analyse og bedre fremstilling av disse dataene. Som en følge av at PCen har blitt vanlig i kontormiljøet har brukerne blitt mer komfortable med bruk av dataverktøy, noe som åpner for en bedre utnyttelse av bedriftens databaser. PC-utviklingen kombinert med utviklingen av gode sluttbrukerverktøy medfører at sluttbrukerne nå selv har muligheten til å utføre kompliserte søk og lage detaljerte rapporter, oppgaver som tidligere måtte utføres av de IT-ansvarlige.

En av de som har betydd mye for innføringen av OLAP som begrep er Dr. E.F. Codd. I [CODD93] presenterer han begrepet slik:

> *«OLAP is the dynamic enterprise analysis required to create, manipulate, animate, and synthesize information from [...] data analysis models.»*

Andre momenter som er viktige i OLAP er muligheten til å behandle store mengder data, lage et ubegrenset antall dimensjoner (mer om dimensjoner står i kapittel 2.4), samt gjennomføre uavhengige søk på kryss av disse dimensjonene. Disse søkene skal kunne være ad hoc søk, dvs. tilfeldige søk som benytter seg av ikke tidligere brukte parametre (verdier).

Som et forsøk på å følge opp de 12 kriteriene Dr. Codd kom med i 1980-årene vedrørende OLTP-systemer, kom han i 1993 med 12 regler for evaluering av OLAP-verktøy [CODD93]. Disse reglene har etter hvert blitt kritisert for å være spesialsydd for ett produkt (som lages av Dr. Codds arbeidsgiver), og som et svar på denne kritikken utvidet han reglene med 6 nye i 1995 [OLAP95]. Men til tross for denne utvidelsen føler mange at definisjonen fortsatt er preget av innflytelse fra enkelte programleverandører, og at de er for utydelige til å ha noen særlig nytteverdi. Derfor prøvde Nigel Pendse og Richard Creeth i 1995 å lage en mer generell spesifikasjon av karakteristikkene til OLAP-produkter. De ville ha en spesifikasjon som var teknologiuavhengig og uten tilknytning til noen spesiell program-leverandør, samtidig som den skulle være kort og enkel å huske [OLAP95]:

> *«Five keywords to summarize OLAP is Fast Analysis of Shared Multidimensional Information – or FASMI for short»*

Dette utdypes videre slik:

- *Fast:* Systemet skal kunne gi en respons på ad hoc spørringer i løpet av 1 til 20 sekunder. Vanlig responstid bør være på ca. 5 sekunder. Dersom responstiden blir for lang vil brukeren sannsynlig vis bli forstyrret i tankegangen og kvaliteten på analysen vil forverres. Denne hastigheten er vanskelig å oppnå med store datamengder, spesielt når brukerens spørringer blir kompliserte.

- *Analysis:* Systemet skal kunne håndtere enhver form for forretningslogikk og statistisk analyse som er relevant for applikasjonen og brukeren, samtidig som det skal være brukervennlig nok for den tiltenkte brukeren.

- *Shared:* En del krav til sikkerhet må være implementert i systemet. Det skal være mulig at flere jobber mot samme data samtidig, og også at flere skriver til samme data uten at feilsituasjoner oppstår. På dette punktet svikter mange av dagens OLAP-produkter, da de ofte antar at applikasjonen vil være kun lesbar og ikke skrivbar.

- *Multidimensional:* Dette er nøkkelkravet til et OLAP-produkt; dersom man skulle bruke bare ett ord for å definere OLAP så ville det vært dette. Systemet må tilrettelegge for et multidimensjonalt syn på dataene, inkludert støtte for hierarkier og multiple hierarkier, da dette er den mest logiske måten å analysere bedrifter på.

- *Information:* Med informasjon menes alle de data og avledede informasjon som er nødvendig for å utføre den nødvendige analysen. Det som er relevant i denne sammenheng er hvor mye informasjon produktene kan behandle og hvor mye lagringsplass som kreves for å ta vare på dataene.

Begrepet OLAP kan fortsatt virke litt svevende, og man kan treffe på litt forskjellige tolkninger i litteraturen som omhandler emnet. Dette er ingen uvanlig situasjon for nye fagområder der forandringer skjer raskt og ofte. Noen av de større leverandørene av OLAP-

verktøy prøver å standardisere begrepet ved å danne et råd kalt «The OLAP Council», men det sliter for tiden litt med at ikke alle har tillit til det. Grunnen til dette er at det hovedsakelig består av store leverandører som har kraftig egeninteresse i saken. Likevel får rådet flere og flere medlemmer, og med tiden kan man forhåpentlig vis få fruktbare resultat av samarbeidet.

OLAP-verktøy kan i bunn og grunn sees på som verktøy for å hjelpe ledelsen i å lede bedriften. I så henseende er det en videreutvikling av ledelsesinformasjonssystemene (LIS) som kom på 1980-tallet. Forskjellen ligger bl.a. i datagrunnlaget og i de mulighetene lederne har til å behandle dataene. Mange av LIS-systemene mislyktes på grunn av at utgangspunktet var for dårlig – søppel inn, søppel ut. I tillegg ga ikke systemene lederne gode nok muligheter til å bearbeide dataene slik de ville. Med OLAP kan lederne stille spørsmålet «Hva om?» og få svar nærmest umiddelbart. For å legge dataene bedre til rette for OLAP-verktøy begynner bedrifter nå å konstruere egne datavarehus som gjør det enklere å skaffe til veie dataene.

2.3 Datavarehus

2.3.1 Definisjon

På lik linje med OLAP har datavarehus blitt et begrep som mange snakker om i dataverden i dag, og på samme måte blir ikke folk enig om en klar definisjon på begrepet. W.H.Inmon[1], mannen som startet denne nye trenden innen databaseteknologi, gir i boken «Building the Data Warehouse» følgende formulering [INMO92]:

> *«A data warehouse is a: subject oriented, integrated, time variant, nonvolatile collection of data in support of management's decision making process.»*

Ut i fra dette kan man altså trekke slutningene at et datavarehus skal være

- *subject oriented:* Orientert rundt temaer som er viktig for organisasjonen. Dette kan være ting som kunder, produkter, politikk, konti, osv.
- *integrated:* Det er viktig at dataene integreres slik at de samme dataene ikke er representert på forskjellige måter eller med forskjellig navn. Et eksempel på dette kan være at tabellene "PåLager" og "Inne", som begge inneholder data av samme type i de operasjonelle databasene, blir overført til en enkelt tabell som heter "Lager" i datavarehuset.
- *time variant:* Tidsaspektet er veldig viktig i datavarehus, og man må alltid knytte dataene opp i mot et visst tidspunkt. Det kan også være aktuelt å følge tidsavhengige trender som man ikke uten videre kan se på grunnlag av de operative data.
- *nonvolatile:* Til slutt er det viktig at alle data lastes inn i datavarehuset på en gang, hvorpå de kan bli aksessert, men helst ikke forandret.
- *in support of management's decision making process:* Det skal være mulig for ledelsen å benytte dataene i den beslutningsprosessen som er en del av deres hverdag, dvs. datavarehuset skal danne grunnlaget for dataanalyse.

[1] William H. Inmon er også kjent som Bill Inmon.

Dette er nok det nærmeste man kommer en definisjon på begrepet, og formuleringen er også den mest brukte. Inmons bok regnes for å være en grunnleggende bok innen fagfeltet i dag, og det meste av teorien rundt datavarehus bygger på denne boken.

2.3.2 Forskjeller mellom et datavarehus og vanlige databaser

For å skille vanlige databaser[1] fra datavarehus kan man se på hvilke data som lagres i de forskjellige systemene, og hvilken struktur det er på dataene. I Tabell 2.1 er det listet opp flere forskjeller som illustrerer hvordan de forskjellige systemene kan behandle data.

Primitive data/ operasjonelle data	Beregnede data/ datavarehus data
Detaljerte	Summerte eller på en annen måte behandlet
Korrekte på gjeldende tidspunkt	Representerer verdier over tid
Tjener saksbehandlerne	Tjener administrasjonen
Kan oppdateres	Lite behov for oppdatering
Krav til prosessering kan beregnes på forhånd	Krav til prosessering kan ikke beregnes på forhånd
Aksesseres en enhet av gangen	Aksesseres et sett av gangen
Transaksjonsdrevet	Analysedrevet
Orientert rundt applikasjoner	Orientert rundt analysen
Kontroll av oppdateringer er viktig med tanke på eierskap	Kontroll av oppdateringer er av liten betydning
Høy tilgjengelighet	Avslappet tilgjengelighet
Ikke redundante	Redundante
Statisk struktur, variabelt innhold	Fleksibel struktur
Mengden data involvert i en prosess er liten	Mengden data involvert i en prosess er stor

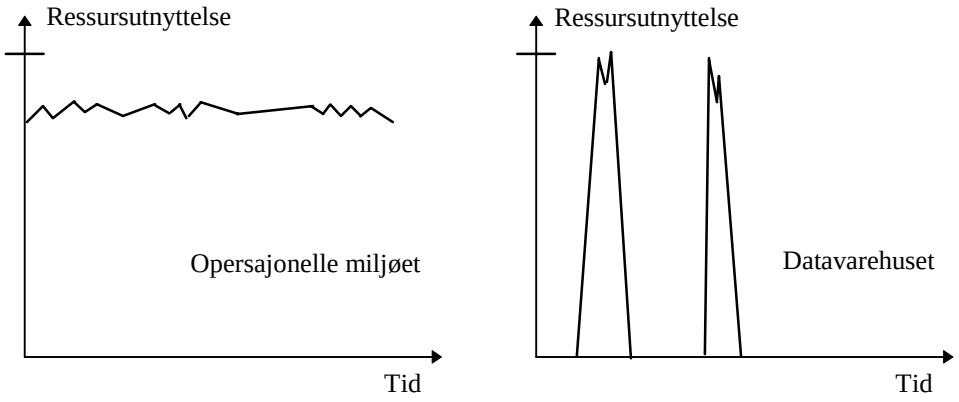
Tabell 2.1 Tabell over forskjeller mellom data i transaksjonsdatabaser og datavarehus [INMO92]

Disse forskjellene i data kan også benyttes som en begrunnelse for å innføre datavarehus. Man kan tenke seg at en enkelt database skal tilfredsstille alle de behovene som er nevnt i denne tabellen. Flere konflikter vil da oppstå når forskjellige behov skal tilfredsstilles samtidig. Ved hjelp av rå og brutal kraft (maskinvare) kombinert med smarte algoritmer og datastrukturer kan man sikkert oppnå en midlertidig tilfredsstillende kapasitet, men hvor nyttig vil dette være i lengden? Etter hver som størrelsen på databasen øker vil ytelsen bli dårligere, og til slutt vil det ikke bli mulig å kompensere for dette med innkjøp av ny maskinvare. Resultatet blir irritasjon både hos administrasjonen og saksbehandlerne.

Hvis man ser på ressursutnyttelsen i disse to forskjellige miljøene ser man enda et argument for å skille de fra hverandre. Det operasjonelle miljøet utnytter den vanlige databasen på en annen måte enn hvordan et datavarehus vil bli utnyttet. Dette illustreres i Figur 2.1, der man ser at det operasjonelle miljøet ligger jevnt på en ressursutnyttelse litt under det maksimale. Til sammenligning utnyttes ressursene i datavarehusmiljøet maksimalt i kortere perioder. Nå vil det ikke være helt korrekt å addere disse grafene sammen, men resultatet av dette vil illustrere at det er to forskjellige krav som må tilfredsstilles, og at en atskillelse av disse kan

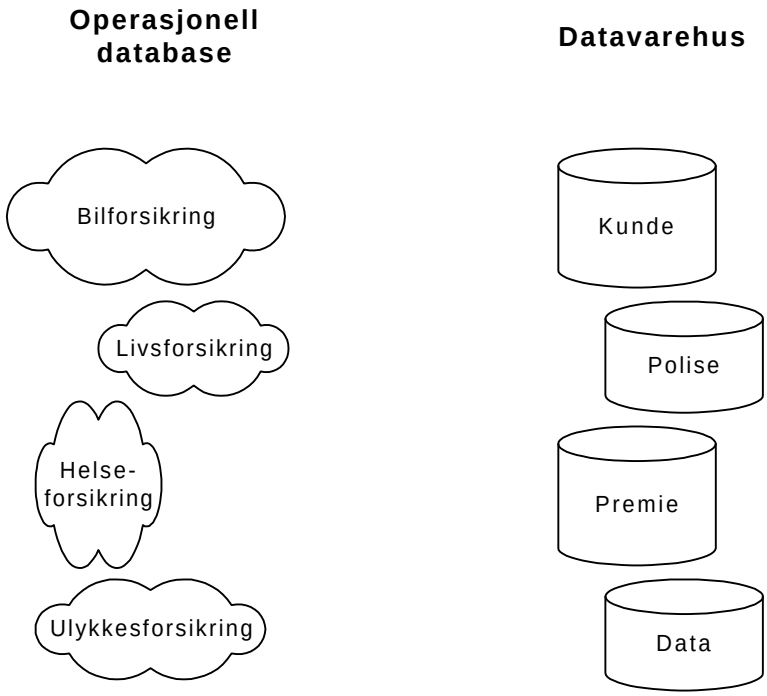
[1] Med vanlige databaser mener man her de allerede eksisterende systemene som alle større bedrifter benytter seg av, gjerne relasjonsdatabaser, men også eldre databasesystemer som nettverksdatabaser o.l. Man kaller gjerne dette miljøet for det operasjonelle miljøet, og benytter i den forbindelse også betegnelsen «operasjonelle databaser».

være nyttig. Når det utføres søk og analytiske beregninger i datavarehusmiljøet samtidig som transaksjoner blir registrert i det operasjonelle miljøet påvirker dette ytelsen slik at begge miljøene får dårligere responstid.



Figur 2.1 Ressursutnyttelse i det operasjonelle miljøet sett i forhold til utnyttelsen i datavarehus miljøet [KENA95]

Ved å gå nærmere inn på kjennetegnene for datavarehus kan man se flere forskjeller sammenlignet med operasjonelle databaser. I Figur 2.2 illustreres hvordan man tenker forskjellig ved struktureringen av data, jfr. det første punktet i definisjonen til Inmon.

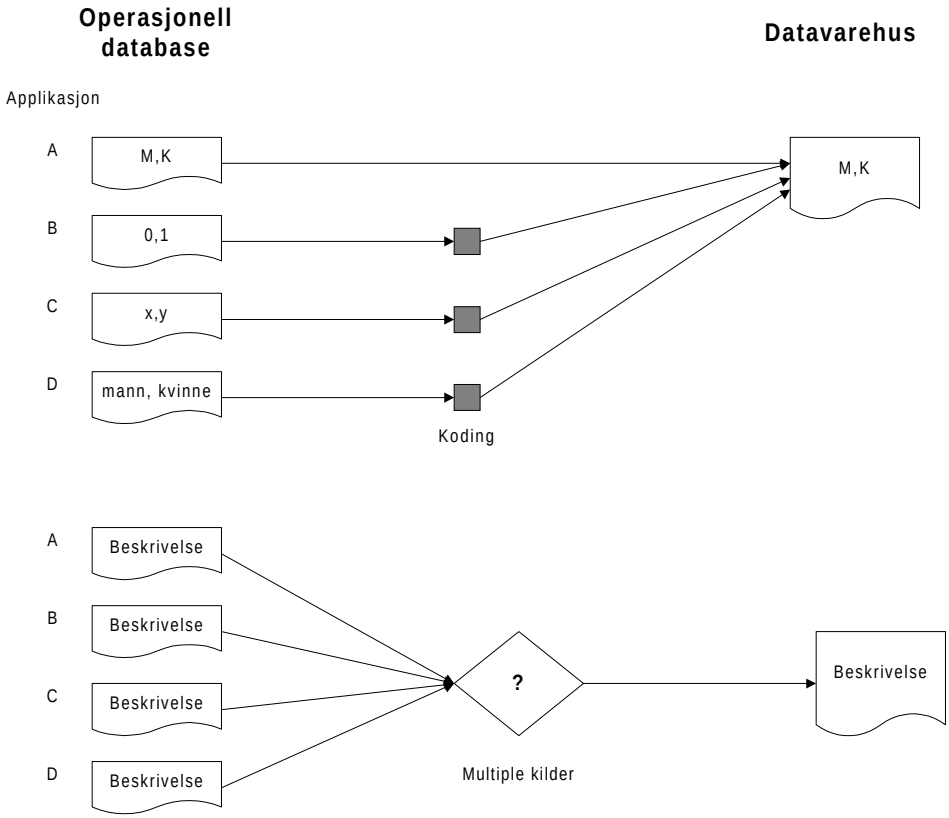


Figur 2.2 Eksempel på data orientert rundt temaer i datavarehus kontra operasjonelle databaser [INMO92]

Det viktige her er måten man tenker på når man skal lage modellen (hvordan dataene organiseres) for datavarehuset. I noen tilfeller vil den allerede eksisterende databasen være

fokusert rundt de samme temaene som er aktuelle i datavarehuset. Dette vil i så fall gjøre det lettere å konstruere datavarehuset og å overføre data mellom databasene.

Kravet til integrering av data fra det operasjonelle systemet til datavarehuset er litt mer konkret og forståelig, se Figur 2.3. Man kan tenke seg at en bedrift registrerer de ansatte i forskjellige databaser, med forskjellig format på representasjonen av kjønn. Grunnen til at informasjon om de ansatte er lagret flere steder kan være så mangt, men ofte har de enkelte delene av bedriften behov for forskjellig informasjon om de ansatte. Lønn kan for eksempel ha sitt system der informasjon om arbeidstider o.l. lagres, mens helseavdelingen har et annet system der informasjon om personalets helse lagres. Når informasjonen fra disse to systemene skal føres sammen i datavarehuset må man passe på at informasjonen om kjønnet til de ansatte blir registrert på kun én måte. Dette innebærer en konvertering av dataformatet under overføringen slik det er illustrert i figuren.

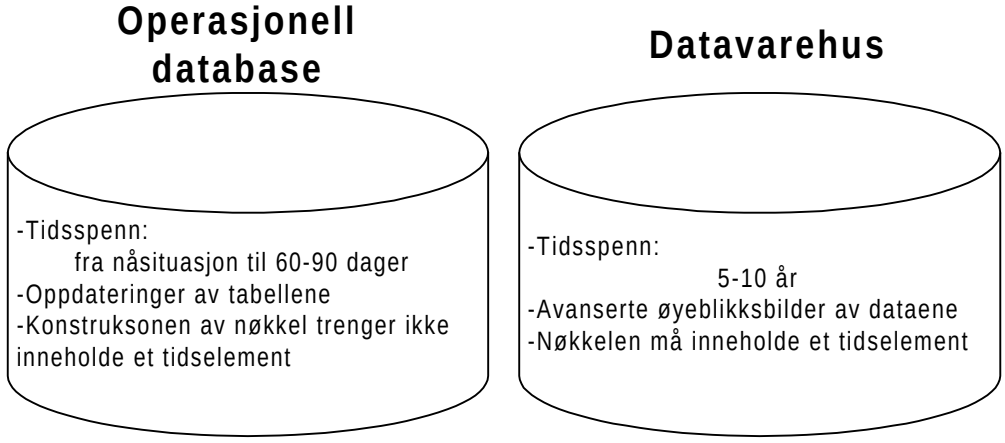


Figur 2.3 Eksempler på integrasjon av data fra det operasjonelle systemet til datavarehuset [INMO92]

Et annet eksempel som illustrerer den integreringen som foregår ved overføringen av data til datavarehuset kan være at et produkt er beskrevet på forskjellig måte i forskjellige systemer. I datavarehuset vil man ha bare én beskrivelse av produktet, og denne bør være god nok til å dekke alle behov som kan oppstå. Dermed må det foregå en konvertering av beskrivelsen, noe som i dette tilfellet vil være vanskelig å automatisere fullt ut. Man kan også se på forskjellige lengdeenheter, forskjellige myntenheter, forskjellige typedefinisjoner, forskjellige tabellnavn o.l. som beskrivelser. Denne siste typen beskrivelser er enklere å konvertere automatisk, slik at alle data blir representert med samme lengdeenhet, myntenhet

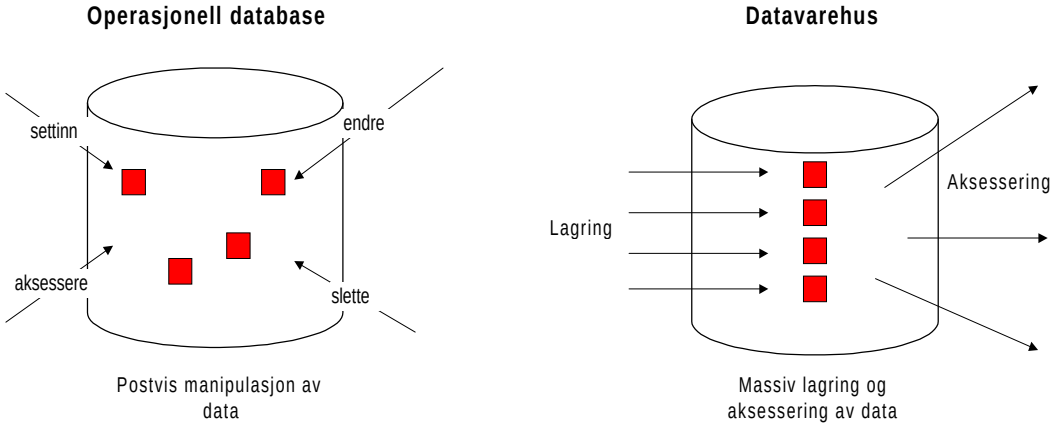
osv. Dette muliggjør bl.a. addering av verdiene, noe som er en viktig egenskap i datavarehus. Addering gjør at man kan summere opp verdier og forenkler bruken av hierarkier.

Tidsaspektet er som tidligere nevnt en viktig del av et datavarehus. Alle data som lagres i et datavarehus må være knyttet til et tidspunkt [INMO92]. I tillegg kan det også være andre aspekter av tid som skiller et datavarehus fra en operasjonell database, noe man kan se av Figur 2.4. Siden datavarehuset skal benyttes til analyse av bl.a. trender i markedet er det naturlig at data fra flere år tilbake er tilgjengelig. Alt etter som hvilke data det er snakk om og hvor stor database man vil ha, lagres dataene i omtrent 5 til 10 år før de flyttes til andre lagringsmedia (som f.eks. båndkassetter). Her eksisterer det varierende behov i forskjellige bedrifter, noe som i noen tilfeller fører til et avvik fra dette tidsanslaget. I den operasjonelle databasen har man ikke like store behov for å ha gamle data tilgjengelig. Man arbeider hele tiden med nåværende verdier, og oppdaterer dataene etter hvert som de forandrer seg. De data som er viktig å bevare blir overført til datavarehuset, og etter hvert som data fornyer seg forsvinner de gamle. Erfaring tilsier at de fleste dataene forblir i slike databaser ca. 60-90 dager før de forandres [INMO92]. På samme måte som i datavarehuset er det en del avvik fra dette tidsanslaget, men det illustrerer likevel forskjellen mellom de to miljøene. Man kan se på den operasjonelle databasen som et bilde av nåsituasjonen, mens datavarehuset på sin side inneholder flere slike bilder knyttet opp til de aktuelle tidspunktene.



Figur 2.4 Forskjeller i behandlingen av tid i de to miljøene [INMO92]

Når det gjelder lagringen og behandlingen av de lagrede dataene er det også her viktige skiller mellom de to typene databaser. I det operasjonelle miljøet foregår det stadige justeringer og oppdateringer av data, mens det i datavarehuset helst ikke skal være nødvendig å forandre på noen data (se Figur 2.5). Den operasjonelle databasen oppdateres og aksesseres postvis, mens datavarehuset blir lastet med data på faste tidspunkt, hvorpå de kan aksesseres av brukerne. Oppdatering av data i dets vanlige betydning eksisterer ikke i datavarehuset, det er bare unntaksvis at man gjennomfører korreksjoner. Dette har konsekvenser for hvordan man konstruerer databasen, da slikt som redundans (lagring av samme data flere steder), normaliseringsgrad o.l. påvirker oppførselen til de to databasene forskjellig.

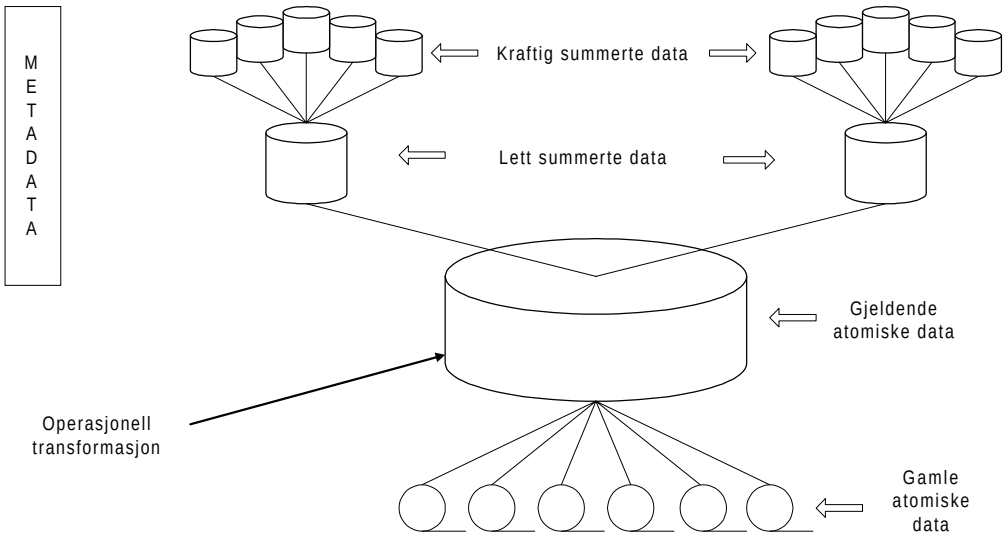


Figur 2.5 Eksempel på hvordan data blir behandlet i de to miljøene [INMO92]

2.3.3 Strukturen til et datavarehus

Et datavarehus blir ikke komplett bare ved å ta vare på en masse data og koble de til et tidspunkt. Det er helt avgjørende at man i tillegg strukturerer dataene slik at ledelsen kan benytte seg av de på en enkel og effektiv måte. Den strukturen som er mest benyttet i dag og som ser ut til å være mest vellykket er illustrert i Figur 2.6. Datavarehuset deles der inn i 5 sentrale deler:

- Kraftig summerte data
- Lett summerte data
- Gjeldene atomiske data
- Gamle atomiske data
- Metadata



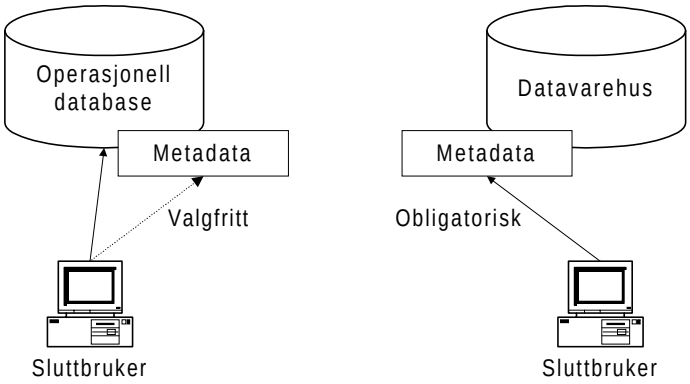
Figur 2.6 Oversikt over strukturen til datavarehuset [INMO92]

Sentralt i strukturen står de gjeldende atomiske dataene som representerer grunnlaget for den analyse som kan utføres på datavarehuset. Alle detaljer er lagret her, og er tilgjengelig for de programmene som utfører søk i datamengden. Det er hit data overføres fra det operasjonelle miljøet etter først å ha blitt konvertert og integrert. Denne delen av datavarehuset må lagres på et medium som er raskt å aksessere, enkelt å skrive til og som tilbyr høy sikkerhet, typisk en harddisk. Tilgangen til dataene for brukerne er avgjørende for suksessen til datavarehuset. Dette har ikke minst kommet frem i forbindelse med f.eks. automatisk genererte rapporter, der mange har uttrykt hvor viktig det er at brukerne kan kontrollere grunnlaget for disse dataene direkte [NEWS96]. Uten denne muligheten vil ikke lederne nødvendig vis kunne stole på de beregnede dataene, og da er hele vitsen borte. Når det gjelder hvor detaljert disse atomiske dataene bør være, så står det mer om dette i kapittel 2.3.6.

Normalt vil mengden atomiske data vokse i størrelse etter hvert som datavarehuset blir eldre. Dersom man ikke gjør noe med dette vil man etter hvert komme i den situasjonen at ad hoc søk og generering av rapporter vil ta for lang tid. Ved å lagre dataene slik at søk o.l. forenkles kan man riktig nok oppnå tilfredsstillende responstider selv med store datamengder, men i lengden vil ikke dette være en akseptabel løsning. Gjennom samtaler med brukerne før, under og etter konstruksjonen av datavarehuset kan man finne de data som ofte etterspørres og/eller beregnes. Med slik kunnskap er det mulig å på forhånd summere viktige data og gjøre de lettere tilgjengelig for brukerne. Det kan være snakk om å summere data som f.eks. salg til en kunde i løpet av en dag, en uke, en måned eller et år. Man skiller gjerne mellom lettere summerte data og kraftig summerte data. Målet er uansett å ha ferdig summerte data tilgjengelig for brukerne slik at responstiden forbedres og antall samtidige brukere kan økes. I et godt konstruert og vedlikeholdt datavarehus vil de fleste av spørsmålene rette seg mot summerte data, og aller mest mot kraftig summerte data (se også kapittel 2.3.9) [INMO92].

Etter hvert som dataene foreldes og mister relevans for brukerne er det viktig å flytte de ut i fra den gjeldende atomiske strukturen og inn i de gamle atomiske dataene. Dersom dette ikke gjøres risikerer man å få problemer med responstiden til tross for bruken av summerte data. Det vanlige er med jevne mellomrom å overføre de data som har blitt gamle nok til et annet lagringsmedium, f.eks. tape (båndkassetter) eller optiske disker. Hvilket lagringsmedium man velger er ikke så viktig, bare det er billig, sikkert og gjør det mulig til å hente informasjonen tilbake igjen dersom nødvendig. I de fleste tilfeller vil man under denne prosessen også fjerne noe av detaljgraden til dataene. Sannsynligheten for at man får behov for detaljerte opplysninger om så gamle data er som oftest så liten at en slik redusering vil være forsvarlig. Som et eksempel kan man anta at fem år gamle data om salg til en gitt kunde skal flyttes ut. For øyeblikket lagres salg av et produkt til denne kunden per uke. Ved overføringen til de gamle atomiske dataene summerer man opp slik at man i stedet får salg per periode (4 uker). I noen tilfeller kan det tenkes at man bare lagrer salg per år, alt dette avhenger av hvilke behov som man ser for seg kan dukke opp senere. Overføringen av data til de gamle atomiske dataene og slettingen i de gjeldende atomiske dataene bør etter hvert foregå automatisk. Men siden det gjerne går en tid (5-10 år) før dataene blir så gamle at de må flyttes ut, er ikke dette noe man må få til å virke fra første stund. Etter at datavarehuset har blitt etablert og tatt i bruk kan man begynne å tenke på dette som et av de neste skrittene.

Den siste delen i strukturen av datavarehuset er metadataene. Disse omtales gjerne som data om data, og inneholder informasjon om det meste vedrørende datavarehuset. Man finner også metadata i det operasjonelle miljøet, men der har de en helt annen rolle enn i datavarehuset, se Figur 2.7. Mens man betrakter metadata på lik linje med dokumentasjon i det operasjonelle miljøet (noe man ser på en gang i blant), er det nærmest obligatorisk for en bruker av et datavarehus å se på metadataene før en analyse settes i gang. I tillegg benyttes metadataene ved overføringen av data fra de operasjonelle databasene til datavarehuset, samt til å holde orden på de forskjellige datastrukturene som eksisterer. Mer om metadata står i det påfølgende kapittelet.



Figur 2.7 Forskjellen mellom bruk av metadata i det operasjonelle miljøet og datavarehuset [INMO92]

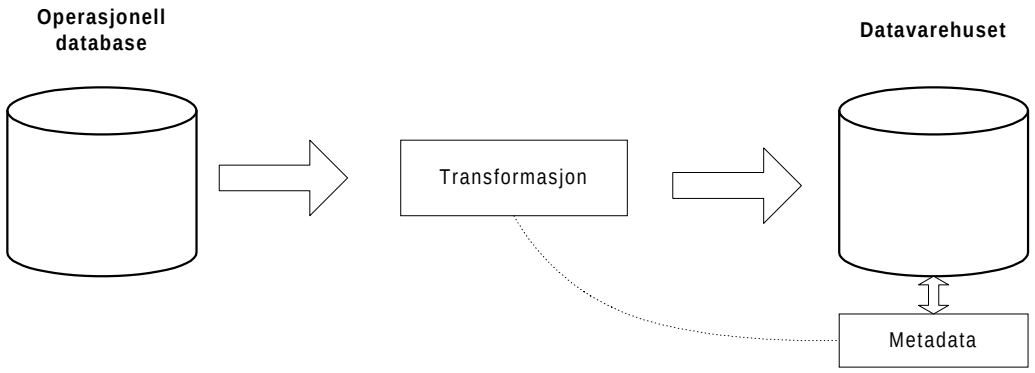
2.3.4 Metadata

Metadata er viktig i ethvert datavarehus, noe som gjenspeiles i omtrent all litteratur som omhandler fagfeltet. Begrepet omfatter veldig mye, fra noe så enkelt som tabellnavnene i en database til kompliserte regler for konvertering av data.

Som det ble forklart i forrige kapittel benyttes metadata på forskjellig vis i det operasjonelle miljøet og i et datavarehus. I tillegg kan man si at det eksisterer to forskjellige hovedtyper metadata i et datavarehus [PERI94]. Den ene typen beskriver de datakildene, attributtene og definisjonene som benyttes, og er laget for databaseadministratorene slik at de kan vedlikeholde dataene og datakildene. Den andre typen er ment for sluttbrukerne og viser hva de forskjellige dataene representerer. Et eksempel på denne siste typen kan være at man har en tabell kalt «omsetning». Metadataene vil da indikere for brukeren hvor i den operasjonelle databasen dataene ble hentet fra, hvordan de ble beregnet og hva de inkluderer.

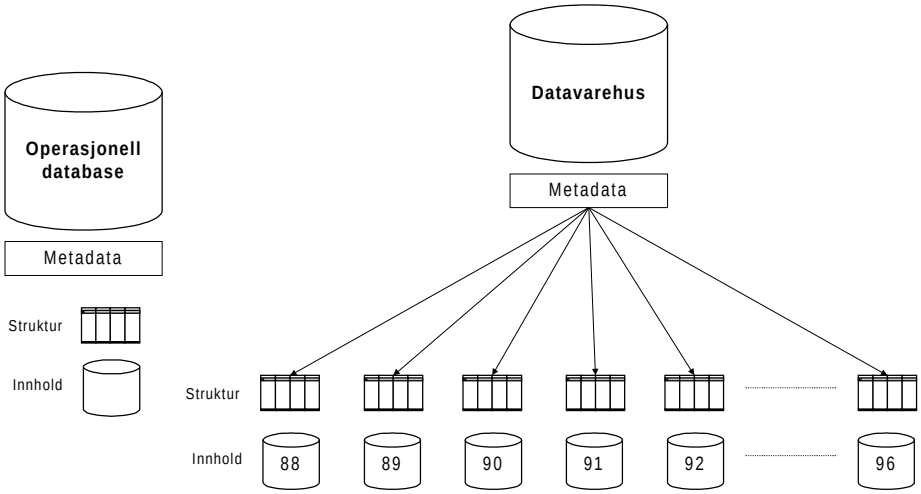
Begge disse typene metadata lagres sammen, men de trenger ikke nødvendig vis ligge på samme sted som selve datavarehuset. Dette er likevel som oftest tilfellet, siden det maksimerer tilgjengeligheten til metadataene.

I Figur 2.8 illustreres en av grunnene til at metadata er så viktig for datavarehuset. Den viser hvordan dataene transformeres når de overføres fra den operasjonelle databasen til datavarehuset. Det er et stort behov for å kontrollere denne transformasjonen, og metadataene i datavarehuset er det ideelle stedet for denne kontrollen.



Figur 2.8 Overføringen av data fra den operasjonelle databasen til datavarehuset ved hjelp av metadata [INMO92]

Som en følge av at data i et datavarehus gjerne lagres i 5-10 år må man regne med at datastrukturen forandrer seg etter hvert. Det er da viktig å ha tilgang til hvilken struktur som gjelder for gamle data slik at man kan aksessere de riktig. Dette er ikke noe problem i den operasjonelle databasen siden den likevel bare inneholder gjeldende data, men for datavarehuset er situasjonen annerledes. Ved å lagre disse datastrukturene i metadataene vil man løse dette problemet, slik det er illustrert i Figur 2.9.



Figur 2.9 Tidsaspektet i et datavarehus gjør at man må kunne håndtere flere strukturer/definisjoner på dataene [INMO92]

Et av problemene man møter på når man skal begynne å tenke på metadata er at man ikke forstår helt hva dette egentlig er. I artikkelen «Planning for a Data Warehouse» kommer Michael Haisten [HAIS95] med følgende uttalelse ang. metadata:

> *«Metadata is the most discussed but least understood. Most people think of a data dictionary first, but this is the least valuable form for definition metadata»*

Det er altså ikke slik at metadata er standardisert og godt forklart. Spørsmål som hvordan man skal håndtere metadata, hvilken programvare som kan benyttes, hvilket format metadata bør være på og mange andre nødvendige detaljer er ting man ikke blir enige om. Det er nok flere grunner til dette, men en av hovedgrunnene er at de forskjellige programleverandørene kjemper for sine egne format. Det har etter hvert blitt opprettet et «Metadata Council» (tilsvarende «The OLAP Council») med representanter fra flere større programleverandører. Det er meningen at disse etter hvert skal definere og standardisere begrepet metadata, men foreløpig har ikke samarbeidet ført til noe konkret. I stedet har faktisk flere ytret seg skeptisk til hva et slikt råd kan prestere, siden det domineres av de store aktørene på markedet [NEWS96]. Noen er også skeptiske til om det i det hele tatt er realistisk å komme frem til en standard for noe som varierer så mye i innhold.

Som en følge av den usikkerheten som eksisterer på dette området er det vanskelig å anbefale spesielle former for metadata. Det kan være lurt å trekke inn brukerne for å få til en god fremstilling av disse dataene, spesielt den delen som han/hun kommer til å benytte seg av selv.

2.3.5 Modellering av et datavarehus

Ved konstruksjonen av datavarehus er det vanlig å modellere strukturen på dataene før man begynner implementasjonen. Man kan ikke regne med å oppnå en ferdig struktur på datavarehuset, men får et godt utgangspunkt for den videre utviklingen. Når man skal modellere datavarehuset er det nyttig å starte med en modell av bedriften. Prism Solutions, et konsulentselskap som fokuserer sterkt på datavarehus, anbefaler sine kunder å følge disse huskereglene i denne fasen [NEWS96]:

- Bland ikke sammen modellen av bedriften og modellen av datavarehuset. Bedriftsmodellen beskriver hvordan bedriften opererer, mens datavarehusmodellen beskriver hvordan man kan analysere og lage rapporter om bedriften.
- Det anbefales å lage tre nivå av bedriftsmodellen som en forløper til datavarehusmodellen. Først bør man ta for seg entitetene til bedriften på et høyere nivå, og ut i fra dette modellerer man tabellene til de spesifikke bedriftsområdene. Til slutt lages en fysisk modell av de aktuelle delene av bedriften.
- En detaljert, fysisk/logisk modell av hele bedriften vil kreve måneder med arbeid og resultere i masse unødvendig informasjon. Det er også en fare for at den allerede er blitt for gammel når man begynner å modellere det første «emneområdet». Begrens derfor detaljgraden i denne delen av modelleringen.
- Datavarehusmodellen kan så konstrueres ved å ta utgangspunkt i det laveste nivået av bedriftsmodellen.

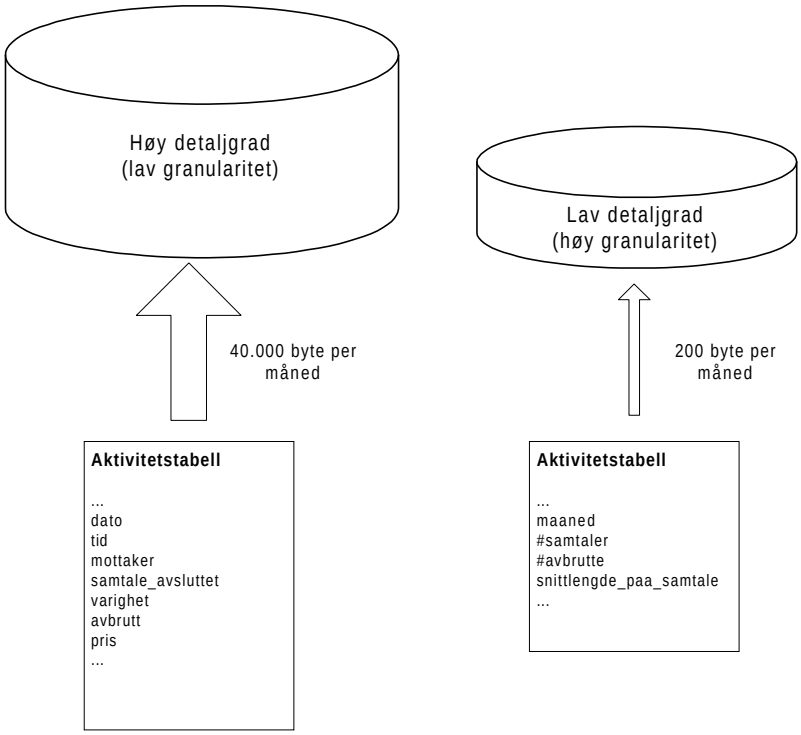
Bruk av disse huskereglene vil innebære ganske mye modellering, men i de fleste tilfeller vil man bli spart for mye arbeid senere.

2.3.6 Granularitet

Granularitet er et ord som benyttes mye i forbindelse med datavarehus, og i følge W.H. Inmon er dette kanskje det viktigste aspektet av alle i et datavarehus [INMO92]. Granulariteten går ut på hvilken detaljgrad man skal ha på de atomiske dataene. Dersom man har et høyt detaljnivå sier man at granulariteten er lav, dersom man har et lavt detaljnivå sier man at granulariteten er høy. Grunnen til at dette er så viktig er at granulariteten er veldig avgjørende for volumet på datavarehuset, samtidig som den påvirker hvilke analyser som kan kjøres mot dataene. Man må balansere disse to aspektene av datavarehuset opp mot hverandre, og forskjellige bedrifter vil ende opp med forskjellige løsninger. I utgangspunktet mener mange at IT-avdelingen bør søke etter en løsning hvor man beholder detaljene og ikke uten videre godtar summering av data (ved overføringen til datavarehuset) [HAIS95]. Skulle det vise seg at størrelsen blir for stor kan man i ettertid summere opp en del av dataene, mens muligheten til å gå andre veien som oftest ikke eksisterer, etter som detaljene går tapt under summeringen. Her må man se på bedriftens behov og gjøre de vurderinger som trengs, det er ikke alltid man trenger alle de detaljene i datavarehuset som eksisterer i den operasjonelle databasen.

Et eksempel som illustrerer betydningen av granularitet kan være nyttig, se Figur 2.10. Anta at det skal lagres informasjon om aktivitetene til et telefonselskaps kunder. En kunde foretar i snitt 200 oppringninger per måned, og for å lagre alle detaljer fra en oppringning kreves det i snitt ca. 200 byte lagringsplass per oppringning. Dette innebærer at det å lagre informasjon tilhørende én kunde krever ca. 40.000 byte lagringsplass per måned. Dersom man i stedet summerer kundedataene under overføringen slik at man bare beholder informasjon vedrørende den måneden samlet vil det innebære en lagringsplass på bare ca. 200 byte per måned (datastrukturen er vist i figuren). Som man ser vil dette innebære betydelige innsparelser i lagringsplass, også fordi færre indeksinnslag er nødvendig. Denne besparelsen kan være avgjørende for hvordan datavarehuset oppfører seg, da store mengder av detaljerte data vil gå ut over responstiden.

I tillegg til mindre lagringskrav kan høyere granularitet også føre til besparelser på prosessorkraft. Sett at man vil vite hvor mange samtaler som ble tatt fra et gitt område forrige måned. Mens man med lav granularitet må lese alle postene for hver kunde (i snitt 200 poster per måned) og summere antallet samtaler, trenger man med høy granularitet bare å lese en post for hver kunde (da den inneholder summen av samtaler gjort av den kunden den måneden). Siden antallet kunder som oftest er stort vil denne forskjellen kunne påvirke responstiden drastisk.



Figur 2.10 Granulariteten har stor innvirkning på størrelsen av databasen [INMO92]

Et lignende eksempel kan benyttes for å vise nytten av å ha lav granularitet. Sett at man skal finne ut om person X ringte opp person Y forrige måned. Denne informasjonen har gått tapt under summeringen av dataene med høy granularitet, og spørsmålet kan ikke besvares. Med lav granularitet, der all informasjon er beholdt, kan man derimot besvare spørsmålet, selv om det krever litt innsats. Hvor mye vekt man skal legge på å kunne besvare slike spørsmål avhenger av hvor ofte de dukker opp. Ofte vil man gå inn for at sjeldne spørsmål ofres til fordel for bedre lagringsplass og raskere respons på søk.

Et kompromiss mellom lav og høy granularitet vil være å benytte flere lag. Dvs. man kan oppbevare de nyeste dataene på en detaljert form for så å summere de når de blir eldre. Dermed kan man oppnå at man har muligheten til å svare på detaljerte spørsmål såfremt de angår data av nyere dato, samtidig som man kan benytte lagringsplassen effektivt. Kombinert med den summeringen av data som ble beskrevet i kapittel 2.3.3 vil dette være en ideell løsning for mange.

2.3.7 Overføring av data

Oppdatering av datavarehuset, dvs. overføring av data fra de operasjonelle databasene til datavarehuset, skjer til faste tidspunkter. Hvor ofte slike oppdateringer gjennomføres varierer etter hvilke behov bedriften har og hvilke data i de operasjonelle databasene det er snakk om. Det kan f.eks. skje hver måned, hver uke, hver natt, og i noen tilfeller også oftere. I dag eksisterer det også systemer som kan oppdateres samtidig som de er i bruk, slik at man kan foreta oppdateringer i arbeidstiden [NEWS96]. Det mest vanlige er likevel å oppdatere hver natt, når systemet ikke er i bruk av andre. Denne oppdateringsfrekvensen gjør det mulig for dataene i de operasjonelle databasene å slå seg litt til ro før de blir overført, samtidig

som at datavarehuset holdes oppdatert. Dersom man overfører data for tidlig risikerer man at man må gå inn i datavarehuset å korrigere, noe som ikke er trivielt (p.g.a. redundans o.l.). Siden strukturen på datavarehuset i utgangspunktet ikke er egnet for korrigering av data kan slikt føre til mye ekstraarbeid for IT-avdelingen.

Overføringsprosessen må etter hvert gjøres automatisk. Det er kun i en innledende fase at man kan tillate seg å legge data inn i datavarehuset manuelt [INMO92]. Dessverre er det ikke dette alltid like enkelt å oppnå, spesielt når man jobber mot eldre databasesystemer. Det eksisterer verktøy som forenkler denne delen, men disse har en prislapp som gjør de mindre aktuelle for flertallet av bedriftene i Norge. Mer informasjon om denne delen av datavarehuset står i kapittel 4.3.

2.3.8 Nettkapasiteten

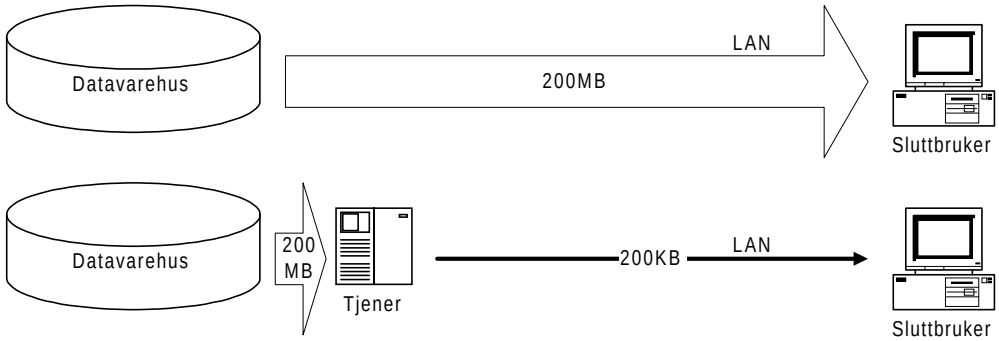
Ved planlegging av et datavarehus er det viktig å ta hensyn til hvordan konstruksjonen vil påvirke nettverket og responstiden over dette. Introduksjonen av et datavarehus vil i de fleste tilfeller medføre en markant økning av nettverkstrafikken, hvor stor denne økning blir avhenger mye av konstruksjonen. Derfor kan det være lurt å vurdere ulike konstruksjonsalternativer også i forhold til nettverkstrafikken.

Slik planlegging har størst betydning når man i tillegg til LAN (Local Area Network) også opererer med WAN (Wide Area Network). Det er linkene i forbindelse med WAN som virkelig sliter når store datamengder skal overføres i tillegg til den allerede eksisterende daglige trafikken. Men selv om man bare skal benytte datavarehuset i forbindelse med LAN kan det være nyttig å tenke litt på nettverket.

I en artikkel i LAN Times [BAUM95] skriver David Baum om en del viktige aspekter knyttet til konstruksjonen av datavarehuset og dets virkning på nettverket. Han peker på en rekke faktorer man bør se på i forbindelse med slike prosjekter, tanker man bør ha gjort på forhånd og forhåndsregler som bør taes for å lette arbeidet. En sammenfatning av dette er:

- Hvor stor båndbredde kreves for å oppdatere datavarehuset?
- Hvor i nettverket bør datavarehusets metadata plasseres i forhold til brukerne og produksjonsdataene?
- Hvilke nettverksprotokoller vil de forskjellige komponentene av datavarehuset kreve for overføring og utveksling av data?
- Hvor er nettverkskapasiteten dårlig, og hvordan kan overføringer av data splittes opp slik at man kan unngå potensielle flaskehalser på disse stedene?
- Vil det kreves egen programvare for å konvertere og/eller overføre dataene blant de forskjellige datatypene?
- Hvilke typer lenker vil overføringen av store mengder data kreve mellom LAN-WAN?
- Vil det bli nødvendig med en generell oppdatering av nettverket helt ut til brukerne?

Hvor mye vekt man behøver å legge på disse punktene er avhengig av hvilken arkitektur man velger på konstruksjonen. Et eksempel på hvordan konstruksjonen kan påvirke mengden data som overføres er illustrert i Figur 2.11. Figuren viser hvordan et søk etter de 10 største kundene til et firma i januar måned kan gjennomføres med to forskjellige arkitekturer. Den første metoden overfører alle kundedata og omsetning for den aktuelle perioden og prosesserer dataene hos klienten. Dette kan innebære overføringer av størrelsesorden 200MB, noe som tar tid selv i raske LAN. Den andre metoden innebærer at man kjører et analyseprogram på selve tjeneren. Dette programmet kan sammen med metadataene utføre et søk på dataene uten å overføre de over nettet først. Bare sluttresultatet blir overført, som kan være i størrelsesorden 200KB. Nå er det viktig å ikke trekke forhastede slutninger ut i fra dette. Det er ikke nødvendigvis den siste metoden som er den beste når man ser på det totale bildet, siden denne metoden vil innebære at all prosessering kjøres på tjeneren og krever store ressurser der. Dessuten vil forskjellen mellom de to arkitekturene avhenge veldig av størrelsen på datavarehuset og størrelsen på rådataene (tallene som benyttes her er bare illustrative).



Figur 2.11 Eksempel på to forskjellige måter å overføre informasjon fra datavarehuset til sluttbrukeren

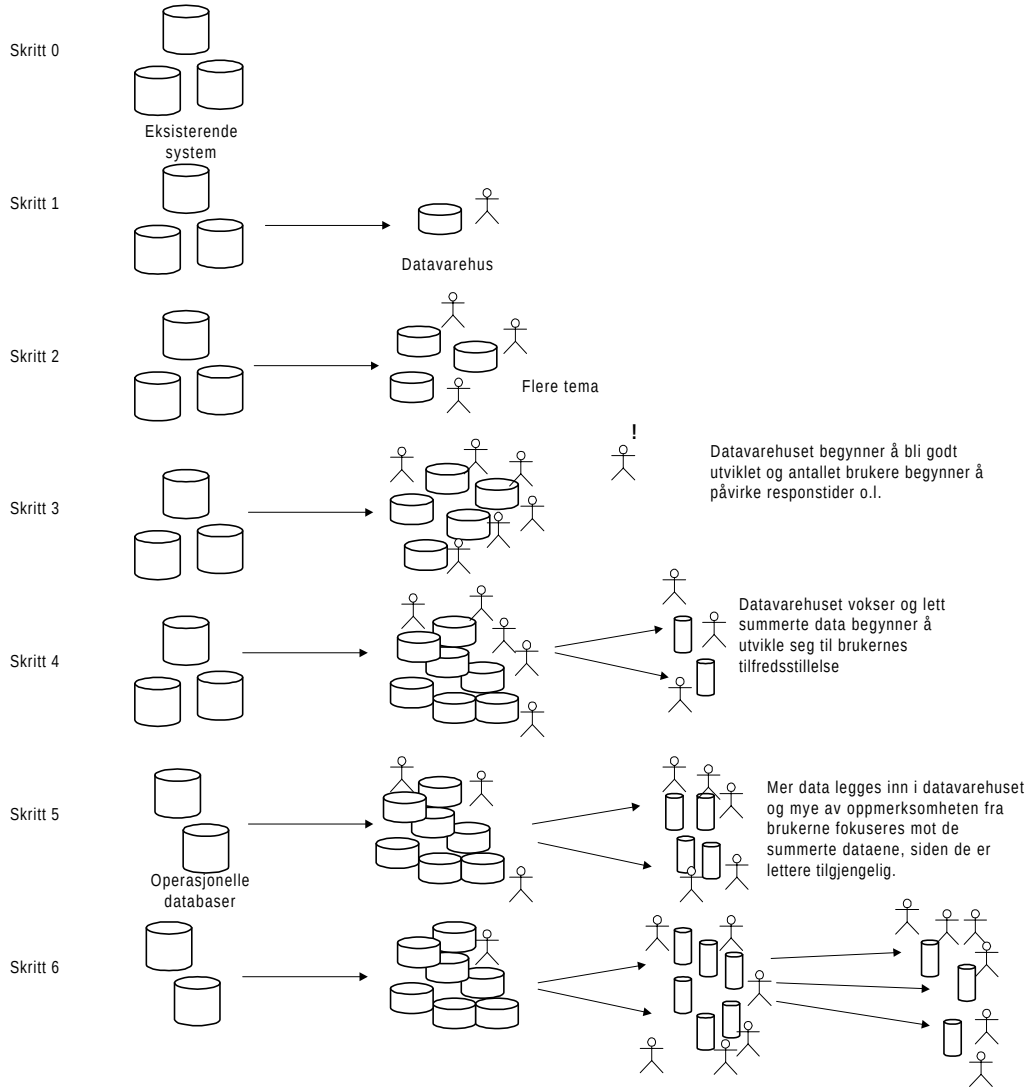
2.3.9 Bruk av et datavarehus

Hvordan innfører man et datavarehus i en bedrift? Hvordan benyttes et datavarehus etter at det har blitt innført? Etter som relativt få har erfaring på dette området i dag er disse spørsmålene vanskelige å svare konkret på. Man kan likevel tenke seg frem til hvordan brukerne ideelt sett kan benytte seg av datavarehuset etter hvert som det utvikles. Innføringen av datavarehuset i bedriften er nemlig noe som går over en lengre periode og gjennomføres skrittvis. På samme måte vil antall brukere og måten de benytter datavarehuset også øke skrittvis.

I Figur 2.12 illustreres hvordan utviklingen av et datavarehus kan skje, der utgangspunktet, skritt 0, er det allerede eksisterende systemet av databaser.

Skritt 1 er begynnelsen på datavarehuset. Man overfører noen av dataene, f.eks. deler av salgsdataene, til en ny database som skal bli datavarehuset. Noen brukere er nysgjerrige og begynner å «leke» seg med analyser av dataene. Etter skritt 2 er datavarehuset utvidet med flere tema og mer data, og flere brukere kommet til. Nysgjerrigheten hos brukerne har gått over til en mer genuin interesse, og de seriøse analytikerne blant ledelsen begynner å benytte seg av det nye systemet. Skritt 3 innebærer at en del av de dataene som var lokalisert i det operasjonelle miljøet har blitt riktig plassert i datavarehuset, og at mange av de mulighetene

som eksisterer har blitt oppdaget av ledelsen. «Alle» vil benytte seg av datavarehuset og kapasiteten begynner å bli så sprengt at brukernes irritasjon øker. For å bøte på dette gjennomfører man i skritt 4 en summering av dataene, slik at brukerne slipper å bruke prosessorkraft på beregninger som er utført tidligere. Dette krever at det har vært en god kommunikasjon mellom brukerne og de IT-ansvarlige slik at riktige data blir summert. Skritt 5 innebærer at brukerne har blitt flinkere til å utnytte de summerte dataene, og gjennom tilbakemelding til IT-avdelingen har de fått tilpasset dataene enda bedre til sine behov. Til slutt kommer skritt 6 der man har både lett summerte og kraftig summerte data. De fleste av brukerne benytter de kraftig summerte dataene, som er detaljert nok i de fleste tilfellene, mens resten bruker de lett summerte dataene. Bare sjelden er det behov for å gå helt ned til de atomiske dataene.



Figur 2.12 Den skrittvise utviklingen til datavarehuset og hvordan brukerne benytter det [INMO92]

Denne fremgangsmåten/situasjonsbeskrivelsen er ingen fasit på hvordan ting skal gjøres eller hvordan brukerne benytter datavarehuset. I noen bedrifter kan det f.eks. være ønskelig å trekke inn summerte data mye tidligere, bl.a. for at brukerne skal kunne gi tilbakemelding på hva de har behov for. Det er også mulig at brukerne ikke vil arbeide mot summerte data i det hele tatt, men bare stoler på de atomiske. Flere har erfaring med at dette siste er tilfellet, spesielt i situasjoner hvor metadataene ikke har vært gode nok [NEWS96]. Men man kan likevel se på oversikten gitt ovenfor som en form for veiledning under utviklingen, og skritt 6 som et slags mål for suksess.

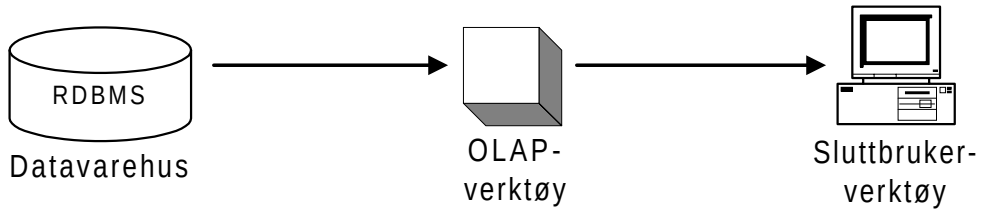
For å få best mulig utnyttelse av de summerte dataene og datavarehuset totalt er det meget viktig med god kommunikasjon mellom brukerne og utviklerne. Dette gjelder i alle fasene av utviklingen, fra første uttesting til datavarehuset er «ferdig». Selv da er det selvsagt viktig med kommunikasjon, ikke minst fordi et datavarehus utvikler seg i takt med bedriften og derfor aldri blir helt ferdig.

2.4 ROLAP eller MOLAP?

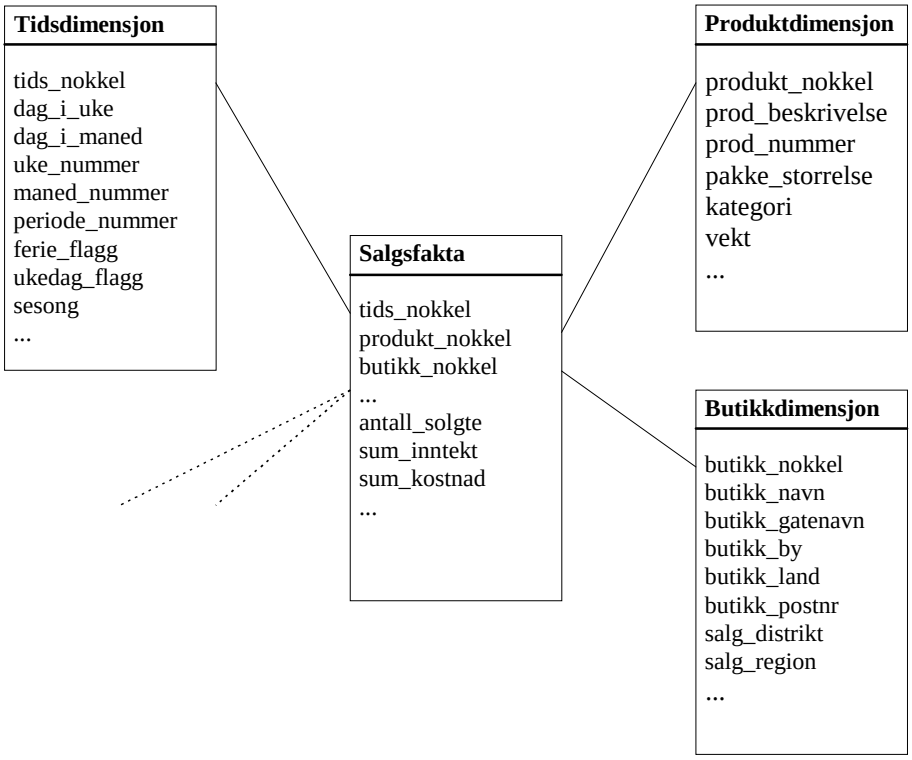
Utviklingen av OLAP har delt seg i to retninger. Den ene retningen mener at dagens teknologi innen relasjonsdatabaser kan benyttes til OLAP, mens den andre mener at multidimensjonale databaser er den beste løsningen for OLAP-verktøy.

2.4.1 Hva er ROLAP?

ROLAP står for «Relational On-Line Analytical Processing». Begrepet viser til bruk av OLAP-verktøy mot relasjonsdatabaser som Oracle, Sybase, Informix, Red Brick, osv. ROLAP-verktøy gir brukerne muligheter til enkelt å utføre analyser og søk i datamengden, forutsatt at dataene ligger til rette for dette (f.eks. ved at man har et datavarehus). Den vanlige arkitekturen til et ROLAP-system er illustrert i Figur 2.13. Man ser at arkitekturen er en flerlags arkitektur med et datavarehus (en relasjonsdatabase) i bunnen, et OLAP-verktøy i midten og et sluttbrukerverktøy på topp. Datavarehuset sørger for at de nødvendige dataene er tilgjengelig for OLAP-verktøyet, som står for analysen og beregningen. Sluttbrukerverktøyet benyttes til å presentere resultatet og til å sette i gang nye analyser. Det eksisterer også verktøy som tar seg av flere deler av denne prosessen, da i første rekke de to siste delene av prosessen (anaslysearbeidet og kommunikasjonen med brukeren).



Figur 2.13 Arkitekturen til et ROLAP-system



Figur 2.14 Et typisk stjerneskjema [KIMB96]

Sentralt i ROLAP står multidimensjonal modellering. I forbindelse med relasjonsdatabaser innebærer denne teknikken at informasjonen deles inn i to forskjellige datastrukturer. Beregninger og numeriske data (som f.eks. inntekt og kostnader) lagres i «faktatabeller», mens dimensjoner som tid, produkt, salgssted o.l. plasseres i egne «dimensjontabeller»[1]. Dimensjontabellene kobles så til faktatabellen slik illustrert i Figur 2.14 og danner et såkalt «stjerneskjema». I kapittel 4.2 utdypes denne formen for skjema ytterligere. Stjerneskjemaet er velegnet for å kunne søke i store datamengder, og gjør det enklere for ROLAP-verktøyene å presentere informasjonen til brukerne. De forskjellige dimensjonene skilles enkelt ut fra datamengden og verktøyene presenterer mulighetene godt strukturert og forståelig for en leder uten noen spesiell utdanning innen data.

[1] For mer om dimensjoner, se kapittel 2.4.2.

Verktøyene gjør det også mulig for brukeren å velge hvilke dimensjoner han/hun vil se på til enhver tid. Etter å ha valgt et sett med dimensjoner som er av interesse er det mulig å gjennomføre «drill-down» og «drill-up». Dette er to begreper som er så vanlige i OLAP-sammenheng at de fortjener en forklaring:

- Drill-down går på det å vise flere detaljer, og benyttes for det meste for å finne forklaringen på mistenkelige data (data som skiller seg ut). Brukeren kan gjøre dette ved noen ganger å bevege seg ned i hierarkiet og andre ganger ved å legge en ekstra rad/attributt til dimensjonen. Et eksempel på å bevege seg ned i hierarkiet kan være som følger:

  > En bruker sitter og ser på tall for den totale produksjonen av enheter ved fabrikken. Han/hun legger merke til at tallene for forrige måned er uvanlig lave, og vil finne ut hvilken maskin dette gjelder. Ved å gå ett nivå lenger ned i hierarkiet presenteres hvor mye hver maskin har produsert, og brukeren ser at maskin nummer 5 har et uvanlig lite tall. Det er nå mulig for brukeren å undersøke direkte med de som har ansvaret for denne maskinen om hva årsaken til produksjonssvikten var.

  Et lignende eksempel hvor man legger til en ekstra rad/attributt kan være:

  > Brukeren ser på en oversikt over salg per produkt for en vilkårlig måned. Han/hun ønsker å se på hvilken kvalitet det selges mest av (hvert produkt selges med forskjellig kvalitet). Ved å legge til attributtet kvalitet vises en oversikt over salg den aktuelle måneden per produkt og kvalitet.

- Drill-up blir mye av det samme, men man beveger seg den andre veien. Dvs. man beveger seg ett nivå opp i hierarkiet eller man fjerner en rad fra rapporten.

Drill-down benyttes vanligvis for å fokusere på tall som vekker interesse, mens drill-up benyttes for å få oversikt over tallene. Disse funksjonene er så mye brukt at en del verktøy har dedikert egne knapper til de. Det varierer litt hvilken funksjon disse knappene utfører, om de hopper et nivå opp/ned eller om de fjerner/legger til en rad. Uansett gir alle de bedre verktøyene muligheten til å utføre begge deler på en eller annen måte.

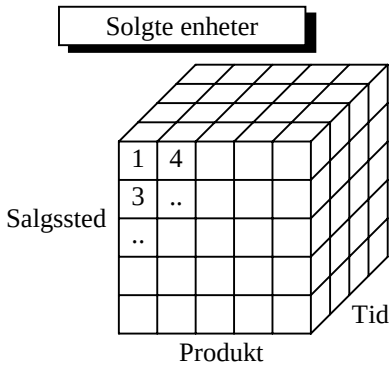
For de som ønsker å vite mer om ROLAP henviser jeg til [ABER95], [KIMB96], [RADE95] og [SAYL95].

2.4.2 Hva er MOLAP (MDD)?

MOLAP står for «Multidimensional On-Line Analytical Processing». Den største forskjellen mellom ROLAP og MOLAP er at førstnevnte jobber mot en relasjonsdatabase, mens den andre benytter en multidimensjonal database (MDD). MOLAP-verktøyet og databaseverktøyet er faktisk så nært knyttet at de kan sees på som ett og samme program. Det er derfor nyttig å vite litt om hva en multidimensjonal database er og hva den er konstruert for:

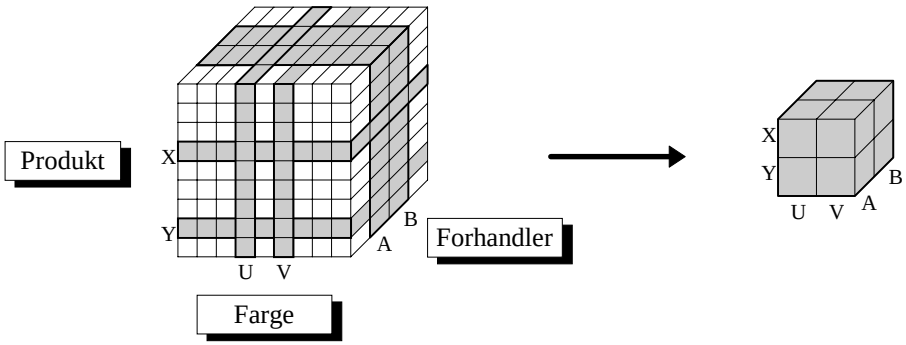
> «En multidimensjonal database er konstruert for å kunne lagre og aksessere data på en effektiv og enkel måte, der dataene er tett koblet til hverandre og hvor de skal lagres, fremstilles og bli analysert fra forskjellige synsvinkler. Disse synsvinklene kalles dimensjoner.» (Oversatt fra [KENA95])

Slike databaser er godt egnet til å støtte OLAP. Dataene blir lagret i kuber hvor sidene i en kube representerer dimensjonene. Det er viktig å merke seg at kubene kan ha mer enn 3 kanter (dimensjoner), selv om dette er vanskelig å visualisere. En kube med tre dimensjoner er illustrert i Figur 2.15.



Figur 2.15 Eksempel på en multidimensjonal kube med tre dimensjoner

Verdier lagres i krysningene mellom dimensjonene. I figuren er det antall solgte enheter av forskjellige produkter fordelt på salgssted og tid. Sluttbrukeren får med de fleste verktøyene presentert dataene i et avansert regnearkmiljø, der han/hun selv kan velge hvilke dimensjoner som skal vises til enhver tid. Brukeren har også muligheten til å gjennomføre såkalte «slice & dice»-operasjoner hvor han/hun begrenser datamengden ved å peke og klikke med musen. Dette er illustrert i Figur 2.16, hvor man har et datasett med dimensjonene produkt, farge og forhandler. Ut ifra dette velger man f.eks. to enheter fra hver dimensjon og setter de sammen til en ny og mer oversiktlig kube. Denne måten å håndtere data på blir sett på som meget enkel og er godt likt blant brukerne [NEWS96].



Figur 2.16 Begrensning av datasettet ved «slice & dice»-operasjoner

Presentasjonen av data på denne måten medfører også en annen egenskap som mange setter pris på. Flere verktøy gir brukeren muligheten til å rotere dimensjonene i 90° av gangen, slik at han/hun lett kan variere hvilke data som det fokuseres på.

Ved bruk av MOLAP-verktøy overføres dataene enten direkte fra OLTP-systemene til den multidimensjonale databasen eller via et datavarehus. Der lagres det som regel kun beregnede data, dersom det er behov for å gjennomføre drill-down ned til atomisk nivå må man kjøre spørringer mot datavarehuset, noe ikke alle MOLAP-verktøy støtter. Dersom man ikke har datavarehuset må man stole på at detaljgraden i den multidimensjonale databasen er god nok.

Siden de multidimensjonale databasene er forskjellige fra hverandre eksisterer det ikke noe standardspråk tilsvarende SQL[1] per i dag. Dette innebærer at det er vanskelig å sette sammen verktøy fra forskjellige leverandører til et system som passer bedriftens behov. Man må i stedet holde seg til én leverandør med de fordeler og ulemper det innebærer.

Det er ikke alltid tilfelle at en multidimensjonal database egner seg for å lagre data. Noen ganger vil en svak kobling mellom dimensjonene føre til problemer. Et eksempel på dette kan være en kube hvor den ene dimensjonen er kunder, den andre dimensjonen er produkter og den tredje dimensjonen er tid. Anta at man har 1000 kunder og 500 forskjellige produkter, og at salget blir registrert per måned. Problemet oppstår når hver kunde i snitt kjøper bare ett produkt hvert år. Dette innebærer at mange av krysningene mellom dimensjonene (cellene) ikke får noen verdi. Likevel må de lagres i databasen og vil ta opp lagringsplass. Dersom kuben inneholder informasjon for et år (12 måneder) vil dette innebære at:

1000 x 500 x 12 = 6.000.000 celler

må lagres, der bare

100 x 1 = 100 celler

vil inneholde verdier forskjellig fra null. I situasjoner som denne sier man at man har «spredte data». Foruten problemet med lagerplass er dette også et problem når programmet skal presentere data for brukeren, i og med at så få av cellene inneholder virkelige verdier. Det eksisterer forskjellige måter å bekjempe dette problemet på, bl.a. splitter noen programmer kuben opp i flere mindre kuber hvor dataene ligger tettere (multicubes). Det er også mulig og la være å lagre celler uten verdi, men dette vil innebære kostnader på prosessorsiden.

2.4.3 Hva er best?

Hvilken strategi som vil lønne seg for en bedrift avhenger helt av situasjonen. Det finnes tilhengere av begge metodene som påstår at deres løsning er overlegen den andre, men det er ofte bare påstand mot påstand. Før man bestemmer seg for hva som er best kan det være nyttig med en oversikt over de fordeler og ulemper som følger med hver strategi:

Fordeler med ROLAP:

- Kommuniserer enkelt med relasjonsdatabaser via SQL. Dette gjør verktøyene godt egnet til å utnytte datavarehus
- Datamengden begrenses av relasjonsdatabasen i bunnen, som kan støtte opp til flere terrabyte data (avhengig av hvilken database som benyttes)

[1] SQL, Structured (English) Query Language. Standardspråk for manipulering av data i relasjonsdatabaser.

- Forandringer i datastrukturen kan utføres uten å måtte kjøre nye beregninger på dataene i databasen
- Gir gode muligheter til å spre arbeidet mellom klienter, tjenere og eventuelle mellomliggende lag for å bedre ytelsen

Ulemper med ROLAP:

- Indeksering, som er nødvendig for å gjennomføre raske søk, krever mye lagerplass
- Støtter bare leseoperasjoner til databasen. Dette p.g.a. den kompliserte konstruksjonen av relasjonsdatabasen, der denormalisering og redundans vanskeliggjør oppdateringer av enkeltposter v.hj.a. sluttbrukerverktøy.
- Det tar gjerne flere måneder å konstruere og innføre et system i en bedrift
- Har ingen innebygde beregningsfunksjoner i databasen, alt må gjennomføres etter spesifikasjoner

Fordeler med MOLAP:

- Gir tilfredsstillende responstid ved *alle* ad hoc spørringer
- Støtter både skrive- og leseoperasjoner til den multidimensjonale databasen (bedre mulighet til å teste ut «hva om»-situasjoner)
- Har mange avanserte beregningsfunksjoner innebygd
- Tidsdimensjonen er ofte direkte støttet i verktøyet
- Lett å konstruere databasen, et system kan være oppe i løpet av dager/uker
- Lite lagringsplass kreves til indeksering
- Data lagres på samme måte som de blir vist til brukeren

Ulemper med MOLAP:

- Alle programmene er unike og kan ikke kommunisere sammen med noe felles standardspråk som SQL
- Mange verktøy greier ikke å håndtere datamengder større enn 50GB, noen har enda lavere grenser (10-20GB)
- Forandringer i datastrukturen fører til at alle data må beregnes på nytt fra bunnen av
- Er ikke egnet når man har mer enn 10-20 dimensjoner (avhengig av verktøy)
- Situasjoner som gir spredte data krever spesiell behandling, noe ikke alle verktøy takler like godt
- Kan ikke uten videre foreta drill-down ned til atomisk nivå siden databasen inneholder beregnede data

På begge områdene er det store forskjeller på hvilke verktøy som støtter hvilke funksjoner, og det er bare de aller dyreste som kan sies å ha alle fordelene som er nevnt.

Etter hvert som verktøyene videreutvikles blir forskjellen mellom ROLAP og MOLAP mindre. Utviklingen går så raskt at allerede nå er mange av de ulempene som er nevnt ovenfor løst. Det eksisterer i dag multidimensjonale databaser som støtter 100GB data og andre som kan utnytte data i relasjonsdatabaser fullt ut. ROLAP-verktøy får etter hvert et grensesnitt som ligner på MOLAP-verktøyenes, og flere avanserte funksjoner bygges inn som standard. Samtidig fører bedre optimalisering av spørringene til at responstiden holdes nede.

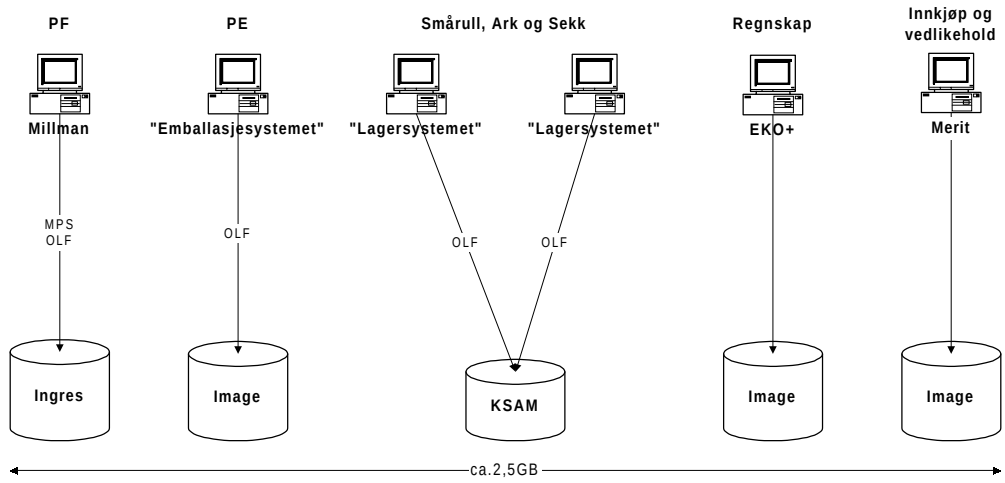
Tidligere var det gjerne slik at dersom datamengden var stor (større enn 50GB) eller dersom dataene skulle gå via et datavarehus så var ROLAP å foretrekke. I dag er ikke dette like sikkert, nye verktøy gjør at MOLAP er aktuelt også i disse tilfellene. Mange av verktøyene er for dyre for middels store norske bedrifter, men det kommer etter hvert verktøy som er billige og som kan utnytte data fra et datavarehus bedre. Det er derfor ingenting i veien for å begynne å tenke på datavarehus som et utgangspunkt. Hvilken verktøyløsning man til slutt bestemmer seg for er likevel sterkt avhengig av problemet.

I [RADE95], [KENA95] og [FINK95] står det mer om multidimensjonale databaser og MOLAP, samt deres forhold til ROLAP.

3. Peterson Ranheims Problem

3.1 Dagens situasjon

Peterson Ranheim AS er en av tolv bedrifter i det norske Peterson-konsernet. Konsernet er fokusert rundt fiberbasert emballasjeproduksjon, og bedriften på Ranheim – like nord for Trondheim – produserer både papir, massivpapp, bølgepapp, emballasje (esker), sekker, smårull og ark. Omsetningen ligger på rundt 500 millioner kroner og antall ansatte er rundt 400. De to største divisjonene i bedriften er papirfabrikken (PF) og pappemballasje (PE). I Figur 3.1 illustreres de viktigste datasystemene knyttet opp mot de forskjellige divisjonene.



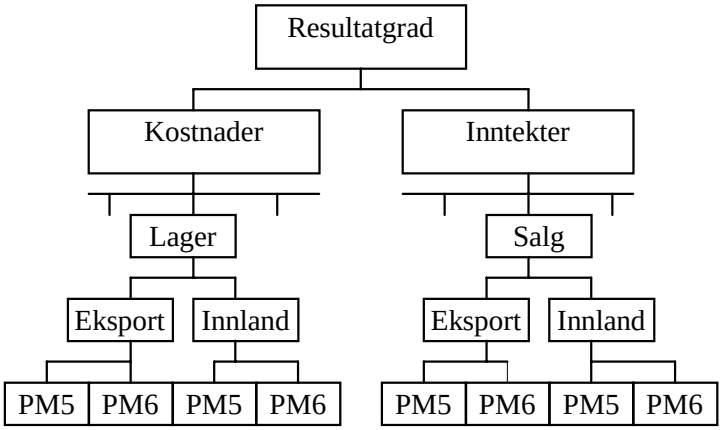
Figur 3.1 Oversikt over databasene og avdelingene hos Peterson Ranheim AS

Det er i hovedsak papirfabrikkens ønsker og behov som er av interesse i denne rapporten. Papirfabrikkens hovedsystem kalles Millman, og dette systemet tar seg av ordrer, lager og fakturering (OLF) samt materiell- og produksjonsstyring (MPS) for divisjonen. I bunnen av systemet benyttes en relasjonsdatabase (Ingres) som for øyeblikket er på ca.500MB.

Peterson Ranheim er representativ for en middels stor norsk industribedrift. Sammenlignet med annen prosessindustri er den i underkant av gjennomsnittet, men oppbygging og bruk av datateknologi er typisk, og problemstillingene i bedriften går igjen i mange andre både større og mindre bedrifter.

Den grunnleggende ledelsesfilosofien i Peterson Ranheim er målstyring. Et av de viktigste resultatene av bedriftens strategiarbeid er sentrale måltall for den neste strategiperioden. Disse måltallene brytes ned til underliggende måltall for hvert område så langt det er hensiktsmessig. I likhet med at budsjettet er et viktig redskap i den økonomiske styringen er disse måltallene viktige verktøy i den daglige oppfølgingen av driften.

Det mest sentrale måltallet er for kapitalrentabilitet. Dette forutsetter igjen en viss resultatgrad som beregnes ut i fra kostnader og inntekter. En forutsetning for å oppnå ønsket resultat er altså at man når målene for inntekter på den ene siden og kostnadskontroll på den andre siden. Inntekter og kostnader brytes så videre ned som vist i Figur 3.2. Det fokuseres i figuren på salgsinntektene og lagerkostnadene.



Figur 3.2 Nedbryting av sentrale måltall (PM står for papirmaskin)

Dette er en topp-ned fremgangsmåte for å finne måltallene. I tillegg har bedriften en ordning med årsavtaler som gjør det mulig å generere måltall ved en bunn-opp tilnærming. Hvert år, som oftest i forbindelse med budsjettbehandlingen, blir det forhandlet frem årsavtaler med de store kundene. Disse avtalene regulerer prisen basert på et forventet uttak (kjøp) i løpet av kommende år. Prisene vil variere med prosentvise økninger/reduksjoner når markedet tilsier det, men begge parter sikrer seg på dette viset en relativ pris i forhold til konkurrentene. Årsavtalene kan også inneholde en rekke andre betingelser, f.eks. knyttet til leveringssikkerhet. Dette kan påvirke størrelsen på det beredskapslageret man må holde for kunden og den geografiske avstanden mellom lageret og kunden. Etter at årsavtalene er på plass kan man sette opp måltall for salg og gjennomsnittlig lagerstørrelse per kunde. Disse kan så akkumuleres opp til høyere nivå og gi måltall per kundegruppe, land osv.

De første edb-baserte verktøyene til måltallsoppfølging var regnskapssystemene. Disse holder orden på inntekter og utgifter, og kan gi periodevise redegjøringer for hvordan man ligger an i forhold til budsjettene. På Ranheim gir regnskapet en oversikt over salg brutt ned på innland/eksport og PM5/PM6.

Siden 70-tallet har Peterson Ranheim benyttet edb-baserte transaksjonssystemer for ordre, lager, skipning og fakturering, og dagens system, Millman, har vært i bruk siden 1990. Det representerte da et stort skritt fremover med tanke på stadig større ytelse og fleksibilitet for måltallsoppfølging på detaljnivå.

I dag inneholder Millman godt over 100 standardrapporter som kan skrives til skjerm og/eller på papir. Hver bruker har etter behov rettigheter til å kjøre et utvalg av disse til det tidspunkt han/hun selv ønsker. Rapportene startes fra egne skjermbilder der man i de fleste tilfellene kan oppgi forskjellige parametre for aktuell tidsperiode, kunde, ordre osv. De aller fleste av rapportene knytter seg til behov i den daglige driften, men mange egner seg også til å hente ut måltall.

3.2 Problemer hos Peterson Ranheim

Det fokuseres her på de problemene som eksisterer i forbindelse med papirfabrikken og rapporteringssystemet, siden bedriften er ivrig etter å løse disse. Det er også her man i første omgang ser for seg at et datavarehus kan være nyttig.

Det har vist seg å være flere problemer knyttet til rapporteringssystemet, men de viktigste kan oppsummeres i følgende fem punkter:

- Det første problemet man ble klar over var et merkbart ytelsestap i transaksjonssystemet under kjøring av rapporter. Dette har man midlertidig forsøkt å løse med investeringer i stadig kraftigere maskinvare, og ved at en rekke rapporter kjøres i «batch» (alene) om natten. Disse ligger da klar på nettverkstilknyttede skrivere om morgenen. Man har også jobbet med «tuning» (fortrinnsvis optimalisering av indekser og spørringer), men ytelsesproblemet ser alltid ut til å komme igjen som følge av stadig økt rapportforbruk, nye krav til funksjonalitet og økte datamengder.

- Et annet problem er selve mengden av rapporter. Ett hundre rapporter er kanskje ikke mye, men siden brukerne kan bestemme en rekke parametre til de fleste rapportene er antall varianter en størrelsesorden høyere. Det kan ikke forventes at en person skal ha oversikten over alle disse. I de verste tilfellene har man til og med utviklet to rapporter til samme formål, og først oppdaget dupliseringen senere. Resultatet er at brukerne benytter en «prøv og feil»-metode helt til de finner riktig rapport, eller at de ikke benytter de rapporteringsmulighetene som finnes fordi de ikke husker fremgangsmåten.

- Et tredje problem knytter seg til flerdimensjonaliteten i datamengden. IT-avdelingen blir relativt ofte forespurt om å endre rapporter på detaljnivå. Dette gjelder ofte den måten data er gruppert på. Rapportene kan være for detaljerte og summere på for mange nivå slik at man mister oversikten, eller de kan mangle grupperinger som er nødvendig for å finne de ønskede data. De vanskeligste endringene skjer når man ønsker å «vri» på dimensjonene. Et eksempel kan bidra til å illustrere dette (se Figur 3.3 og vedlegg A):

  > En salgsrapport summerer salg i kroner med gruppering på PM5/PM6 på øverste nivå, land på neste og kunde som siste nivå. Man får på denne måten ut bl.a. en oversikt over summen av totalt salg for PM5 (papirmaskin 5), og også totalt salg til hvert land for PM5. Problemet oppstår når man vil vite hva det totale salget til en gitt kunde er både for PM5 og PM6. Dette må da løses ved en manuell summering av tallene, siden denne informasjonen ikke står direkte i rapporten.

  Inntil nylig (før Datatorget, se kapittel 3.3) har man ikke hatt noe annet å tilby enn nye rapporter for å løse slike problemer, rapporter som IT-avdelingen må lage. Ad hoc spørringer kommer i samme kategori, og har hittil blitt løst ved at IT-avdelingen har hjulpet til. Dette begrenser hvor mange ad hoc spørringer som kan gjennomføres, og fører til at brukerne får mindre utbytte av de data som er tilgjengelig.

SALGSRAPPORT
Periode 12.03 1996 - 14.04 1996

PM5

Norge:	Kunde	kg	Kr/tonn
	Centa	45633	3262
	...	...	...
Sum Norge (PM5): ...			
Sverige:	Kunde	kg	Kr/tonn
	Nycos	75988	3149
	...	...	...
Sum Sverige (PM5): ...			
Tyskland:	Kunde	kg	Kr/tonn
	...	...	...
	...		
Sum Tyskland (PM5): ...			
	...		
	Sum totalt PM5: ...		

PM6

Norge:	Kunde	kg	Kr/tonn
	...	...	...
	...		
Sum Norge (PM6): ...			
Sverige:	Kunde	kg	Kr/tonn
	...	...	...
	...		
Sum Sverige (PM6): ...			
...			
	Sum totalt PM6: ...		

Figur 3.3 Eksempel på strukturen i en rapport fra Millman-systemet

- Man har også et problem med sammenstillingen av data fra ulike systemer. I likhet med mange andre bedrifter ønsker Peterson Ranheim på sikt ledelsesinformasjons-systemer der man kan foreta drill down og detaljanalyse av måltallene. Men dette forutsetter en sammenstilling av data fra ulike systemer, samt en viss bearbeiding av disse – også i form av kommentarer. Et mer akutt problem er at ulike divisjoner opererer med budsjettall i egne systemer, uavhengig av Millman. Dette kompliserer fremstillingen av salg i forhold til budsjett ganske mye. Generelt gjelder dette alle måltallene, da transaksjonssystemet ikke er konstruert med tanke på slike sammenligninger. En mulig løsning på dette problemet er legge inn støtte for det i transaksjonssystemet, noe man har prøvd for salgsbudsjett. Dessverre har det vært vanskelig å få til den nødvendige fleksibiliteten med dette systemet, siden det i utgangspunktet er et standardsystem tilpasset bedriftens behov av en ekstern leverandør. Det generelle problemet er enda vanskeligere å håndtere, da de aktuelle målparametrene kan endre seg ved strategiarbeidet som skjer årlig.

- Det siste hovedproblemet er knyttet til tidsaspektet. Millman holder rede på varelageret til enhver tid. Dette innebærer at hver eneste produserte pakke[1] er registrert med en rekke data, blant annet produksjonstidspunkt. I tillegg registreres det når en pakke skipes (sendes) og faktureres. Teoretisk er det derfor mulig å beregne hvilke pakker som var på lager ved et bestemt tidspunkt ved å sjekke at produksjonstidspunktet er mindre enn det gitte tidspunktet samtidig som at skipningstidspunktet er større:

  produksjonstidspunkt < lagertidspunkt < skipningstidspunkt

  I praksis oppstår imidlertid en hel del avvik som gjør at dette ikke vil virke tilfredsstillende. Returer fra kunder gir varer på lager selv om pakkene er skipet, ompakking av varer kan gi ny produksjonsdato, ruller kan settes på vent mens det avgjøres om de skal godkjennes som kurant vare, osv. Dette gjør at man nå skriver ut rapporter over lagerstatus med jevne mellomrom. Muligheten for en fleksibel og løpende analyse av lagerutviklingen er dessverre ikke mulig, noe man kan se av følgende eksempel:

  > En dag oppdager ledelsen at måltallet for lager av et visst produkt er overskredet. Det vil si, størrelsen på lageret av produktet er for stort. Dermed setter ledelsen fokus på lagersituasjonen for å finne ut hva som har skjedd eller er i ferd med å skje, siden lagerkostnadene har en betydelig innvirkning på bedriftens resultat.
  >
  > Det viser seg at overskridelsen gjelder PM 5, og at forholdet gjelder et lager i utlandet. Denne informasjonen kan man skaffe til veie med dagens system uten alt for store problemer. Men når ledelsen ønsker å finne ut hvilket land og hvilken kunde det gjelder må IT-avdelingen kobles inn. Lagerdataene må plukkes ut fra Millman og legges inn i Excel før man kan gjennomføre en analyse som vil gi svaret på disse spørsmålene. Det viser seg å være en kunde i Tyskland som ikke har gjort det antall avrop (avhentninger av produserte produkter) som var forventet på forhånd.
  >
  > Ledelsen ønsker nå en samtale med denne kunden, og i den forbindelse hadde det vært ønskelig med en del historiske data. Dette kan være data som f.eks. avrop fra kunden hittil dette året og tilsvarende for samme periode forrige år. Dessverre er det ikke mulig å skaffe til veie denne informasjonen med dagens system, i alle fall ikke uten bruk av uforholdsmessig mye ressurser. Dermed må ledelsen gå til møtet dårligere rustet enn hva de kunne ha gjort med et bedre system.

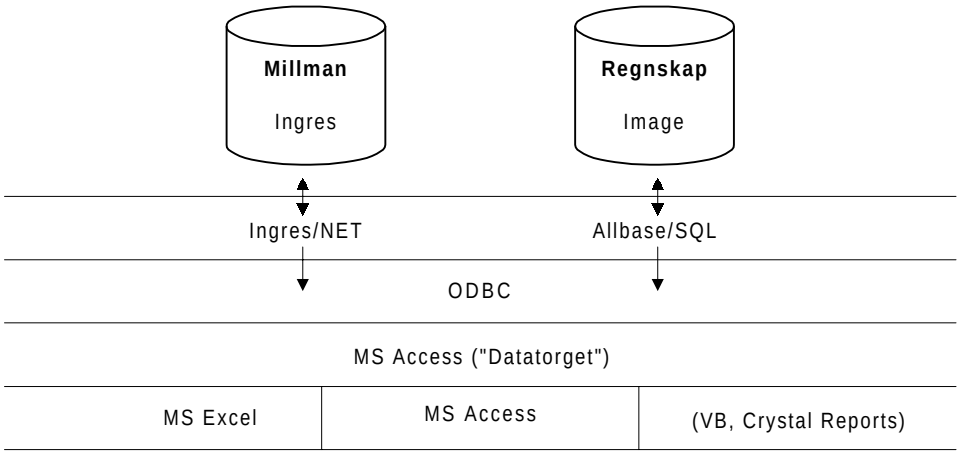
  Eksemplet viser én situasjon hvor et datavarehus hadde vært nyttig, flere eksisterer. Daglig dukker det opp situasjoner hvor et datavarehus kan være til hjelp for ledelsen.

[1] Produserte enheter er enten ruller eller paller, og som generelle betegnelse på disse benyttes «pakke».

3.3 Datatorget

Utviklingen av løsninger for å komme de nevnte problemene til livs har foregått med tiltakende styrke de siste fem årene. Det første skrittet bedriften gjennomførte var å kjøre perioderapporter som ble overført til PC og videre behandlet med regneark. Dette var først og fremst for å tilfredsstille regnskapsavdelingens behov. Overføringen av data har gradvis blitt mer strukturert, og nylig er alle overføringer samlet i et konsept som kalles «Datatorget».

Datatorget er en MS Access database som inneholder data fra Millman og regnskap, se Figur 3.4. Overføringen av data er ikke helt enkel, siden det er forskjellige databasesystemer skal aksesseres. Til å plukke ut dataene fra Ingres-databasen til Millman benyttes programmet «Ingres/NET», hvorpå man gjennom ODBC[1] legger dataene inn i Access-databasen. Litt mer komplisert er det med regnskap, selv om man også her greier seg med ett program for å plukke ut dataene. Grunnen til dette er at Image-databasen, som er databasen til regnskap, ikke er en relasjonsdatabase. Dermed kan ikke SQL benyttes direkte, men man må gå via programmet «Allbase/SQL» for å hente dataene. Deretter følges samme metode som for Millman for å legge dataene inn i Access-databasen (ODBC).



Figur 3.4 Oversikt over Datatorget hos Peterson Ranheim AS

Det knytter seg en viss usikkerhet til bruken av MS Access som databasesystem for Datatorget. Foreløpig har det ikke dukket opp noen problemer, men det kan raskt bli tilfelle dersom Datatorget vokser seg stort og antallet samtidige brukere øker. Derfor ser IT-avdelingen på MS SQL Server som en mulig erstatning, og håper at denne også skal kunne benyttes som en grunnstein i et eventuelt datavarehus. Dersom bedriften ønsker å utvide Datatorget til et mer fullstendig datavarehus er dette et nødvendig skritt å ta. MS Access er ikke kraftig nok til å kunne håndtere de datamengdene og den bruken som et datavarehus vil innebære. Om en kraftigere relasjonsdatabase er løsningen er derimot ikke så sikkert, det finnes også alternative multidimensjonale databaser.

[1] Open DataBase Connection, standard for overføring av data mellom forskjellige databaser.

Som sluttbrukerverktøy benytter bedriften for øyeblikket mest regnearket MS Excel, da dette er et verktøy som allerede er kjøpt inn og ikke innebærer ekstra kostnader. En annen fordel er at brukerne er kjent med programmet fra før av og har sjelden behov for ekstra opplæring. For å tilpasse MS Excel til Datatorget ytterligere vurderer de å benytte MS Visual Basic[1],et utviklingsprogram som er tett knyttet til Excel. I tillegg til MS Excel benyttes også MS Access som et sluttbrukerverktøy, og de har laget flere makroer her som letter forståelsen for brukerne. Selv om de nå vurderer å gå over til MS SQL Server kommer de til å beholde MS Access som et støtteverktøy for sluttbrukerne. For å gi brukerne bedre muligheter til å lage gode rapporter har de også anskaffet en lisens av Crystal Reports, et rapporterings- og utviklingsverktøy (se kapittel 4.4). Dersom dette viser seg å svare til forventningene kommer de sannsynlig vis til å kjøpe inn flere lisenser og innføre det hos noen brukere.

Det kan være interessant å se på hvilke kriterier Datatorget oppfyller med tanke på at det skal kunne videreføres til et datavarehus:

- Data overføres til Datatorget hver natt.
- Allerede nå foregår det en integrering av data fra Millman og regnskap. Samtidig forenkles navngivningen av entitetene slik at de gir mening til sluttbrukerne.
- Atomiske data på pakkenivå blir ikke lagret, men dataene blir tatt vare på med en tilstrekkelig detaljgrad til at de kan analyseres tilfredsstillende.
- Salgsdata for inneværende år genereres i sin helhet hver natt og erstatter fullstendig dataene som ligger i Datatorget. Dette gjøres for å få med eventuelle korrigeringer av fakturaer gjennomført i transaksjonssystemet. I et eventuelt datavarehus må man prøve å finne en bedre løsning på dette, da man neppe kan ta seg tid til å generere disse dataene hver gang.
- Lagerdata legges til eldre data og gjør at man kan se status for lagret tilbake i tid.
- Noen summering av dataene gjøres ikke, den begrensede datamengden har gjort at responstiden likevel har vært akseptabel.
- Metadata benyttes ikke i noen større grad, og liten innsats har blitt gjort på dette området.
- Backup blir gjort hver natt sammen med resten av filtjeneren. Dette er ikke noe problem, da Datatorget foreløpig bare er en ca. 10MB stor Access-fil på tjeneren.

Som man ser har Datatorget mange trekk fra datavarehus, og muligheten til å utvikle dette konseptet videre til et datavarehus må sies å være til stede.

[1] MS Visual Basic er et visuelt utviklingsprogram som hjelper brukeren å programmere i språket Basic.

3.4 Kostnadsramme

Innføringen av et datavarehus i en bedrift er knyttet til kostnader på lik linje med de fleste utviklingsprosjekter. Etter som dette området er veldig nytt eksisterer det for øyeblikket flere «mindre» bedrifter som påstår at de kan levere komplette systemer som dekker de fleste behov ved innføringen av datavarehuset. Felles for disse systemene er at de har en pris som ligger langt over det en norsk mellomstor bedrift har mulighet til å etterkomme. Etter hvert har det kommet systemer i andre prisklasser som kan sies å være «hyllevare», tilsvarende programmer som MS Office (Word, Excel, PowerPoint), Lotus Notes o.l. Disse systemene har klare mangler sett i forhold til de dyrere systemene, men her må man foreta en avveining mellom pris og ytelse. En annen effekt av at markedet er såpass nytt er at det stadig dukker opp nye aktører med ny programvare, i tillegg til stadige oppgraderinger av den programvaren som allerede eksisterer.

Peterson Ranheim er som så mange andre mellomstore bedrifter ute etter å utnytte mest mulig av denne nye teknologien uten å måtte foreta for store investeringer. De legger vekt på ønsket om å bygge opp datavarehuset i flere skritt som går over en lengre tidsperiode. På denne måten blir det mulig å unngå de store initielle kostnadene som kan være vanskelig å forsvare. For IT-avdelingen er det lettere å bruke penger over driftsbudsjettet enn gjennom store investeringer, en problemstilling som også gjelder for mange andre bedrifter.

Selv om man prøver å begrense kostnadene må man være villig til å investere en viss sum i ny programvare o.l. For Peterson Ranheims del er det satt en grense på 200.000,- NOK. Dette beløpet er en øvre grense for hva bedriften føler er lønnsomt å bruke på et slikt prosjekt, og er ment å dekke kostnader som:

- Nødvendig programvare til datavarehuset og sluttbrukerne
- Avtaler om vedlikehold og støtte fra leverandører
- Bruk av konsulenttjenester
- Kursing av brukerne

Arbeidstimer utført av de ansatte på IT-avdelingen er ikke tatt med her, men det forutsettes at arbeidet kan gjennomføres parallelt med den daglige driften av avdelingen. Når det gjelder hvor kostnader for utvidelse av maskinparken plasseres vil dette avhenge av hvilken form for utvidelse det er snakk om. Innkjøp av ekstra disk og minne bør kunne dekkes av denne summen, mens innkjøp av en ekstra tjenermaskin vil måtte gå over et eget innkjøpsbudsjett. Det bør også nevnes at siden innføringen av datavarehuset er tenkt gjennomført over tid er ikke dette beløpet helt fastlåst, det kan tenkes at det blir forandret etter som behovene forandrer seg.

3.5 Problemstilling

Mange ledere har ønsker om informasjon som kan hjelpe dem til å styre bedriften best mulig. De vil ha muligheten til å finne ut hva som skjer på markedsfronten og hvorfor det skjer [NOLA96]. En liste over ønsker og behov som mange vil kjenne seg igjen i kan se slik ut:

Lederne vil

- vite hva som egentlig skjer med bedriften
- vite hvordan kundene «ser ut»
- vite hva kundene kjøper
- selge kundene det de kan være interessert i å kjøpe
- ha muligheten til å stille nye spørsmål som har dukket opp og som kan ha påvirkning på bedriften
- ha muligheten til å sette investeringsmulighetene opp mot hverandre og velge de som vil gi best resultat for bedriften
- ha en kilde til nye ideer, ideer som kan testes ut raskt og enkelt. Dersom testingen er positiv vil de ha muligheten til å implementere ideene med et minimum forbruk av tid og penger
- vite hvor pengene blir brukt
- vite om pengene blir brukt effektivt

Det de ikke vil ha er

- «pyntede» rapporter over hva andre ønsker skjer med bedriften
- fem forskjellige rapporter fra fem forskjellige avdelinger, der ingen er i overensstemmelse med hverandre
- at det tar 12 måneder fra de gjør noen forandringer til man finner ut om de virker som planlagt
- at nye ideer blir holdt igjen av analyser og prioriteringer slik at implementeringen av ideen tar 12-24 måneder

Selv om denne listen ikke dekker alle ønsker en leder kan ha, og kanskje også har noen som ikke er så viktige, så illustrerer den noe av poenget bak et datavarehus. Problemet som mange bedrifter i Norge står ovenfor er at størrelsen på investeringene til et slikt datavarehus ofte ikke står i samsvar med de ressurser bedriften har tilgjengelig. Det hjelper heller ikke at datavarehus tilsynelatende fører til små besparelser, i alle fall i begynnelsen. De store datavarehusløsningene som eksisterer i dag er i de fleste tilfellene alt for dyre for norske bedrifter. Det er derfor ønskelig å se på hvilke deler av datavarehusteknologien som kan benyttes uten at man legger så mye penger i det. Det er også interessant å se på hvilke sluttbrukerverktøy som kan tilfredsstille disse behovene samtidig som kostnadene holdes nede.

Peterson Ranheims problem er spesielt knyttet til rapporteringen, som nevnt tidligere i rapporten. De ønsker å bedre ledelsens tilgang til informasjon uten å foreta så alt for store investeringer. Det er også interessant å undersøke til hvilken grad programmer som MS Excel og mer spesialiserte verktøy som f.eks. PowerPlay[1] kan tilfredsstille disse behovene.

[1] Se kapittel 4.4

4. Vurdering av løsninger

Et viktig spørsmål som må stilles på dette stadiet er om datavarehus er den riktige løsningen på problemene til Peterson Ranheim AS. Kan det være andre løsninger som er bedre egnet? Er det i det hele tatt mulig å innføre et datavarehus uten å måtte investere store summer i konsulenttjenester, programvare og maskinvare?

Ikke alle er overbevist om at datavarehus er løsningen på alle bedrifters problem. I [COMP95] diskuterer Ståle Lian hvorvidt datavarehus bare er et moteord som vil forsvinne på lik linje med de store ledelsesinformasjonssystemene (LIS) som kom på 1980-tallet. Konklusjonen er at fremtidsutsiktene for datavarehus er bedre enn hva de var for LIS på sin tid, men det er vanskelig å være sikker på det endelige resultatet. Behovet for bedre beslutningsstøtte er definitivt tilstede, men om det endelige svaret på dette vil være datavarehuset er enda usikkert.

En av farene med den populariteten datavarehus har i dag er at mange bedrifter ikke tenker nøye nok igjennom hvilke problemer de har og hvordan de kan løses. I stedet ser de på datavarehus som en slags «vidunderkur» og setter i gang med en implementasjon på et alt for tidlig tidspunkt. En del skeptikere har trukket frem en del mislykkede datavarehus som bevis på at dette ikke er noen god løsning. Dessverre har de fleste en tendens til å generalisere og si at datavarehus ikke har noen nytte i det hele tatt. Etter min mening er dette feil, det eksisterer flere eksepler på at datavarehus er en nyttig løsning på vanskelige problemer.

Hvilke alternativer har man så på Peterson Ranheim:

- Man kan fortsette å utvide maskinkapasiteten og optimalisere Millman ytterligere for å holde ytelsen til transaksjonssystemet på et akseptabelt nivå. Dette vil på kort sikt være en akseptabel løsning på problemet med ytelsen, men det gjør ikke noe med de uoversiktlige og utilstrekkelige rapportene, sammenstiller ikke noen data fra andre systemer og forbedrer ikke situasjonen med manglende historiske data.

- Ved å erstatte Millman-systemet med et nyere og bedre system vil man kunne løse noen av problemene, men ikke alle. Det vil også være dyrere enn hva som er forsvarlig i denne omgang. Bortsett fra de problemene som er nevnt fungerer systemet bra i dag og det er få grunner utover disse til å anskaffe et nytt system.

- Det er mulig å fortsette slik man har det i dag, med et datatorg som en avlastning til Millman-systemet. Dette løser flere av problemene, men vil bare være en kortvarig løsning. Dersom antall brukere og datamengden øker vil programvaren som benyttes (MS Access) ikke kunne tilfredsstille behovene. Kun til en viss grad vil brukerne på egen hånd kunne utføre de funksjonene som er ønskelig (analyser, rapporter o.l.).

- Man kan bygge videre på det man har i dag og benytte teknikker fra datavarehus-miljøet til å forbedre situasjonen. Ved å innføre et slikt mini-datavarehus kan man benytte mange av de nyttige verktøyene som dukker opp på markedet til å tilfredsstille brukernes behov. Dette er kanskje den mest realistiske løsningen og den vil kunne løse de aller fleste av problemene. Ved å gjennomføre prosjektet skrittvis over tid unngår man også de store investeringene.

- Man kan kjøpe et større datavarehuskonsept eller et større MOLAP-verktøy fra en av de store leverandørene. Dette vil sannsynligvis løse problemene, men prisen for dette er alt for høy for en middels stor norsk bedrift. Det er kanskje noen bedrifter som har råd til dette alternativet, men nytteverdien vil neppe stå i stil til kostnadene.

- Det er også en mulighet å erstatte alle dagens systemer med ett stort system som tar seg av alt, f.eks. et SAP-system[1]. Dette alternativet vil muligens kunne løse alle problemene uten å innføre noe eget datavarehus, men det er en ekstremt kostbar løsning. Som i det forrige alternativet er dette alt for dyrt til en middels stor norsk bedrift. Dersom dette skal være aktuelt må det samtidig settes i sammenheng med at alle de andre systemene må byttes ut eller fornyes. Først da er det snakk om kostnader i den samme størrelsesordenen som et SAP-system vil innebære.

Som nevnt vil det beste alternativet være å benytte Datatorget som et utgangspunkt og videreutvikle dette til et datavarehus. I Tabell 4.1 er det gitt en oversikt over de kriteriene som ligger til grunn for dette valget.

Kriterier for valg av løsningsalternativ	
Ytelse:	Et datavarehus vil sørge for at rapporteringen (som krever så mye av Millman-systemet) blir overført til et annet system og ytelsen på Millman vil bedres.
Brukervennlighet:	Et datavarehus vil gjøre det mulig for brukerne å lage egne rapporter v.hj.a. enkle verktøy.
Antall samtidige brukere:	Ved å kjøre datavarehuset på et system som er bedre egnet enn MS Access vil flere brukere kunne utnytte datavarehuset samtidig.
Antall spørringer/rapporter som kan utføres per dag:	Den korte responstiden på spørringer sørger for at antall spørringer som kan utføres per dag øker drastisk. Dette gjør det mulig for ledelsen å «leke» med dataene på forskjellige måter.
Data fra flere kilder:	Et datavarehus gjør det mulig å samordne data fra flere forskjellige systemer.
Utvidelse:	Et datavarehus vil være enkelt å utvide med kommende bedre sluttbrukerverktøy som kommuniserer bedre med databasen.
Datatorget:	Bedriften har allerede et system som ligger godt til rette for en utvidelse til et datavarehus.
Pris:	Det vil ikke medføre for store investeringer å innføre et datavarehus på denne måten, man holder seg innenfor kostnadsrammen.

Tabell 4.1 Kriterier for valg av løsningsalternativ

[1] SAP – Systems, Applications and Products in Data Processing. Et av verdens største uavhengige programvaresystemer som ble grunnlagt i Tyskland i 1972. Står for leveranser av komplette løsninger der forskjellige moduler settes sammen til å tilfredsstille bedrifters behov.

4.1 Konstruksjon av datavarehuset

Når man først har bestemt seg for å konstruere et datavarehus er det flere ting å ta hensyn til. I [MIMN95] beskriver Pieter Mimno en del valg som må foretas ved utviklingen av datavarehuset. Sentralt i artikkelen står tre spørsmål:

- **Teknikk for dataoverføring:** skal man benytte seg av de muligheter standard relasjonsdatabaser gir (logging, speiling, rekonstruksjon) eller skal man benytte spesielle overføringsverktøy?
- **Måldatabase for datavarehuset:** skal man benytte en vanlig relasjonsdatabase som grunnlag for datavarehuset, eller skal man benytte spesialiserte databaser konstruert med datavarehus for øyet (som f.eks. Red Brick)?
- **Sluttbrukerverktøy:** skal man benytte standardiserte klient/tjener-verktøy som f.eks. MS Excel eller mer spesialiserte analyseverktøy som Power Play og Impromptu?

For Peterson Ranheims del vil kostnadsrammen automatisk gi svaret på noen av disse spørsmålene, og det samme gjelder mange andre bedrifter i Norge. Likevel kan det være lurt å ha øynene åpne for interessante alternativ som kan dukke opp etter hvert.

De spesielle overføringsverktøyene som er nevnt i det første punktet er per i dag kostbare og dermed lite aktuelle for bedrifter av Peterson Ranheims størrelse. I tillegg er mange av verktøyene konstruert for å være en del av et komplett datavarehussystem, noe som gjør de dårlig egnet til å benyttes alene.

Ved valg av måldatabase ligger mange av de samme årsakene til grunn for valget. De databasene som er spesielt designet for datavarehus koster for mye i dag. Det er også grunn til å tro at en vanlig relasjonsdatabase vil kunne tilfredsstille brukernes krav til responstider siden størrelsen på datavarehuset i en slik bedrift vil være begrenset. Peterson Ranheim benytter i dag MS Access til Datatorget, men vurderer å gå over til MS SQL Server. Dette databasesystemet har nettopp kommet i en ny utgave, og ser ut til å ha en del funksjoner som kan være nyttig i fremtiden (bl.a. støtte for kommunikasjon med WWW-tjenere[1]). Uansett vil MS Access ikke være kraftig nok til å kunne støtte et datavarehus i full drift.

Når det gjelder valg av sluttbrukerverktøy er det anledning til å være litt avventende og prøve ut noen forskjellige løsninger før man bestemmer seg. Et godt utgangspunkt kan være å utnytte det man allerede har av verktøy, for så å kjøpe inn mer spesialiserte verktøy etter hvert. Men da må man passe på at konstruksjonen er fleksibel nok til å kunne tilpasses krav fra disse.

En ting som det er viktig å være klar over når man skal konstruere et datavarehus er at det ikke kan konstrueres på samme måte som et annet informasjonssystem. Det er ikke mulig å sette seg ned og gå igjennom de forskjellige fasene med forstudie, konstruksjon, implementasjon o.l. som er en del av vanlige prosjekter. Grunnen til dette er at det ikke er mulig å kartlegge alle detaljene rundt de behov og ønsker som eksisterer før datavarehuset er på plass. Ledelsen vet bl.a. ikke på forhånd hvilke summerte data de har behov for, det er noe de oppdager ved bruk av datavarehuset. Man må derfor bare begynne med implementeringen på et tidlig tidspunkt og heller tilpasse konstruksjonen til brukernes tilbakemeldinger etter hvert som de får erfaring med systemet. Sånn sett ser det ut til at Peterson Ranheim har tatt en riktig avgjørelse når de startet arbeidet med Datatorget.

[1] WWW, World Wide Web. Se mer om dette i kapittel 4.4.

Dersom bedriften bestemmer seg for å gå videre med Datatorget og dermed lage et datavarehus i en relasjonsdatabase medfører dette en del nye avgjørelser. Den første går på om man skal konstruere databasen med et stjerneskjema eller ikke. Dette er den desidert vanligste måten å implementere datavarehus i relasjonsdatabaser på, og mangelen på gode alternativer gjør at beslutningen er relativt enkel å ta. Men når man først har bestemt seg for denne arkitekturen dukker det opp en del andre valg som krever grundig gjennomtenkning. Disse valgene går på konstruksjonen av selve stjerneskjemaet og omtales nærmere i kapittel 4.2.3.

4.2 Stjerneskjema

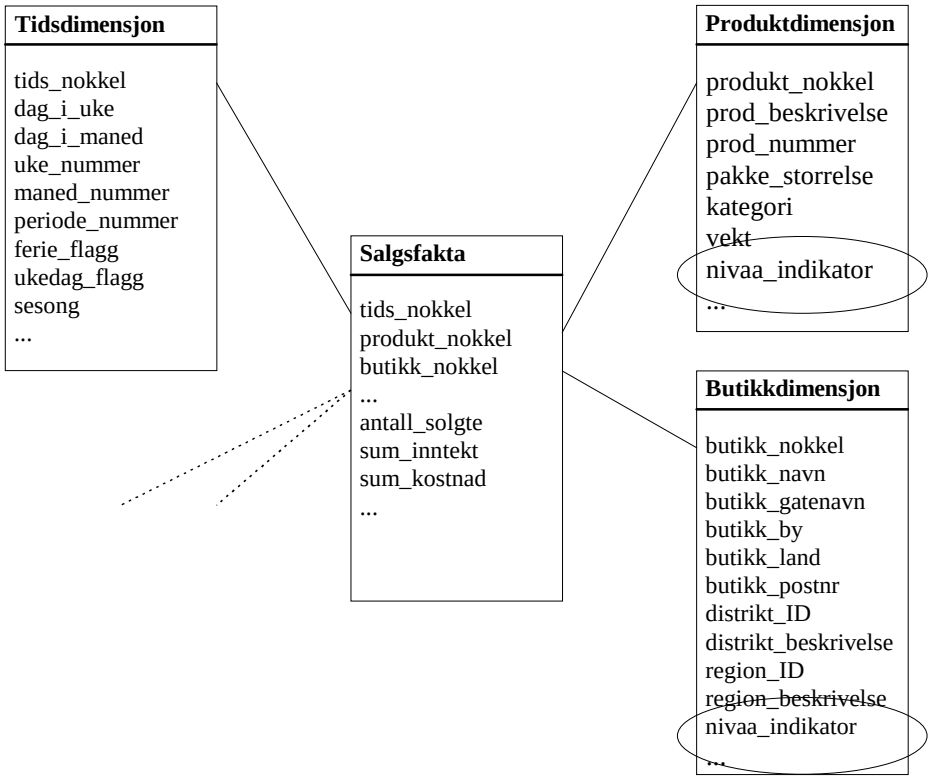
Stjerneskjemaet har fått sitt navn ut ifra den formen en modell av en slik database får på papiret. Som nevnt i kapittel 2.4.1 består et stjerneskjema av en faktatabell i sentrum og dimensjontabeller koblet til denne. Dette skjemaet kalles gjerne et «klassisk stjerneskjema», og representerer den enkleste utgaven av stjerneskjema. I tillegg eksisterer det en del videreutviklinger av denne kalt «sammensatte skjema»[1] og «snøflakskjema» [ARCH95].

Det klassiske stjerneskjemaet har flere fordeler:

- Det er lett å forstå
- Det er enkelt å definere hierarkier
- Antall fysiske sammenslåinger av tabeller som må foretas («joins») reduseres i forhold til vanlige normaliserte skjema
- Det er enkel å vedlikeholde
- Det gir enkle metadata

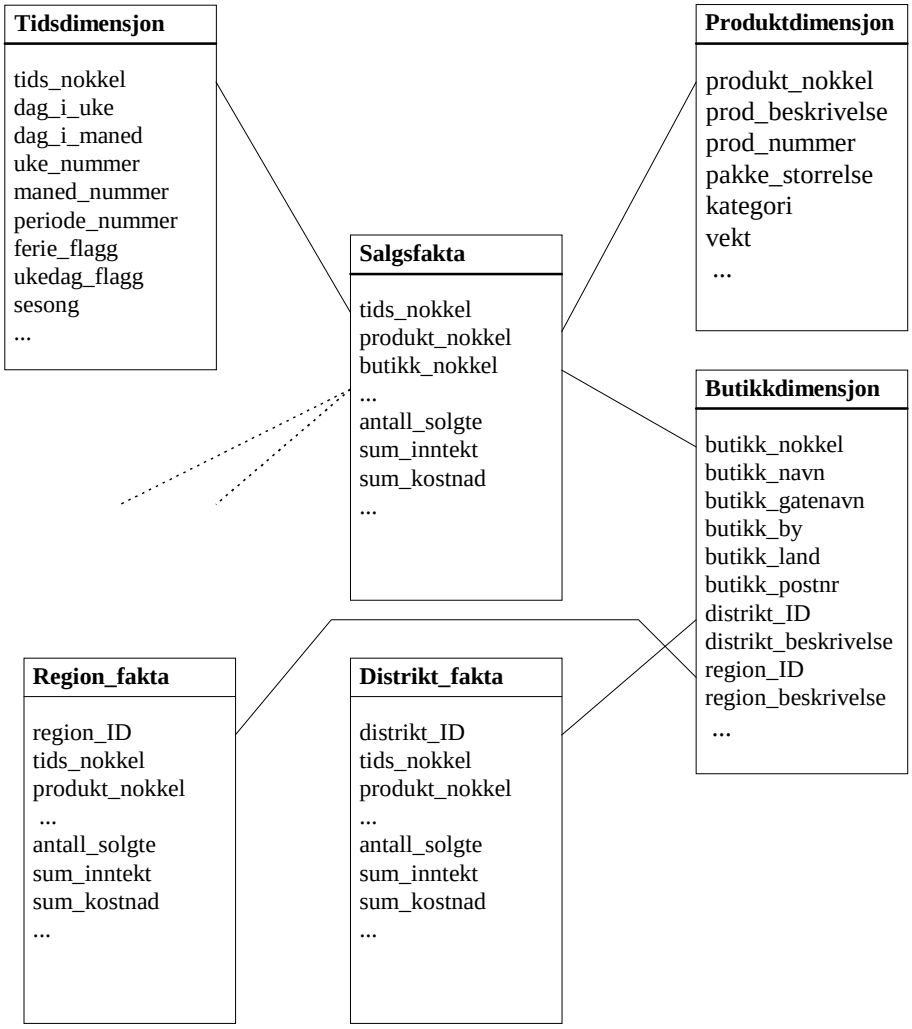
Men det eksisterer også en del ulemper med dette skjemaet. Store dimensjontabeller kan medføre at sammenslåingen av tabeller krever mye ressurser, og indekseringen av den store faktatabellen vil kreve mye lagerplass. Men den største ulempen kommer som en følge av at man må ha en nivåindikator i dimensjontabellen. Denne nivåindikatoren benyttes til å spesifisere hierarkiene og til å sørge for at oppsummerte data kan lagres sammen med de atomiske. I Figur 4.1 illustreres et slikt stjerneskjema med en nivåindikator som sier om dataene i faktatabellen tilhører en enkelt butikk, et distrikt eller en region (nivå = 1,2 eller 3). Problemet med denne indikatoren er at den begrenser fleksibiliteten til modellen og er en potensiell feilfaktor. Dersom programmet eller personen som lager spørringen glemmer nivåindikatoren kan dette føre til at resultatet består av data som ikke hører hjemme der. Derfor er dette skjemaet best egnet i sterkt kontrollerte datavarehus, helst datavarehus der bruk av faste rapporter dominerer. Likeledes vil man i mindre datavarehus av denne typen ikke merke ulempene med nivåindikatoren like sterkt.

[1] Fritt oversatt fra «Fact Constellation Schema».



Figur 4.1 Klassisk stjerneskjema med nivåindikator [KIMB96]

En mulig løsning på problemet med nivåindikatoren er å benytte et sammensatt skjema. Dette skjemaet splitter faktatabellen opp slik at hvert hierarki får hver sin faktatabell, se Figur 4.2.



Figur 4.2 Sammensatt skjema der man har egne faktatabeller for distrikt og regioner [KIMB96]

Ved å dele opp faktatabellen på denne måten oppnår man at f.eks. distrikt-faktatabellen inneholder *kun* data summert i henhold til distriktet og at hoved-faktatabellen inneholder *kun* atomiske data. Det er ingen poster med en «butikk_nokkel» som gir verdier i mer enn én faktatabell, doble tellinger blir ikke mulig og nivåindikatoren er ikke nødvendig.

Ulempen med dette skjemaet er at de summerte faktatabellene kan bli nokså kompliserte og at det ofte kreves flere SQL-setninger for å svare på spørsmål. Et eksempel på dette er når man vil ha en oversikt over det totale salget i en region prosentvis fordelt på distriktene i regionen. Da må data hentes både fra region-faktatabellen og distrikt-faktatabellen før man setter tallene sammen.

Det tredje skjematypen som ble nevnt ovenfor er snøflakskjema. Denne typen løser også problemet med nivåindikatoren, men gjennom normalisering av dimensjontabellene. Bortsett

fra at ulempen med indikatoren forsvinner medfører skjemaet få forbedringer av det klassiske skjemaet. Når så normaliseringen i tillegg fører til et større antall tabeller gjør dette at skjemaet stort sett blir brukt bare i spesielle tilfeller.

For en mer grundig innføring i de forskjellige stjerneskjemaene som eksisterer viser jeg til [ARCH95], [KIMB96] og [REDB95].

4.2.1 Sakte forandrende dimensjoner

Noen av dimensjonene i et stjerneskjema inneholder data som for det meste er konstant og sjelden forandrer seg. Det kan f.eks. være en oversikt over de ansatte ved bedriften der data som sivilstatus, adresse o.l. forandrer kun seg en gang iblant. Slike dimensjoner kalles gjerne «sakte forandrende dimensjoner» [KIMB96]. Når slike forandringer skjer er det viktig å vite hvordan man skal oppdatere datavarehuset i forhold til dette. Det er tre forskjellige måter å oppdatere datavarehuset på, og valget man gjør påvirker muligheten til å analysere de historiske dataene senere.

Den første og enkleste måten å gjøre denne oppdateringen på er å skrive over de gamle dataene i dimensjontabellen. Denne metoden forhindrer at man kan avgrense data til før og etter forandringen. Noen ganger er ikke dette så farlig, men ofte vil det være nyttig å ha muligheten til å analysere forholdene med forandringen som et skillemerke. Dersom dette er tilfelle må man oppdatere datavarehuset på en annen måte.

For å bevare de historiske dataene kan man lage en ny post i dimensjontabellen der den nye verdien er representert. Deretter benytter man det nye innslaget til fremtidige koblinger mot faktatabellen. For å kunne gjøre dette må man generere en ny nøkkelverdi til posten i dimensjontabellen. I stedet for å benytte f.eks. personnummeret som nøkkel i en ansatt-dimensjon kan man benytte personnummeret *pluss et versjonsnummer*. Antall siffer i versjonsnummeret avgjør hvor mange forskjellig «kopier» det kan være av posten. Håndteringen av denne prosessen gjøres av overføringssystemet, noe som krever at metadataene holder orden på hvilke nøkkelverdier som allerede er benyttet.

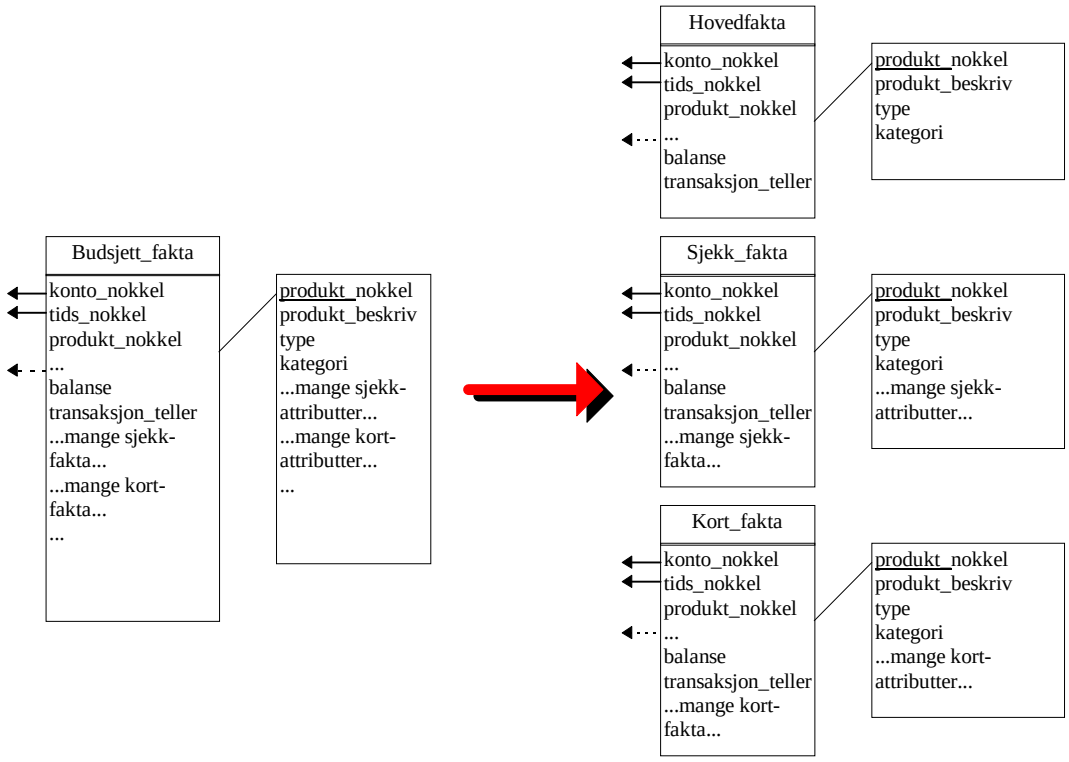
Som nevnt vil denne metoden føre til en splittelse i de historiske dataene. Dette vil gjøre det mulig å se på data før forandringen med den gamle verdien og data etterpå med den nye. Men noen ganger kan det være ønskelig med en mer fleksibel måte å analysere data før og etter forandringen. Siden metoden nevnt ovenfor splitter historien vil man f.eks. ikke ha muligheten til å benytte den nye verdien til en attributt på gamle data eller omvendt. For å få til dette må man benytte en tredje måte å registrere forandringen på. Man legger til et nytt felt i dimensjontabellen som representerer «nåværende verdi». Samtidig forandrer man feltet med den gamle verdien til et felt som representerer «gammel verdi» eller «original verdi». I tillegg kan man også legge til et felt som forteller når forandringen fant sted. Denne metoden «husker» bare den originale verdien og den nåværende verdien, alle mellomliggende verdier går tapt (alternativt huskes den *forrige* og nåværende verdien). Dersom en mer detaljert historie er nødvendig må man benytte metode nummer to. For å kunne løse alle aktuelle problemstillinger hadde det vært mulig å kombinere metode to og tre, men resultatet av en slik kombinering vil ikke være særlig godt egnet for sluttbrukerne p.g.a. den kompleksiteten det vil innebære.

Til slutt kan det nevnes at av disse metodene er det mest vanlig å benytte de to første. Den tredje metoden benyttes kun unntaksvis [KIMB96].

4.2.2 Heterogene dimensjoner

I noen tilfeller har man dimensjoner som skal beskrive et større antall heterogene enheter. Et eksempel på dette kan være en produktdimensjon som skal beskrive forskjellige typer produkter, der hver type produkt har sine distinkte attributter. Resultatet av å ha alle disse samlet blir en faktatabell og dimensjontabell med veldig mange attributter, men hvor bare et fåtall benyttes for hver post i databasen. De andre har nullverdier fordi de ikke er relevante med den typen produkt.

For å løse denne situasjonen kan man lage en faktatabell og en dimensjontabell hvor man legger kjernen av attributtene til de forskjellige produktene. Dette muliggjør søk på kryss av de forskjellige produkttypene. I tillegg lager man spesielle faktatabeller og dimensjontabeller til hver enkelt type av produktet der de spesielle attributtene legges. Dermed er det også mulig å søke i dybden på produkttypene hver for seg. I Figur 4.3 illustreres et eksempel hvor man har bl.a. sjekker og bankkort som forskjellige typer av et produkt.



Figur 4.3 Eksempel på konsekvensen av heterogene dimensjoner

4.2.3 Viktige valg relatert til stjerneskjema

I slutten av kapittel 4.1 ble det nevnt at man må ta en del viktige valg ved konstruksjonen av et stjerneskjema. Disse valgene kan oppsummeres i 9 punkter, der mange er knyttet opp til elementer diskutert ovenfor:

- Man må identifisere prosessene som foregår i bedriften (f.eks. innkjøp, salg, frakt, lagerkontroll, produksjon) og ut ifra disse lage en eller flere faktatabeller til hver valgte prosess.

- Hvilken granularitet skal det være på faktatabellene? Skal de inneholde de individuelle transaksjonene, dagens, ukens eller månedens transaksjoner?
- For hver faktatabell må det avgjøres hvilke dimensjontabeller som skal kobles til. Først må det kobles til nok dimensjoner til at en unik nøkkel for faktatabellen kan etableres, deretter kan det legges til eventuelle andre nyttige dimensjoner.
- Man må bestemme alle målbare og beregnede fakta som skal ligge i faktatabellen.
- Hvilke attributter skal dimensjontabellene ha? Beskrivelsen skal være komplett og terminologien hensiktsmessig.
- Hvilken metode skal benyttes ved forandringer i data tilhørende dimensjoner som er tilnærmet konstante (sakte forandrende dimensjoner)?
- Man må avklare en del ting vedrørende krav til fysisk lagringsplass, bl.a. aktuelle summeringer/beregninger, bruk av heterogene dimensjoner og eventuelle minidimensjoner[1].
- Hvor mye haster det med å gjøre informasjonen tilgjengelig. Må gårsdagens data være tilgjengelig i dag? Merk at dette ikke er det samme som granularitet. Det kan godt være nødvendig å kunne skille data fra hverandre på et daglig nivå, men ikke nødvendig å oppdatere datavarehuset oftere enn hver uke.
- Det kan være lurt å allerede nå tenke på hvor lenge data skal ligge i datavarehuset. Noen endelig avgjørelse trenger man ikke å ta på dette tidspunkt, men man bør vite hvorvidt det er snakk om 2 eller 10 år.

Den naturlige rekkefølgen å gjennomføre disse punktene vil være å begynne øverst og så jobbe seg nedover. Men for å kunne svare på de må man ha god kjennskap til brukernes behov og bedriftens grunnleggende prosesser. I bedrifter av Peterson Ranheims størrelse har gjerne IT-avdelingen allerede den nødvendige kunnskapen, men det kan likevel være lurt å foreta noen «intervjuer». Kommunikasjonen med brukerne er meget viktig og det er derfor bare en fordel å opprette kontakt på et tidlig tidspunkt. På den måten kan man også unngå at gapet mellom brukernes behov og systemets muligheter blir så stort at det påvirker bruken av datavarehuset. Dersom brukerne ikke benytter seg av tilbudet vil ikke IT-avdelingen få de tilbakemeldingene de trenger for å tilpasse systemet, og man er inne i en ond sirkel.

Prosessen som er forklart ovenfor er utdypet ytterligere i [INMO92] og [KIMB96].

[1] Ved veldig store dimensjoner er det noen ganger mulig å finne en del attributter som har lik verdi for flere av postene i dimensjonen. Disse attributtene kan da trekkes ut i en egen minidimensjon for å spare plass og tid ved søk.

4.3 Overføringen av data

Siden de spesielle overføringsverktøyene er dyre vil det være mest aktuelt å se på andre måter å overføre dataene på. I dag løser Peterson Ranheim hele overføringsproblemet ved at de genererer alle data til Datatorget på nytt i løpet av natten. Dette har gått bra så langt fordi datamengden fortsatt er såpass liten, men dette er et forhold som ikke kommer til å vare for evig. En av grunnene til at de velger å generere alt på nytt er for å sikre seg at alle eventuelle korrigeringer i transaksjonssystemet blir registrert. Dette er viktig i alle datavarehus, noe de alternative teknikkene også tar hensyn til.

Hvordan skal man så gå frem for å overføre data fra det ene systemet til det andre? Overføringsprosessen er gjerne den delen av konstruksjonen som skaper de største problemene og krever størst innsats. For å sikre et tilfredsstillende resultat er det en rekke faktorer man må ta hensyn til. Den naturlige rekkefølgen å gjøre ting på i henhold til [INMO92] og [KIMB96] kan oppsummeres som følger:

- Det første som må skje ved overføringen av data fra transaksjonssystemet er at rådataene fra transaksjonssystemet må leses. Dette kan være enkelt eller vanskelig avhengig av hvilke systemer som eksisterer. Relasjonsdatabaser representerer det enkle alternativet siden man da kan benytte SQL til å hente ut dataene. Proprietære systemer kan derimot by på større problemer, etter som disse ofte ikke tillater en direkte tilgang til dataene. For Peterson Ranheims del eksisterer det både relasjonsdatabaser og proprietære databaser i transaksjonssystemet, noe som kan by på problemer. Foreløpig har man løst dette med. tredjeparts programvare (Allbase/SQL) som hjelper til med lesingen av dataene, en løsning som også er vanlig i større datavarehussystemer.

- Det neste skrittet er å identifisere hvilke data som har forandret seg siden forrige overføring. I de fleste tilfeller vil man kunne benytte en av følgende fire teknikker:

    1. Dersom alle data blir tidsstemplet ved en forandring kan man sammenligne dette tidspunktet mot tidspunktet for den forrige overføringen. Dette er en meget enkel og effektiv måte å skille ut data som har blitt forandret, men dessverre er det få systemer som tilbyr en slik funksjon.

    2. Det kan genereres en «delta-fil» av applikasjonene. Denne filen inneholder de forandringene som er gjort med applikasjonen og kan benyttes til å finne hvilke data som har forandret seg. På samme måte som i punkt 1 er også denne teknikken en enkel og effektiv metode. Men dessverre er det få applikasjoner som lager slike delta-filer, og metoden kan sjelden benyttes.

    3. Man kan benytte seg av loggfila til databasesystemet (dersom den er tilgjengelig). Denne fila inneholder mye av den samme informasjonen som delta-filen i punkt 3, men det er noen betydelige forskjeller. Mange av disse systemene benytter loggfilen i forbindelse med gjenoppbygning av tapte data og beskytter derfor filen godt. Dette innebærer også at den er konstruert med helt andre mål i sikte enn å fortelle hvilke data som har forandret seg siden sist man sjekket. Dermed er den i utgangspunktet relativt dårlig egnet til vårt formål.

    4. Som en siste utvei er det mulig å bevare en kopi av hvordan dataene så ut ved forrige lesning og sammenligne denne med nåværende status. Dette er ikke en spesielt god løsning og representerer på mange måter den siste utveien. Men siden man sjelden har muligheten til å gjennomføre noen av de metodene nevnt ovenfor er det en ganske vanlig løsning.

- Dersom det er forandringer i noen av de sakte forandrende dimensjonene må man generere nye nøkkelverdier for de dataene som har forandret seg. Dette kan enten gjøres slik forklart i kapittel 4.2.1, nemlig ved at man legger til et versjonsnummer til nøkkelen, eller ved at alle nøkkelverdiene er sekvensielle heltall. Begge metodene innebærer omtrent like mye administrativt arbeid. Med versjonsnummer risikerer man å få store nøkler som kan føre til tregere søking, mens med sekvensielle heltall vil man få små nøkler og dertil raskere søking. Hvor store ytelsesforskjellene blir avhenger av størrelsen på datavarehuset. En liten ulempe med sekvensielle heltall er at man må slå opp i en egen kryssreferansetabell ved genereringen for å finne riktig verdi tilhørende den aktuelle dimensjonen. Denne ulempen er dog ikke større enn at man ofte foretrekker sekvensielle heltall fremfor versjonsnummer på grunn av ytelsesforbedringene det medfører.

- Før man overfører dataene til datavarehuset må man legge forholdene best mulig til rette for innlastingen. Med dette menes å arrangere dataene i de riktige radene og kolonnene slik at en direkte innlasting er mulig. I tillegg gjennomføres den konverteringen som er nødvendig for at datavarehuset får integrerte data. Dette utføres før man overfører dataene til datavarehussystemet, siden datakonverteringer o.l. ofte er enklere å utføre der dataene ligger enn i datavarehuset. Generelt lønner det seg å utføre så mye som mulig av overføringsprosessen der dataene ligger, spesielt dersom dette medfører at store datamengder komprimeres til mindre datamengder.

- Ofte vil det være nyttig om man også gjennomfører summasjoner og andre beregninger forut for innlastingen til datavarehuset. Dette er spesielt aktuelt ved store datavarehus der den massive sorteringen og summeringen kan bli svært ressurskrevende for datavarehuset.

- Selve innlastingen av dataene er neste skritt i prosessen. Det anbefales bruk av egne datalastere ([KIMB96]) som muliggjør lasting av hele grupper med data samtidig, men her vil behovene variere med datamengden. I mange tilfeller vil det være tilstrekkelig å laste inn dataene post for post med SQL. En fordel med å laste inn data gruppevis er at man kan utnytte SMP[1] eller MPP[2] bedre og man kan skru av indekseringen under lastingen.

- Før man slipper brukerne løs på dataene vil det være lurt med en eller annen form for kvalitetssikring. Dette kan f.eks. gjøres ved å sjekke at rapporteringen er komplett, dvs. at alle systemer som skulle ha overført data har gjort dette. I tillegg kan man ta stikkprøver ved å summere noen fakta og så sammenligne dette med tidligere historie eller fastsatte grenseverdier for hva som er normalt. Det er også mulig å innføre kvalitetssikringen på et tidligere stadie i overføringsprosessen, gjerne ved hjelp av spesielle verktøy. Det eksisterer f.eks. verktøy som ser at 99% av alle graviditeter er knyttet til hunnkjønn, og ut ifra dette forstår at den prosenten av graviditeter som er hannkjønn er feil [CELK95]. Denne måten å forsikre seg om at datagrunnlaget stemmer er ofte svært nyttig, problemet er at slike verktøy koster penger og må vike for trange budsjetter.

[1] SMP – Simultaneous Multi Processing.

[2] MPP – Massive Parallel Processing.

- Det siste delen i overføringsprosessen er publiseringen av dataene. Det vil være fordelaktig å ha en fast rutine som brukerne er vant til og forventer. Meldingen som sendes ut bør inneholde informasjon om hvilke deler av datavarehuset som har blitt oppdatert, hvordan oppdateringen har gått (om dataene kan benyttes) og eventuelle forandringer på andre deler av datavarehuset som kan påvirke bruken.

I tillegg til disse punktene er det også andre elementer ved dataoverføringen som krever oppmerksomhet. Under hele prosessen må man forholde seg til et tidsvindu som begrenser hva man kan gjennomføre. Tidsvinduet består av to klokkeslett som begrenser tiden man kan benytte til overføringen av data. For å sikre seg at ikke denne prosessen mislykkes bør man på et tidlig tidspunkt avklare hvor stort tidsvindu man har til rådighet og jobbe ut ifra det. Det kan f.eks. ha betydning for om man må benytte gruppelasting av data eller om man greier seg med postvise innlastinger.

Ett annet tema som påvirker overføringen er nettkapasiteten. Båndbredden på nettverket vil ha betydning for hvor lang tid det tar å fysisk overføre dataene mellom systemene. Det er ønskelig at denne delen av prosessen tar så kort tid som mulig for å utnytte tidsvinduet maksimalt. Dette er en av grunnene til at man anbefaler å gjennomføre det meste av prosesseringen forut for overføringen, slik at kun data av betydning blir overført.

4.4 Aktuelle verktøy

For Peterson Ranheims del er det aktuelt å se på en del sluttbrukerverktøy for å bedre utnyttelsen av Datatorget. I den forbindelse er det naturlig å se på verktøy som også kan benyttes sammen med et datavarehus, siden bedriften beveger seg i den retningen.

Det er mulig å dele verktøyene inn i flere forskjellige grupper. En slik inndeling kan være:

- Rapportskrivere og ad hoc spørreverktøy (Crystal Reports, Impromptu)
- Mindre OLAP-verktøy (PowerPlay, Pablo)
- Komplette datavarehussystemer / større OLAP-verktøy (Essbase, DSS/Server, Gentium)

For å ta den siste gruppen først, det er de komplette systemene og større OLAP-verktøyene som gir de beste og mest komplette løsningene jevnt over. I noen tilfeller har produsentene inngått avtaler med konsulentfirmaer slik at det tilbys konsulenthjelp sammen med kjøpet av produktet. Prisene viser også dette, og nivået ligger høyt over den kostnadsrammen som gjelder for et slikt prosjekt hos Peterson Ranheim. Det er bare i de største bedriftene i Norge at slike verktøy virkelig kan være aktuelle. Funksjonaliteten som tilbys fra slike verktøy er riktig nok bedre enn de billigere verktøyene, men med de datamengdene som eksisterer i mange norske bedrifter vil alternativene være gode nok.

De mindre OLAP-verktøyene er av større interesse for Peterson Ranheim. Disse har støtte for en del analytiske beregninger og tilbyr gode brukergrensesnitt. De mest aktuelle verktøyene i denne klassen benytter mindre multidimensjonale databaser med de fordeler dette innebærer. Problemet man møter med disse er at de ikke greier å utnytte datavarehuset spesielt godt. Dette er fordi koblingen mellom de multidimensjonale databasene og de eksterne relasjonsdatabasene er svak, og oftest lastes data inn i verktøyets database med en flat datafil forut for analyser. Direkte spørringer mot datavarehuset er ikke vanlig enda, men noen verktøy begynner å tilby også dette. Sannsynligvis vil de fleste produsentene komme med nye utgaver som tilbyr bedre funksjonalitet på dette området om ikke lenge.

Rapportskrivere og ad hoc spørreverktøy er budsjettverktøyene i datavarehussammenheng. Likevel er disse verktøyene meget nyttige, og er kanskje de som gir mest nytteverdi i forhold til prisen. Disse verktøyene lager SQL-setninger ut ifra brukernes spesifikasjoner og trekker på denne måten ut informasjon fra datavarehuset. Det grafiske brukergrensesnittet forenkler spesifikasjonen av de ønskede dataene og tilbyr en mengde forskjellige måter å presentere svarene på. I Peterson Ranheims tilfelle er slike verktøy meget aktuelle, da de vil kunne løse mange av bedriftens problem i kombinasjon med et datavarehus. Noen analytisk funksjonalitet gir de ikke, men det finnes mange situasjoner hvor dette ikke er påkrevet (jfr. rapportene fra Millman). Et av de største problemene med dagens rapportskrivere er muligheten til å utnytte SQL fullt ut. Kodegeneratoren er ofte ikke god nok til å generere optimaliserte SQL-setninger som plukker ut detaljerte data fra små segmenter av datavarehuset. Dette kan skape problemer i store datavarehus der datamengden er på flere hundre gigabyte. SQL-koden kan da føre til tunge søk som påvirker ytelsen og responstiden til andre. Men på samme måte som at de multidimensjonale verktøyene forbedres kommer det også nye og bedre utgaver her.

For å summere opp forskjellene mellom disse kategoriene kan man sette opp forskjellige egenskaper i en tabell slik gjort i Tabell 4.2. Siden bruksområdene varierer en del vil det være begrenset hva man kan få ut av en slik tabell, men den kan gi en oversikt over hva man kan forvente seg. Det eksisterer også store individuelle forskjeller innen hver kategori, noe som innebærer at alle verktøyene i en kategori ikke nødvendigvis har de egenskapene som er oppført eller at de har noen som ikke er oppført. Deler av tabellen er hentet fra [SAYL95].

Egenskap	Rapportskrivere	Mindre OLAP-verktøy	Komplette systemer
Tilbyr avanserte analytiske beregninger	Nei	Ja	Ja
Støtter en rekke forskjellige brukertyper	Nei	Delvis	Ja
Sørger for hurtig henting av data fra datavarehuset	Delvis	Nei (MDD)	Ja
Støtter veldig store datavarehus (100GB+)	Ja	Nei	Ja (ikke med MDD)
Er enkle å bruke	Delvis	Ja	Ja
Er enkle å vedlikeholde for IT-avdelingen	Delvis	Delvis	Ja
Kommuniserer lett med andre verktøy/systemer	Ja	Nei	Nei
Pris (NOK)[1]	1.000-5.000	3.000-100.000	200.000+

Tabell 4.2 Egenskaper til verktøyene i de forskjellige kategoriene

For å få et mer komplett system vil det være mulig å kombinere verktøy fra flere kategorier. Noen av de billige spørreverktøyene kan f.eks. benyttes til å legge data inn i mindre OLAP-verktøy, hvorpå brukerne bearbeider de videre.

[1] Prisene er beregnet ut ifra eksempler på verktøy i kategoriene og tar utgangspunkt i amerikanske priser, variasjoner kan derfor forekomme. Prisene gjelder stort sett for en brukerlisens, flere for de komplette systemene.

Peterson Ranheim vil sannsynlig vis kunne løse mange av sine problem med billige verktøy fra de to første kategoriene. Det er derfor interessant å se litt nærmere på noen aktuelle kandidater:

4.4.1 Crystal Reports

Crystal Reports 4.5 er en rapportskriver og et utviklingsverktøy som kommer i flere utgaver. Peterson Ranheim har allerede kjøpt inn en lisens for uttesting, og er godt fornøyd så langt . Litt avhengig av hvilken utgave man velger (vanlig/professionell,16-bits/32-bits) ligger prisen på dette verktøyet på ca. 2.000-4.000 kroner per lisens.

Dette verktøyet egner seg godt til å lage standardrapporter ut ifra de dataene som ligger i Datatorget, og rapportene distribueres til brukerne ved f.eks. e-post. Verktøyet tilbyr ellers meget gode muligheter til å presentere dataene, inkludert 80 forskjellige graf-stiler. Kommunikasjonen med databasen går bl.a. via ODBC, men programmet har også direkte støtte for MS Access, MS SQL Server, IBM DB2/2 og en del andre relasjonsdatabaser. Brukervennligheten er ikke like god som det en del multidimensjonale databaser tilbyr, men det er mulig å benytte drill-down på grafene også her. For å sikre god kommunikasjon med andre verktøy benyttes OLE2.0 standarden[1] som er utviklet av Microsoft. Som en liten ekstra detalj kan det også lages oversiktlige rapporter av NT-loggen på en enkel måte (Peterson Ranheim benytter i stor grad Microsoft NT).

Dersom brukerne selv skal sitte og lage rapporter kan det være lurt å anskaffe Crystal Info, et annet produkt fra Crystal. Dette har mye av den samme funksjonaliteten som Crystal Reports, men er i tillegg et gruppeverktøy. Det inkluderer bl.a. en planleggingsmodul som muliggjør kjøring av rapporter til egendefinerte tidspunkt og automatisk spredning til andre brukere. Dersom brukeren prøver å lage en rapport som allerede eksisterer vil programmet varsle om dette for å spare databasen for ekstraarbeid. Prisen på dette programmet er ca. 3.000 kroner.

Mer informasjon om Crystal Reports og Crystal Info finnes på Internett, http://www.crystalinc.com/.

4.4.2 Impromptu

Impromptu fra Cognos har mange likheter med Crystal Reports. Det finnes i to utgaver, en vanlig for brukerne og en litt større for administratorene. Prisen er ca. 4.000 for den vanlige utgaven og ca. 5.000 for den administrative utgaven.

Programmet er beregnet på at brukere selv skal kunne lage sine rapporter og stille spørsmål til databasen uten kunnskap om SQL. Flere maler er tilgjengelig for å hjelpe brukeren i å presentere dataene på en oversiktlig måte og til å hente data fra databasen. I tillegg kan et eget skriptspråk benyttes til å programmere forskjellige rapport-makroer som letter rapporteringen for brukerne. På samme måte som Crystal Reports kan OLE 2.0 benyttes til å inkludere objekter fra andre program i rapporten og omvendt. En egen tidsplanlegger følger med som muliggjør kjøring av rapporter utenom vanlig arbeidstid slik at ressurskrevende oppgaver ikke påvirker responstiden ellers. Det følger også med en rekke ODBC-verktøy, både for å aksessere andre databaser og for å konfigurerer/vedlikeholde ODBC-miljøet.

[1] OLE – Object Linking and Embedding.

Blant de større relasjonsdatabasene som støttes direkte er MS SQL Server, ORACLE, SYBASE SQL Server, SYBASE System 10 og Gupta SQLBase.

Figur 4.4 Skjermbilde som illustrerer en presentasjon av data i Impromptu

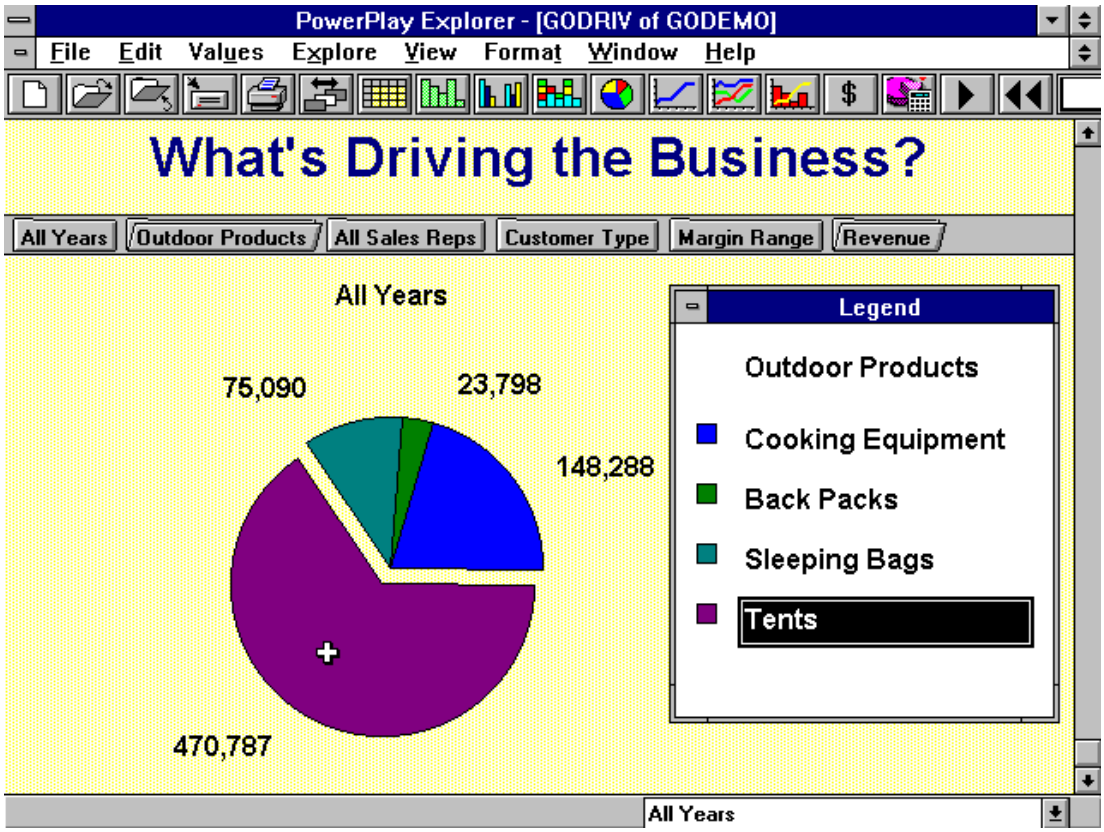
Cognos lager også PowerPlay, som er et MOLAP-verktøy i den lavere prisklassen. Disse to verktøyene samarbeider ganske godt, og ved å plukke ut data fra relasjonsdatabasen med Impromptu for så å legge de inn i PowerPlay oppnår man store fordeler på analysesiden.

4.4.3 PowerPlay

Som nevnt er dette et MOLAP-verktøy av den mindre typen. Den multidimensjonale databasen har derfor tilsvarende begrensninger på hvor store mengder data den kan håndtere. Generelt bør man ikke ha mer enn 8-10 dimensjoner eller ca. 2 millioner poster i databasen. Prisen er ca. 5.000 kroner for den første lisensen, men faller noe ved kvantumskjøp. Det eksisterer også en valgfri tjenerutgave som forbedrer ytelsen til databasen betraktelig (tillater 10-20 millioner poster), og denne koster ca. 35.000 kroner (NT-plattform).

Den virkelig store fordelen med PowerPlay er det intuitive brukergrensesnittet som tilbys. Den multidimensjonale måten å presentere dataene på virker veldig naturlig, og brukerne trenger lite eller ingen opplæring for å ta i bruk verktøyet. Ved å hjelp av musen kan brukeren flytte dimensjoner ut og inn fra synsfeltet, sette begrensninger i dimensjonene, foreta drill-down og drill-up osv. Han/hun kan også påvirke presentasjonen av dataene ved

f.eks. å skjule nullverdier eller fremheve tall som skiller seg ut fra egendefinerte grenseverdier. Etter hvert som man lager rapporter kan man lagre disse og plukke de frem igjen ved senere anledninger. Ved hjelp av OLE kan man også lenke informasjon fra PowerPlay inn i andre applikasjoner som f.eks. MS Word, MS Excel, Lotus Notes.



Figur 4.5 Skjermbilde fra PowerPlay

Svakheten til verktøyet er tilgangen på data fra relasjonsdatabasen. Det er ikke mulig å benytte slike data direkte uten først å føre de over til den multidimensjonale databasen. For at dette ikke skal bli et for stort problem følger det med en avansert oversetter som gjør det mulig å laste databasen med flate, todimensjonale datafiler. Oversetteren analyserer disse filene og bestemmer den multidimensjonale strukturen, inkludert drill-down strukturen. Den gir brukeren muligheten til å forandre på navn, omorganisere og omgruppere innen dimensjonene. I tillegg til dette har verktøyet en innebygget forståelse av tid, noe som er nyttig i de fleste tilfeller.

Men det eksisterer også andre måter å laste data inn i databasen. PowerPlay kan direkte aksessere datakilder som dBASE, Paradox, MS Excel og Lotus 1-2-3. I tillegg kan man, som nevnt, benytte Impromptu som et mellomledd. En rekke ODBC-drivere blir tilgjengelig gjennom spørringer laget i Impromptu, og gjør at store relasjonsdatabaser som ORACLE og SYBASE SQL Server kan aksesseres. En kombinasjon av disse to verktøyene blir meget kraftig og fører til et meget godt kostnytte forhold.

Dersom man i tillegg tar i bruk tjeneren vil man forbedre innlastingen av data ytterligere. Denne lagrer komprimerte multidimensjonale datafiler som består av data hentet direkte fra relasjonsdatabaser eller flate datafiler. Klientene kan så aksessere og manipulere disse datafilene som om de var lokale filer. I et forsøk på å finne en åpen løsning er tjeneren også tilrettelagt for at MS Visual Basic skal kunne aksessere dataene gjennom OLE 2.0.

Mer informasjon om Cognos' Impromptu og PowerPlay finnes på Internett, http://www.cognos.com/.

4.4.4 MS Excel

Det kan kanskje virke litt merkelig å ta med MS Excel her, men faktum er at dette verktøyet har mange nyttige funksjoner i forbindelse med analyse av data. Styrken til Excel ligger i bruken av pivottabeller. Denne enkle tilleggsfunksjonen gjør at verktøyet kan vise og manipulere data på forskjellige måter samt utføre en del nyttige kalkulasjoner. Et enkelt eksempel på bruk av pivottabell er at man markerer et datasett i regnearket og fører dette over til en pivottabell. Da blir man spurt om hva hver kolonne i datasettet representerer, om det er en kolonne eller rad i krysstabellen eller om det er verdier som skal brukes til beregningene. Til slutt presenteres dataene slik man har spesifisert.

Regneark er i de fleste tilfeller godt kjent blant brukerne fra før av, og denne måten å presentere data på er lettforståelig. Dermed kreves det lite opplæring før man kan ta i bruk de mulighetene som eksisterer. Ekstra nyttig er det at man kan programmere avanserte makroer som hjelper brukerne til å utnytte datamengden best mulig. I tillegg kan man på mange måter se på verktøyet som et gratis verktøy, da man ofte har kjøpt det inn på forhånd i forbindelse med andre behov.

Det er ikke meningen at man skal se på MS Excel som en *erstatning* for andre OLAP-verktøy, men mer som et tillegg. Det kan fungere som et utgangspunkt for å få brukerne interessert i datavarehuset, og kan hjelpe brukerne med å gi nyttige tilbakemeldinger til IT-avdelingen.

4.4.5 Intranett og WWW

I forbindelse med den oppdelingen av verktøy som ble gjort ovenfor kunne man også ha lagt til en ekstra kategori som inneholdt WWW-klienter og WWW-tjenere. Grunnen til at dette ikke ble gjort er at man først nylig har begynt å tenke på denne typen verktøy som aktuelle kandidater til analytisk prosessering. Derfor er også utvalget av eksisterende verktøy for lite til at man kan plassere de i en kategori for seg selv.

Mye av æren for at slike verktøy har blomstret opp må gå til Internett. Samtidig er det nettopp Internett som gjør at mange er skeptiske til slike verktøy, eller rettere sagt til miljøet de opererer i. Det som i første rekke skremmer folk er mangelen på sikkerhet, muligheten for at noen utenforstående skal få tak i konfidensiell informasjon. En løsningen på dette problemet er å flytte verktøyene fra det åpne miljøet som Internett er og i stedet benytte de i et «Intranett». Et Intranett er i realiteten bare et annet ord for lokalnett, men hvor man i tillegg benytter teknologi fra Internett.

Det er flere måter man kan se for seg at disse verktøyene vil kunne tilpasses datavarehus. Man kan tenke seg at man i fremtiden vil produsere rapporter og automatisk konvertere de til HTML-format[1] som så WWW-klienter (f.eks. Netscape) deretter kan lese. En annen løsning kan være at man lar WWW-tjenerne aksessere datavarehuset og generere sider ut i fra de kriteriene som er spesifisert fra WWW-klienten.

Noen mener at WWW-klientene ikke er fleksible nok og ikke gir den ønskede funksjonaliteten til at de vil lykkes [NEWS96]. Men med de muligheter som følger med det nye programmeringsspråket Java[2] vil slike verktøy kunne tilfredsstille mange av brukernes behov om ikke lenge. Det eksisterer allerede eksempler på Internett hvor Java benyttes til å utføre drill-down og drill-up i et datasett på en meget elegant måte[3]. Hvis man da i tillegg tar med at brukergrensesnittet knapt kan bli enklere enn det disse verktøyene tilbyr er det klart at de har en fremtid på området. Hvilke begrensninger som eksisterer er det ingen som er fullstendig klar over i dag.

I [FOLE96] får man nok et tegn på at dette har noe for seg. Der omtales MS SQL Server 6.5 (Beta), hvor det bl.a. kommer frem at denne utgaven vil ha utvidet støtte for datavarehus. I tillegg til dette inkluderes en «Web Assistant» som gjør at brukerne automatisk kan formatere data for WWW og sende de til en WWW-tjener. Det er også mulig å gå andre veien, fra WWW til databasen. Ved hjelp av «Internet Connector», som følger med MS Internet Information Server (Microsofts WWW-tjener), får man nemlig direkte tilgang til MS SQL Server 6.5. Det er litt usikkert hvor god denne tilgangen er, men med tiden vil den sikkert bli tilfredsstillende.

For Peterson Ranheim vil en slik løsning ha flere fordeler. Bedriften har allerede et lokalnett hvor de kjører Microsofts WWW-tjener og brukerne har sine WWW-klienter. I tillegg vurderes det å benytte MS SQL Server til datavarehuset, et valg som kan vise seg å være nyttig på flere måter. Problemet er at ingen i bedriften har erfaring med å programmere i Java, noe som fører til at en viss innsats må ytes. Kombinert med at utviklingen på området fortsatt er på et tidlig stadium gjør dette at WWW-verktøy alene ikke er noen løsning på bedriftens problem. Men med de store mulighetene som eksisterer vil det være lurt å ha disse i bakhodet under konstruksjonen av datavarehuset..

[1] HTML står for HyperText Markup Language, og er språket som benyttes for å spesifisere layout på en WWW-side.

[2] Klienter som støtter Java gjør det mulig å kjøre programmer hos klienten i stedet for at all informasjon må overføres via nettet.

[3] http://www.itivity.com/

5. Konklusjon

Skillet mellom datavarehus og OLAP er for tiden veldig flytende. På samme måte hersker det mye usikkerhet rundt hvilke verktøy som er OLAP-verktøy og hvilke som ikke er det. Mye tid gikk derfor med til å sette seg inn i begge disse fagområdene til å begynne med. Etter hvert ble det mer forståelig hva de inneholdt, og det ble klart at begge deler hørte hjemme i en rapport om datavarehus. Koblingen mellom dem er så sterk at dersom man først har konstruert et datavarehus er det liten grunn til ikke å benytte OLAP. Det vil derfor være hensiktsmessig å ha OLAP i tankene under konstruksjonsprosessen, slik at resultatet er godt egnet for slike utvidelser.

Nesten alt som står om datavarehus retter seg mot store databaser som har opp mot flere terrabyte med data. Sammenlignet med disse vil databasene til Peterson Ranheim og lignende bedrifter i Norge bli små. Likevel eksisterer det også her et behov for bedre løsninger enn hva man har i dag, noe blant annet problemene hos Peterson Ranheim viser. Etter hvert vil leverandørene forstå at det ligger et marked også hos de små og mellomstore bedriftene (SMB), og som et tegn på dette har det i diskusjonsgruppene dukket opp et nytt begrep kalt «datamart» (datatorg). Sannsynligvis vil man om ikke lenge se løsninger som er billigere og bedre egnet til slike «datatorg» enn hva dagens produkter er.

Datavarehus er likevel ikke noen vidundermedisin som alle bedrifter bør kaste seg over. Mange av de verktøyene som er nevnt i rapporten kan også jobbe direkte mot vanlige transaksjons-systemer, forutsatt at det er snakk om relasjonsdatabaser. En slik løsning kan spare IT-avdelingen for mye av det arbeidet som kreves for å konstruere et datavarehus. Men i mange tilfeller benyttes gamle systemer som ikke bruker SQL, og noen ganger har man et behov for å samle informasjon som er spredt over flere systemer. Det er i første rekke her man bør vurdere et datavarehus som et aktuelt alternativ. Dersom man i tillegg har andre problemer med de gjeldende systemene, slik situasjonen er hos Peterson Ranheim, vil et datavarehus være enda mer aktuelt.

I løsningen som er skissert i denne rapporten anbefales det å benytte en relasjonsdatabase med et stjerneskjema til datavarehuset. Denne anbefalingen er gjort på grunnlag av at dette vil være en billig løsning hvor man kan utnytte de erfaringene man allerede har med Datatorget. I tillegg vil løsningen være fleksibel med hensyn på hvilke verktøy bedriften kan tenkes å kjøpe senere. Ikke minst vil dette innebære at bedriften kan foreta utbyggingen av Datatorget skrittvis uten å måtte investere i uforholdsmessig dyre verktøy med en gang.

Helt til slutt anbefales det nok en gang å se på de muligheter Intranett og WWW-verktøy gir. Utviklingshastigheten innen dette området tilsier at mange gode alternativer vil dukke opp i løpet av kort tid, og det lave kostnadsnivået gjør slike løsninger meget attraktive.

Vedlegg

A. Eksempel på en rapport

Dessverre er tallene i denne rapporten konfidensielle, og kan derfor ikke offentliggjøres. Vedlegget er av denne grunn fjernet fra den utlagte rapporten.

B. Bibliografi

[ABER95] Aberdeen Group, Inc.:
«Data Warehouse Query Tools: Evolving to Relational OLAP»
Market Viewpoint, 7. juli 1995
Internett: http://www.strategy.com/dwf/aber_.htm#D

[ARCH95] Archer Decision Sciences, Inc.:
«Star Schema 101»
Internett: http://www.aol.com/nraden/index.html

[BAUM95] David Baum:
«Warehouse Mania»
LAN Times, 20.november 1995

[BUTL95] Butler Group:
«Data Warehousing; Management Guide, Strategies and Technologies»
September 1995

[CELK95] Joe Celko og Jackie McDonald:
«Don't Warehouse Dirty Data»
Datamation, 15. oktober 1995

[CODD93] E.F. Codd, S.B. Codd og C.T. Salley:
«Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate»
Internett: http://www.arborsoft.com/papers/coddTOC.html

[COMP95] Ståle Lian:
«Data Warehousing - motebegrep eller kommet for å bli?»
Computerworld, nr. 45 1995

[DEPO96] Barbare DePompa:
«Warehousing - Stack That Data - The Web, parallelism, and data-cleaning tools will help users organize information»
Information Week, 29. januar 1996

[FINK95] Richard Finkelstein:
«Multidimensional Database: Where Relational Fears to Tread»
Database Programming & Design, april 1995

[FOLE96] John Foley:
«SQL Server Links To Web - New release boasts data warehousing, Web access capabilities»
Information Week, 26. februar 1996

[HACK95] Dr. Richard Hackathorn:
«Data Warehousing Energizes Your Enterprise»
Datamation, 1. februar 1995

[HAIS95] Michael Haisten:
«Planning for a Data Warehouse»
Internett: http://www.netlynx.com/~dbai/v9nla4.html

[INMO92] William H. Inmon:
«Building The Data Warehouse»
John Wiley & Sons, Inc. (Wiley), 1.utgave 1992, ISBN 0-471-56960-7

[KENA95] Kenan Technologies:
«An introduction to Multidimensional Database Technology»
Internett: http://www.kenan.com/acumate/mddb_end.htm

[KIMB96] Ralph Kimball:
«The Data Warehouse Toolkit»
John Wiley & Sons, Inc. (Wiley), 1.utgave 1996, ISBN 0-471-15337-0

[McNA96] Giles McNamee:
«Eyeing The Data Warehouse — Database vendors will be winners, but not all can deliver solutions yet»
Information Week, 5. februar 1996

[MIMN95] Pieter Mimno
«3 decisions critical to make for warehouse development»
Application Development Trends 1995, December 1995, Section: Data Warehouse;

[NEWS96] Innlegg i diskusjonsgruppen «comp.databases.olap» på USUNET i 1996

[NOLA96] Peter Nolan:
«What is the Data Warehouse All About?»
Nolan Consultants
Internett: http://www.ozemail.com.au/~pnolan/

[OLAP95] Nigel Pendse og Richard Creeth:
«The OLAP Report»
Business Intelligence Ltd., Utgitt 1995, ISBN 1-898-085-21-8

[PERI94] Ron Peri:
«Linking Databases: Planning Is Key»
Internett: http://www.cmp.com/interop/linking.html

[RADE95] Neil Raden:
«Data, Data Everywhere»
Information Week, 30. oktober 1995

[RADE96] Neil Raden:
«Technology Tutorial — Modeling a Data Warehouse — Value to an organization means turning data into actionable information»
Information Week, 29. januar 1996

[REDB95] Red Brick Systems:
«Star Scehmas and STARjoin Technology»
Internett: http://www.redbrick.com/rbs/whitepapers/star_wp.html

[SAYL95] Michael J. Saylor, et.al.:
«True Relational OLAP: The future of decision support»
Database Journal, november/desember utgaven 1995