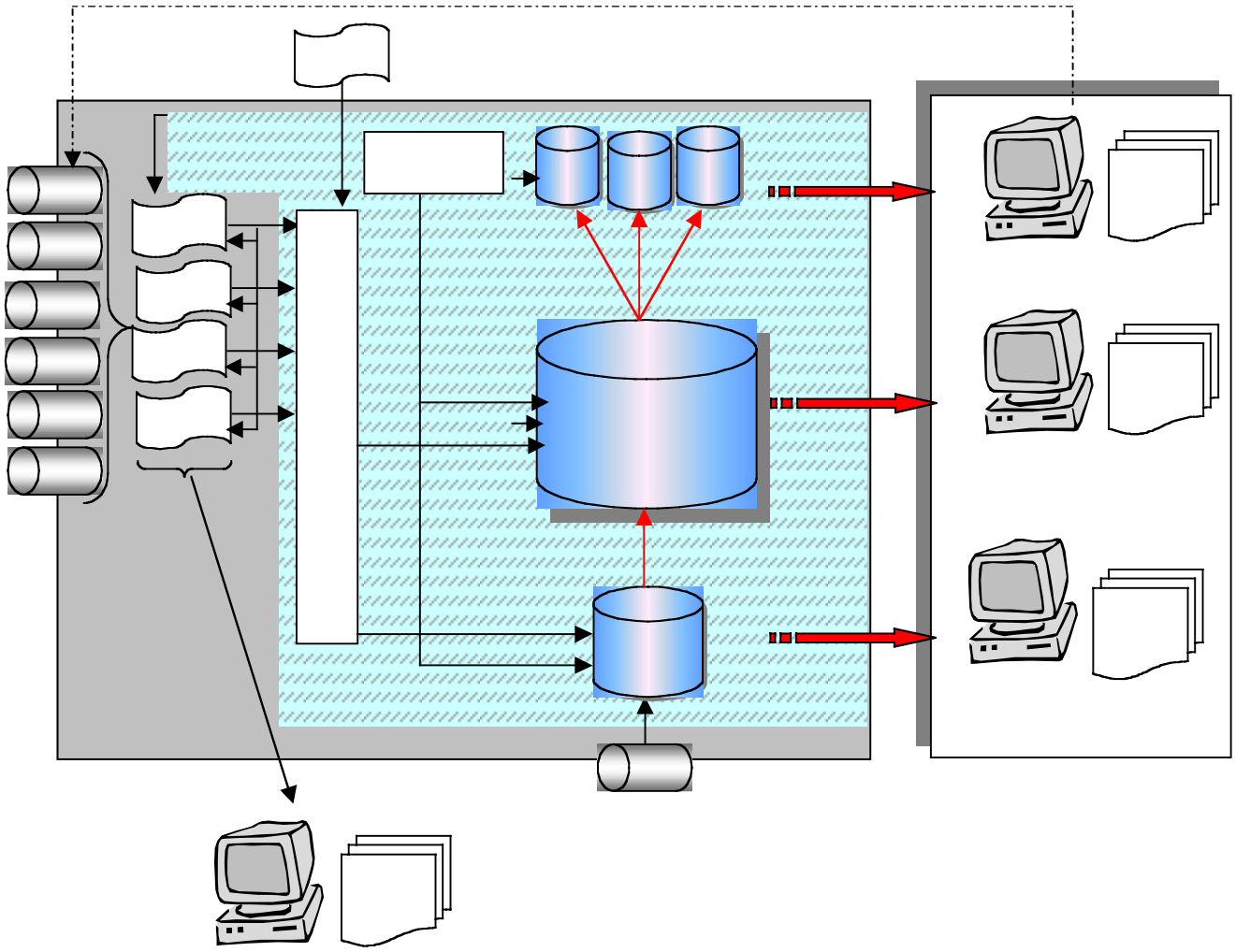
Data warehousing og Knowledge Management

Strukturering af Informationer til Analyseformål



AALBORG UNIVERSITET
MASTER i
INFORMATIONSTEKNOLOGI,
INDUSTRIEL IT (MII)
PROJEKT GRUPPE: IT i BYGGERIET
& IT i INDUSTRIEL PRODUKTION
FORÅR 2001

Strukturering af informationer til analyseformål

Master i Informationsteknologi, Industriel IT – 2. år, AAU

Titel: **"Strukturering af informationer til analyseformål"**
Tema: **"Data warehousing og Knowledge Management"**
Projektperiode: **1. september 1999 til 30. maj 2001**

Forfattere:

Kim Friis Hansen

Thomas Hundebøl Thomsen

René Aaholm

Synopsis:

Data warehousing has quickly evolved into a unique and popular business application class. Early builders of data warehouses already consider their systems to be key components of their IT strategy and architecture. Numerous examples can be cited of highly successful data warehouses developed and deployed for businesses of all sizes and all types.

This project report deals with data warehouse theory, technology and its implementation in businesses.

Our objective is to describe the project process when data warehouse technology is developed and implemented in a business organization.

What is a data warehouse? A definition requires discussion of many key attributes of a data warehouse system. This project report will introduce key concepts surrounding the data warehousing technology.

Vejledere: **Kaj A. Jørgensen og Per Christiansson**
Oplagstal: **5**
Sideantal: **39**
Bilagsantal og-art: **0 bilag, 0 appendix**
Afsluttet den: **30. maj 2001**

Indholdsfortegnelse

1. Indledning

Virksomheder genererer og indsamler store mængder data, som bruges i den daglige drift. Udfordringen består i at udnytte hele værdien af disse data, dvs. at kunne udtrække de nyttige informationer, der gemmer sig i den store mængde af data.

Et data warehouse er stedet, hvor informationerne indsamles og struktureres for at skabe overblik over organisationen og dens aktiviteter.

En af de vigtigste forskelle mellem et data warehouse og en almindelig database er tidsdimensionen. Et data warehouse fortæller fx ikke blot hvem, der køber mest, men også hvad der er sket over en given periode. Med andre ord kan det skabe et grundlag for strategiske beslutninger.
En anden vigtig forskel er, at data warehouse er et centralt indsamlingssted for informationer fra hele organisationens univers. I stedet for at skulle forbinde flere forskellige datakilder for at finde de ønskede informationer, skal man kun kigge ét sted. Her får man adgang til både enkelte, sammenlagte og opsummerede informationer fra hele organisationen, dvs. produktion, distribution, salg, økonomi, osv.

Et data warehouse giver mulighed for analyser af arbejdsgange og aktiviteter både inden for og uden for organisationen. Dermed kan man vurdere, hvor ressourcerne bedst kan anvendes for at optimere indtjeningen. Data warehouse omfatter beslutningsgrundlag inden for følgende områder:

• Strategivalg

• Fastholdelse af ret kurs

• Optimering af indkøb og lager

• Optimering af kundeservice

• Medarbejderressourcer og -udvikling

• Strukturændringer i branchen

• Konkurrenceforhold.

Organisationens virksomhedssystemer som økonomi, indkøb, produktion, logistik, osv. leverer en stor mængde data til data warehouse systemet. Her bliver disse data struktureret på en måde, der gør det nemt at præsentere et sikkert beslutningsgrundlag.

Knowledge Management drejer sig om at strukturere og dele viden for at gøre vores informationer og viden operationelle, dvs. til strategiske værktøjer.

En hændelse i form af en ordre, en ansættelse eller en intern transaktion registreres som data og indgår i det fælles informationslager – data warehouse. Herfra kan informationerne lægges sammen, kombineres, og deres indbyrdes relationer analyseres for at tilføre organisationen ny viden. Denne viden udgør en platform for strategiske beslutninger på ledelsesplan og de handlinger i organisationen, der skal sikre implementering af strategierne.

2. Problemformulering

Formålet med denne rapport er, at anvise et praktisk projektforløb ved analyse af implementeringen af et data warehouse. Der fokuseres i rapporten på praktisk etablering af et data warehouse med reference til teorien og teknikker i data warehouse. Der udføres ikke implementering af et data warehouse i dette projekt.

Opgavens problemstilling kan formuleres således:

Hvorledes kan man gennem et projektforløb, udvikle og implementere et data warehouse, der bidrager til bedre beslutninger i en virksomhed ?

Som et eksempel i rapporten, anvendes en fiktiv virksomhed kaldet CaseFirm.

CaseFirm er en virksomhed, som producerer søm og skruer til såvel hjemmemarkedet som til eksport. Produktsortimentet er bredt inden for standardtyper, og der indgår også enkelte typer på basis af licensproduktion. Alle standardtyper produceres hovedsageligt til lager, hvorfra de løbende sælges til grossister, detailkæder, entreprenører samt til virksomheder i byggeindustrien.

Produktsortimentet er meget varieret både inden for søm og skruer. Alle typer bliver produceret på eget udstyr i serier på flere maskiner. Varerne pakkes i henholdsvis små pakninger med 5 - 25 stk. eller kasser med 200 - 1.000 stk., hvorefter de lægges på lager. Produkterne fremstilles på få special maskiner til henholdsvis søm og skruer, der omstilles til de forskellige typer. Omstillingen er tidskrævende, og der tilstræbes derfor en produktion i store serie af hver type. Ofte må produktionen dog omstilles midt i en serie for at kunne efterkomme aktuelle ordre af andre typer.

Afsætning af produkterne er kendetegnet ved at være varierende og i nogen grad sæsonpræget. Leveringstiden er kort, og er ofte en betingelse for at kunne gennemføre et salg. Der planlægges derfor efter at alle typer altid er lagervarer, men ind imellem oplever de at det ikke er muligt at levere visse typer, mens der ligger store mængder af andre typer på lageret.

I visse perioder af året er varelageret meget omfattende og kapitalkrævende for virksomheden. Hver kvartal gennemfører økonomiafdelingen en analyse af lagerbeholdning med henblik på at kunne minimere beholdningen og producere just in time. Kombineret med oplysningerne fra salgsafdelingen om forventningerne til salget i det kommende kvartal, danner dette grundlag for planlægning af den næste kvartals produktion, men ofte når informationerne først frem til produktionen 3 til 4 uger inde i det nye kvartal.

I dag er informationerne registeret på flere systemer. I produktionen anvendes et system til styring af råvarer og til planlægning. I lageret anvendes et andet system til styring af færdigvarer og til pakning og forsendelse af varer til kunder. Administrationen bruger sammen med salgsafdelingen et særskilt økonomistyresystem. Disse systemer er ikke integreret bortset fra et fælles netværk og centrale servere.

Efter den seneste kvartalsanalyse af varestrømmene, der viste meget store lagre er der formuleret et indsatsområde om udvikling af bedre metoder til at minimere lagerbeholdningen af de enkelte varenumre, og samtidig kunne levere varen. Virksomheden har derfor besluttet, at igangsætte et pilotprojekt til afklaring af om data warehouse kan anvendes til at optimere produktionen.

Pilotprojektet skal blandt andet give svar på følgende spørgsmål:

- Hvad er data warehouse?
- Hvordan er arkitekturen i et data warehouse?
- Hvilke datamodeller anvendes i data warehouse?
- Hvilke krav stiller data warehouse til data?
- Hvilke værktøjer anvendes i forbindelse med data warehouse?
- Kan data warehouse give informationer om forventninger til afsætning af de enkelte varer?
- Kan data warehouse opfylde behov for rapportering?
- Kan data warehouse anvendes til analyseformål, f.eks. til at finde nye sammenhænge i eksisterende data?
- Hvilke typer beslutningsstøtte kan data warehouse supportere?

3. CaseFirm – Beskrivelse af virksomheden

Virksomheden CaseFirm producerer søm og skruer. Med en stigende efterspørgsel og produktportefølje, har virksomheden besluttet, at sælge sine produkter gennem følgende salgskanaler: grossister, detailkæder, entreprenører samt til virksomheder i byggeindustrien.

I de seneste år har CaseFirm etableret nye produktionsenheder, som følge af en stigende efterspørgsel fra virksomhedens kunder. Da virksomheden primært har fokuseret på ekspansion, har den ikke brugt mange ressourcer på at måle effektiviteten af denne ekspansion.

CaseFirms analyser af varestømme viser meget store lagre og ledelsen er begyndt at fokusere på organisations ydeevne. Virksomheden har et godt datagrundlag for sin omsætning og lager-beholdning i virksomheden som helhed, men mangler data om de enkelte varenumre og deres afsætning og de indbyrdes relationer.

Ledelsen har derfor bedt virksomhedens IT Afdeling udarbejde et forslag til implementering af et data warehouse.

3.1. Projekt Plan

IT afdelingen udarbejder en projekt plan med følgende mål og afgrænsning:

Projekt mål
At danne et data warehouse med det formål, at bidrage til analyse af data vedrørende salg og lagerbeholdning, for produkter produceret og solgt af CaseFirm.

Projekt afgrænsning
Projektet afgrænses til analyse af salg og lagerbeholdning i relation til produkterne.

3.1.1. Definition af virksomhedens behov

Projektgruppen identificerede følgende områder som værende interessante, for at kunne definere og forstå virksomhedens behov:
· Et produkts livscyklus
· Salgskanaler
· Organisationens struktur
· Definition af salg og lagerbeholdning
· Hvad ønsker brugerne ?

3.1.2. Et produkts livscyklus.

Projektgruppen analyserede først et produkts livscyklus. Hver produktionsenhed har en udviklingsgruppe tilknyttet, der afprøver nye produkt ideer. Kun når produktionsprocessen er fuldstændig defineret og det nye produkt er godkendt, bliver produktinformationer tilføjet til databasen. Når produktinformationerne er tilstede i databasen, kan produktionsenhederne begynde at producere produkterne. Produktmodellerne er meget varieret både inden for søm og skruer. Fabrikken har et lager af produktmodeller. Når lagerbeholdningen af en model falder til et forudbestemt niveau, igangsættes der en ordre om at producere flere modeller. Når en model er produceret, lægges den på lager, indtil den bestilles via salgskanalerne.

Salgskanalerne er ansvarlig for at sælge modellen. Når det besluttes at ophøre med at producere en model, gemmes data om modellen i 6 måneder efter at den sidste enhed af modellen er blevet solgt eller kasseret. Produktdata fjernes samtidig med at disse data fjernes.

3.1.3. Salgskanaler

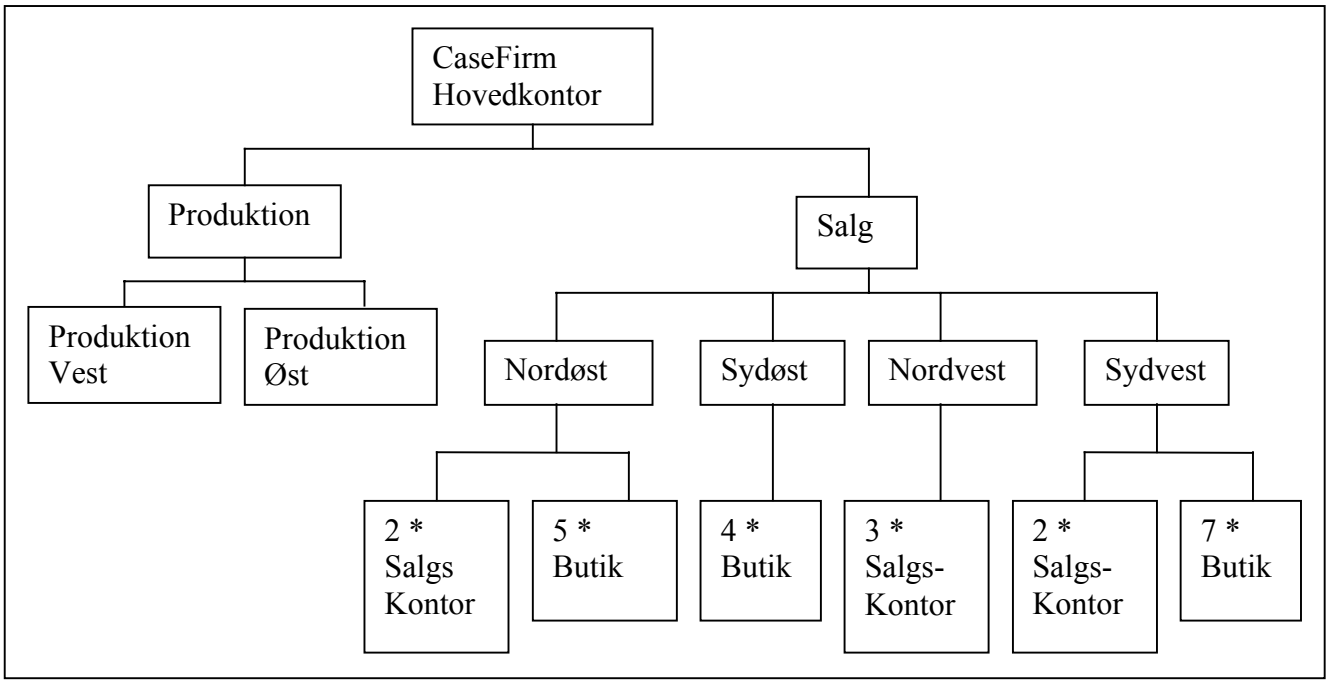
Der er 2 typer af salgsteder: hovedkontorets salgsafdeling og butikkerne. Hovedkontorets salgsafdeling sælger kun til virksomheder. Virksomheder betaler vejledende pris, med mindre der er aftalt rabat. Til hver kunde er der knyttet en salgskonsulent. Hver salgskonsulent har flere kunder. CaseFirm har p.t. 3000 virksomheder som kunder. En kunde kan afgive ordre direkte til en salgskonsulent eller til
hovedkontorets salgsafdeling. Ordre der modtages fra hovedkontorets salgsafdeling bliver sendt direkte fra fabrikken til kunden. En kunde kan have flere leveringsadresser. Det er muligt for kunderne, at afgive ordrer til levering på flere lokationer, hvis de ønsker det.

Hovedkontorets salgsafdeling placerer ordrerne på de produktionsenheder, der ligger nærmest leveringsadresser. Hvis en kunde afgiver ordrer til levering på flere addresser, opdeles ordren til en ordre pr. leveringsadresse. Hovedkontorets salgsafdeling modtager i gennemsnit 500 ordrer pr. dag, 5 dage om ugen. Hver ordrer består i gennemsnit af 10 produktmodeller.

En butik sælger direkte til privat kunder. Hvis ikke der er aftalt rabat, betales den vejledende pris. Selvom hvert salg bliver registreret som en ordre, gemmes der ikke data om kunden. En butik kan afgive ordrer til en produktionsenhed. Lederen af butikken er ansvarlig for hvilke produkter der lagerføres og sælges fra butikken. En butik genererer i gennemsnit 1000 ordrer pr. dag 7 dage om ugen. Hver ordre indeholder i gennemsnit 2 produktmodeller.

3.1.4. Organisationens struktur

Virksomhedens organisationsstruktur er vist i nedenstående figur.



Figur 3.1 CaseFirm organisationsstruktur

3.1.5. Definition af salg og lager

Det er nødvendigt at klart definere salg og lager for at kunne analysere disse faktorer.

Lager:
For hver produktmodel adderes kostprisen for hver komponent der indgår i produktmodellen og summen af alle komponenternes kostpris er lig med produktmodellens kostpris.

Salg:
For hver produktmodel multipliceres antallet af solgte enheder med vejledende pris. Summen af alle ordrelinier der indeholder modellen er lig med salget af produktmodellen.

3.1.6. Hvad ønsker brugerne ?

Formålet med projektet er at danne en samling af data, som brugerne kan analysere effektivt. Projektgruppen identificerede en række typiske spørgsmål som brugerne kunne forventes at data ville kunne svare på.

1. Hvad er det gennemsnitlig antal af produktmodeller på lager og genbestilling denne måned i hver produktionsenhed ?

2. Hvilke modeller og produkter er ikke blevet solgt i sidste uge ?

3. Hvilke modeller tilhører top 5 listen sidste måned målt på salg ?

Brugerne ønsker endvidere at kunne analysere disse spørgsmål over en 3 årig periode, for at se ændringer over tid.

3.1.7. Adgang til operationelle data

Næste skridt, efter at have defineret virksomhedens behov er, at finde de nødvendige data til at danne et data warehouse. For at gøre dette er det nødvendigt at se på ER (Entitet Relation) modellerne for alle relevante data i det operationelle system:

3.2. OLTP

Hovedformålet med et data warehouse er, at "trække" information ud af operationelle data. Dette fører til de basale funktioner i et data warehouse: udtræk af data fra eksterne og interne operationelle databaser og derefter rense, transformere, kontrollere og organisere data i et optimalt format, med henblik på, at give nem og hurtig adgang til analyse af virksomhedens ydeevne.
I virksomheden CaseFirm anvendes applikationer til at støtte den daglige drift. Disse dag-til-dag forretningsapplikationer er sædvanligvis baseret på relationelle databaser, der er optimeret mht. transaktionsorienterede operationer.

3.2.1. Definition af OLTP

I det følgende defineres OLTP, med henblik på at give en forståelse af forskellen mellem applikationer til den daglige forretningsdrift og applikationer der bidrager til at informere og støtte beslutningstagerne i virksomheden.

Definition
Ved OLTP – Online Transaction Processing menes alle de applikationer, der anvendes i den daglige forretning: fx: ordresystem, lagerstyringssystem eller lønsystem. Denne type af applikationer benævnes undertiden også som OSS - (Operationelle Støtte Systemer). Disse systemer håndterer tusindvis, måske endda millioner af transaktioner om dagen og sikrer konsistens mellem alle relaterede tabeller i databasen. Alle opdateringer af relaterede data afspejles i databasen. Indholdet af database tabellerne og deres indbyrdes relationer, vil i et OLTP system, ændres løbende gennem en arbejdsdag i en virksomhed.

De rapporter og forespørgsler der dannes fra en database af denne type, vil give et øjebliksbillede af den temporære tilstand i databasen.

3.3. Datakilder

CaseFirm er nu beskrevet og vi har samlet brugernes krav til vores data warehouse. Vi skal nu sikre os at alle nødvendige dataelementer eksisterer, så vi kan tilfredstille brugernes krav.

Det bedste sted at starte er i de eksisterende OLTP systemer. Vi skal identificere de OLTP applikationer, der anvendes til at producere virksomhedens forretningsdata og dermed er datakilder til vores data warehouse.

Hver enkelt OLTP applikation har en specifik teknik til at håndtere data i databasen. Det er vigtigt at vide, hvorledes applikationen håndterer selv simple transaktioner som sletning af poster (fx: bliver posten fysisk slettet fra databasen eller er den markeret som slettet og forbliver i database tabellen). Disse forhold kan få indflydelse på, hvorledes vores data warehouse kan implementeres.

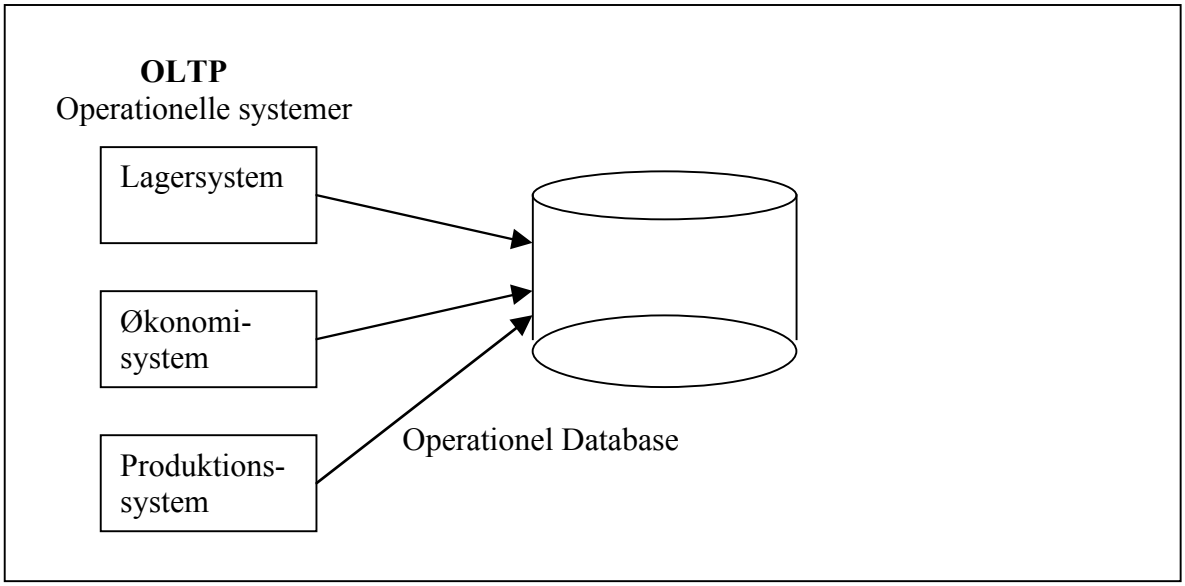
Vores analyse af databasen, hvor vi finder de nødvendige dataelementer i OLTP applikationerne er meget vigtig. Det er vigtigt at forstå den præcise betydning af hver eneste feltværdi, der potentielt er kilde til vores data warehouse, så vi undgår forkerte fortolkninger.

De spørgsmål der skal besvares af databaseanalysen er som følger:

1. Er alle værdier tilstede i OLTP databasen, så brugernes krav kan opfyldes ?
2. Hvilke nødvendige dataelementer kan ikke hentes fra OLTP databasen.
3. Under hvilke omstændigheder/processer ændres data ?
4. Hvordan håndterer de enkelte OLTP applikationer deres data ?

Vi kan af beskrivelse af CaseFirm identificere tre operationelle systemer (OLTP applikationer).

1. Lagersystem
2. Økonomisystem
3. Produktionssystem



Figur 3.2 CaseFirm operationelle systemer

3.3.1 Lagersystem

I lageret anvendes et system til styring af færdigvarer og til pakning og forsendelse af varer til kunder. Afdelingen har følgende styringsopgave:

• Lagerstyring, der fokuserer på styring af lagerbeholdningernes niveau. Dette inkluderer også den fysiske materialehåndtering, der omfatter aktiviteter såsom: Fremtage, placere, plukke og pakke varer i forbindelse med, at varer flyttes fra et sted til et andet.
Lagerstyringen har til opgave at servicere de tre områder: materiale-, produktions- og distributionsstyring med informationer om lagerstatus (råvarer, varer-i-arbejde og færdigvarer), ind- og udlevering, lagerplanlægning osv., således at der kan sikres et optimalt vare-/serviceflow.

3.3.2 Økonomisystem

Administrationen bruger sammen med salgsafdelingen et særskilt økonomistyresystem.

Økonomisystemet i CaseFirm sikrer:

- effektiv registrering og rapportering af virksomhedens økonomiske data
- kvalitet og fuldstændighed i registrering og rapportering af virksomhedens økonomiske data
- Ordrebehandling

Modtagelsen af en ordre, involverer indtastning og genindtastning af ordre- og kundeoplysninger hos sælger, salgsafdeling, produktion, lager og økonomiafdelingen. Ordre- og kundeinformationerne indtastes og genindtastes i forskellige uafhængige systemer, hvilket naturligvis øger risikoen for menneskelige fejl, men samtidig besværliggør muligheden for at overskue, hvor langt kundens ordre er nået i systemet.

3.3.3 Produktionssystem
I produktionen anvendes et system til styring af råvarer og til planlægning og afdelingen har følgende styringsopgave:

• Produktionsstyring, der omfatter de aktiviteter, der vedrører materialeflowet fra råvarelagre, varer i arbejde på forskellige produktionsprocesser, mellemvarelagring til færdigvarelagre.
Målet for produktionsstyringen er inden for en række produktionstekniske forudsætninger at styre produktionen, således at et fastlagt serviceniveau kan opfyldes til lavest mulig omkostning (inklusiv kapitalbinding). Dette opnås ikke ved at efterstræbe de lavest mulige produktionsomkostninger alene, men via en optimal kombination af serviceniveau, produktionsomkostninger og kapitalbinding.

Produktionsstyringen er en del af produktionssystemet. Produktionsstyringen omfatter mere end blot de fysiske aktiviteter såsom direkte bearbejdning eller montering af produkter. Den skal sikre en optimering af de ressourcer der skal samarbejde, eksempelvis maskiner til bearbejdning og montering, materiale og personale med uddannede arbejdsevner.

3.4 Datamodeller
I det følgende beskrives virksomhedens operationelle database.

3.3.1. Definition af Entitets-Relations (ER) modeller
Der er to relevante modelleringsteknikker i et data warehouse miljø: ER modellering og dimensionel modellering.

ER modellering producerer en datamodel indenfor det specifikke domæne ved hjælp af to basale koncepter: entiteter og og relationer mellem disse entiteter. Detaljerede ER modeller indeholder også attributter, som kan være egenskaber ved entiteter eller deres relationer.
ER Modellen er et abstraktions værktøj, der kan anvendes til at forstå og simplificere data relationer i virksomhedens verden og komplekse system miljø.

ER Modellering
I det følgende defineres de basale termer i ER modellering.

Grundlæggende koncepter
En ER model er repræsenteret ved et ER diagram, der anvender tre grundlæggende grafiske symboler til at repræsentere data: entitet, relation og attribut.

Entitet
En entitet er defineret som en person, et sted, ting eller begivenhed, der er af interesse for virksomheden. En entitet repræsenterer en klasse af objekter, der er ting i den virkelige verden, som kan observeres og klassificeres ved deres egenskaber og karakteristika. Eksempler på entiteter: produkt, produktmodel, komponent, kunde.

Relationer
En relation er repræsenteret med linier tegnet mellem entiteter. Det viser den strukturelle interaktion og association mellem entiteter i en model. En relation indeholder en tekstuel betegnelse som fx.:

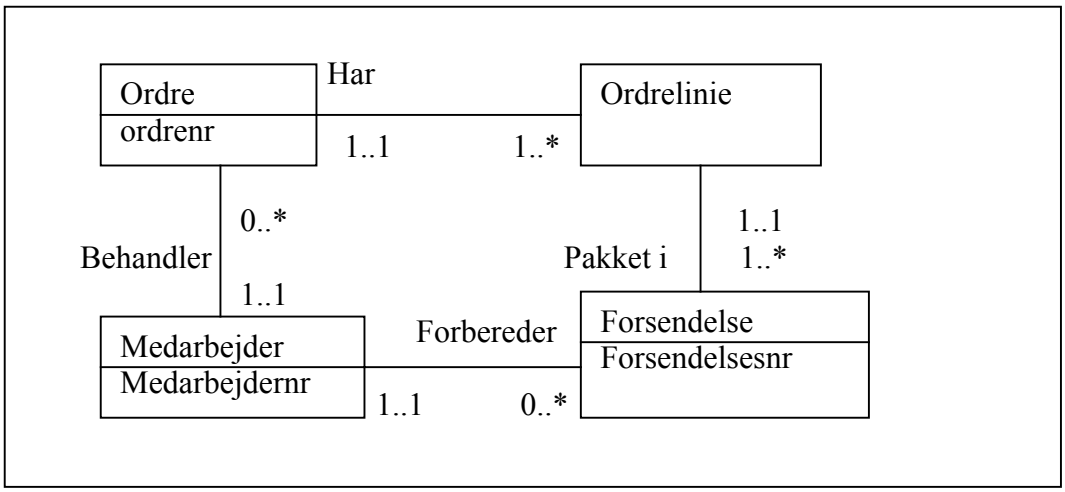
ejer, tilhører, har. Relationen mellem to entiteter kan defineres ved deres kardinalitet. Det er det maksimale antal instanser af en entitet der er relateret til en enkelt instans i en anden tabel og omvendt.

De mulige kardinaliteter er: en-til-en (1:1), en-til-mange (1:M) og mange-til-mange (M:M).

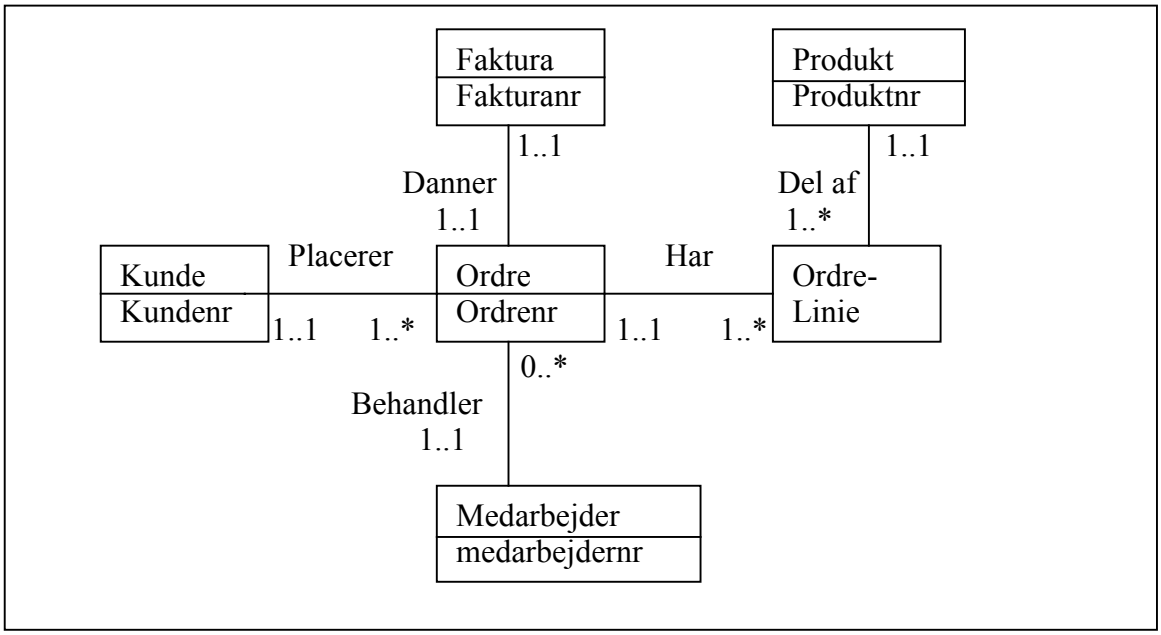
Attributter
Attributter beskriver de karakteristiske egenskaber ved entiteter. I en entitet som produkt, kunne produktid, beskrivelse og pris for eksempel være attributter. Et attributnavn skal være unikt i en entitet og skal være selvforklarende.

CaseFirms datamodeller
Virksomheden har implementeret følgende datamodeller i deres operationelle systemer (OLTP):

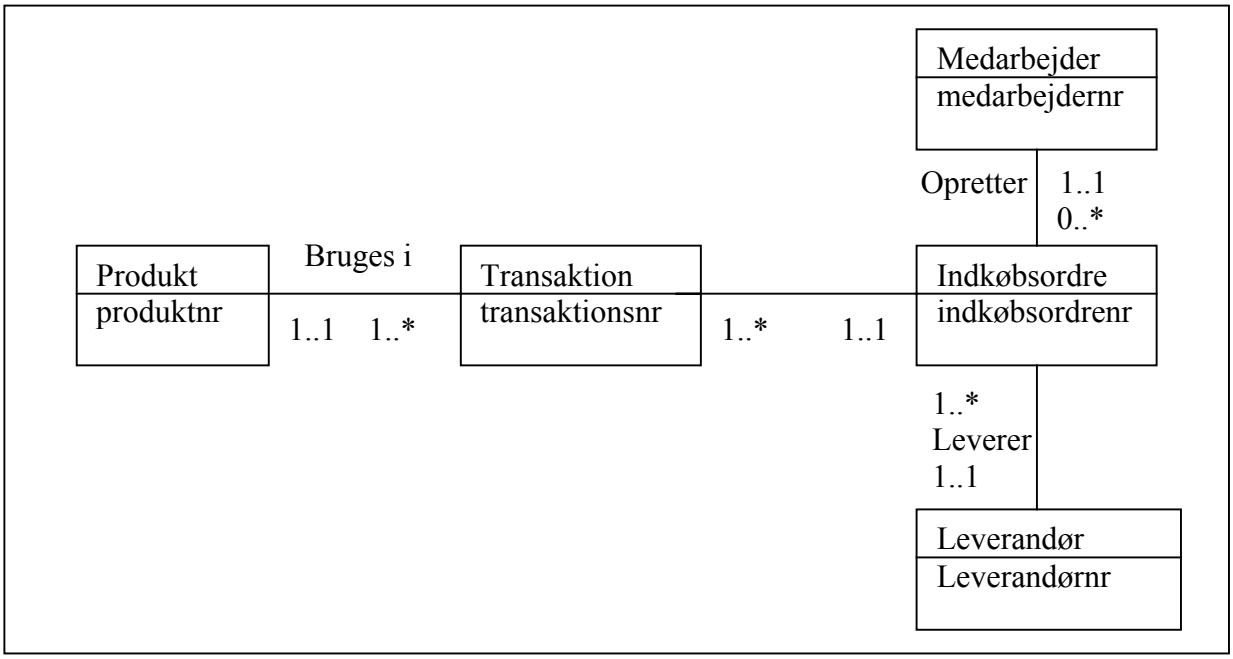
Lagersystemet:



Økonomi/Ordre:



Produktion:



Tabeller

Ordre
(ordrenr, ordredato, faktureringsvej, faktureringsby, faktureringspostnr, status, kundenr, medarbejdernr)
Primærnøgle: ordrenr, medarbejdernr

Medarbejder
(medarbejdernr, stillingsbetegnelse, fornavn, mellemnavn, efternavn, adresse, arbejdstelefon, hjemmetelefon, emailadresse, koen, gage, ansatdato)
Primærnøgle: medarbejdernr

Ordrelinie
(ordrenr, produktnr, antalenheder)
Primærnøgle: ordrenr, produktnr

Forsendelse
(forsendelsesnr, antal, forsendelsesdato, afslutdato, ordrenr, produktnr,medarbejdernr)
Primærnøgle: forsendelsesnr

Faktura
(fakturanr, datodannet, datobetalt, kreditkortnr, ordrenr)
Primærnøgle: fakturanr

Kunde
(kundenr, kundenavn, kundegade, kundeby, kundepostnr, kundetlf, kundefax, kundeemail)
Primærnøgle: kundenr

Produkt
(produktnr, produktnavn, serienr, enhedspris, antalpaalager, genbestillingsniveau, genbestillingsantal)
Primærnøgle: produktnr

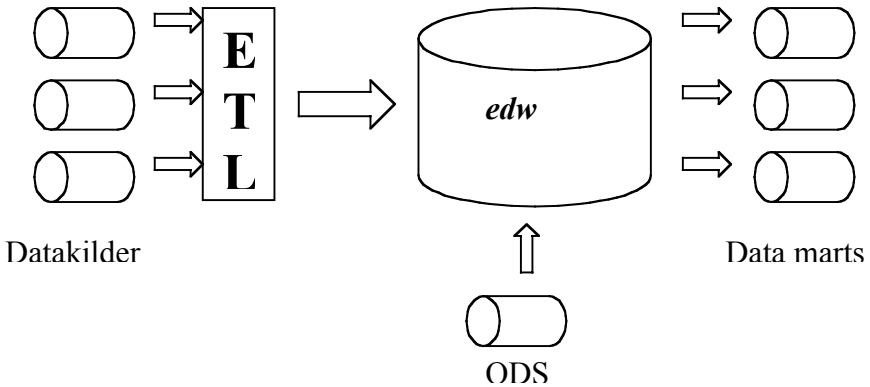
4. Data warehouse arkitektur

4.1. Arkitektur

4.1.1. Komponenter

Et data warehouse består af en række forskellige komponenter:

- Datakilder, der består af komponenter, der genererer data eksempelvis fra en proces. Disse datakilder kan være uafhængige af hinanden, således data i de enkelte datakilder ikke har forbindelse med de øvrige.
- ETL (Extraction, Transformation, Load). Dette programlag trækker data fra datakilder og flytter dem ind i et data warehouse. I dette lag foretages eksempelvis udregninger med data fra flere datakilder og dataene tilrettes efter nogle forud definerede kriterier. Det er også i dette lag metadata tilføjes.
- Selve midten af et data warehouse system er lagret, hvor de transformerede data gemmes. Denne del kaldes entreprise data warehouse (edw). Dette lag modtager kontinuerligt data fra ODS- og ETL-laget. I dette lag kan data ikke ændres. De kan kun indlægges, læses eller slettes.
- ODS-laget (operational data store) er en kombination af datakilde og edw. I dette lag kan data opdateres (som en datakilde), men data foreligger også som udregninger og integrerede data (som et edw).
- Data marts er det sted hvor de fleste brugere tilgår data warehouse systemet. Dette lag er struktureret efter funktioner eksempelvis salgsafdeling og produktion. Dette lag trækker data ud af edw, og er formet efter de krav de enkelte brugere af systemet stiller.



Figur 4.1. De forskellige lag i et data warehouse system.

4.1.2. Fordele ved datawarehouse teknologien

Nogle af de fordele der er ved data warehouse teknologien er at hvis man indlægger kriterier i ETL-laget, som udregner specielle værdier, vil disse værdier have referencer i systemet. De enkelte udregnede værdier vil ikke være der, men de data, der skal bruges til udregningen er tilgængelige og systemet ved hvor de befinder sig.
Det betyder at de samme data kan bruges til både rapportering og analyser.

Det betyder også, at der er hurtig adgang til dataene, da dataene ikke er udregnede værdier, men kun indekserede relationer samlet i emner. Det betyder, at når der rettes en forespørgsel til systemet, er det ikke nødvendigt at søge alle data igennem. Systemet kan finde de relevante data efter indekseringen.

4.1.3. Definition af data warehouse

Et data warehouse kan karakteriseres på følgende måde:

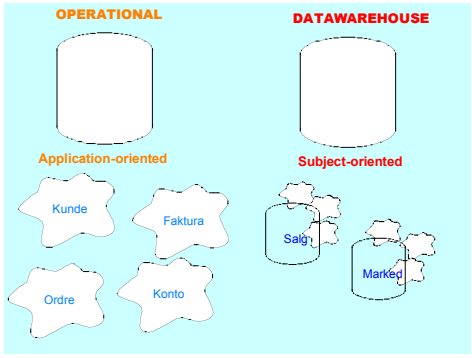
- emne orienteret
- integreret
- tidsdimension
- data er uforanderlige

Emne orienteret:
Data er ordnet i emner i stedet for funktioner.

Eksempler på emner:

- Kunde
- Produkt
- Sælger
- Transaktion
- Salg
- Marketing

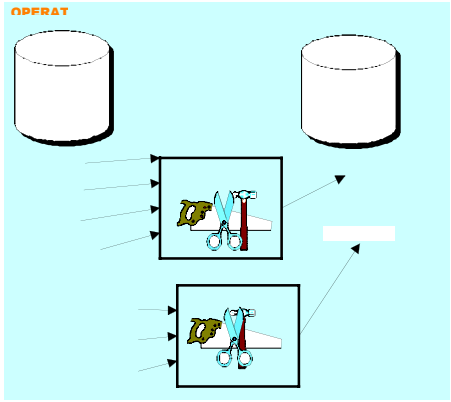
Enhver funktion har data, som relaterer sig til emner. For eksempel har regnskabsafdelingen relationer til kunde, faktura, ordre og konto.



Figur 4.2. Forskel på operationelle databaser (funktions ordnet) og data warehouse (emne ordnet) *(Lundbeck)*

I data warehouse modellen er det emnerne der har relationer til funktionerne.
Eksempelvis har kunde relationer til regnskabsafdelingen.

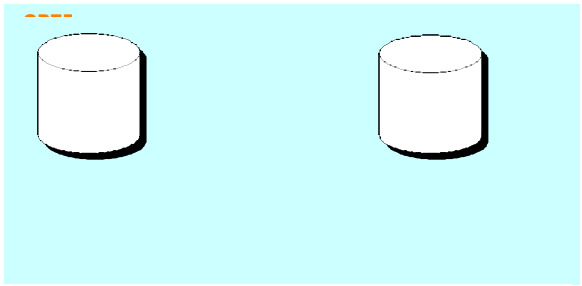
Integreret:
De enkelte data i et data warehouse er sammensat af data fra mange datakilder fra det operationelle miljø. Denne proces er nærmere beskrevet i afsnittet Transformation under ETL.



Figur 4.3. Forskel på operationelle databaser (enkelte data) og data warehouse (integrerede data) *(Lundbeck)*

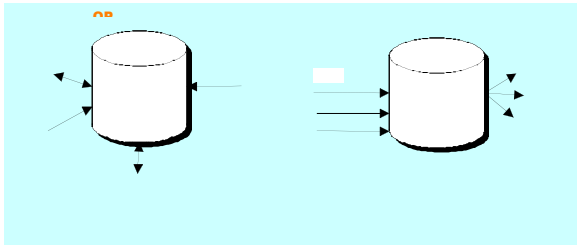
Tidsdimension:
De enkelte data i et data warehouse forbliver ens over tid. Det betyder, at man kan søge efter data som er lagret i et data warehouse for lang tid siden. Det skyldes, at data er lagret som samplinger af data i stedet for som rå data. Hver sampling er tidsstemplet og kan derfor ikke ændres.

I en operationel database ændrer dataene sig ofte. De bliver ændret eller måske helt slettet. Derfor kan man ikke søge efter gamle data i et operationelt miljø.



Figur 4.4 Forskel på operationelle databaser (ændrer sig over tid) og data warehouse (ændrer sig ikke over tid) *(Lundbeck)*

Data er uforanderlige:
Data i et data warehouse kan ikke ændres. Det skyldes, at alle data er tidsstemplet.
Hvis inputdataene ændrer sig, bliver der lagt en ny dataværdi ind i data warehouse.
Man kan kun skrive og læse i et data warehouse.
I en operationel database bliver de enkelte data ændret, når der sker ændringer i det operationelle miljø.



Figur 4.5. Forskel på operationelle databaser (data kan ændres) og data warehouse (data kan ikke ændres) *(Lundbeck)*

Ud over at data i et data warehouse er emneordnet, integrerede, tidstro og ikke til at ændre, skal de også være granulære.
Det vil sige, opdelt i så små mængder som muligt. Det drejer sig for eksempel om ordredata, hvor den mindste mængde er de enkelte ordrer.
Det har den fordel, at de kan indgå i andre sammenhænge end dem er var tiltænkt til. Se afsnittet om Data mining.

4.2. Datamodel

I lighed med andre typer databaser anvendes også en relationel database som f.eks Oracle, DB2 mfl. til lagring af de data der indgår i et data warehouse.

Data i et data warehouse adskiller sig på flere måder fra det operationelle datasystem og inden en datamodel for data warehouse præsenteres, foretages en kort sammenligning med et operationel datasystem. Nogle af de væsentligste forskelle fremgår af tabel 4.6.

Data warehouse	Operationelle datasystem
Lang tidsramme	Kort tidsramme
Statisk	Data kan ændres
Opsummeret	Acces til enkelte record
Ad hoc forespørgsler	Standard transaktioner
Opdateres periodisk	Real time opdatering
Data dreven	Event driven

Tabel 4.6 Sammenligning af data warehouse og det operationelle datasystem

I et data warehouse fastholdes data over en længere tidsramme ofte 8-10 år eller længere. Det ligger i konceptet at der kontinuerligt overføres tidsstemplede data fra de forskellige kilder uden at der tilsvarende slettes data. Det betyder, at datalageret for et data warehouse vokser med tiden. I det operationelle datasystem hvor der er behov for en hurtig adgang til data vil der med mellemrum blive foretaget en oprydning i ældre uaktuelle data. I økonomisystemet fastsætter regnskabsloven dog at alle transaktioner først kan slettes efter 5 år.

I et data warehouse forbliver de indlæste data uændret, jf. afsnit om arkitektur. I det operationelle system kan data ændres, f.eks. hvis et produkt ændrer navn eller ved opdatering af prisændringer. Der er her mulighed for både at oprette, ændre og slette data.

I et data warehouse kan der i forbindelse med ETL være foretaget en form for opsummering af data. Denne kan foretages mere eller mindre fin masket f.eks. til dagligt salg af vare til hver enkelt kunde eller mere grov masket ved opsummering af salg til hver enkelt kunde per uge eller måned. I det operationelle system kan hver varelinie for hvert salg til de enkelte kunder genfindes indtil de slettes.

I et data warehouse foretages brugernes adgang til data via ad hoc forespørgsler. Det vil ofte være tilpassede forespørgsler der opfylder behov for rapportering rundt i virksomheden. I det operationelle system er der i høj grad opbygget formularer til at klare de standard transaktioner der er i forbindelse med virksomhedens drift, f.eks. ordremodtagelse, fakturering mm.

I et data warehouse foretages en periodisk overførsel af data via ETL. Cyklusen for denne opdatering bør afspejle virksomhedens behov for at kunne opfange ændringer i den omverden som virksomheden indgår i. Hvis denne er meget omskiftelig skal der ofte opdateres hvori mod med en stabil omverden kan opdateringen foretages med længere mellemrum. I den operationelle datasystem sker oprettelse, redigering eller sletning af data i ”real time” det vil sige med øjeblikkelig effektuering af alle ændringer.

Et data warehouse er data dreven i sin opbygning. Den opbygges således at sammenhængende data også ligger samlet i databasen for at imødekomme behov for udtræk af data. Det operationelle system opbygges til hurtig af effektuere de events som forekommer i driften af virksomheden, f.eks. ordrebehandling, fakturering mm.

For at kunne i mødekomme kravene til data warehouse benyttes dels en normaliseret relationsdatabase med tidsstemplede data (historiske transaktioner) i datalageret og dels stjerne diagrammer med opsummerede data til de afdelingsbaserede data marts.

4.3. Normaliseret del

Datamodellen for den normaliserede del af data warehose bygger på samme model som for andre typer af operationelle databaser. På baggrund af en klassificering og komposition opbygges en taksonomi af tabeller med ER-modellering af de data der er udvalgt til at indgå i data warehouse.

I den normaliserede del (edw) ligger dataene i den detaljeringsgrad der er valgt i forbindelse med indlæsning af data til data warehouse. Disse data placeres i normaliserede tabeller for at opnå en effektiv database uden redundans samt for at begrænse brugen af null-værdier. For at bringe databasen på minimum tredje normalform (NF) skal følgende være opfyldt for alle tabeller:

1.NF. Hvis der er linier i tabellen, som indeholder samme primærnøgleværdi, og felter, der indeholder gentagne data på disse linier, så skal disse felter placeres i en ny tabel sammen med en kopi af primærnøglen. Desuden skal alle værdier i tabellen være atomare, dvs. de må ikke kunne opdeles yderligere.

2. NF. Tabellen skal være på 1. NF. Herudover gælder det, at hvis der i en tabel med en sammensat nøgle er felter, der er direkte afhængige af en del af primærnøglen, skal disse felter flyttes over i en anden tabel sammen med en kopi af delprimærnøglen.

3. NF. Tabellen skal være på både 1. og 2. NF. Herudover gælder det, at hvis der er felter i tabellen, som er direkte afhængige af et felt (determinanten), som ikke indgår i primærnøglen, skal disse felter flyttes over i en anden tabel sammen med en kopi af determinanten.

Der findes flere normalformer, men de 3 første vil opfylde behovet for en effektiv database.

4.4. Stjerneskema

Stjerneskemaer er et koncept med stor styrke i forbindelse med udtræk af data fra data warehouse. Ideen bag design af stjerneskemaer er at foretage en forbehandling af data således at de hurtig kan anvendes af brugerne. Navnet er fremkommet på baggrund af en datamodel med en stor central tabel omgivet at underordnede tabeller i en stjerne form. Der foretages en klassificering således at alle data i et stjerneskema vedrører et enkelt emner, f.eks. varesalg. Det betyder, at brugerne ikke skal foretage en SQL-forespørgsel i de store datamængde der ligger i data warehouse med deraf følgende ventetid mens forespørgslen blev afviklet. Der kan skabes en langt hurtigere adgang til de mindre stjerneskemaer, som indeholder langt færre data.

Stjerneskemaer indeholder mere eller mindre redundante data, som accepteres for at opnå en hurtigere tilgang til dataerne. Data marts er opbygget på grundlag af stjerneskemaer.

Den centrale tabel benævnes ”fact table”. Den første del af fact tabellen indeholder nøgler til dimensionstabellerne og den anden del indeholder kalkulerede værdier af data fra den normaliserede del af data warehouse. Data i fact tabellen er normalt numeriske og ofte additive, dvs. består af opsummerede værdier.

De omkringliggende dimensionstabeller indeholder den tekstmæssige beskrivelse til værdierne i fact tabellen. Dimensionstabeller kan indeholder op til mange attributter med følgende karateristik:

- tekst
- egenskaber der adskiller (discrete)
- definere constraints
- giver kolonne overskrift til analyser

Attributterne i dimensionstabellerne har i høj grad værdi i forbindelse med analyse af data fordi disse medvirker til at beskrive og sætte data ind i en kontekst.

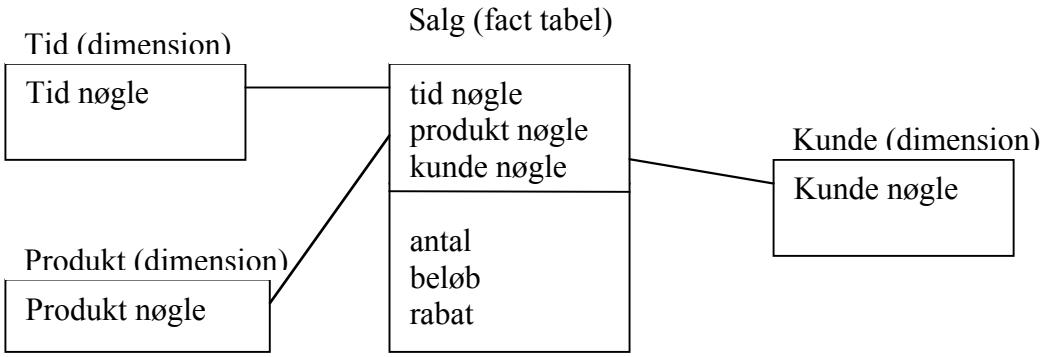
Hver dimensionstabel har en primær nøgle, som ofte er en numerisk værdi der genereres automatisk af systemet i forbindelse med oprettelse af en ny rekord.

Et eksempel på tre dimensionstabeller indeholder f.eks. følgende attributter:

TID	PRODUKT	KUNDE
Tid nøgle	Produkt nøgle	Kunde nøgle
Dag	Operationel Id	Navn
Måned	Produkt navn	Gade
År	Produkt klasse	Husnr.
Kvartal	Farve	By
Finansår	Størrelse	Land

Tabel 4.7 Eksempel på dimensionstabeller

Sammensættes de viste dimensionstabeller med fact tabellen opstår stjerne formen.



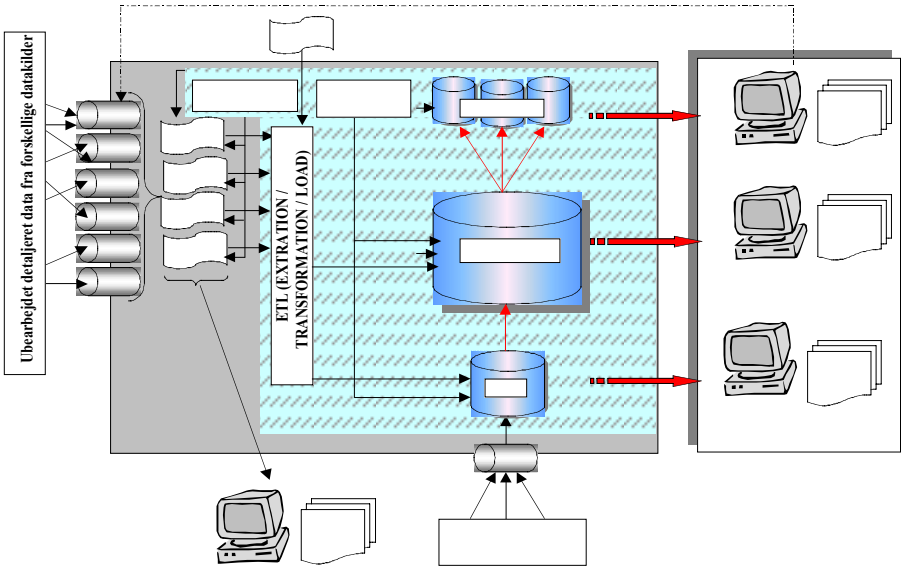
Figur 4.8 Stjerneskema for salg

Der opbygges således flere stjernetabeller på grundlag af de behov for analyse af data som er fremkommet fra de enkelte afdelinger forud for oprettelse af et data warehouse. I den forbindelse skal behovet også klarlægges for hvor detaljeret opløsning af de opsummerede data skal være. Detaljeringsgraden af data i stjerneskemaet skal afklares i forhold til hvilke analyser der skal foretages, hvad er den højst acceptable opløsning og hvor meget lagerplads er der til rådighed.

Stjerneskemaer skal jævnlig opdateres med de seneste data, der er overført til den normaliserede del af data warehouse. Opdatering foretages med SQL forespørgsler, der tilpasses hver enkelt stjerneskema.

5. ETL

For at få data ind i et data warehouse skal de både udtrækkes (*extraction*) fra de enkelte datakilder, omformateres til en fælles form og muligvis kombineres med andre data fra andre datakilder (*transformation*) og indlæses i data warehouse (*load*).
Der findes forskellige teknikker til de enkelte faser. Nogle af disse teknikker vil blive gennemgået her i dette afsnit med deres tilhørende fordele og ulemper.



Figur 5.1. Oversigt over strukturen i et system med datakilder, ETL, DW og data marts (Lundbeck)

5.1. Extraction

Da datakilder ofte opsamler data fra en eller anden forudbestemt proces, er det som regel ikke hensigtsmæssigt, at det er datakilderne som skubber (push) data ned i et data warehouse. Det er ofte mere hensigtsmæssigt, at det er et data warehouse, som trækker (pull) data ud af en datakilde. Det skyldes blandt andet, at når der er mange forskellige datakilder, som leverer data til et data warehouse skal man have styr på hvilke data der bliver læst ind hvornår, og gennem hvilket filter.

Langt fra alle operationelle data trækkes ind i et data warehouse. Det gælder kun for de data, som opfylder nogle stillede krav. Eksempelvis skal data sammenlignes med mange andre operationelle data i andre filsystemer eksempelvis i forhold til datastruktur og logiske sammenhænge for at komme i betragtning som data i data warehouse.
Når data skal lægges ind i et data warehouse, bliver de ofte omformateret. Det kan være så simpelt som at ændre datokoden fra dd.mm.åå til åå.mm.dd. Ofte bliver omformateringen dog meget mere kompleks.

5.1.1. Hvordan får man data fra det produktive miljø ind i et data warehouse?

I extraction laget er der et midlertidigt lager, der kaldes ”legacy” (~arv). Dette lager modtager (~arver) kontinuerligt data fra datakilderne og det er herfra data til transformationsprocessen tages.

Umiddelbart ville der være fornuft i at læse filerne med data i legacy miljøet, da alle data er tilgængelige og umiddelbart kan lægges ind i data warehouse. Det viser sig dog, at det ikke er så fornuftigt alligevel. Data ændrer sig stadig over tid i legacy miljøet, da der kontinuerligt lægges data fra det produktive miljø. Det betyder, at alle data skal læses for at holde øje med om noget ændrer sig og hvad disse ændringer betyder for dataenes videre vej til data warehouse.
Det stiller derfor store krav til maskinkraft at læse data fra legacy miljøet direkte til data warehouse. Derfor er det generelt ikke en løsning som anvendes.
I stedet kan man anvende data, som ikke ændres mere. Det kan eksempelvis ske i form af anvendelse af backup. I de fleste firmaer er backup noget som kører automatisk og der er styr på hvad der backes op og hvor ofte.
Der er flere fordele ved at bruge backup data som input til data warehouse:

- Man kan nøjes med at søge gennem de data, der er ændret. Det betyder man ikke behøver at søge gennem alle de data, som ikke er ændret.
- Man kan lave hele søge- og transformationsprocessen off line. Det betyder, at man ikke bruger dyr maskinkraft fra online systemet.
- Backup data er billige at få fat i, da backup alligevel skal laves hvad enten de skal bruges til data warehouse eller ej.

Der er dog også nogle ulemper ved at bruge backup data. Én af dem er, at backup fortrinsvis er beregnet til genopretning af databaser. Det betyder, at det kræver noget bearbejdning og tolkning af datastrukturen for at den kan bruges som input til et data warehouse.
Generelt set er backup det bedste sted, at få input til et data warehouse.

Hvis man vælger at bruge backup data som input til et data warehouse, vil dataene i det produktive miljø og data warehouse miljøet blive forskudt omkring et døgn. Det betyder, at dataene ikke er synkrone i forhold til hinanden.
Hvis man vil have det til at køre mere synkront, vil det give nogle af de ulemper, som nævnt ovenfor, såsom lang processortid, da alle data skal læses flere gange for at holde øje med ændringer.

Jo kortere tidsforskydning man ønsker, jo dyrere bliver det med maskinkraft og vedligeholdelse. Derfor vælger mange firmaer i praksis, at acceptere en tidsforskydning på over et døgn med de fordele (~ penge) og ulemper (~ tid) det giver.

Der er to teknikker som ofte anvendes til at finde data der skal lægges ind i et data warehouse; ”data replikation” og ”cdc (changed data capture)”.

Replikation:
Denne teknik går ud på, at tage en kopi af dataene og lægge dem ind i data warehouse. Det betyder, at man skal identificere dataene, som skal overføres til data warehouse inden overførslen for ikke at overføre alle data. Det gøres ved at sætte en trigger, som automatisk fanger data, når visse betingelser er opfyldt. Denne trigger lægges ind manuelt af administratoren af data warehouse, som også definerer betingelserne. Når dataene er fanget overføres de til data warehouse.
En fordel ved replikation er at kun de data, som er defineret i triggeren til at blive fanget, bliver det. Alle andre data springes over.
En anden fordel er, at dataene er veldokumenterede og formatet er veldefineret og at dataene kan lægges ind i data warehouse straks, det vil sige der er ikke ret stor tidsforskydning mellem ”datafangst” og indlægning i data warehouse.

En ulempe ved at benytte replikation til dataopsamling er, at der kræves maskinkraft og ekstra I/O i legacy miljøet. Der er forskel på hvor mange I/O der skal bruges i de enkelte transaktioner, så man kan ikke blot afsætte en vis maskinkraft og antal I/O til opgaven. Der oftest ikke overskud af nogle af delene i legacy miljøet.
En anden ulempe er, at triggeren skal vedligeholdes manuelt, da der hele tiden sker ændringer i dataene.
Endnu en ulempe er den hurtige indlægning af data. Det gør, at dataene ikke er blevet stabile og derfor stadig kan ændre sig.
Den væsentligste ulempe ved at bruge replikation er dog, at det ikke er muligt at foretage en grundig transformationsproces på grund af den hurtige indlægning af data.
Det betyder, at de data der indlægges i et data warehouse stort set er de samme som der er i legacy miljøet, det vil sige alle data bliver lagt ind som rå data uden transformation. Meningen med at lave et data warehouse er, at dataene skal transformeres og integreres. Derfor er det en meget dårlig ide, at bruge replikation til at hente data fra det produktive miljø til data warehouse.

CDC (Changed data capture):
Denne teknik benytter sig af de logfiler, der hører med til en backupprocedure. Disse logfiler indeholder oplysninger om hvilke data der er ændret og dermed taget backup af, og hvad ændringen består i.
En fordel er, at transformationen kan laves off line, og der derfor ikke skal laves roll back, dvs. ændringer fra data warehouse tilbage til legacy miljøet. Det skyldes, at dataene ikke ændrer sig og er stabile, da det er logfiler fra backup båndene. Det er også tid til at lave transformationsprocessen, inden dataene lægges ind i data warehouse. Der skal ikke indlægges en trigger manuelt, da man kan bruge oplysningerne i logfilerne til at finde ud af hvilke data der er ændret og hvilke der er relevante at trække ud til videre behandling.

Der er dog også en række ulemper ved denne teknik:
Båndet indeholder en masse data, der ikke skal lægges ind i data warehouse, men som er nødvendige for backup. Det vil sige der skal læses en masse unødig data for at finde det, der er relevant.
Formatet på de data der er på båndene er ofte af et meget andet format end det dataene skal indlæses i data warehouse med, da det kun er logfilerne og ikke selve datafilerne man bruger i denne teknik.
Logfilerne indeholder ofte kun adresser i stedet for dataværdier og disse adresser kan have en meget kompleks indbyrdes sammenhæng. Det vil sige det kan være svært, at tolke de data der er i logfilerne.
Det skyldes også, at dataene i logfilerne indeholder oplysninger om de konkrete formater i de forskellige databaser. Det gør, at hvis der indhentes data fra flere datakilder med forskellige formater, giver det komplekse systemer til at transformere data.

Den metode der anvendes oftest til at hente data til et data warehouse er CDC, da den har de fleste fordele. Det drejer sig specielt om, at der er muligt at tilføje metadata og lave en grundig transformation.
Hvis dataene blot lægges ind i et data warehouse uden nogen form for transformation, som nævnt under replikation, er det unødvendigt med et data warehouse. Så kan man lige så godt læse fra de enkelte databaser som bruges til datakilder.

5.2. Transformation

En traditionel database indeholder separate data, mens et data warehouse indeholder integrerede data. Det vil sige, at de enkelte data kan indeholde mange informationer.
Når man skal lægge data ind i et data warehouse er det vigtigt, at det sker efter nogle kendte standarder. Det skyldes, at man skal have en fælles betydning af de forskellige begreber, f.eks. ”Kunde” eller ”CaseFirm skrue nr. 7”.
Hvis man ikke har en fælles begrebsforståelse, er det ikke muligt senere at identificere de enkelte data.
For at lave denne fælles begrebsforståelse, skal alle data, som lægges ind i datawarehous’et, igennem et ”transformations filter”.
I dette filter bliver dataene tildelt navne efter den fælles begrebsforståelse og de bliver delt eller samlet og summeret alt efter hvordan filtret er lavet og hvad de skal bruges til senere.
Nogle af de ting, som transformationslaget tager sig af er håndtering af følgende:

- Forskellige applikationer, da de enkelte data i et data warehouse kan indeholde data fra forskellige applikationer.
- Forskellige værktøjer, eksempelvis Oracle eller Informix.
- Forskellige strukturer, eksempelvis hierarkisk og relationel struktur.
- Forskellige operativsystemer, eksempelvis Windows og UNIX.
- Forskellige lagringssprog, eksempelvis EBCDIC og ASCII.
- Forskellige enheder, eksempelvis stk., pakker eller paller skruer/søm, eller præfikser.
- Detaljeringsgraden af dataene, eksempelvis vil data fra en datakilde være mere detaljeret end data i et data warehouse, der ofte vil være opsummerede.
- Oversættelse af data, eksempelvis udregning af gennemsnit fra forskellige datakilder.
- Konvertering af interne datastrukturer, eksempelvis rækkefølge af de enkelte data.
- Udvælgelse af data, da ikke alle kildedata skal gemmes i et data warehouse.
- Sammenhængen mellem datakilder, eksempelvis skal der data fra forskellige datakilder til at lave et datasæt, der skal gemmes i et data warehouse.
- Generelt flow af data ind i et data warehouse, da der skal holdes styr på hvilke data der er i et data warehouse.
- Tidsmærkning af data, da det kan være en vigtig parameter, hvornår kildedata er blevet produceret og hvornår data er indlæst i et data warehouse og evt. også hvornår data ikke er relevante mere.
- Processortid for de(n) maskine(er), som skal lave transformationen, eksempelvis hvis det er meget store datamængden fra en produktion eller lignende der skal transformeres til et data warehouse.
- Produktion af metadata. Der er store muligheder for at producere metadata ud fra de processor som sker under transformationen. Disse metadata kan være meget anvendelige senere til analyseformål.

Som det ses af ovenstående liste, er det et stort og omfattende program der skal til at håndtere alle disse ting. Dette filter bruger op til 80 % af de ressourcer som bruges til udvikling af et data warehouse. (Kilde: www.billimnon.com)
Det betyder, at der skal lægges meget arbejde i at udvikle disse værktøjer. Der findes værktøjer, som automatisk kan generere disse programmer. Disse værktøjer har nogle fordele, som manuelt genererede programmer ikke har. Eksempelvis skal der laves mange transformationsprogrammer til de enkelte datakilder, hvis de er af forskellige typer.

Hver gang der eksempelvis bliver lavet ændringer i datastrukturer eller bliver tilføjet en ekstra datakilde, skal programmet rettes til (=programmeres om). Det vil sige, at den manuelle vedligeholdelse af programmet bliver svær og dyr.
Generelt anbefales det, at anvende disse værktøjer. Kun når der er tale om små prototyper og lignende bør man anvende manuelt genererede programmer til håndtering af transformation af data.
Et data warehouse er en iterativ proces. Det vil sige, at det udvikler sig hele tiden, fordi der hele tiden bliver læst nye data ind i et data warehouse.
En stor del af de metadata, som lægges i et data warehouse kommer af sig selv under transformationsprocessen.
Det gælder for eksempel hvilke datakilder de enkelte dataelementer kommer fra, hvilke dataelementer der bruges som input og hvilket output det resulterer i efter transformationsprocessen, som også kan gemmes.
Udover de ovenfor nævnte, er der en række fordele ved at danne metadata under transformationsprocessen. Metadataene bliver holdt opdaterede, da hele indlæsningen foregår automatisk.

En af de mest komplekse operationer under transformationsprocessen, er når flere datakilder skal levere data til et datasæt til et data warehouse. Det betyder at et enkelt datasæt kan være afhængig af flere datakilder. Udover selve dataværdierne, kan også rækkefølgen af dataene kan have betydning. Det betyder, at der skal være en form for "buffer", der kan gemme nogle værdier indtil de skal bruges til beregning af nye dataværdier til data warehouse. Hvis én af datakilderne ikke leverer de data, der forventes til det tidspunkt der forventes, vil det kunne sætte transformationsprocessen i stå. Det betyder også , at der skal være en timeout funktion, som enten sletter eller går videre i datamængden, således én defekt datakilde ikke kan sætte transformationsprocessen i stå.

Et andet problem kan være, at det kræver meget maskinkraft, at skanne gennem store mængder data for at finde de data man leder efter. Det kan være man skal skanne flere gange, for at de rigtige data er tilgængelige fra processen.

Et tredje problem kan være midlertidige data. Under transformationsprocessen kan der dannes midlertidige data, for eksempel et antal af en bestemt type skruer i et afgrænset tidsrum. Hvis transformationsprocessen senere forsøger, at få fat i den type data vil det ikke lykkes, da de kun blev lagret midlertidigt. Denne mangel på data skal programmet også kunne håndtere.

De data, som kan lægges ind i et data warehouse behøver ikke kun være dagligt producerede data. Det kan også være arkiverede data, hvis det er aktuelt. Det kræver en del manuelt arbejde, men da det kun skal laves én gang, er det en overskuelig opgave. Det kan meget vel være en meget stor opgave, men den er kendt og afgrænset. Man skal dog være opmærksom på, at dataene skal være så veldokumenterede, at man kan producere de nødvendige metadata.
Dagligt producerede data er nemme at få fat i, men da det skal foregå hele tiden, er det den del af dataene som man skal tage mest hensyn til under planlægningsfasen med hensyn til maskinkraft.

5.2.1. Finpudsning

Når dataene er kommet på samme form og sprog og næsten er klar til at komme ind i data warehouse skal de finpudses. Dette gøres for at den senere brug af data bliver så hensigtsmæssig som muligt.

Det kan eksempelvis være noget med postnummeret og postdistrikt. Hvis postnummer og –distrikt ikke passer sammen, skal postnummeret ændres. Et andet eksempel er titlen på personen og rækkefølgen af navnet, hvis der står ”Peter” og ”Hansen” skal der stå ”Hr. Peter Hansen” og ikke ”Fr. Hansen Peter”. Der kan også være en stavekontrol af alle data.
Denne finpudsning af data kan være meget avanceret, og hvis den placeres i toppen af transformationsprocessen, vil det give en meget effektiv finpudsning af data inden de overføres til data warehouse.
Transformationen kan foregå to steder:

1. I serveren som håndterer legacy data.
2. I serveren som håndterer datavarehuset.

Der er fordele og ulemper ved begge dele.
Fordelene ved den første er eksempelvis, at alle nødvendige data er tilstede på det rette tidspunkt sammen med stor maskinkraft.
Ulemperne er eksempelvis, at selv om der er stor maskinkraft, er der ikke ret meget af den, som kan bruges til transformation. Hovedparten af maskinkraften bruges til dataopsamling. En anden ulempe er at det er kompliceret, at lave integrationsprocessen på legacy serveren, da dataene ændrer sig hele tiden.
En af fordelene ved at foretage transformationen på datavarehus serveren er eksempelvis, at maskinkraft ikke er en mangelvare.
En ulempe er, at data ikke altid er tilgængelige på det tidspunkt de skal bruges.

5.3. Load

Når transformationsprocessen er ovre og dataene er arrangeret i rækker med data læses dataene ind i data warehouse.
Selve indlæsningen af data foregår løbende når der ikke er anden aktivitet. Det vil sige, indlæsning af data i et data warehouse har lavere prioritet end læsning/udtræk af data.
Det betyder, hvis der er stor aktivitet i et data warehouse i løbet af en almindelig arbejdsdag, vil indlæsningen af data foregå efter normal arbejdstid.
Indlæsningsdataene lægges ind i mindre portioner (batches), når der er brug for det (pull).
Når dataene indlæses bliver de tidsstemplet, således man kan holde kontrol med, hvornår de er lagt ind i data warehouse.
En af de vigtigste funktioner i load processen er at holde styr på tidspunktet for indlæsningen, hvilken tabel dataene er placeret i og hvor meget dataene fylder. Disse oplysninger gemmes i en logfil.
Disse oplysninger kan brugerne søge i, således der ikke skal søges gennem alle data i data warehouse for at finde de data der søges.

6. Metadata

Metadata er data om data. Det vil sige de rådata der kommer fra eksempelvis en proces kan tilknyttes oplysninger om de pågældende data, således man kan se hvor dataene kommer fra og hvad de drejer sig om.
Man kan godt sammenligne metadata med vejskilte.
Hvis man kender den vej man skal, lægger man ikke mærke til skiltene. Det gælder for eksempel ens vej fra hjemmet til sin arbejdsplads. Den vej har man kørt flere hundrede gange. Hvis man derimod ikke kender vejen, lægger man mærke til hvad der står på vejskiltene og de er nødvendige for at finde den rigtige vej. Det gælder for eksempel hvis man er på ferie et nyt sted.
Det samme gælder for data: Hvis man ved hvilke data man skal have fat i, og hvor de er placeret, behøver man ikke metadata til at forklare noget om de forskellige data.
Hvis man derimod ikke ved hvor dataene er placeret og hvordan man finder dem, har man brug for oplysninger (~metadata) der fortæller noget om de enkelte data for at finde de rigtige.

Der er to måder at håndtere metadata på: centraliseret og distribueret.

6.1. Centraliseret

Den centraliserede arkitektur er som ordet siger et centralt sted, hvor alle metadata er defineret og gemt. Det gør det let at overskue og definere nye parametre, da definitionerne kun skal være et sted og alle metadata ligner hinanden i struktur og form.
Der er dog også en del ulemper ved denne form for lagring af metadata. Det drejer sig eksempelvis om, at metadata ikke er autonome og dermed ikke kan ændres af flere forskellige brugere på én gang. Det betyder også, at hele strukturen for metadata skal lægges fast inden man starter med at indlægge metadata, og alle metadata skal indlægges inden man kan læse nogle af metadataene. Det er også vanskeligt at opdatere metadata, når de først er indlagt.
Der er en variant som kaldes replikeret centraliseret arkitektur. Det går ud på, at når alle metadata er samlet centralt, kopieres de ud til de funktioner, som skal bruge dem. De personer kan så ændre i dataene, men det bliver ikke opdateret tilbage til den centrale database. Der er dermed ingen kontrol med de data, som kopieres til andre funktioner.
Alt i alt er det ikke en fleksibel metode, og den har større ulemper end den har fordele. Den er oprindeligt opstået da man kørte alt data warehouse på main frame maskiner. Da det er en centraliseret løsning, behøvede man ikke at tage hensyn til, at data kan være på forskellige maskiner i forskellige systemer.

6.2. Distribueret

Den distribuerede arkitektur siger næsten også med navnet hvad den går ud på. Metadatene lagres forskellige steder i systemet. Det kan være i forskellige data marts, i andre data warehouses m.v. Der er fuld kontrol over dataene i de enkelte systemer, således der både kan rettes, tilføjes og slettes data uden det påvirker metadata i de andre systemer. Dataene i de lokale systemer kan deles med andre systemer, som har brug for de pågældende data. Det registreres i en logfil hvilke forbindelser de enkelte systemer har med hinanden, således der altid er styr på hvor de enkelte metadata kommer fra og hvilke indbyrdes relationer de har.

Man kan vælge at gøre specifikke data ”lokalt autonome”. Det betyder, at de ikke kan deles med andre systemer. Det kan eksempelvis være en fordel hvis dataene er en del af en helhed som ikke er afsluttet, eller hvis man ikke ønsker at andre har adgang til dem af forskellige grunde. Den der ejer metadataene bestemmer selv om de skal gøres tilgængelige for andre systemer eller ej.
Når metadata deles med andre systemer, kan modtageren ikke ændre i dataene. Det er kun ejeren af metadata, der kan ændre i de pågældende data. Andre systemer kan frit analysere, gemme og sammenligne metadataene.
Da distribuerede metadata gemmes i almindelige tabeller som i relationelle databaser, kan de tilgås med de teknikker der gælder for denne form for database. Det drejer sig eksempelvis om API (Application Program Interface) og SQL (Structured Query Language), som kan udtrække data direkte fra databasen.
Det er oftest de distribuerede teknikker der anvendes, da dataene også er distribuerede.

6.3. Hvad bliver til metadata?

Der er to former for metadata; teknisk orienterede og brugerorienterede.
De teknisk orienterede metadata bruges fortrinsvis af det det teknisk administrative personale mens de brugerorienterede data fortrinsvis bruges af brugerne af systemet.
De teknisk orienterede kan eksempelvis være følgende:

- Beskrivelse af datakilden
- Beskrivelse og strukturer af tabeller
- Beskrivelse og definitioner af attributter
- Beskrivelse, definitioner og sammenhænge af primær- og fremmednøgler
- Brugsmønstre for data.

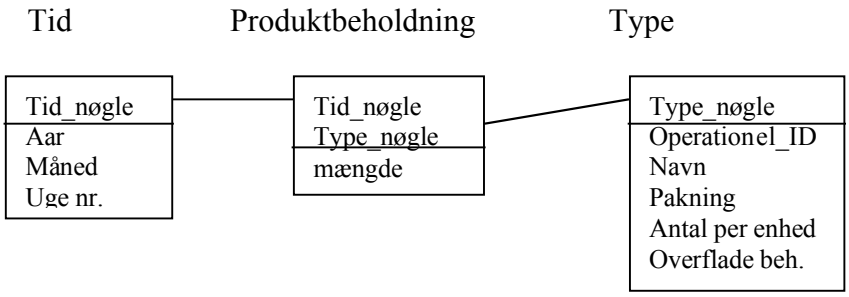
De brugerorienterede kan eksempelvis være følgende:

- Ejerskab af data
- Aliasing af data, dvs. flere forskellige navne for samme datatype.
- Definitioner og om disse er redigeret til afdelingen.

For at holde distribuerede metadata opdaterede, skal de synkroniseres. Det kan foregå automatisk efter en fastlagt plan. Systemet kontrollerer de enkelte kilder for om metadata er opdateret. Hvis de oprindelige metadata er opdaterede, opdateres de distribuerede metadata også. Hvis der ikke er sket ændringer i de oprindelige metadata, ændres de distribuerede metadata heller ikke.
Synkronisering af metadata kan kun foregå, når der sker ændringer i metadataene. Det vil sige, hvis der indlæses data i et data warehouse, som ikke ændres, er det ikke nødvendigt at synkronisere data.
En vigtig ting ved metadata er ejerskab. Der kan kun være én ejer af metadata. Alle systemer kan læse og bruge metadata, men ejeren er fuldt ansvarlig for dataene og vedkommende er den eneste der kan oprette, ændre og slette metadata. Det har blandt andet den fordel, at man altid ved hvor man skal henvende sig, hvis der er problemer med metadata, nemlig ejeren af de pågældende data.
Normalvis er ejeren ikke kun en person, men en hel organisation, eksempelvis produktionsafdelingen, regnskabsafdelingen eller salgsafdelingen.
Tabeller med metadata har specifikke navne. Det betyder, at de enkelte afdelinger ikke bare kan oprette tabeller med de navne de selv ønsker. Hvis et nyt ønsket navn findes i forvejen kan det ikke bruges til andre tabeller.

For at holde styr på historikken med metadata kan anvendes en teknik der kaldes ”versioning”. Det går ud på, at alle ændringer over tid af metadata registreres og lagres således man kan holde styr på hvilke metadata der hører til hvilke tidspunkter.
Det kan eksempelvis dreje sig om definitioner af metadata som ændrer sig. Det betyder, at metadata med samme værdier kan betyde forskellige ting, alt efter hvilken version af definitionen af metadata der var gældende, da dataene blev lagt ind.
En af de væsentligste fordele ved versioning er, at man kan genskabe historikken for de givne data. Det kan man have brug for, når der er sket mange ændringer af definitioner af metadata.
I CaseFirm kunne metadata for en OLAP for lagerbeholdningen indeholde:

Navn: Lager beholdning.
Indhold: Den aktuelle lagerbeholdning opgjort ultimo ugentlig baseret på en model for til- og afgang fra lageret for de seneste 4 år. Der foretages en status af lagerbeholdningen en gang årligt.
Struktur: Dataene er struktureret i en fact- og to dimensionstabeller.



Definition: Mængden af varer opgøres i antal pakninger. Typetabellen indeholder oplysninger om antallet af enheder per pakning.
Kilde: Alle data stammer fra den strukturerede del af data warehouse.
Navne: Det salgsmæssige navn er sammen med de operationelle nummer for produktet angivet i typetabellen.
Bearbejdning: Den aktuelle lagerbeholdning er for hver enkelt varetype opsummeret henholdsvis for hver uge, måned og år. Summen baseres på til- og afgang fra lageret.
Revision: Modellen er senest revideret 17. marts 2001 efter ønske fra lageret om at overfladebehandlingen indgår i oplysning om varen.
Opdatering foretages månedlig, senest den 4. hverdag i måneden.
Brugere: Lager og produktionen.

Figur 6.1. Eksempel på metadata

7. OLAP

Arkitekturen for data warehouse er nu beskrevet, men hvilke værktøjer er der til udnyttelse af dataene?

7.1. Definition

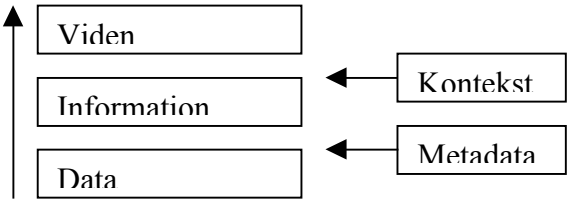
On**L**ine **A**nalytical **P**rocessing er et bruger interface koncept til præsentation af større datamængder. Dataene er opdelt i emner efter en oparbejdning og er placeret i en del af data warehouse konceptet, der benævnes data mart.

OLAP er et koncept til analyser af store datamængder. De kan være placeret i et data warehouse, men konceptet kan også anvendes med andre databaser som kilder. OLAP er således med til at danne bro mellem data warehouselaget og brugerlaget.

OLAP bliver ikke betragtet som et datalager teknologi, men i konceptet indgår en strukturering af dataene i emner i en dimensional database, f.eks. via stjerne skemaer i et data mart.

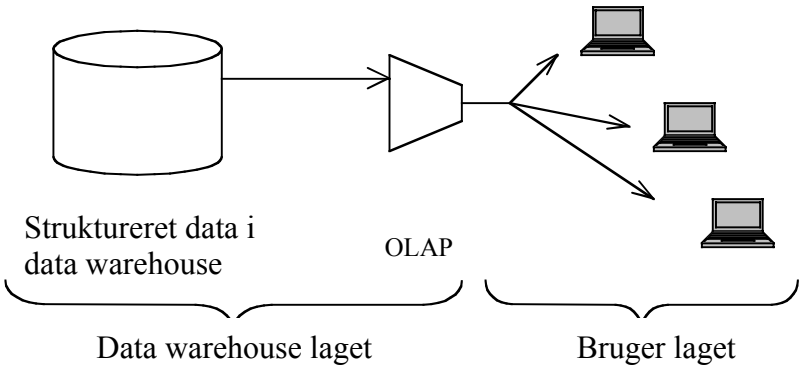
7.2. Fra data til information

OLAP anvendes hvor man skal gøre data til information. Informationerne opdeles i emner og giver brugerne afgang til at udfører forespørgsler i dataene.



Figur 7.1. Sammenhæng mellem data, information og viden

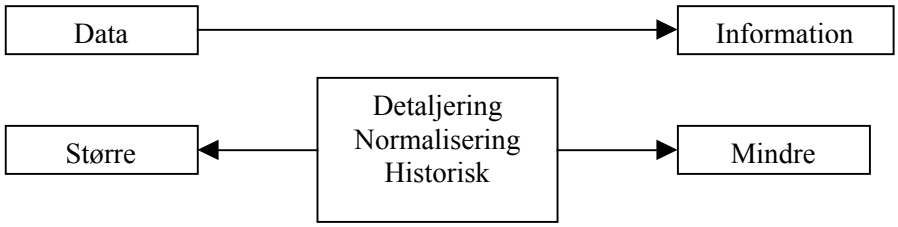
Der foretages en gruppering af dataene hvor de summeres op for flere dimensioner (clustering). Det kan f.eks. være salget af de enkelte varer der summeres over tid, lokalitet og sælger. Det kan også være information om antal fejl samt hvilke fejl der er tale om i producerede varer kombineret med forløbet tid siden maskinen sidst blev efterjusteret.



Figur 7.2. Sammenhæng mellem lagene.

Tilgængeligheden af informationerne er hurtigere end i de strukturerede rådata, idet datamængden bliver mindre og dermed kan der gennemføres en høj indeksering og at målrette informationerne til de enkelte afdelinger / brugere. Ved opbygning af systemet, der består af flere OLAP tabeller (data mart) udvælges de informationer, der er behov for i de enkelte afdelinger. Her får CaseFirm således et redskab til at kunne besvare spørgsmålene:

- hvornår, (er den solgt)?
- hvad (var varetypen)?
- hvor (blev den solgt)?
- hvem (solgte)?



Figur 7.3. Datastruktur ændres ved overførsel til OLAP.

Ydeevnen i form af hurtig ekspedition og servicering af mange samtidige brugere forbedres i OLAP ved at dataene samles på nye måder. De bevirker at der reduceres i detaljeringsgraden f.eks. ved et dagligt salg af mange enkelt varer summeres til et tal. Normalisering nedsættes således der bliver færre relationer mellem tabellerne og forespørgslerne afvikles derved hurtigere. Oplysninger går ikke så langt tilbage i tiden og mængden af data bliver derfor færre, f.eks. medtages kun data for de seneste 4 dage.

Der er flere synonymer for OLAP bl.a. data mart, lettere summeret eller afdelingsstruktureret. Data mart bruges som betegnelse for lagring af de data som OLAP anvender til forespørgsler.

Der er flere udbydere af standard software til OLAP, bl.a. Oracle, Cognos, Microstratrategy DSS Agent m.fl.

I forbindelse med en eventuel investering af et data warehouse i CaseFirm er der behov for OLAP software til beslutningsstøtte og til analyse af information.

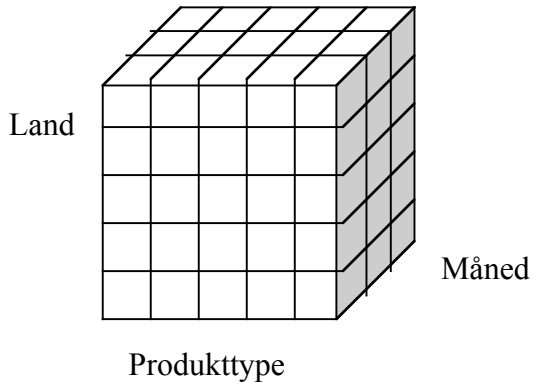
7.3. Funktioner

OLAP skal som minimum understøtte følgende funktioner:

- multidimensionel
- varieret detaljeringsgrad
- rotation
- flere visnings former

7.3.1. Multidimensional

Dataene samles således at der kan opbygges en flerdimensioneret tabel, hvor rådataene er opsummeret for udvalgte dimensioner.

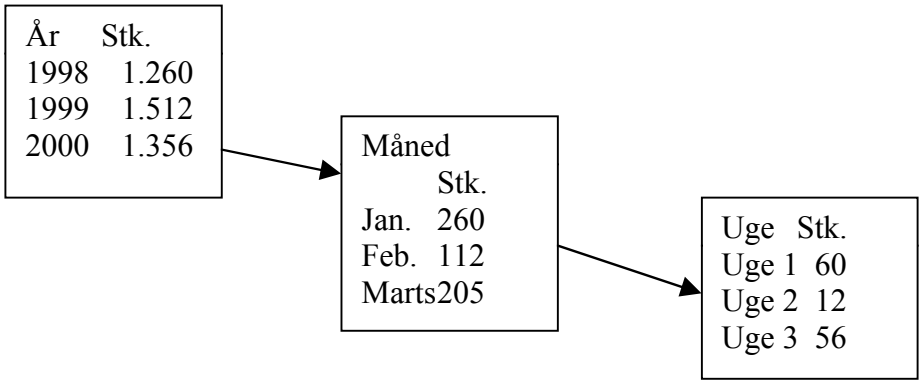


En opdeling i flere kategorier kan f.eks. være efter produkttype, land og hvornår er salget foretaget. Det kan illustreres ved at informationerne ligger i skæringspunkterne i en retvinklet prisme, som vist i figur 7.4.

Figur 7.4. Salg opdelt på type, tid og land.

7.3.2. Varieret detaljeringsgrad

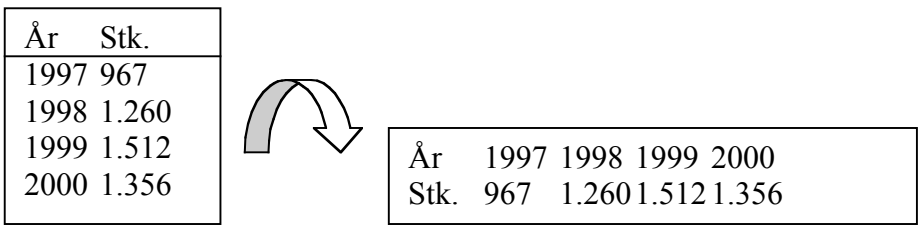
En funktion indeholder mulighed for at zoome ind på dataene, f.eks. at kunne ændre visning af varesalg per år til per måned, per uge og per dag.



Figur 7.5. Salget kan opløses i flere detaljeringsgrader.

7.3.3. Rotation

En af funktionerne er at kunne dreje informationerne med få klik hvis de ønskes vist horisontalt i stedet for i en vertikal tabel.

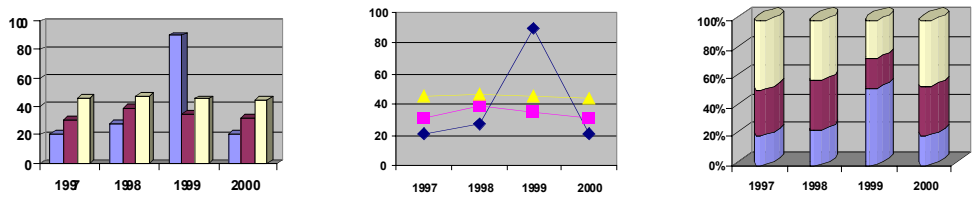


Figur 7.6. Oplysningerne kan drejes om x- og y-aksen.

En rotation giver ikke flere informationer. En fordel kan være at brugeren fanger andre sammenhænge ved at dreje skemaet, som ellers ikke ville træde frem.

7.3.4. Multipel præsentation

Der er mulighed for at skifte mellem præsentation i tabelform og flere grafiske figurer.



Figur 7.7. Præsentation af informationer grafisk.

Figuren til venstre er god til at sammenligninger hvor der både kan ses salget af hver produkt for de enkelte år og til dels udviklingen over flere år. Den midterste figur illustrerer det samme men er bedre til at følge udviklingen af hver enkelt produkt over tid. Figuren til højre viser den procentvise fordeling per år, men indeholder ingen information over den volumenmæssige udvikling over tid.

I CaseFirm kan den midterste figur f.eks. være velegnet til at følge salget af udvalgte varer over en periode på flere år samt til at illustrere om en variation i afsætningen er sæsonbetinget. Tilsvarende kan udviklingen i lagerbeholdningen illustreres over en periode.

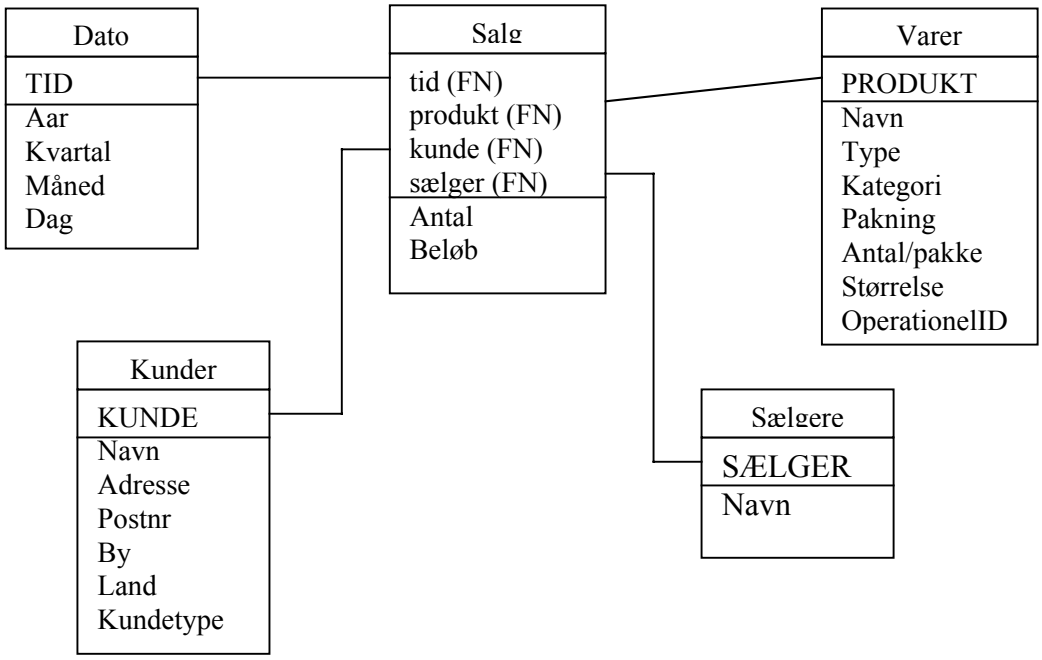
7.4. Datamodel

Datamodellen for OLAP bygger på dimensionelle databaser, der opbygges på grundlag af et stjerneskema med fact- og dimensionstabeller i et data mart. I facttabellen indgår den opsummerede værdi og i de tilhørende dimensionstabeller samles oplysningerne om de tilknyttede enheder til værdien.

I CaseFirm kunne et OLAP for salgsafdelingen over afsætning sammensættes som vist i figur 7.8. Facttabellen er kædet sammen via en fremmednøgle til hver dimension. Med en høj grad af indeksering opnås en høj ydeevne i forespørgslerne.

Hvilke data der skal udvælges og bearbejdningen af disse skal forinden afklares i samarbejde med salgsafdelingen. Det er en forudsætning for succes at informationerne opfylder et konkret behov i forbindelse med beslutningsstøtte i afdelingen.

I den viste model i figur 7.8. er der mulighed for dels at følge salget af de enkelte varer over tid og opdelt på henholdsvis kunde og lande. Tilsvarende kan omsætningen beskrives med samme opdeling.



Figur 7.8 Stjernediagram for varesalg.

Behovet for informationer til beslutningsstøtte skal gennemgås med hver enkelt afdeling forud for implementering af et data warehouse i CaseFirm. Efterhånden som afdelingerne lærer systemet at kende vil de givet få ønsker om at udbygge med flere OLAP over nye emner eller redigere de eksisterende indenfor de tilstedeværende data i de strukturerede data i data warehouse, og det kan også lade sig gøre.

7.5. ROLAP og MOLAP

ROLAP står for overførsel af data til OLAP fra en relationsdatabase. Dataene overføres til et stjerne skema med fact- og dimensionstabeller.

MOLAP er betegnelsen, hvis der er tale om overførsel af data til OLAP fra en multidimensional database. Ved MOLAP placeres de opsummerede data også i multidimensionale tabeller.

7.6. Opdateringsfrekvens

Brugerne har kun mulighed for at læse ”gamle” informationer i OLAP, og opdatering af data skal derfor fastlægges i forhold til behov og tekniske muligheder. Der skal forinden også være nyere data i data warehouse før det giver mening at opdatere. Den tid der går for at opdatere OLAP skal også sammenlignes med hvor ofte det skal foretages. En daglig opdatering er ikke hensigtsmæssigt hvis det tager 24 timer.

Detaljeringsgraden af de opsummerede data giver et aktuelt minimum for intervallet imellem hver opdatering. Hvis der opsummeres minimum for hver uge er der ingen grund til at overføre nye data med kortere mellemrum.

Behovet kan være forskelligt, således at nogle opdateres med kortere mellemrum end andre. Det skal nøje afklares med de enkelte afdelinger / brugere af OLAP hvor ofte de enkelte emner skal opdateres.

7.7. Fleksibel information

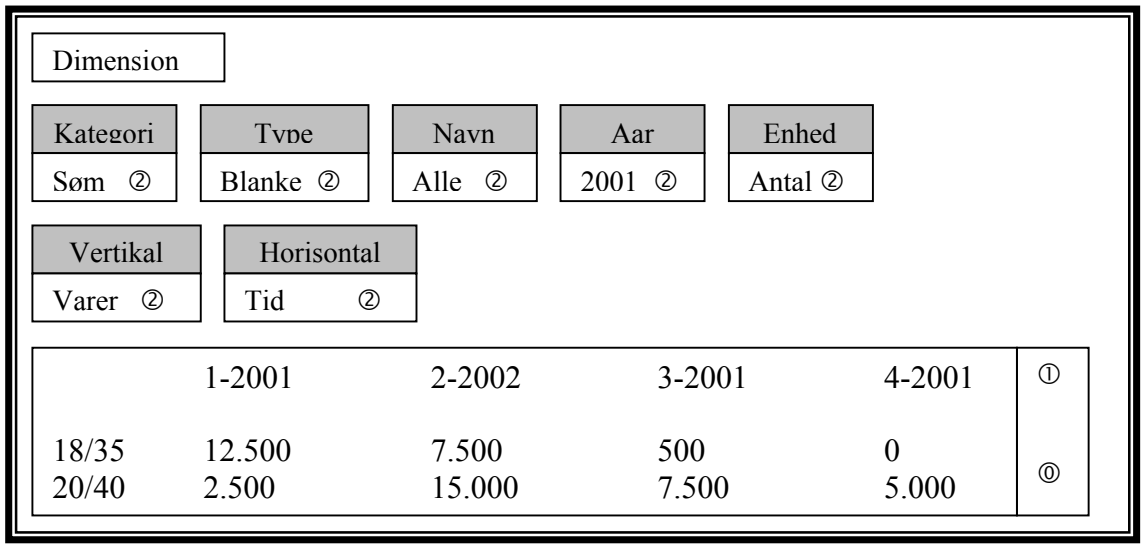
Med opsætning med flere OLAP emner tilpasset behovet for beslutningsstøtte i virksomheden og i de enkelte afdelinger kan OLAP imødekomme behovet for rapportering i CaseFirm.

Et system til støtte af styring af lageret kunne omfatte en samling af flere data fra flere afdelinger. Hvor der tidligere er gennemført en planlægning af produktionen for hver kvartal kan perioden nedsættes til f.eks. månedlig fordi resurseforbruget til fremskaffelse af informationer bliver minimeret.

Planlægning af den kommende periodes produktion involverer flere processer. Dels skal der skabes overblik over beholdningen på lageret og dels ligger der varer på lageret som er solgt men endnu ikke udleveret. Yderligere skal der indgår oplysninger om varer der ligger i pipelinen, og er under produktion, samt et estimat for salget i den kommende periode.

Salgsafdelingen kan levere et estimat for salget i den kommende periode på basis af en OLAP der analyserer afsætningen i den foregående periode kombineret med forventningerne til udviklingen i markedet.

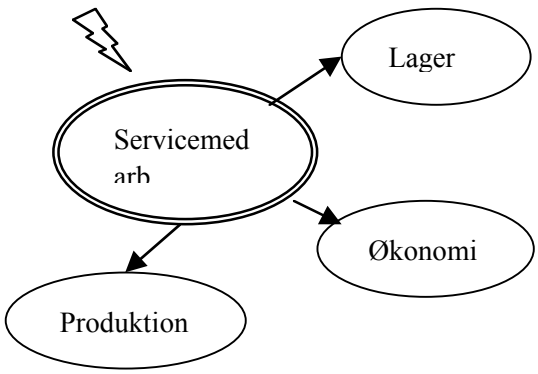
Den aktuelle lagerbeholdning kan oplyses med stor nøjagtighed. Ligesom det er kendt hvilke varer der er solgt, men endnu ikke udleveret samt hvilke varer der er under produktion herunder seriestørrelsen, og hvornår disse forventes at være tilgængelig for salg. Disse oplysninger kan samles og præsenteres for planlæggeren af produktionen via OLAP. Kombineret med rapport fra salgsafdelingen over et forventet salg kan disse danne grundlag for planlægning af månedens produktion af de forskellige typer søm og skruer.



Figur 7.9. Brugerinterface for varer på lager ved månedens udgang

Brugerinterfacet for aktuelle lagervarer kunne oprettes i stil med figur 7.9. Via kombinationsbokserne er det muligt at vælge kategori, type, varenavn, periode og enhed. Enten kan alle vælges eller der kan foretages en afgrænsning til en vare. Visning kan svitses mellem varer på vertikal liste og tid på horisontal led eller modsat.

I forbindelse med beslutningsstøtte der kræver løbende opdatering af informationer er OLAP ikke egnet. Der er en større eller mindre tidsforskydning i forbindelse med opdatering af data warehouse og OLAP, som betyder at modellen ikke er egnet til at holde servicemedarbejdere i salgsafdelingen opdateret med aktuelle informationer. Kontakter en kunde CaseFirm for at købe x antal vare y skal en servicemedarbejder kunne se informationer dels om der må sælges til kunden, er varen på lager og hvis ikke hvornår vil det være muligt at leverer. Det fordrer et system hvor informationerne er opdateret med de seneste ændringer, ligesom servicemedarbejderen ved gennemførelse af salget skal kunne lægge en ordre ind på kunden. Det kan OLAP ikke imødekomme.



Figur 7.10. Informationsbehov ved salg.

7.8. Datamining

Datamining kan foregå i såvel de strukturerede del af data warehouse og i OLAP. W.H. Inmon beskriver to typer af dataminere dels ”explorers” og ”farmeres”. Explorers søger ustruktureret efter nye sammenhænge og finder ind imellem store guldklumper, mens farmere afprøver mere kendte former for sammenhænge og finder ofte små guldklumper. I forhold til denne sammenligning er OLAP mest egnet til farmere, idet dataene på forhånd er udvalgt i forventning om at de kan imødekomme et kendt behov for informationer. Det vil samtidig også være informationer der ofte efterspørges og tilgængeligheden bør derfor også være hurtig og brugervenlig. Det forhindrer dog ikke at der ud af kendte sammenhænge kan opdages nye, som virksomheden ikke har været bekendt med.

Det kan resultere i at der oprettes nye OLAP, eller i datamining for explorers i de strukturerede data i data warehouse.

8. Konklusion

Formålet med denne rapport har været, at anvise et praktisk projektforløb ved implementering af et data warehouse. Den gennemgående problemstilling har været en praktisk etablering af et data warehouse med udgangspunkt i data warehouse teorien og -teknologi.

Projektgruppen kan med denne rapport konkludere følgende:

1. Data warehouse er en avanceret teknologi, der kræver grundlæggende viden om virksomhedssystemer, databaser og data warehouse teori.
I dette projekt har vi primært koncentreret os om selve data warehouse delen. På sidste semester havde vi en del teori om database teknologi, mens viden om virksomhedssystemer har været overladt til selvstudier.

2. Data warehouse kan give informationer om og analyser af virksomhedens arbejdsgange og dermed bidrage til, at optimere udnyttelsen af virksomhedens ressourcer.

Rapporten har endvidere redegjort for: hvad et data warehouse er, arkitekturen i et data warehouse, datamodeller der anvendes, værktøjer der anvendes i forbindelse med data warehouse, analyse af nye sammenhænge i eksisterende data og typer af beslutningsstøtte der understøttes.

Nogle af gruppemedlemmerne har arbejdet med data warehouse teknologien før, mens andre ikke har.
Da de enkelte gruppemedlemmer har haft forskellige udgangspunkt til emnerne i denne rapport, har det givet forskelligt udbytte.

Det kan dog konkluderes at vi alle er blevet en hel del klogere og erhvervet viden om de forskellige teknikker og begreber der er indeholdt i data warehouse teknologien.

9. Litteraturliste

1. Gawadi, Nadya; Taghizadeh, Siavash: H. Lundbeck's .ppt-præsentation fra MII-seminaret 12.-14. oktober 2000.

2. Corey, M. J.; Michael, A.; Abramson, I.; Taub, B; "Oracle8 Data Warehousing"; Oracle Press; 1998.

3. Inmon, W. H; "Data Relationships in the Data Warehouse"; artikel; 2000; www.billimnon.com.

4. Inmon, W. H; Young Ed; "What is a data mart"; artikel; 1997; www.billimnon.com.

5. Inmon, W. H.; "OLAP and Data Warehouse"; artikel; 2000; www.billimnon.com.

6. Inmon, W. H.; "Data mining: an architecture part 1 og 2"; artikel; 1997; www.billimnon.com.

7. Winsberg, P.; "Modeling the Data Warehouse and Data Mart", InfoDB volume 10 nr. 3; www.dbaint.com/InfoDBDW.html.

8. Gray, Paul; Watson, Hugh J.; "Decision support in the Datawarehouse"; Prentice Hall; ISBN 0137960794

9. Tanler, Rick; "The Intranet Datawarehouse"; John Wiley & Sons, Inc.; ISBN 0471180041

10. Meyer, Don; Cannon, Casey; "Building a better Datawarehouse"; Prentice Hall; ISBN 0138907579

11. Yazdani, Sima; Wong, Shirley S.; "Datawarehousing with Oracle (an administrator's handbook)"; Prentice Hall; ISBN 0135705576

12. Koulopoulos, Thomas M.; Frappaolo, Carl; "Smart things to know about Knowledge Management"; Capstone; ISBN 1-84112-041-3, 1999

13. Mattison, Rob; "Web Warehousing and Knowledge Management"; McGraw-Hill; ISBN 0-07-041103-4, 1999