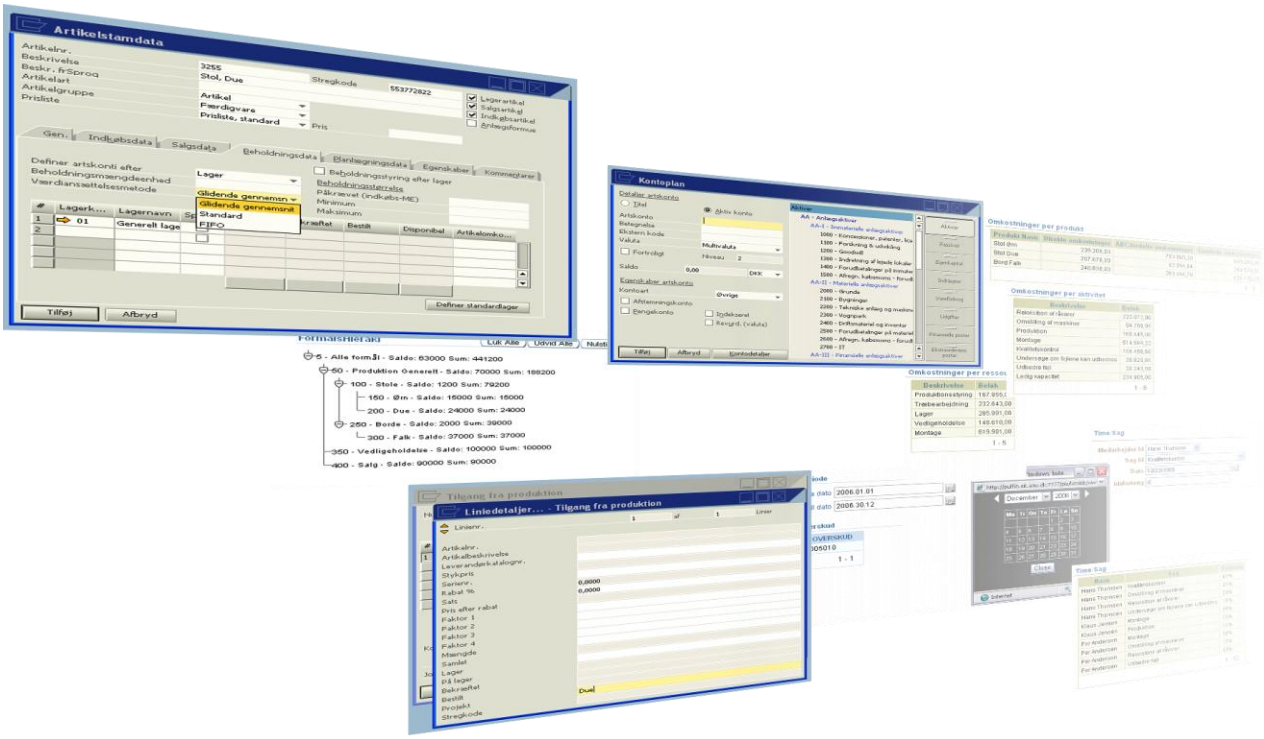
ABC-rapportering baseret på Variabilitetsprincippet og ERP

7. semester - Økonomistyring og informatik
Aalborg Universitet, januar 2007



Jens Frøkjær
Michael Knudsen
Ivan Vigsø Sand Larsen
Carsten Schou Nielsen
Tom Oddershede

Forord

På 7. semester på cand.merc. uddannelsen i økonomistyring og informatik er der udarbejdet tre rapporter delt ud over tre seminarer. Denne rapport er en sammenskrivning af disse, idet temaerne er stærkt sammenhængende.
Temaet for Seminarrapport 1 er "Grundregnskabet og udvalgte økonomistyringsmodeller samt relevant teknologi for systemanvendelse". Herunder vil variabilitetsprincippet blive præsenteret som registreringsgrundlag for virksomheden.
Temaet for Seminarrapport 2 er "beskrivelsesteknikker". Arbejdet vil tage udgangspunk i redskaber til beskrivelse og design af informationssystemer i virksomheder. Der vil ligeledes blive inddraget yderligere en økonomisk model til styring af omkostningsfordeling.
Temaet for Seminarrapport 3 er "Produktionsstyring og virksomhedssystemer". Herved vil arbejdet tage udgangspunkt i et system, som virksomheder kan benytte til at understøtte deres drift og produktionsstyring samt generere ledelsesinformation.

Jens Frøkjær

Michael Knudsen

Ivan Vigsø Sand Larsen

Carsten Schou Nielsen

Tom Oddershede

Indhold

I

Variabilitetsprincippet

1 Indledning

Dette hovedafsnit omhandler variabilitetsprincippet, og hvordan det udvidet med registrering af indtægter kan bruges til at løse Vagn Madsens opgaver. Værktøjer til udvikling af en applikation til håndtering af førnævnte registrering præsenteres, eksempelvis ER-diagrammer, proceshierarkier og CRUD matrix. Endvidere vises tabeldesignet og SQL for databasen. Til slut præsenteres brugergrænseflader udviklet i Oracle Application Express.

1.1 Sammenskrivning

I introduktionen til dette kapitel blev temaet for Hovedafsnit I på modstående side præsenteret. Idet denne rapport er blevet sammenskrevet af tre seminarrapporter vil der forekomme elementer fra andre af rapporterne i dette afsnit. Baggrunden for dette er, at der indgår elementer i seminarrapporterne, som er hensigtsmæssige at implementere tidligere i processen. Således indgår der en række elementer i Seminarrapport 2, som vil hænge naturligt sammen med udviklingen i Seminarrapport 1. Ved at flytte disse elementer vil der opnås en bedre sammenhæng og et mere naturligt flow i den samlede rapport. For at give et metodemæssigt overblik er emnerne for de tre seminarer gengivet her.

1. Her bliver der arbejdet med variabilitetsprincippets registrerings- og rapporteringsteknik, som skal illustreres ved hjælp af Vagn Madsens regnskabsopgaver. Arbejdet med seminaret vil omhandle økonomisk teori og databaseteknik. Arbejdet skal illustrere de økonomiske og informationsmæssige problemstillinger og udfordringer, som erhvervsøkonomer vil kunne støde på i virksomhederne. Herved skal dette seminar ses som en træning i at benytte erhvervsøkonomiske modeller og termer, sammen med de muligheder, IT systemer giver i hverdagen.

2. Seminaret vil fokusere på to forskellige områder. På baggrund af variabilitetsprincippet vil der i første del blive implementeret Activity Based Costing (ABC), for

at gøre det muligt at fordele omkostninger. Herved vil der søges at opnå et system, som i højere grad vil tage højde for at fordele omkostninger til hvert enkelt omkostningsobjekt i virksomheden. Der vil ligeledes blive taget udgangspunkt i værktøjer til design og beskrivelse af informationssystemer i virksomheden. Der vil blive illustreret og forklaret, hvordan en sådan udviklingsproces vil kunne foregå. Ligeledes vil disse værktøjer blive brugt til at identificere aktiviteter til brug i det designede ABC system.

3. Seminaret vil tage udgangspunkt i at beskrive, hvad Enterprise Resource Planning systemer er, og hvordan disse bliver benyttet i virksomheder. Efterfølgende vil der blive gået i dybden med SAP Business One, som er blevet benyttet i undervisningen. Ved at analysere funktionaliteten vil der blive udarbejdet en analyse af, hvordan systemet vil kunne understøtte ABC rapporteringen, og hvilke dele der skal benyttes for at kunne levere data til ABC systemet.

Dette danner baggrunden for struktureringen af denne rapport, hvor Hovedafsnit I primært vil beskæftige sig med registrering ud fra variabilitetsprincippet. Hovedafsnit II vil beskæftige sig med udvidelsen for at kunne rapportere med ABC systemet. Hovedafsnit III vil tage udgangspunkt i, hvordan ERP systemer kan benyttes som kildesystem for rapporteringen med ABC.

1.2 Gennemgående eksempel

For at konkretisere problemstillingen, bliver der benyttet et gennemgående eksempel i rapporten. Eksemplet består af en en produktionsvirksomhed, der producerer to stole (Ørn og Due) samt et bord (Falk). Der vil være forskel på stolene, idet Ørn er luksus modellen og Due er discount. Virksomheden har et lager, der produceres til. Eksemplet vil hjælpe til med at illustrere, hvordan data registreres, både på omkostnings- som indtægtssiden, desuden anvendes hierarkimodellen fra variabilitetsprincippet. Ligeledes vil eksemplet fungere som fordelingseksempel under ABC.

2 Fokusområde

2.1 Variabilitetsprincippet

Der findes ikke et idealsystem for registrering af oplysninger i virksomheder. Alt efter hvilken opgave, der skal udføres, kræves der forskellige former for registreringer. Dette grunder i, at forskellige virksomheder har forskellige krav til hvilke beregninger, denne registrering skal danne grundlag for.

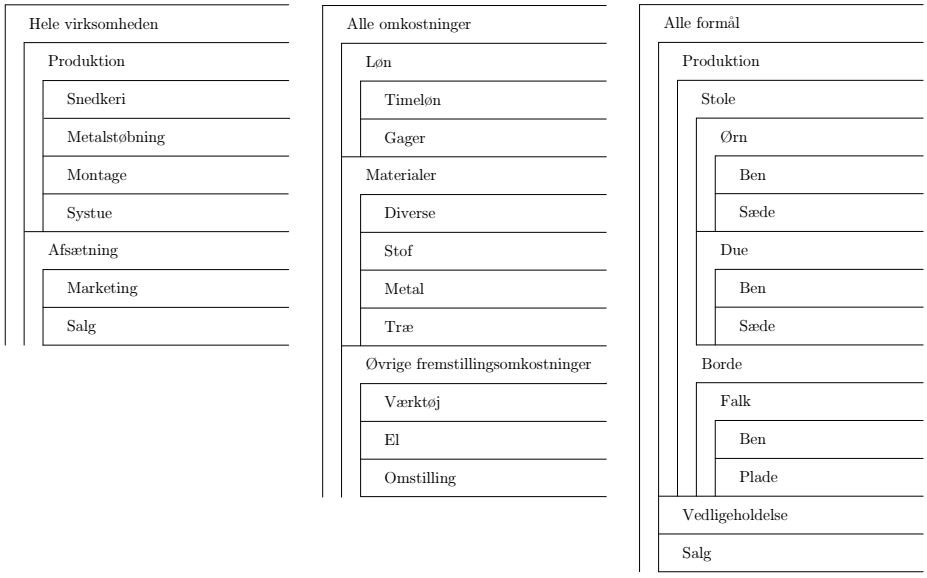
Ifølge [Mad63] har enhver omkostning en faktor, hvormed den varierer proportionalt. Dette gælder dog ikke de faste omkostninger[1]. I variabilitetsprincippet registreres dog ikke alle disse faktorer individuelt. Derimod samles de i nogle større grupper som omkostningerne tilnærmelsesvis varierer proportionalt med, hvilket kaldes variabilitetsfaktorerne.

Variabilitetsregnskabet håndterer kun omkostninger og ikke indtægter eller udgifter[2]. Omkostninger er et forbrug af ressourcer til budgetsatser[3]. Variabilitetsregnskabet karakteriserer omkostninger ud fra forbrugets art, sted og formål. Disse opbygges som hierarkier af konti, indenfor hvilke omkostninger registreres så langt nede i hierarkiet, som det er muligt uden arbitrær fordeling[4]. Omkostningens art er ressourcetypen, eksempelvis forbrugt timeløn eller materialer. Stedet repræsenterer den organisatoriske del af virksomheden, hvor forbruget har fundet sted.

I Figur 1(a) på næste side ses et eksempel på, hvordan stedsdimensionens hierarki kan bygges op i forhold til det gennemgående eksempel. Formålet med omkostningen kan være virksomhedens produkter, eller mere overordnede formål som eksempelvis afsætning.

Figur 1(b) på den følgende side viser et eksempel på den hierarkiske opbygning af arts-

[1] [Mad63], side 133
[2] [Mad63], side 132
[3] [BI04], side 82
[4] [Mad63], side 134



(a) Stedshierarki (b) Artshierarki (c) Formålshierarki

Figur 1: Eksempler på omkostnings-hierarkier

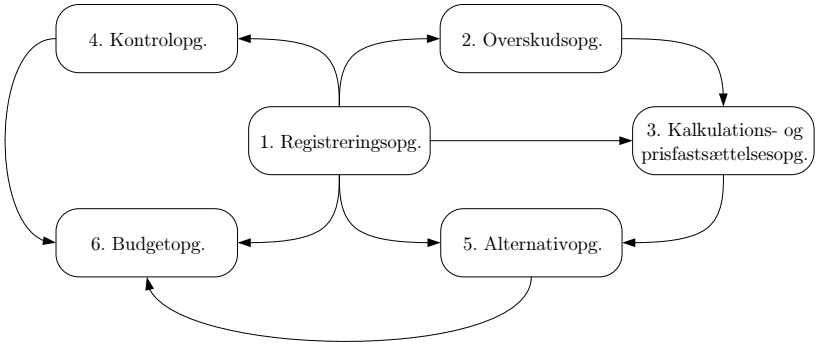
dimensionen. Her vises, at omkostningerne kan kategoriseres som løn, materialer eller øvrige fremstillingsomkostninger. Længere nede i hierarkiet vil omkostningsarten blive mere og mere specifik. For eksempel kan materialer blandt andet kan være stof eller metal.

Figur 1(c) viser et eksempel på en hierarkisk opbygning af formålsdimensionen. Her ses blandt andet, at produktgruppen stole indeholder produkterne Ørn og Due. Hvis en maskine producerer både Ørn og Due, skal denne maskines omkostninger (eksempelvis vedligeholdelse) placeres umiddelbart på niveauet over selve produkterne. Dette er tilfældet, hvis omkostningerne ikke kan henføres direkte til de enkelte produkter. Omvendt kan maskinens el-forbrug måles ved produktionen, og dermed kan denne omkostning konteres helt nede på produktniveau. Så længe en arbitrær fordeling undgås kan omkostninger, der umiddelbart fremtræder som værende fælles for flere konti, alligevel henføres til flere forskellige konti. Ved eksempelvis direkte måling af tidsforbrug undgås den arbitrære fordeling[5].

Ifølge [Mad63] har regnskabsvæsenet seks opgaver[6]: 1. Registreringsopgaven, 2. Overskudsopgaven, 3. Kalkulations- og Prisfastsættelsesopgaven, 4. Kontrolopgaven, 5. Alternativopgaven og 6. Budgetopgaven. Ofte opbygges et registreringssystem, som danner et regnskabsgrundlag for en enkelt af opgaverne, hvilket begrænser dets anvendelsesmuligheder. I kontrast hertil står variabilitetsregnskabet, som ikke danner grundlag for en enkelt opgave, men derimod danner grundlag for, at alle opgaverne kan løses på lige fod. Figur 2 på modstående side viser sammenhængen mellem Vagn Madsens seks opgaver. Eksempelvis viser pilen mellem registreringsopgaven og kalkulations- og pr-

[5][Isr93], side 18
[6][Mad63], side 18

isfastsættelsesopgaven, at registreringsopgaven skal være udført, før kalkulations- og prisfastsættelsesopgaven kan udføres. Denne opgave kræver dog også, at overskudsopgaven er udført. Generelt kan det ses ud fra Figuren 2, at registreringsopgaven skal være udført, før nogen af de andre opgaver kan udføres.



Figur 2: Sammenhæng mellem Vagn Madsens seks opgaver

Variabilitetsregnskabets kontering af flere produkters fællesomkostninger på en "overkonto", besværliggør udførsel af eksempelvis kalkulations- og prisfastsættelsesopgaven. Da omkostningerne ikke er fordelt på produktniveau, kendes den præcise omkostning pr. produkt ikke. Eksempelvis kan vedligeholdelse af produktionsbygningerne ikke henføres direkte til enkelt-produkter i variabilitetsregnskabet. Alligevel skal disse omkostninger dækkes af netop produkterne. For yderligere præcision i eksempelvis prisfastsættelsesopgaven, bør der ske en fordeling af omkostningerne helt nede på produktniveau i afrapporteringen.

2.2 Afgrænsning af opgaver

En virksomhed skal bruge alle Vagn Madsens opgaver for at kunne udarbejde en fyldestgørende rapportering[7]. I henhold til omfanget af denne rapport er der valgt ikke at implementere alle Vagn Madsens seks opgaver. Der er i denne denne opgave valgt primært at løse registreringsopgaven, overskudsopgaven samt kalkulations- og prisfastsættelsesopgaven. Dermed er der fravalgt løsning af alternativ-, budget- og kontrolopgaven. Fravalget hænger logisk sammen som følge af Figur 2. Det vil ikke være muligt at udføre budget- og alternativopgaven uden først at have udarbejdet registrerings- samt kalkulations- og prisfastsættelsesopgaven. Kontrolopgaven skal ligeledes holdes op mod budgettet, hvilket også gør løsningen af denne til sekundær, når budgetopgaven ikke implementeres . Ved at begynde med registreringsopgaven lægges grundlaget for Vagn Madsens øvrige opgaver. Herefter virker det naturligt at bygge videre efter Figur 2 og netop arbejde videre med overskudsopgaven samt kalkulations- og prisfastsættelsesopgaven.

[7] [Mad63], side 19

2.3 Implementering af indtægter

Registrering af indtægter er ikke direkte kommenteret i [Mad63] udlægning af variabilitetsregnskabet. Andre har dog senere foreslået implementering af indtægter i registreringen som for eksempel [Rie94]. Også [BI04] påpeger manglen på indtægter i variabilitetsprincippet[8].

Specielt er indtægtssiden vigtig for at kunne løse overskudsopgaven. Ligeledes giver implementeringen af indtægtssiden også muligheder for at kunne udføre budget- og alternativopgaven i variabilitetsprincippet. Sammenhængen mellem indførelsen af indtægter og variabilitetsprincippet ses ud af Figur 3. Ved denne implementering opnås mulighed for i nogen grad at udføre alle Vagn Madsens opgaver.

I Figur 3 er opgaverne gradinddelt efter løsbarhed ved det givne informationsniveau. Der er anvendt tre niveauer:

- ÷ - kan ikke løses
- (✓) - kan løses i nogen grad
- ✓ - kan løses fuldt ud

Det skal understreges, at der er mulighed for at forbedre princippet yderligere ved hjælp af andre værktøjer, som senere kan implementeres og derved give mulighed for yderligere præcision. Dette vil blive behandlet i Hovedafsnit III på side 66.

	Omkostninger	*Indtægter*
1. Registreringsopgaven	✓	✓
2. Overskudsopgaven	÷	(✓)
3. Kalkulations- og prisfastsættelsesopgaven	(✓)	(✓)
4. Kontrolopgaven	✓	✓
5. Alternativopgaven	÷	(✓)
6. Budgetopgaven	÷	(✓)

Figur 3: Sammenhæng mellem registreringsprincipper og Vagn Madsens seks opgaver

Argumentationen for at overskudsopgaven muliggøres er, at det med indtægtssiden bliver muligt at fastslå virksomhedens overskud i en given periode. Dette var ikke muligt før, idet kun omkostningerne blev registreret. Herved gives mulighed for udtræk af data

[8] [BI04], side 80

omkring overskuddet, som hjælper ledelsen i virksomheden ved at levere data til interne overvejelser og beregninger på udbytte og skat[9].

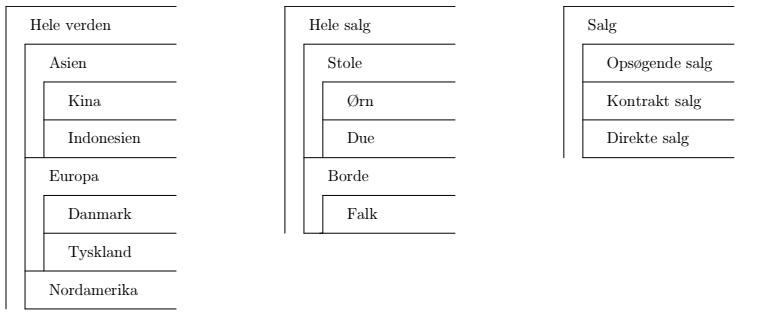
Indførelsen af indtægtssiden påvirker også muligheden for at udføre alternativ- og budgetopgaven. Det er naturligt, at begge opgaver opnår samme fordele, da de er tæt bundet sammen. Ved at have muligheder for at trække på indtægter bliver det muligt for virksomheden at planlægge rammer og handlingsplaner for fremtiden og beskrive disse i budgettet. Det er ikke muligt at udtale sig om resultatet uden, at virksomheden kan se. hvad den tjener på sine produkter, hvilket vil medføre, at der i budgettet ikke kan ses, hvad virksomheden vil tjene. Samme problematik går igen i alternativopgaven, idet den umiddelbart går ud på at udarbejde forskellige budgetscenarier indenfor virksomhedens rammer. Dette er baggrunden for at implementere indtægtsregistreringen i variabilitetsprincippet.

Der er flere teoretiske muligheder for at implementere indtægter i variabilitetsprincippet. Ud fra hvilken tanke, der ligger bag implementeringen, er der mulighed for at variere kompleksiteten i registreringen. I implementeringen af indtægter er der valgt at registreringen skal ligge tæt op imod registreringen af omkostninger. En måde til at registrere indtægterne kan være at benytte det eksisterende tabeldesign. Herved vil en indtægt blive set som en negativ omkostning i tabellerne. Her vil alle indtægter altså komme til at stå som minus i databasetabellerne. Dette vil i praksis gøre opgaver, som bestemmelse af overskud, simple, idet data dertil kan hentes i en enkelt tabel. Ved at benytte denne metode til registrering vil databasen blive minimal i omfang. Ulempen ved at benytte et samlet sted at registrere indtægter og omkostninger vil være, hvis der ifølge den teoretiske diskussion ikke er identiske klasser for arter, formål og steder i virksomheden. En anden ulempe ved at registrere i én tabel vil også være, hvis en registrering med værdien nul skal registreres, vil det resultere i, at det ikke er muligt at fastslå, om det er en indtægt eller omkostning for virksomheden. Endvidere vil registrering af krediterede omkostninger fremstå som en negativ omkostning, og vil dermed ikke kunne skelnes fra indtægter.

Derfor er det mest hensigtsmæssigt at implementere to separate tabelstrukturer til systemet. Herved kommer registreringen af omkostninger og indtægter til at stå for sig selv. Indtægtssiden og omkostningssiden skal så kobles sammen ved hjælp af for eksempel sammenhængen med produkter eller tid. Ved at benytte dette alternativ bliver der også mulighed for at oprette hierarkier omkring art, sted og formål på indtægtssiden. Dette vil umiddelbart ikke være et krav til den simple indtægtsregistrering, men vil kunne hjælpe med til i mere komplekse forespørgsler at kunne give virksomheden et bedre billede af indtægtsstrukturen. Hierarkier på indtægtssiden kan konstrueres som det er illustreret ved Figur 4 på næste side.

På Figur 4(a) på den følgende side kan ses et eksempel på stedshierarkiet i forhold til det gennemgående eksempel. I modsætning til omkostningsregistreringen, hvor der registreres, hvor i virksomheden omkostningen er afholdt, så registreres stedet i verden hvor, indtægten er opnået. På nederste niveau er den enkelte kunde. Over kunden er

[9] [Mad63], side 18



(a) Stedshierarki (b) Artshierarki (c) Formålshierarki

Figur 4: Eksempler på indtægts-hierarkier

dennes geografiske position, for eksempel by, land og verdensdel.

Det kan ligeledes ses på Figur 4(b), at artshierarkiet adskiller sig markant i indtægtsregistrering i forhold til omkostningsregistrering. Arten, der registreres, er det enkelte produkt i virksomhedens varesortiment. På niveauerne over er der produktserier, og igen over dem er alle produkterne.

På Figur 4(c) ses et eksempel på formålshierarkiet på indtægtssiden. Dette hierarki skal beskrive måden, den givne indtægt er opnået på. Dette kan for eksempel være opsøgende salg eller direkte salg.

2.4 De tre valgte opgaver

Vi har, som skrevet i Afsnit 2.2 på side 15, valgt at koncentrere os om Vagn Madsens tre første opgaver. Vi vil efterfølgende give en klarere karakteristik af opgaverne og, hvad de nøjagtigt skal indeholde i forbindelse med, hvordan virksomhederne praktisk skal benytte sig af registreringssystemet.

2.4.1 Registreringsopgaven

Registreringsopgaven i variabilitetsprincippet har til opgave at sørge for, virksomhedens omkostninger bliver registreret, således det senere bliver muligt at kunne lave relevante dataudtræk. Registreringsopgaven skal ligeledes tage højde for, at det skal være muligt for virksomheden at benytte sig af de arts- formåls- og stedshierarkier, som variabilitetsprincippet foreskriver. Herved bliver det, som beskrevet, muligt for virksomheden at kunne registrere omkostningerne i disse hierarkier. Det er vigtigt, at data bliver registreret så præcist som muligt for senere at sikre validiteten af det data, der trækkes ud af systemet. Dog skal registreringsmodellen ikke være mere kompleks end, registreringer forholdsvis enkelt kan opdeles og konteres de korrekte steder.

Ved at indføre indtægter i variabilitetsregnskabet udvides registreringsopgaven. Den

simpleste måde at registrere indtægter på vil være blot at kontere et salg og et beløb. Denne registrering kan naturligvis give større mulighed for at løse flere andre af Vagn Madsens opgaver. Omvendt vil denne simple registrering afskære muligheden for mere avanceret analyse på et senere tidspunkt. Umiddelbart bør registreringen også omfatte kunde- og produktinformation for at muliggøre rapportering på enkeltprodukter eller markeder.

2.4.2 Overskudsopgaven

Vagn Madsen har defineret overskuddet som "en periodes indtægter, fratrukket de udgifter, som er medgået til at fremskaffe indtægterne". Ifølge definitionen kan overskudsopgaven, logisk nok, ikke løses uden at tage indtægterne med i beregningen. Som nævnt under registreringsopgaven vil en mere detaljeret registrering give mulighed for analyse af en periodes aktiviteter. Dermed er det ikke blot muligt at løse overskudsopgaven for hele virksomheden, men også på produkt eller produktserie-niveau. Ligeledes er det vigtigt, at parametre som tid, afsætningssted og kunde bliver registreret i systemet, således det bliver muligt for virksomheden at kunne finde overskuddet og endnu vigtigere hvilke kunder, der har givet overskud og på hvilke markeder.

2.4.3 Kalkulations- og prisfastsættelsesopgaven

Kalkulations- og prisfastsættelsesopgaven er yderst vigtig i en virksomhed, for at kunne sikre profitabel drift. Det skal være muligt at udarbejde kalkuler på produkter, produktserier og enkeltordrer. Herved bliver det muligt for virksomheden at sætte prisen, således at den afspejler et korrekt billede af omkostningsstrukturen, hvilket gerne skulle lede til profit. Opgaven kan udvides med et antal af forskellige muligheder og beregningsmetoder, hvorved den kan blive meget kompleks. Kalkulationsopgaven kræver ikke at der bliver registreret andre data end dem, som er beskrevet i registreringsopgaven. Dog skal det tilstræbes, at de fremstillede hierarkier under denne opgave er så præcise som muligt, hvilket vil påvirke nøjagtigheden af kalkulationerne.

Indførelsen af indtægtssiden er ikke direkte nødvendig for at kunne udføre Kalkulations- og prisfastsættelsesopgaven. Det giver dog nogle fordele, fordi det gør det muligt at trække på historik over tidligere indtægter. Herved kan det blive muligt at benytte disse data som et beslutningsgrundlag.

3 CASE værktøjer

I dette kapitel vil der blive gennemgået de forskellige skridt, der bør tages i forbindelse med udvikling af en applikation. Kapitlet vil indeholde en præsentation af CASE værktøjet Oracle Designer, som indeholder faciliteter til at modellere en applikation helt fra analysefasen til det færdige produkt. I dette kapitel vil der hovedsageligt blive benyttet de værktøjer Oracle Designer bruger til at udarbejde de forskellige diagrammer og matrixer, som skal bruges til at udarbejde et tabeldesign. Det skal bemærkes, at skærmbilleder og diagrammer i rapporten ikke er fra Oracle Designer, idet de er videreført for pænere layout. Selve applikationen vil blive udviklet i Application Express. Værktøjerne beskrives i den rækkefølge, de typisk vil blive benyttet i under udviklingsforløbet.

3.1 Oracle Designer

Oracle Designer er et CASE værktøj, der indeholder en lang række hjælpemidler til udvikling af applikationer. Der arbejdes på to niveauer; *Upper* CASE og *Lower* CASE[10]. Upper CASE omhandler de analytiske værktøjer, der anvendes i forbindelse med udarbejdelse af for eksempel entitet-relation diagrammer, procesdiagrammer og CRUD matrixer. Lower CASE omhandler hovedsageligt design editoren, hvor der kan opbygges det reelle tabeldesign.

I denne rapport vil der hovedsageligt fokuseres på Upper CASE delen, hvor der udarbejdes forskellige analytiske diagrammer.

Oracle Designer er tæt forbundet med databasen. På basis af de analytiske diagrammer, der er blevet udarbejdet i Upper CASE delen, kan tabeldesignet genereres automatisk til et såkaldt server-model-diagram, som derefter kan eksporteres til SQL, der kan afvikles direkte i databasen.

[10][ADI99], side 5

3.2 Procesdiagrammer

Procesdiagrammer bruges til at identificere de enkelte arbejdsprocesser og sammenhængen mellem disse. Et procesdiagram bruges oftest til at beskrive en enkelt arbejdsopgave, og hvordan denne påvirker de forskellige processer i virksomheden.

Der findes forskellige måder at beskrive disse processer. For eksempel kan processer beskrives ved hjælp af UML gennem aktivitets- og swimlanediagrammer[11]. En ældre men stadig meget brugt diagrammeringsteknik er flow diagrammer[12]. Det er den teknik, som bliver brugt af Oracle Designer, og også det der benyttes i denne rapport.

Et sådant diagram består af forskellige elementer. Det første, der sker, er en hændelse, hvorefter processen beskrives ved hjælp af procestrin, dataindtastning, rapporter og beslutninger. En proces slutter oftest med en eller flere udgående hændelser.

Processer kan både være manuelle eller automatiserede. Dette angives i et procesdiagram ved at farve de manuelle processer grå. Beslutninger angives ved en rombe, der har et antal udgående streger, som repræsenterer de forskellige beslutninger.

3.3 Funktions- og applikationshierarkier

Et funktionshierarki viser, hvordan virksomhedens funktioner hænger sammen. Funktionshierarkiet er tæt forbundet med procesdiagrammerne, men ser processerne fra en anden synsvinkel. Procesdiagrammerne ser opgaven på tværs af de forskellige aktører og afdelinger, mens funktionshierarkier viser hvilke processer, der hører under hvilke funktioner.

Applikationshierarkiet minder meget om funktionshierarkiet, men med den forskel, at alle beslutninger og manuelle funktioner er eliminerede. Når applikationshierarkiet er opbygget, er dette udgangspunkt for opbygningen af applikationen.

3.4 Entitet-relation-diagrammer

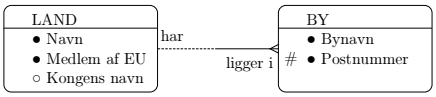
Entitet-relation-diagrammet (ER-diagram) er en model, der bygger på *opfattelser* af den rigtige verden. Det består af nogle entiteter, der modellerer ting eller objekter fra den *rigtige verden*[13]. En entitet har nogle attributter, som er egenskaber ved en given entitet.

En entitet er afbilledet i et ER-diagram som et rektangel med afrundede hjørner. Entiteternes indbyrdes forhold bliver afbilledet med forskellige linietyper. En stiplet linie

[11] [Lar02], side 607–608
[12] [Pre05], side 348
[13] [SKS01], side 8

angiver, at der er valgfri deltagelse fra en entitet til en anden. En fuldt optrukket linie angiver, at en entitet skal have deltagelse af den anden entitet på oprettelsestidspunktet. Derudover angiver en gaffel på enden af linien, at den modsatte entitet kan have deltagelse af flere af den givne entitet.

Figur 5 viser et eksempel på to entiteter `land` og `by`. Det ses, at der er en en-til-mange relation fra `by` til `land`. Det resulterer i, at et land kan have flere byer, men en by ligger kun i et land. Derudover ses det, at der er tvungen deltagelse fra `by` til `land`, hvorimod der er valgfri deltagelse fra `land` til `by`. Dette betyder også, at en by skal ligge i et land, men et land indeholder ikke nødvendigvis en by.



Figur 5: Eksempel på to entiteter

3.5 CRUD matrix

Når applikationshierarkiet og ER-diagrammet er færdige, er det muligt at udarbejde en CRUD matrix. Denne lister alle processer fra applikationshierarkiet horisontalt og alle entiteter fra ER-diagrammet vertikalt. For hver proces angives nu for hver entitet, hvilke rettigheder denne proces skal have. Dette angives med et C for *create*, et R for *read*, et U for *update* og et D for *delete*.

Når CRUD matrixen er udarbejdet, bør alle entiteter være berørt på mindst en eller anden måde af en af processerne. Hvis dette ikke er tilfældet, bør der overvejes, om denne entitet er nødvendig. Det kan selvfølgelig også være, at denne entitet bliver brugt i opgaver, som ikke er modelleret i procesdiagrammerne.

4 Udvikling af applikation

I dette kapitel vil diagrammerne, der skal bruges i forbindelse med variabilitetsprincippet, blive gennemgået. Diagrammerne vil blive brugt til at illustrere aktiviteter og identificere de forskellige processer i de forskellige opgaver. Derudover vil de forskellige skridt for at nå til det egentlige tabeldesign blive gennemgået. Endvidere vil tabeldesignet blive præsenteret samt SQL til oprettelse af denne. Endelig vil brugergrænsefladen blive præsenteret.

4.1 Manuelle og automatiserede processer

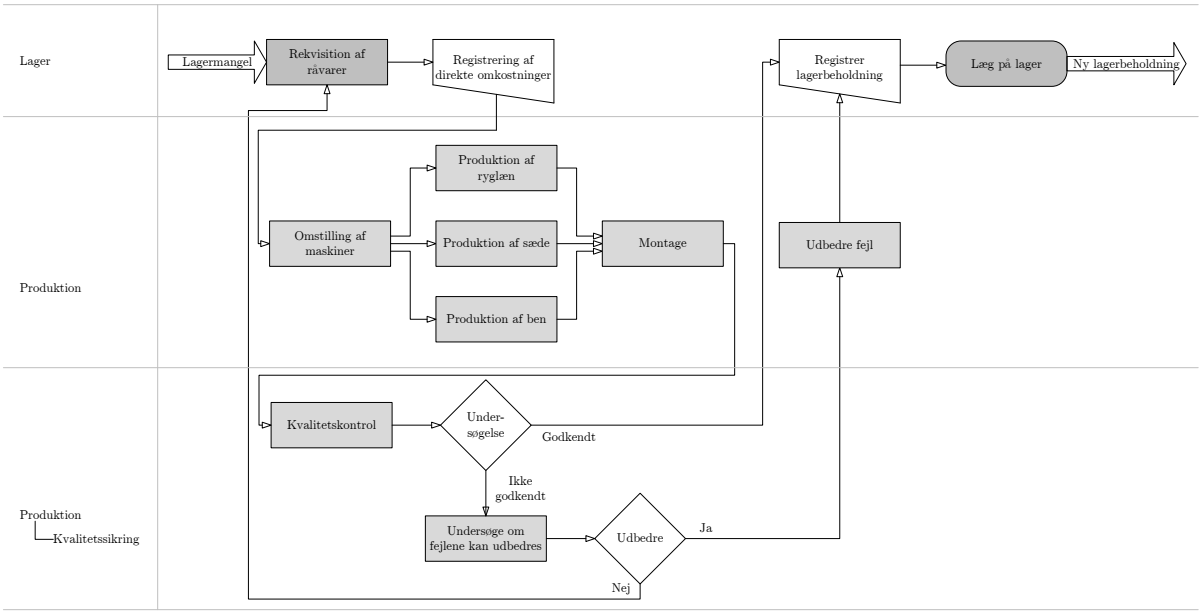
Ved opbygning af eksempelvis procesdiagrammer skal processerne kategoriseres som værende manuelle eller automatiserede. Dette har betydning senere, når applikationshierarkiet opbygges. Dette hierarki tager udgangspunkt i funktionshierarkiet, men de manuelle processer og decision points fjernes. Applikationshierarkiet består dermed kun af automatiserede processer. Applikationshierarkiet danner sammen med ER-diagrammet grundlag for udarbejdelsen af CRUD matrixen.

I denne rapport defineres en automatiseret proces som en proces, der involverer brug af IT systemet.

4.2 Procesdiagrammer

I dette afsnit vil procesdiagrammer, som kan bruges til at beskrive forskellige arbejdsgange i virksomheden, blive præsenteret.

I eksemplet i Figur 6 på næste side er arbejdsgangen for produktion af en stol vist.



Figur 6: Procesdiagram for produktion

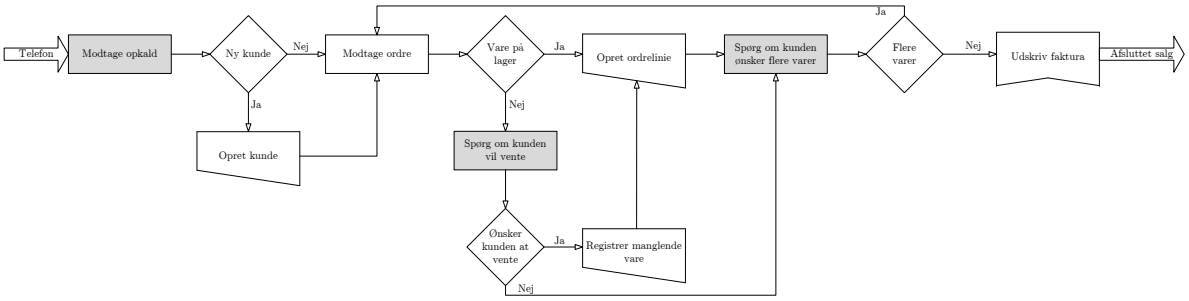
Eksemplet tager udgangspunkt i virksomheden fra det gennemgående eksempel, som blev præsenteret i Afsnit 1.2 på side 12. Denne virksomhed er en lagerproducerende møbelfabrik, hvilket bevirker at varemangel er den handling, der sætter produktionen i gang. I en sådan virksomhed vil der foreligge prognoser og analyser for at understøtte, hvad der skal være på lager for at modvirke sæsonudsving med videre.

Som det kan ses af Figur 6, er produktionsprocessen en parallelisering af ryglæn, sæde og ben. Dette er gjort, fordi der ikke er årsagssammenhæng mellem produktionen af disse. Herved menes, at et ryglæn ikke forårsager produktion af et sæde og så videre. Det er hvad, der ligger på lager, der bestemmer, hvad der produceres, og hvornår dette sker. Efter produktion af disse tre emner, samles de til en stol. Dernæst udføres kvalitetskontrol i kvalitetssikringsafdelingen under produktion. Hvis stolens kvalitet kan godkendes, lægges den på lager. Ellers udføres endnu en kontrol, der skal afgøre, om det kan betale sig at reparere stolen, så den opnår den ønskede kvalitet. I det tilfælde udbedres fejlene i produktionen. Hvis det ikke er økonomisk forsvarligt at reparere fejlene, smides stolen ud, og der påbegyndes produktion af en ny.

Ud fra diagrammet ses det, at der er tre aktører i dette eksempel: `Lager`, `produktion` og `kvalitetskontrol`. Det ses, at det er lageret, der sætter produktionen i gang. Når møblet er samlet, sættes kvalitetskontrollen i gang. I denne virksomhed ligger afdelingen kvalitetssikring under produktionen.

Der kan argumenteres for, at der for eksempel kan være flere lagertilgange end dem procesdiagrammet viser. Her er det op til den enkelte virksomhed at beslutte, hvor detaljeret udførelsen af dette diagram skal være. Hovedreglen er, at hvis det ikke har betydning for forståelsen af processen eller for eventuel identifikation af aktiviteter, skal

der udvises påpasselighed med udbygning, da det vil kunne gøre procesforståelsen mere uoverskuelig. Ofte ville en enkelt proces uddybes yderligere ved at lave en *open down* på en sådan, hvilket bevirker, at denne ene proces kan beskrives yderligere ved hjælp af et helt nyt procesdiagram. Et eksempel på en proces, der kunne beskrives yderligere, er montage.



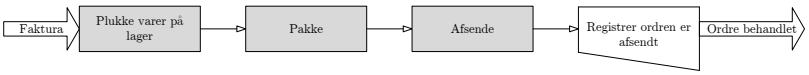
Figur 7: Procesdiagram for salg

Figur 7 viser procesdiagrammet for en salgssituation. Denne viser, at et telefonopkald i dette tilfælde er den handling, der starter et salg. Diagrammet viser, at der er en række processer, der gentages for hver vare, der ønskes. Når ikke flere varer ønskes, udskrives fakturaen.

Der kan argumenteres for, at der også ville være et procesdiagram for opsøgende salg. Dette er dog undladt, da en sådan ikke vil tilføje nye relevante problemstillinger, men blot være gengivelser af det, der er udarbejdet i de andre procesdiagrammer.

Da salgsprocessen ikke involverer andre funktioner i virksomheden end salg, er der ikke benyttet "swimlanes".

Det kan ses, at eksempelvis processen **Opret kunde** er automatiseret, selvom det rent faktisk kræver manuelt arbejde fra en medarbejder. Men ifølge definitionen vil **Opret kunde** blive kategoriseret som en automatiseret proces.



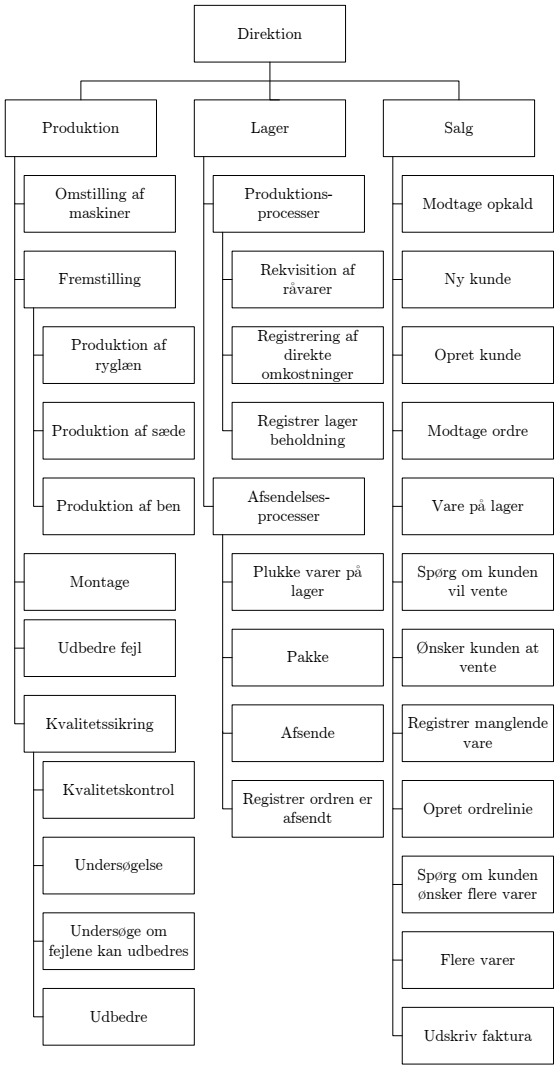
Figur 8: Procesdiagram for afsendelse fra lager

I Figur 8 illustreres, hvad der sker på lageret, efter salgsafdelingen har udskrevet en faktura. Varerne findes på lageret, pakkes og afsendes. Derefter registreres det, at varerne er afsendt.

4.3 Funktions- og applikationshierarkier

Figur 9 på den følgende side viser funktionshierarkiet for processerne fra Figur 6, Figur 7 og Figur 8. Virksomheden er delt op i tre overordnede funktioner; **salg**, **lager**

og **produktion**. Derudover har produktionen en kvalitetssikringsafdeling. Processerne fra procesdiagrammet er her lagt ind under de relevante funktioner i virksomheden. For at øge overblikket er der eksempelvis i lageret indført to overordnede processer; **produktionsprocesser** og **afsendelsesprocesser**.

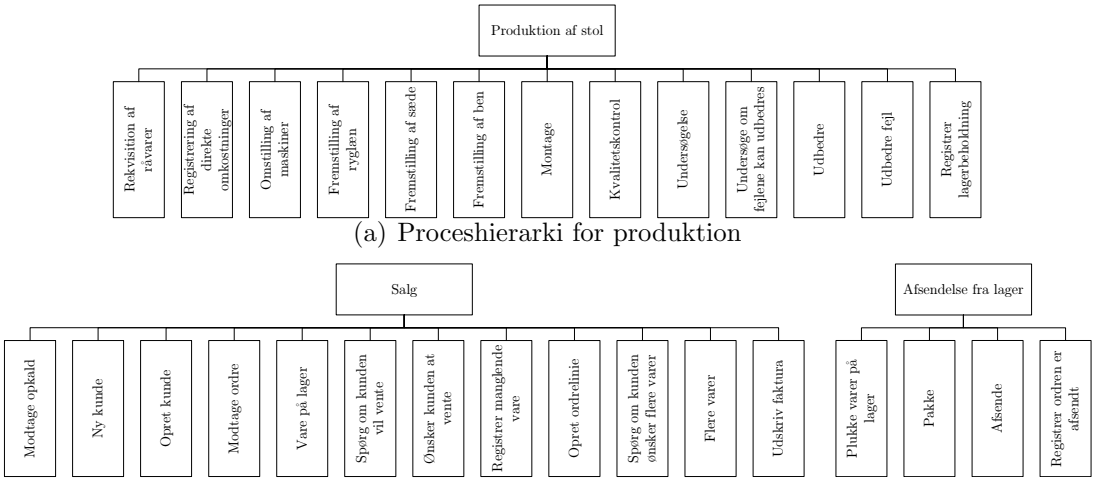


Figur 9: Funktionshierarki

I løbet af produktionsprocessen, afbilledet på Figur 10(a) på næste side sker der ting i flere forskellige afdelinger. I proceshierarkierne ses alle delprocesser, men funktionen, i hvilken delprocessen foregår, kan ikke ses i proceshierarkiet.

I funktionshierarkiet i Figur 9 bliver processerne delt ud på de relevante funktioner i virksomheden.

Proceshierarkierne for processerne, illustreret i Figur 7 og Figur 8 på foregående side, kan ses i Figur 10 på modstående side.



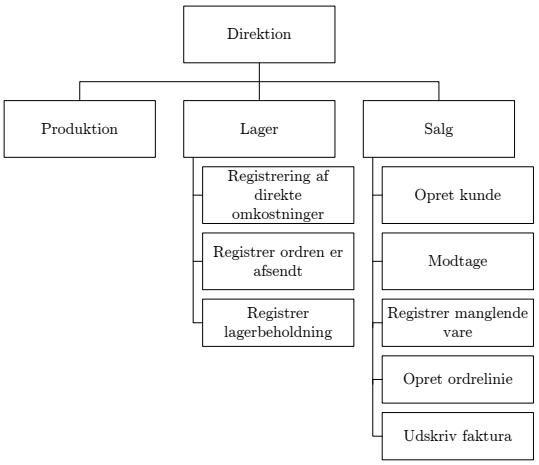
(a) Proceshierarki for produktion

(b) Proceshierarki for salg

(c) Proceshierarki for afsendelse fra lager

Figur 10: Proceshierarkier

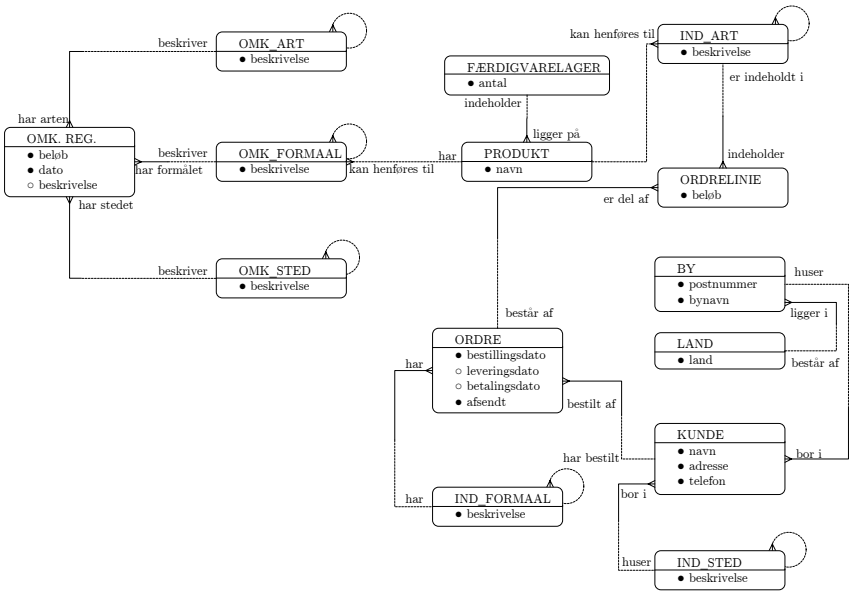
I Figur 11 ses applikationshierarkiet. Det er dannet på baggrund af funktionshierarkiet, hvor de manuelle processer og decision points udelades. I funktionshierarkiet er der, for at øge overblikket, indført flere overordnede processer. Ved genereringen af applikationshierarkiet er disse overordnede processer fjernet, da de kun indeholder en delproces hver.



Figur 11: Applikationshierarki

4.4 ER-diagrammer

I dette afsnit vil der blive udviklet ER-diagrammer for variabilitetsprincippet, inklusive udvidelsen ved indtægter.



Figur 12: ER-diagram for omkostnings- og indtægtsregistrering

Figur 12 viser det komplette ER-diagram for tabeldesignet for variabilitetsprincippet. Her ses i venstre side omkostningsregistreringen og i højre side indtægtsregistreringen, bundet sammen af `Produkt`-entiteten. På for eksempel `Omkostningsregistrering`-entiteten ses, at **beskrivelse** er valgfrit, hvorimod **beløb** og **dato** er påkrævet. Dette udmønter sig i, at i tabeldesignet vil **beskrivelse** være *nullable*. I tabeldesignet vil entiteter, der ikke har nogen naturlig primær nøgle, få tildelt en syntetisk primær nøgle.

4.5 CRUD matrix

Efter at have udarbejdet ER-diagrammet og applikationshierarkiet kan CRUD matrixen fremstilles. Processen **Registrering af direkte omkostninger** sker eksempelvis, når der fremtages råvarer fra lageret til produktion. Som det fremgår af CRUD matrixen, skal denne proces kunne oprette, hente, opdatere og slette i entiteten `Omkostningsregistrering`. Derudover skal denne proces kunne hente data fra entiteterne `Omkostningsart`, `Omkostningsformål`, `Omkostningssted` og `Produkt`. Det betyder, at disse entiteter skal indeholde data forud for omkostningsregistreringen. Dette muliggør visning af kontonavne ved omkostningsregistrering og kan forhindre, at en omkostning forsøges konteret på et kontonummer, der ikke eksisterer.

Modtag ordre processen kan oprette, hente og opdatere i entiteten `Ordrelinie`. Det kan virke uhensigtsmæssigt, at det ikke kan lade sig gøre at slette en ordrelinie, hvis der sker fejlindtastning. Men da fejlindtastning ikke er en del af **Modtag ordre** processen, skal der ikke tages hensyn til dette.

Som det kan ses i Figur 13 kan processen **Udskriv faktura** kun læse. **Udskriv faktura** er det sidste, der sker, i løbet af et salg. Da denne proces ikke skal tilføje eller ændre i data, har den kun mulighed for at læse.

	Registrering af direkte omkostninger	*Registrer ordre afsendt*	*Opret kunde*	*Modtag ordre*	*Registrer manglende vare*	*Opret ordrelinie*	*Udskriv faktura*	*Registrer lagerbeholdning*
Omkostningsregistrering	CRU							
Omkostningsart	R							
Omkostningsformål	R							
Omkostningssted	R							
Produkt	R				U	R	R	RU
Indtægtsart						R		
Ordrelinie						CRU	R	
Ordre		U		CRU			R	
Indtægtsformål				R				
Kunde			CRU	R			R	
Indtægtssted			R					
Land			CRU	R			R	
By			CRU	R			R	

Figur 13: CRUD matrix

Efter at have udfyldt CRUD matrixen kan der udarbejdes skærmbilleder i eksempelvis Oracle Application Express. Når en proces som **Udskriv faktura** kun har mulighed for at hente data fra et antal entiteter, skal der fremstilles et skærmbillede, der kun giver mulighed for at fremvise og udskrive data. Dette ses i modsætning i **Opret ordrelinie**, der jo netop har mulighed for at ændre i de data, som **Udskriv faktura** kun har mulighed for at læse. Umiddelbart ville det være lettest, hvis de to processer kunne benytte den samme formular. Men da de ikke har samme rettigheder til at ændre i entiteterne, skal de have forskellige formularer.

4.6 Databasens formål og opbygning

Det primære formål med en database (DBMS) er at have et effektivt og relativ simpelt værktøj til at gemme og hente data på. Databaser er designet til at kunne håndtere store mængder af data, samtidig med at de giver mulighed for sikkerhed i form af blandt andet brugerkontrol og transaktionssikkerhed, selv når systemet bryder ned[14]. Der findes forskellige former for databaser, men den mest udbredte er relationsdatabasen, som består af flere tabeller, som bindes sammen ved hjælp af joins.

I mange tilfælde vil der, som alternativ til et DBMS, kunne vælges at gemme sine data i forskellige filer, som forskellige brugere kan læse og gemme efter behov. Dette kan dog skabe en del problemer så som redundans og inkonsistente filer. Redundans vil sige, at

[14][SKS01], side 1

de samme data kan forekomme flere steder, hvilket kan medføre, at der kun bliver rettet et sted, og dette leder dermed til inkonsistens. Samtidig kan der opstå overraskende opførsel i forbindelse med at flere brugere arbejder på systemet samtidigt[15].

Da systemet, der skal udvikles i forbindelse med denne rapport, er et regnskabssystem, hvor der potentielt kan være mange brugere af systemet, og hvor datasikkerhed er meget vigtig, vil Oracle DBMS blive brugt.

Som tidligere nævnt er redundans og inkonsistens ikke hensigtsmæssigt. Selvom der benyttes et DBMS, så sikrer det ikke automatisk, at der ikke er redundante og inkonsistente data. For at sikre, at dette ikke opstår, benyttes ofte de tre normalformer. De er designet til at afhjælpe dette problem.

4.6.1 Normalisering

Normaliseringen kan deles op i forskellige skridt (normalformer), som én efter én udbygger kravene til tabeldesignet. Hver enkelt normalform er afhængig af, at den forrige normalform er opfyldt (for eksempel kræver anden normalform, at første normalform er opfyldt)[16]. Der findes seks normalformer, men ofte bliver kun de første tre brugt, hvilket også vil være tilfældet i denne rapport[17].

Første normalform

Den første normalform lægger grund for nogle helt basale krav til tabellerne. Det første krav til tabellen er, at alle kolonner indeholder atomare værdier. Det vil sige, at hver kolonne kun indeholder én værdi. Det vil sige, at en adressekolonne, der både indeholder vejnavn og postnummer, ikke er atomar. Samtidigt siger første normalform, at tabellen ikke må indeholde repeterende grupper. Et eksempel på en repeterende gruppe er, at en medarbejdertabel indeholder fem kolonner, hvori der kan skrives de børn, medarbejderen har. Det sidste der kræves, for at en tabel er på første normalform, er at den skal indeholde en primærnøgle. Det vil sige, at der skal udpeges en eller flere kolonner, der entydigt kan identificere hver enkelt række i tabellen.

Anden normalform

Den anden normalform bygger videre på første normalform, hvorfor det er krævet, at tabellen er på første normalform. Derudover siger anden normalform, at alle ikke-nøgle attributter skal være fuldt funktionelt afhængige af primærnøgle.

[15] [SKS01], side 3
[16] [SKS01], side 257
[17] [Kin01], side 393-395

Det vil sige, at hvis en kolonne kun er afhængig af en del af primærnøglen, så skal denne flyttes ud i en anden tabel.

Tredje normalform

Igen bygger tredje normalform videre på de andre normalformer, så derfor kræver tredje normalform, at tabellen er på anden normalform, før normaliseringen starter. Tredje normalform siger derudover, at ingen ikke-nøgle attributter er transitivt afhængige af primærnøglen.

Dette vil sige, at hvis en kolonne er afhængig af en eller flere kolonner, som ikke er primærnøglen, så skal denne kolonne flyttes ud som en anden tabel. Med andre ord, så skal alle kolonner kun være bestemt af primærnøglen. Det klassiske eksempel på dette er postnummer og bynavn-relationen, hvor bynavnet er bestemt ud fra postnummeret, og postnummeret er bestemt ud fra primærnøglen.

Disse krav til overholdelse af normalformer vil blive anvendt under implementeringen af registreringssystemet i databasen. Herved opnås en databasestruktur, hvor redundans undgås og hvor integritet af data sikres. Designovervejelser og beslutninger vil blive taget nærmere op i Afsnit 4.7, hvor databasen designes.

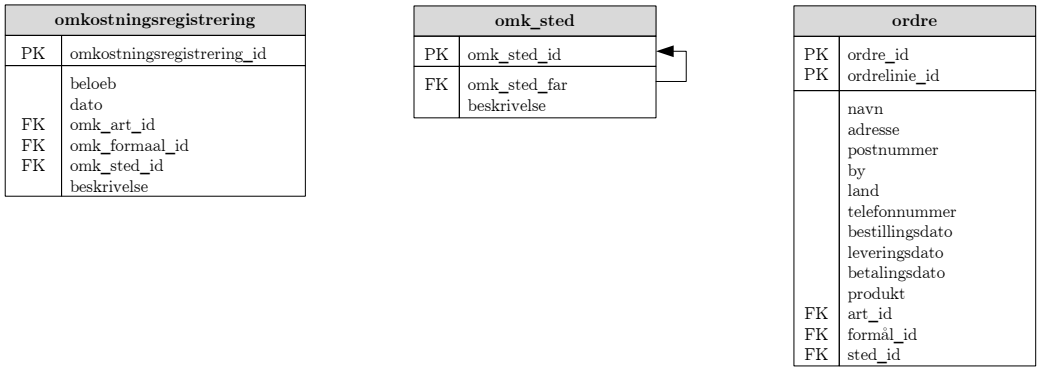
4.7 Tabeldesign

I dette afsnit vil designet af de forskellige tabeller, der skal bruges for at lave de forskellige registreringer i henhold til variabilitetsprincippet, blive gennemgået. Først vil der blive arbejde på omkostningssiden, og derefter vil dette blive koblet sammen med indtægtssiden med udgangspunkt i ER-diagrammet fra Afsnit 4.4 på side 27.

4.7.1 Omkostningssiden

Den helt centrale tabel er `omkostningsregistrering`. Den første kolonne i denne tabel er **omkostningsregistrerings_id** kolonnen, som er primærnøgle i tabellen. Denne primærnøgle er en syntetisk nøgle, som styres af en sekvens i Oracle, da der ikke umiddelbart findes nogen logisk nøgle, der kan bruges. Den næste kolonne, **beloeb**, registrerer omkostningens størrelse. Den næste kolonne registrer tidspunkt og dato for omkostningen. De næste tre kolonner, **art_id**, **formaal_id** og **sted_id**, indeholder fremmednøgler til de tre arts-, formåls- og stedshierarki som beskrevet i Afsnit 2.1 på side 13. Den sidste kolonne er en beskrivelse af den omkostning, der er blevet udført. Hele tabellen kan ses i Figur 14(a) på næste side.

Det næste, der skal laves, er de tre hierarki-tabeller, der repræsenterer arts-, formåls- og steds-hierarkierne. De tre første kolonner er ens i opbygning i de tre tabeller, så derfor vil disse kun blive gennemgået for stedsdimensionen, `omk_sted`. Den første kolonne i tabellen



(a) Omkostningsregistreringstabellen (b) Stedsdimensionstabellen (c) Ordretabellen

Figur 14: Udsnit af tabeldesignet

er **omk_sted_id** kolonnen, som igen er en syntetisk primærnøgle. For at tabellen skal kunne repræsentere et hierarki, skal der være en kolonne, der referer til primærnøglen i tabellen selv. Dette kaldes en father-son-relation, hvor kolonnen **omk_sted_far** er en fremmednøgle til **omk_sted_id** kolonnen i samme tabel. På denne måde vil der i **omk_sted_far** kolonnen kunne angives, hvilken afdeling, der ligger et niveau højere i hierarkiet. Den sidste kolonne er **beskrivelse**, som indeholder en tekstuel beskrivelse af det enkelte sted. Stedsdimensionen er vist i Figur 14(b)

Artsdimensionen, `omk_art`, indeholder de samme kolonner som stedsdimensionen. Derfor vil denne tabel ikke blive gennemgået yderligere.

Den sidste dimension på omkostningssiden er formålet, `omk_formaal`. Igen er strukturen den samme som på de tidligere dimensioner. Derudover indeholder tabellen også kolonnen **produkt_id**, som er en fremmednøgle til primærnøglen i `produkt`-tabellen. Denne relation vil blive brugt senere til at koble omkostninger sammen med indtægterne. Denne kolonne vil være udfyldt når et formål kan relateres til et givet produkt. Dette medfører også, at kolonnen kan indeholde *NULL* referencer, hvilket senere vil gøre, at der ikke umiddelbart kun kan benyttes et *natural join* mellem `omk_formaal` og `produkt`-tabellen.

4.7.2 Indtægtssiden

For at kunne sammenholde omkostninger med indtægter kræver det, at salg til kunderne registreres i grundregnskabet. Derfor er den centrale tabel på indtægtssiden `ordre`. Denne tabel indeholder ordrenummer, kundedata, varebeskrivelse samt fremmednøgler til de tre dimensioner. Denne tabel kan ses i Figur 14(c). Dog ses det tydeligt fra figuren, at dette tabeldesign ikke overholder de førnævnte normalformer, hvorfor denne skal normaliseres ud til flere tabeller, før den bør tages i brug.

Kundedata og varekøbene redundante og i strid med anden normalform, idet de ikke er fuldt funktionelle afhængige af primærnøglen. Kundedata er kun afhængige af **ordre_id**.

Varekøb er kun afhængig af **ordrelinie_id**. Derfor skal disse data flyttes ud til to nye tabeller, henholdsvis `kunde` og `ordrelinie`. Samtidig flyttes **ind_sted_id** til `kunde`-tabellen, og **ind_art_id** flyttes med til `ordrelinie`-tabellen. Dette skyldes, at stedshierarkiet er direkte afhængigt af den enkelte kunde. Det samme gælder for artshierarkiet der er direkte afhængigt af den enkelte ordrelinie. I den nye kunde tabel bryder kolonnebynavn nu med tredje normalform, idet den er transitiv afhængig af primærnøglen, da denne er afhængig af **postnummer** og **land**. Derfor flyttes **postnummer**, **land** og **bynavn** ud i en ny tabel `by_land` med den syntetiske primærnøgle **by_land_id**. Samtidig bliver der lavet en fremmednøgle i `kunde`-tabellen, der hedder **by_land_id**.

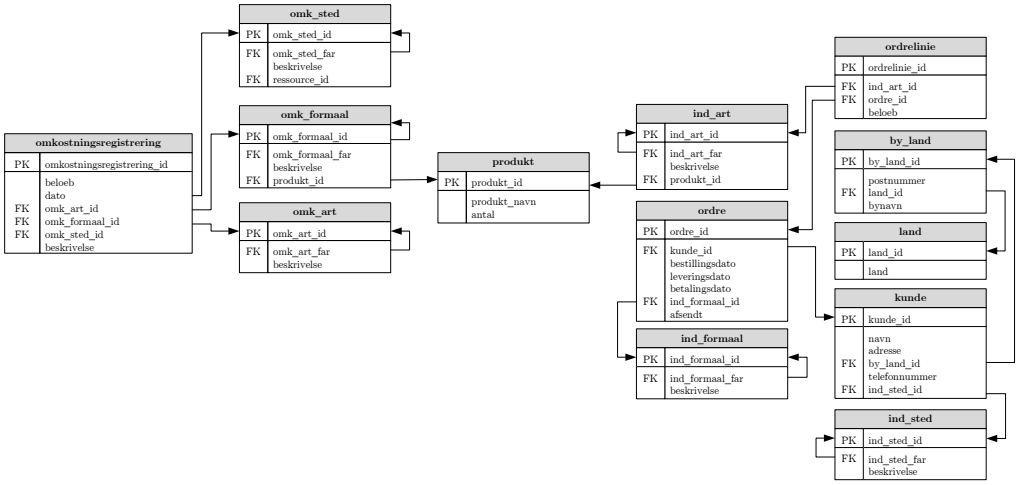
I tabellen `by_land` findes nu kolonnen **land**. Teoretisk set bryder denne ikke med hverken første, anden eller tredje normalform, da den er fuldt funktionel afhængig af primærnøglen, og den er ikke transitiv afhængig af primærnøglen, da hverken postnummer og bynavn enkeltvis eller tilsammen kan afgøre landet. Vi har valgt at flytte landet ud i en separat tabel alligevel, idet dette minimerer mulighederne for fejlindtastninger betydeligt. Denne tabel vil komme til at indeholde forudindtastede data, som yderst sjældent skal redigeres. Havde landet forblevet i `by_land`-tabellen, ville det have været muligt for brugerne at stave landenavnet forkert, og dermed skabe inkonsistens i databasen. Herved ville udtræk baseret på lande optræde unøjagtig.

De tre dimensioner for indtægter har samme opbygning som dimensionerne for omkostninger blot med den ene ændring, at fremmednøglen **produkt_id** er flyttet til dimensionen `ind_art`. I forbindelse med stedsdimensionen bør der rettes fokus på, at der kan opstå redundante oplysninger, idet `kunde`-tabellen indeholder et **postnummer**. Det samme kunne logisk set fremgå af stedsdimensionen, men da denne er en generisk struktur, som baserer sig på empiri, er det ikke sikkert, at **postnummer** vil forekomme i stedsdimensionen, og derfor er det besluttet at medtage **postnummer** i `kunde`-tabellen.

Tabellen `produkt` bruges til at binde omkostningssiden og indtægtssiden sammen med. På omkostningssiden refereres de formål, der kan henføres til et produkt, til det enkelte produkt i `produkt`-tabellen. Det samme gøres på indtægtssiden, hvor de indtægtsarter, der kan henføres til et produkt, refereres til `produkt`-tabellen. Det samlede tabeldesign kan ses i Figur 15 på den følgende side, som også afspejler det udarbejdede ER-diagram fra Afsnit 4.4 på side 27. SQL til hele tabeloprettelsen kan ses i Bilag A på side 93.

4.8 SQL

I dette afsnit vil et udsnit af de forskellige SQL-sætninger til henholdsvis at oprette tabellerne og indsætte i disse blive illustreret. SQL-sætningerne afspejler designet af databasen, samt forbereder databasen til registrering efter variabilitetsprincippet. Vi har valgt at designe databasen i forhold til det gennemgående eksempel, for at demonstrere hvordan registreringen kan foregå på det praktiske niveau.



Figur 15: Komplet tabeldesign

4.8.1 Tabeloprettelse

I dette afsnit gennemgås udsnit af de SQL-sætninger, der skal bruges for at oprette tabellerne. Vi har valgt enkelte tabeller ud for at beskrive de vigtigste funktioner i SQL-sætningerne.

SQL-sætning 1 viser oprettelse af `formål`-tabellen på omkostningssiden. Linierne 2–5 opretter de forskellige kolonner som beskrevet i Afsnit 4.7 på side 31. I Linie 2 bliver den primære nøgle defineret. Det skal bemærkes, at der i tabeloprettelsen benyttes datatypen *varchar2* i stor udstrækning. Dette er et bevidst valg, hvilket kan lette arbejdet med registreringen i databasen. De eneste steder, der ikke vælges at benytte *varchar2*, er, hvis kolonnen skal bruges til at regne på eller skal joines. Der er tilføjet to *constraints*, som definerer fremmednøglerne i tabellen. Linie 6 laver father-son-relationen mellem de to kolonner i denne tabel. Derudover laver Linie 7 fremmednøglen til `produkt`-tabellen. Yderligere bliver der i Linie 8 kontrolleret for, at der ikke refereres til den samme række i tabellen.

```
1 CREATE TABLE OMK_FORMAAL (
2       OMK_FORMAAL_ID      NUMBER(12) PRIMARY KEY,
3       OMK_FORMAAL_FAR     NUMBER(12),
4       BESKRIVELSE         VARCHAR2(200),
5       PRODUKT_ID          NUMBER(12),
6       CONSTRAINT OMK_FORMAAL_FATHER_SON FOREIGN KEY (OMK_FORMAAL_FAR) REFERENCES
            OMK_FORMAAL(OMK_FORMAAL_ID),
7       CONSTRAINT OMK_FORMAAL_PRODUKT FOREIGN KEY (PRODUKT_ID) REFERENCES PRODUKT(
            PRODUKT_ID),
8       CONSTRAINT OMK_FORMAAL_INGENREKURSIV CHECK(OMK_FORMAAL_FAR <> OMK_FORMAAL)
9 );
```

SQL-sætning 1: Oprettelse af formålshierarkiet

SQL-sætning 2 på næste side viser kommandoen til oprettelse af `Ordre`-tabellen, som ligger på indtægtssiden. Her ses det i Linie 4–6, at datatypen *Date* benyttes til kolonnen. Herved opnås mulighed for at benytte de forskellige funktionaliteter, Oracle tilbyder

med henblik på datoformatet. I mange tilfælde bliver formatet på *Date* angivet i SQL. Dette har vi valgt at undlade, idet der vil blive kontrolleret, at formatet indtastes korrekt i den grafiske brugerflade. De to constraints på Linierne 8–9 er af samme type, som er beskrevet i SQL-sætning 1 på modstående side. Derudover er der tilføjet en ekstra CHECK-constraint i Linie 10, hvor der bliver kontrolleret, at **leveringsdato** ikke forekommer før **bestillingsdato**. Dette er sket for at forhindre logiske indtastningsfejl i databasen.

```
1  CREATE TABLE ORDRE (
2  ORDRE_ID              NUMBER(12) PRIMARY KEY,
3  KUNDE_ID              NUMBER(12),
4  BESTILLINGSDATO       DATE,
5  LEVERINGSDATO         DATE,
6  BETALINGSDATO         DATE,
7  IND_FORMAAL_ID        NUMBER(12),
8  CONSTRAINT ORDRER_KUNDE FOREIGN KEY (KUNDE_ID) REFERENCES KUNDE(KUNDE_ID),
9  CONSTRAINT ORDRER_FORMAAL FOREIGN KEY (IND_FORMAAL_ID) REFERENCES IND_FORMAAL(
       IND_FORMAAL_ID),
10 CONSTRAINT LEVERING_EFTER_BESTILLING CHECK (LEVERINGSDATO >= BESTILLINGSDATO)
```

SQL-sætning 2: Oprettelse af `Ordre`-tabellen

4.8.2 Indsættelse i tabellerne

Efter at databasestrukturen er blevet oprettet skal der indsættes data i modellen. Der vil her blive indsat data til at kunne illustrere funktionaliteten af registreringssystemet. Tabeller som `omkostningsregistrering` samt `ordre` skal reelt først fyldes ud med data fra systemet, når dette er implementeret. Før dette kan virke, skal der indsættes data i hjælpetabeller for at opnå funktionalitet i registreringssystemet. For at gøre dette på en struktureret måde, benyttes det gennemgående eksempel. I Figur 1 på side 14 og Figur 4 på side 18 ses, hvordan hierarkierne i det gennemgående eksempel ser ud. Hierarkierne skal oprettes ud fra figurerne, og SQL-sætningen skal derfor afspejle designet i disse. Et eksempel på oprettelse af hierarkier kan ses i SQL-sætning 3 på næste side.

SQL-sætningen 3 på den følgende side illustrerer, hvordan der bliver indsat data til håndtering af hierarkier i registreringssystemet. De indsatte data stammer fra Figur 1(c) på side 14. I Linie 1 startes med at oprette den overordnede fader til hele hierarkiet. Dette ses ved, at der på anden parameter indsættes værdien *NULL*. Denne værdi bruges senere logisk af databasen og applikationerne til at identificere topniveauet i hierarkiet. Herefter benyttes første og anden parameter til at beskrive hierarkisammenhængen imellem alle formålene. Vi har valgt at benytte en afstand mellem værdierne, for at have mulighed for udvidelse af systemet. Beskrivelsen af, hvad brugeren ser, når denne arbejder med hierarkiet, bliver betegnet i tredje parameter. Ellers er `omk_formål` lidt speciel, idet det er denne tabel, som bruges i forbindelse med sammenkædningen af indtægts- og omkostningssiden. Dette ses ved fjerde parameter, hvor der kan angives et **produkt_id**. Hvor dette ikke benyttes, bliver der bare indsat værdien *NULL*. Men ved de tre produkter i nederste niveau af hierarkiet er der benyttet en fremmednøgle til tabellen `produkt`. Dette er udover `ind_art` den eneste af hierarkitabellerne, som anvender denne kobling.

```
1 INSERT INTO OMK_FORMAAL VALUES (5,NULL,'Alle formål',NULL);
2 INSERT INTO OMK_FORMAAL VALUES (50,5,'Produktion Generelt',NULL);
3 INSERT INTO OMK_FORMAAL VALUES (100,50,'Stole',NULL);
4 INSERT INTO OMK_FORMAAL VALUES (150,100,'Ørn', 1);
5 INSERT INTO OMK_FORMAAL VALUES (200,100,'Due',2);
6 INSERT INTO OMK_FORMAAL VALUES (250,50,'Borde',NULL);
7 INSERT INTO OMK_FORMAAL VALUES (300,250,'Falk',3);
8 INSERT INTO OMK_FORMAAL VALUES (350,5,'Vedligeholdelse',NULL);
9 INSERT INTO OMK_FORMAAL VALUES (400,5,'Salg',NULL);
```

SQL-sætning 3: Indsættelse af formål

Indsætningen af data i hierarkitabellerne `omk_art`, `omk_sted` på omkostningssiden samt `ind_art`, `ind_formål` og `ind_sted` foregår på samme måde som i SQL-sætning 3, og derfor vil disse ikke blive beskrevet yderligere. På samme måde sættes data ind i `produkt`, `by_land`, `land` samt `kunde` ved hjælp af simple SQL-sætninger. Dette gøres for at have et grundlag for test af systemet, men der vil typisk blive oprettet flere poster i disse tabeller i forbindelse med brugen af registreringssystemet.

Den sidste SQL-sætning, der bliver beskrevet, er et eksempel på en registrering i `omkostningsregistrering`. Dette bliver gjort for at forklare princippet i, hvordan en omkostning bliver registreret. Dette er typisk ikke noget, som vil blive gjort under oprettelse af databasen, men i stedet vil SQL-sætningen blive brugt senere under implementeringen af den grafiske brugerflade. Sætningen bliver beskrevet her, idet den er essentiel for funktionaliteten af systemet. SQL-sætningen 4 består af en INSERT kommando. Der bliver kaldt en forudoprettet sekvens, som bliver benyttet til automatisk at generere et fortløbende *omkostningsregistrerings_id* til primærnøgle. Derved sikres, at brugeren ikke skal tænke på at nummerere sine registreringer på omkostninger, og endvidere opnås en sikkerhed imod indsættelsesfejl grundet ikke-unikke primærnøgler. Løsningen med automatisk fortløbende sekvenser bliver benyttet i bred udstrækning i hele databasen på de steder, hvor behovet er. I tredje, fjerde, og femte parameter indtastes de forskellige referencer til omkostningernes art, formål og sted. Dette kan umiddelbart virke lidt overskueligt, idet det kan være svært at overskue hierarkierne ud fra tabelindsætningen. Derfor skal det understreges, at dette skal implementeres i den grafiske brugerflade, således at disse nøgler fremkommer lang mere naturlige. Herved bliver det nemt for brugeren at benytte registreringssystemet.

```
1 INSERT INTO OMKOSTNINGSREGISTRERING
2 VALUES(OMKOSTNINGSREGISTRERING_SEQ.NEXTVAL,3504,300,100,200);
```

SQL-sætning 4: Indsættelse i omkostningsregistrering

4.8.3 SQL til den grafiske brugergrænseflade

Efter sætningerne til tabeloprettelse og indsættelse af data er blevet beskrevet, følger her et eksempel på en SQL-sætning, som skal bistå den grafiske brugergrænseflade i registreringssystemet. Disse sætninger skal primært benyttes til at trække data ud fra tabeller, samt skabe den nødvendige sammenhæng mellem disse data.

SQL-sætning 5 viser, hvordan overskudsopgaven udregnes. SQL-sætningen består af to *subselects* på Linie 2 og 4. Begge subselects udregner henholdsvis de samlede indtægter og omkostninger. På Linie 3 trækkes resultatet fra de to subselects fra hinanden. WHERE-delen udvælger data mellem de to valgte datoer (`:P30_FRADATO` og `:P30_TILDATO`), som refererer til de to felter i Application Express. Der bliver benyttet join kommandoer, som bruges til at sammenkæde de enkelte tabeller ved hjælp af fremmednøgler. Det bør her bemærkes, at datoerne konverteres til et andet datoformat. Dette skyldes, at Application Express's datepicker ikke kan vælge et format, der kan bruges direkte i SQL-sætningerne. Læg yderligere mærke til, at der i Linie 5 udvælges fra `dual`-tabellen. Dette skyldes at resultatet kommer fra en udregning, og derfor skal den yderste select ikke udvælge fra nogen decideret tabel.

```
1  SELECT ( NVL((SELECT SUM(BELOEB) FROM ORDRELINIE NATURAL JOIN ORDRE
2  WHERE BETALINGSDATO>=TO_DATE(:P30_FRADATO, 'YYYY.DD.MM') AND
3  BETALINGSDATO <=TO_DATE(:P30_TILDATO, 'YYYY.DD.MM')),0)) -
4   (NVL((SELECT SUM(BELOEB) FROM OMKOSTNINGSREGISTRERING WHERE DATO
5  >=TO_DATE(:P30_FRADATO, 'YYYY.DD.MM') AND DATO
6  <=TO_DATE(:P30_TILDATO, 'YYYY.DD.MM')),0)) AS OVERSKUD FROM DUAL
```

SQL-sætning 5: SQL til overskudsopgaven

SQL-sætning 6 viser, hvordan træet til Figur 18 på side 41 opbygges. SQL-sætningen indeholder flere selects. Den første select udvælger saldoen på en given konto. Den anden select summer saldoen på en konto med summen af dens underkonti.

```
1  SELECT OMK_FORMAAL_ID id,
2    OMK_FORMAAL_FAR pid,
3    OMK_FORMAAL_ID || ' - ' || BESKRIVELSE || ' - Saldo: ' ||
4     NVL((SELECT SUM(BELOEB) FROM OMKOSTNINGSREGISTRERING WHERE
5       OMKOSTNINGSREGISTRERING.OMK_FORMAAL_ID=OMK_FORMAAL.OMK_FORMAAL_ID),
6       0)||
7      'Sum: '||
8     NVL((SELECT SUM(BELOEB) FROM OMK_FORMAAL INDERST LEFT JOIN
9      OMKOSTNINGSREGISTRERING ON
10       INDERST.OMK_FORMAAL_ID=OMKOSTNINGSREGISTRERING.OMK_FORMAAL_ID
11     CONNECT BY PRIOR INDERST.OMK_FORMAAL_ID = INDERST.OMK_FORMAAL_FAR
12     START WITH INDERST.OMK_FORMAAL_ID = OMK_FORMAAL.OMK_FORMAAL_ID),0)
13     name,
14   NULL link,
15   NULL a1,
16   NULL a2
17   FROM OMK_FORMAAL
```

SQL-sætning 6: SQL til overskudsopgaven

På Linie 4–5 er der en subselect, der beregner saldoen på kontoen, givet ved den yderste forespørgsel. Dette gøres ved at aggregere ved hjælp af funktionen `SUM` på tabellen `omkostningsregistrering`, som indeholder beløbet for de enkelte omkostningsregistreringer. På Linie 8–12 er endnu en subselect, der beregner saldoen for denne konto samt alle underkonti. Dette gøres ved, først at udføre en rekursiv forespørgsel i tabellen `omk_formaal` for senere at joine den med `omkostningsregistrering`, hvor selve omkostningen registreres. Sidst aggregeres over beløbet ved hjælp af funktionen `SUM`.

Vi har valgt ikke at inddrage yderligere SQL-sætninger til funktionaliteten i rapporten,

idet de typisk anvender samme metoder som i SQL-sætning 5 på foregående side.

4.9 Brugergrænseflade

Efter databasedesign samt oprettelse af databasen er udført, skal der udarbejdes en grafisk brugergrænseflade til registreringssystemet. Udarbejdelsen af den grafiske brugergrænseflade kan betegnes som en mulighed for brugeren for, at kunne benytte dette uden at tænke på den funktionalitet, som ligger bagved. Det er derfor vigtigt, at en brugergrænseflade er logisk opbygget, samtidigt med den er let at bruge selv med mindre kendskab til systemet.

4.9.1 Design af brugergrænseflade

Brugergrænsefladen til systemet bliver udarbejdet i Oracles eget værktøj, Application Express. Værktøjet giver mulighed for at kunne oprette en webbaseret løsning ved hjælp af en række prædefinerede guides og modeller. Det giver den fordel, at det forholdsvis hurtigt er muligt at udarbejde en funktionel brugergrænseflade til sin database uden at have stor teknisk indsigt i programmeringssprog. Oracle databasen giver adgang til, at der kan benyttes andre teknologier til implementering af brugergrænsefladen, som for eksempel .NET, ASP og PHP. Herved vil det være muligt for designeren af systemet at tilpasse brugergrænsefladen ned til mindste detalje.

Indtil nu har adgangen til registrering i databasen været præget af simple SQL-sætninger til oprettelse af tabeller og indsættelse af data. Disse operationer vil blive mulige i brugergrænsefladen, men der vil ligeledes opstå et behov for at udføre forespørgsler på allerede registrerede data. Disse forespørgsler skal hjælpe med til at udføre de tre opgaver i Vagn Madsens regnskabsteori, som løses i denne rapport. SQL funktionaliteten i disse opgaver er beskrevet i Afsnit 4.8.3 på side 36.

Et nøgleelement i et IT-system er dets brugervenlighed. Dette omhandler både selve udseendet, og hvordan interaktionen med systemet foregår. I "Oracle Application Express" kan der vælges mellem en lang række skabeloner og farver. I en af dem er den gennemgående farve rød. Farver giver brugeren forskellige associationer og oplevelser af systemet. Ifølge [Mol94] vil netop rød give brugeren højere blodtryk[18]. Derfor bør der til rutine-arbejde vælges en anden, mere neutral farve. Dette udelukker ikke brugen af rød, men rød bør bruges med omtanke. I visse situationer kan det være hensigtsmæssigt, eksempelvis hvis der opstår fejl i systemet. Derfor er der valgt at benytte en skabelon til brugergrænsefladen med et mere konservativt farveskema.

[18][Mol94], side 70

4.9.2 Præsentation af brugergrænseflade

I det følgende afsnit beskrives, hvordan de enkelte opgaver i brugergrænsefladen implementeret. Dette vil blive illustreret ved hjælp af skærmbilleder, hvor funktionalitet samt overvejelser omkring designet vil blive kommenteret og diskuteret.

Registreringsopgaven

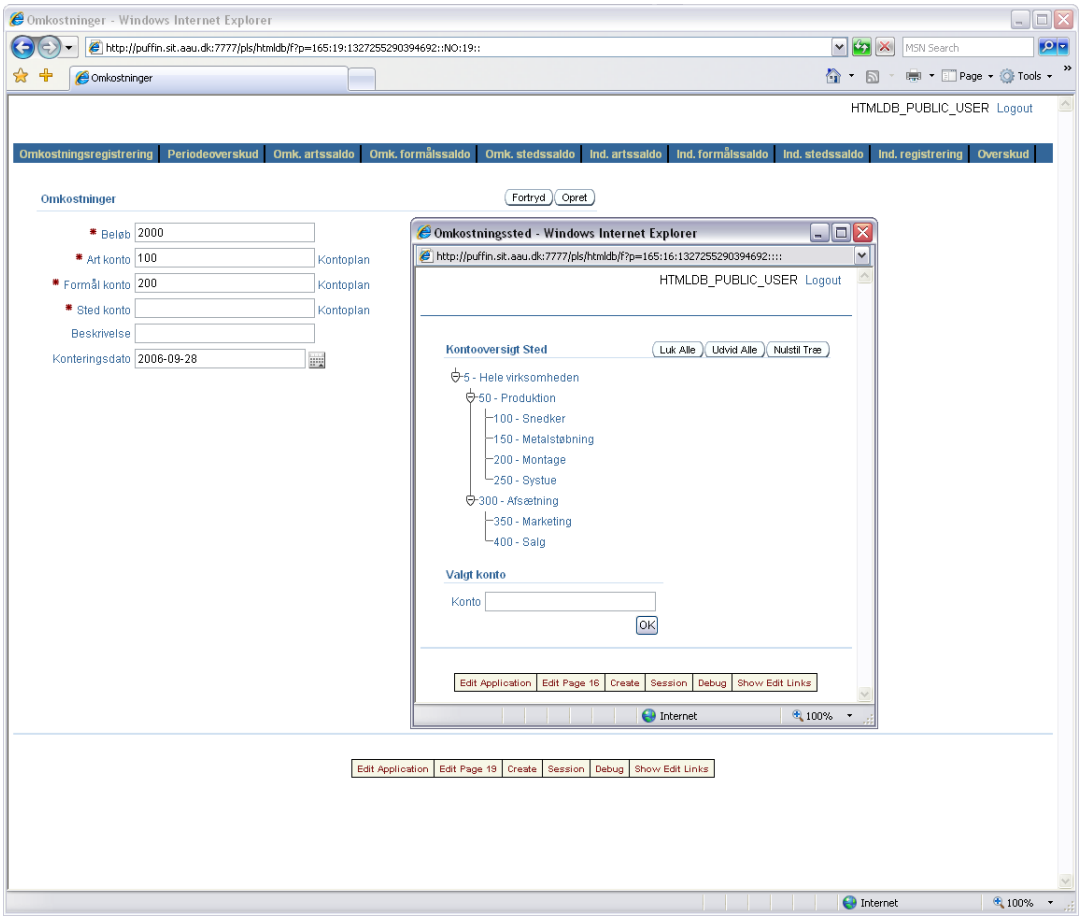
Figur 16 på næste side viser, hvordan omkostningsregistreringsopgaven er repræsenteret i brugergrænsefladen. Som det kan ses, indeholder skærmbilledet seks tekstfelter. I første tekstfelt skal beløbet registreres. I de næste tre felter, skal de tre konti for henholdsvis art, sted og formål skrives. Her er det muligt at indtaste en værdi direkte for en konto i hierarkistrukturen, hvilket muliggør, at den erfarne medarbejder kan indtaste kontonummeret fra hukommelsen. Ved siden af hvert af disse felter er der et link, der får en side til at "poppe" op. Der skal den korrekte konto ud fra det foruddefinerede hierarki vælges. Dette er til hjælp, hvis der er tvivl om, hvor omkostningen skal registreres. Samtidig øges brugervenligheden. Denne popup kan også ses af Figur 16 på den følgende side. Efter at der er valgt en konto i popup billedet, bliver denne værdi automatisk overført til det primære vindue. Det skal bemærkes, at denne løsning i Application Express er implementeret ved hjælp af JavaScript, da Application Express ikke umiddelbart understøtter træer som popup.

Overskudsopgaven

Figur 17 på side 41 viser, hvordan overskudsopgaven løses. Dette skærmbillede indeholder to datofelter, der afgør den periode, hvori overskuddet/underskuddet skal udregnes. Når perioden er angivet, trykkes der blot på "beregn"-knappen, hvorefter resultatet vil blive vist i rapporten nedenunder. Der kan her argumenteres for, at der kan oprettes en række andre rapporter, som vil kunne øge funktionaliteten af systemet yderligere. Et overskud på produktgrupper vil være en mulighed og vil kunne implementeres med få tilpasninger af den eksisterende forespørgsel.

Prisfastsættelses- og kalkulationsopgaven

Figur 18 på side 41 viser et eksempel på et af de skærmbilleder, som skal bruges for at udføre prisfastsættelses- og kalkulationsopgaven. Den viser kontoplanen i hierarkiform, hvor `saldo` betegner saldoen på den aktuelle konto, og `sum` betegner de summerede konti (inklusive kontoen selv), der ligger under kontoen i hierarkiet. Herved bliver det muligt for virksomheden at finde sine omkostninger eller indtægter indenfor hvert enkelt område i hierarkistrukturen. På denne måde kan virksomheden let sammenholde, hvilke omkostninger der har været indenfor hvert enkelt område. Dette skal derefter holdes op mod en produktionskapacitet og en given periode for at kunne lave kalkulationer. Der



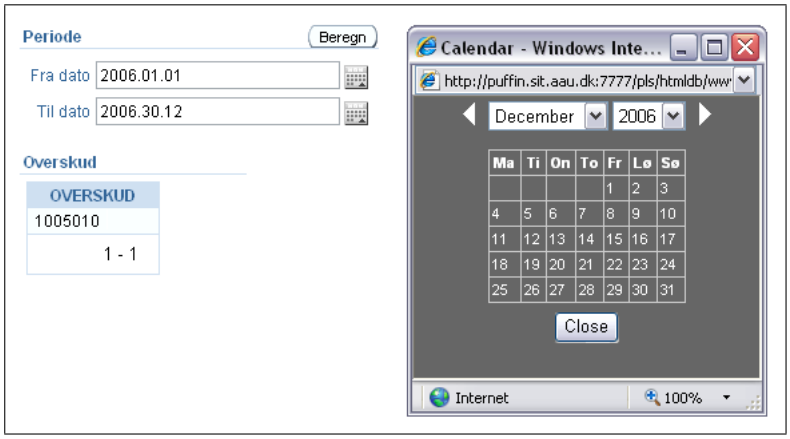
Figur 16: Registrering af en omkostning med sted-hierarkiet vist som træ

vil på denne side kunne laves en række valgmuligheder for at automatisere opgaverne for brugeren. Dog indeholder træerne informationen, som skal bruges for at løse opgaven.

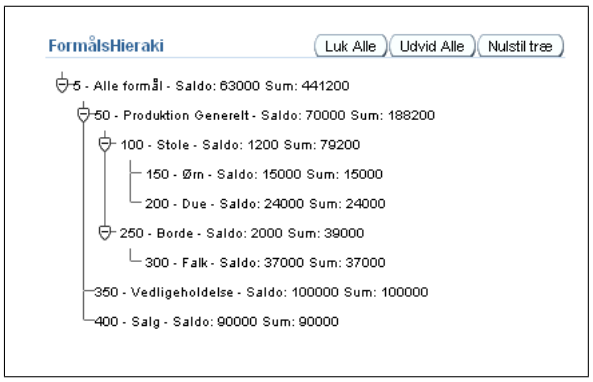
4.9.3 Automatisering af registreringer

Et spørgsmål, som har rejst sig under implementeringen af brugergrænsefladen til systemet, er, om det er muligt at automatisere registreringen af omkostninger. Det vil være en fordel for brugeren, hvis der var mulighed for, at de rutinemæssige registreringer kunne vælges direkte fra en liste. Dette vil sikre en langt hurtigere arbejdsgang i forbindelse med registreringsprocessen, da de tre dimensioner vil være prædefinerede i omkostningen. Ligeledes vil det minimere muligheden for, at der opstår fejl i registreringen, idet dimensionerne er angivet på forhånd. Rent teknisk kunne automatiseringen af registreringer løses ved at lave en tekstboks med valgbare værdier i brugergrænsefladen og en tabel med disse i databasen.

I forhold til diskussionen giver en øget automatisering af systemet mindre fleksibilitet. Hvis alle registreringer af omkostninger er automatiserede, vil det ikke være muligt at



Figur 17: Rapport der viser overskuddet i en given periode



Figur 18: Formålshierarkiet med konto-saldo samt sum af undertræ

registrere usædvanlige omkostninger korrekt. Derfor vil det altid være hensigtsmæssigt, at det manuelt er muligt at vælge alle dimensioner under registreringen.

Hvis dette var tilfældet, var der ingen grund til at have forskellige konti, som registreringen skal henføres til. Derfor ville det kræve, at databasedesignet redesignes, hvis der ønskes denne løsning.

5 Opsummering

I oplægget til dette hovedafsnit blev der valgt at fokusere omkring selve variabilitetsprincippet, og hvordan det kan implementeres i et IT system. Herved er arbejdet blevet baseret på, hvilke muligheder denne tankegang giver omkring omkostningsregistreringen. Derved har det også været nødvendigt at acceptere de svagheder, som variabilitetsprincippet har i sin oprindelige form. For at kunne benytte systemet til at løse regnskabsopgaverne har det dog været nødvendigt at implementere en ekstra dimension i form af indtægter. Herved mener vi, at det modificerede variabilitetsprincip bliver langt mere fleksibelt og løser sine opgaver i højere grad.

Implementeringen af indtægtssiden i variabilitetsprincippet er sket umiddelbart uden at have nogen konkret teori til rådighed omkring emnet. Derfor er indtægtsimplementeringen sket efter samme princip som omkostningsregistreringen. Implementeringen af indtægter vil typisk også ske på anden vis, men den version, som optræder i denne rapport, løser de opstillede problemstillinger tilfredsstillende.

Der blev fra start fokuseret på at løse tre af Vagn Madsens opgaver i forbindelse med regnskabet. Umiddelbart har det logiske tabeldesign i systemet dannet grundlaget for løsbarheden af opgaverne. Det kan fastslås, at der med det traditionelle variabilitetsprincip var mulighed for at løse opgave et og tre, men indtægtsregistreringen giver langt større muligheder i forbindelse med implementeringen af opgave to. Forslagene til den grafiske brugergrænseflade i systemet illustrerer funktionaliteten og viser, at de tre opgaver kan løses. Der kan argumenteres for, at det ville være hensigtsmæssigt at udvide med ekstra funktionalitet, men da systemet kan betragtes som et basissystem, kan det senere tilpasses direkte efter den virksomhed, som ønsker at benytte systemet. Systemet kan med få tilpasninger ligeledes udvides til at kunne løse alle Vagn Madsens seks opgaver.

I arbejdet med implementeringen af systemet er der opstået en del forståelse for design og kodning af sådanne systemer. Til baggrund for denne proces er der blevet udarbejdet en række diagrammer for at hjælpe med det egentlige design af systemet. Det er ikke alle disse diagrammer, som har haft en direkte forbindelse til den videre implementering

af systemet. Virksomheden vil dog kunne benytte disse diagrammer til at analysere og forstå hvilke arbejdsgange der er i virksomheden. Herved sikres det, at der opnås en beskrivelse og afdækning af det system, der skal implementeres, således at det direkte afspejler virksomhedens behov og krav.

I arbejdet med design har det været en prioritet, at applikationen skulle være brugervenlig. Således har udfordringen bestået i at kunne få det underliggende design af systemet til at understøtte de funktioner, som registreringssystemet skal løse. Oracles Application Express har vist sig at være hurtigt og let at bruge til udvikling af sådanne applikationer. Herved menes, at det er muligt for utrænede at opnå funktionelle systemer på kort tid, hvis blot det underliggende design af databasen er korrekt. Dette gælder hvis Application Express har skabeloner til yderligere funktionalitet, der ønskes. Det er erfaret, at der kan opstå vanskeligheder, hvis der ønskes speciel funktionalitet, som ikke er prædefineret. Her kan programmet virke yderst vanskeligt at arbejde med, og udarbejdelse af løsningerne tager oftest en del forsøg og tid. Her kan der argumenteres for, at andre redskaber på markedet vil kunne løse opgaven bedre. Dog vil disse kræve, at der startes med at indhente viden og knowhow omkring brugen, før disse kan benyttes til egentligt design af applikationer. Efterfølgende vil det dog være en fordel, idet der bliver langt større frihed til at implementere og øge muligheden for funktionalitet i systemet.

Systemet, der er beskrevet i denne rapport, vil kunne benyttes til omkostningsregistrering i virksomheder i dag. Dog har udviklingen gjort, at nye teknologier med fordel vil kunne benyttes til at forbedre dette yderligere. Herved vil det være hensigtsmæssigt at implementere en eller flere metoder til fordeling af omkostninger, for at kunne udarbejde mere præcise rapporter vedrørende virksomhedens omkostninger.

II

ABC

6 Indledning

6.1 Introduktion

I Hovedafsnit I blev en løsning til variabilitetsprincippet med indtægter udarbejdet. Herunder blev der udarbejdet en analyse af Vagn Madsens seks regnskabsopgaver og af hvilke opgaver, som variabilitetsprincippets registrering kunne løse. Efter implementeringen af indtægter i registreringen blev det muligt at løse alle seks regnskabsopgaver. Dette illustreres i Figur 19. Figuren er her udvidet til at vise, hvad en mulig fordeling af kapacitetsomkostningerne vil afhjælpe.

- ÷ - kan ikke løses
- (✓) - kan løses i nogen grad
- ✓ - kan løses fuldt ud

	Omkostninger	*Indtægter*	*Fordeling*
1. Registreringsopgaven	✓	✓	✓
2. Overskudsopgaven	÷	(✓)	(✓)
3. Kalkulations- og prisfastsættelsesopgaven	(✓)	(✓)	✓
4. Kontrolopgaven	✓	✓	✓
5. Alternativopgaven	÷	(✓)	(✓)
6. Budgetopgaven	÷	(✓)	(✓)

Figur 19: Sammenhæng mellem registreringsprincipper og Vagn Madsens seks opgaver

I forbindelse med løsningen af regnskabsopgaverne ses det, at det kun er registrering-

sopgaven og kontrolopgaven, der kan løses fuldt ud i virksomheden ved hjælp af variabilitetsprincippet. Figur 19 på forrige side skal læses, som de vertikalt stående trin danner baggrund for det næste. Tilføjelsen af indtægter gør, at overskudsopgaven ligeledes bliver mulig at løse. Dog er det stadig ikke muligt at løse de fleste af opgaverne fuldt ud. Dette skyldes hovedsageligt, at omkostninger, som ikke er delt ud på hvert enkelt produkt, er svære at henføre til, hvor de reelt bliver brugt. For at forbedre dette vil en fordeling skulle finde sted, hvilket ikke er tilfældet i variabilitetsregnskabet. Nøglen til at løse dette er fordeling af omkostninger ned på omkostningsobjekter. Der er udarbejdet en lang række værktøjer til fordeling af omkostninger, for eksempel Standard-Cost, Full-Cost og Activity Based Costing. Den fælles tankegang for disse værktøjer er at kunne fordele virksomhedens kapacitetsomkostninger, så de afspejler virkeligheden. Herved vil det blive muligt i højere grad at analysere, hvilke reelle omkostninger virksomhederne har ved et bestemt omkostningsobjekt. Kolonne tre i Figur 19 på foregående side viser, hvordan en fordeling af omkostninger vil influere på muligheden for at løse de seks regnskabsopgaver.

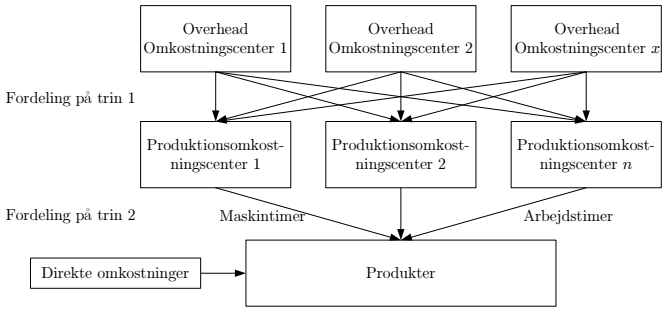
Ud fra Figur 19 på forrige side ses det, at muligheden for at udføre specielt kalkulations- og prisfastsættelsesopgaven bliver forbedret. Baggrunden for dette er, at det bliver muligt at fordele de fleste omkostninger ned på hvert enkelt produkt ud fra fordelingsnøgler, som i så høj grad som muligt afspejler virksomheden. Implementeringen af fordeling vil også kunne forbedre løsningen af alternativ- samt budgetopgaven. Argumentet for dette er også her, at det bliver muligt at opnå et bedre kendskab til virksomhedens omkostningsstruktur. Det er dog ikke muligt fastslå kapacitetsudnyttelsen i virksomheden ved hjælp af fordeling, hvilket kræver yderligere værktøjer. Eksempelvis har Activity Based Costing en videreudvikling til budgettering kaldet Activity Based Budgetting (ABB). Det skal bemærkes, at problemstillingen omkring det, at der i virksomheden bliver produceret nogle produkter for at afsætte andre produkter, ikke bliver behandlet i disse værktøjer. Disse komplementære produkter vil typisk blive behandlet som værende et produkt ved eksempelvis overskudsberegning. Derfor må virksomheden have disse i baghovedet og forholde sig til disse, når der budgetteres.

6.2 Implementering af omkostningsfordeling

Udviklingen i omkostningsstrukturen har gjort, at det i højere grad bliver nødvendigt at fordele kapacitetsomkostningerne i virksomheden ud på, hvor de reelt hører til. Dette hænger sammen med, at den teknologiske udvikling har bevirket, at en omkostning tit er indirekte og bliver forbrugt flere steder i virksomheden. Dette bliver også illustreret i, at virksomhedens produkter bliver mindre håndgribelige, og derved bliver begreber som vejledning, kundepleje og service vigtige faktorer. Disse kan ikke henføres direkte til produkter, og uden en reel fordeling bliver det svært at fastslå, hvor pengene tjenes. Der eksisterer dog flere muligheder for at implementere fordelingen af omkostninger, og disse har hver deres fordele og ulemper.

En traditionel måde at fordele omkostningerne på har været kendt som Full-Cost regnsk-

abet. Full-Cost princippet fungerer som en totrins-model, hvor virksomhedens ressourcer i form af kapacitetsomkostninger fordeles til produkter. Et skema over Full-Cost omkostningsfordelingen kan ses i Figur 20.



Figur 20: Grundstruktur i Full-Cost, frit efter [KC98]

I første lag bliver omkostningerne lagt ud i en række overheadomkostningscentre. I andet lag bliver der defineret en række produktionsomkostningscentre, hvorpå omkostninger fra de overliggende centre fordeles. Denne fordeling sker ud fra hvilke produktionsomkostningscentre, som vedrører overheadomkostningscentrene. En sådan fordeling kan være meget kompleks, men kan også bygge på simple iagttagelser af virksomhedens processer. For at fordele omkostningerne videre fra produktionsomkostningscentrene til de enkelte produkter bliver forholdet mellem de budgetterede akkumulerede omkostninger og den budgetterede aktivitet beregnet. Dette bliver udtrykt i enten maskintimer eller løntimer. Herefter bliver de indirekte omkostninger fordelt på produkterne ved at multiplicere tillægssatsen med mængden af fordelingsnøglen, som er forbrugt af hvert produkt[19]. Hermed bliver det muligt for en virksomhed at få identificeret en struktur for kapacitetsomkostninger og fordele disse ud på produkter.

Full-Cost fordelingen har dog også en række ulemper, hvilket medvirker til unøjagtigheder i fordelingen. Der kan argumenteres for, at der i alle fordelinger vil eksistere svagheder, og en fordeling aldrig vil kunne afspejle virkeligheden 100 %.

- Risiko for omkostninger fordeles til produkter, som de ikke vedrører.
- Produktspecifikke omkostninger kan i nogle tilfælde også fordeles.
- Alle fællesomkostninger fordeles.
- Fordeling af omkostninger bliver upræcis, idet fordelingen ikke er baseret på træk på aktiviteter.

Ved at lægge alle omkostninger ned i omkostningscentre vil virksomhedens produkter komme til at trække på en hel afdeling, selvom der kun bliver benyttet et minimalt træk på en del af afdelingen. Dette vil kunne give nogle relativt store fejlallokeringer

[19] [BI04], side 29

af omkostningerne. Denne sammenhæng fremgår også af punkt to, hvor omkostninger til for eksempel en maskine, som kun vedrører et enkelt produkt, bliver lagt ned i et omkostningscenter, som også arbejder med andre produkter. Omkostningssystemet lægger ligeledes op til, at alle kapacitetsomkostningerne bliver fordelt, hvor det til tider vil være relevant ikke at fordele omkostninger, som for eksempel er virksomhedsbevarende. Ved at omkostningerne bliver baseret på de enkelte omkostningscentre ud fra, hvor mange timer de benytter, og ikke, hvad der reelt foregår i processen, vil der kunne opstå skæve fordelinger. Her tænkes på, at et træks tidsmæssige forbrug ikke nødvendigvis illustrerer den reelle omkostning.

En anden måde at fordele omkostninger er ved hjælp af den amerikanske Activity Based Costing. Her er grundtanken, at der ud fra nogle specificerede ressourcepuljer bliver fordelt omkostninger ud på de aktiviteter, som virksomheden udfører i driften. Virksomhedens omkostningsobjekter trækker herved på disse aktiviteter efter et givet mønster. Argumentationen for at benytte Activity Based Costing frem for Full-Cost er, at dette giver en mere præcis fordeling af omkostningerne, idet aktivitetsbegrebet er langt mere præcist end produktionsomkostningscentrene sammenholdt med en bedre fordelingsmåde ned til omkostningsobjekterne. Dette gør også, at Activity Based Costing systemet vil blive mere kompliceret i brug og derved mere omkostningstungt for virksomheden at benytte frem for en Full-Cost fordeling[20].

Eftersom virksomheder har behov for at fordele omkostninger, skal der eksistere en vis kompleksitet i omkostningsstrukturen. Eksempelvis vil virksomheder, som kun producerer et produkt, ikke have behov for fordelingen. Ligeledes er det vigtigt at tage hensyn til virksomhedens omkostningsstruktur, idet full-cost fordelingen kan være hensigtsmæssig frem for ABC fordelingen. Her er det vigtigt, at virksomheden analyserer sin omkostningsstruktur igennem og vælger den bedst egnede løsning.

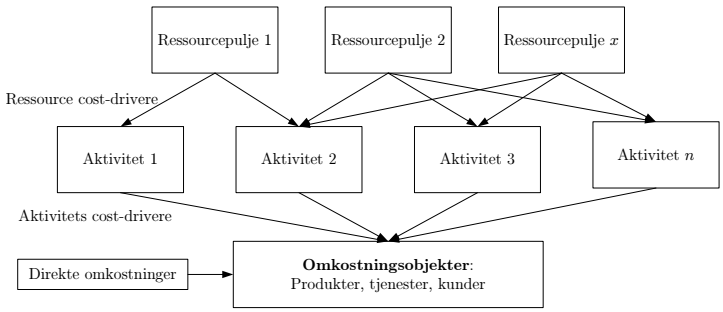
I de seneste år har tendensen været, at Activity Based Costing tankegangen har været den foretrukne. Dette illustreres også ved, at store danske virksomheder også har taget Activity Based Costing til sig, hvilket bliver beskrevet i [BI03b]. Derfor er der i denne rapport valgt Activity Based Costing i systemet. En yderligere beskrivelse af Activity Based Costing kan findes i Afsnit 7.1 på modstående side.

[20] [Buk06], side 335–336

7 Fokusområde

7.1 Activity Based Costing

Activity Based Costing (ABC) tager udgangspunkt i, at hovedparten af alle aktiviteter, som virksomheder udfører, har til formål at skabe overskud gennem produktion og salg af produkter. Omkostninger skal derved fordeles på enheder, produkter, produktgrupper, kunder og kundegrupper således, de svarer til disse gruppers træk på aktiviteten[21]. En skitse af grundstrukturen i ABC kan ses i Figur 21. Det er vigtigt at pointere, at dette blot er grundstrukturen. I praksis vil ABC blive tilpasset hver enkelt virksomhed ud fra de designkriterier, som bliver stillet op ud fra dette diagram. Det er derfor ikke muligt at angive noget standardsystem i ABC.



Figur 21: Grundstruktur i ABC systemet, frit efter [KC98]

ABC systemet kan som de traditionelle fordelingsregnskaber ses som en totrins-procedure. Forskellen ligger i fordelingen af omkostningerne, hvor ABC systemet fokuserer på aktiviteter. Aktiviteterne bliver efterfølgende fordelt ud på virksomhedens forskellige omkostningsobjekter. Den store udfordring er her at få omkostningerne fordelt korrekt ud således, at ABC systemet giver et reelt billede af situationen. For at fordele disse benyttes fordelingsnøgler, der betegnes cost-drivere. Cost-drivere er fordelingsnøgler, der

[21] [BI03a], side 6

giver et billede af den årsagsbestemte sammenhæng mellem forbruget af ressourcer, aktiviteter og omkostningsobjekter[22]. Der er her tale om henholdsvis ressource cost-drivere og aktivitets cost-drivere. Ressource cost-drivere benyttes i ABC systemets første trin, hvor ressourcer/omkostninger fordeles ud på de aktiviteter, i hvilke de indgår. I andet trin benyttes aktivitets cost-drivere til at fordele aktivitetsomkostningerne ud på omkostningsobjekter. Her bliver fordelingen lagt på baggrund af omkostningsobjektets træk på aktiviteterne, og derved kommer fordelingsnøglerne til at afspejle omkostningsobjektets reelle ressourcetræk. Der findes tre forskellige typer af aktivitets cost-drivere til fordeling:

- Transaktionsdrivere.
- Varighedsdrivere.
- Direkte måling (intensitetsdriveren).

Transaktionsdriveren anvendes til fordeling, hvis det er antallet af gange et omkostningsobjekt, der trækker på aktiviteten, som måles. Denne bruges, når alle omkostningsobjekter trækker på aktiviteten på samme måde. Her må der ikke være forskel på varigheden eller mængden i trækket på aktiviteten. Transaktionsdriveren er klart den letteste at måle og derved også den, som medfører færrest omkostninger for virksomheden. Varighedsdrivere tager udgangspunkt i den tid, det tager at udføre aktiviteten. Driveren benyttes i tilfælde af, at omkostningsobjekternes måde at trække på aktiviteterne på er meget varierende og dermed ikke med fordel kan måles med transaktionsdriveren. Eksempler på dette kan være store forskelle i omstillingstid på en maskine eller varierende længde af kundebesøg. Varighedsdriveren kræver større og mere præcis viden om, hvordan virksomhedens arbejdsrutiner fungerer, og den er derfor mere omkostningstung at benytte for virksomheden. Direkte måling er nødvendig, hvis ingen af de to ovenstående drivere kan anvendes. Altså, hvis det ikke er muligt at fordele aktivitetsomkostningerne i form af antallet af gange, de er udført, eller hvis det ikke tidsmæssigt kan opdeles. Denne benyttes i tilfælde af, at det er nødvendigt at henføre omkostningerne direkte til omkostningsobjekterne ved anvendelse af arbejdssedler, edb-registreringer med videre[23].

De direkte omkostninger, som for eksempel er ved produkterne, bliver i ABC henført nøjagtigt, som de bliver i de traditionelle fordelingsregnskaber. Det vil sige på de produkter, som de vedrører[24].

Dette er grundstrukturen i ABC, som virksomhederne implementerer ud fra. Som før nævnt, er det kun en model, og i praksis vil den typisk se anderledes ud, når den bliver tilpasset den enkelte virksomhed. Ligeledes vil mange virksomheder vælge kun at indføre dele af ABC systemet samt kun benytte sig af det indenfor enkelte sektioner eller afdelinger i virksomheden. Dette kan dog kun lykkes, når afdelingen er autonom. ABC

[22][BI03a], side 9
[23][BI03a], side 22
[24][BI04], side 27

er i kraft af sine komplicerede fordelingsdrivere ressourcekrævende og derved omkostningstung at implementere i virksomhederne. Dette betyder, at indførelsen kræver grundige overvejelser og kalkulationer, før der omlægges til ABC. Dette gøres ved en simpel offeromkostningsbetragtning for at beregne nytten ved indførslen af ABC. Ligeledes vil en ændring i virksomhedens dagligdag, såsom nye maskiner, arbejdsmetoder og nye typer kunder, medføre behovet for at gendesigne ABC systemet i virksomheden. Sker dette ikke, vil systemet ikke give det nøjagtige billede af omkostningsfordelingen, som det var tiltænkt, og virksomhedens økonomistyring vil blive unøjagtig. Dette er også en del af den kritik, som modellen får; at den er for kompliceret at bruge i praksis[25].

Det er vigtigt at pointere, at ABC systemet ikke er en direkte beslutningsmodel i forhold til at træffe beslutninger om, hvorvidt det er lønsomt at blive ved med at producere et bestemt produkt. Grunden til dette er, at alle kapacitetsomkostningerne er fordelt ud på forskellige produkter, og hvis et af disse tages ud af produktion, vil det kun være de direkte omkostninger, som forsvinder. De resterende kapacitetsomkostninger vil således skulle deles ud på resten af virksomhedens produktsortiment.

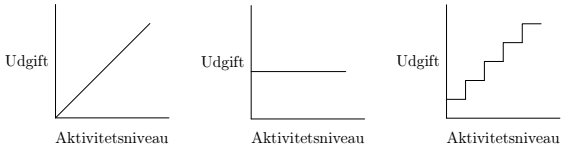
7.2 Ledig kapacitet i ABC

I ABC-systemet eksisterer der ledig kapacitet i forbindelse med ressourcerne. Ledig kapacitet opstår, når en given ressource ikke bliver benyttet fuldt ud, hvilket udspringer af ressourcernes variabilitet. Problemstillingen kan illustreres ved, at hvis der er beregnet at en lastbil kan have et helt læs, men en aktivitet kun udnytter 80 % af dette læs, så kan de sidste 20 % her betragtes som den ledige kapacitet på ressourcen. I mange tilfælde kan der optimeres i virksomheden, således at den ledige kapacitet mindskes, men den vil altid eksistere. Et redskab til at afhjælpe problemet med ledig kapacitet er ved at lave strukturen på virksomhedens ressourcer så variable som muligt.

En anden form for ledig kapacitet opstår, hvis virksomheden ikke udnytter sine ressourcepuljer fuldt ud. Herved bliver de aktiviteter, som trækker på de givne ressourcepuljer, uhensigtsmæssigt store og afspejler ikke omkostningen korrekt. Grunden til dette er, at der eksisterer forskellige typer af ressourcer, som ikke optræder efter samme mønster. [KC98], beskriver at ressourcerne har forskellige udgiftsmønstre. Der opereres i ABC med tre typer af ressourceomkostninger, fleksibel, fast og trinvis. Det er illustreret i nedenstående Figur 22 på den følgende side, hvordan disse fungerer.

Her illustreres, hvordan ressourcerne fungerer i virksomheden. I den fleksible stiger udgiften i takt med, at aktivitetsniveauet stiger. Dette vil være den foretrukne for virksomheden, idet omkostningen til ressourcen altid vil hænge tæt sammen med kapaciteten af denne. Den faste ressourceomkostning er der altid, uanset hvor meget behov der er for denne. Dette er typisk faste ting som bygninger med videre. Virksomheden har i denne forbindelse ikke nogen direkte mulighed for at påvirke størrelsen af denne omkostning. Den trinvise omkostning stiger i takt med forskellige niveauer nås. Dette kan forklares

[25] [Buk06], side 335–336



(a) Fleksibel (b) Fast (c) Trinvis

Figur 22: Typer af ressourceomkostninger

med, at der er en vis kapacitet, men hvis aktivitetsniveauet overstiger en grænse, tvinges virksomheden til at investere i ny ressourcekapacitet, som virksomheden typisk ikke er i stand til at udnytte fuldt ud. Virksomheder vil typisk forsøge at begrænse denne overkapacitet ved at lægge budgetter for periodens produktion.

Den ledige kapacitet vil typisk blive håndteret i ABC modellen ved at lave fordeling fra ressourcerne ned til en aktivitet, som repræsenterer den ledige kapacitet. Hvor meget, den ledige kapacitet skal repræsentere, vurderes med måling når ABC systemet designes. Virksomhederne har typisk statistikker til rådighed, som kan være til hjælp i denne proces.

7.3 Praktisk implementering af ABC

I dette afsnit identificeres aktiviteter til anvendelse i et ABC system. Nogle af disse aktiviteter benyttes efterfølgende i et eksempel.

7.3.1 Identificering af aktiviteter i ABC

For at kunne udarbejde en fungerende applikation i forbindelse med ABC er det nødvendigt at få fastlagt aktiviteter i virksomheden. Dette kræver typisk en stor gennemgående analyse af virksomheden, medarbejderne og de arbejdsgange, som foreligger. Præcision i ABC systemet nås ud fra specifikation af mange aktiviteter[26]. Herved lægges der op til, at det gælder for virksomheden om at finde så mange aktiviteter som muligt og derved også dele de forskellige aktiviteter op i mindre aktiviteter. Dette har dog designmæssigt den ulempe, at antallet af aktiviteter vokser markant, og det derved kan være svært at overskue, vedligeholde og benytte ABC systemet. Ligeledes argumenterer [Buk06] for, at selvom virksomheden forsøger at gøre aktiviteterne så heterogene som muligt, kan dette ikke lykkes fuldt ud. Derved er det et individuelt designspørgsmål i, hvor stor grad aktiviteterne skal detaljeres. [KC98] beskriver også sammenhængen af kompleksitet og præcision af systemet som et tradeoff. Her nævnes det, at der skal søges at nå et mål, som giver et retvisende resultat indenfor 5–10 %. Her skal målet være at have det bedste omkostningssystem, som balancerer fejlomkostninger grundet unøjagtige målinger med

[26][Buk06], side 347

omkostningerne af målingerne[27].

Som et redskab til at identificere aktiviteter i virksomheden kan de allerede konstruerede procesdiagrammer benyttes fra Afsnit 4.2 på side 23. Der er i denne rapport udviklet enkelte procesdiagrammer til eksemplificering. Men typisk vil virksomheden have behov for at udvikle disse til alle arbejdsgangene. Til at illustrere dette vil der her blive taget udgangspunkt i Figur 6 på side 24. Ud fra de enkelte processer i procesdiagrammet kan aktiviteter typisk udledes. Der kan være tilfælde, hvor der ikke kan argumenteres for, at en proces direkte kan henledes til en aktivitet, hvilket kræver en behandling af hver enkelt proces for sig. Ud fra procesdiagrammet kan aktiviteterne illustreret i Figur 23 identificeres.

Rekvisition af råvarer	Omstilling af maskiner
Produktion af ryglæn	Produktion af sæde
Produktion af ben	Montage
Kvalitetskontrol	Undersøge om fejlene kan udbedres
Udbedre fejl	

Figur 23: Aktiviteter i produktion

Det er vigtigt, at procesdiagrammet er udarbejdet korrekt, da alle aktiviteterne ellers ikke kan identificeres. Processer som **Registrering af direkte omkostninger** samt **Registrer lagerbeholdning** er ikke taget med under aktiviteter. Begrundelsen for dette er, at dette sker automatisk i EDB anlægget og derved bruger et ubetydeligt træk på ressourcerne. Dette betyder ikke, at alle registreringsprocesser i procesdiagrammet skal udelades, da de skal inkluderes, hvis der er tale om mindre automatisering som for eksempel, hvis en medarbejder sidder og taster disse ind. Decision points er ikke taget med under aktiviteter, idet disse typisk kun er med for at illustrere, at der er flere mulige veje at gå. Derved ligger der en proces foran, hvor aktiviteten er opstået.

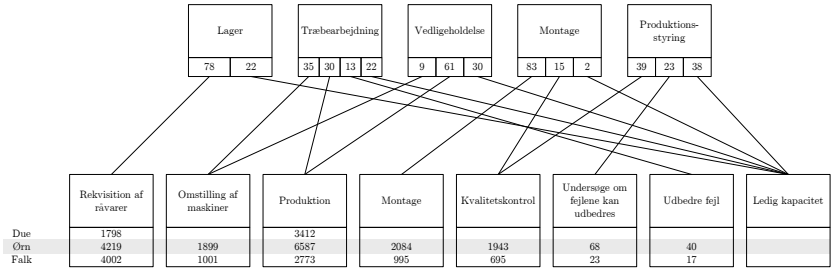
Efter at aktiviteterne er blevet identificeret, skal de herefter kædes sammen med ressourcerne ved hjælp af cost-driverne. Det er her, det meste arbejde for virksomheden ligger, idet der kræves, at der udarbejdes målinger og estimater på, hvordan aktiviteterne trækker på virksomhedens ressourcer.

7.3.2 Eksempel på struktur

I dette afsnit vil der blive vist et eksempel på en ABC struktur, som vil blive brugt i implementationen i Application Express. Strukturen er vist i Figur 24 på den følgende side. For at simplificere eksemplet, er der valgt ikke at medtage alle aktiviteter fra Afsnit 7.3.1 på forrige side.

For at lette forståelsen af ABC strukturen i eksemplet er der valgt at benytte en lidt anden tegnemåde end den traditionelle. Det er ikke en ændring af selve ABC modellen,

[27] [KC98], side 102



Figur 24: Eksempel på ABC struktur

men mere en måde til at komme væk fra de uoverskuelige diagrammer, med en masse streger, der går over hinanden.

I eksemplet er valgt en kombination af stregsystemet og et kolonnesystem. Fra ressourcer til aktiviteter er stregen til den enkelte aktivitet relateret til en kasse, der viser den procentsats, den trækker på ressourcen med. På denne måde vil der hurtigt kunne dannes et overblik over, hvordan ressourcefordelingerne er. Fra aktivitet til produkt viser et kolonnesystem, hvor meget de enkelte produkter trækker på aktiviteterne.

Tallene kunne være udarbejdet på baggrund af en periode på en måned. Trækket, der er tilskrevet i produktkolonnerne, er det totale træk fra hele periodens produktion. For at udregne den ledige kapacitet tages en given periodes produktion og det træk, produktionen har på samtlige aktiviteter. Hvis dette tal bliver mindre end den totale ressources kapacitet, vil der være ledig kapacitet. Der vil også kunne forekomme, at en periodes forbrug er større end ressourcens kapacitet. Dette kan ske, hvis kapaciteten er fastsat forkert, eller hvis tiden, som er afsat til for eksempel reparation og vedligeholdelse, er blevet brugt til produktion i denne periode[28].

[28] [BI04], side 58

8 Udvikling af applikation

I dette kapitel vil diagrammerne, der skal bruges til udvidelsen med ABC, blive præsenteret. Derudover vil tabeldesignet samt SQL blive vist og forklaret. Endelig vil skærmbillederne for tilføjelsen til brugergrænsefladen blive vist.

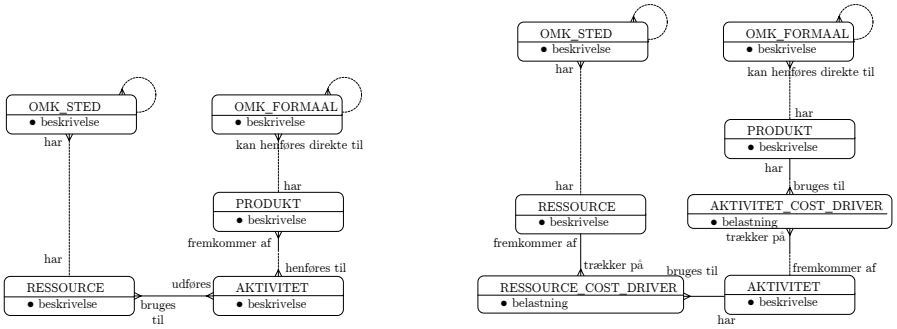
8.1 ER-diagrammer

I forbindelse med dette kapitel vil det allerede udviklede design fra Afsnit 4 på side 23 blive udvidet med ABC. Derfor vil der blive udviklet et ER-diagram for denne udvidelse, som senere vil blive omdannet til et tabeldesign. Figur 25 på næste side viser denne udvidelse. Entiteterne `omk_formaal`, `omk_sted` og `produkt` er henholdsvis formålsdimensionen, stedsdimensionen og bindeleddet mellem omkostninger og indtægter, som allerede findes i Figur 12 på side 28. Derudover er der tilføjet to nye entiteter; `ressource` og `aktivitet`.

`Ressource`-entiteten beskriver den valgte ressource. Her ses det, at ressourcepuljerne bliver defineret ud fra stedsdimensionen i registreringen. Dette strider imod litteraturen, hvor der generelt bliver beskrevet, at ressourcepuljerne skal tage udgangspunkt i finanskontoplanen som i [KC98, BI04]. Den valgte løsning vil kunne bruges i den nuværende form. Det er dog muligt at øge detaljeringsgraden af stedsdimensionen. Argumentet for dette er, at stedsdimensionen kan specificeres yderligere og kan nærme sig kompleksiteten af en kontoplan, hvis dette findes nødvendigt. Derudover ses det, at der er en relation fra `omk_formaal` til `produkt`. Dette grunder i, at en omkostning kan være direkte i forhold til et produkt og derfor ikke skal fordeles. Derimod skal indirekte omkostninger fordeles, hvorfor der skal angives cost-drivere for disse. En omkostning kan derimod *ikke* være begge dele, og derfor skal kun en af disse referencer udfyldes.

`Aktivitet`-entiteten beskriver de forskellige aktiviteter i ABC. Her ses det, at der er en mange-til-mange relation mellem `ressource` og `aktivitet` entiteterne. Det gælder også for entiteterne `aktivitet` og `produkt`. Udover de viste attributter skal det være muligt at kunne angive cost-driverne. Men da disse skal angives for kombinationen af de to entiteter, skal cost-driveren angives på relationen mellem de to entiteter, og derfor skal der oprettes nye entiteter, der indeholder cost-driveren som attribut. Denne udvidelse kan ses i Figur 25(b).

Stedsdimensionen vælges til at definere ressourcerne og de direkte omkostninger vælges ud fra formålsdimensionen. Grunden til, at de direkte omkostninger vælges ud fra formålsdimensionen er, at de er nemmere at identificere herudfra. Det samme gælder for stedsdimensionen, som minder en hel del om ressourcer i ABC, hvorfor denne er valgt til at definere ressourcerne ud fra. Stedsdimensionens enkelte konti er ikke homogene med hensyn til reversibilitet, og derfor vil de enkelte ressourcer heller ikke være homogene. Derimod vil artsdimensionen være mere hensigtsmæssig at beregne reversibilitet ud fra. Hvis de enkelte artskonti har specificeret reversibilitet vil denne reversibilitet kunne *fordeles* på samme måde som omkostninger igennem ABC systemet.



(a) Konceptuelt ER-diagram for ABC udvidelsen

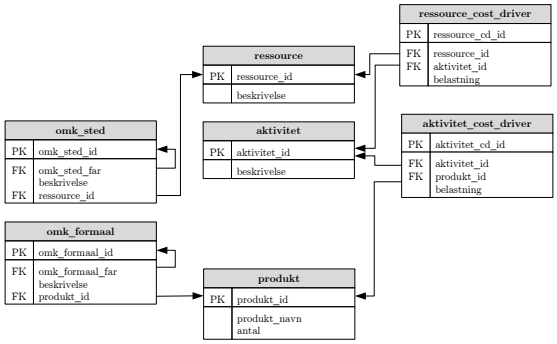
(b) ER-diagram med driver entiteter for ABC udvidelsen

Figur 25: ER-diagrammer over ABC

8.2 Tabeldesign

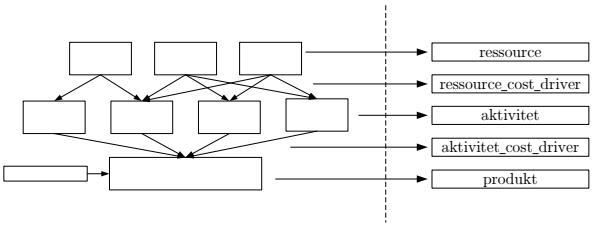
Efter, at der er blevet udarbejdet ER-diagrammer for ABC udvidelsen, kan det egentlige tabeldesign udarbejdes. Der bliver i den forbindelse bygget videre på omkostningsregistreringssystemet fra Afsnit 4.7 på side 31, hvor de ekstra ABC-tabeller bliver koblet på. ABC registreringen kommer til at bestå af fire ekstra tabeller; `aktivitet`, `ressource_cost_driver`, `aktivitet_cost_driver` samt `ressource`. Disse tabeller bliver koblet sammen til omkostningsregistreringen ved hjælp af tabellerne; `omk_formaal`, `omk_sted` samt `produkt`. Sammenhængen mellem tabellerne kan ses i Figur 26 på næste side.

Ud fra Figur 26 på modstående side ses de to tabeller for cost-drivere. Formålet med at have disse tabeller er at håndtere mange-til-mange relationen mellem henholdsvis



Figur 26: Tabeldesign ABC delsystemet

`omk_sted` og `aktivivitet` samt `produkt` og `aktivitet`. Herved bliver det ved hjælp af cost-driverne muligt at registrere sammenhængen mellem ressourcer, aktiviteter samt omkostningsobjekter i ABC systemet. I Figur 27 visualiseres, hvordan den grundlæggende ABC model er forbundet til tabellerne i databasen. Her ses det, at de tre grupper af objekter i ABC systemet bliver implementeret ved hjælp af en tabel hver. Mellem disse objekter er der en række relationer, som viser sammenhængen mellem ressourcer, aktiviteter samt omkostninger. Denne sammenhæng kan i databaseteorien betragtes som en mange-til-mange relation, som er beskrevet lige ovenfor og vil blive brugt til sammenkædningen af ressourcer, aktiviteter og omkostningsobjekter. Tabellen `aktiviteter` vil blive udfyldt med aktiviteter, som er fremkommet ifølge analyse i virksomheden, som beskrevet i Afsnit 7.3.1 på side 54. De to typer af cost-drivere vil i systemet blive fastlagt for at illustrere funktionaliteten. Det vil ligeledes kræve en gennemgribende analyse af virksomheden for at få alle disse fastlagt. Dette vil også kunne løses ved hjælp af beskrivelsen i Afsnit 7.3.1 på side 54. Herved vil tabellerne umiddelbart ikke være et sted, hvor almindelige brugere af systemet vil opnå adgang til at redigere. Dette vil typisk foregå ved implementeringen af systemet, og ved yderligere vedligeholdelse og opdateringer af ABC systemet.

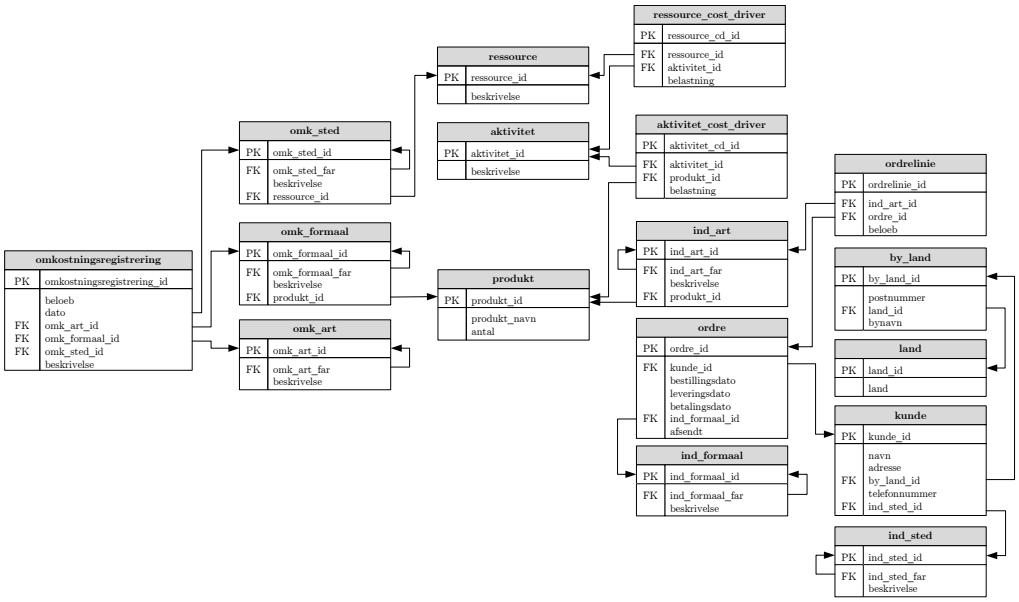


Figur 27: Sammenhæng mellem ABC og tabeller

SQL til udarbejdelse af tabeller er valgt ikke at medtage i dette afsnit, idet den ikke indeholder nye aspekter i forhold til oprettelse af tabeller, end hvad der er beskrevet i Afsnit 4.8 på side 33.

Herved kan det fulde tabeldesign af registreringssystemet ses i Figur 28 på næste side. Læg mærke til, at stort set hver enkelt entitet fra Figur 12 på side 28 er blevet afbilledet over til en tabel. `Færdigvarelager`-entiteten blevet modelleret som attributten **antal** i

`produkt`-tabellen, da denne kun indeholder en attribut, som lige så godt kunne være en egenskab ved produktet.



Figur 28: Tabeldesign ABC implementeret i variabilitetsprincippet

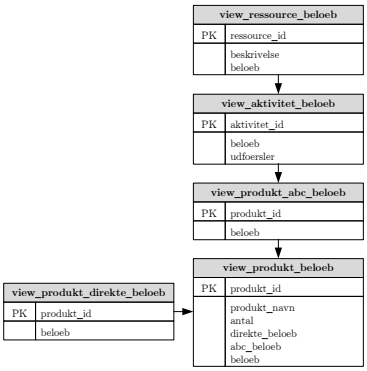
8.3 SQL

I det følgende afsnit vil SQL til implementering af ABC blive beskrevet. Da der i Afsnit 4.8 på side 33 allerede er blevet beskrevet de grundlæggende funktioner i SQL, vil dette afsnit omhandle de nye funktioner, der bliver implementeret. Tabeller og relationer til ABC systemet er oprettet i databasen ud fra tabeldesign i Figur 28. For at lette implementationen af ABC beregningerne er der blevet implementeret en række *views* i databasen.

Et view i en database kan betragtes som et specifikt udtræk af data fra en eller flere tabeller i databasen, hvor kun de data, som er relevante i en given situation, er til rådighed. Dette kan i visse tilfælde medvirke til at øge sikkerheden i databasen, da der kan være data i tabeller, som enkelte brugere ikke skal have adgang til. Ved at benytte views er det muligt kun at vælge de data, som brugeren skal kunne se, selvom den logiske databasemodel indeholder en lang række andre data[29]. Dog er det ikke grundet sikkerhed, at der vil blive benyttet views i implementeringen. Ved at oprette views mellem tabellerne `ressource`, `aktivitet` samt `produkt` vil det blive betydeligt lettere at tilgå de beregninger, som skal laves i cost-driver tabellerne. Disse views kan eventuelt materialiseres for at forbedre hastigheden[30]. Her bliver beregningerne lavet en gang, og det er derefter muligt at trække alle relevante data fra disse views uden at skulle

[29] [SKS01], side 113
[30] [SKS01], side 126

forespørge på flere tabeller og udføre joins imellem disse. Figur 29 illustrerer, hvordan sammenhængen mellem de enkelte views i ABC systemet fungerer.



Figur 29: Views i ABC

Et eksempel på views ses i nedenstående SQL-sætninger 7 og 8. Disse to eksempler er taget med for at illustrere et simpelt view samt et view, som bygger på et foregående view og derved bliver mere kompliceret.

```
1  CREATE VIEW VIEW_RESSOURCE_BELOEB AS
2    SELECT RESSOURCE.RESSOURCE_ID,RESSOURCE.BESKRIVELSE, SUM(beloeb) AS beloeb
3      FROM RESSOURCE, OMK_STED, OMK_FORMAAL, OMKOSTNINGSREGISTRERING
4      WHERE RESSOURCE.RESSOURCE_ID = OMK_STED.RESSOURCE_ID AND
5        OMK_STED.OMK_STED_ID = OMKOSTNINGSREGISTRERING.OMK_STED_ID AND
6        OMK_FORMAAL.OMK_FORMAAL_ID = OMKOSTNINGSREGISTRERING.OMK_FORMAAL_ID AND
7        OMK_FORMAAL.PRODUKT_ID IS NULL
8    GROUP BY RESSOURCE.RESSOURCE_ID, RESSOURCE.BESKRIVELSE;
```

SQL-sætning 7: SQL til `view_ressource_beloeb`

SQL-sætning 7 viser, hvordan `view_ressource_beloeb` bliver lavet. Viewet viser hver ressource og dennes samlede konterede beløb. På Linie 2–3 udvælges de forskellige kolonner, der skal vises i viewet. Her bliver alle omkostningsregistreringerne ført hen på deres respektive ressourcer, og det samlede beløb for hver ressource bliver udregnet. Linie 4–7 joiner de tre tabeller. Til sidste grupperer Linie 8 resultatet på **ressource_id** og **beskrivelse**.

```
1  CREATE VIEW VIEW_PRODUKT_ABC_BELOEB AS
2    SELECT PRODUKT.PRODUKT_ID,
3    SUM(AKTIVITET_COST_DRIVER.BELASTNING *
4     (VIEW_AKTIVITET_BELOEB.BELOEB / VIEW_AKTIVITET_BELOEB.UDFOERSLER)) AS beloeb
5    FROM PRODUKT, AKTIVITET_COST_DRIVER, VIEW_AKTIVITET_BELOEB
6    WHERE PRODUKT.PRODUKT_ID = AKTIVITET_COST_DRIVER.PRODUKT_ID AND
7      AKTIVITET_COST_DRIVER.AKTIVITET_ID = VIEW_AKTIVITET_BELOEB.AKTIVITET_ID
8    GROUP BY PRODUKT.PRODUKT_ID;
```

SQL-sætning 8: SQL til `view_produkt_abc_beloeb`

SQL-sætning 8 viser, hvordan `view_produkt_abc_beloeb` bliver lavet. Viewet viser hvert produkt fra `produkt` tabellen med deres fordelte omkostninger. Linie 3–5 udvælger

produkt_id og aktiviteterne fordelt ned på deres respektive produkter. Kolonnen **udfoersler** på `view_aktivitet_beloeb` angiver, hvor mange gange den enkelte aktivitet udføres totalt på produkterne. Linie 6–7 joiner de to tabeller, og på Linie 8 grupperes resultatet. Alle SQL sætninger, der er blevet brugt i dette hovedafsnit, kan ses i Bilag B på side 96.

8.4 Brugergrænseflade

I dette afsnit vil tilføjelser til brugergrænsefladen samt funktionaliteten deri blive beskrevet. Der vil blive bygget op omkring brugergrænsefladen, som blev præsenteret i Afsnit 4.8.3 på side 36. Umiddelbart håndterer denne brugergrænseflade registreringer i systemet, rapporter til at løse overskudsopgaven, samt prisfastsættelses- og kalkulationsopgaven. Umiddelbart skal der ikke laves ændringer i forbindelse med registreringen af omkostninger og overskudsopgaven. ABC rapporter skal implementeres i forbindelse med løsningen af prisfastsættelses- og kalkulationsopgaven, idet der ved fordeling af omkostningerne opstår behov for udvidede rapporter.

Omkostninger per produkt

Produkt Navn	Direkte omkostninger	ABC-fordelte omkostninger	Samlede omkostninger
Stol Ørn	239.208,00	753.885,38	993.093,38
Stol Due	257.676,00	82.894,84	340.570,84
Bord Falk	240.638,00	383.494,78	624.132,78
			1 - 3

Figur 30: Skærmbillede af omkostninger per produkt

Figur 30 viser ABC rapporteringen. Her ses en liste over samtlige produkter med deres respektive direkte og fordelte omkostninger. Derudover ses de samlede omkostninger.

Omkostninger per aktivitet

Beskrivelse	Beløb
Rekvisition af råvarer	223.072,98
Omstilling af maskiner	94.799,95
Produktion	160.445,00
Montage	514.584,23
Kvalitetskontrol	158.499,60
Undersøge om fejlene kan udbedres	38.629,65
Udbedre fejl	30.243,59
Ledig kapacitet	234.905,00
	1 - 8

(a) Omkostninger per aktivitet

Omkostninger per ressource

Beskrivelse	Beløb
Produktionsstyring	167.955,00
Træbearbejdning	232.643,00
Lager	285.991,00
Vedligeholdelse	148.610,00
Montage	619.981,00
	1 - 5

(b) Omkostninger per ressource

Figur 31: Skærmbilleder af ABC rapporter

Figur 31 viser nogle yderligere rapporter, som kan være brugbare i forbindelse med ABC rapportering. Figur 31(a) viser alle aktiviteterne med deres omkostninger fordelt fra de enkelte ressourcer. Figur 31(b) viser de enkelte ressourcer med deres omkostninger summeret.

8 Opsummering

I Hovedafsnit I har fokus været implementering af variabilitetsprincippet med indtægter. I dette hovedafsnit har fokus været at videreudbygge rapporteringsdelen med ABC samtidig med at kunne benytte forskellige analyseværktøjer.

I forbindelse med implementering af ABC viste en analyse, at specielt prisfastsættelsesopgaven i større grad kan løses bedre end med blot variabilitetsprincippet som grundlag for rapportering. ABC viste sig at være mere anvendelig end den nært slægtede Full-Cost fordeling.

På baggrund af procesdiagrammerne, der blev udarbejdet i Afnit 4.2 på side 23, blev disse også brugt til at identificere de forskellige aktiviteter, der kan være i ABC modellen. Sideløbende er der blevet udarbejdet et ER-diagram på baggrund af databasedesignet fra Afsnit 4.7 på side 31 udbygget med ER-diagrammering af ABC entiteter. Her blev formålsdimensionen brugt til at identificere de direkte omkostninger, og stedsdimensionen blev brugt til at definere ressourcerne.

I forbindelse med implementeringen blev der benyttet et antal views for at dele udregninger ud i et mindre antal beregninger for at lette udregningerne. Til slut er hele ABC delen blevet implementeret i Application Express således, at der kan udtrækkes rapporter på såvel produktbasis som på aktivitets- og ressourcebasis.

III

ERP

10 Indledning

I de to tidligere hovedafsnit har fokus har været på softwareudvikling ved hjælp af Oracle Designer og Application Express. Dette hovedafsnit adskiller sig fra de to andre på den måde, at der fokuseres på anvendelse af et eksisterende *Enterprise Resource Planning* (ERP) system. Der findes forskellige ERP systemer så som Microsoft Dynamics AX, Microsoft C5, SAP R/3 og SAP Business One. I denne rapport vil der blive anvendt SAP Business One, som udvikles og forhandles af SAP, der er en af de største leverandører af ERP systemer ifølge [Haz06, Koc01, Rel06].

Hovedafsnittet behandler problemstillingen vedrørende, hvorledes SAP Business One kan anvendes i forbindelse med ABC. Der vil være en teoretisk gennemgang vedrørende ABCs generelle anvendelse i forbindelse med ERP systemer. Derudover vil der være en gennemgang af, hvorledes SAP Business One praktisk vil kunne benyttes i forbindelse med ABC rapportering. Herunder vil der blandt andet være en gennemgang af, hvor de forskellige elementer i ABC skal findes i SAP Business One samt hvilke data, det ikke kan levere og som derfor skal findes eksternt.

11 Fokusområde

I dette kapitel vil ERP systemer blive gennemgået generelt, herunder hvilke egenskaber, der er tilgængelige i et sådant system. Derudover vil SAP Business One blive introduceret samt en gennemgang af den mest vigtige funktionalitet. Til sidst vil der være en analyse af, hvordan ERP systemer understøtter ABC.

11.1 ERP generelt

Et ERP system samler alle virksomhedens afdelinger og funktioner i et enkelt IT system. Dette har til formål at opfylde alle de forskellige afdelingers individuelle krav. Ofte har de forskellige afdelinger, så som for eksempel salgsafdeling og produktionsafdeling, hvert sit IT system, der er optimeret til lige nøjagtig de funktioner, der er behov for i den afdeling. Disse systemer er ofte ikke designede til at kunne dele informationer. ERP derimod kombinerer alle disse delsystemer til et system, der afvikles på baggrund af en fælles database, hvilket gør, at de forskellige afdelinger nemt og hurtigt kan dele informationer. Eksempler på disse delsystemer (moduler) kan være kundehåndtering, finans, lagerstyring og produktionsstyring.

Fordelen ved at have alle virksomhedens funktioner samlet i et stort system er, at administrative arbejdsgange i virksomheden mindskes. For eksempel kan der i en virksomhed, som ikke benytter sig af ERP, når en ordre bliver afgivet, blive indtastet data i et salgssystem i salgsafdelingen. Denne ordre vil derefter blive printet ud og blive afleveret til lageret eller eventuelt produktionsafdelingen. I disse afdelinger vil ordren blive indtastet i afdelingernes IT systemer. I et sådant forløb er der risiko for, at ordren bliver væk undervejs, hvilket kan medføre utilfredse kunder eller annullerede ordrer. Derudover kan en sådan papirgang være med at forlænge ordrens behandlingstid betydeligt. Endeligt er der risiko for, når data skal indtastes i flere systemer, at disse bliver fejlindtastet i nogle af afdelingerne, og dermed vil den endelige ordre, når den når frem til kunden, være forkert. Det samme problem opstår, når for eksempel en kunde ønsker at ændre sin ordre. Så skal disse opdateringer igennem hele papirgangen igen. Med et ERP system vil

informationerne automatisk blive sendt videre til den næste afdeling, når de er færdigbehandlede. Opdateringer kan ske i hvilken som helst afdeling, og disse opdateringer vil derefter være tilgængelige i alle andre afdelinger.

Et af formålene med et ERP system er ligeledes at skabe et grundlag, hvorfra beslutninger i virksomheden kan træffes. Hermed betragtes ERP systemet som et redskab, som kan benyttes af virksomhedens beslutningstagere til at styre den adfærd, der har relevans i forhold til virksomhedens økonomi. I en virksomhed, der ikke benytter sig af et ERP system, skal beslutningstagerne, der ønsker et overblik over virksomhedens generelle ydeevne, tage stilling til data, der kommer fra forskellige systemer. Disse kan dog være modstridende. I et ERP system er der kun en version af data, som beslutningstagerne skal tage stilling til. Derudover findes der ofte moduler integreret direkte i ERP systemet, som kan udtrække data fra de enkelte moduler, der er relevante i forhold til den aktuelle beslutning.

Ifølge [Koc06] er der fem grunde til, en virksomhed skal benytte sig af et ERP system.

Finansielle oplysninger. Som nævnt ovenfor, kan de oplysninger, som beslutningstagerne skal forholde sig til, ofte variere i forhold til, hvilken afdeling der spørges. Dette kan skyldes, at nogle oplysninger ikke er blevet tastet ind i systemet, eller der er sket fejl under indtastningen. ERP systemet arbejder kun ud fra ét datagrundlag, og derfor findes der ikke forskellige versioner af data.

Kundeordre data. ERP systemet indeholder alle data om en given ordre. På denne måde bliver arbejdet med at holde styr på en ordre nemmere, end hvis disse oplysninger var delt ud i forskellige systemer. Dette gør koordination af for eksempel produktion, lager og forsendelse lettere.

Standardisering af produktionen. I produktionen sker det ofte, at der benyttes forskellige metoder og computersystemer til at lave næsten det samme. ERP systemer indeholder metoder til at automatisere nogle af de enkelte produktionsskridt.

Bedre work-flow. ERP systemet hjælper til med, at produktionsstrømmen glider bedre og skaber bedre overblik. På denne måde kan levering af produkter til kunden planlægges bedre.

Human Resource koordinering. Med et ERP system bliver det nemmere for human resource afdelingen at skabe overblik over medarbejdernes tidsforbrug, samt få overblik over den enkelte medarbejder, således at der bedre kan tilbydes fordele og service.

Ifølge [Koc06] sker det ofte, at virksomheder, der forsøger at indføre et ERP system, må opgive. Dette skyldes hovedsageligt to grunde. Sådanne nedbrud skyldes ofte, at de medarbejdere, der skal benytte systemet, ikke er enige i, at ERP systemet er bedre end de gamle, og dermed opstår der modstand mod forandringer. Ligeledes sker det ofte, at virksomhederne tilpasser systemet så meget til virksomhedens egne behov, at det

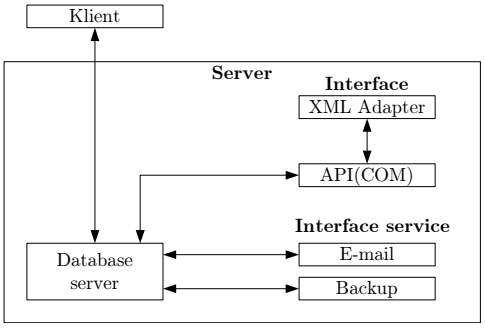
ender med, at systemet bliver ustabilt og sværere at vedligeholde. Da et ERP system dækker så mange af virksomhedens funktioner, vil det være fatalt, hvis dette bryder ned, så derfor sker det ofte ifølge [Koc06], at virksomhederne fravælger ERP, når det konstateres, at det er ustabilt. Tilpasning af ERP systemet kan i nogle tilfælde ikke undgås, idet forskellige virksomheder har forskellige krav til, hvordan et sådant system skal opføre sig, og alle disse krav kan en leverandør af ERP system selvfølgelig ikke tage højde for i en standarddistribution.

11.2 SAP Business One

I denne rapport er SAP Business One valgt til at illustrere anvendelsen af et ERP system i virksomheden. SAP Business One er udviklet af firmaet SAP, som tilbyder en række styringssystemer til virksomheder indenfor blandt andet produktionsstyring, supply chain management og ledelsesinformation. Blandt disse systemer er SAP Business One, som er udviklet som styringssystem til små og mellemstore virksomheder. SAP Business One er en løsning til disse virksomheder, hvor det også er muligt at få en mere branchespecifik løsning ved at benytte produktet "mySAP all in one". mySAP købes og udbygges i moduler, og kan således skræddersyes direkte til den enkelte virksomhed. Endelig forhandler SAP en pakke, som hedder SAP R/3, og som indeholder yderligere funktionalitet.

SAP Business One virker, som beskrevet ovenfor, ved at få virksomhedens afdelinger til at samarbejde om et fælles grundlag, den centraliserede database. Arkitekturen i systemet kan ses i Figur 32 på modstående side, hvor en installation af en enkelt server er illustreret. Installationen er baseret på en to lags klient/server arkitektur, som tillader virksomheder at integrere systemet i sit allerede eksisterende Microsoft Windows netværk. Det skal arkitekturmæssigt bemærkes, at SAP understøtter flere databaser. Dog er Oracle ikke understøttet. I Figur 32 på næste side ses det ligeledes, at serveren i SAP systemet stiller en række standardgrænseflader for databaseadgang til rådighed, hvilket giver mulighed for at benytte systemet med eksterne E-mail og backupsystemer. Ligeledes giver arkitekturen mulighed for ved hjælp af et API[a] at kunne benytte XML standarden, så systemet kan levere data til for eksempel hjemmesider samt virksomhedens eget lokale netværk. Dette kan være en fordel for virksomheder, som for eksempel vil vise sin lagerstatus på en hjemmeside. Muligheden for at kommunikere med andre systemer gør også, at systemet kommunikerer direkte med samarbejdspartnere som for eksempel leverandører, banker og kurérbureauer. Herved kan rutinemæssig kommunikation med disse foregå automatiseret. Dette kan for eksempel illustreres ved automatisk varebestilling. Medarbejderne i virksomheden kan opnå adgang til det centrale system fra de netværksmaskiner, som har SAP-klienten installeret. Det er muligt for administrationen at lave brugerrettigheder, og opsætte regler, for hvad de enkelte brugere må udføre af funktioner.

[a] Application Programming Interface



Figur 32: Arkitektur i Sap Business One, frit efter [SAP05a]

SAP Business One kan deles op i seks moduler, som bliver bundet sammen af en fælles brugergrænseflade. Disse moduler arbejder tæt sammen. Dette gør det muligt at lave dataudtræk og rapporter baseret på elementer tværs over modulerne. Som det fremgår af Figur 33, tillader denne opbygning ligeledes individuelle tilpasninger af brugergrænsefladen samt "drag and relate" teknologi, som gør det muligt at trække objekter mellem de forskellige moduler. Som en anden mulighed tilbyder SAP Business One ligeledes et software developer kit, der gør det muligt for virksomheden at udvikle egne applikationer og værktøjer, som kan hjælpe til at opfylde specielle behov i samarbejde med systemet. I nedenstående afsnit vil de seks moduler i Figur 33 blive beskrevet yderligere.

Rapportering og datanavigering					
Generelle teknologier (Drag&Relate, alarmer, brugerfladetilpasninger...)					
Finans	Salg	Service	Indkøb	Lager	Produktion
◇Kontoplan ◇Kontosegmenter ◇Bogføring af post ◇Journalpostering ◇Ændringsbilag ◇Periodisk transaktion ◇Flere valutaer ◇Finansrapporter ◇Budget ◇Omkostningssted ◇Moms ◇Flere perioder ◇Indbetaling ◇Checks ◇Kreditter ◇Bilag ◇Betalingshenstand ◇Kontoudskrift og afstemning	◇Tilbud ◇Ordre ◇Faktura ◇Levering ◇Returnering ◇Prisliste i flere valutaer ◇Kundestyring ◇Beregning af bruttooverskud ◇Kontaktstyring ◇Muligheder og pipelinestyring ◇Outlook-integration	◇Styring af servicekontrakter ◇Service-administration ◇Aktivitetsoversigt ◇Vidensbase	◇Indkøbsordre ◇Restordre ◇Levering af indkøb ◇Returnering af indkøb ◇Faktura på indkøb ◇Kreditnota på indkøb	◇Lagerstyring ◇Lagerforespørgsel ◇Prisliste ◇Modtaget på lager ◇Leveret fra lager ◇Lagertransaktioner ◇Lageroverførsel ◇Serienumre ◇Styring af partier ◇Pick og pack ◇Kitting	◇Kalkulation på materialer ◇Arbejdsordrer ◇Produktionsordrer ◇Prognoser ◇MRP wizard ◇Anbefalingsrapport ◇Endelig montage håndtering
Softwareudviklingspakke					

Figur 33: Opbygning af SAP Business One, frit efter [SAP05b]

11.2.1 Finans

Finansmodulet i systemet har til opgave at holde styr på pengestrømmene i virksomheden. Her bliver der fastlagt en række faste parametre som for eksempel valutakurser og momssatser. Ligeledes indeholder finansmodulet styringen af virksomhedens konti.

Finansmodulet fungerer som et bindeled med tæt tilknytning til alle andre moduler, idet processer i moduler i næsten alle tilfælde vil have indflydelse på elementer på kontoplanen eller trække på nogle af de stamdata, som Finans det tilbyder. Herved danner finansmodulet basis for anvendelsen af resten af modulerne.

11.2.2 Salg

I dette modul er det muligt at håndtere virksomhedens kunder. At håndtere kunder omhandler ikke blot at have mulighed for at have en liste over, hvilke kunder virksomheden har, men ligeledes at kunne lave statistik, lønsomhedsanalyse og afsætningsanalyse på kunderne. Ligeledes giver modulet mulighed for at følge en bestemt proces med henblik på at sikre salg, således det kan fastlægges hvilke processer, der skal benyttes i interaktionen med en kunde. Herved kan der udarbejdes detaljerede statistikker over succeskriterierne for at opnå salg og ligeledes hvilke medarbejdere, der er mest effektive til at afslutte ordrerne. Ud over dette giver salgsmodulet også mulighed for ordrehåndtering samt prislister på virksomhedens produkter, tilpasset de forskellige kunder. Salgsmodulet arbejder specielt tæt sammen med lagerfunktionen og finansmodulet.

11.2.3 Service

Servicemodulet tilbyder organisering af service til virksomhedens samarbejdspartnere. Modulet kan betragtes som en option, idet det ikke altid er relevant i alle typer virksomheder. I modulet er der mulighed for at opstille serviceprogrammer og procedurer for at kunne håndtere, planlægge samt prisfastsætte disse. Desuden kan modulet bruges som en vidensdatabase, det er muligt at trække på under rådgivningen af kunder.

11.2.4 Indkøb

Indkøbsmodulet står primært for at for at håndtere samarbejde med virksomhedens leverandører til køb af råvarer, halvfabrikata samt andre ydelser. Her bliver bestillinger, leveringer og returneringer af varer håndteret. Modulet arbejder her sammen med lagermodulet, som håndterer den fysiske omgang med varerne. Ved at benytte indkøbsmodulet får virksomheden et redskab til styre virksomhedens leverandører samt udarbejde statistikker over hvilke leverandører, der handles mest med og opnås de største fordele ved.

11.2.5 Lager

Lagerstyringsmodulet har til opgave at holde virksomhedens lagerstatus opdateret. Herved kan modulet til enhver tid meddele hvilke varer, som er på hvilke lagre, samt værdien af disse varer. Dette sker ud fra de fast definerede lagerprincipper, som også bliver

defineret i dette modul. Lagermodulet har ligeledes til opgave at styre priser på delprodukter, som bruges til produktion, serienumre på produkter, samt indleveringer og udleveringer til virksomhedens lager. Derudover bliver forsendelses- og pakningsprocedurer registreret på lageret, for at kunne hjælpe med fastlæggelse af omkostningerne forbundet med dette. Lageret arbejder specielt sammen med indkøbsfunktionen, idet varemangel typisk vil resultere i en ny indkøbsordre. Ligeledes tilgås lagermodulet fra salgsmodulet med forespørgsler til den aktuelle lagerstatus.

11.2.6 Produktion

Produktionsstyringsmodulet har til formål at planlægge og styre produktionen i virksomheden. Dette medfører styring af råvarer brugt i produktion, pluklister samt beregning af, hvad produktionsprisen på produkter er. Ligeledes styrer modulet hvilke varer og serier, der bliver produceret, og hvornår dette sker. Dette modul fungerer altså som et værktøj til at optimere arbejdskraften i virksomheden og derved få mest muligt ud af sit produktionsapparat. I virksomheder, som udelukkende beskæftiger sig med serviceydelser, har produktionsstyringsmodulet ikke den største anvendelse. Modulet arbejder tæt sammen med lagermodulet, hvor det skal sikre, at der altid kan leveres de fornødne råvarer til produktionen.

Med disse seks delsystemer får virksomheden et system, som muliggør styring af langt de fleste faktorer i hverdagen. Ved integrationen opnås en sammenhæng mellem systemerne, som også er hovedtankegangen ved den patenterede "Drag & Relate" funktion. Ideen med denne er, at det skal være muligt for brugeren at trække data rundt i systemet, som det kendes fra Microsoft Windows miljøer. Herved bliver det muligt at kombinere data på alle tænkelige måder, og grænsefladerne simplificerer processen, da disse sammenligninger ellers skulle foregå ved hjælp af opstilling af en række formler eller udførelse af en forespørgsel direkte på databaseniveau. Denne funktion er beregnet til at gøre, at den almindelige bruger, der ikke har særlig indsigt i den tekniske del af systemet, hurtig vil kunne benytte systemet til at løse sine dagligdagsopgaver.

11.3 ERP og Activity Based Costing

Efter at den generelle funktionalitet og tankegange om ERP systemer er blevet beskrevet i foregående afsnit, vil der i dette afsnit blive set på, hvordan implementeringen af ABC kan foregå i et ERP system. Dette afsnit vil først tage udgangspunkt i teorien, hvor der vil bliver arbejdet med, hvordan ERP systemer og ABC systemet kan implementeres. Herefter vil der blive taget udgangspunkt i SAP Business One for at medtage en praktisk vinkel på problemstillingen og ende ud med en analyse på, hvordan ABC kan implementeres i dette.

I dette afsnit vil der blive set på, hvordan ABC i et ERP system integreres og/eller sammenkobles. Til at belyse denne problemstilling vil der blive lagt vægt på, hvad

[KC98] skriver vedrørende sammenkoblingen. Der inddrages en artikel, der omhandler de generelle problemstillinger ved sammenkoblingen af ABC tankegangen i et generelt ERP system.

[KC98] beskriver problemstillingen ved sammenkoblingen af ABC og ERP. I Figur 34 vises de fire niveauer, som [KC98] inddeler virksomheds IT systemer i. De, der arbejdes med i problemstillingen i dette projekt, ligger på niveau to og tre. Niveau fire består af fuldt integrerede systemer, som samler data fra hele virksomheden automatisk. Disse bruges blandt andet som ledelses- og strategiværktøjer. ERP systemerne, der fokuseres på i denne rapport, vil ligge på niveau to, og ABC systemet vil være på niveau tre. Det er først, når disse integreres eventuelt med andre systemer, at niveau fire nås. Definitionen på niveau 4 er, hvor alle virksomhedens systemer interagerer fuld ud med hinanden.

System aspekter	**Niveau I systemer** *Ufuldstændig*	**Niveau II systemer** *Afrapporteringsstyret*	**Niveau III systemer** *Specialiseret*	**Niveau IV systemer** *Integreret*
Datakvalitet	◇Mange fejl ◇Stor spredning	◇Ingen overraskelse ◇Overholder revisionsstandarder	◇Delt database ◇Selvstændige systemer ◇Uformel sammenkobling	◇Fuld sammenkobling af databaser og systemer
Ekstern afrapportering	◇Utilstrækkeligt	◇Skræddersyet til afrapporteringsbehovet	◇Niveau II system vedligeholdt	◇Afrapporteringssystemer
Produkt/kunde omkostninger	◇Utilstrækkeligt	◇Utilstrækkeligt ◇Skjulte omkostninger og indtægter	◇Adskillige selvstændige ABC systemer	◇Integrerede ABM systemer
Operationel og strategisk kontrol	◇Utilstrækkeligt	◇Begrænset tilbagemelding ◇Forsinket tilbagemelding	◇Adskillige selvstændige præstationsmålende systemer	◇Operationel og strategisk præstationsmålende systemer

Figur 34: Fire niveau modellen og omkostningssystem design, frit efter [KC98]

Det vil i nogle tilfælde være svært direkte at placere de givne systemer på et bestemt niveau, da systemer tit vil indeholde funktionalitet, der ligger udenfor. Derfor foretages tit en vurdering, hvor placeringen sker efter hovedfunktionen i systemet.

	Operationel kontrol	**ABC system**
Omkostning for brugte ressourcer	Aktuelle	Standard
Hyppighed af opdateringer	Kontinuerlige	Periodiske (kvartalsvis, halvårlig eller årlige)
Krav til målinger	Meget præcise	Estimering tilpasset efter behov, mere præcis (varigheds- og intensitetsdrivere) når det er omkostningsberettiget
Ramme for system	Ansvarscenter	Hele værdikæden: Fra leverandør og produktudvikling gennem produktion, administration, kunder og postordresalg
Fokus for system	Ressourceforbrug: Omkostninger af leverede ressourcer	Ressourcebrug: omkostninger af forbrugte ressourcer
Omkostningsvariabilitet	Fokus på kortsigtede og variable omkostninger	Grad af variabilitet bliver identificeret ved hjælp af attributter, men det er ikke en central funktion. Omkostninger bliver variable eftersom ressourcetilgang følger ressourcebehov
Anvendelsesmuligheder	Mest anvendelig i forudsigelige processer. Ikke anvendelig ved vilkårlige og dømmende aktiviteter	Generel anvendelig; kan følge typen af costdriver (transaktion, varighed og direkte måling) for at bestemme typen af underliggende processer
Komplementære Systemer	Ikke finansielle målinger (Kvalitet, cyklustider)	Behovsbaseret kunde segmentanalyse; konkurrent og strategisk information

Figur 35: Sammenligning af operationel kontrol og ABC systemer, frit efter [KC98]

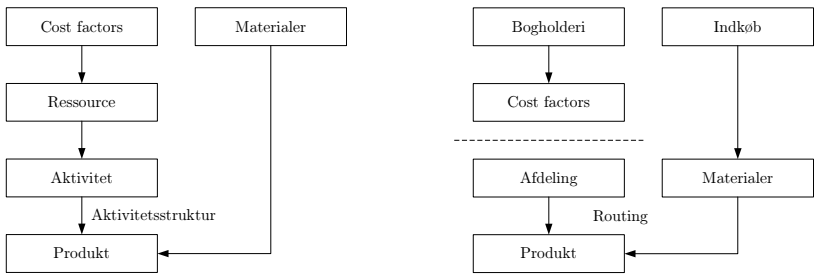
I Figur 35 illustreres fokus for de to systemer. ERP systemet er et meget præcist værktøj, hvor der lægges vægt på, aktuelle data er helt korrekte og fuldt opdaterede. Hvorimod ABC lægger mere vægt på en periodemæssig betragtning, der gerne skulle kunne bruges, selv om der kommer små ændringer i de aktuelle data. Denne viden er vigtig, når der skal

arbejdes med de to systemer, både som informationskilde, og når de forsøges integreret. Uden disse overvejelser er der risiko for at acceptere data, der bygger på et forkert grundlag. Et eksempel på dette kan være, at der ikke skal bruges "her og nu" tal til ABC, men derimod et periodegennemsnit.

Ifølge [KC98] er brugen af ERP systemer en forbedret måde at levere data til ABC, både til kontrol-, videns- og beslutningsgrundlag for ledelsen[31]. Virksomheder skal dog være opmærksomme på, om de data, der bruges i den aktuelle situation, generelt er brugbare, eller om de afspejler en kortsigtet trend[32]. Det er derfor vigtigt, at virksomheden er fuldstændig klar over, hvad de enkelte systemer og værktøjers forskelle og mål er, for ikke at anvende disse forkert og derved ikke få den ønskede effekt eller måske endda få direkte forkerte data[33].

I praksis anbefaler [KC98], at ABC anvendes til at udregne produktomkostningerne og ERP systemet til at registrere og undersøge, om der er væsentlige ændringer i omkostningerne, som kræver at virksomheden redigerer opsætningen af ABC systemet[34]. Det anses, at det er vigtigt ikke at tvinge et af systemerne til at udføre hele processen. Ydermere mener [KC98], at sammensætningen af alle virksomhedens systemer er vigtig for at sikre alle aspekter er behandlet korrekt.

[TL00] beskriver en mulig måde at sammenkoble et ABC og et ERP system. Det er vigtigt at bemærke, at der i denne artikel er lagt vægt på, at det er grundidéen i ERP systemet, der ses på, og ikke specifikke systemer. Dette kan medføre, at [TL00] i sit forsøg på en generel løsning ikke kan svare på konkrete spørgsmål og problemstillinger, der måtte være i den aktuelle situation. Dette anses ikke som et problem for artiklens anvendelse i denne rapport, da formålet med at inddrage den, er at få belyst nogle generelle problemstillinger ved sammenkoblingen af ABC og ERP systemer.



(a) Activity Based Costing (b) Enterprise Resource Planning

Figur 36: Sammenligning af ABC og ERP strukturer, frit efter [TL00]

Figur 36 sætter grundstrukturen fra et ABC og et generelt ERP system op ved siden af hinanden. Når der ses på ligheder, vil produktet være det samme i begge systemer. Over det ligger aktiviteter i ABC og afdelinger i ERP. Disse ligger meget tæt op af hinanden.

[31] [KC98], side 299
[32] [KC98], side 279 – 280
[33] [KC98], side 286 – 287
[34] [KC98], side 300

Aktiviteten kan i ABC godt indeholde en større grad af funktionalitet, hvorimod afdelingen i ERP typisk kun foretager en arbejdsgang. Overgangen fra aktivitet/afdeling sker i ABC ved aktivitetscost-driverne og i ERP ved (routing) produktionsstyring/planlægning. I ABC modellen er der et link mellem aktiviteter og ressourcer. Denne sammenhæng er ikke helt så åbenlys i ERP systemet. Materialer er identiske i begge systemer.

Læses Figur 36 på foregående side oppefra ses det, at der i ERP systemet er et led mere end i ABC modellen, nemlig indkøb og regnskabsregistrering. Denne information vil endvidere være opdateret løbende. ABC systemet bruger derimod data beregnet ud fra en tidligere periode. *Cost factors* i de to kan ikke umiddelbart sammenlignes, da de to systemer ikke har samme detaljeringsgrad, hvilket også betyder, at det ikke er muligt at lave samme link mellem cost factors og afdelinger, som eksisterer mellem ressourcer og aktiviteter i ABC systemet.

En sammenkobling mellem de to systemer er mulig til trods for de forskelle, der eksisterer. Det kræver dog, at der findes sammenfaldne eller lette-at-erstatte-data i de to systemer. Dette kan dog også betyde, at der skal laves små ændringer eller tilføjelser i et eller begge systemer for at få det til at fungere. Hvis der findes en løsning på disse problemer, vil det over tid give et mere præcist ABC grundlag og dermed også et bedre beslutningsgrundlag for ledelsen.

Der kan argumenteres for, at det er muligt at indføre en bedre fordeling af indirekte omkostninger i ERP systemet. En sådan vil dog ikke medvirke til at øge overskueligheden af ERP systemet. Det er ikke alle, der har brug for tilgang til alle funktioner i ERP systemet. Selvom der er mulighed for at afgrænse den enkelte bruger, kan der godt opstå utilsigtede situationer. [KC98] og [TL00] konkluderer begge, at det er mest hensigtsmæssigt at køre de to systemer sideløbende og integrere dem med hensyn til opdatering af data.

Der er både problemer og fordele ved at bruge ABC og ERP sammen. En fordel er, at kildedata fra ERP systemet sikrer et mere opdateret informationsgrundlag. Problemerne fremkommer, når de relevante steder at opsamle data i ERP systemet skal findes. Da ERP og ABC systemet ikke er designede til samme formål, vil der ikke fremkomme de samme poster i begge systemer. Derfor kræver det tilpasning af ABC designet for at sammenkoble de to systemer.

Der findes IT systemer, der benyttes i større virksomheder, som er en samlet løsning, hvor et ABC system inkluderet i basisproduktet. Disse systemer er ikke yderligere behandlet, da de ikke relaterer sig til problemstillingerne i denne rapport.

12 Integration af ABC og SAP Business One

Dette kapitel omhandler, i hvilket omfang SAP Business One understøtter ABC, eller om det overhovedet er muligt at benytte sig af ABC sammen med SAP Business One. Som udgangspunkt er der ikke indbygget et decideret ABC modul i SAP Business One, så det er ikke muligt at benytte sig af ABC i programmet uden at foretage nogle ændringer. Derfor vil der i dette afsnit blive gennemgået, hvad der skal gøres for at benytte sig af ABC sammen med SAP Business One. Der er to åbenlyse muligheder for at implementere ABC i forbindelse med SAP Business One. Enten kan det forsøges at blive implementeret direkte ind i systemet, eller også kan SAP Business One benyttes som et kildesystem, som kan levere data til et ABC system uden for ERP systemet[35]. Som det er beskrevet i Afsnit 11.3 på side 73, er der blevet argumenteret for, at det ikke er hensigtsmæssigt at implementere ABC systemet direkte i SAP Business One. Derfor vil SAP Business One blive brugt som kildesystem.

I de følgende sektioner vil der blive argumenteret for, at SAP Business One ikke er i stand til at levere de relevante informationer, der skal bruges i forbindelse med ABC. Derfor vil nogle af ABC registreringerne blive flyttet ud i et eksternt registreringssystem, hvorefter både dette og SAP Business One kan fungere som kildesystemer til et eksternt ABC program.

For at benytte ABC, skal der logisk set bruges følgende komponenter:

- Direkte omkostninger
- Ressourcer
- Aktiviteter
- Omkostningsobjekter

[35] [BI04], side 70

- Ressourcecost-drivere
- Aktivitetscost-drivere

I de følgende sektioner vil der blive gennemgået, hvor disse i praksis kan findes i SAP Business One og hvilke, der bør flyttes ud i et eksternt registreringssystem. Derudover vil der blive gennemgået hvilke tabeller i den bagvedliggende Microsoft SQL Server database, der skal udvælges fra for at kunne hente data direkte ud fra ERP system og ind et eksternt ABC system.

Udover identifikation af de forskellige komponenter vil der blive diskuteret, hvordan SAP Business One håndterer omkostningsbegrebet.

12.1 Omkostninger

Når SAP Business One skal benyttes i forbindelse med et ABC system, er det vigtigt at få omkostningsbegrebet lagt fast. Dette skyldes, at omkostninger kan opgøres på forskellige måder i ERP systemer, hvilket kan påvirke resultatet. Generelt er opfattelsen af ABC, at det er mest hensigtsmæssigt at basere det på omkostninger[36]. Når det er registreringerne af omkostninger, som skal benyttes, er det vigtigt at få fastlagt, hvilke steder data skal komme fra, samt hvordan disse skal bestemmes.

En omkostning kan variere ud fra, hvordan den fastsættes, og SAP Business One giver mulighed for at benytte forskellige principper til beregningen af en omkostning. Beregningsformerne tager udgangspunkt i teorien og udgør; Standard, glidende gennemsnit og FIFO som SAP Business One. Der kan argumenteres for, at der findes andre beregningsformer, men disse understøttes ikke direkte i SAP Business One. Hvilken beregningsform, der skal benyttes, er generelt op til virksomheden at beslutte, men der kan være forhold i omgivelserne og markedet, som gør nogle mere hensigtsmæssige.

12.1.1 Standardprincippet

Standardprincippet for beregning af omkostninger tager udgangspunkt i en prisliste, som er i systemet. Denne prisliste er genereret ud fra, hvad virksomheden forventer priserne vil være på ressourcerne. Den kan ligeledes hentes fra eksterne samarbejdspartnere som leverandører med videre. Det er også muligt for virksomheden at basere prislisten således, prisen stiger med en vis procentdel over en periode, netop hvis der er en typisk fast prisudvikling på ressourcen. Princippet lægger op til, at omkostningen på ressourcen lægger sig tæt op af den udgift, virksomheden har, hvis den skal genanskaffe ressourcen. Fordelen ved dette er, at virksomheden altid har den aktuelle pris på ressourcen, og der bliver taget højde for den under beregningen af omkostningen. Ulempen kan være, at princippet ikke tager højde for, at virksomheden kan have et stort lager af varer, hvor

[36][BI04], side 38

der har foregået en stor prisudvikling. Herved kan der forekomme upræcis fastlæggelse af omkostninger.

12.1.2 Glidende gennemsnit

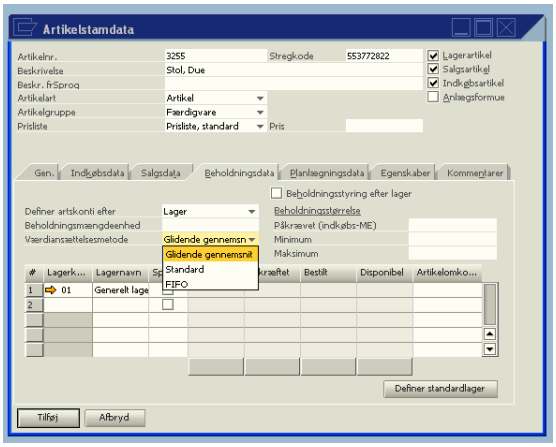
Glidende gennemsnit bliver beregnet som gennemsnitsprisen på et givet tidspunkt i en periode[37]. Perioden er valgfri, men typisk er det halvårlige eller årlige perioder, der vælges. Herved vil en omkostning blive beregnet ud fra gennemsnitsprisen på ressourcer på et givet tidspunkt. Ulempen ved denne metode er, at store udsving i priserne på ressourcer i en kort periode vil give misvisninger af omkostninger, så længe den periode indgår i det glidende gennemsnit. Ligeledes vil en pludselig ændring i omkostningen på en ressource tage lang tid, inden den får indflydelse på prisen i det glidende gennemsnit. Modsat kan der siges, at regelmæssige svingninger i prisen ikke får indflydelse på omkostningerne, og virksomheden opnår derfor en solid base for fastlæggelse af sine omkostninger.

12.1.3 First In First Out

First In First Out (FIFO) metoden fungerer, som navnet antyder, på den måde, at ressourcerne bliver forbrugt i den rækkefølge, de er tilgået lageret[38]. Således bliver der i ERP systemet udarbejdet en liste over hvilke varer, som er købt hvornår samt til hvilken pris. Herefter bliver forbruget af ressourcer baseret på den liste. Ulemperne ved denne metode er, at hvis der er udsving af priserne, bliver det utroligt svært arbejde med omkostningerne, idet priserne kan svinge meget. Herved kan det være vanskeligt, hvis virksomheden har mange ressourcer på lager, og prisen pludseligt ændres, fordi omkostningerne ikke vil følge med, før lageret er forbrugt. Modsat kan der argumenteres for, at virksomheden altid har mulighed for at beregne den reelle omkostning, der har været ved forbruget af ressourcen.

Ud fra disse tre principper er det valgt at basere omkostningerne ud fra en prisliste i SAP Business One. Måden dette bliver identificeret på er, at der i artikelstamdata i lagerstyringsmodulet bliver valgt den aktuelle værdiansættelsesmetode for lageret. Dette ses i Figur 37 på næste side. Efter dette er valgt, sørger systemet for at lave beregningerne efter denne metode, og omkostningerne får den form, som ønskes til ABC delen. Ved at vælge denne metode er der blevet valgt, hvordan omkostningerne bliver opfattet imellem ERP og ABC systemet. Opgaven er herefter at kunne trække disse data ud fra ERP systemet, således at de kommer til at afspejle de korrekte ressourcepuljer. Der vil blive udarbejdet et forslag til dette i Afsnit 12.3 på den følgende side.

[37] [CN05], side 155 – 157
[38] [CN05], side 156 – 157



Figur 37: Artikeldata i SAP Business One

12.2 Identificering af direkte omkostninger

Det første, der skal findes, når et ABC system skal udarbejdes, er de direkte omkostninger. De direkte omkostninger kan i SAP Business One identificeres ved hjælp af projektkoder. Allerede fra tilbudsøjeblikket kan der tilføjes en projektkode til hver enkelt linie, således at de enkelte linier kan henføres til et givet projekt. Disse projekter kan i dette tilfælde anvendes til at navngive de enkelte omkostningsobjekter. I denne rapport vil omkostningsobjekterne være produkterne, så en projektkode kunne for eksempel være stolen med produktnavnet "Due".

Produktionsordren kan derefter laves på baggrund af en kundeordre. Data om produktionsordrerne kan findes i tabellerne `OWOR` og `WOR1`. Når produktionsordren bliver udført, sker dette i menupunktet **tilgang fra produktion** under modulet **Produktion**. Her kan projektkoden indtastes, altså i dette tilfælde omkostningsobjektet. På denne måde er de direkte omkostninger i forhold til de valgte omkostningsobjekter blevet adskilt.

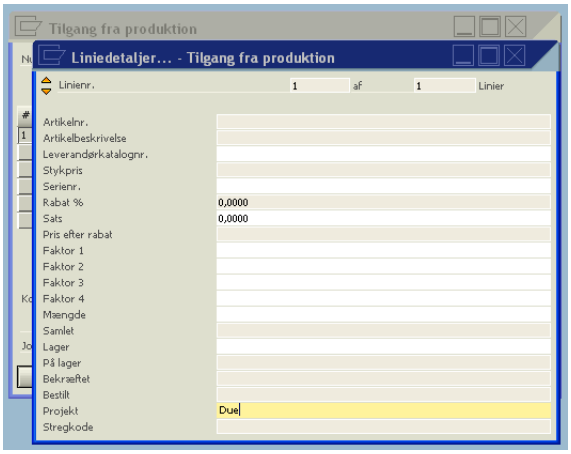
Data om tilgang fra produktionen kan findes i tabellerne `OIGN` og `IGN1`. Tabellen `IGN1` indeholder alle liniedata, hvori både artikelnummer og den tilhørende projektkode kan findes. Det vil sige, at i denne tabel vil der kunne findes de direkte omkostninger i forhold til de givne projektkoder.

Figur 38 på næste side viser et skærmbillede fra SAP Business One, hvor der bliver tildelt en tilgang fra produktionen til omkostningsobjektet "Due".

12.3 Identificering af ressourcer

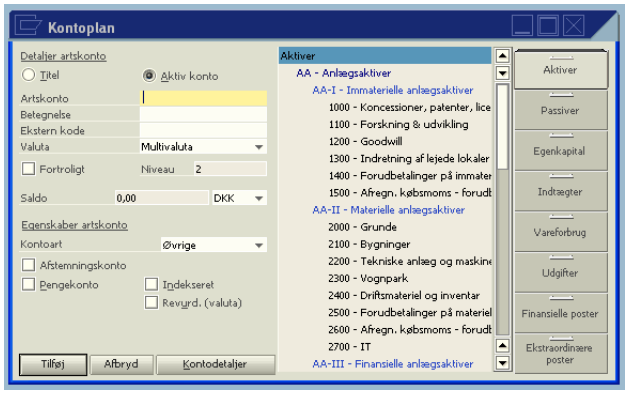
Når de direkte omkostninger er fundet, er ressourcerne det næste, der skal identificeres. Disse kan ifølge litteraturen findes ud fra virksomhedens kontoplan[39]. Kontoplanen er

[39] [KC98], side 86



Figur 38: Produktkoder i SAP Business One

i SAP Business One direkte tilgængelig i modulet **Økonomi** og underpunktet **Kontoplan**. Denne kontoplan er vist i Figur 39. Et af kritikpunkterne ved at bruge kontoplanen som baggrund for ressourcerne er, at det ikke er muligt at adskille direkte og indirekte omkostninger. Men dette løses, da de direkte omkostninger allerede er identificeret i Afsnit 12.2 på forrige side ved hjælp af projektkoder. På den måde kan der relativt nemt finde ud af, hvor stort et beløb på hver konto, der er direkte i forhold til et omkostningsobjekt, og derfor kan fratrækkes.



Figur 39: Kontoplan i SAP Business One

Data fra kontoplanen kan findes i `OACT` tabellen i Microsoft SQL Server, der ligger bag SAP Business One. Kolonnen **AccName** indeholder navnet på kontoen, og kolonnen **CurrTotal** indeholder saldoen på den pågældende konto. For eksempel udvælger SQL-sætning 9 kontoen med kontonummeret "60930", som er *Husleje* og dennes saldo. På denne måde kan alle de konti, der er brug for, udvælges og herefter indsættes i ABC systemet.

```
1 SELECT AccName, CurrTotal FROM OACT WHERE AccCode=60930
```

SQL-sætning 9: SQL til udvælgelse af ressourcer

12.4 Identificering af aktiviteter

Identificering af aktiviteter i ABC systemet er en vigtig proces for virksomheden. Arbejdsprocesser skal specificeres og deles op således, det er muligt at beskrive dem som en mængde af aktiviteter, som derefter bliver henledt til omkostningsobjektet. Umiddelbart stiller SAP Business One ikke direkte en mulighed for at trække aktiviteterne ud af systemet.

Stedet, hvor det er naturligt at lede efter aktiviteter, vil være i produktionsmodulet. Dette modul tager dog udgangspunkt i logistik i produktionen i form af styklister, produktionsordrer med videre. For at identificere aktiviteter er der brug for at kunne identificere de enkelte arbejdsprocesser i produktionen, hvilket ikke er muligt.

Et sted i SAP Business One, hvor det er muligt at finde nogle aktiviteter, er under salgsmuligheder, hvor det er muligt at trække data og statistik over møder og tiltag gjort overfor kunder i forbindelse med salg. Der kan argumenteres for, at disse møder og samtaler skal bringes ned som aktiviteter i ABC systemet, idet det er en omkostning, som skal fordeles ned på omkostningsobjekterne i virksomheden. Herved vil virksomheden muliggøre fordeling af salgsomkostninger ned på hvert enkelt omkostningsobjekt, hvilket virker fornuftigt, idet der kan være stor forskel på salgsomkostningerne ved hvert enkelt omkostningsobjekt. Disse data kan trækkes ud fra tabellen `OPR1` i SAP Business One. Et problem ved dette udtræk af data er, at det kun er de aktuelle møder, som bliver trukket ud, og forberedelsestid til møder bliver derfor ikke inkluderet i disse. Hermed skal forberedelsestiden komme et andet sted fra. Dette kan eksempelvis være et *time/sag* system. En beskrivelse og implementering af et sådant system findes i Afsnit 12.6 på næste side.

Der er til aktiviteter brug for yderligere eksternt data, som vil kunne blive leveret af et decideret produktionsstyringssystem, hvor tidsforbrug og de forskellige arbejdsopgaver i form af aktiviteter er angivet. Dette system vil have et naturligt samarbejde med SAP Business One i form af personalehåndtering, lønudbetaling med videre, og det vil kunne tilgås ved hjælp af det interface, SAP Business One stiller til rådighed. Der eksisterer yderligere aktiviteter, som ikke har decideret tilknytning til produktion eller salg. Disse skal også håndteres, hvilket typisk vil blive gjort i virksomhedens ABC system. Her vil SAP Business One ikke kunne give nogle brugbare data. I forbindelse med identifikation af aktiviteter vil det være nødvendigt at analysere hvilke aktiviteter, som indgår i virksomhedens drift. Her kan procesdiagrammer benyttes, hvilket er beskrevet i Afsnit 4.2 på side 23.

Umiddelbart vil SAP Business One ikke levere en stor mængde data til aktivitetshåndteringen til et ABC system. Begrebet *aktivitet* findes dog i SAP Business One, men har her en anden betydning. Baggrunden for, at det ikke er muligt at trække aktiviteter er, at SAP Business One ikke er udarbejdet til at understøtte den ret specifikke ABC tankegang, hvor den aktivitetsbaserede fordeling er unik.

12.5 Identificering af omkostningsobjekter

I ABC kan der vælges forskellige objekter som omkostningsobjekter. Derfor skal der først og fremmest besluttes, hvad virksomheden ønsker som omkostningsobjekt. De to mest åbenlyse omkostningsobjekter er *kunder* og *produkter*. Disse to objekter kan findes to forskellige steder i SAP Business One.

Hvis kunder ønskes som omkostningsobjekter, kan disse findes i **Forretningspartner** modulet under **Forretningspartner-stamdata** undermenuen. Her findes alle forretningspartner, herunder også kunderne.

Data fra **Forretningspartner-stamdata** menuen kan findes i `OCRD` tabellen. Her findes alle forretningspartnere som for eksempel leverandører og kunder. Attributten **CardName** indeholder navnet på forretningspartneren. Men da der kun ønskes kunder som omkostningsobjekter, skal de andre forretningspartnere filtreres fra. Dette sker ved hjælp af kolonnen **CardType**. I alle rækker, hvor kolonnen **CardType** indeholder et `C`, er forretningspartneren en kunde. SQL-sætning 10 viser, hvordan alle kunder udvælges fra SAP Business One

```
1 SELECT CardName FROM OCRD WHERE CardType='C'
```

SQL-sætning 10: SQL til udvælgelse af kunder

Vælges produkter derimod som omkostningsobjekter, skal disse data findes et andet sted. Disse data findes i modulet **Lagerstyring** og underpunktet **Artikelstamdata**.

Data fra **Artikelstamdata** menupunktet findes i tabellen `OITM`. Attributten **ItemName** angiver navnet på produktet, og attributten **ItemCode** angiver produktnummeret. SQL-sætning 11 udvælger alle produkter og deres produktnummer, så disse kan bruges direkte i et eksternt ABC system.

```
1 SELECT ItemCode, ItemName FROM OITM
```

SQL-sætning 11: SQL til udvælgelse af produkter

I denne rapport antages det, at produkterne er omkostningsobjekter, hvorfor den sidste mulighed skal anvendes.

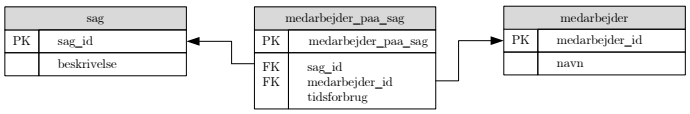
12.6 Identificering af ressource cost-drivere

For at få mulighed for at fordele ressourcernes omkostninger på aktiviteter er det nødvendigt at registrere eksempelvis medarbejderes tidsforbrug på aktiviteterne. SAP Business One giver begrænset mulighed for at registrere, hvor lang tid en medarbejder bruger på eksempelvis et møde med en kunde. Dette giver dog kun en begrænset hjælp til at beregne ressource cost-drivere. Selvom tiden brugt på møder bliver registreret, kendes jo ikke den tid, medarbejderen bruger på andre aktiviteter. Derudover vil brug af kunder som omkostningsobjekter medføre, at eksempelvis omkostninger ved møder

bliver direkte, og altså ikke skal fordeles. Endvidere registreres kun, hvor lang tid selve mødet har taget og ikke, hvor lang tids forberedelse medarbejderen har haft før mødet.

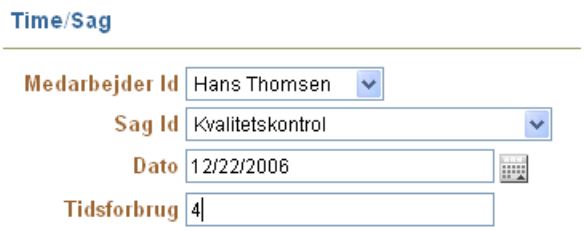
Derfor er det nødvendigt at anvende systemer udover SAP Business One til at registrere medarbejdernes tidsforbrug. Afhængig af hvilke ressourcer, der arbejdes med i den aktuelle ABC implementation, kan det også være nødvendigt at registrere forbrug af eksempelvis råmaterialer. For at udvikle applikationer til registrering af ressourceforbrug kan Oracle Application Express anvendes. Denne giver mulighed for hurtigt at udvikle grafiske brugergrænseflader, der giver let adgang til at oprette og ændre data.

Til udvikling af denne time/Sag applikation i Oracle Application Express kræves først et tabeldesign. Dette kan ses i Figur 40.



Figur 40: Tabeldesign for time/sag system

Tabeldesignet viser, at en "Sag" (aktivitet) har en beskrivelse og en syntetisk primærnøgle. Da dette blot er et eksempel på en applikation, er mængden af oplysninger reduceret til et minimum. Det kan naturligvis udvides til at håndtere yderligere relevante oplysninger. I SAP Business One registreres medarbejdere med et medarbejdernummer. For senere at kunne sammenkoble data fra SAP Business One med data fra time/sag gemmes medarbejdernummeret fra SAP Business One også i time/sag. Da en medarbejder kan arbejde på flere aktiviteter, og en aktivitet kan have flere medarbejdere tilknyttet, er det nødvendigt med en mange-til-mange relation mellem `Medarbejder` og `Sag`. Derudover skal det være muligt at periodisere en medarbejders tidsforbrug, så en medarbejder kan også arbejde på den samme aktivitet flere gange. Dermed skal tidspunktet og tidsforbruget for arbejdet på aktiviteten registreres. Tabellen `Medarbejder_paa_sag` indeholder også en syntetisk primærnøgle. SQL til disse tabeller kan ses i Bilag C på side 98.



Figur 41: Registrering af tidsforbrug for medarbejder

I Figur 41 ses et skærmbillede fra time/sag registreringsapplikationen. I Figur 42 på modstående side ses afrapportering af ressource cost-driverne. Disse er beregnet ved hjælp af SQL-sætning 12 på næste side.

For at simplificere eksemplet udvælger SQL-sætningen ikke ressource cost-driverne for

Time/Sag

Navn	Sag	Forbrug
Hans Thomsen	Kvalitetskontrol	41%
Hans Thomsen	Omstilling af maskiner	24%
Hans Thomsen	Rekvisition af råvarer	20%
Hans Thomsen	Undersøge om fejlene kan udbedres	15%
Klaus Jensen	Montage	25%
Klaus Jensen	Produktion	75%
Per Andersen	Montage	11%
Per Andersen	Omstilling af maskiner	39%
Per Andersen	Rekvisition af råvarer	17%
Per Andersen	Udbedre fejl	33%
		1 - 10

Figur 42: Rapportering af medarbejderes tidsforbrug på aktiviteter

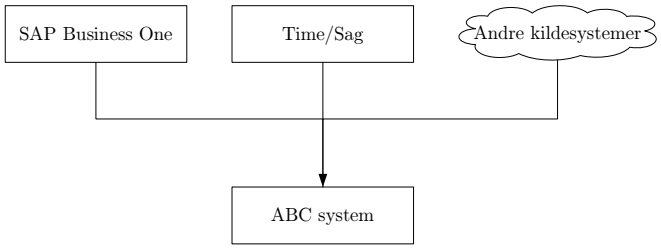
en periode. Periodisering kan dog nemt realiseres ved at filtrere dataene ved hjælp af WHERE-sætninger.

```
1  SELECT
2      MEDARBEJDER_ID, SAG_ID, SAGTIDSFORBRUG, SAMLETTIDSFORBRUG,
3      SAGTIDSFORBRUG/SAMLETTIDSFORBRUG AS PROCENTTIDSFORBRUG
4  FROM
5      (
6          SELECT
7              MEDARBEJDER_ID, SAG_ID, SUM(TIDSFORBRUG) AS SAGTIDSFORBRUG,
8              (
9                  SELECT SUM(TIDSFORBRUG)
10                 FROM MEDARBEJDER_PAA_SAG
11                 WHERE MEDARBEJDER_PAA_SAG.MEDARBEJDER_ID=YDRE.MEDARBEJDER_ID
12             ) AS SAMLETTIDSFORBRUG
13         FROM MEDARBEJDER_PAA_SAG YDRE
14         GROUP BY MEDARBEJDER_ID, SAG_ID
15     )
```

SQL-sætning 12: SQL til time/sag

Det er dog vigtigt at bemærke, at Figur 42 blot viser, at der er mulighed for at beregne ressource cost-driverne ved hjælp af Oracle Application Express. I Figur 43 på næste side kan det ses, hvordan data trækkes ud fra SAP Business One, time/sag applikationen med flere og samles i et nyt system. Det er her, data skal sammenkobles, og ressource cost-driverne fra time/sag applikationen bruges sammen med omkostningerne fra SAP Business One, og der ud fra kan kalkulationen om omkostningsfordeling foregå.

Indtil videre er der kun blevet beregnet ressource cost-driverne til ressourcer, der vedrører medarbejdere. Selvfølgelig skal andre cost-drivere findes. Nogle kan findes ved hjælp af andre kildesystemer, som indeholder informationer, der vedrører den enkelte ressource.



Figur 43: Oversigt over SAP Business One med flere som kildesystemer

12.7 Identificering af aktivitetscost-drivere

I forbindelse med identificering af aktiviteter skal der trækkes data ud fra SAP Business One. Drivere er dog ikke understøttet direkte af SAP Business One, så derfor skal dette kun være data, som skal danne grundlag for driverne. SAP Business One skal levere omkostningsobjekterne til systemet samt antallet af disse.

Aktiviteterne, der skal kædes sammen med omkostningsobjekterne ved hjælp af cost-drivere, fremkommer fra ABC systemet og SAP Business One, som det er beskrevet i Afsnit 12.4 på side 82. Idet der bliver arbejdet med data fra et eksternt system, og SAP Business One ikke understøtter cost-drivere, bliver håndteringen af disse nødt til at foregå ekstern i ABC systemet.

Sammenhængen mellem aktiviteter og omkostningsobjekter skal fastslås ved hjælp af målinger. Disse vil typisk blive udført i forbindelse med udarbejdelsen af ABC systemet, samt når der sker ændringer i virksomhedens arbejdsmetoder. Der kan endvidere implementeres en række registreringsinstrumenter, som hjælper med at registrere hvilke aktiviteter, som vedrører de enkelte omkostningsobjekter. Her vil SAP Business One ikke have nogen værktøjer, som kan hjælpe, så dette skal også registreres i eksterne værktøjer som for eksempel produktionsstyringssystemer og time/sag styringsmodulet, som er beskrevet i Afsnit 12.6 på side 83.

Når en virksomhed planlægger produktionsgangen for et produkt, skal det fastlægges hvilke afdelinger og arbejdsprocesser, produktet skal igennem. Dette er ofte i ERP systemer betegnet som routing, også nævnt i Afsnit 11.3 på side 73. Da det må antages, at virksomheden kender hvilken kapacitet, der findes i de givne afdelinger, vil det enten ved hjælp af direkte måling eller ved et estimat være muligt at beregne hvilken omkostning, aktiviteten har. På denne måde vil både direkte omkostninger såsom arbejdsløn og indirekte omkostninger som maskinomkostninger, elforbrug med mere blive målt. En mulig måde at implementere dette i praksis vil være, hvis medarbejderen selv indtaster sit forbrug i systemet. Dermed vil disse data kunne beregne den faktiske omkostning, der skal tilskrives produktet. Dette er dog ikke tilgængeligt i SAP Business One og kan derfor ikke bruges i dette projekt.

13 Opsummering

Dette hovedafsnit har omhandlet ERP systemer i samarbejde med ABC. Fokus har været på at opnå en generel forståelse af ERP systemer og samtidig opnå praktisk forståelse indenfor et specifikt ERP system, nemlig SAP Business One.

Formålet med at benytte sig af et ERP system er at sikre, at virksomheden arbejder med det samme datagrundlag, således at både arbejdsgange bliver hurtigere, og risikoen for fejl bliver minimeret. Alt dette er til gavn for kunderne, idet det sandsynligvis giver hurtigere levering og mere tilfredse kunder.

Et overblik over SAP Business One viste, at systemer er delt op i forskellige moduler, som hver varetager deres interesseområde. Ydermere giver SAP Business One mulighed for, at systemet kan udbygges med egen funktionalitet ved hjælp af den indbyggede API.

En analyse af ERP systemer og ABC viste, at det er mest hensigtsmæssigt at lade ERP systemet og ABC systemet være to uafhængige systemer, hvor ERP systemet så leverer data til ABC systemet.

I forbindelse med implementering af ABC sammen med SAP Business One blev der udarbejdet et eksternt time/sag modul i Oracle Application Express, som skal håndtere medarbejdernes tidsforbrug på forskellige aktiviteter. På denne måde findes ressourcerne i SAP Business One med udgangspunkt i kontoplanen. Cost-driverne kan ikke direkte findes i systemet, så derfor skal andre systemer bringes i anvendelse. Omkostningsobjekterne og de direkte omkostninger kan findes i SAP Business One.

Alt i alt kan det konkluderes, at SAP Business One ikke alene kan benyttes til et fuldt udbygget ABC system. Men som kildesystem kan det levere nogle data til et eksternt ABC system.

14 Konklusion

Denne rapport er en sammenskrivning af tre seminaropgaver. Dette har resulteret i tre hovedafsnit, der hver behandler et overordnet emne, dog med et fælles mål for øje. Hovedafsnittene afspejler tidsmæssigt semestrets forløb, hvor de tre seminarer har resulteret i et hovedafsnit hver med en række fælles ressourcer.

Hovedafsnit I beskæftigede sig med variabilitetsprincippet. Målet var at danne et registreringsgrundlag, som kunne bruges i forbindelse med løsning af tre af Vagn Madsens seks opgaver. Der blev udarbejdet forskellige diagrammer, der kunne bruges til at beskrive arbejdsgangene i virksomheden og på den måde skabe en bedre forståelse i forbindelse med tabeldesignet. Ud fra disse diagrammer blev virksomheden fra det gennemgående eksempel beskrevet, hvilket havde til formål at forstå virksomheden og derved bistå i det videre udviklingsarbejde. Hovedafsnittet endte ud i en applikation udarbejdet i Oracle Applikation Express, der kan håndtere registrering med hensyn til art, sted og formål. I denne applikation er det derudover muligt at få vist rapporter, som kan hjælpe til med at løse de tre valgte opgaver. Hermed vises det, at det er lykkedes at designe og implementere et system, der kan håndtere registreringen ud fra Vagn Madsens opgaver samt løse de første tre. Systemet danner ligeledes grundlag for den videre udvikling i Hovedafsnit II.

Hovedafsnit II omhandlede ABC som rapportering og variabilitetsprincippet som registrering. I dette afsnit blev der igen udarbejdet diagrammer til brug ved tabeldesign. Hovedvægten i dette afsnit var at komme frem til en løsning, der kan fungere uden at ændre ved fremgangsmåden ved registreringsgrundlaget, som blev beskrevet i Hovedafsnit I. Det vil sige, at der ikke behøves ekstra registrering af oplysninger. På denne måde, kan gamle registreringer stadig bruges med ABC som rapportering. At få registreringer ud fra variabilitetsprincippet til at danne grundlag for en ABC rapportering rummede en lang række udfordringer. Måden, dette er gjort på i praksis er, at ressourcerne fra ABC modellen tager udgangspunkt i stedsdimensionen fra variabilitetsprincippet. Dette blev gjort ud fra den betragtning, at stedsdimensionen hang tættest sammen med ressourcebegrebet, og det derfor var det naturlige valg. Ud fra registreringsprincippet, som er beskrevet i Hovedafsnit I, blev de direkte omkostninger i ABC identificeret ud fra

formålsdimensionen, idet der i systemet lå en naturlig kobling direkte til produktet. Det var derfor ikke umiddelbart muligt at benytte stedsdimensionen i dette design. Ud fra de begrænsninger, som naturligt ligger i at benytte variabilitetsprincippet som registrering i forbindelse med ABC, er der i dette hovedafsnit blevet udviklet og beskrevet et fungerende forslag til at benytte ABC med variabilitetsregistrering som grundlag.

Hovedafsnit III beskæftigede sig med ERP. Derved adskiller dette hovedafsnit sig fra de to første, da det tager en anden vinkel på problematikken. ABC tankegangen fra hovedafsnit II bliver benyttet, hvorefter et andet datagrundlag for dette blev undersøgt. Herved blev der i dette hovedafsnit udarbejdet forslag til, hvordan SAP Business One kan levere data til en ABC rapportering. I forbindelse med denne undersøgelse viste det sig, at SAP Business One ikke var i stand til at levere alle data, der skulle bruges i forbindelse med ABC. Derfor blev der sideløbende udviklet en time/sag applikation, der kan levere ressource cost-driverne. De data, som SAP Business One er i stand til at levere, er de direkte omkostninger, ressourcerne, samt omkostningsobjekterne.

De tre hovedafsnit fungerer som en emneopdeling af de tre seminaremner, som har eksisteret på semestret. Der er i denne rapport forsøgt at skabe en naturlig sammenhæng mellem disse. Dette er sket ud fra, hvordan den logiske arbejdsproces ville være, og har medvirket i en iterativ proces. Specielt i Hovedafsnit I er der inddraget en del teori og arbejde, som reelt først blev bearbejdet tidsmæssigt i Hovedafsnit II. Ud fra den logiske arbejdsmetode tilstræbes det dog, at den rækkefølge tingene opstår i dette projekt, vil være mest hensigtsmæssig under udviklingen af en sådan applikation.

Litteratur

[ADI99] Kenneth Atkins, Paul Dirksen og Zikia Askin Ince. *Oracle Designer Generation*. McGraw-Hill, 1999.

[BI03a] Per Nikolaj Bukh og Poul Israelsen. *Aktivitetsbaseret Økonomistyring*. Jurist- og Økonomforbundets Forlag, 2003.

[BI03b] Per Nikolaj Bukh og Poul Israelsen. *Aktivitetsbaseret økonomistyring: Danske virksomheders erfaringer med Activity Based Costing*. Jurist- og Økonomforbundet, 2003.

[BI04] Per Nikolaj Bukh og Poul Israelsen. *Activity Based Costing*. Jurist- og Økonomforbundets Forlag, 2004.

[Buk06] Per Nikolaj Bukh. De nye ABC-teknikker: En analyse af time-driven ABC. *Økonomistyring & Informatik*, volume 21(4), side 335-385, 2006.

[CN05] John Christensen og Mogens Nielsen. *Virksomhedens Årsregnskab*. Syddansk Universitetsforlag, 2005.

[Haz06] John Hazard. Sap's breakout: Vendor steps up efforts to capture market share. http://www.channelinsider.com/article/SAPs+Breakout+Vendor+Steps+Up+Efforts+to+Capture+Market+Share/170668_1.aspx, 2006.

[Isr93] Poul Israelsen. *Activity- versus Variability-Based Management Accounting*. Jurist- og Økonomforbundets Forlag, 1993.

[KC98] Robert S. Kaplan og Robin Cooper. *Cost & Effect*. Harward Business School Press, 1998.

[Kin01] Konrad King. *SQL Tips and Techniques*. Thomson Course Technology, 2001.

[Koc01] Christian Koch. *ERP-systemer—erfaringer, resourcer, forandringer*. Ingenøren|bøger, 2001.

[Koc06] Christopher Koch. The ABCs of ERP. http://www.cio.com/research/erp/edit/erpbasics.html, 2006.

[Lar02] Craig Larman. *Applying UML And Patterns*. Prentice Hall PTR, 2002.

[Mad63] Vagn Madsen. *Regnskabsvæsenets opgaver og problemer—I ny belysning*. Gyldendal, 1963.

[Mol94] Rolf Molich. *Brugervenlige edb-systemer*. Ingeniøren|bøger, 1994.

[Pre05] Roger S. Pressman. *Software Engineering - A Practioner's Approach*. McGraw-Hill, 2005.

[Rel06] SAP Investor Relations. Third quater 2006 results. *SAP Investor, Magazine For Investors*, side 1-2, 2006.

[Rie94] Paul Riebel. Core features of the 'einzelkosten- und deckungsbeitragsrechning. *The European Accounting Review*, side 515-543, 1994.

[SAP05a] SAP. SAP Business One - en komplet virksomhedsløsning udviklet specielt til små og mellemstore virksomheder. www.sap.com/denmark/smb /businessone/brochures/SAP_whitepaper_2005.pdf, 2005.

[SAP05b] SAP. SAP Business One, bedre lønsomhed, bedre kontol. http://www.sap.com/denmark/smb/businessone/brochures/sap_business_one_solution_brief_2005.pdf, 2005.

[SKS01] Abraham Silberschatz, Henry F. Korth og S. Sundarshan. *Database System Concepts*. McGraw-Hill, 4. udgave, 2001.

[TL00] M. Seppänen T. Lahikainen J. Paranko. Implementing activity-based costing in an enterprise ressource planning system. *Preprints of the 11th International Working Seminar on Production Economics*, 1, 2000.

A SQL til variabilitetsprincippet

```
1  CREATE TABLE OMK_ART (
2      OMK_ART_ID        NUMBER(12) PRIMARY KEY,
3      OMK_ART_FAR       NUMBER(12),
4      BESKRIVELSE       VARCHAR2(200),
5      CONSTRAINT OMK_ART_FATHER_SON FOREIGN KEY (OMK_ART_FAR) REFERENCES OMK_ART (
           OMK_ART_ID),
6      CONSTRAINT OMK_ART_INGENREKURSIV CHECK(OMK_ART_FAR <> OMK_ART)
7  );
8
9  CREATE TABLE OMK_STED (
10     OMK_STED_ID            NUMBER(12) PRIMARY KEY,
11     OMK_STED_FAR           NUMBER(12),
12     BESKRIVELSE            VARCHAR2(200),
13     CONSTRAINT OMK_STED_FATHER_SON FOREIGN KEY (OMK_STED_FAR) REFERENCES OMK_STED(
           OMK_STED_ID),
14     CONSTRAINT OMK_STED_INGENREKURSIV CHECK(OMK_STED_FAR <> OMK_STED)
15 );
16
17 CREATE TABLE OMK_FORMAAL (
18     OMK_FORMAAL_ID  NUMBER(12) PRIMARY KEY,
19     OMK_FORMAAL_FAR NUMBER(12),
20     BESKRIVELSE VARCHAR2(200),
21     CONSTRAINT OMK_FORMAAL_FATHER_SON FOREIGN KEY (OMK_FORMAAL_FAR) REFERENCES
           OMK_FORMAAL(OMK_FORMAAL_ID),
22     CONSTRAINT OMK_FORMAAL_INGENREKURSIV CHECK(OMK_FORMAAL_FAR <> OMK_FORMAAL_ID)
23 );
24
25 CREATE TABLE AKTIVITET (
26     AKTIVITET_ID     NUMBER(12) PRIMARY KEY,
27     BESKRIVELSE VARCHAR2(200)
28 );
29
30 CREATE TABLE OMKOSTNINGSREGISTRERING (
31     OMKOSTNINGSREGISTRERING_ID  NUMBER(12) PRIMARY KEY,
32     BELOEB                      NUMBER(12) NOT NULL,
33     DATO                        DATE DEFAULT SYSDATE NOT NULL,
34     OMK_ART_ID                  NUMBER(12) NOT NULL,
35     OMK_FORMAAL_ID              NUMBER(12) NOT NULL,
36     OMK_STED_ID                 NUMBER(12) NOT NULL,
37     BESKRIVELSE                 VARCHAR2(255),
38     CONSTRAINT OMK_REGISTRERING_ART FOREIGN KEY (OMK_ART_ID) REFERENCES OMK_ART(
           OMK_ART_ID),
```

```
39      CONSTRAINT OMK_REGISTRERING_FORMAAL FOREIGN KEY (OMK_FORMAAL_ID) REFERENCES
            OMK_FORMAAL(OMK_FORMAAL_ID),
40      CONSTRAINT OMK_REGISTRERING_STED FOREIGN KEY (OMK_STED_ID) REFERENCES OMK_STED
            (OMK_STED_ID)
41  );
42
43  CREATE TABLE IND_ART (
44      IND_ART_ID          NUMBER(12) PRIMARY KEY,
45      IND_ART_FAR         NUMBER(12),
46      BESKRIVELSE         VARCHAR2(200),
47      PRODUKT_ID          NUMBER(12),
48      CONSTRAINT IND_ART_FATHER_SON FOREIGN KEY (IND_ART_FAR) REFERENCES IND_ART(
            IND_ART_ID),
49      CONSTRAINT IND_ART_PRODUKT FOREIGN KEY (PRODUKT_ID) REFERENCES PRODUKT(
            PRODUKT_ID),
50      CONSTRAINT IND_ART_INGENREKURSIV CHECK(IND_ART_FAR <> IND_ART)
51  );
52
53
54  CREATE TABLE IND_STED (
55      IND_STED_ID             NUMBER(12) PRIMARY KEY,
56      IND_STED_FAR            NUMBER(12),
57      BESKRIVELSE             VARCHAR2(200),
58      CONSTRAINT IND_STED_FATHER_SON FOREIGN KEY (IND_STED_FAR) REFERENCES IND_STED(
            IND_STED_ID),
59      CONSTRAINT IND_STED_INGENREKURSIV CHECK(IND_STED_FAR <> IND_STED)
60  );
61
62
63  CREATE TABLE IND_FORMAAL (
64      IND_FORMAAL_ID      NUMBER(12) PRIMARY KEY,
65      IND_FORMAAL_FAR     NUMBER(12),
66      BESKRIVELSE         VARCHAR2(200),
67      CONSTRAINT IND_FORMAAL_FATHER_SON FOREIGN KEY (IND_FORMAAL_FAR) REFERENCES
            IND_FORMAAL(IND_FORMAAL_ID),
68      CONSTRAINT IND_FORMAAL_INGENREKURSIV CHECK(IND_FORMAAL_FAR <> IND_FORMAAL)
69  );
70
71  CREATE TABLE LAND (
72      LAND_ID     NUMBER(12) PRIMARY KEY,
73      LAND        VARCHAR2(200)
74  );
75
76  CREATE TABLE BY_LAND (
77      BY_LAND_ID      NUMBER(12) PRIMARY KEY,
78      POSTNR          VARCHAR2(20),
79      LAND_ID         NUMBER(12),
80      BYNAVN          VARCHAR2(200),
81      CONSTRAINT POST_LAND FOREIGN KEY (LAND_ID) REFERENCES LAND(LAND_ID)
82  );
83
84
85  CREATE TABLE KUNDE (
86      KUNDE_ID            NUMBER(12) PRIMARY KEY,
87      NAVN                VARCHAR2(200),
88      ADRESSE             VARCHAR2(200),
89      BY_LAND_ID          NUMBER(12),
90      TELEFONNUMMER       VARCHAR2(20),
91      IND_STED_ID         NUMBER(12),
92      CONSTRAINT KUNDE_STED FOREIGN KEY (IND_STED_ID) REFERENCES IND_STED(
            IND_STED_ID),
93      CONSTRAINT KUNDE_BY_LAND FOREIGN KEY (BY_LAND_ID) REFERENCES BY_LAND(
            BY_LAND_ID)
94  );
95
96  CREATE TABLE ORDRE (
97      ORDRE_ID                NUMBER(12) PRIMARY KEY,
98      KUNDE_ID                NUMBER(12),
```

```
99          BESTILLINGS_DATO        DATE,
100         LEVERINGS_DATO          DATE,
101         BETALINGS_DATO          DATE,
102         IND_FORMAAL_ID          NUMBER(12),
103         CONSTRAINT ORDRER_KUNDE FOREIGN KEY (KUNDE_ID) REFERENCES KUNDE(KUNDE_ID),
104         CONSTRAINT ORDRER_FORMAAL FOREIGN KEY (IND_FORMAAL_ID) REFERENCES IND_FORMAAL(
                IND_FORMAAL_ID),
105         CONSTRAINT LEVERING_EFTER_BESTILLING CHECK (LEVERINGS_DATO >= BESTILLINGS_DATO
                )
106 );
107
108 CREATE TABLE ORDRELINIE (
109         ORDRELINIE_ID       NUMBER(12) PRIMARY KEY,
110         ORDRE_ID            NUMBER(12),
111         IND_ART_ID          NUMBER(12),
112         BELOEB              NUMBER(12),
113         CONSTRAINT ORDRELINIE_KUNDE FOREIGN KEY (ORDRE_ID) REFERENCES ORDRE(ORDRE_ID),
114         CONSTRAINT ORDRELINIE_ART FOREIGN KEY (IND_ART_ID) REFERENCES IND_ART(
                IND_ART_ID)
115 );
```

B SQL til ABC

```
1  CREATE TABLE RESSOURCE (
2      RESSOURCE_ID      NUMBER(12) PRIMARY KEY,
3      BESKRIVELSE VARCHAR2(200)
4  );
5
6  CREATE TABLE OMK_STED (
7      OMK_STED_ID       NUMBER(12) PRIMARY KEY,
8      OMK_STED_FAR          NUMBER(12),
9      BESKRIVELSE VARCHAR2(200),
10     RESSOURCE_ID      NUMBER(12),
11     CONSTRAINT OMK_STED_FATHER_SON FOREIGN KEY (OMK_STED_FAR) REFERENCES OMK_STED(
           OMK_STED_ID),
12     CONSTRAINT OMK_STED_INGENREKURSIV CHECK(OMK_STED_FAR <> OMK_STED_ID),
13     CONSTRAINT OMK_STED_RESSOURCE FOREIGN KEY (RESSOURCE_ID) REFERENCES RESSOURCE(
           ressource_ID)
14 );
15
16 CREATE TABLE OMK_FORMAAL (
17     OMK_FORMAAL_ID   NUMBER(12) PRIMARY KEY,
18     OMK_FORMAAL_FAR NUMBER(12),
19     BESKRIVELSE VARCHAR2(200),
20     PRODUKT_ID   NUMBER(12),
21     CONSTRAINT OMK_FORMAAL_FATHER_SON FOREIGN KEY (OMK_FORMAAL_FAR) REFERENCES
           OMK_FORMAAL(OMK_FORMAAL_ID),
22     CONSTRAINT OMK_FORMAAL_PRODUKT FOREIGN KEY (PRODUKT_ID) REFERENCES PRODUKT(
           PRODUKT_ID),
23     CONSTRAINT OMK_FORMAAL_INGENREKURSIV CHECK(OMK_FORMAAL_FAR <> OMK_FORMAAL_ID)
24 );
25
26 CREATE TABLE AKTIVITET (
27     AKTIVITET_ID      NUMBER(12) PRIMARY KEY,
28     BESKRIVELSE VARCHAR2(200)
29 );
30
31 CREATE TABLE RESSOURCE_COST_DRIVER (
32     RESSOURCE_CD_ID NUMBER(12) PRIMARY KEY,
33     RESSOURCE_ID      NUMBER(12),
34     AKTIVITET_ID      NUMBER(12),
35     BELASTNING   NUMBER(12,4),
36     CONSTRAINT RESSOURCE_CD_RESSOURCE FOREIGN KEY (RESSOURCE_ID) REFERENCES
           RESSOURCE(RESSOURCE_ID),
45
```

```
37      CONSTRAINT RESSOURCE_CD_AKTIVITET FOREIGN KEY (AKTIVITET_ID) REFERENCES
            AKTIVITET(AKTIVITET_ID)
38  );
39
40  CREATE TABLE AKTIVITET_COST_DRIVER (
41      AKTIVITET_CD_ID NUMBER(12) PRIMARY KEY,
42      AKTIVITET_ID    NUMBER(12),
43      PRODUKT_ID  NUMBER(12),
44      BELASTNING  NUMBER(12),
45      CONSTRAINT AKTIVITET_CD_AKTIVITET FOREIGN KEY (AKTIVITET_ID) REFERENCES
            AKTIVITET(AKTIVITET_ID),
46      CONSTRAINT AKTIVITET_CD_PRODUKT FOREIGN KEY (PRODUKT_ID) REFERENCES PRODUKT(
            PRODUKT_ID)
47  );
```

C SQL til time/sag

```
1  CREATE TABLE SAG (
2      SAG_ID          NUMBER(12) PRIMARY KEY,
3      BESKRIVELSE     VARCHAR2(200)
4  );
5
6  CREATE TABLE MEDARBEJDER (
7      MEDARBEJDER_ID  NUMBER(12) PRIMARY KEY,
8      NAVN            VARCHAR2(200)
9  );
10
11 CREATE TABLE MEDARBEJDER_PAA_SAG (
12     MEDARBEJDER_PAA_SAG_ID  NUMBER(12),
13     MEDARBEJDER_ID  NUMBER(12),
14     SAG_ID          NUMBER(12),
15     DATO            DATE,
16     TIDSFORBRUG NUMBER(4,2),
17     CONSTRAINT MEDARBEJDER_PAA_SAG_PK PRIMARY KEY(MEDARBEJDER_PAA_SAG_ID),
18     CONSTRAINT MEDARBEJDER_FK FOREIGN KEY(MEDARBEJDER_ID) REFERENCES MEDARBEJDER(
           MEDARBEJDER_ID),
19     CONSTRAINT SAG_FK FOREIGN KEY(SAG_ID) REFERENCES SAG(SAG_ID)
20 );
```