eFuture – Kubebaseret ad hoc rapportering fra SQL databaser

Henrik Mønsted

Kgs. Lyngby 2003
IMM-THESIS-2003-67

eFuture – Kubebaseret ad hoc rapportering fra SQL databaser

Henrik Mønsted

Kgs. Lyngby 2003

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS: ISSN 1601-233X

Forord

Denne rapport er udarbejdet i foråret/efteråret 2003, som eksamensprojekt ved instituttet for Informatik og Matematisk Modellering på DTU. Projektopgaven er stillet af Novo Nordisk IT A/S (NNIT) og arbejdet er udført i afdelingen WebSolutions med Ahn Louise Larsen som ekstern vejleder. Michael R. Hansen, Associate Professor, fra DTU har fungeret som intern vejleder.

Det forudsættes, at læseren af denne rapport har et grundlæggende kendskab til graf- og databaseteori. Er dette ikke tilfældet findes der i appendiks 14 en kort introduktion til de teoretiske begreber og metoder, som anvendes i rapporten. Endvidere kan oplyse, at dansk anvendes i rapporten, engelsk i kildetekst og UML-diagrammer.

Lyngby, 28. november 2003

Henrik Mønsted

Resumé

Ideen bag *ad hoc rapportering* er at lade brugere, som ikke har kendskab til den underliggende *databases* teknologi og struktur, opstille deres egne rapporter som et alternativ til prædefinerede rapporter. Således bliver brugerne bedre i stand til at træffe forretningsmæssige beslutninger. Ad hoc rapportering består af tre overordnede faser: forespørgsel/dataudtrækning, formatering og præsentation af resultaterne, og analyse af resultaterne. I stedet for at præsentere resultaterne i en færdig rapport kan *multidimensionale kuber* benyttes til simpel analyse ved at lade brugeren foretage såkaldte *drill down*-operationer på kubens data.

I projektet undersøges, hvordan *grafteori* kan anvendes til at modellere tabeller og relationer i databasen, og hvordan *grafalgoritmer* kan benyttes ved dataudtrækningen. Projektmålet er at designe og implementere en *webbaseret prototype* af ad hoc værktøjet, som indeholder de tre faser i ad hoc processen, og præsentere resultatet af forespørgslen i et kube interface.

Jeg udvikler en prototype, som benytter et *XML metadata-lag* til at præsentere brugeren for et forretningsorienteret syn på den fysiske databasestruktur. Udviklingsplatformen er *Microsoft .NET framework*, og det implementerede software bliver afprøvet ved at integrere det til et eksisterende tidsregistreringssystem. Integrationen viser os, at prototypen fungerer godt, og produktet er et simpelt og stærkt alternativ til de eksisterende *business intelligence*-værktøjer. På baggrund af disse resultater konkluderer jeg, at den valgte metode er en effektiv måde at opnå ad hoc rapporterings funktionalitet på.

Projektet er udført i samarbejde med Novo Nordisk IT A/S.

Nøgleord

Ad hoc rapportering, databaser, grafteori, graflgoritmer, business intelligence, multidimensionale kuber, Drill down, Microsoft .NET framework, webbaseret prototype, XML metadata-lag.

Abstract

The idea of *ad hoc reporting* is to let end users without any knowledge of the underlying *database* technology and structure create their own reports and gain a better business insight than predefined reports can offer. Ad hoc reporting involves three main processes: extracting data, formatting and presenting the results, and analysing the results. Instead of presenting the results in a static report, *multidimensional cubes* can be used for easy analysis by letting the user *drill through* the breadth and depth of the data.

The project involves a study of how *graph theory* can be used to model entities and relations in the database, and how *graph algorithms* can be used in the data extraction process. The project goal is to make an architectural design and implement a *web based prototype* of the ad hoc query tool that includes the above three processes and presents the results in a cube interface.

I develop a prototype that uses an *XML metadata layer* that presents the user with a business-oriented view of the physical database structure. The development platform is the *Microsoft .NET framework* and the implemented software is tested by integrating to an existing time registration system. This integration shows us that the prototype works well and is a simple and powerful alternative to existing *business intelligence* tools. Based on these results I conclude that the chosen method is an effective way to obtain ad hoc reporting functionality.

The project is carried out in cooperation with Novo Nordisk IT A/S.

Keywords

Ad hoc reporting, databases, graph theory, graph algorithms, business intelligence, multidimensional cubes, Drill down, Microsoft .NET framework, web based prototype, XML metadata layer.

Indholdsfortegnelse

1 Indledning

1.1 Vision

"Et af virksomhedens største assets i dag er dens forretningsdata dvs. data der fødes til eller fra dens it-systemer. I de sidste 10 år har man i stigende grad aggregeret data i såkaldte datavarehuse, således at man med forskellige værktøjer kunne fortage rapportering på disse.

Der findes flere fortolkninger af ordet ad-hoc rapportering, generelt forstås rapportering fra et system eller datavarehus, hvor modtageren af rapporteringen i stor udstrækning selv vælger parametre for sin rapport, samt hvornår han/hun vil have rapporten. Begreber som "Business Intelligence", "OLAP" (On Line Analysis Processing) og "Data Mining" har også vundet indpas i denne forbindelse. Alle dækker de over behovet for at danne sammenhænge i data, som så kan danne grundlag for forretningsbeslutninger.

På markedet findes en lang række værktøjer til dette formål. Værktøjerne stiller meget forskellige krav til brugerne og har forskellige prisstrukturer og funktionalitet. Typisk for disse værktøjer er der en sammenhæng imellem fleksibilitet og tekniske krav til brugeren, således at meget fleksible værktøjer stiller høje tekniske krav til brugeren, og værktøjer med lav fleksibilitet stiller lave krav til brugeren. NNIT fik i 2002 udarbejdet en rapport, hvor i det fremgik, at der er et udækket behov for prisbillige rapporteringsværktøjer med høj fleksibilitet og lave krav til brugerens tekniske færdigheder[1].

Det er ambitionen at lave et sådant værktøj, der betragter databasens struktur som en graf og bruger grafteoretiske algoritmer til at udlede sammenhængene i databaser. Værktøjet skal præsentere brugeren for velkendte elementer fra sit system, hvorfra brugeren kan vælge hvilke data er interessante at kikke på. Resultatet af en søgning skal præsenteres i en *kube*, således at brugeren efterfølgende kan gruppere data i forskellige dimensioner."

Thomas Albertsen
Area Manager og opgavestiller
Novo Nordisk IT

[1] Jfr. Afsnit 1.3

1.2 Problemformulering

NNIT har tidligere leveret et system med ad hoc rapporteringsfacilitet. Ideen bag ad hoc forespørgslen er at bruge grafteori til at modellere tabeller og relationer og benytte grafteoretiske metoder til at bestemme, hvordan data udvælges.

Ideen ønskes videreudviklet og generaliseret til brug på en vilkårlig database struktur. Der ønskes en beskrivelse og diskussion af algoritmen samt en implementering med webgrænseflade. Som minimum skal Microsoft SQL Server databaser understøttes og også gerne Oracle database.

Der ønskes designet en kube, der kan bruges i forbindelse med ad hoc værktøjet. Kuben er en multidimensional struktur, hvor aggregerede data præsenteres inddelt i brugerdefinerede grupperinger. Endvidere gør kuben det muligt for brugeren at vælge en række analyseoperationer, som ændrer visningen og hjælper brugeren med at nå frem til de ønskede resultater.

Der ønskes implementeret en prototype.

Følgende problemstillinger kan evt. indgå i projektet:

- Den nuværende algoritme tager ikke hensyn til kredse i grafen.
- P.t. er alle relationer ens – kan man drage nytte af at specificere relationstyper.
- Fordele og ulemper ved at koble kuben tæt hhv. løst med dataudtrækningsalgoritmen.
- Diskussion af sikkerhedsspørgsmål og aggregeringsniveauer.

1.3 Markedsanalyse

NNIT A/S fik i 2002 udarbejdet en rapport af en gruppe HD studerende fra Handelshøjskolen i København, hvori markedet for ad hoc rapporteringsværktøjer blev analyseret med henblik på at få udviklet et nyt produkt kaldet eFuture. Produktets planlagte funktionalitet bygger på erfaringer fra et system, *Cocpit*, som NNIT tidligere har udviklet. Cocpit bliver brugt i Novo Nordisk til at registrering og behandling af klager vedr. deres produkter. Systemet indeholder en rapporteringsdel, der tillader brugerne at lave deres egne forespørgsler. Denne rapporteringsdel danner grundlag for

ideen til eFuture, men jeg vil understrege, at der væsentlig forskel på den version af eFuture, som HD rapporten beskriver, og versionen, som jeg skal udvikle i denne opgave. Rapporteringsdelen af Cocpit er en integreret del af Cocpits database, og det er derfor langt fra et generisk ad hoc rapporteringsværktøj. Desuden indbefatter oplægget til HD rapportens ide til eFuture ikke OLAP og Drill down funktionaliteter.

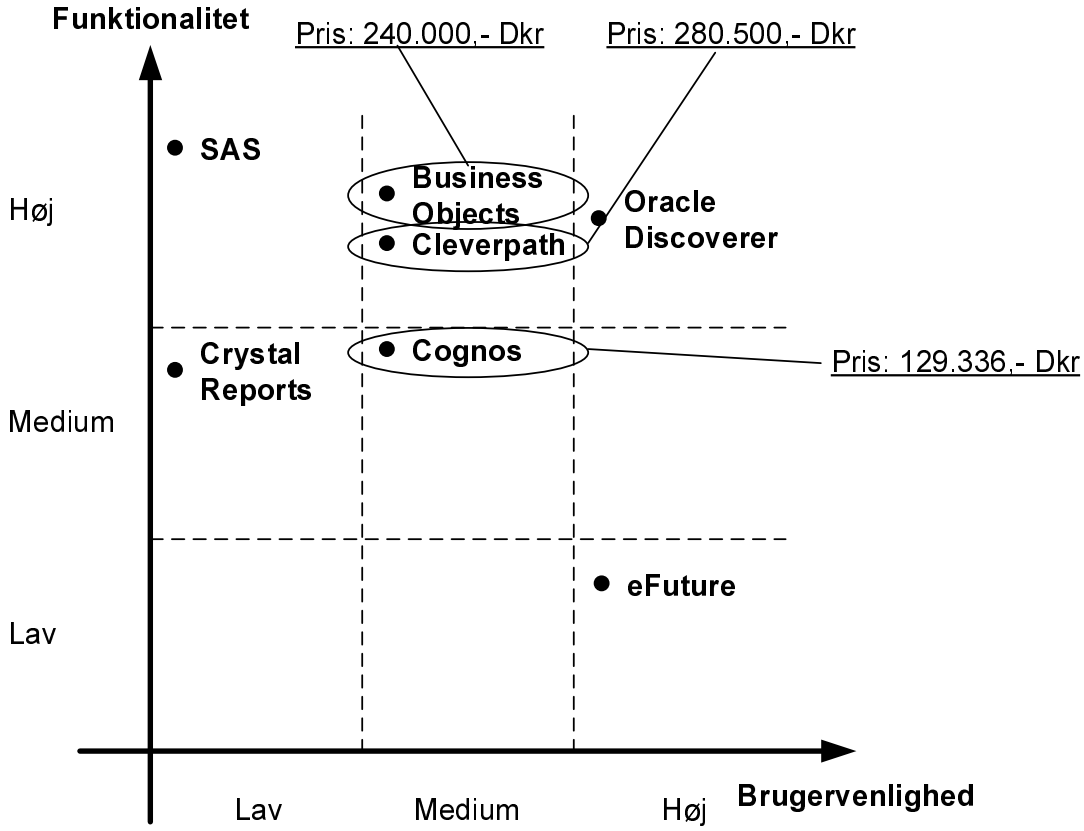
Seks eksisterende produkter blev undersøgt i HD rapporten og deres egenskaber er beskrevet i Tabel 1-1. Jeg har selv afprøvet en del af de værktøjer, som rapporten omtaler, og vil kort opsummere de konklusioner, som danner grundlag for ønsket om et nyt produkt.

	Web adgang	Avanceret Statistik	Diagramering	Kan bruges mod alle gængse	Bruger typer (for definering af rapport)	Eksport til Excel	Brugerdefineret rapporter	Schedulering	OLAP	Drill down	Antal "Ja"
Cognos Query	Ja	Nej	Ja	Ja	Bruger	Ja	Ja	Nej	Nej	Ja	6
Business Objects	Ja	Ja	Ja	Ja	Superbruger	Ja	Ja	Nej	Ja	Ja	8
Clever Path	Ja	Ja	Ja	Ja	Superbruger	Ja	Ja	Ja	Nej	Ja	8
Crystal Reports	Ja	Ja	Ja	Ja	Superbruger/ Ekspert	Ja	Ja	Nej	Nej	Nej	6
SAS	Ja	Ja	Ja	Ja	Ekspert	Ja	Ja	Ja	Ja	Ja	9
Oracle Discoverer	Ja	Nej	Ja	Ja	Alle	Ja	Ja	Ja	Ja	Ja	8
eFuture	Ja	Nej	Nej	Ja	Bruger	Ja	Ja	Nej	Nej	Nej	4

Tabel 1-1 - Produktmatrix over målpunkter

Ud fra produkternes egenskaber blev der i HD rapporten foretaget en segmentering, i forhold til hvor megen funktionalitet produkterne tilbyder, og hvor brugervenlige de er i forbindelse med at

lave ad hoc rapporter. Denne segmentering fremgår af Figur 1-1, hvor der også er vist nogle priseksempler på produkterne. Disse priser er estimater for, hvad det koste at bringe systemerne i drift inkl. omkostninger til implementering og uddannelse af brugere.



Figur 1-1 – Segmentering af produkter efter funktionalitet og brugervenlighed

HD rapporten konkluderede følgende:

- 150 konkurrerende produkter, men kun ca. 10 store konkurrenter er meget kendte hos kunderne. Dette tyder på, at der er stærk branding i markedet, og at det er let at komme ind men svært at få ordentligt fodfæste.
- Virksomheder med mere end 20 ansatte er potentielle kunder. Der var i år 2000 ca. 26.000 sådanne virksomheder i Danmark, hvilket betyder, at markedet potentielt er meget stort.
- 12 ud af 13 af de adspurgte potentielle kunder så behov for grafisk rapportering. eFuture bør udvides med sådan funktionalitet inden det introduceres på markedet.

Som nævnt omhandler rapporten en mindre funktionel udgave af eFuture end den, som jeg skal udvikle. Hvis man opdaterer Tabel 1-1 med OLAP og Drill down, vil eFuture sikkert rykke op i

segmentet for produkter med medium funktionalitet, og dermed positionere sig endnu stærkere i forhold til de øvrige produkter.

1.4 Struktur & fremgangsmåde

Rapporten er opbygget med henblik på at give læseren en fornuftig præsentation af projektets indhold. Dette vil sige, at de enkelte opgaver ikke nødvendigvis er udført i den dokumenterede rækkefølge. Rapporten er struktureret som et typisk softwareudviklingsprojekt og består af faserne:

Analyse

Her vil jeg beskrive ideen med ad hoc rapportering og indføre kube-begrebet. Desuden vil jeg definere, hvorledes databasebegreber kobles med grafteoretiske begreber. Når begreberne er på plads vil jeg præsentere et case-eksempel, som beskriver eFuture systemets funktionalitet. Endeligt vil jeg analysere de problemer, som bliver identificeret undervejs i projektet, og opskrive en kravspecifikation for systemet.

Design

Dette afsnit indledes med en overordnet arkitekturbeskrivelse af systemet. Dernæst vil jeg gennemgå designet af hver enkelt komponent, som der er fundet behov for ud fra kravspecifikationen. Jeg vil opskrive klassediagrammer og beskrive metodernes funktionalitet med henblik på implementering.

Implementering

Her vil jeg beskrive de overvejelser, som ligger til grund for den endelige implementering. Desuden vil jeg fremhæve de dele af implementeringen, som fortjener særlig opmærksomhed.

Integration til tidsregistreringssystem

Da tiden tillod det, har jeg udført et mini-projekt, som gik ud på at implementere og afprøve eFuture i forbindelse med NNITs tidsregistreringssystem. I dette afsnit vil jeg kort fortælle om processen og de erfaringer, som jeg fik undervejs.

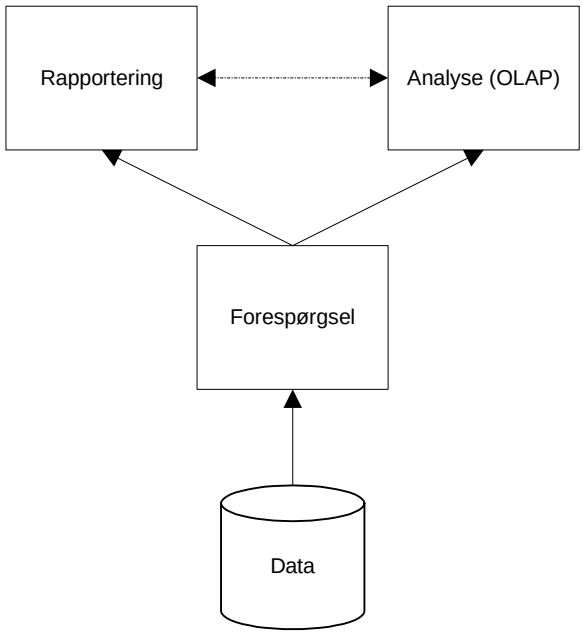
Test

Da der er tale om en prototype vil jeg ikke gennemføre en fuldstændig test af systemet. I stedet vil jeg præsentere de forskellige discipliner, der findes indenfor afprøvning af software, og vise eksempler på deres udførelse.

2 Analyse

2.1 Beskrivelse af ideen med Ad hoc rapportering

Ideen med ad hoc rapportering er at sætte brugeren i stand til selv at foretage en interaktiv rapportering frem for at skulle kigge på prædefinerede rapporter. Desuden er formålet at give brugeren en nem måde at undersøge data på uden, at han/hun behøver at kende den underliggende datastruktur. På baggrund af de resultater, som undersøgelserne viser, vil brugeren kunne træffe forretningsmæssige beslutninger. Ad hoc rapportering er en proces bestående af tre trin: Først udtrækkes de relevante data – denne del kaldes forespørgslen. Dernæst formateres data, så det præsenteres på ordentligt måde – dette kaldes rapporteringen. Endeligt kan brugeren vælge at analysere data – dette trin refereres ofte til som *Online Analytical Processing*, eller bare OLAP. Der findes en lang række ad hoc rapporteringsværktøjer af forskellig udformning, men fælles for dem alle er, at de lader brugeren opbygge sin forespørgsel ved hjælp af ord, som han/hun let kan relatere til sit daglige arbejde. Derfor behøver man ikke nødvendigvis kende til SQL for at kunne anvende værktøjet.



Figur 2-1 - Ad hoc processen

Lad os starte med at kigge nærmere på den første del af ad hoc processen. Forespørgslen, hvori data udtrækkes til den videre behandling, er af flere årsager et af de mest kritiske punkter. For at Ad hoc

værktøjerne har så lille belastning som muligt på produktionssystemet, hentes data som regel fra datavarehuse eller kopier af produktionsdatabasen. Således risikerer man ikke at ødelægge vigtige informationer i den egentlige produktionsdatabase. Produktionsdata er ikke altid nemme at arbejde med, og derfor anvender ad hoc værktøjer et metalag, som er placeret imellem brugeren og produktionsdatabasen. Metalaget benyttes til at oversætte den fysiske datastruktur til nogle forretningsmæssige begreber, som brugeren lettere kan relatere til. Hvert ad hoc værktøj på markedet har sin egen måde at benytte metalaget på og dermed forskellige måder at lade brugeren opbygge sine forespørgsler på. Nogle værktøjer anvender sit eget ”sprog”, som brugeren er nød til at lære for at have glæde af systemet. Dette er imidlertid et problem, eftersom brugerne ofte ikke er gode til boolesk logik. Undersøgelser[2] viser, at menneskets forståelse af ”and/or”-begreber ligger langt fra computerens, når det gælder om at opbygge logiske udtryk. Denne erfaring har firmaet Speedware [SPD1] udnyttet med sit produkt, ESPERANT, som lader brugeren definere sin forespørgsel ved brug af helt naturligt sprog. Det er min vurdering, at en sådan løsning er meget kompliceret at få implementeret på en tilfredsstillende måde og vil række ud over omfanget af dette projekt. I problemformuleringen lægges der op til, at man anvender grafteori til at optimere søgningen i databasen. Det vil være oplagt at benytte metalaget til at beskrive grafdatastrukteren.
Fase 2 i ad hoc processen er rapporteringen, hvor de data, som returneres ved forespørgslen, skal præsenteres for brugeren. Dette kan variere fra en simpel præsentation i et regneark, som brugeren selv kan arbejde videre med, til avancerede rapporter med diagrammer og anden grafik. I de kommercielle ad hoc værktøjer er der tydeligvis lagt stor vægt på rapporteringen. Dette skyldes i høj grad, at flotte brugergrænseflader og muligheden for at printe fancy rapporter, er en væsentlig salgsparameter. I dette projekt er der ikke stillet nogle krav til at systemet skal kunne udskrive færdige rapporter. Derimod ønskes en kubebaseret repræsentation, hvilket bringer os over i ad hoc processens sidste trin, analysen, som jeg vil beskrive i det følgende afsnit.

2.2 Kubebaseret repræsentation

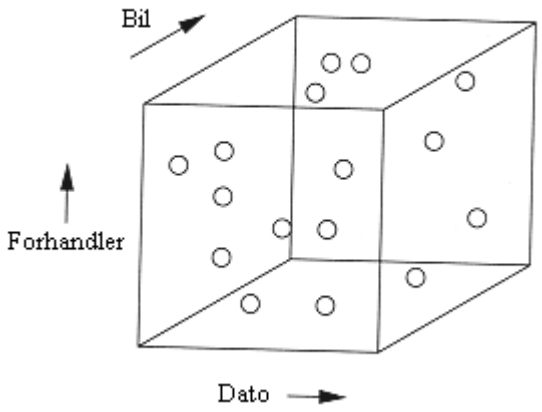
Analysen eller OLAP-delen består i at betragte data i et multidimensionalt eller tidsorienteret view, også kaldet en kube, og udføre en række operationer på kuben for at nå frem til de ønskede resultater. Kubebegrebet dækker over to forskellige typer, nemlig en *datakube* og en *formel kube*. Forskellen er, at datakuben indeholder de ”rå” data, som er hentet fra databasen, mens en formel

[2] Undersøgelser bl.a. foretaget af Dr. Paul Dorsey, Dulcian Inc.

kube viser totaler eller gennemsnit indenfor alle mulige grupperinger. For at illustrere dette vil jeg starte med at vise et eksempel på, hvordan data organiseres i et multidimensionalt rum. Eksemplet bygger på følgende database skema:

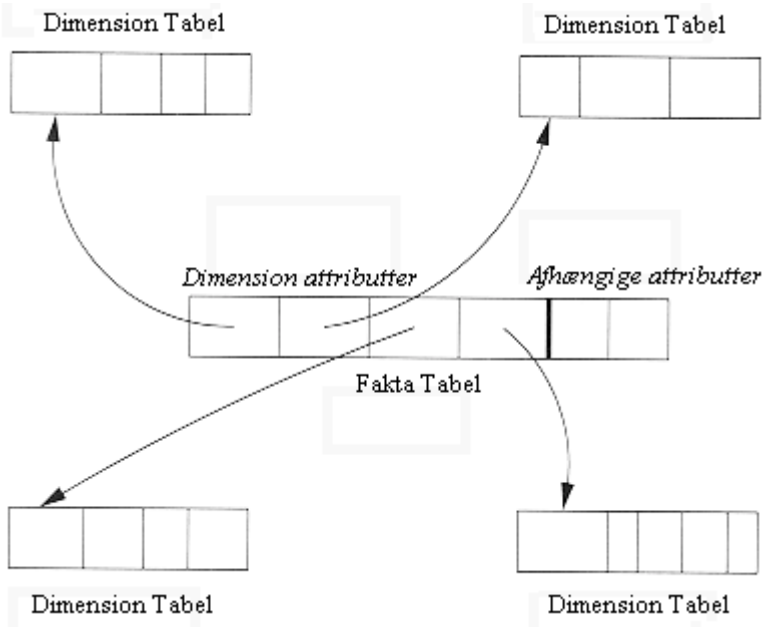
```
Bilsalg(Stelnummer, Dato, Forhandler, Pris)
Biler(Stelnummer, Model, Farve)
Forhandlere(Navn, Landsdel, Telefon)
```

På Figur 2-2 er vist en datakube, hvor hvert punkt repræsenterer et bilsalg med oplysninger om dato, forhandler og bil.



Figur 2-2 - Data organiseret i et multidimensionalt rum

Skemaet for kuben er et såkaldt stjerne-skema, som består af en fakta-tabel med referencer til en række dimensionstabeller samt en eller flere afhængige attributter. Dimensionstabellerne beskriver de mulige værdier indenfor hver dimension.



Figur 2-3 - Stjerne-skema for kuben

Et eksempel på en fakta-tabel kunne se således ud:

```
Bilsalg(Dato, Forhandler, Stelnummer, Pris)
```

Her optræder `Forhandler` som fremmednøglereference til `Navn` i dimensionstabellen, `Forhandlere`, og `Stelnummer` som fremmednøglereference til `Stelnummer` i dimensionstabellen, `Biler`. `Pris` er en afhængig attribut, og `Dato` henviser til tiden, som er en fysisk størrelse, som sjældent har sin egen tabel i databasen. I analysesammenhæng er vi ofte interesseret i f.eks. at finde ud hvor mange biler vi har solgt i en bestemt uge. For nemt at kunne svare på et sådan spørgsmål, vil det være en fordel at forestille sig, at man har følgende dimensionstabel:

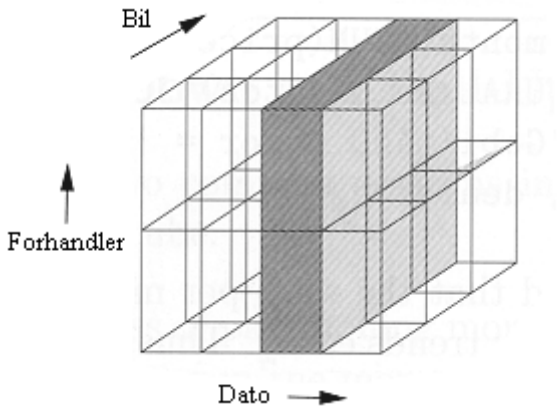
```
Tidspunkter(Dag, Uge, Måned, År)
```

I OLAP sammenhæng vil en afhængig attribut som regel være en sum, et gennemsnit eller en optælling. En kube partitioneres typisk indenfor hver dimension, således at kuben opdeles i mindre kuber. F.eks. vil det være naturligt at partitionere Bil-dimensionen efter modeller og farver. Forhandler-dimensionen kan f.eks. inddeles efter om de er placeret øst eller vest for Storebælt.

Dette kaldes også at gruppere data og refererer til SQL-udtrykkets `GROUP BY` klausul. Generelt kan man opskrive skemaet for kuben på følgende SQL-form:

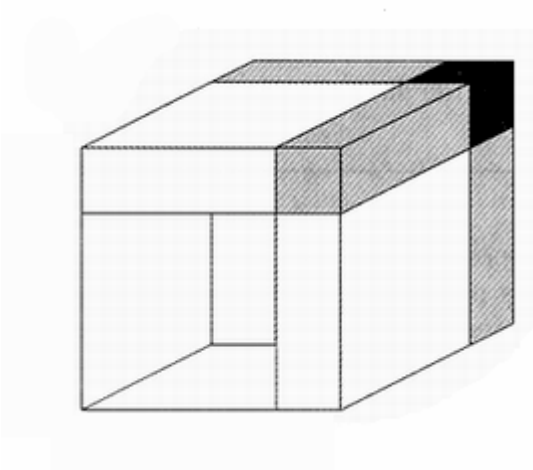
```
SELECT grupperingsattributter og aggregeringer
FROM fakta-tabeller joinet med 0 eller flere dimensionstabeller
WHERE betingelser som attributter skal opfylde
GROUP BY grupperingsattributter
```

Dette kaldes også en ”slicing and dicing” forespørgsel. På Figur 2-4 er valgt en skive(slice) i kuben. Denne skive indeholder alle bilsalg indenfor en bestemt tidsperiode for alle forhandlere og alle biler.



Figur 2-4 - En valgt skive i en partitioneret kube

Den formelle kube viser som sagt aggregeringer indenfor alle mulige grupperinger. Jeg indfører nu begrebet kubeoperatoren `KUBE(F)`, som tilfører en ekstra værdi, *, til hver dimension i fakta-tabellen `F`. Værdien * symboliserer ”alle mulige” aggregeringer indenfor den dimension, hvor den optræder.



Figur 2-5 - Kubeoperatoren illustreret

Anvender vi kubeoperatoren på `Bilsalg` fakta-tabellen, så giver `Stelnummer` attributten ikke noget ønskeligt resultat, da `Stelnummer` er primær nøgle i `Biler`. Dette medfører, at vi summerer over alle datoer og forhandlere, og at summen beregnes for hver enkelt bil med det stelnummer. I stedet udskifter vi `Stelnummer` med `Farve` og opnår følgende relation:

```
Bilsalg(Dato, Forhandler, Farve, Værdi, Antal)
```

De to nye attributter `Værdi` og `Antal` fortæller hhv. den totale pris for biler solgt på den givne dato af den givne forhandler og antallet af biler i den kategori. Ved at påfører kubeoperatoren kan man forestille sig at to tupler i relationen, `KUBE(Bilsalg)`, ser således ud:

```
('25-6-2003', 'Bents Automobiler', 'Sort', 740.000, 4)
('25-6-2003', *, 'Sort', 3.680.000, 21)
```

Den første siger, at den 25. juni 2003 solgte Bents Automobiler fire sorte biler til en samlet værdi af 740.000 Kr. Den anden siger, at d. 25. juni 2003 blev der solgt 21 sorte biler til en samlet værdi af 3,68 mil. Kr. hos alle forhandlere tilsammen. Forskellen på to tupler er, at vi har udvidet forespørgslen fra at omfatte en til at omfatte alle forhandlere. Denne operation kaldes for *Roll-up*, og går man den modsatte vej kaldes det *Drill-down*.

2.3 Kobling imellem databaser og grafer

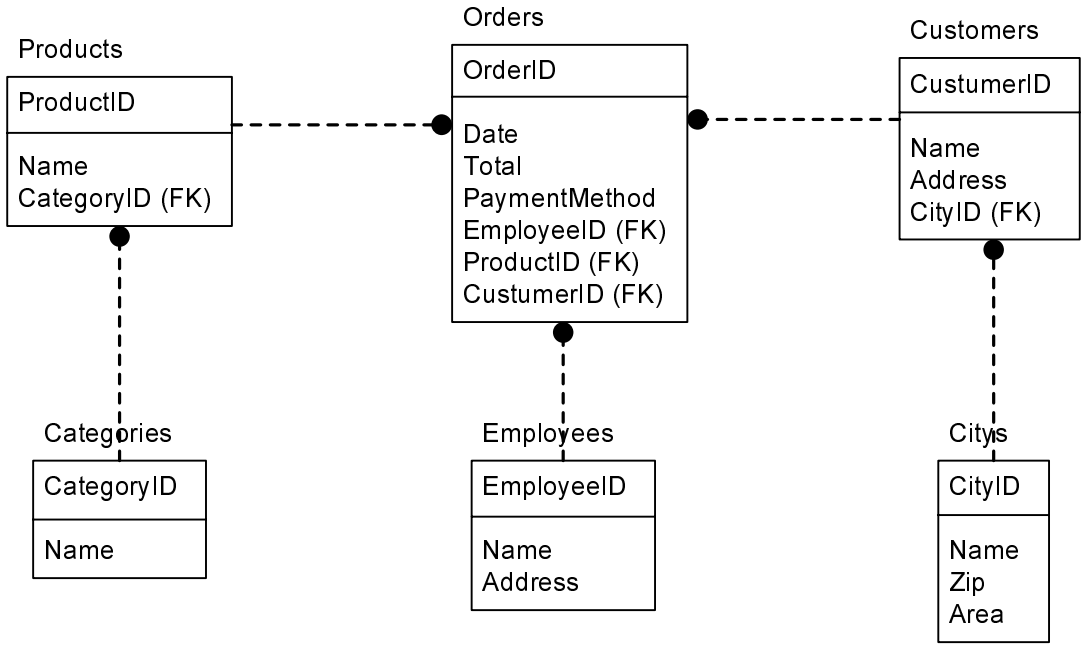
Inden jeg præsenterer en case baseret beskrivelse af systemet, der ønskes udviklet, er det nødvendigt at få defineret, hvordan databasebegreberne oversættes til grafteoretiske begreber. Uden denne kobling vil læseren ikke have det fulde udbytte af afsnit 2.4. Grafteori anvendes i mange forskellige sammenhænge, og det er naturligt også at kombinere grafteori og databaser. I dette projekt har jeg valgt at repræsentere et databasediagram ved en ikke-orienteret graf, hvor tabeller udgør punkterne og relationer udgør kanterne. Således er to punkter naboer i grafen, hvis og kun hvis der findes en relation imellem de to tabeller, som punkterne repræsenterer. Når man arbejder med databaser, stræber man altid efter at opnå det hurtigst mulige svar på de forespørgsler, man sender til databasen. I det databasediagrammet er repræsenteret ved en graf, kan vi eksempelvis anvende Dijkstras algoritme til at finde korteste veje imellem de enkelte tabeller, og måske dette kan hjælpe os, når SQL-forespørgslerne skal konstrueres. De punkter, som ligger på den korteste vej, beskriver de tabeller, som skal stå i SQL-sætningens FROM klausul. De kanter, som udgør den korteste vej, beskriver netop de join-udtryk, som skal inkluderes i SQL-sætningens WHERE klausul. I afsnit 2.5 vil jeg se nærmere på, hvilke muligheder/begrænsninger denne kobling giver i forbindelse med udviklingen af et Ad hoc rapporteringsværktøj.

2.4 Case baseret beskrivelse af systemet

Jeg vil nu beskrive et scenario, der opridser problemstillingen og viser produktets overordnede funktionalitet.

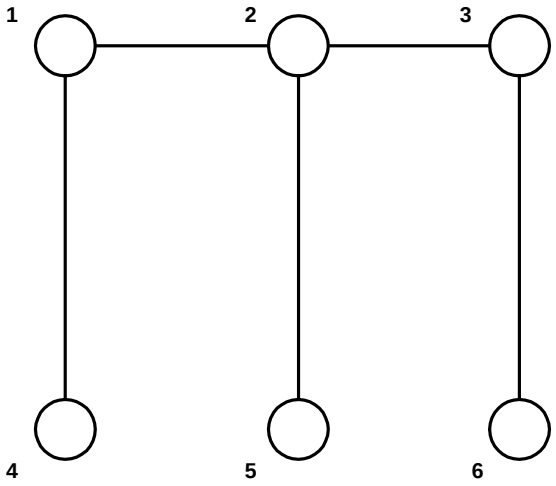
En mellemstor dansk virksomhed sælger fødevarer til private kunder i hele landet. Virksomheden registrerer alle deres ordrer i et system, som gemmer data i en Microsoft SQL Server database. Chefen for virksomheden er interesseret i at få svar på en række spørgsmål, som skal hjælpe ham med at træffe nogle vigtige forretningsmæssige beslutninger. F.eks. er virksomheden i gang med at iværksætte en stor reklamekampagne for nogle nye produkter, og for at finde ud af hvor i landet de skal annoncere, vil de gerne vide i hvilke geografiske områder, at deres lignende produkttyper sælger bedst. Et andet problem er, at chefen står overfor at skulle uddele en bonus til sine dygtigste medarbejdere, og han ønsker derfor en opgørelse over, hvor meget de enkelte medarbejdere har faktureret det seneste år. Der findes et rapporteringsmodul til det eksisterende ordresystem, men dette modul er yderst kostbart og kræver en specialist til at betjene det. I øvrigt er dette modul kun i

stand til at generere prædefinerede standardrapporter. Derfor ønsker chefen et system, som gør det muligt at få svar på førnævnte spørgsmål, og som kan betjenes uden være specialist. Endvidere ønsker man at kunne udforske resultaterne nærmere ved f.eks. at gruppere data, samt bestemme hvilken rækkefølge man ønsker at se disse grupperinger.



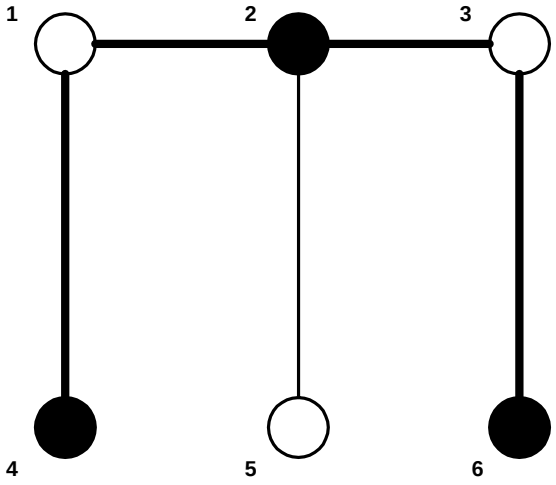
Figur 2-6 - E/R diagram for virksomhedens database

Betjeningen af systemet skal foregå via af et web interface, hvori indholdet af ordredatabasen er repræsenteret. Det forudsættes ikke, at brugeren har noget kendskab til databasens opbygning, og repræsentationen består af en oversigt med letforståelige navne, som brugeren kan relatere til sit daglige arbejde. F.eks. vælger brugeren felterne **Geografisk område**, **Produkt kategori** og **Summen af ordretotaler**. Når brugeren har valgt de felter, som han/hun ønsker at bygge sin analyse på, sendes forespørgslen til serveren. På serveren har vi brug for en komponent til at behandle forespørgslen, denne komponent kalder vi kernen. Kernens overordnede funktion er at hente de data, som skal præsenteres for brugeren. Første trin er at oversætte brugerens forespørgsel til en SQL sætning. Dette kan gøres ved brug af nogle grafteoretiske metoder. Databasens opbygning kan repræsenteres som graf, hvori tabeller udgør punkterne og relationer udgør kanterne.



Figur 2-7 - Graf som afspejler E/R diagrammet

Det er nu muligt at anvende en grafalgoritme til at bestemme den korteste vej imellem punkterne (tabellerne), som er indeholdt i forespørgslen, og således ved vi, hvilke tabeller, der skal joines i SQL sætningen. På Figur 2-8 er markeret de punkter, som svarer til brugerens valgte felter, ligesom at den korteste vej er indtegnet. Det betyder altså, at tabellerne svarende til punkt 1, 2, 3, 4 og 6 skal joines i SQL sætningen.



Figur 2-8 - Graf med valgte felter og korteste vej

Ved at sammenligne grafen med E/R diagrammet på Figur 2-6 kommer vi frem til følgende SQL sætning:

```
SELECT Citys.Area, Categories.Name, SUM(Orders.Total)
FROM Categories, Products, Orders, Customers, Citys
WHERE     Categories.CategoryID     =     Products.CategoryID     AND
      Products.ProductID = Orders.ProductID AND
      Orders.CustomerID = Customers.CustomerID AND
      Customers.CityID = Citys.CityID
GROUP BY Citys.Area, Categories.Name
```

Når SQL sætningen er blevet genereret, anvendes den til at hente data fra SQL Serveren. Endeligt skal data præsenteres for brugeren, således at han/hun kan arbejde videre med resultatet. Derfor vælger jeg at præsentere resultatet af forespørgslen i en kube, som kan vise data inddelt i grupperinger og tillader en række interaktive analysemuligheder.

Geografisk område	Produktkategori	Sum af ordretotaler
Sjælland	Kød	kr 2.485,26
	Mælkeprodukter	kr 7.895,12
	Drikkevarer	kr 4.346,00
	Slik	kr 16.255,78
	Sum	kr 30.982,16
Jylland	Kød	kr 3.321,34
	Mælkeprodukter	kr 11.566,16
	Drikkevarer	kr 6.547,98
	Slik	kr 22.456,12
	Sum	kr 43.891,60
Fyn	Kød	kr 1.786,45
	Mælkeprodukter	kr 6.588,81
	Drikkevarer	kr 3.845,32
	Slik	kr 12.875,64
	Sum	kr 25.096,22
Øvrige	Kød	kr 1.164,94
	Mælkeprodukter	kr 4.687,22
	Drikkevarer	kr 2.487,91
	Slik	kr 9.864,73
	Sum	kr 18.204,80

Figur 2-9 - Eksempel på kubepræsentation

På Figur 2-9 er vist resultatet af brugerens forespørgsel. Nu er brugeren interesseret i at se, hvordan salget fordeler sig geografisk for hver enkelt produktkategori. Ved ombytning af grupperingskolonnerne opnås præsentationen vist på Figur 2-10.

Produktkategori	Geografisk område	Sum af ordretotaler
Kød	Sjælland	kr 2.485,26
	Jylland	kr 3.321,34
	Fyn	kr 1.786,45
	Øvrige	kr 1.164,94
	Sum	kr 8.757,99
Mælkeprodukter	Sjælland	kr 7.895,12
	Jylland	kr 11.566,16
	Fyn	kr 6.588,81
	Øvrige	kr 4.687,22
	Sum	kr 30.737,31
Drikkevarer	Sjælland	kr 4.346,00
	Jylland	kr 6.547,98
	Fyn	kr 3.845,32
	Øvrige	kr 2.487,91
	Sum	kr 17.227,21
Slik	Sjælland	kr 16.255,78
	Jylland	kr 22.456,12
	Fyn	kr 12.875,64
	Øvrige	kr 9.864,73
	Sum	kr 61.452,27

Figur 2-10 - Ombytning af grupperingskolonner

Brugeren kan også vælge at skjule en eller flere kolonner i kuben. I dette tilfælde ønsker brugeren kun at kigge på summen af ordretotaler inddelt efter produktkategori. Derfor udføres en Roll-up operation på kolonnen, Produktkategori, og resultatet ses på Figur 2-11.

Produktkategori	Sum af ordretotaler
Kød	kr 8.758,00
Mælkeprodukter	kr 30.737,00
Drikkevarer	kr 17.227,00
Slik	kr 61.452,00

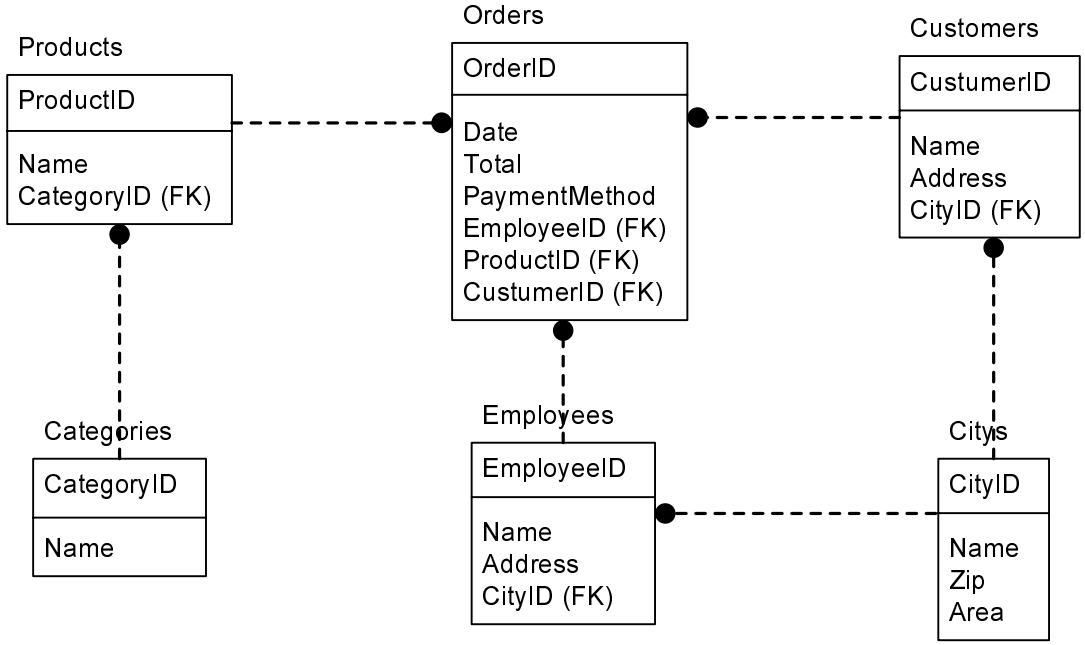
Figur 2-11 - Resultat af Roll-up operation

Herefter kan brugeren vælge at foretage en Drill-down operation på Produktkategori-kolonnen, hvilket vil bringe os tilbage til repræsentationen, som er vist på Figur 2-10.

2.5 Problemanalyse

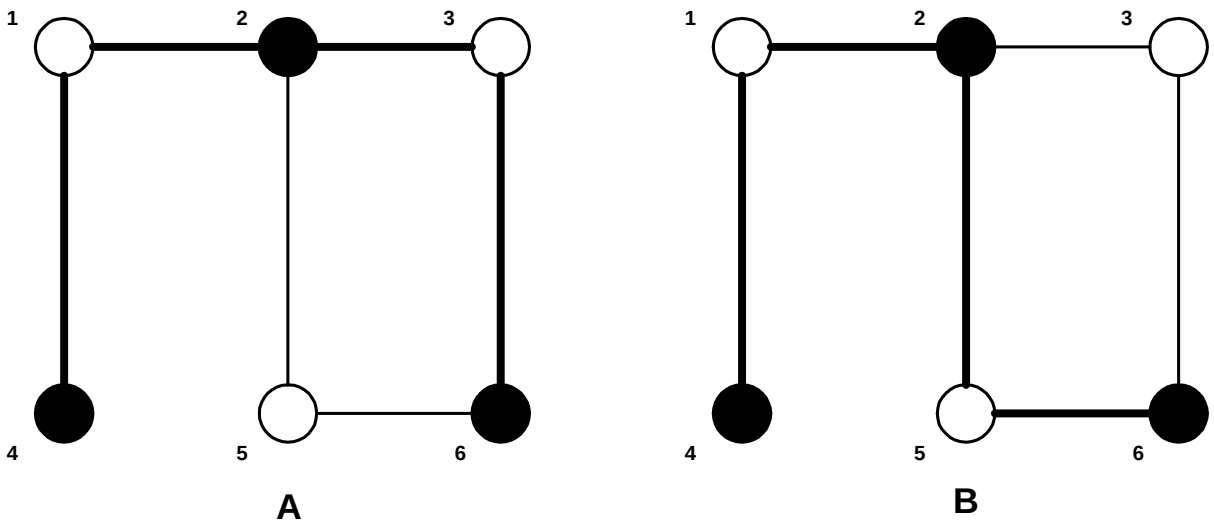
2.5.1 Forespørgsel - grafteoretiske problemer

Algoritmen for forespørgslen siger, at ved en givet mængde af tabeller, som brugeren har valgt:
For hvert par af tabeller i mængden, find den korteste vej og tilføj de tabeller og join-udtryk, som befinder sig på den vej, til SQL-udtrykket.
Dette betyder, at hvis der findes mere end en korteste vej imellem to tabeller, så kan vi ikke med sikkerhed sige, hvilken vej der vil blive valgt. Altså opstår der et problem i tilfælde, at grafen indeholder kredse. Problemet er illustreret på Figur 2-12, som stammer fra case-eksemplet. Her har jeg tilføjet en attribut, *CityID*, i Medarbejer-tabellen, som er en fremmednøglereference til *Name* i By-tabellen, således at der nu opstår en kreds i grafen.



Figur 2-12 – E/R-diagram med kreds

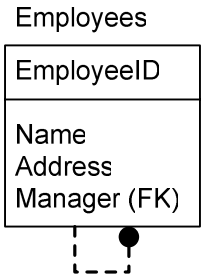
Lad os prøve med den samme forespørgsel, som i case-eksemplet, nemlig at vi vælger `Citys.Name, Categories.Name og SUM(Orders.Total)`. Det ses på Figur 2-13, at der findes to forskellige korteste veje. Vejen 4-1-2-3-6 betyder, at joinet mellem `Orders` og `Citys` sker via `Customers`, og vejen 4-1-2-5-6 betyder, at joinet mellem `Orders` og `Citys` sker via `Employees`.



Figur 2-13 - To korteste veje i grafen

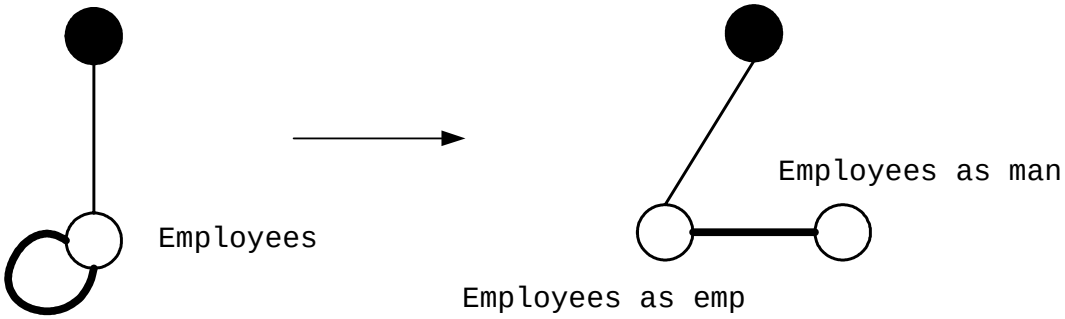
Det kan godt være, at begge join-udtryk giver mening, men hvad nu hvis brugeren kun er interesseret i den ene sammenhæng. Hvordan afgøres hvilken vej, der er den ”rigtige”? Her er man nok nødsaget til at modtage input fra brugeren om, hvilken sammenhæng han/hun ønsker at se. Et af kravene til eFuture er imidlertid, at brugeren ikke behøver at kende til den underliggende databasestruktur. Hvordan får vi beskrevet mulighederne på en måde, som brugeren forstår? Måske ske kunne man tilføje en beskrivelse af hvert join-udtryk i metadata og i tilfælde af en kreds anvende denne beskrivelse til at spørge brugeren. Jeg forudser dog, at dette kan medføre store implementeringsmæssige vanskeligheder, da logikken for håndtering af kredse vil blive ganske omfattende. Et andet problem er, at jo flere kredse grafen indeholder des flere mulige korteste veje, vil der findes, og det vil ikke være ønskværdigt at bede brugeren træffe så mange beslutninger i forbindelse med forespørgslen. På baggrund af disse overvejelser kan jeg konkludere, at det vil være en stor fordel, hvis man ikke tillader kredse i grafen. Dette skal man altså tage hensyn til, når man konstruerer metadata.

Man kunne også forestille sig, at en tabel indeholder en fremmednøglereference til primærnøglen i samme tabel, hvilket også kaldes for et *self-join*. På Figur 2-14 er vist `Employee` tabellen fra tidligere, men jeg har tilføjet en attribut, `Manager`, som refererer til `EmployeeID`. `Manager` attributten siger altså, hvem der er den pågældende medarbejderens chef.



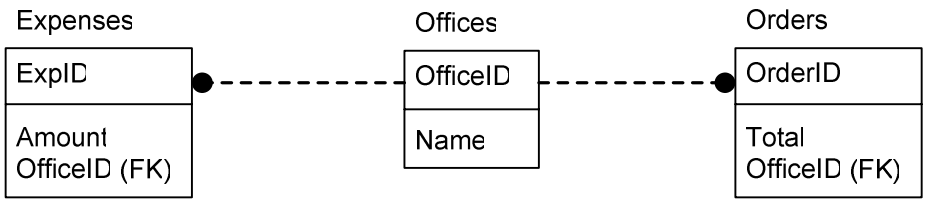
Figur 2-14 - Eksempel på self-join

Dette svarer jo faktisk til, at grafen indeholder en sløjfe. Jeg foreslår, at dette problem løses på samme måde som i SQL, hvor der oprettes et imaginært kopi af tabellen, som man efterfølgende kan referere til med et andet navn – man benytter et *alias*. I grafen vil det medføre, at to punkter repræsenterer den samme tabel. Jeg har forsøgt at illustrere på Figur 2-15, hvordan jeg forestiller mig *aliaset* kan repræsenteres i grafsammenhæng.



Figur 2-15 - Anvendelse af alias

Indtil videre har jeg behandlet alle relationstyper ens og anvendt den samme grafrepræsentation uanset om, der er tale om 1-1, 1-mange eller mange-mange relationer. I nogle tilfælde, kan en bestemt følge af relationer medføre, at man ikke opnår det ønskede resultat. Et eksempel på dette kan vises med de tre tabeller i databasediagrammet på Figur 2-16. Her ses et lille udsnit af en database for et firma, som gemmer oplysninger om hhv. udgifter og ordrer for de enkelte kontorer.



Figur 2-16 – Databasediagram med Mange-1-Mange relation

`Expenses` og `Orders` tabellerne indeholder begge fremmednøgler, der er relateret til `OfficeID` i `Offices` tabellen. Begge disse relationer har kardinaliteten Mange-1, så samlet set er der tale om en Mange-1-1-Mange relationsfølge. I Tabel 2-1 har jeg fyldt nogle data i tabeller, som jeg vil bruge til at illustrere problemstillingen med nogle konkrete tal.

Expenses

ExpID	*Amount*	*OfficeID*
1	3	1
2	5	1
3	1	2

Offices

OfficeID	*Name*
1	France
2	Spain

Orders

OrderID	*Amount*	*OfficeID*
1	5	1
2	10	1
3	7	2

Tabel 2-1 - Tabeller med data

Lad os antage, at vi vil se de samlede udgifter og de samlede ordreindtægter for hvert kontor. Dette gøres med følgende SQL sætning:

```
SELECT Offices.Name, SUM(Expenses.Total), SUM(Orders.Total)
FROM Offices, Expenses, Orders
WHERE Offices.OfficeID = Expenses.OfficeID AND
      Offices.OfficeID = Orders.OfficeID
GROUP BY Offices.Name
```

`Offices.Name`	`SUM(Expenses.Total)`	`SUM(Orders.Total)`
France	16	30
Spain	1	7

Tabel 2-2 - Resultat af SQL forespørgsel

Beregner man værdierne ud fra Tabel 2-1, så giver summen for området Frankrig hhv. 8 og 15, hvilket ikke stemmer overens med resultatet af SQL forespørgslen i Tabel 2-2. Dette skyldes den måde, som en SQL forespørgsel bliver behandlet på. Dette sker nemlig på følgende måde: Først tages krydsproduktet mellem de tabeller, som indgår i FROM linjen: (`Offices` X `Expenses`) X `Orders`, hvilket resulterer i en tuppel med 8 kolonner. Dernæst påføres WHERE sætningen, således at vi kun har de rækker tilbage, som opfylder WHERE udtrykkene. Endeligt udvælges de

kolonner, som optræder i SELECT linjen. På denne måde opnår man i vores tilfælde, at hvert omkostningsbeløb og hvert salgsbeløb bliver talt med to gange. Er der en måde, hvorpå dette problem kan undgås? En enkel måde at få det korrekte resultat på er, at indsætte DISTINCT i hver SUM udtryk, men dette holder kun så længe, at alle de værdier, der summeres over, er forskellige.
For at få de rigtige tal, bliver man nødt til at splitte forespørgslen op, således at summeringerne tages fra de enkelte tabeller og ikke fra den joinede tabel. I eFuture bør der tages højde for at sådanne relationsfølger kan forekomme i databaserne. Dette kan enten gøres ved, at man specificerer relationstyperne i metadata og tjekker om den korteste vej i forbindelse med en forespørgsel indeholder netop denne relationsfølge. En måde at modellere relationstyperne grafteoretisk kan være ved at anvende orienterede grafer eller ved at tilføje vægte til kanterne. Problemet kan dog også løses ved ikke at tillade, at denne relationsfølge optræder i grafen.

2.5.2 Kuben

Vi har nu set på de problemer, som jeg umiddelbart har kunnet identificere i relation til forespørgslen. Næste spørgsmål er, hvordan dataudtrækningsalgoritmen skal kobles sammen med kuben. Her ser jeg to muligheder, en løs og en tæt kobling. Med en løs kobling menes, at algoritmen og kuben implementeres som to adskilte komponenter. Den eneste forbindelse imellem disse er, at kube-komponenten modtager resultatet af forespørgslen fra den anden komponent, og disse data formateres efterfølgende for brugeren på web-grænsefladen. Alternativt kan man vælge at koble dataudtrækningen tæt sammen med kuben, således at algoritmen bliver en del af kuben. En væsentlig fordel ved den løse kobling er, at systemet bliver meget fleksibelt. Det ville være nemt at udvide systemet med nye moduler, som f.eks. et rapporteringsmodul til at lave færdige rapporter med, eller en selvstændig Windows applikation uden web-grænseflade. Denne løsning ligner også den, som de fleste konkurrerende produkter har valgt. De bruger ofte en form for server, som kører i baggrunden og bearbejder forespørgslerne. Den tætte kobling har den fordel, at kubeoperationerne bliver lettere at implementere, og med et fornuftigt design vil der sandsynligvis kunne spares en del dataoverførsel imellem server og klient. Ulempen ved den tætte kobling er, at de implementeringsmæssige udfordringer i forbindelse med kuben primært omhandler den grafiske præsentation, og det er sjældent en god ide at blande dette med dataudtrækningsfunktionaliteten.

Ligesom det gælder om at være opmærksom på hvilke relationer, man tager med i metadata, så er det også vigtigt at have styr på de data, som ligger i databasen. Man skal være særligt opmærksom på, at dimensionstabellerne ikke indeholder dubletter. Med dubletter mener jeg, at den samme nøgle ikke må optræde flere gange i dimensionstabellen. Dette vil nemlig medføre forkerte aggregeringer. I Tabel 2-3 er vist en faktatabel og en dimensionstabel, som jeg vil bruge til at illustrere problemet. Dimensionstabellen indeholder to ens nøgler, hvilket betyder, at tuplerne i faktatabellen vil blive multipliceret med to, og brugeren, der har lavet forespørgslen, vil tro, at salget af *Piratos* og *Matador Mix* er dobbelt så højt, som det reelt er.

Faktatabel

Navn	*KategoriID*	*Sum(ordretotaler)*	*Antal*
Coca Cola	3	1.325,64	119
Matador Mix	2	828,58	54
Mørbrad	1	1.633,95	12
Piratos	2	671,46	47
Yoghurt	4	1.083,72	134

Kategorier

KategoriID	*Navn*
1	Kød
2	Slik
3	Drikkevarer
2	Slik

Tabel 2-3 - Inkonsistens i dimensionstabel

Det er altså en vigtig del af implementeringen, at man gennemgår relationsdatabasen og sikrer sig, at dimensionstabellerne ikke indeholder dubletter. Tabel 2-3 viser også et andet problem, som kan forekomme. Den nederste tuppel i faktatabellen indeholder en reference til en nøgle, som ikke forekommer i dimensionstabellen. Dette medfører, at nogle data ikke kommer med i kuben, og det ville jo ikke være godt. Der to måder, som man kan løse problemet på. Enten kan man opdatere databasen og indsætte nye nøgler i dimensionstabellerne, eller også kan man tage højde for det i implementeringen af kuben, således at man tilføjer en ekstra dimension i kuben, hvori man opsamler de tupler i faktatabellen, som ikke er relateret til en dimension.

Der er en del udfordringer ved kuben, som er relateret til præsentationen på skærmen. Det er selvfølgelig ikke realistisk at vise kuben i en flerdimensionel grafik, og der skal derfor tages en beslutning om, hvordan kuben præsenteres i 2-D. Umiddelbart mener jeg, at den visning, som er illustreret på Figur 2-10 og Figur 2-11 giver det bedste overblik. I denne forbindelse bør det

overvejes, hvordan man præsenterer hierarkiske grupperinger. Man kunne forestille sig, at vi havde en dimension, der så således ud:

```
Kontinent - Land – Landsdel - By
```

Her er der tale om en følge af 1-mange relationer, og man bør være opmærksom, at dimensioner af denne slags altid bør optræde i den viste rækkefølge. Det samme gælder naturligvis for tidsdimensionen, som vi tidligere har snakket om. Et andet problem, som der ligeledes skal tages højde for i designet, er om det skal være muligt at oprette brugerdefinerede grupperinger. Brugeren kunne måske være interesseret i at inddele kuben i nogle intervaller, som f.eks. aldersgrupper. Dette afhænger dog først og fremmest af, om databasens udformning gør det muligt at foretage sådanne inddelinger.

Endeligt vil jeg nævne nogle sikkerhedsproblemer, der kan opstå i forbindelse med kuben. Ofte er der tale om meget følsomme data, som man analyserer, og man ønsker derfor at kontrollere, hvad de enkelte brugere har lov til at se og på hvilket niveau, at de må se data. Det kan det være, at en bruger gerne må se de aggregerede værdier, men ikke detaljerne. Et eksempel kunne være, at en projektleder vil undersøge, hvor mange sygedage, der er registreret hans/hendes projekt i forhold til de andre projekter i afdelingen. Her vil det være naturligt, at projektleder må se antallet af sygedage på projektniveau, men ikke på projektdeltagerniveau. En måde at løse dette på, kunne være at angive et fortrolighedsindeks for hver attribut i metadata. Når en bruger med lavere fortrolighedsindeks så forsøger at medtage en attribut i sin forespørgsel, vil systemet ikke tillade dette.

2.6 Kravspecifikation

Jeg har nu analyseret de problemstillinger, som er nævnt i problemformuleringen, samt de spørgsmål, der dukkede op undervejs i analysen. På baggrund af analysen og de krav, som er blevet klarlagt gennem samtaler med vejlederne, føler jeg mig nu parat til at lave en oversigt over de centrale begreber i systemet og de krav, som der stilles til det.

2.6.1 Begrebsskema

Aktører	De personer, som betjener systemet.
Systemadministrator	Person, der installerer og vedligeholder systemet.
Bruger	Den almindelige bruger af systemet.
Server	Serveren, hvorpå eFuture systemet kører.
Database	Relationsdatabase, som indeholder de data, som man ønsker at analysere.
Tabel	Svarende til et entitetsæt i E/R-modellen.
View	En tabel, som er konstrueret ud fra en eller flere tabeller i databasen.
Attribut	En søjle i tabellen med tilhørende datatype.
Relation	En forbindelse imellem to attributter.
Relationstype	1-1, 1-Mange eller Mange-Mange.
Tuppel	En række i en tabel.
Komponent	Værdi med tilhørende attribut.
Metalag	Et interface imellem kernen og databasen, som anvendes til at koble databasen med grafdatastrukturen og oversætte tabel- og attributnavne for brugeren.
Kerne	Serverkomponenten, som behandler forespørgslen.
Forespørgsel	Formuleringen af det problem man ønsker at analysere.
Gruppe	En inddeling af felter, som gør det lettere for brugeren at

	finde de felter, som han/hun søger efter.
Felt	Et felt, som brugeren kan vælge at medtage i sin forespørgsel. Feltet er knyttet til en attribut i databasen vha. metalaget. Feltet er beskrevet ved et semantisk navn[3].
Resultatdatasæt	Resultatet af en forespørgsel.
Tuppel	En række i et Resultatdatasæt.
Kube	Relationsdatabase, som indeholder de data, som man ønsker at analysere.
Dimension	En attribut, som kubens data skal inddeles efter.
Measure	En beregning/aggregering med tilhørende attribut (de tal man ønsker at analysere).
View	Kubens aktuelle udseende.
Condition	En betingelse, som tuplerne i resultatdatasættet, skal opfylde.
Kubeoperation	En handling, som udføres på kuben.
Roll Up	Kubeoperation, som udføres på en dimension og skjuler de efterfølgende dimensioner i kuben.
Drill Down	Kubeoperation, som udføres på en dimension og viser den næste dimension i kuben.
Move Left	Kubeoperation, som udføres på en dimension og flytter dimensionen en plads til venstre kuben.
Move Right	Kubeoperation, som udføres på en dimension og flytter dimensionen en plads til højre i kuben.
Move Far Left	Kubeoperation, som udføres på en dimension og flytter dimensionen helt til venstre i kuben.
Gruppering	Inddeling af data indenfor en dimension, f.eks. er Mænd og Kvinder to grupperinger indenfor dimensionen Køn.
Aggregeringstype	De måder at aggregere data på, som kendes fra SQL.

[3] Letforståeligt sprog for brugeren.

2.6.2 Krav til systemet

Beskrivelse	ID
Skal understøtte Microsoft SQL Server og Oracle.	1
Skal have web brugergrænseflade.	2
Skal benytte grafteori til at modellere tabeller og relationer.	3
Skal benytte et metalag.	4
Skal benytte grafteoretiske metoder til at komme fra forespørgsel til resultatdatasæt.	5
Skal kunne håndtere udtryk, der er sammensat af grupper og felter.	6
Forespørgslen skal opbygges ved hjælp af ord, som brugeren kan relatere til.	7
Det skal være muligt at analysere data fra resultatdatasættet i en kube.	8
Systemadministratoren skal have mulighed for at definere grupperinger.	9
Brugeren skal kunne vælge dimensioner og measures.	10
Brugeren skal kunne angive conditions.	11
Kuben skal kunne indeholde op til 5 dimensioner og 1 measure.	12
Measures skal kunne bestå af følgende aggregeringstyper:	
Sum (SUM)	13
Antal (COUNT)	14
Maksimum (MAX)	15
Minimum (MIN)	16
Følgende kubeoperationer skal implementeres:	
Roll Up	17
Drill Down	18
Move Left	19
Move Right	20
Move Far Left	21

2.6.3 Supplerende krav

Der gælder følgende forudsætninger for aktørerne:

- Systemadministratoren skal have erfaring med databaser og kuber.
- Brugeren behøver ikke have at kendskab til SQL.

Jeg vil kort opsummere en række yderligere krav, som er relateret til diskussionen fra analyseafsnittet. Det er systemadministratorens rolle at sørge for at disse krav overholdes for det enkelte system. På baggrund af analysen har jeg besluttet, at:

- Grafen må **ikke** indeholde kredse.
- Grafen må gerne indeholde sløjfer.
- Grafen må **ikke** indeholde relationsfølgen: Mange-1-1-Mange.
- Dimensionstabeller må **ikke** indeholde dubletter.

Således bliver det ikke nødvendigt at specificere relationstyperne i systemet.

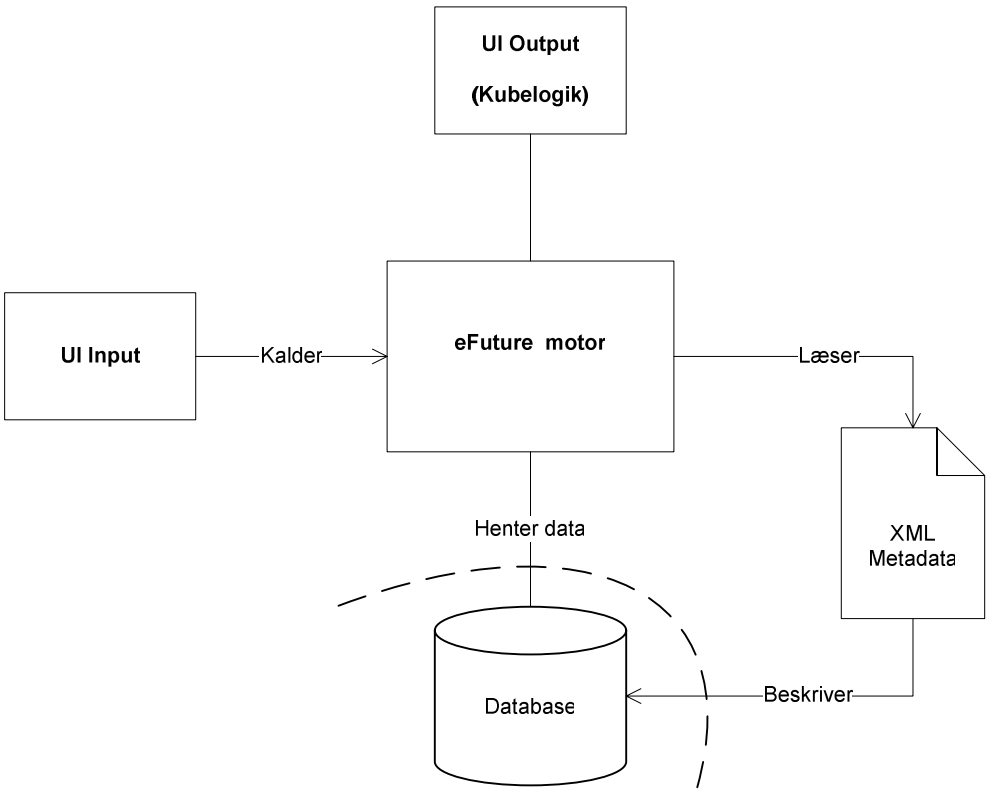
3 Design

3.1 Introduktion til designet

Med begreberne og kravene på plads er det nu tid til at påbegynde designet af systemet. Jeg vil starte med at skitsere den overordnede systemarkitektur. Derefter vil jeg gennemgå designet for de enkelte komponenter og opskrive klassediagrammer[4] eller matematiske modeller for komponenternes funktionalitet. Nogle områder af designet vil læne sig tæt op ad eller delvist overlappe implementeringen. Så vidt muligt vil jeg benytte UML notation til at beskrive designet. Denne form egner sig dog ikke til at beskrive al funktionalitet, og i disse tilfælde vil jeg opstille passende matematiske modeller. Jeg forestiller mig en systemarkitektur, som er vist på Figur 3-1. Systemet læner sig tæt op af begreberne fra kravspecifikationen og består af følgende fem hovedkomponenter:

Database	Relationsdatabase, som indeholder de data, der skal analyseres. *Er egentlig ikke en del af eFuture, men essentiel for at systemet kan fungere, så jeg vælger at medtage den her.*
XML Metadata	Metalaget bestående af et XML dokument, som beskriver databasen.
eFuture motor	Kernen i systemet, som modtager forespørgslen og genererer SQL udtrykket, som benyttes til at fylde resultatdatasættet.
UI Input	Den grafiske webgrænseflade, hvori brugeren definerer sin ad hoc forespørgsel.
UI Output	Den grafiske webgrænseflade, som indeholder kuben og giver brugeren mulighed for at analysere data.

[4] Datatyper i klassediagrammerne er valgt med henblik på implementering i VB.NET.



Figur 3-1 - Oversigt over eFuture systemets arkitektur

3.2 Database

Databasen er naturligvis væsentlig for systemets funktionalitet, men her er ikke noget designarbejde, der skal udføres. Systemet læner sig op af en eksisterende database, som i følge *Krav ID 1* enten skal være af typen Microsoft SQL Server eller Oracle.

3.3 XML Metadata

Jeg vælger at bruge et XML skema [W3S1] til at beskrive data i metalaget[5], da dette giver stor fleksibilitet. I dette afsnit vil jeg beskrive indholdet af metadata, som består af et hovedelement <FutureMeta>, der er inddelt i 5 sektioner: <MetaVersion>, <Options>, <Database>, <Relations>, <Structure>. En oversigt over det endelige design kan findes i form af en DTD[6] i appendiks 9, og et eksempel på et komplet XML metadata dokument kan findes i appendiks 12.

Det første element, <MetaVersion>, indeholder en streng, som anvendes til at specificere versionen af metadata.

[5] Denne funktionalitet svarer til *Krav ID 4*
[6] Document Type Definition – beskriver XML dokumentets struktur og gyldigt indhold.

3.3.1 Indstillinger - <Options> elementet

Denne sektion benyttes til at sætte nogle generelle indstillinger for eFuture, som har betydning for eFuture motorens funktionalitet.

<DateDiffStyle>

Denne indstilling angiver, hvordan en forskel mellem to datoer bliver evalueret, og kan antage følgende værdier:

Værdi	Betydning
0	Tidsdelen af datoen er ignoreret. Dette betyder, at forskellen mellem 2002-02-05 02:00 og 2002-02-06 23:00 og forskellen mellem 2002-02-05 23:00 og 2002-02-06 02:00 opfattes ens, nemlig 1 dag, selvom den første forskel er 45 timer, hvilket er tættere på 2 dage, imens den anden forskel kun er 3 timer.
1	Her bliver forskellen returneret som et decimaltal. F.eks. bliver forskellen mellem 2002-02-05 02:00 og 2002-02-06 23:00 beregnet til 1,875, imens forskellen mellem 2002-02-05 23:00 og 2002-02-06 02:00 bliver beregnet til 0,125.

<DecimalSeparatorFlag>

Indstilling, som bliver brugt til at specificere, hvilket decimalseparatortegn, som bliver anvendt af hhv. databasen og brugeren. Den siger altså om, der skal skrives ”1,5” eller ”1.5”. Følgende værdier kan antages:

Værdi	Betydning
0	Database separator: "." (punktum) Bruger separator: "." (punktum)
1	Database separator: "," (komma) Bruger separator: "." (punktum)
2	Database separator: "." (punktum) Bruger separator: "," (komma)
3	Database separator: "," (komma) Bruger separator: "," (komma)

<EmptyStringIncludesNullString>

Denne indstilling kan antage værdien "true" eller "false". Hvis indstillingen er ”true” betyder det, at en condition, som sammenligner om en streng er lig med den tomme streng, bliver betragtet som, om strengen er lig med den tomme streng eller om strengen er NULL.

3.3.2 Database - <Database> elementet

Denne sektion indeholder oplysninger om databasen, som metadata beskriver. Den består af 4 element, hvoraf det ene er frivilligt om man vil inkludere.

<Server>

Dette element beskriver navnet på database serveren, som enten kan være ”SQLServer” eller ”Oracle”. eFuture motoren har behov for at vide, hvilken type, der er tale om, da de to servere bruger forskellige syntakser indenfor SQL.

<ServerVersion>

I tilfælde af, at flere versioner af eksempelvis Microsoft SQL Server understøttes, har vi brug for at vide versionsnummeret. Dette felt er ikke nødvendigt, da den første version af eFuture kun understøtter en version af Microsoft SQL Server og Oracle.

<DisplayName>

Elementet indeholder navnet på databasen, som skal vises for brugeren. Systemadministratoren har også glæde af denne felt, når der skal arbejdes direkte med XML dokumentet.

<ConnectionString>

Her angives de oplysninger, som skal bruges for at oprette forbindelse til databasen. Login-navn og adgangskode behøver ikke specificeres her.

3.3.3 Relationer - <Relations> elementet

Denne sektion indeholder et antal <Relation> elementer, som beskriver relationerne imellem tabellerne. Det bør bemærkes, at man udelukkende specificerer den del af databasen, som det skal være muligt at analysere. Derfor er mængden af relationer i metadata er en delmængde af alle relationer i databasen.

<Relation>

Elementet består af tre underelementer: <Table1>, <Table2> og <JoinExpression>. <Table1> og <Table2> angiver navnene på de tabeller (eller views), som er relaterede. Det samme par af tabeller må kun optræde i ét <Relation> element.

<JoinExpression>

Dette element indeholder det udtryk, som benyttes til at joine de to tabeller, f.eks. "TabelA.AttributX = TabelB.AttributY". I tilfælde af, at to tabeller bliver joinet på mere end en attribut, håndteres det på følgende måde: "TabelA.AttributX1 = TabelB.AttributY1 AND TabelA.AttributX2 = TabelB.AttributY2". *Outer joins* kan også angives. På Microsoft SQL Server skrives et *left join* som "TabelA.AttributX *= TabelB.AttributY" og et *right join* som "TabelA.AttributX =* TabelB.AttributY". På Oracle databasen skrives *left join* udtrykket "TabelA.AttributX = TabelB.AttributY(+)" og *right join* udtrykket tilsvarende "TabelA.AttributX(+) = TabelB.AttributY".

3.3.4 Struktur - <Struture> elementet

I denne sektion defineres sammenhænge mellem databasens tabeller og attributter og forespørgslens grupper og felter med navne, som er letforståelige for brugeren[7]. <Structure> elementet indeholder en række <Group> elementer, som hver har en navneattribut og en hintattribut. Hintattributten er ikke nødvendig at inkludere, og den kan anvendes til hjælpetekster og tip om de enkelte grupper, som vises for brugeren. Hver <Group> element indeholder en række <Field> elementer, som har følgende opbygning:

```
<Field SemName="xxx">
    <Presentation>
        <Hint>aaa</Hint>
        <Format>bbb</Format>
         <MeasureType>1</MeasureType>
         <DimensionType>2</DimensionType>
    </Presentation>
    <DBInfo>
        <DBTable>TabelA</DBTable>
```

[7] Denne funktionalitet svarer til *Krav ID 7*

```
        <DBExpression type="string">TabelA.AttributX</DBExpression>
    </DBInfo>
</Field>
```

Det bør undgås, at der eksisterer to grupper af samme navn, men der må gerne være to felter med samme semantiske navn, blot de optræder i hver sin gruppe.

<Presentation>

Her angives de egenskaber for et felt, som henvender sig til præsentationen på brugergrænsefladen. Elementet indeholder 4 underelementer, hvoraf de første to ikke er nødvendige at inkludere. <Hint> elementet svarer til hintattributten for en gruppe, og <Format> kan bruges til at specificere den formatering, der knytter sig til feltet, f.eks. antal decimaler.

<MeasureType>

Denne indstilling benyttes til at angive, hvilke aggregeringstyper, der kan vælges i forbindelse med feltet. Følgende værdier kan antages:

Værdi	Betydning
0	Feltet kan ikke indgå som measure i en forespørgsel.
1	Der kan kun vælges aggregeringstypen, COUNT.
2	Alle fire aggregeringstyper[8] kan vælges for dette felt.

<DimensionType>

Indstilling, som bliver brugt til at specificere dimensionstypen for feltet. I den første version af eFuture har jeg valgt ikke at skelne imellem dimensionstyper, og derfor kan man kun vælge:

Værdi	Betydning
0	Feltet kan **ikke** inkluderes i en forespørgsel som dimension.
1	Feltet **kan** inkluderes i en forespørgsel som dimension.

[8] Denne funktionalitet svarer til *Krav ID 13-16*

<DBInfo>

Dette element rummer selve oversættelsen til databasen. Det består af et eller flere <DBTable> elementer, et <DBExpression> element og endeligt et <DBCondition> element, som ikke er nødvendigt. <DBExpression> elementet indeholder SQL udtrykket, som anvendes i databasen, samt en attribut, som angiver datatypen - "string", "real", "int" eller "date". Dette udtryk refererer oftest blot til en attribut i en tabel, f.eks. TabelA.AttributX, men kan også være et udtryk, som f.eks. "TabelA.TidISekunder/3600" (fordi data bliver gemt i sekunder, men brugeren er interesseret i antal timer). Et udtryk kan også inkludere to tabeller, men aggregeringsfunktioner er ikke tilladt. For hver tabel i <DBExpression> elementet skal der være et <DBTable> element, og alle tabeller i <DBTable> skal optræde i en relation i <Relations> sektionen. En database attribut bør altid præfikses med tabelnavnet, dog kan det udelades, såfremt attributnavnet er unikt.

Her er et eksempel på et <DBInfo> element, som indeholder to tabeller:

```
<DBInfo>
    <DBTable>TabelA</DBTable>
    <DBTable>TabelB</DBTable>
    <DBExpression type="real">TabelA.AttributX*TabelB.AttributY</DBExpression>
</DBInfo>
```

<DBCondition> elementer bliver brugt til at tilføje conditions, som skal inkluderes i forespørgslen. Lad mig demonstrere anvendelsen af dette felt med et eksempel: Antag, at der i Products-tabellen fra case-eksemplet var en *Status*-attribut, der sagde om et produkt var aktivt. Negativ status betyder, at produktet er udgået af produktsortimentet. Nu er vi kun interesseret i at inkludere de aktive produkter vores forespørgsler, og derfor tilføjes en <DBCondition> til felterne i Products-tabellen:

```
<DBInfo>
    <DBTable>Products</DBTable>
    <DBExpression type="string">Products.Name</DBExpression>
    <DBCondition>Products.Status &gt;= 0</DBCondition>
</DBInfo>
```

Betingelsen i <DBCondition> elementet skal skrives i den rette SQL syntaks svarende til databasen. Bemærk! '<' og '>' skal skrives som hhv. '<' og '>', da der er tale om et XML dokument.

3.3.5 Alternative løsninger og udvidelser

En anden mulighed er at implementere metalaget ved at gemme dets indhold i en almindelig database. Dette ville dog kræve, at der blev udviklet et administrationsmodul til at redigere

metadata, da man ikke uden videre kan ændre indholdet af databasen. Fordelen ved at have metadata i XML format er, at det kan skrives i hånden og er meget let at læse. Desuden kan det XML metadata let præsenteres for brugeren vha. XSLT[9]. En anden god ting med XML formatet er, at det er betydeligt lettere at ændre skemaet for et XML dokument, end det er at ændre skemaet for en relationsdatabase.

Oftest vil det være tilfældet, at en del af databasen indeholder informationer, som ikke er relevante at inkludere i undersøgelserne. Der kan også være tale om følsomme data, som man ikke ønsker at give adgang til. Som det blev nævnt tidligere i analyseafsnittet, kan man specificere afgangsniveauer for bruger og felter i metadata. Denne mulighed har ikke valgt at inkludere i denne første version af metadata, men det kunne gøres ved at tilføje to elementer: <User> element tilføjes til <FutureMeta> og beskriver systemets brugere og deres adgangsniveau, og <AccesLevel> element tilføjes til hver <Field> og angiver det adgangsniveau, der kræves for det enkelte felt. Adgangsniveauer kunne også angives på gruppenivaeu.

3.4 eFuture motor

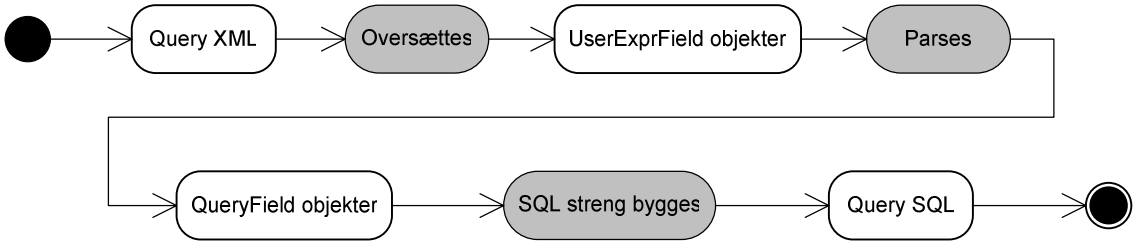
Som nævnt i introduktionen til dette afsnit er eFuture motoren kernen i systemet. Det er her logikken, der fører os fra forespørgsel til resultatdatasæt, skal implementeres. De øvrige komponenter kommunikerer udelukkende igennem eFuture motoren, så der er også behov for en del hjælpefunktioner, som eksempelvis skal anvendes til at hente indholdet, der skal vises på grænsefladen. Jeg vil forsøge at designe eFuture motoren med henblik på at gøre den så fleksibel som mulig, således at andre systemer nemt kan udnytte dens funktionalitet. Derfor vælger jeg også at anvende en løs kobling imellem kube og dataudtrækningen, som jeg beskrev det i afsnit 2.5.2.

3.4.1 Opbygning af SQL streng

Hovedfunktionaliteten i eFuture motoren er, at der modtages en forespørgsel, som er blevet defineret vha. UI input komponenten, og der returneres et SQL udtryk, der benyttes til at fylde resultatdatasættet. Jeg forestiller mig, at processen overordnet set indeholder trinene, som er vist på Figur 3-2. Først modtages forespørgslen i XML-format (se definitionen i afsnit 3.5) og denne

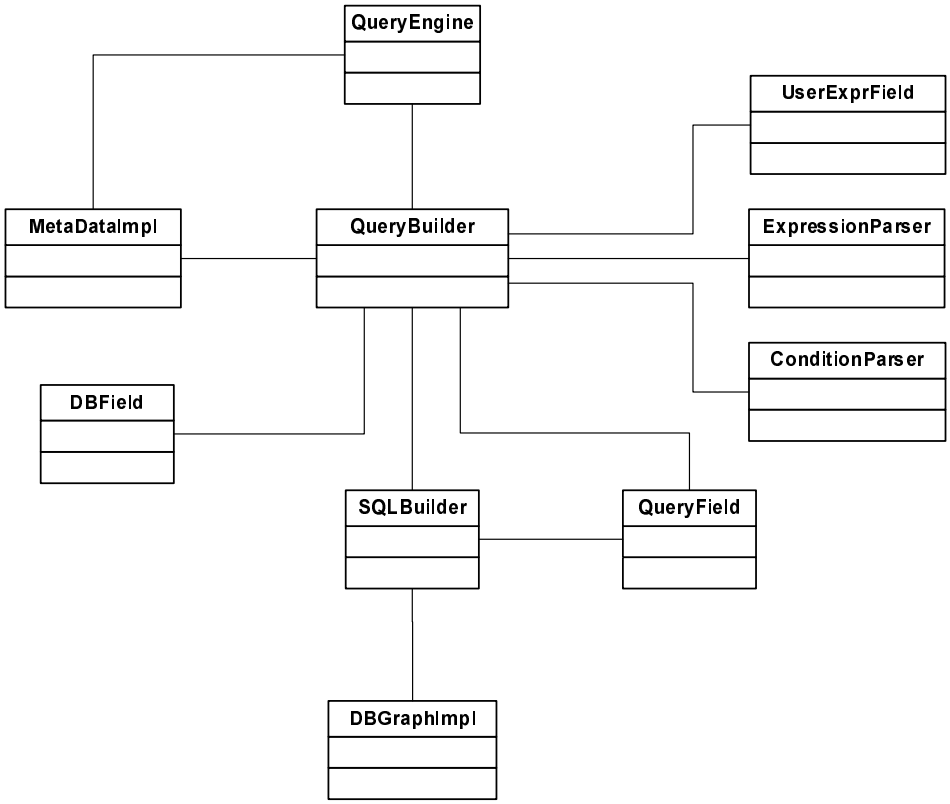
[9] XSLT er et sprog til at transformere XML dokumenter til andre XML dokumenter.

oversættes til en Collection[10] af UserExprField objekter, således at vi har et UserExprField objekt for hvert element i forespørgslen. UserExprField objekterne indeholder felternes navne, som de vises for brugeren, og disse skal derfor hver især parses og oversættes til SQL udtryk, som databasen kan forstå. Når dette er sket har vi en Collection af QueryField objekter, som benyttes til at bygge den endelige SQL streng.



Figur 3-2 - Processen fra forespørgsel til SQL

På baggrund af ovenstående beskrivelse vurderer jeg, at vi har behov for følgende klasser:



Figur 3-3 - Klassediagram for eFuture kernen

[10] VB-datatype – et Collection objekt er et ordnet sæt af elemter, der ikke nødvendigvis er af samme type.

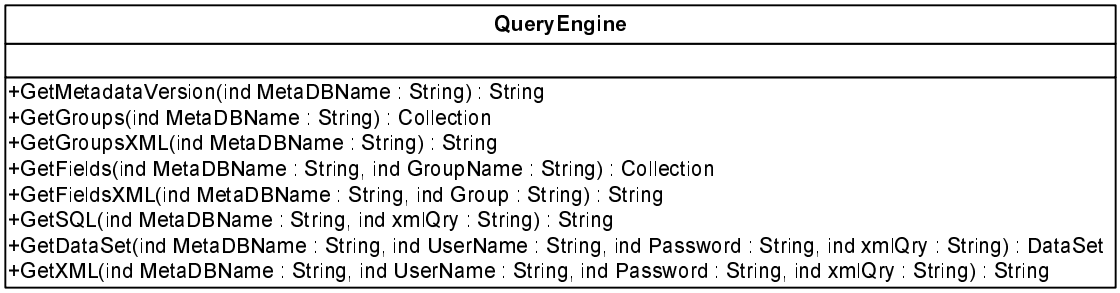
Klasse	Beskrivelse
QueryEngine	Interface til eFuture kernen, som opretter de øvrige objekter, når den bliver kaldt fra UI komponterne.
QueryBuilder	Dette er hovedklassen i eFuture kernen, som indeholder metoden, der konverterer forespørgslen i XML-format til en SQL streng baseret på indholdet af metadata filen.
ExpressionParser	Benyttes til at parse og oversætte UserExprFields til gyldige SQL udtryk. (herunder oversættelse fra semantiske navne til database felter etc.)
ConditionParser	Benyttes til at parse og oversætte conditions til gyldige SQL udtryk.
SQLBuilder	Denne klasse bygger den endelige SQL streng ud fra en Collection af QueryFields.
MetaDataImpl	Benyttes til at læse metadata.
DBGraphImpl	Denne klasse genererer JOIN udtrykket, som skal tilføjes til den endelige SQL streng, ud fra et givent sæt af tabeller. Her implementeres korteste vej algoritmen, som bestemmer hvordan tabellerne skal joines.
UserExprField	En instans af denne klasse repræsenterer et felt, som er blevet valgt af brugeren. Informationerne i denne klasse er set fra brugerens synspunkt.
QueryField	En instans af denne klasse repræsenterer et felt, som er blevet valgt af brugeren. Informationerne i denne klasse er set fra databasens synspunkt.
DBField	Benyttes til at holde database informationer for et felt.

På Figur 3-3 er vist det overordnede klassediagram, som viser sammenhæng mellem klasserne. Jeg vil nu gå i detaljer med hver enkelt klasse og beskrive de attributter og metoder, som jeg mener de skal indeholde. For hver klasse opskriver jeg et klassediagram på UML form, der viser attributter med datatyper, og metoder med indgående parametre og returparametre.

QueryEngine

Dette er som nævnt interfacet til eFuture kernen, som indeholder den funktionalitet, som tilbydes til eksterne komponenter. Som vist i klassediagrammet på Figur 3-4 indeholder klassen 8 metoder,

hvoraf de første fem benyttes til at hente data fra metadata, og de sidste tre er relateret til selve ad hoc forespørgslen.

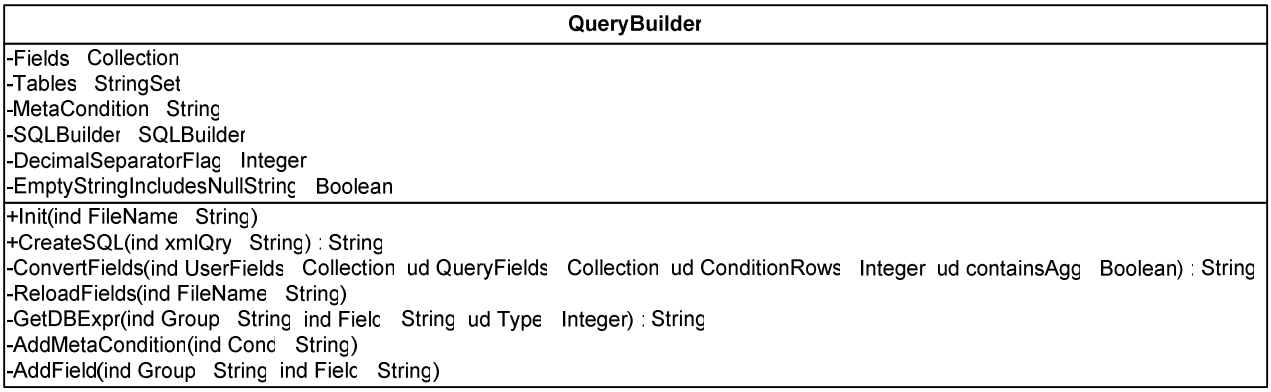


Figur 3-4 - Klassediagram for QueryEngine

Metode	Beskrivelse
GetMetadataVersion	Returnerer versionen af metadata.
GetGroups	Returnerer navn og hint for alle grupper i metadata.
GetGroupsXML	Returnerer navn og hint for alle grupper i XML format.
GetFields	Returnerer navn, hint og datatype for alle felter i en given gruppe.
GetFieldsXML	Returnerer navn, hint og datatype for alle felter i en given gruppe i XML format.
GetSQL	Konverterer en forespørgsel i XML format til en SQL streng, som returneres.
GetDataSet	Returnerer resultatdatasættet.
GetXML	Returnerer resultatet af en forespørgsel som XML.

QueryBuilder

Denne klasse er opbygget med henblik på, at den skal kunne bevare sin tilstand. Ved instantiering af klassen kaldes Init-metoden, hvorved alle felter i metadata indlæses og indstillinger for databasen gemmes. Disse data gemmes indtil objektet destrueres eller intialiseres på ny, således at metadata ikke behøver at blive indlæst hver gang, der modtages en XML forespørgsel. Klassen indeholder metoden til at konvertere UserExprFields til QueryFields, samt metoden, der omsætter forespørgslen i XML-format til en SQL streng.



Figur 3-5 - Klassediagram for QueryBuilder

Attribut	Beskrivelse
Fields	En Collection af DBFields, som bliver dannet ved initialiseringen af QueryBuilderen
Tables	Indeholder de tabeller, som indgår i en forespørgsel. Bliver opdateret efterhånden som ExpressionParseren behandler for de valgte felter.
MetaCondition	Streng, som bliver genereret for hver forespørgsel og den indeholder de conditions, som er specificeret i metadata for de valgte felter. Bliver opdateret efterhånden som ExpressionParseren behandler for de valgte felter.
SQLBuilder	Bliver brugt til at generere SQL strengen.
DecimalSeparatorFlag	Holder værdien for separatorvalget, der er givet i metadata.
EmptyStringIncludesNullString	Holder værdien, som er givet i metadata.

Metode	Beskrivelse
Init	Denne funktion kaldes, når objektet instantieres. Den læser metadata og gemmer oplysninger om felter og databaseindstillinger.
CreateSQL	Hovedfunktionen, som konverterer XML forespørgslen til en SQL streng. Dette gøres ved først at oversætte felterne i forespørgslen til UserExprFields. Efterfølgende konverteres felterne med ConvertFields og endeligt kaldes SQLBuilderen.

ConvertFields	Konverterer de valgte felter fra UserExprField format til QueryField format.
ReloadFields	Benyttes til at indlæse alle felter fra metadata i Fields.
GetDBExpr	Denne funktion kaldes hver gang ExpressionParseren parser et UserExprField og returnerer database udtryk med tilhørende datatype. Desuden opdateres Tables og MetaCondtion.
AddMetaCondition	Benyttes af GetDBExpr metoden til at opdatere MetaCondition strengen.
AddField	Benyttes af ReloadFields til at tilføje et felt til Fields.

ExpressionParser

Denne klasse benyttes til at parse et udtryk og oversætte til gyldige SQL udtryk[11]. Jeg vælger at anvende top-down parsning og vil derfor opskrive en grammatik for parseren. Et udtryk kan enten være aggregeret eller ikke-aggregeret. For hver af de to typer ser den overordnede grammatik således ud:

E = E'-'E | E'+'E | M
M = M'*'M | M'/'M | T | '('E')'

Hvorvidt udtrykket er aggregeret eller ej afhænger af om T er aggregeret eller ej. Et ikke-aggregeret T er enten en konstant eller et felt:

T = Const | Field
Const = DATE | STRING | *Number*
DATE = *#YYYY-MM-DD#*
STRING = '*TEXT*'
FIELD = *GroupName.FieldName*

Et aggregeret T er enten en konstant, count(*) eller en brug af en aggregeringsfunktion på et ikke-aggregeret udtryk:

T = Const | Count(*) | Sum(E) | Count(E) | Avg(E) | Max(E) | Min(E)
hvor E er ikke aggregeret.

[11] Denne funktionalitet svarer til *Krav ID 6*

Der er imidlertid et problem med E og M. Vi kan nemlig ikke ud fra næste tegn afgøre, hvilken af de første to produktioner, der skal vælges, idet de har det samme præfiks. Derfor må vi foretage en venstre-faktorisering[12], og grammatikken kommer til at således ud:

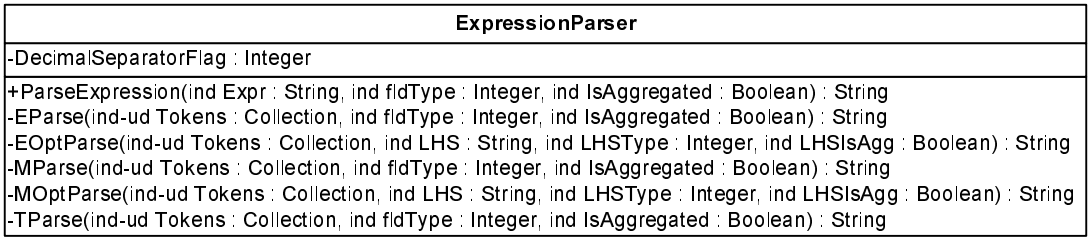
E = M EOpt
EOpt = '-' M EOpt | '+' M EOpt | Empty

M = T MOpt | '('E')' MOpt
MOpt = '*' M | '/' M

Desuden er der en række begrænsninger i forhold til T's type, f.eks. giver det ikke mening at multiplicere to strenge eller at trække en integer fra en streng.
Følgende kombinationer er tilladt:
* : (real, real), (int, int), (real, int), (int, real)
/ : (real, real), (int, int), (real, int), (int, real)
+ : (real, real), (int, int), (real, int), (int, real), (string, string)
- : (real, real), (int, int), (real, int), (int, real), (date, date)

+ af strenge er strengkonkatenering
- af datoer returnerer antal dage mellem de to datoer, hvorved resultatet er af type int eller real.

Ovenstående regler implementeres i en klasse, som er vist på Figur 3-6.



Figur 3-6 - Klassediagram for ExpressionParser

[12] Metoden, som jeg anvender, er beskrevet i standard litteratur om sprog og parsere – bl.a. [Hop01]

Attribut	Beskrivelse
DecimalSeparatorFlag	Holder værdien for separatorvalget, der er givet i metadata.

Metode	Beskrivelse
ParseExpression	Hovedfunktionen, som tager et udtryk fra en UserExprField og returnerer det tilsvarende SQL udtryk. Udtrykket scannes først og oversættes til en Collection af Tokens, som behandles med EParse, MParse og TParse metoderne.
EParse	Implementeringen af førnævnte grammatik.
EOptParse	Implementeringen af førnævnte grammatik.
MParse	Implementeringen af førnævnte grammatik.
MOptParse	Implementeringen af førnævnte grammatik.
TParse	Implementeringen af førnævnte grammatik.

ConditionParser

Denne klasse benyttes til at parse conditions[13] og oversætte til gyldige SQL-betingelser. Dette gøres på samme måde som for ExpressionParserens vedkommende, nemlig ved brug af top-down parsning. Inden jeg opskriver grammatikken for parseren, vil nævne de tre typer conditions, som systemet skal kunne håndtere:

1. Null / not null
2. Sammenligning med en konstant: = , < , > , <> , <= , >= , like (strenge), notlike (strenge).
3. I en liste af konstanter.

Det er muligt at benytte wildcards i strengsammenligninger. Microsoft SQL Server og Oracle benytter wildcardsymbolet '%'. Eksempelvis kan man angive en condition, som vælger tupler for hvilke det gælder, at en streng begynder med 'ab' ved at skrive "like 'ab%'". Vil man sammenligne med en liste af konstanter skrives det på følgende måde: "in ('ab'; 'ac'; 'ad')" eller "in (2; 8; 14), hvor sidstnævnte vælger de tupler, hvor udtrykket er lig med 2, 8 eller 14. Conditions kan sammensættes med '&' (AND) og '|' (OR) operatorerne samt parenteser. Præcedensfølgen bestemmes af databasen, og for både SQL Server og Oracle gælder der, at '&' har den højeste præcedens. Ved sammenligning af strenge angives følsomheden overfor store og små bogstaver

[13] Denne funktionalitet svarer til *Krav ID 17*

(*case sensitivitet*) på databaseniveau. Som standard er SQL Server ikke *case sensitiv*, og Oracle er *case sensitiv*.

Grammatiken for en condition ser således ud:
COND = (COND) | COND '&' COND | COND '|' COND | TERM
TERM = Null | NotNull | QUAL Const | Const | 'in(' CONSTLIST ')'
QUAL = ‘=’ | ‘<’ | ‘>’ | ‘<>’ | ‘<=’ | ‘>=’ | ‘like’ | 'notlike'

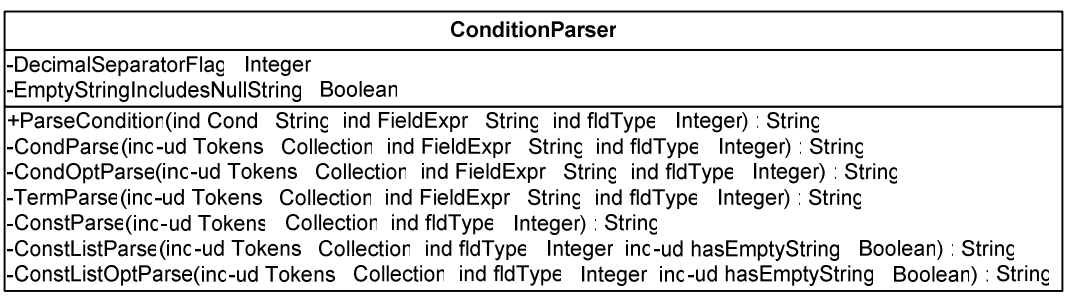
Her er Const en konstant defineret på samme måde som for ExpressionParseren, og CONSTLIST en liste af konstanter efter følgende grammatik:
CONSTLIST = Const | Const ';' CONSTLIST

Igen foretager jeg en ventre-faktorisering og COND-reglen er omskrives til:
COND = (COND) CONDOpt | TERM CONDOpt
CONDOpt = '&' COND CONDOpt | '|' COND CONDOpt | Empty

På tilsvarende måde omskriver vi CONSTLIST:
CONSTLIST = Const CONSTLISTOpt
CONSTLISTOpt = ';' Const CONSTLISTOpt | Empty

Ud over at følge grammatikken skal betingelsen desuden være gyldig for typen af udtrykket. De eneste tilladte sammenligninger på tværs af typer er sammenligninger af real med int, f.eks. kan skrives ”<= 12,5 | > 25”

Ovenstående regler implementeres i en klasse, som er vist på Figur 3-7.



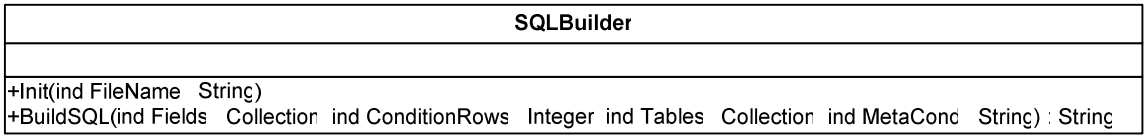
Figur 3-7 - Klassediagram for ConditionParser

Attribut	Beskrivelse
DecimalSeparatorFlag	Holder værdien for separatorvalget, der er givet i metadata.
EmptyStringIncludesNullString	Holder værdien, som er givet i metadata.

Metode	Beskrivelse
ParseCondition	Hovedfunktionen, som tager et udtryk fra en UserExprField og returnerer det tilsvarende SQL udtryk. Udtrykket scannes først og oversættes til en Collection af Tokens, som behandles med CondParse, TermParse, ConstParse og ConstListParse metoderne.
CondParse	Implementeringen af førnævnte grammatik.
CondOptParse	Implementeringen af førnævnte grammatik.
TermParse	Implementeringen af førnævnte grammatik.
ConstParse	Implementeringen af førnævnte grammatik.
ConstListParse	Implementeringen af førnævnte grammatik.
ConstListOptParse	Implementeringen af førnævnte grammatik.

SQLBuilder

Denne klasse benyttes af QueryBuilderen til at konstruere SQL sætningen. Indledningsvist skal grafen valideres, så vi sikrer os, at alle tabeller optræder i grafen, og at grafen er sammenhængende. Hovedfunktionen konstruerer SELECT, FROM, WHERE, GROUP BY og ORDER BY udtrykkene, og sætter dem sammen med JOIN udtrykket til den endelige SQL streng. Selve grafen er repræsenteret i DBGraphImpl, og det er også i denne klasse metoden, som beregner JOIN udtrykket, befinder sig.



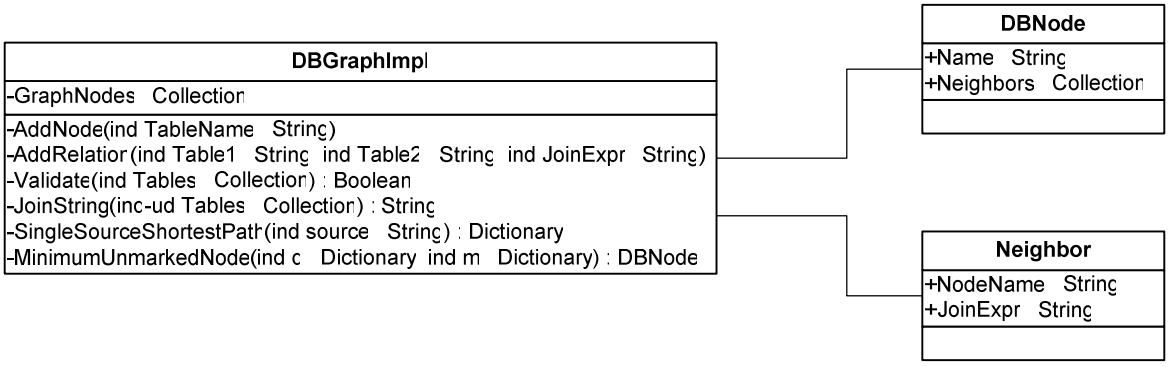
Figur 3-8 -- Klassediagram for SQLBuilder

Metode	Beskrivelse
Init	Denne funktion kaldes, når objektet instantieres. Den læser metadata og opbygger grafen ud fra informationen om relationerne.
BuildSQL	Hovedfunktionen, som kaldes med en Collection af QueryFields og returnerer den endelige SQL streng. De øvrige indgående parametre er antallet af Conditions, en Collection med tabeller som indgår i forespørgslen, og MetaCondition strengen, som QueryBuilderen har lavet.

3.4.2 Generering af JOIN udtrykket

DBGraphImpl

Til at repræsentere grafen laver jeg to klasser, DBNode og Neighbor. Grafen udgøres af en Collection af DBNodes, hvor hver DBNode er et punkt i grafen med tilhørende navneattribut. Desuden indeholder hvert DBNode objekt en Collection, Neighbors, af samme længde som antallet af nabopunkter. Neighbor klassen består af to attributter: NodeName, som beskriver navnet på nabopunktet, og JoinExpr, som er JOIN udtrykket, der udgør kanten imellem de to punkter. DBGraph-klassen indeholder metoder til at opbygge grafen, samt Validate-funktionen, som SQLBuilderen bruger til at validere grafen. I hovedfunktionen JoinString genereres JOIN udtrykket ved brug af Dijkstra korteste vej algoritmen[14]. JOIN udtrykket er sammensat af de enkelte JOIN udtryk, som svarer til kanterne på den korteste vej imellem alle punkter, der indgår i forespørgslen.



Figur 3-9 - Klassediagram for DBGraphImpl

[14] Dijkstras algoritme er beskrevet i appendiks 0

Attribut	Beskrivelse
GraphNodes	Grafen repræsenteret ved en Collection af DBNodes.

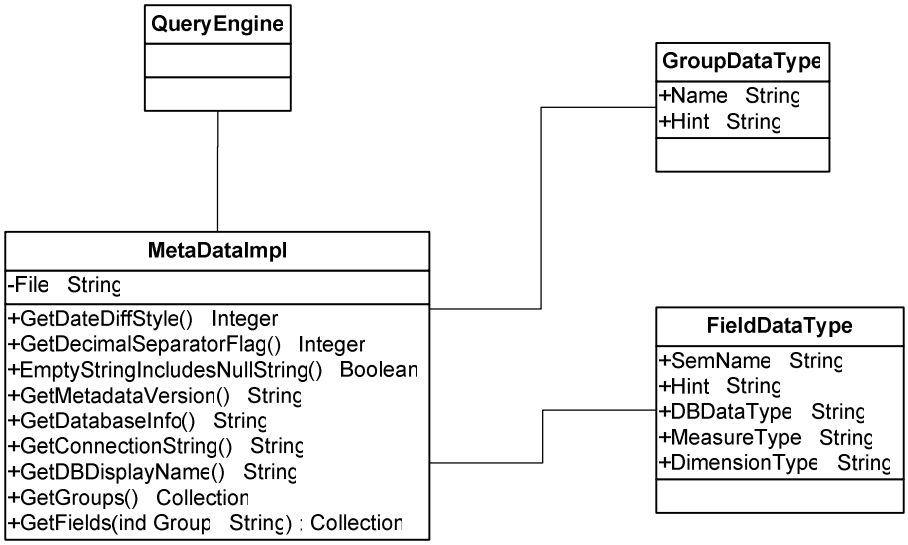
Metode	Beskrivelse
AddNode	Tilføjer et punkt til grafen. I tilfælde af, at der allerede eksisterer et punkt med det givne navn, ændres ingenting.
AddRelation	Tilføjer en kant til grafen. De to punkter tilføjes til hinandens naboliste.
Validate	Validerer at alle tabeller er i grafen, og at grafen er sammenhængende.
JoinString	Beregner det endelige JOIN udtryk udfra en Collection, som indeholder navne på tabellerne i forespørgslen. Denne Collection opdateres med de tabeller, som indgår i det endelige JOIN udtryk.
SingleSourceShortestPath	Finder de korteste veje fra startpunktet til alle øvrige punkter i grafen ved brug af Dikstras algoritme. Returnerer et Dictionary[15] med de foregående punkter i den korteste vej til hvert punkt.
MinimumUnmarkedNode	Hjælpefunktion til SingleSourceShortestPath, som returnerer det punkt med den korteste vej til startpunktet, som endnu ikke er blevet markeret.

3.4.3 Håndtering af metadata

MetaDataImpl

eFuture kernen skal kunne læse informationerne fra metadata. Til dette formål har vi behov for en klasse, der indeholder metoder til at hente de enkelte data, samt to datatype klasser, der beskriver hhv. grupper og felter. Klassen kalder jeg for MetaDataImpl, og dens klassediagram kan ses på Figur 3-10.

[15] VB-datatype, som ligner en Collection med udvidet funktionalitet.



Figur 3-10 - Klassediagram for MetaDataImpl

Attribut	Beskrivelse
File	Beskriver navnet og stien på metadatafilen.

Metode	Beskrivelse
GetGroups	Returnerer en Collection af GroupDataType objekter.
GetFields	Returnerer en Collection af FieldDataType objekter for en given gruppe.

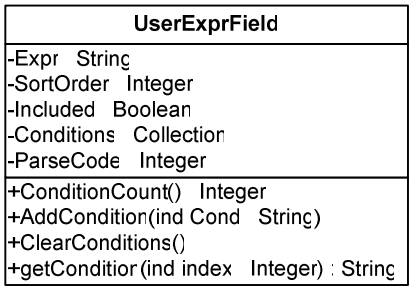
De øvrige metoder og attributter føler jeg ikke, at der er behov for at knytte yderligere kommentarer til, da de læner sig tæt op af metadatabeskrivelsen i afsnit 3.3.2.

3.4.4 Datatyper og Strukturer

I dette afsnit vil jeg kort beskrive tre væsentlige klasser, som rummer de datastrukturer, der benyttes til opbevare informationer om forespørgslens felter. De tre klasser blev kort beskrevet i begyndelsen af afsnittet, og jeg vil præsentere deres klassediagrammer og en kort beskrivelse af de metoder, som jeg mener de skal indeholde.

UserExprField

En instans af denne klasse repræsenterer et felt, som er blevet valgt af brugeren. Når XML forespørgslen modtages af eFuture motoren, bliver den oversat til UserExprFields, således at der for hvert felt i forespørgslen oprettes et UserExprField objekt. Jeg vil understrege at informationerne i denne klasse er set fra brugerens synspunkt, og Expr og Condition attributterne indeholder derfor de semantiske feltnavne.



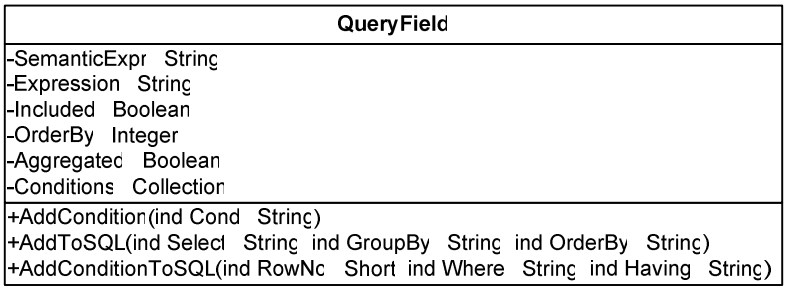
Figur 3-11 - Klassediagram for UserExprField

Attribut	Beskrivelse
Expr	Udtrykket for feltet, som brugeren har valgt.
SortOrder	Her angives hvilken sorteringsrækkefølge, der skal gælde for det pågældende felt. Følgende værdier kan antages: 0 – ingen 1 – stigende 2 – faldende
Included	Denne værdi fortæller om feltet skal inkluderes i resultatdatasættet eller ej. Det kan være at feltet kun optræder i en condition, og Included værdien sættes derfor til *False*.
Conditions	En Collection af strenge, som indeholder condition udtrykkene.
ParseCode	Denne værdi bruges til at returnere meddelelser fra parsningen.

Metode	Beskrivelse
ConditionCount	Returnerer antallet af conditions for det pågældende felt.
AddCondition	Tilføjer en condition streng til feltet.
ClearConditions	Fjerner alle conditions for feltet.
getConditions	Returnerer condition strengen for det givne indeks.

QueryField

En instans af denne klasse repræsenterer et felt, som er blevet valgt af brugeren. Denne gang er indholdet dog set fra databasen synspunkt. Hvert objekt indeholder informationer fra UserExprField klassen plus de oversatte SQL udtryk. Desuden indeholder klassen et par metoder, som benyttes til at konstruere den endelige SQL streng.



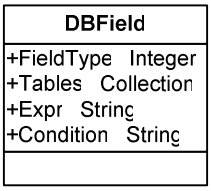
Figur 3-12 - Klassediagram for Queryfield

Attribut	Beskrivelse
SemanticExpr	Det semantiske udtryk for feltet, som brugeren har valgt. Dvs. samme værdi, som Expr-attributten i UserExprField.
Expression	SQL udtrykket, som svarer til det semantiske navn for feltet.
Included	Denne værdi fortæller om feltet skal inkluderes i resultatdatasættet eller ej. Det kan være at feltet kun optræder i en condition, og Included værdien sættes derfor til *False*.
OrderBy	Her angives hvilken sorteringsrækkefølge, der skal gælde for det pågældende felt. Følgende værdier kan antages: 0 – ingen 1 – stigende 2 – faldende
Aggregated	Denne værdi fortæller om udtrykket indeholder en aggregeret værdi eller ej.
Conditions	En Collection af strenge, som indeholder et condition udtrykkene. Her er der igen tale om de oversatte SQL udtryk.

Metode	Beskrivelse
AddCondition	Tilføjer en condition streng til feltet.
ClearConditions	Fjerner alle conditions for feltet.
getConditions	Returnerer condition strengen for det givne indeks.

DBField

Denne klasse benyttes til at gemme databaseinformationer for felterne i metadata. Ved initialiseringen af QueryBuilderen bliver felterne i metadata parset, og der oprettes et DBField objekt for hvert felt.



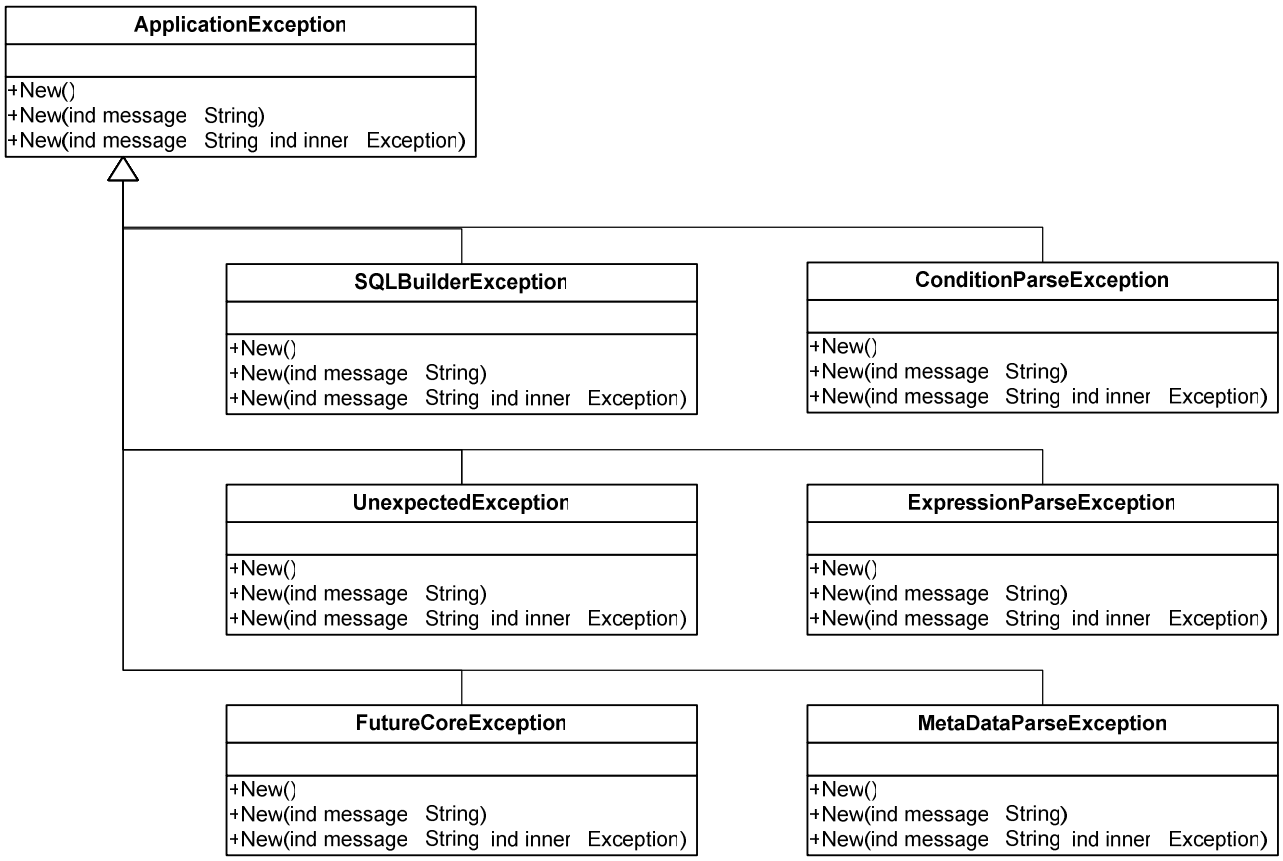
Figur 3-13 - Klassediagram for DBField

Attribut	Beskrivelse
FieldType	Angiver feltets datatype. Følgende værdier kan antages: 11 = Real 12 = Integer 13 = String 14 = Date
Tables	Collection af strenge, som indeholder navnene på de tabeller, der indgår i feltets databaseudtryk.
Expr	Databaseudtrykket for feltet.
Condition	Databaseudtrykket for feltets metacondition.

3.4.5 Fejlhåndtering

Jeg vil nu definere et antal Exception-klasser, som skal det benyttes til den strukturerede fejlhåndtering. På
Figur 3-14 er vist en oversigt over klasserne. Jeg definerer seks klasser, som alle nedarver fra basisklassen, ApplicationException. De forskellige Exception-klasser anvendes alt efter hvor i eFuture motoren, at en fejl er opstået. Hver Exception-klasse indeholder tre metoder. Den første metode initialiserer et nyt objekt. Den anden metode initialiserer et nyt objekt med en

fejlmeddelelse. Den sidste metode initialiserer et nyt objekt med en fejlmeddelelse og en reference til den indre exception, som har forårsaget den.

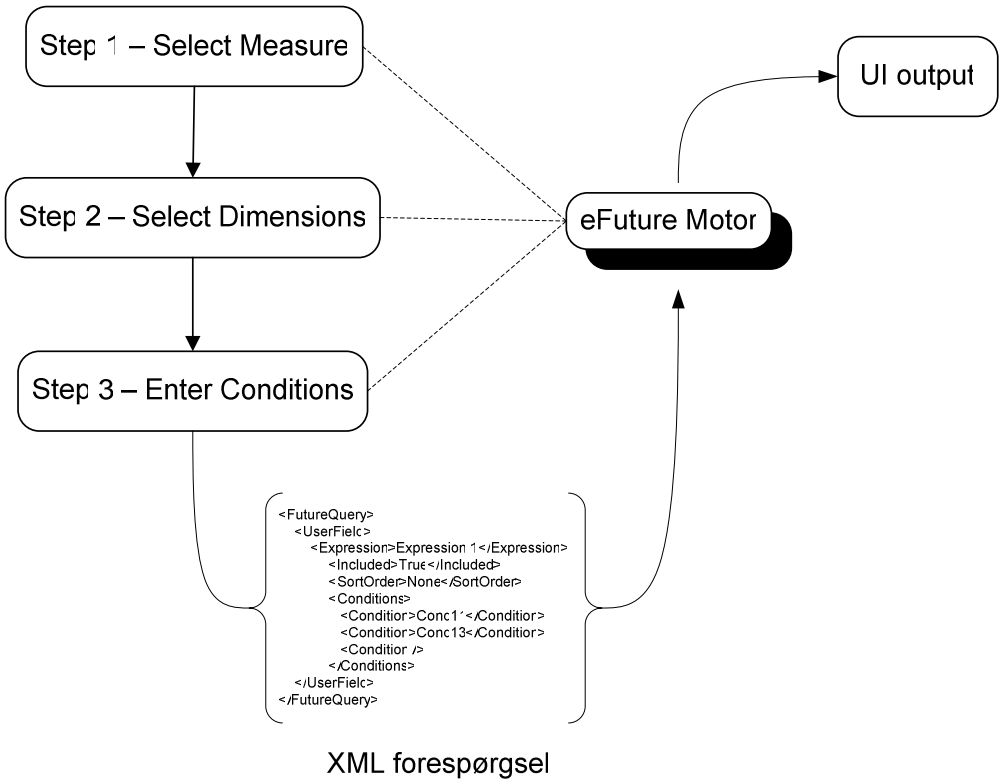


Figur 3-14 - Klassediagram for Exceptionklasserne

Klasse	Beskrivelse
ApplicationException	Basisklasse, som de øvrige exceptionklasser nedarver fra.
FutureCoreException	Benyttes til at sende fejlmeddelelser fra eFuture motoren til brugeren.
MetaDataParseException	Beskriver fejl, som er opstået under parsningen af metadata.
ExpressionParseException	Beskriver fejl, som er opstået i Expressionparseren.
ConditionParseException	Beskriver fejl, som er opstået i Conditionparseren.
SQLBuilderException	Beskriver fejl, som er opstået i SQLBuilderen.
UnexpectedException	Benyttes til øvrige fejlmeddelelser, som opstår uventet.

3.5 UI input

Det er nu tid til at designe systemets grafiske brugergrænseflade, som ifølge *Krav ID 2* skal være web-baseret. Første skridt er finde ud af, hvordan brugeren skal kunne definere sin forespørgsel. Jeg vil lægge særligt vægt på, at grænsefladen skal være overskuelig og nærmest selvforklarende, således at en bruger, der ikke tidligere har arbejdet med systemet, kan gå lige i gang med sine ad hoc forespørgsler. Mine erfaringer med andre Ad hoc værktøjer er, at det ofte kræves, at brugeren skal læse en tyk manual inden, at arbejdet kan påbegyndes. Desuden vil jeg pointere, at UI input grænsefladen designes med henblik på, at resultatet af Ad hoc forespørgslen skal præsenteres i en kube. Efter at have overvejet nogle forskellige muligheder har jeg besluttet, at UI input processen skal bestå af tre trin, som er vist på Figur 3-15. I det første trin vælger brugeren en measure - ”Hvad er det jeg vil måle på?”. I andet trin vælges mellem 1 og 5 dimensioner – ”Hvordan skal data inddeles?”. I tredje trin angiver brugeren conditions – ”Hvilke betingelser skal data opfylde?”. Herefter opsamles informationer fra de tre trin og oversættes til en XML tekststreng, som eFuture motoren bruger til at generere det endelige SQL udtryk. De stiplede linjer på figuren symboliserer, at der hentes informationer fra metadata, som præsenteres på grænsefladen. Hvad de to pile markeret med A og B symboliserer, vil jeg komme ind på i afsnit 3.6.



Figur 3-15 - UI processen

XML tekststrengen[16], som udgør forespørgslen, består af rodelementet <FutureQuery> samt et antal <UserField> elementer, som hver indeholder følgende elementer:

Element	Beskrivelse
Expression	Udtrykket for feltet, som brugeren har valgt.
SortOrder	Her angives sorteringsrækkefølgen for det pågældende felt.
Included	Denne værdi fortæller om feltet skal inkluderes i resultatdatasættet eller ej. Det kan være at feltet kun optræder i en condition, og Included værdien sættes derfor til *False*.
Conditions	Indeholder nul eller flere <condition> elementer.
Condition	Udtrykket for en betingelse, som brugeren har angivet.

[16] DTD skemaet er vedlagt i Appendiks 11.

Her et eksempel på en XML forespørgslen:

```
<FutureQuery>
    <UserField>
        <Expression>Expression 1</Expression>
        <Included>True</Included>
        <SortOrder>None</SortOrder>
        <Conditions>
            <Condition>Cond11</Condition>
        </Conditions>
    </UserField>
    <UserField>
        <Expression>Expression 2</Expression>
        <Included>True</Included>
        <SortOrder>Asc</SortOrder>
        <Conditions>
            <Condition>Cond22</Condition>
        </Conditions>
    </UserField>
</FutureQuery>
```

3.6 UI output (Kube)

Efter forespørgslen er blevet defineret i UI input, og eFuture motoren har konstueret det endelige SQL udtryk, skal resultatdatasættet præsenteres for brugeren. Denne præsentation skal ske i form af en kube, som det er angivet i *Krav ID 8*. Kubens design antager den form, som blev vist i case-eksemplet i afsnit 2.4 med tilføjelse af knapper til at udføre de enkelte kubeoperationer[17]. Hver gang brugeren trykker på en af disse knapper sendes den nye forespørgsel til eFuture motoren, hvilket er illustreret med pil A på Figur 3-15. Pil B symboliserer, at brugeren vælger at definere en helt ny ad hoc forespørgsel og bliver ledt tilbage til det første trin i UI input. Den første opgave er at formatere resultatdatasættet, så det har form af en kube. Formateringsprocessen består af 4 trin:

[17] Denne funktionalitet svarer til *Krav ID 17-21*.

1. Resultatdatasættet indlæses.
2. Rækkedubletterne fjernes for de enkelte grupperinger. Med en rækkedublet forstås, at der eksisterer to ens komponenter i to efterfølgende tupler.
3. Der indsættes en summeringsrække for hver gruppering.
4. Summeringer beregnes og indsættes for hver gruppering.

1

Geografisk område	Produktkategori	Sum af ordretotaler
Sjælland	Kød	kr 2.485,26
Sjælland	Mælkeprodukter	kr 7.895,12
Sjælland	Drikkevarer	kr 4.346,00
Sjælland	Slik	kr 16.255,78
Jylland	Kød	kr 3.321,34
Jylland	Mælkeprodukter	kr 11.566,16
Jylland	Drikkevarer	kr 6.547,98
Jylland	Slik	kr 22.456,12
Fyn	Kød	kr 1.786,45
Fyn	Mælkeprodukter	kr 6.588,81
Fyn	Drikkevarer	kr 3.845,32
Fyn	Slik	kr 12.875,64
Øvrige	Kød	kr 1.164,94
Øvrige	Mælkeprodukter	kr 4.687,22
Øvrige	Drikkevarer	kr 2.487,91
Øvrige	Slik	kr 9.864,73

2

Geografisk område	Produktkategori	Sum af ordretotaler
Sjælland	Kød	kr 2.485,26
	Mælkeprodukter	kr 7.895,12
	Drikkevarer	kr 4.346,00
	Slik	kr 16.255,78
Jylland	Kød	kr 3.321,34
	Mælkeprodukter	kr 11.566,16
	Drikkevarer	kr 6.547,98
	Slik	kr 22.456,12
Fyn	Kød	kr 1.786,45
	Mælkeprodukter	kr 6.588,81
	Drikkevarer	kr 3.845,32
	Slik	kr 12.875,64
Øvrige	Kød	kr 1.164,94
	Mælkeprodukter	kr 4.687,22
	Drikkevarer	kr 2.487,91
	Slik	kr 9.864,73

3

Geografisk område	Produktkategori	Sum af ordretotaler
Sjælland	Kød	kr 2.485,26
	Mælkeprodukter	kr 7.895,12
	Drikkevarer	kr 4.346,00
	Slik	kr 16.255,78
	Sum	
Jylland	Kød	kr 3.321,34
	Mælkeprodukter	kr 11.566,16
	Drikkevarer	kr 6.547,98
	Slik	kr 22.456,12
	Sum	
Fyn	Kød	kr 1.786,45
	Mælkeprodukter	kr 6.588,81
	Drikkevarer	kr 3.845,32
	Slik	kr 12.875,64
	Sum	
Øvrige	Kød	kr 1.164,94
	Mælkeprodukter	kr 4.687,22
	Drikkevarer	kr 2.487,91
	Slik	kr 9.864,73
	Sum	

4

Geografisk område	Produktkategori	Sum af ordretotaler
Sjælland	Kød	kr 2.485,26
	Mælkeprodukter	kr 7.895,12
	Drikkevarer	kr 4.346,00
	Slik	kr 16.255,78
	Sum	kr 30.982,16
Jylland	Kød	kr 3.321,34
	Mælkeprodukter	kr 11.566,16
	Drikkevarer	kr 6.547,98
	Slik	kr 22.456,12
	Sum	kr 43.891,60
Fyn	Kød	kr 1.786,45
	Mælkeprodukter	kr 6.588,81
	Drikkevarer	kr 3.845,32
	Slik	kr 12.875,64
	Sum	kr 25.096,22
Øvrige	Kød	kr 1.164,94
	Mælkeprodukter	kr 4.687,22
	Drikkevarer	kr 2.487,91
	Slik	kr 9.864,73
	Sum	kr 18.204,80

Kubeoperationernes funktionalitet vil jeg beskrive med følgende matematiske model:

Definition af begreber for kuben

Vi har et resultatdatasæt, RDS, som består af et antal tupler, Tpl:

RDS = Tpl*

Kuben er defineret ved et resultatdatasæt og et view, V:

K = RDS x V

Et view består af en liste af ikke-tom liste af dimensioner, en liste af skjulte dimensioner og en measure:

V = DimList x HdimList x M

DimList = $<d_1, .., d_i>$, i > 0

HdimList = $<h_1, .., h_i>$, i ≥ 0

En skjult dimension er en dimension, som optræder i resultatdatasættet, men ikke er med i kuben.

Operationer på Kuben

Fælles for de fem kubeoperationer er, at de alle udføres på en enkelt dimension og resulterer et view.

Kubeoperation = RollUp | DrillDown | MoveLeft | MoveRight | MoveFarLeft

Kubeoperation: K x Dim → V

RollUp funktionen kan kun udføres på dimension i, når der er mere end i elementer i d-listen:

$RollUp((<d_1, .., d_i, .., d_k>, <h_l, .., h_m>, M), d_i) = (<d_1, .., d_i>, <d_{i+1}, .., d_k, h_l, .., h_m>, M)$, $k > i$

DrillDown funktionen kan kun udføres på det sidste element i listen af dimensioner og listen af skjulte dimensioner ikke er tom.

$DrillDown((<d_1, .., d_i>, <h_l, , h_2, .., h_m>, M), d_i) = (<d_1, .., d_i, h_l>, <h_2, .., h_m>, M), HdimList \neq Ø$

MoveLeft funktionen kan udføres på alle på nær den første dimension i kuben.

$MoveLeft((<d_1, .., d_{i-1}, d_i, .., d_k>, <h_l, .., h_m>, M), d_i) = (<d_1, .., d_i, d_{i-1}, .., d_k>, <h_l, .., h_m>, M), i \neq 1$

MoveRight funktionen kan udføres på alle på nær den sidste dimension i kuben.

$MoveRight((<d_1, .., d_i, d_{i+1}, .., d_k>, <h_l, .., h_m>, M), d_i) = (<d_1, .., d_{i+1}, d_i, .., d_k>, <h_l, .., h_m>, M), i < k$

MoveFarLeft funktionen kan udføres på alle på nær de to første dimension i kuben.

$MoveFarLeft((<d_1, .., d_{i-1}, d_i, d_{i+1}, .., d_k>, <h_l, .., h_m>, M), d_i) = (<d_i, d_1, .., d_{i-1}, d_{i+1}, .., d_k>, <h_l, .., h_m>, M), i > 2$

4 Implementering

Implementeringen læner sig tæt op af designbeskrivelsen, og derfor vil jeg ikke gennemgå, hvordan hver enkel komponent er implementeret. I stedet vil jeg fremhæve nogle enkelte dele af implementeringen, som jeg føler, er særligt interessante. For en samlet oversigt over implementeringen henviser jeg til klassediagrammerne i appendiks 13.

4.1 Væsentlige aspekter af implementeringen

Implementeringsdelen af dette projekt bygger på Microsofts .NET framework[18]. Kernen i .NET frameworket er *Common Language Runtime* (CLR), der kan sammenlignes med Java Virtual Machine (JVM). CLR'en aktiverer objekter, udfører sikkerhedstjek på dem, lægger dem i hukommelsen, eksekverer dem og foretager garbage-collection af dem.

Figur 4-1 - Oversigt over .NET framework

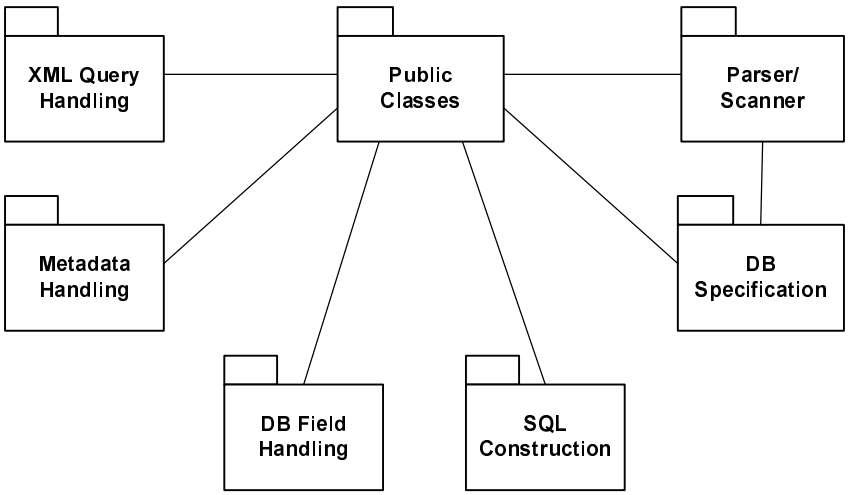
På disse punkter ligner CLR meget JVM, men en væsentlig forskel er dog, at CLR understøtter mere end 20 programmeringssprog. Jeg har valgt at benytte Visual Basic sproget til at løse implementeringsopgaverne i dette projekt, som i øvrigt omfatter to forskellige programmeringsdiscipliner. eFuture motoren er implementeret som en .NET *assembly*, hvilket er en logisk DLL[19] og et *manifest*, som indeholder en detaljeret beskrivelse (metadata) af assembly'en.
UI input og output er implementeret ved brug af ASP.NET Web Forms, som adskiller sig væsentligt fra traditionel ASP programmering ved at tillade adskillelse af applikationslogik(server-side kode) og præsentationslag(HTML).

Jeg vil bemærke, at der er tale om en prototype, og jeg har derfor ikke ønsket at bruge særlig meget tid på grafik og layout på grænsefladen.

[18] .NET frameworket er en integreret Windows komponent, som benyttes til at bygge og afvikle applikationer.
[19] Dynamic Link Library – refereres også til som et Windows COM modul.

eFuture motor

Denne del er implementeret i overensstemmelse med designet, og jeg har inddelt klassediagrammerne i syv UML pakker for at opnå en højere grad af overskuelighed.



Figur 4-2 - Pakkeoversigt for eFuture motor

I enkelte af pakkerne har jeg fundet det nødvendigt at tilføje et par klasser og metoder i forhold til, hvad jeg havde angivet i designet. Dette gælder bl.a. de tilfælde, hvor der skal indlæses data fra en XML tekst. Valget stod imellem at anvende en træ-baseret parser, som konstruerer et træ af XML-noder, eller en *stream*-baseret parser, som læser XML-filen som en strøm. Jeg valgte den *stream*-baserede SAX[20] parser, da den ikke optager nær så meget hukommelse, som den træ-baserede parser. SAX parseren fungerer på den måde, at den læser datastrømmen og rejser *events* efterhånden som den støder på *tags* eller tekststrenge. De klasser, som implementerer SAX parseren, er GetMetaFields og GetMetaGroups i *Metadata Handling* pakken, samt de seks klasser, som er navngivet "…Reader".

En anden detalje, som jeg vil nævne, er, at QueryBuilder klassen kan benyttes på en måde, så dens tilstand fastholdes. Dermed bevares informationerne om grafen, så den ikke skal genopbygges ved hver forespørgsel. Der er kun behov for at genopbygge grafen i tilfælde af, at det benyttede metalag ændres eller udskiftes. Dette kræver selvfølgelig, at applikationen, som benytter QueryBuilderen holder liv i objektet. I vores tilfælde, hvor der er tale om en webapplikation, kan dette gøres ved at

[20] Simple API for XML.

gemme QueryBuilder objektet i den såkaldte *cache*[21]. Her er også muligt at angive en afhængighed, som I vores tilfælde f.eks. kunne være til en metadata XML-fil. Dette betyder, at hvis XML-filen ændres eller slettes, så vil ASP.NET processen registrere ændringerne og give besked til cachen om at fjerne objektet.

UI input og UI Output

Som tidligere nævnt er disse to komponenter implementeret ved brug af ASP.NET Web Forms. Web Forms gør web programmering en del lettere end tidligere, da man ikke længere behøver at skrive *in-line* ASP scripts og tage hensyn til traditionel top-down parsning af ASP siderne. En anden væsentlig detalje i ASP.NET er den såkaldte *web.config* fil, som bruges til at angive egenskaberne for den enkelte webapplikation. I dette projekt anvender jeg web.config til at gemme oplysninger om stien til metadata XML-filen samt adgangsoplysningerne til databasen. Desuden er det herfra, at *session state management* kontrolleres. I ASP.NET kan man vælge imellem tre indstillinger:

Indstilling	Beskrivelse
In-process	Her er det web serveren, der håndterer session state. Dette betyder, at hvis web serveren genstartes, så mistes oplysningerne om session state.
Out-of-process	Session state styres her af en separat ASP.NET Session State Service, hvilket betyder, at session state bevares i tilfælde af at web serveren genstartes.
SQL Server	Dette er den sikreste af tre indstillinger, og her gemmes session state oplysningerne i en database.

Jeg har valgt in-process indstillingen, da denne giver bedste performance for applikationen. De to andre indstillinger er langsommere, da de medfører en del overhead i kommunikationen. Implementeringen af UI Input omfatter tre *aspx*-filer – en for hvert trin på Figur 3-15. *aspx* er betegnelsen for en WebForm, og hver fil består af en HTML del og en *codebehind* del, som indeholder VB koden, som afvikles på serveren. Jeg har ikke valgt at gøre meget ud af det grafiske layout, men blot fokuseret på, at funktionaliteten på siderne er enkel og god. Dette betyder, at sidernes HTML del ikke indeholder særligt meget kode. Desuden bliver indholdet af siderne

[21] *Cache* er egentlig navnet på .NET objektet, som benyttes til at kontrollere applikationstilstande.

genereret dynamisk ud fra metadata, når siderne indlæses, og metoderne, som håndterer dette, findes i codebehind. Brugeren har mulighed for at vælge èn measure og fem dimensioner, som det er blevet specificeret i *Krav ID 10* og *12*. Måden, som jeg har valgt at implementere UI input på, betyder, at der laves et roundtrip til serveren imellem hvert trin i indtastningen. Derfor er der behov for at overføre brugerens valgte parametre imellem websiderne, og dette kan gøres på flere forskellige måder. Jeg har valgt at løse dette problem ved at tilføje de valgte parametre til URL'ens QueryString. Denne løsning har jeg valgt, fordi den giver et godt billede af applikationens funktionalitet, da man nemt kan se både de synlige og skjulte dimensioner. Ulempen er dog, at brugeren selv kan redigere i QueryString'en, hvilket kan medføre uhensigtsmæssige resultater og i værste tilfælde fejl. I en endelig version af systemet bør parameteroverførslen foregå via ASP.NET *Server* objektet[22].

Den primære overvejelse, som skulle gøres i henhold til UI output, bestod i at afgøre, hvordan selve kuben skulle implementeres. I første omgang kiggede jeg på Excel pivottabeller, som er en del af Office Web Components [OFF1]. Pivottabellerne har den klare fordel, at man nemt kan anvende Excels øvrige funktionalitet til at analysere data og opbygge diagrammer. Ulempen ved pivottabellerne er, at de kræver, at brugeren har installeret en bestemt version af Office, og dette mener jeg vil hæmme produktets fleksibilitet. Min næste ide var at implementere kuben i et .NET DataGrid, som gør det nemt at præsentere store mængder data på webgrænsefladen. Desværre skulle det vise sig, at det ikke var muligt at tilpasse DataGrid'et på den måde, som jeg har beskrevet i designafsnittet. Den endelige løsning blev at implementere kuben ved brug af en traditionel HTML tabel, og formateringen følger de fire trin fra afsnit 3.6. Kubeoperationerne udføres ved aktivering af hyperlinks med passende ikoner, og URL'erne for hvert hyperlink beregnes med metoder, som er implementeret ud fra den matematiske model, som jeg beskrev i samme afsnit. Endelig vil jeg nævne, at Null værdier i resultatdatasættet oversættes til "N/A", således at disse opsamles under hver dimension.

[22] Metoden er beskrevet på [MSDN1].

5 Integration til tidsregistreringssystem

I dette afsnit vil jeg beskrive gennemførelsen af et mini-pilotprojekt, som gennemgår de trin, der skal til for at anvende eFuture systemet i forretningssammenhæng. Baggrunden er, at NNIT i dag anvender et system – NIKU – til at registrere projekter, opgaver, tidsforbrug, sygedage, etc. Chefer og projektledere har dagligt behov for at danne sig et overblik over de enkelte projekters status, således at de kan holde styr på tidsestimater og budgetter. Til dette formål har NNIT et værktøj til at udtrække rapporter, men brugerne er ikke specielt tilfredse med dette værktøj. Databasen, som NIKU systemet anvender, er en Microsoft SQL Server, og derfor oplagt at teste eFuture systemet på.

Det første trin i integrationsprocessen består i at opskrive metadata. Indstillingerne for databasen er straight-forward efter, at jeg er blevet tildelt læseadgang til databasen. Nu gælder det om at bestemme hvilke attributter, der er interessante at medtage, og til at hjælpe mig med dette har jeg min NNIT vejleder, der har et godt kendskab til databasens struktur. En vigtig ting, som jeg vil nævne er, at nogle data i databasen er meget personfølsomme, og som NNITs politik forbyder mig at se. Løsningen på dette problem er at udelade disse følsomme data fra metadata, således at de ikke bliver tilgængelige i eFuture. En ting, som jeg ønsker at fremhæve, er et par eksempler på, hvordan man intelligent kan udnytte metalagets muligheder. Det første eksempel går ud på, at vi ønsker at opstille tidsdimensionen og opretter et felt i metadata, som vi kalder *ÅR*. Der er ingen ”år-attribut” i databasen, men i stedet anvender vi SQL funktionen DATEPART på en dato-attribut og får således året ud. Tilsvarende gøres for de øvrige tidsdimensioner.

```
<...>
<DBExpression type="int">DATEPART(year, Orders.OrderDate)</DBExpression>
<...>
```

Et andet trick, som jeg faldt over, er at benytte SQL funktionen CASE…WHEN til at oprette sine egne grupperinger. For eksempel kan vi oprette et felt, *LAGERSTATUS*, som inddeler produkter efter om, der er *få*, *nogle* eller *mange* på lager.

```
<...>
<DBExpression type="string">
CASE  WHEN Products.UnitsInStock < 5 THEN 'A few'
      WHEN Products.UnitsInStock >= 5 AND Products.UnitsInStock < 10 THEN 'Some'
      ELSE 'Many'
END
</DBExpression>
<...>
```

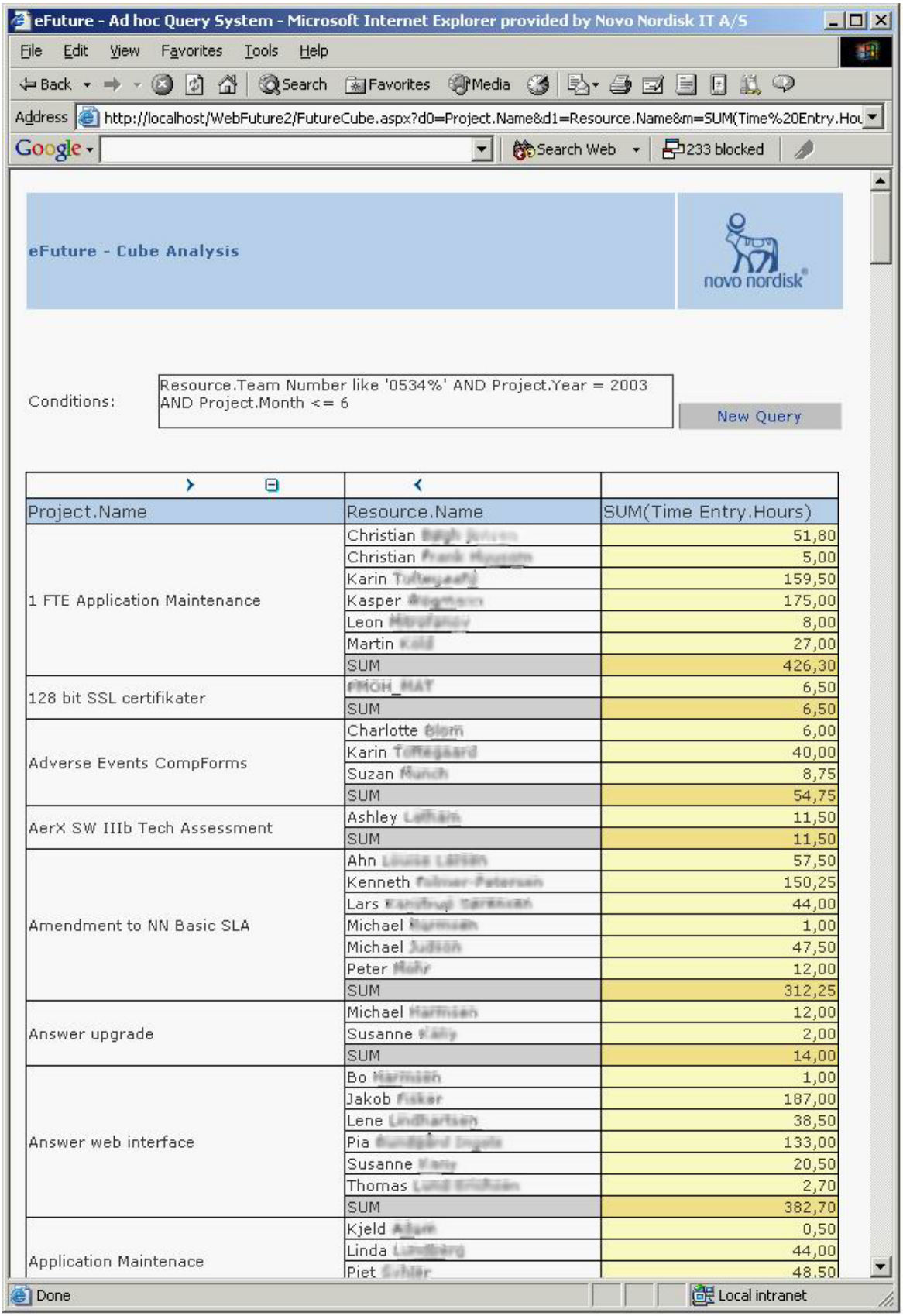
Endeligt vil jeg nævne SQL funktionen DATEDIFF, som beregner differencen imellem to datoer. Denne bruger vi til at definere et felt, *LEVERINGSTID*, som fortæller antallet af dage, som er gået fra en ordre er modtaget til ordren er leveret.

```
<...>
<DBExpression type="int">DATEDIFF([Day], OrderDate, DeliveryDate)</DBExpression>
<...>
```

Med metalaget færdigskrevet er det på tide at gemme dokumentet i en folder på serveren og specificere stien til filen i *web.config*. Vi er nu klar til at foretage de første forespørgsler og vil undersøge følgende problemstillinger:

- Hvad er det samlede timeforbrug i første halvdel af 2003 for WebSolutions afdelingen inddelt efter projekt og medarbejder?

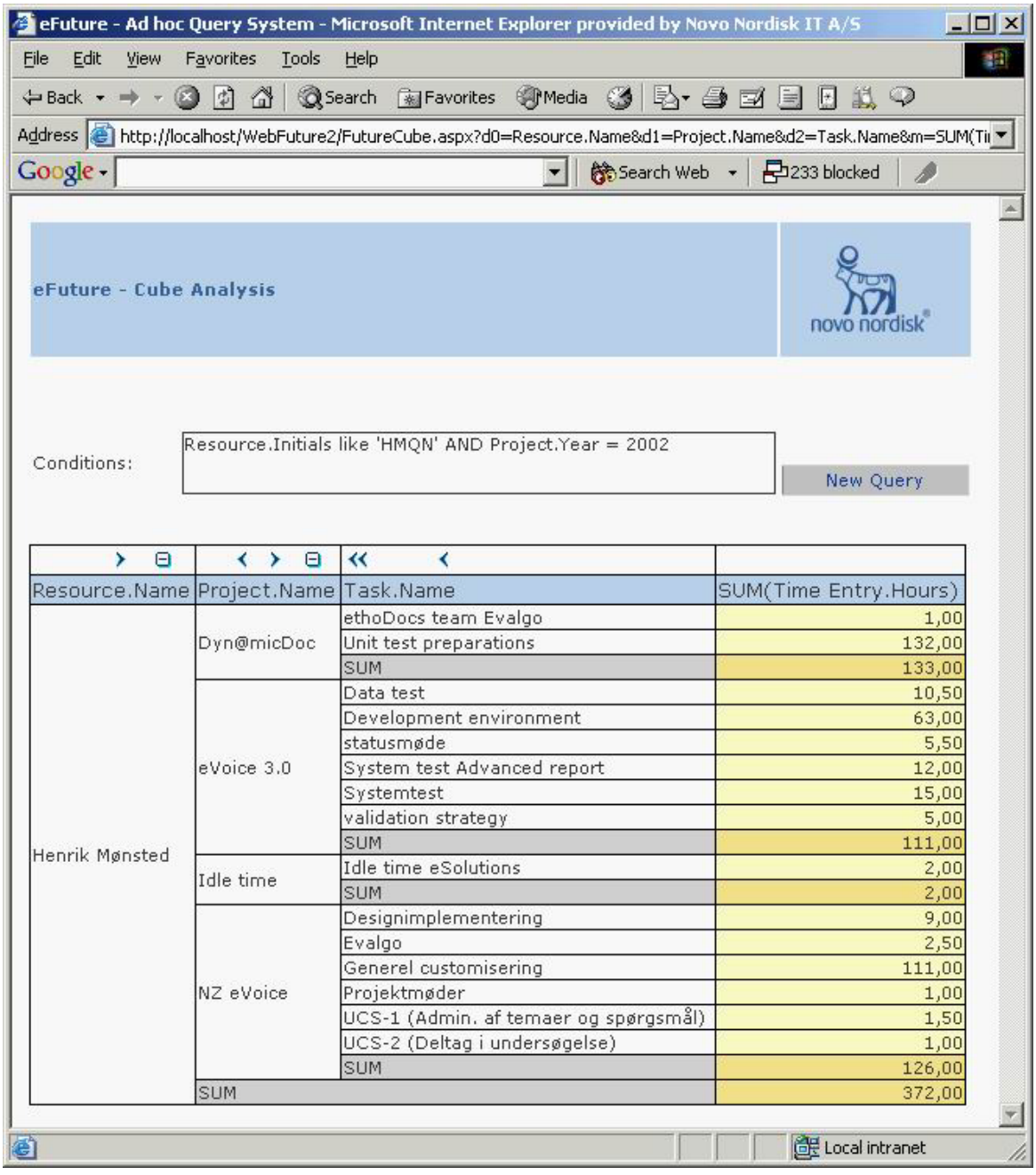
Resultatet (Figur 5-1) kom frem på skærmen ca. 4 sekunder efter, at forespørgslen var blevet afsendt. Som det fremgår af skærmbilledet vises en oversigt over projekter, navnene på projektdeltagerne samt summen af timer, som der er blevet registreret. Ved klik på ’-’ ikonet over Projekt-kolonnen skjules kolonnen med projektdeltagerne, og der vises nu kun summen af timeregistreringer for hvert projekt. Den anden mulighed er at bytte om på de to dimensioner – dette gøres ved at klikke på en af pilene over hver kolonne.



Project.Name	Resource.Name	SUM(Time Entry.Hours)
1 FTE Application Maintenance	Christian [illegible]	51,80
	Christian [illegible]	5,00
	Karin [illegible]	159,50
	Kasper [illegible]	175,00
	Leon [illegible]	8,00
	Martin [illegible]	27,00
	SUM	426,30
128 bit SSL certifikater	[illegible]	6,50
	SUM	6,50
Adverse Events CompForms	Charlotte [illegible]	6,00
	Karin [illegible]	40,00
	Suzan [illegible]	8,75
	SUM	54,75
AerX SW IIIb Tech Assessment	Ashley [illegible]	11,50
	SUM	11,50
Amendment to NN Basic SLA	Ahn [illegible]	57,50
	Kenneth [illegible]	150,25
	Lars [illegible]	44,00
	Michael [illegible]	1,00
	Michael [illegible]	47,50
	Peter [illegible]	12,00
	SUM	312,25
Answer upgrade	Michael [illegible]	12,00
	Susanne [illegible]	2,00
	SUM	14,00
Answer web interface	Bo [illegible]	1,00
	Jakob [illegible]	187,00
	Lene [illegible]	38,50
	Pia [illegible]	133,00
	Susanne [illegible]	20,50
	Thomas [illegible]	2,70
	SUM	382,70
Application Maintenace	Kjeld [illegible]	0,50
	Linda [illegible]	44,00
	Piet [illegible]	48,50

Figur 5-1 - Skærmbillede fra eFuture

- Hvor mange timer har Henrik Mønsted (HMQN) registreret på forskellige projektopgaver i 2002.



Resource.Name	Project.Name	Task.Name	SUM(Time Entry.Hours)
Henrik Mønsted	Dyn@micDoc	ethoDocs team Evalgo	1,00
		Unit test preparations	132,00
		SUM	133,00
	eVoice 3.0	Data test	10,50
		Development environment	63,00
		statusmøde	5,50
		System test Advanced report	12,00
		Systemtest	15,00
		validation strategy	5,00
		SUM	111,00
	Idle time	Idle time eSolutions	2,00
		SUM	2,00
	NZ eVoice	Designimplementering	9,00
		Evalgo	2,50
		Generel customisering	111,00
		Projektmøder	1,00
		UCS-1 (Admin. af temaer og spørgsmål)	1,50
		UCS-2 (Deltag i undersøgelse)	1,00
		SUM	126,00
	SUM		372,00

Figur 5-2 - Skærmbillede fra eFuture

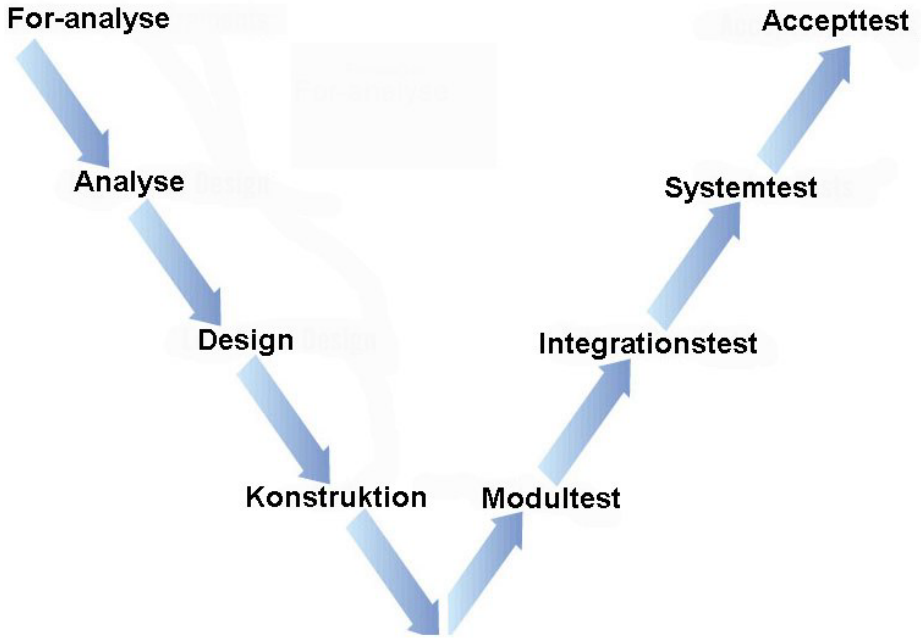
Her er der blevet tilføjet en kolonne, således at detaljeringen nu øges til også at vise de enkelte opgaver, som ligger under hvert projekt.

6 Test

I dette afsnit vil jeg beskrive de overordnede testfaser i et softwareudviklingsprojekt. Jeg vil komme ind på, hvordan man planlægger og udfører en test, samt vise eksempler fra de tests, som jeg har udført i løbet af dette projekt. Jeg har valgt ikke at udføre en fuldstændig test af eFuture implementeringen. Dette er gjort ud fra den vurdering, at projektets omfang ville blive for stort i forhold til den tid, som jeg havde til rådighed, og at der er tale om en prototype. De testdokumenter, som jeg har udarbejdet undervejs i projektet, kan findes i appendiks 13.

6.1 Principper og metoder

V-modellen, som er vist på Figur 6-1, er nok den bedst kendte model at teste software efter. Den illustrerer rækkefølgen af udviklings- og testaktiviteter i et softwareudviklingsprojekt. Desuden viser den, hvilken udviklingsfase, der hører sammen en testfase. En anden ting, som man kan læse ud fra modellen er, at jo senere i projektforløbet man opdager en fejl jo sværere og dyrere er den at reparere. Det vil altså sige, at en fejl fra analysen, som identificeres under systemtesten, er dyrere end at finde en konstruktionsfejl under modultesten.



Figur 6-1 - V-modellen

Ideen med V-modellen er, at testopgaverne planlægges sideløbende med udviklingen. Systemtesten planlægges altså parallelt med analysen. Der er to begreber mere, der er væsentlige at få på plads i forbindelse med test. Dette er yderligere to V'er, nemlig Verifikation og Validering. Verifikation er processen at afgøre, om produktet er konsistent med de associerede krav og standarder. Validering er processen at vurdere om softwaren er i overensstemmelse med kravspecifikationen. Disse definitioner kan oversættes:
Validering – Udfører vi den rigtige opgave?
Verifikation – Udfører vi opgaven rigtigt?

Modultest

Den første test, som skal udføres, er en modultest – eller *unit test*, som den også kaldes. Testen består i at teste systemets komponenter enkeltvis for at sikre, alle komponenter fungerer korrekt og opfylder deres specifikation. Et modul kan i vores tilfælde, hvor der er anvendt et objektorienteret sprog, betragtes som:

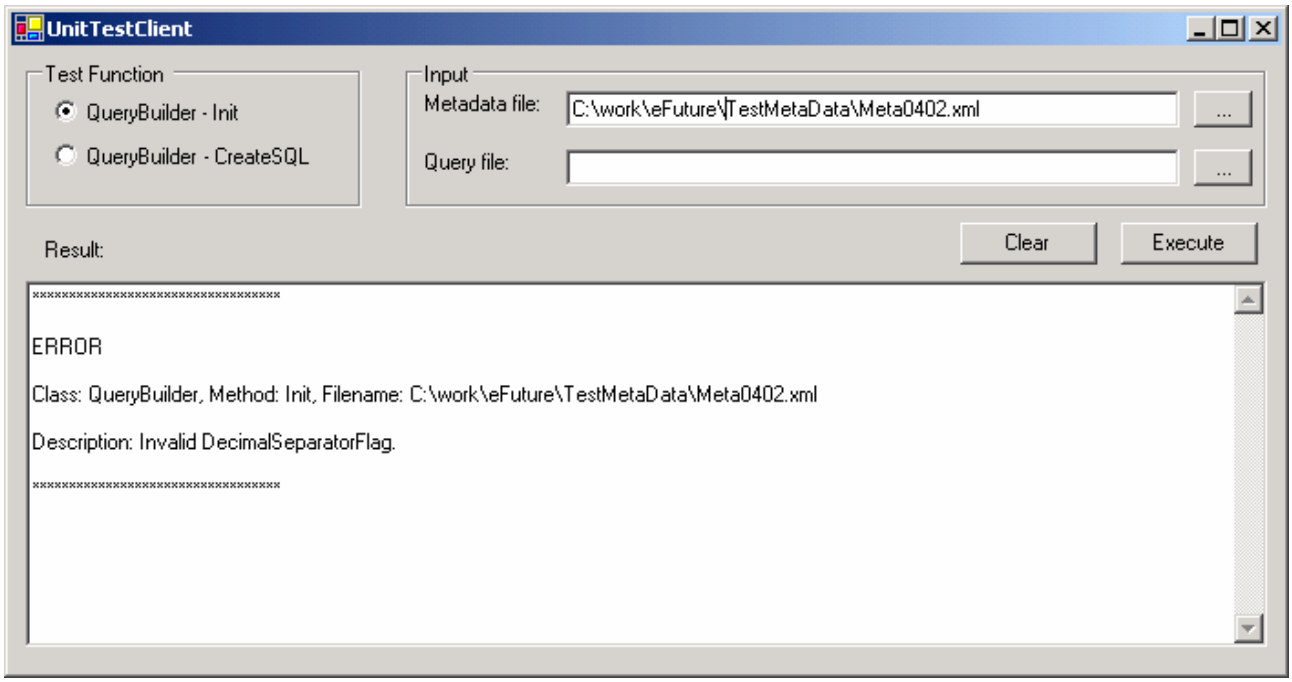
- En pakke bestående af et antal klasser.
- En klasse bestående af et antal metoder og egenskaber.

Modultestens formål er at validere de enkelte metoders funktionalitet, og i en lagdelt arkitektur vil man vælge at starte på det laveste niveau (data niveau) og bevæge sig op igennem klasserne efterhånden som modultestene bliver godkendt. En ting, som der skal tages stilling til inden modultesten kan påbegyndes, er om den skal udføres som en *Black Box* eller en *White Box* test. Black Box består i at udfærdige en liste over inputs til et modul og en tilsvarende af forventede outputs. Modulet har bestået testen, når alle generede outputs stemmer overens med de forventede outputs. White Box går ud på at teste hver eneste instruks eller forgrening i modulets kode med henblik på at dække et bestemt omfang. Der skelnes imellem tre typer dækning:

- Statement-coverage måles på antallet af testede instrukser.
- Decision-coverage dækker i forhold til antallet af forgreninger (IF, CASE OF, osv.).
- Path-coverage indenticerer antallet af mulige veje.

White Box test kan være meget tidskrævende, og derfor kombineres den ofte med Black Box ud fra en risikoanalyse af de enkelte modulers funktionalitet.

Jeg vil vise et eksempel på en black box modultest, som jeg har udført på Init-metoden i QueryBuilder klassen. Til at udføre de enkelte trin har jeg lavet en simpel Windows Form (Figur 6-2), hvor man vælger funktionen, der skal afprøves, og angiver input.



Figur 6-2 - Klient til modultest

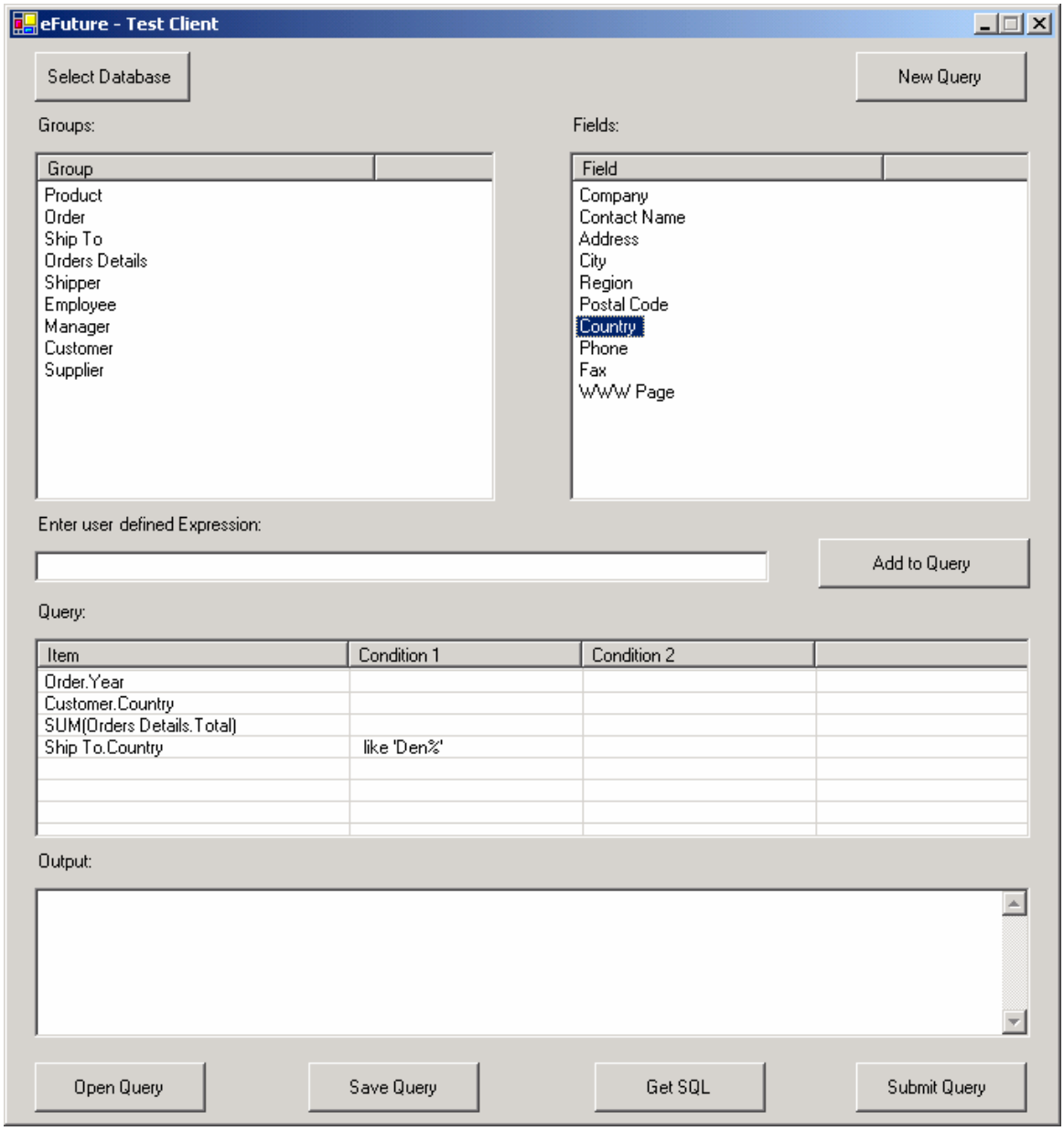
I dette tilfælde har jeg forberedt en række metadata XML-filer, som hver især indeholder en af de fejltyper, som kan forekomme i metalaget. Tabel 6-1 viser testplanen for modultesten, som udfyldes med de aktuelle resultater efterhånden, som trinene bliver gennemført. Dette kan f.eks. gøres ved at indsætte skærmbilleder, som det der er vist på Figur 6-2.

Test Område: 01		**Test Case: 1.1**		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
1	Metadata fil: Meta0401.xml	Følgende tekst vises i tekstfeltet: “QueryBuilder has been initialized.”		
2	Metadata fil: Meta0402.xml	Der vises en fejl med følgende beskrivelse: ”Invalid DecimalSeparatorFlag”		
3	Metadata fil: Meta0403.xml	Der vises en fejl med følgende beskrivelse: ”Invalid DateDiffStyle”		
4	Metadata fil: Meta0404.xml	Der vises en fejl med følgende beskrivelse: “Invalid 'EmptyStringIncludesNullString' value”		
5	Metadata fil: “xxx”	Der vises en fejl med følgende beskrivelse: “Error parsing metadata. Possibly invalid filename”		
6	Metadata fil: Meta0405.xml	Der vises en fejl med følgende beskrivelse: “Not all tables are in graph. Missing table: PRODUCTSERROR”		
7	Metadata fil: Meta0406.xml	Der vises en fejl med følgende beskrivelse: “The table graph is disconnected.”		
8	Metadata fil: Meta0407.xml	Der vises en fejl med følgende beskrivelse: “Relation used twice.”		
9	Metadata fil: Meta0408.xml	Der vises en fejl med følgende beskrivelse: “Metadata contains the same field name twice.”		

Tabel 6-1 - Testplan for modultest

Integrationstest

Her testes om systemets grænseflader er implementeret korrekt. Det drejer sig dels om de interne grænseflader mellem moduler og dels om de eksterne grænseflader. I relation til eFuture kan der være tale om at teste grænsefladen mellem eFuture motoren og UI komponenterne. Et eksempel på dette er at teste om GetSQL metoden returnerer det rette SQL-udtryk ved at sammenligne med resultatet af en manuel beregning (foretaget i hånden) ud fra databasediagrammet. Endvidere kan man teste om resultatdatasættet, der returneres fra GetDataSet metoden, indeholder de korrekte data. Dette gøres ved at indtaste det manuelt beregnede SQL-udtryk i SQL Server Query Analyser (for Microsoft SQL Server) og sammenligne de to datasæt. For at kunne teste eFuture motorens individuelle funktionalitet har jeg udviklet en testklient, som kan ses på Figur 6-3. En anden test går ud på at afprøve integrationen til hhv. SQL Server og Oracle databaserne. Derfor har jeg udarbejdet to metadatafiler, som ved hjælp af testklienten benyttes at afprøve eFuture motorens samspil med de to databaser. Filerne hedder Northwind.xml (SQL Server) og Oracle.xml, og de kan findes i appendiks 13.



Figur 6-3 - Klient til test af eFuture motor

Systemtest

Nu betragtes systemet som en helhed. Systemtesten foretages fra brugergrænsefladen og skal eftervise, at den funktionalitet, som kravspecifikationen beskriver, er til stede og virker. Det er en god ide at angive sammenhængen imellem kravene og de enkelte test cases, så det tydeligt kan spores, om systemegenskaberne er testet til fulde. En anden tommelfingerregel er at undgå lange test cases, da man i tilfælde af fejl skal starte forfra med test casen. I appendiks 13 er vist et eksempel på en systemtestplan, hvor følgende krav til eFuture systemet er bliver afprøvet:

Testområde: Definering af forespørgsel	
Krav ID	**Testet i test case nr.**
10	2.1
11	2.2
12	2.1
13	2.3
14	2.2
15	2.4
16	2.5

Accepttest

Formålet med denne test er, at den skal danne grundlag for en beslutning, om systemet opfylder de krav, der er blevet stillet til det. Således adskiller accepttesten sig fra de øvrige tests, som alle går ud på at finde fejl. Her er der tale om en accept af hele produktet, herunder dokumentation og forretningsgange.

Testkonklusion

Jeg vil afslutningsvis redegøre for resultaterne af de tests, som jeg har gennemført i løbet af projektet. Størstedelen af de fejl, som har vist sig, er blevet fundet ved modultests i løbet af konstruktionsfasen. Yderligere er der blevet fundet et par mindre fejl i forbindelse ved afprøvningen gennem testklienten (Figur 6-3). Der er ikke fundet nogen fejl under systemtesten. Alle fejl, som har vist sig, er blevet rettet, og systemet fungerer nu yderst stabilt.

Der findes en lang række andre former for tests. Man er ofte interesseret i at undersøge det færdige systems ydeevne. Dette kan f.eks. være svartider i forbindelse med online systemer eller antallet af

transaktioner, der kan behandles i løbet af et givent tidsrum. Det ville være rart at teste eFuture systemets ydeevne for at sammenligne det med andre ad hoc værktøjer, men det er svært at opnå nøjagtige resultater, da disse vil afhænge af den underliggende databases performance.

7 Konklusion

Projektmålet var at designe og implementere en prototype på et ad hoc rapporteringsværktøj, som anvender grafteori til at modellere databasetabeller og -relationer og benytte grafteoretiske metoder til at bestemme, hvordan data udvælges. Endvidere skulle værktøjet kunne anvendes på en vilkårlig database struktur og have en webgrænseflade, hvori resultatet af forespørgslen blev præsenteret i en kube. Kuben skulle tilbyde brugeren en række analysemuligheder, som bl.a. drill-funktionalitet.

Første skridt bestod i at analysere ad hoc processen og kubebegrebet. Efter at have defineret koblingen imellem databaser og grafer, opstillede jeg et case-baseret eksempel med det formål, at det skulle give overblik over systemets tilsigtede funktionalitet. Med begreberne og metoden på plads analyserede jeg de problemer, som jeg havde identificeret i forbindelse med hhv. forespørgslen og kuben. De væsentligste problemer omhandlede kredse i grafen og relationssammensætninger, som ville medføre ukorrekte beregninger. Løsningen på kredsproblematikken er at lade brugeren levere input til de beslutninger, som skal tages, hver gang en kreds identificeres. Desværre kunne jeg ikke finde en ”smart” automatiseret løsning på disse problemer. Jeg er dog overbevist om, at beslutningen om ikke at tillade kredse i grafen er den rigtige beslutning i forhold til projektets tidsramme.

eFuture motoren blev designet med henblik på fremtidige udnyttelsesmuligheder. Her tænker jeg særligt på muligheden for at skrive brugerdefinerede udtryk, som ikke er relevant i forhold til den implementerede kube, men i høj grad øger eFuture motorens fleksibilitet. En anden fordel ved designet er anvendelsen af XML-formattet, som bl.a. betyder, at tredjeparts klienter nemt kan integrere til eFuture motoren. Således er det ikke kun .NET komponenter, som har glæde af dens funktionalitet.

Prototypen fungerer som helhed rigtig godt, og alle kravene, som var specificeret, er blevet opfyldt. Jeg nåede ikke, at gennemføre og dokumentere en fuldstændig test af systemet, og dette skal naturligvis gøres inden produkt kan sættes i produktion. Det er svært at konkludere, at vi har vundet noget ved at indføre den graforienterede algoritme til dataudtrækningen, da jeg ikke kan dokumentere denne påstand med specifikke tal. Jeg mener dog godt, at påstanden kan forsvares, idet de grafteoretiske metoder, som anvendes, sikrer, at vi altid opnår det ”rigtige” join-udtryk for

hver forespørgsel, så længe der ikke er kredse i grafen. Med det rigtige join-udtryk menes, at resultatet af forespørgslen er i overensstemmelse med brugerens semantiske opfattelse af problemet.

Case-studiet, hvor prototypen blev integreret til et eksisterende tidsregistreringssystem, løftede sløret for nogle af de muligheder, som eFuture giver, og bekræftede, at systemet med fordel kan anvendes i ”den virkelige verden”. Det er nok en smule forkert at sammenligne eFuture med de øvrige ad hoc værktøjer på markedet. Jeg mener ikke, at der findes et eneste produkt, der ligesom eFuture giver brugeren en så enkel adgang til at få svar på sine spørgsmål. De fleste konkurrerende produkter kræver i modsætning til eFuture, at brugeren installerer en stor mængde software på sin maskine og i øvrigt trænes i brugen af systemet.

Jeg konkluderer, at projektmålet er blevet udfyldt, da protypen har vist, at kombinationen af at anvende grafteori ved dataudtrækning og den kubebaserede præsentation er et stærk middel til at opnå en effektiv ad hoc rapporterings funktionalitet.

Projektevaluering

Jeg indledte projektet med at studere og afprøve en række ad hoc værktøjer for at danne mig et indtryk af disses funktionalitet. Denne erfaring har sammen udviklingen af eFuture givet mig et godt kendskab til denne type produkter, hvilket jeg forhåbentligt kan drage nytte af i fremtiden.
Fra et programmeringsteknisk synspunkt har jeg også høstet en masse erfaringer. Før dette projekt har jeg udelukkende beskæftiget mig med Java, og min viden om webudvikling har været begrænset til Applets og HTML. Det var været særdeles spændende at stifte bekendtskab med .NET teknologien, som Microsoft satser kraftigt på og konstant udvider med spændende funktionalitet, og jeg er sikker på, at der vil ske mange spændende ting indenfor dette område de næste par år. Desuden har jeg lært meget af at arbejde med XML-formattet, og jeg har opnået en god indsigt i de muligheder semistrukturerede data giver.

I begyndelsen af projektet førte jeg en logbog med mødereferater og noter til mit igangværende arbejde. Dette var en stor hjælp, eftersom jeg i perioder havde travlt med andre fag på DTU, og der gik somme tider op til en uge imellem, at jeg arbejdede på projektet. Ved at læse i logbogen kunne jeg hver gang hurtigt komme videre med de aktuelle opgaver. I den sidste halvdel af projektet, hvor

jeg arbejdede dagligt på projektet, var der ikke længere behov for denne logbog. En anden erfaring er, at det i mange faser af et softwareudviklingsprojekt er en fordel at have mere end 2 øjne på problemerne. Sammenlignet med mine tidligere projekter, hvor vi har to eller flere deltagere, har jeg i dette projekt savnet en sparringspartner – særligt i design og implementeringsfasen.

På baggrund af implementeringsresultaterne og evalueringen af projektforløbet konkluderer jeg, at projektet alt i alt har været en succes.

7.1 Fremtidigt arbejde

Her vil jeg kort nævne den funktionalitet, som jeg mener, at det kunne være interessant at udvide eFuture systemet med, inden det bliver introduceret på markedet. Det er min vurdering, at implementeringen af nedenstående punkter ikke omfatter væsentlige tekniske udfordringer, men i høj grad vil forbedre kundernes indtryk af produktet.

- Automatisk generering af metadata ud fra en eksisterende database.
- Håndtering af flere measures på samme tid.
- GUI til redigering af metadata.
- Angivelse af aggregeringsniveau i metadata, således at brugerne opnår forskellige rettigheder.
- Udskrivningsfunktionalitet, som gør det muligt at printe hele eller dele af kuben.
- Hurtig adgang til gemte forespørgsler.
- Eksport af resultatdatasæt til Excel.

8 Referencer

[Hop01] Hopcroft, John E., Introduction to automata theory, languages, and computation, 2. edition, ISBN 0201441241, Addison-Wesley, 2001.

[Ull02] Ullman, Widom, Garcia-Molina, Database systems – The complete book, ISBN 0130319953, Prentice Hall, 2002.

[Lar02] Larman, Craig, Applying UML and Patterns, Second edition, ISBN 0130925691, Prentice Hall, 2002.

[Gro94] Groff, Weinberg, LAN times – Guide to SQL, ISBN 007882026, McGraw-Hill, 1994.

[Nie95] Nielsen, Frank, Grafteori – algoritmer og netværk, ISBN 8788764494, Scantryk, 1995.

[Abi00] Abiteboul, Buneman, Suciu, Data on the web – From relations to semistructured data and XML, ISBN 155860622X, Morgan Kaufmann, 2000.

[Tha02] Thai, Lam, .NET Framework Essentials, ISBN 0596003021, O'reilly, 2002.

[Ric02] Richter, Jeffrey, Applied Microsoft .NET Framework Programming, ISBN 0735614229, Microsoft, 2002.

[And01] Anderson, Francis, Homer, Howard, Sussman, Watson, Professional ASP.NET, ISBN 1861004885, Wrox, 2001.

[Con02] Connell, John, Coding techniques for Microsoft Visual Basic.NET, ISBN 0735612544, Microsoft, 2001.

[Vin93] Vinje, Staal Poul, Test af Software, ISBN 8757113440, Teknisk Forlag, 1993.

[Sha00] R. Sharp, J. T. Kristensen, Software Development Projects, DTU, 2000.

[MSDN1] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconPassingServerControlValuesBetweenPages.asp

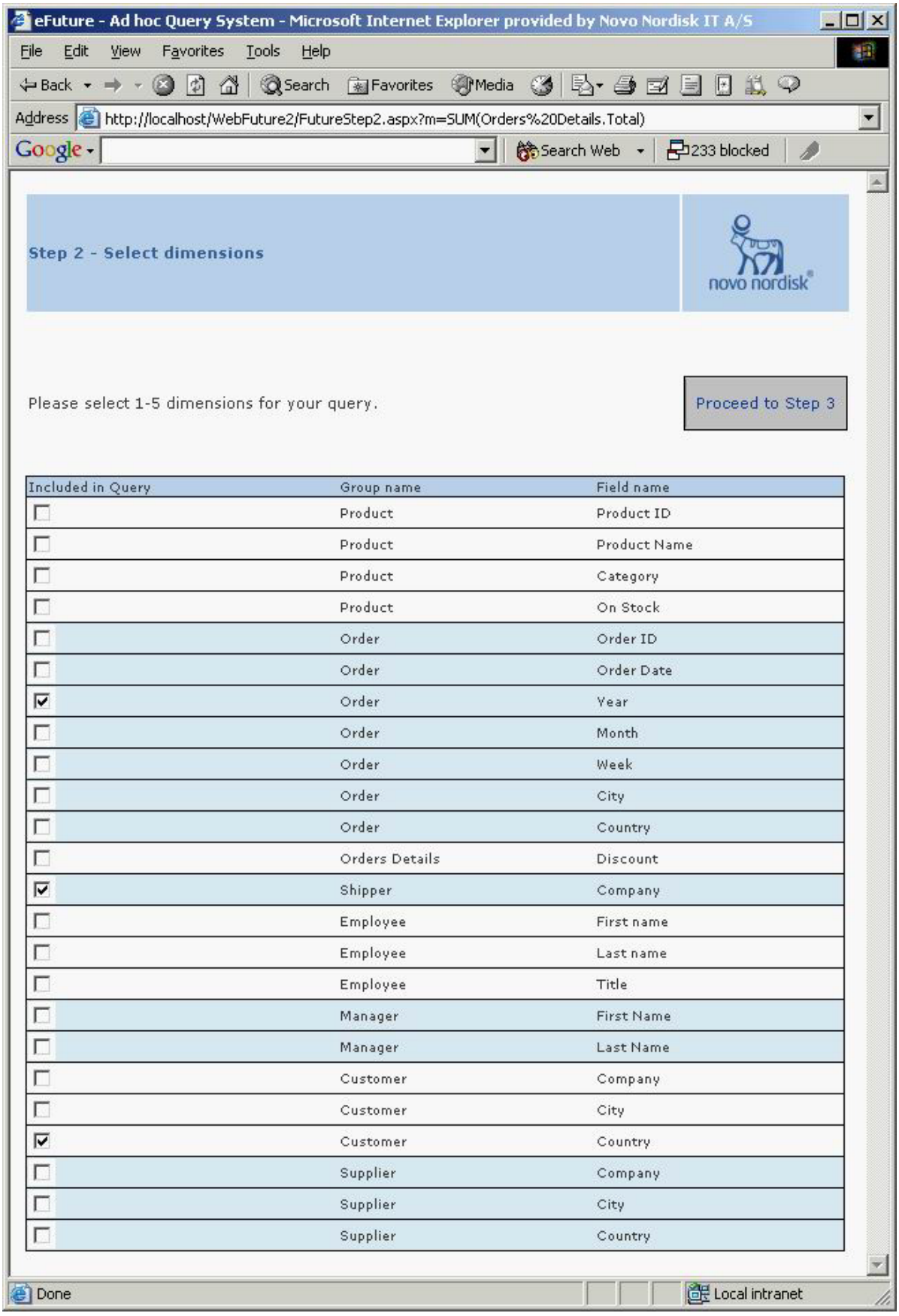
[OFF1] http://office.microsoft.com/assistance/preview.aspx?AssetID=HA010565431033&CTT=6&Origin=EC010553071033

[SPD1] http://www.speedware.com

[W3S1] http://www.w3schools.com/xml/default.asp

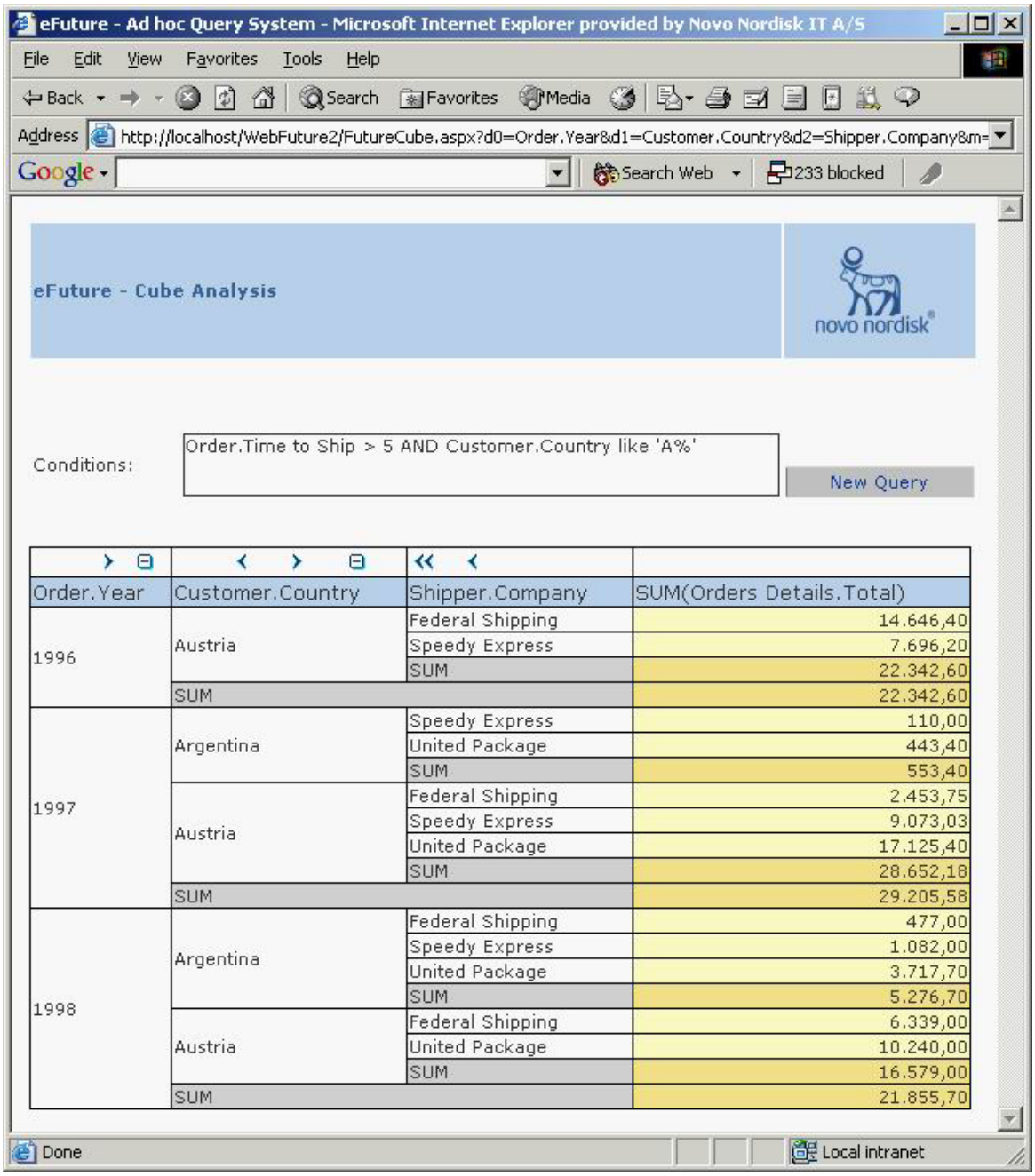
[DUL1] http://www.dulcian.com/

9 Appendiks: Skærmbilleder fra eFuture

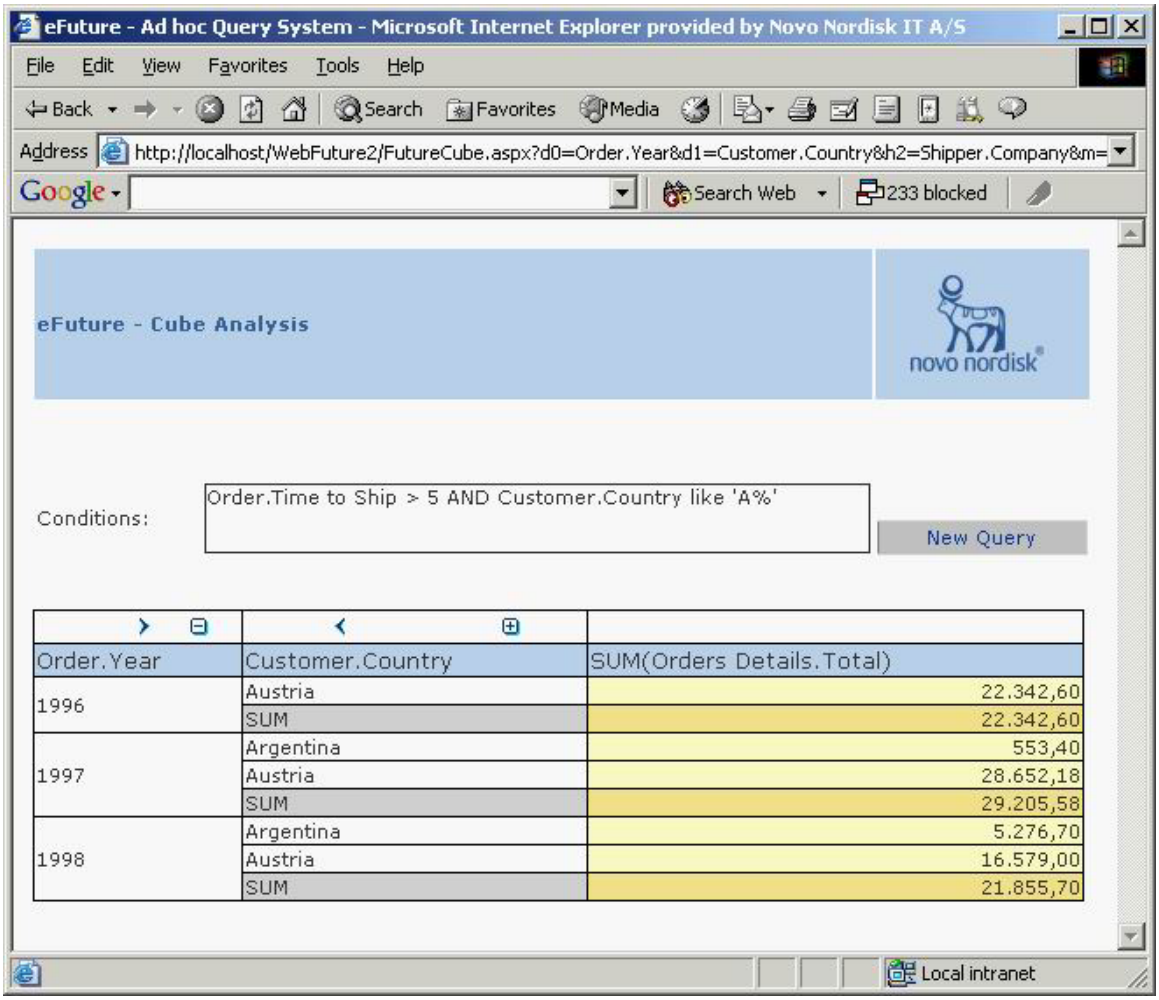

eFuture - Ad hoc Query System - Microsoft Internet Explorer provided by Novo Nordisk IT A/S
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebFuture2/FutureStep2.aspx?m=SUM(Orders%20Details.Total)
Google Search Web 233 blocked
Step 2 - Select dimensions
novo nordisk
Please select 1-5 dimensions for your query.
Proceed to Step 3
Included in Query | Group name | Field name
Product | Product ID
Product | Product Name
Product | Category
Product | On Stock
Order | Order ID
Order | Order Date
Order | Year
Order | Month
Order | Week
Order | City
Order | Country
Orders Details | Discount
Shipper | Company
Employee | First name
Employee | Last name
Employee | Title
Manager | First Name
Manager | Last Name
Customer | Company
Customer | City
Customer | Country
Supplier | Company
Supplier | City
Supplier | Country
Done
Local intranet

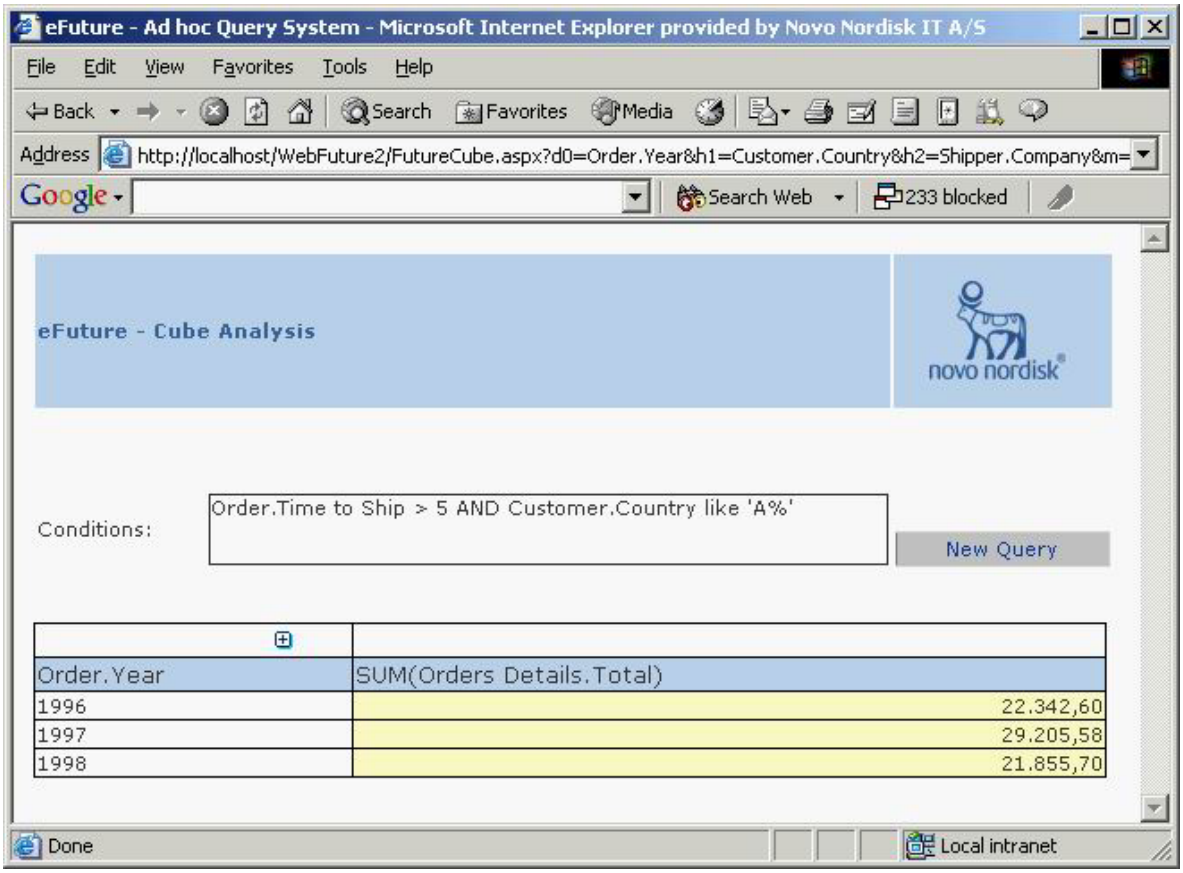

eFuture - Ad hoc Query System - Microsoft Internet Explorer provided by Novo Nordisk IT A/S
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebFuture2/FutureStep3.aspx?d0=Order.Year&d1=Shipper.Company&d2=Customer.Country&
Google Search Web 233 blocked
Step 3 - Select conditions
novo nordisk
Please enter the conditions for your query.
Submit Query
Group name | Field name | Symbol | Value
Product | Product ID | Select filter type...
Product | Product Name | Select filter type...
Product | Category | Select filter type...
Product | Unit price | Select filter type...
Product | On Stock | Select filter type...
Product | On Order | Select filter type...
Order | Order ID | Select filter type...
Order | Order Date | Select filter type...
Order | Year | Select filter type...
Order | Month | Select filter type...
Order | Week | Select filter type...
Order | Time to Ship | > | 5
Order | Freight | Select filter type...
Order | City | Select filter type...
Order | Country | Select filter type...
Orders Details | Total | Select filter type...
Orders Details | Discount | Select filter type...
Shipper | Company | Select filter type...
Employee | First name | Select filter type...
Employee | Last name | Select filter type...
Employee | Title | Select filter type...
Manager | First Name | Select filter type...
Manager | Last Name | Select filter type...
Customer | Company | Select filter type...
Customer | City | Select filter type...
Customer | Country | like | 'A%'
Supplier | Company | Select filter type...
Supplier | City | Select filter type...
Supplier | Country | Select filter type...
Select filter type...
=
<
>
<>
<=
>=
Done
Local intranet

Order.Year	Shipper.Company	Customer.Country	SUM(Orders Details.Total)
1996	Federal Shipping	Austria	14.646,40
		SUM	14.646,40
	Speedy Express	Austria	7.696,20
		SUM	7.696,20
	SUM		22.342,60
1997	Federal Shipping	Austria	2.453,75
		SUM	2.453,75
	Speedy Express	Argentina	110,00
		Austria	9.073,03
		SUM	9.183,03
	United Package	Argentina	443,40
		Austria	17.125,40
		SUM	17.568,80
	SUM		29.205,58
1998	Federal Shipping	Argentina	477,00
		Austria	6.339,00
		SUM	6.816,00
	Speedy Express	Argentina	1.082,00
		SUM	1.082,00
	United Package	Argentina	3.717,70
		Austria	10.240,00
		SUM	13.957,70
	SUM		21.855,70



Order.Year	Customer.Country	Shipper.Company	SUM(Orders Details.Total)
1996	Austria	Federal Shipping	14.646,40
		Speedy Express	7.696,20
		SUM	22.342,60
	SUM		22.342,60
1997	Argentina	Speedy Express	110,00
		United Package	443,40
		SUM	553,40
	Austria	Federal Shipping	2.453,75
		Speedy Express	9.073,03
		United Package	17.125,40
		SUM	28.652,18
	SUM		29.205,58
1998	Argentina	Federal Shipping	477,00
		Speedy Express	1.082,00
		United Package	3.717,70
		SUM	5.276,70
	Austria	Federal Shipping	6.339,00
		United Package	10.240,00
		SUM	16.579,00
	SUM		21.855,70

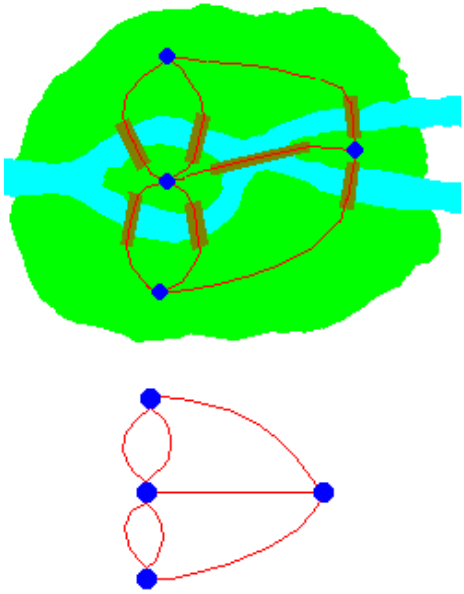

eFuture - Ad hoc Query System - Microsoft Internet Explorer provided by Novo Nordisk IT A/S
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebFuture2/FutureCube.aspx?d0=Order.Year&d1=Customer.Country&h2=Shipper.Company&m=
Google Search Web 233 blocked
eFuture - Cube Analysis
novo nordisk
Conditions:
Order.Time to Ship > 5 AND Customer.Country like 'A%'
New Query
Order.Year
Customer.Country
SUM(Orders Details.Total)
1996
Austria 22.342,60
SUM 22.342,60
1997
Argentina 553,40
Austria 28.652,18
SUM 29.205,58
1998
Argentina 5.276,70
Austria 16.579,00
SUM 21.855,70
Local intranet


eFuture - Ad hoc Query System - Microsoft Internet Explorer provided by Novo Nordisk IT A/S
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebFuture2/FutureCube.aspx?d0=Order.Year&h1=Customer.Country&h2=Shipper.Company&m=
Google Search Web 233 blocked
eFuture - Cube Analysis
novo nordisk
Conditions:
Order.Time to Ship > 5 AND Customer.Country like 'A%'
New Query
Order.Year
SUM(Orders Details.Total)
1996 22.342,60
1997 29.205,58
1998 21.855,70
Done
Local intranet

10 Appendiks: Introduktion til Grafer og databaser

10.1 Grafteori

I 1736 introducerede den schweiziske matematiker Leonhard Euler begrebet grafteori i artiklen ”Solutio problematis ad geometriam situs pertinentis”[23]. I artiklen beskriver han, hvordan det er muligt at bevæge sig rundt i Königsberg (Kaliningrad) ved at krydse hver af de 7 broer over floden, Pregel, nøjagtigt en gang.



Figur 10-1 - Broerne i Königsburg

Siden dengang er grafteorien blevet udviklet betydeligt, og i dag anvendes grafteori indenfor mange forskellige områder. Eksempler på anvendelsen af grafteori kan være løsning af ligningssystemer eller planlægning af computernetværk, hvor grafen repræsenterer computere og kabler.

En graf G beskrives ved en endelig mængde *punkter* V og en endelig mængde *kanter* E. En kant, som forbinder to punkter p og q, siges at være incident med p og q. I det tilfælde, hvor p = q, kaldes k for en *sløjfe*. *Valensen* v(p) af et punkt p er antallet af kanter, som er incidente med p. En kant,

[23] ”Løsningen på et problem i forbindelse med beliggenhedsteori”

hvor rækkefølgen af dens endepunkter har betydning, kaldes for en orienteret kant. Hvis alle kanter i en graf er orienterede, kaldes den for en *orienteret* graf. Tilsvarende kaldes en graf, hvor ingen kanter er orienterede, for en *ikke-orienteret* graf. Et væsentligt begreb indenfor grafteori er isomorfi. To grafer kaldes *isomorfe*, såfremt deres repræsentation symboliserer den samme graf. (evt. figur)
To punkter i en graf, der er forbundet med mere end en kant, siges at være forbundet med en *multipel kant*. En graf G' = (V', E') er en *delgraf* af G = (V, E), hvis V' $\subseteq$ V og E' $\subseteq$ E. I nogle tilfælde er man interesseret i at modellere et problem, hvor kanterne i grafen f.eks. repræsenterer omkostninger, og man tildeler derfor en talværdi til hver kant. En sådan graf kaldes for en *vægtet graf*.

10.1.1 Klasser af grafer

En graf V kaldes en *vej*, når punkterne navngives p_1, p_2, ..., p_m så p_i og p_j er forbundet netop, når |i-j| = 1. Endvidere gælder det, at en vej ikke indeholder nogle multiple kanter. En *sammenhængende* graf er en graf, hvori der findes en vej fra et punkt til et hvert andet punkt. En sammenhængende graf, hvis punkter alle har valens 2, kaldes en *kreds*. En kant, som ikke er en del af en kreds, kaldes for en *bro*. Blandt de væsentligste klasser af grafer er træer. Et *træ* er en sammenhængende graf, som ikke indeholder nogen kredse.

Nogle af de typiske spørgsmål, som man støder på indenfor grafteorien, er f.eks. om der findes en vej imellem 2 punkter, eller om findes der en kreds i grafen. Man kan også undersøge om en graf er sammenhængende, hvilket gøres ved at finde et *udspændende træ* for grafen. Et udspændende træ for en graf G er en delgraf, som udgør et træ og indeholder alle punkter i G.

10.1.2 Repræsentation af grafer

Den nemmeste måde at forstå en graf på for den menneskelige opfattelsesevne er ved en figur. I tilfælde af, at vi ønsker at arbejde med grafer på en computer, har vi brug at repræsentere grafer på anden vis. Dette kan f.eks. gøres ved en incidensmatrix, der for orienteret graf uden sløjfer med P punkter og K kanter, består af P x K matrix A = (a_{rs}), der er defineret ved, at

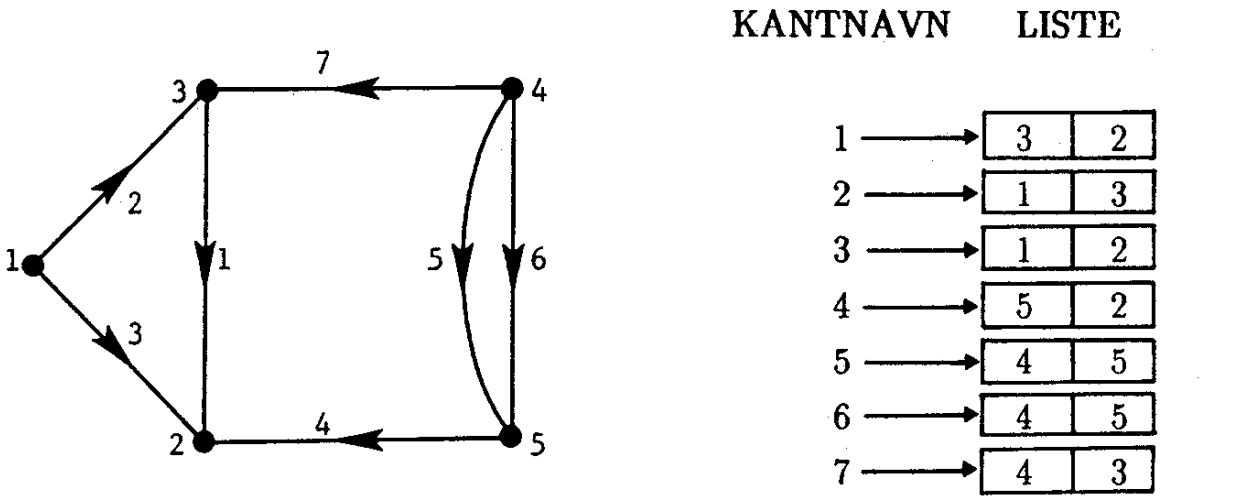
$$a_{rs} = \begin{cases} 1 \text{ når } k_s \text{ begynder i } p_r \\ -1 \text{ når } k_s \text{ slutter i } p_r \\ 0 \text{ ellers} \end{cases}$$

På figur X er vist et eksempel på en incidensmatrix, og det fremgår, at hver søjle i matricen indeholder netop et 1 og netop et -1. Når man skal lagre en incidensmatrix i computerens hukommelse, har man brug for at lagre P · K tal, hvilket ikke er særligt effektivt, eftersom vi ved, at P-2 tal i hver søjle indeholder 0'er og altså ingen nyttige informationer. Dette er grunden til, at incidensmatricer sjældent bruges til at repræsentere grafer i computersammenhæng. En graf kan repræsenteres ved 2K tal, idet man gemmer begyndelsespunkt og endepunkt for hver kant. De to repræsentationstyper, som jeg nu vil beskrive opfylder netop dette pladskrav.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \end{bmatrix}$$

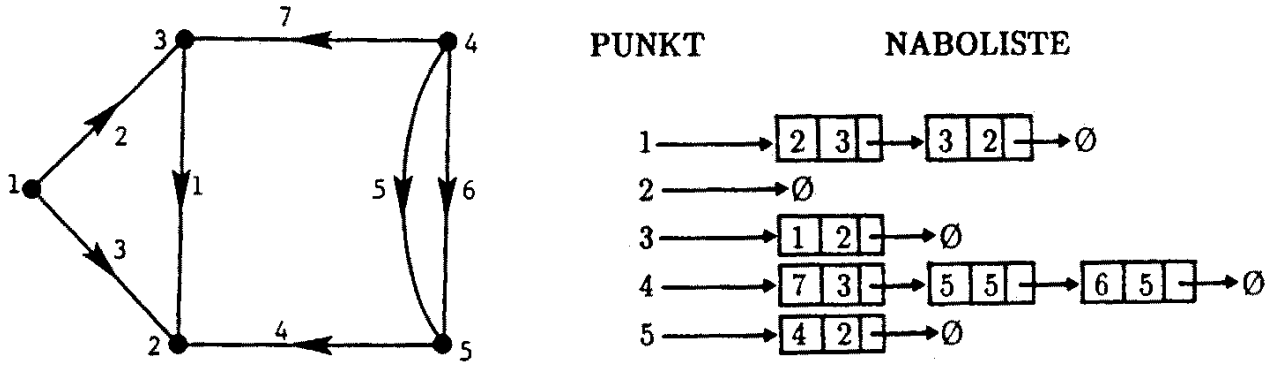
Figur 10-2 - Incidensmatrix og graf

I stedet for at anvende vektorer/matricer kan man anvende hægtede lister. Liste datastrukturen består af et antal elementer, der hver især indeholder en vis information samt adressen på det næste element i listen. En kantlisterepræsentation af en graf består af en liste for hver kant, som indeholder kantens begyndelsespunkt og endepunkt. Denne repræsentation indeholder K lister og kan med fordel anvendes i tilfælde af, at man vil udføre ordre såsom at finde begyndelses- og endepunkter for en kant eller sletning af en kant.



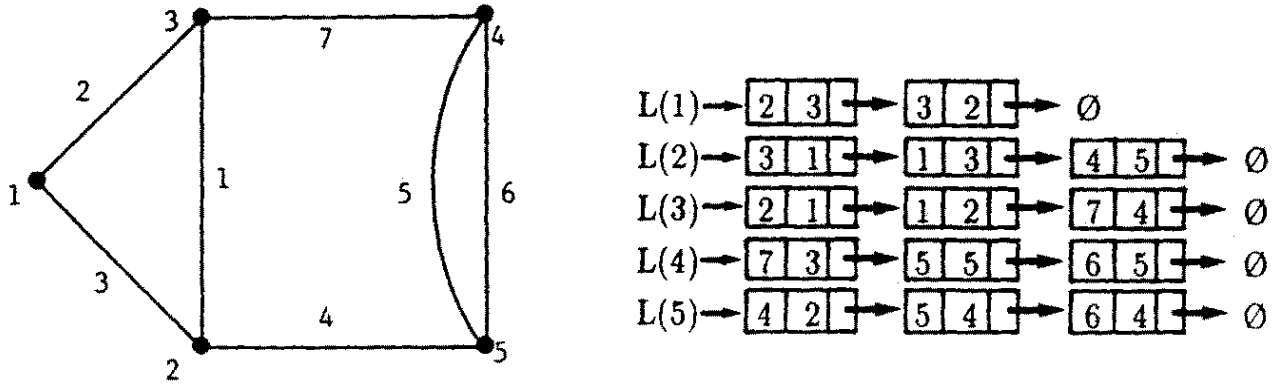
Figur 10-3 - Repræsentation ved kantlister

En anden måde at repræsentere en graf på er vha. nabolister og består P lister. Hvert element i listen for et punkt p_i indeholder kanten k_j, endepunktet q_i og adressen på det næste element i listen. Nabolisterepræsentationen er effektiv, hvis man ønsker at undersøge hvilke kanter, der starter i p_i.



Figur 10-4 - Repræsentation ved nabolister

Vi har nu set, hvordan kantlister og nabolister anvendes til at repræsentere orienterede grafer, men i nogle tilfælde kan man have brug for at repræsentere ikke-orienterede grafer. I dette tilfælde optræder en kant mellem p og q to gange i nabolisten, som det fremgår af figur X. Dette giver mening, eftersom en ikke-orienteret graf blot svarer til en orienteret graf, hvor der for hver kant i den ikke-orienterede graf eksisterer to modsat rettede kanter i den orienterede kant.



Figur 10-5 - Repræsentation af ikke-orienterede grafer

10.1.3 Dijkstras algoritme

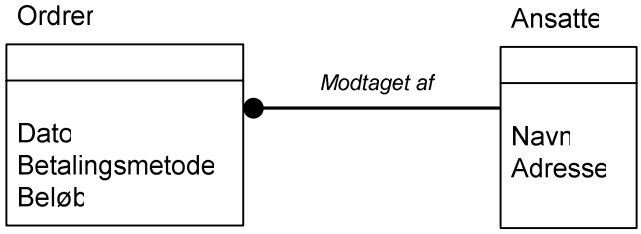
Denne algoritme blev udviklet af E.W. Dijkstra og er en såkaldt grådig algoritme, fordi den i hvert skridt vælger, hvad der umiddelbart synes at være bedst og derved opnår et optimalt resultat. Dijkstras algoritme er særdeles populær i sammenhæng med datalogiske problemstillinger og er desuden forholdsvis enkel at implementere. Algoritmen gør det, at den finder den korteste vej fra et punkt til *alle* øvrige punkter i sammenhængende graf med ikke-negative kantvægte. Resultatet af algoritmen er altså et udspændende træ for grafen.

For en graf **G = (V,E)**, hvori vi ønsker at finde den korteste vej fra punktet s, holder Dijkstras algoritme styr på to sæt af punkter:
S: indeholder de punkter, for hvilke den korteste vej fra source allerede er fundet.
T = V-S: indeholder de resterende punkter.
Desuden er der brug for følgende datastukturer:
d: et array af længden **P**, hvor værdien **d**(p_i) er længden af den korteste vej fra source til punktet p_i.
pred: et array af længden **P**, hvor værdien **pred**(p_i) er det foregående punkt i den korteste vej fra source til punktet p_i.

Algoritmens er nærmere beskrevet på s. 158 i *Grafteori – algoritmer og netværk* [Nie95].

10.2 Databaseteori

Den mest udbredte måde at beskrive et database design er ved hjælp af den såkaldte ”Entity-Relationship” (E/R) model. Modellen består tre hovedelementer: entitetsæt, attributter og relationer. En *entitet* kan betegnes som at være et abstrakt objekt, og et *entitetsæt* er en samling af entiteter med fælles egenskaber. Hvis man sammenligner med objekt-orienteret programmering, kan en entitet sammenlignes med et objekt, og entitetsættet med en klasse af objekter. Lad os f.eks. kigge på en database for en salgsafdeling og deres afgivne ordre. Her vil hver ordre udgøre en entitet og sættet bestående af alle ordrer vil være entitetsættet. Endvidere kan man forestille sig et entitetsæt, som indeholder en oversigt over de ansatte i afdelingen. Til hvert entitetsæt er der knyttet nogle *attributter*, hvilke svarer til egenskaberne for entiteterne i sættet. I eksemplet med salgsordrene kunne det tænkes, at ordre entitetsættet har attributterne: Dato - den dato, hvor ordren er modtaget. Betalingsmetode - kontant, kreditkort, osv. Beløb - den samlede pris på ordren. Endeligt er der *relationer*, som er forbindelser imellem to eller flere entitetsæt. Man kan f.eks. have en relation ”modtaget af” imellem Ordrer og Ansatte entitetsættene, som angiver navnet på medarbejderen, der har modtaget ordren.



Figur 10-6 - E/R diagram

Entiteter, attributter og relationer repræsenteres i et E/R diagram som vist på Figur 10-6. Pilen, som peger på Ansatte entiteten, angiver, at en ordre er modtaget af nøjagtig en medarbejder. Dette kaldes for en *Mange-1* relation fra Ordrer til Ansatte. Bemærk, at en entitet i Ansatte kan forbindes til mange entiteter i Ordrer. Tilsvarende har vi en *1-Mange* relation fra Ansatte til Ordrer. I tilfælde af, at der både eksisterer en *Mange-1* relation fra en entitet P til en entitet K og fra K til P, siger man, at vi har en *1-1* relation imellem P og K.

Databasens struktur eller *skema*, som det også kaldes, beskrives nemt i E/R modellen og oversættes efterfølgende til en relationsmodel, som danner grundlag for implementeringen og er tilpasset den

pågældende DBMS[24]. I relationsmodellen er entitetsæt og relationer fra E/R-diagrammet blevet oversat til tabeller. Ofte oversættes et entitetsæt direkte til en tabel, hvor navnene på søjlerne svarer til attributterne fra E/R-diagrammet, men dette gælder ikke altid. I Tabel 10-1 er der givet et eksempel på, hvordan E/R-diagrammet fra Figur 10-6 kunne oversættes. Hver række i Ordrer tabellen refererer til en ordre. Rækkerne kaldes også for *tupler*, og disse består af en komponent for hver søjle i tabellen.

Ordrer

ID	*Dato*	*Betalingsmetode*	*Beløb*	*Modtaget_af*
1	17-03-2003	Kontant	127,25	5232
2	18-03-2003	Dankort	550,00	5496
3	18-03-2003	Kontant	395,75	5232

Ansatte

ID	*Navn*	*Adresse*	*Afdeling*
5232	John Jensen	Skolevej 2	Gentofte
5496	Lis Petersen	Pile Alle 15	Frederiksberg

Tabel 10-1 - Tabeller i relationsmodel

Structured Query Language (SQL) er et standardsprog til at foretage forespørgsler og ændringer i databaser. Sproget er ganske let at lære og en god indgangsvinkel kan opnås på adressen: *http://www.w3schools.com/sql/default.asp.*

[24] Database Management System – f.eks. MS SQL Server 2000 eller Oracle

11 Appendiks: DTD definitioner

”FutureMeta.dtd” – DTD skema for metadata.

```
<!ELEMENT FutureMeta (MetaVersion, Options, Database, Relations, Structure)>
<!ELEMENT MetaVersion (#PCDATA)>
<!ELEMENT Options (DateDiffStyle, DecimalSeparatorFlag, EmptyStringIncludesNullString)>
<!ELEMENT DateDiffStyle (#PCDATA)>
<!ELEMENT DecimalSeparatorFlag (#PCDATA)>
<!ELEMENT EmptyStringIncludesNullString (#PCDATA)>
<!ELEMENT Database (Server, ServerVersion?, DisplayName, ConnectionString)>
<!ELEMENT Server (#PCDATA)>
<!ELEMENT ServerVersion (#PCDATA)>
<!ELEMENT DisplayName (#PCDATA)>
<!ELEMENT ConnectionString (#PCDATA)>
<!ELEMENT Relations (Relation*)>
<!ELEMENT Relation (Table1, Table2, JoinExpression)>
<!ATTLIST Relation
       desc CDATA ""
>
<!ELEMENT Table1 (#PCDATA)>
<!ELEMENT Table2 (#PCDATA)>
<!ELEMENT JoinExpression (#PCDATA)>
<!ELEMENT Structure (Group+)>
<!ELEMENT Group (Field+)>
<!ATTLIST Group
       name CDATA #REQUIRED
       hint CDATA ""
>
<!ELEMENT Field (Presentation, DBInfo)>
<!ATTLIST Field
       SemName CDATA #REQUIRED
>
<!ELEMENT Presentation (Hint?, Format?, MeasureType, DimensionType)>
<!ELEMENT Hint (#PCDATA)>
<!ELEMENT Format (#PCDATA)>
<!ELEMENT MeasureType (#PCDATA)>
<!ELEMENT DimensionType (#PCDATA)>
<!ELEMENT DBInfo (DBTable+, DBExpression, DBCondition?)>
<!ELEMENT DBTable (#PCDATA)>
<!ELEMENT DBExpression (#PCDATA)>
<!ATTLIST DBExpression
```

```
      type (string | int | real | date) #REQUIRED
>
<!ELEMENT DBCondition (#PCDATA)>
```

”FutureQuery.dtd” – DTD skema for en XML forespørgsel.

```
<!ELEMENT FutureQuery (UserField+)>
<!ELEMENT UserField (Expression, Included, SortOrder, Conditions)>
<!ELEMENT Expression (#PCDATA)>
<!ELEMENT Included (#PCDATA)>
<!ELEMENT SortOrder (#PCDATA)>
<!ELEMENT Conditions (Condition*)>
<!ELEMENT Condition (#PCDATA)>
```

12 Appendiks: Eksempel på XML metadata

“Northwind.xml”

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
	<MetaVersion>1</MetaVersion>
	<Options>
		<DateDiffStyle>1</DateDiffStyle>
		<DecimalSeparatorFlag>2</DecimalSeparatorFlag>
		<EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
	</Options>
	<Database>
		<Server>SQLServer</Server>
		<ServerVersion>7.0</ServerVersion>
		<DisplayName>Northwind</DisplayName>
		<ConnectionString>Initial Catalog=Northwind;Datasource=ol37144
		</ConnectionString>
	</Database>
	<Relations>
		<Relation desc="rel">
			<Table1>Employees emp</Table1>
			<Table2>Orders</Table2>
			<JoinExpression>emp.EmployeeID = Orders.EmployeeID</JoinExpression>
		</Relation>
		<Relation>
			<Table1>Orders</Table1>
			<Table2>Shippers</Table2>
			<JoinExpression>Orders.ShipVia = Shippers.ShipperID</JoinExpression>
		</Relation>
		<Relation desc="rel1">
			<Table1>Employees emp</Table1>
			<Table2>Employees man</Table2>
			<JoinExpression>man.Employeeid =* emp.ReportsTo</JoinExpression>
		</Relation>
		<Relation>
			<Table1>Orders</Table1>
			<Table2>Customers</Table2>
			<JoinExpression>Orders.CustomerID = Customers.CustomerID
			</JoinExpression>
		</Relation>
```

```
        <Relation>
            <Table1>Orders</Table1>
            <Table2>[Order Details]</Table2>
            <JoinExpression>Orders.OrderID = [Order Details].OrderID
            </JoinExpression>
        </Relation>
        <Relation>
            <Table1>[Order Details]</Table1>
            <Table2>Products</Table2>
            <JoinExpression>[Order Details].ProductID = Products.ProductID
            </JoinExpression>
        </Relation>
        <Relation>
            <Table1>Products</Table1>
            <Table2>Suppliers</Table2>
            <JoinExpression>Products.SupplierID = Suppliers.SupplierID
            </JoinExpression>
        </Relation>
        <Relation>
            <Table1>Products</Table1>
            <Table2>Categories</Table2>
            <JoinExpression>Products.CategoryID = Categories.CategoryID
            </JoinExpression>
        </Relation>
    </Relations>
    <Structure>
        <Group name="Product" hint="Product Hint">
            <Field SemName="Product ID">
                <Presentation>
                <MeasureType>1</MeasureType>
                <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Products</DBTable>
                    <DBExpression type="int">Products.ProductID
                    </DBExpression>
                </DBInfo>
            </Field>
            <Field SemName="Product Name">
                <Presentation>
                <MeasureType>1</MeasureType>
                <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
```

```
                <DBTable>Products</DBTable>
                <DBExpression type="string">Products.ProductName
                </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Category">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Categories</DBTable>
                <DBExpression type="string">Categories.CategoryName
                </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Unit size">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Products</DBTable>
                <DBExpression type="string">Products.QuantityPerUnit
                </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Unit price">
        <Presentation>
        <MeasureType>2</MeasureType>
        <DimensionType>0</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Products</DBTable>
                <DBExpression type="real">Products.UnitPrice
                </DBExpression>
        </DBInfo>
</Field>
<Field SemName="On Stock">
        <Presentation>
        <MeasureType>2</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
```

```
                        <DBTable>Products</DBTable>
                        <DBExpression type="int">Products.UnitsInStock
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="On Order">
                <Presentation>
                <MeasureType>2</MeasureType>
                <DimensionType>0</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Products</DBTable>
                        <DBExpression type="int">Products.UnitsOnOrder
                        </DBExpression>
                </DBInfo>
        </Field>
</Group>
<Group name="Order">
        <Field SemName="Order ID">
                <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="int">Orders.OrderID</DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Order Date">
                <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="date">Orders.OrderDate
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Year">
                <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
```

```
                <DBInfo>
                    <DBTable>Orders</DBTable>
                    <DBExpression type="int">DATEPART(year, Orders.OrderDate)
                    </DBExpression>
                </DBInfo>
            </Field>
            <Field SemName="Month">
                <Presentation>
                    <MeasureType>0</MeasureType>
                    <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Orders</DBTable>
                    <DBExpression type="int">DATEPART(month, Orders.OrderDate)
                    </DBExpression>
                </DBInfo>
            </Field>
            <Field SemName="Week">
                <Presentation>
                    <MeasureType>0</MeasureType>
                    <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Orders</DBTable>
                    <DBExpression type="int">DATEPART(wk, Orders.OrderDate)
                    </DBExpression>
                </DBInfo>
            </Field>
            <Field SemName="Time to Ship" hint="The number of days from the order
was placed to the order was shipped">
                <Presentation>
                    <MeasureType>2</MeasureType>
                    <DimensionType>0</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Orders</DBTable>
                    <DBExpression type="int">DATEDIFF([Day], OrderDate, ShippedDate)
                    </DBExpression>
                </DBInfo>
            </Field>
            <Field SemName="Freight">
                <Presentation>
                    <MeasureType>2</MeasureType>
                    <DimensionType>0</DimensionType>
```

```
                </Presentation>
                <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="real">Orders.Freight</DBExpression>
                </DBInfo>
        </Field>
</Group>
<Group name="Ship To" hint="ShipTo Hint">
        <Field SemName="Name">
                <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="string">Orders.ShipName
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Address">
                <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="string">Orders.ShipAddress
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="City">
                <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="string">Orders.ShipCity
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Region">
                <Presentation>
                        <MeasureType>1</MeasureType>
```

```
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="string">Orders.ShipRegion
                        </DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Postal Code">
                    <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="string">Orders.ShipPostalCode
                        </DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Country">
                    <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>Orders</DBTable>
                        <DBExpression type="string">Orders.ShipCountry
                        </DBExpression>
                    </DBInfo>
                </Field>
            </Group>
            <Group name="Orders Details">
                <Field SemName="Order ID">
                    <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>[Order Details]</DBTable>
                        <DBExpression type="int">[Order Details].OrderID
                        </DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Unit Price">
```

```
                        <Presentation>
                                <MeasureType>2</MeasureType>
                                <DimensionType>0</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>[Order Details]</DBTable>
                                <DBExpression type="real">[Order Details].UnitPrice
                                </DBExpression>
                        </DBInfo>
                </Field>
                <Field SemName="Quantity">
                        <Presentation>
                                <MeasureType>2</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>[Order Details]</DBTable>
                                <DBExpression type="int">[Order Details].Quantity
                                </DBExpression>
                        </DBInfo>
                </Field>
                <Field SemName="Total">
                        <Presentation>
                        <MeasureType>2</MeasureType>
                        <DimensionType>0</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>[Order Details]</DBTable>
                                <DBExpression  type="real">[Order  Details].UnitPrice  *
[Order Details].Quantity
                                </DBExpression>
                        </DBInfo>
                </Field>
                <Field SemName="Discount">
                        <Presentation>
                        <MeasureType>2</MeasureType>
                        <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>[Order Details]</DBTable>
                                <DBExpression type="real">[Order Details].Discount
                                </DBExpression>
                        </DBInfo>
                </Field>
```

```
</Group>
<Group name="Shipper">
	<Field SemName="Company">
		<Presentation>
		<MeasureType>1</MeasureType>
		<DimensionType>1</DimensionType>
		</Presentation>
		<DBInfo>
			<DBTable>Shippers</DBTable>
			<DBExpression type="string">Shippers.CompanyName
			</DBExpression>
		</DBInfo>
	</Field>
	<Field SemName="Phone">
		<Presentation>
		<MeasureType>0</MeasureType>
		<DimensionType>1</DimensionType>
		</Presentation>
		<DBInfo>
			<DBTable>Shippers</DBTable>
			<DBExpression type="string">Shippers.Phone
			</DBExpression>
		</DBInfo>
	</Field>
</Group>
<Group name="Employee" hint="Employees">
	<Field SemName="First name">
		<Presentation>
		<MeasureType>1</MeasureType>
		<DimensionType>1</DimensionType>
		</Presentation>
		<DBInfo>
			<DBTable>Employees emp</DBTable>
			<DBExpression type="string">emp.FirstName
			</DBExpression>
		</DBInfo>
	</Field>
	<Field SemName="Last name">
		<Presentation>
		<MeasureType>1</MeasureType>
		<DimensionType>1</DimensionType>
		</Presentation>
		<DBInfo>
			<DBTable>Employees emp</DBTable>
```

```
                <DBExpression type="string">emp.LastName</DBExpression>
        </DBInfo>
</Field>
<Field SemName="Birth Date">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Employees emp</DBTable>
                <DBExpression type="date">emp.BirthDate</DBExpression>
        </DBInfo>
</Field>
<Field SemName="Title">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Employees emp</DBTable>
                <DBExpression type="string">emp.Title</DBExpression>
        </DBInfo>
</Field>
<Field SemName="Hired Date">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Employees emp</DBTable>
                <DBExpression type="date">emp.HireDate</DBExpression>
        </DBInfo>
</Field>
<Field SemName="Extension">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Employees emp</DBTable>
                <DBExpression type="string">emp.Extension
                </DBExpression>
        </DBInfo>
</Field>
```

```
</Group>
<Group name="Manager">
	<Field SemName="First Name">
		<Presentation>
		<MeasureType>1</MeasureType>
		<DimensionType>1</DimensionType>
		</Presentation>
		<DBInfo>
			<DBTable>Employees man</DBTable>
			<DBTable>Employees emp</DBTable>
			<DBExpression type="string">man.FirstName
			</DBExpression>
		</DBInfo>
	</Field>
	<Field SemName="Last Name">
		<Presentation>
		<MeasureType>1</MeasureType>
		<DimensionType>1</DimensionType>
		</Presentation>
		<DBInfo>
			<DBTable>Employees man</DBTable>
			<DBTable>Employees emp</DBTable>
			<DBExpression type="string">man.LastName</DBExpression>
		</DBInfo>
	</Field>
</Group>
<Group name="Customer">
	<Field SemName="Company">
		<Presentation>
		<MeasureType>1</MeasureType>
		<DimensionType>1</DimensionType>
		</Presentation>
		<DBInfo>
			<DBTable>Customers</DBTable>
			<DBExpression type="string">Customers.CompanyName
			</DBExpression>
		</DBInfo>
	</Field>
	<Field SemName="Contact Name">
		<Presentation>
		<MeasureType>0</MeasureType>
		<DimensionType>1</DimensionType>
		</Presentation>
		<DBInfo>
```

```
                <DBTable>Customers</DBTable>
                <DBExpression type="string">Customers.ContactName
                </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Address">
        <Presentation>
        <MeasureType>0</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Customers</DBTable>
                <DBExpression type="string">Customers.Address
                </DBExpression>
        </DBInfo>
</Field>
<Field SemName="City">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Customers</DBTable>
                <DBExpression type="string">Customers.City
                </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Region">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
                <DBTable>Customers</DBTable>
                <DBExpression type="string">Customers.Region
                </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Postal Code">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
```

```
                        <DBTable>Customers</DBTable>
                        <DBExpression type="string">Customers.PostalCode
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Country">
                <Presentation>
                <MeasureType>1</MeasureType>
                <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Customers</DBTable>
                        <DBExpression type="string">Customers.Country
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Phone">
                <Presentation>
                <MeasureType>0</MeasureType>
                <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Customers</DBTable>
                        <DBExpression type="string">Customers.Phone
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Fax">
                <Presentation>
                <MeasureType>0</MeasureType>
                <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>Customers</DBTable>
                        <DBExpression type="string">Customers.Fax
                        </DBExpression>
                </DBInfo>
        </Field>
</Group>
<Group name="Supplier" hint="Supplier Hint">
        <Field SemName="Company">
                <Presentation>
                <MeasureType>1</MeasureType>
                <DimensionType>1</DimensionType>
```

```
        </Presentation>
        <DBInfo>
            <DBTable>Suppliers</DBTable>
            <DBExpression type="string">Suppliers.CompanyName
            </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Contact Name">
        <Presentation>
        <MeasureType>0</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
            <DBTable>Suppliers</DBTable>
            <DBExpression type="string">Suppliers.ContactName
            </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Address">
        <Presentation>
        <MeasureType>0</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
            <DBTable>Suppliers</DBTable>
            <DBExpression type="string">Suppliers.Address
            </DBExpression>
        </DBInfo>
</Field>
<Field SemName="City">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
            <DBTable>Suppliers</DBTable>
            <DBExpression type="string">Suppliers.City
            </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Region">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
```

```
        </Presentation>
        <DBInfo>
            <DBTable>Suppliers</DBTable>
            <DBExpression type="string">Suppliers.Region
            </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Postal Code">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
            <DBTable>Suppliers</DBTable>
            <DBExpression type="string">Suppliers.PostalCode
            </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Country">
        <Presentation>
        <MeasureType>1</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
            <DBTable>Suppliers</DBTable>
            <DBExpression type="string">Suppliers.Country
            </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Phone">
        <Presentation>
        <MeasureType>0</MeasureType>
        <DimensionType>1</DimensionType>
        </Presentation>
        <DBInfo>
            <DBTable>Suppliers</DBTable>
            <DBExpression type="string">Suppliers.Phone
            </DBExpression>
        </DBInfo>
</Field>
<Field SemName="Fax">
        <Presentation>
        <MeasureType>0</MeasureType>
        <DimensionType>1</DimensionType>
```

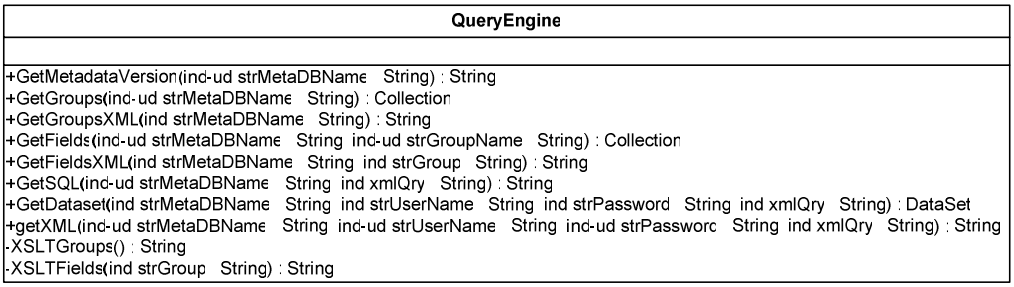
```
                        </Presentation>
                        <DBInfo>
                              <DBTable>Suppliers</DBTable>
                              <DBExpression type="string">Suppliers.Fax
                              </DBExpression>
                        </DBInfo>
                  </Field>
                  <Field SemName="WWW Page">
                        <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                              <DBTable>Suppliers</DBTable>
                              <DBExpression type="string">Suppliers.HomePage
                              </DBExpression>
                        </DBInfo>
                  </Field>
            </Group>
      </Structure>
</FutureMeta>
```

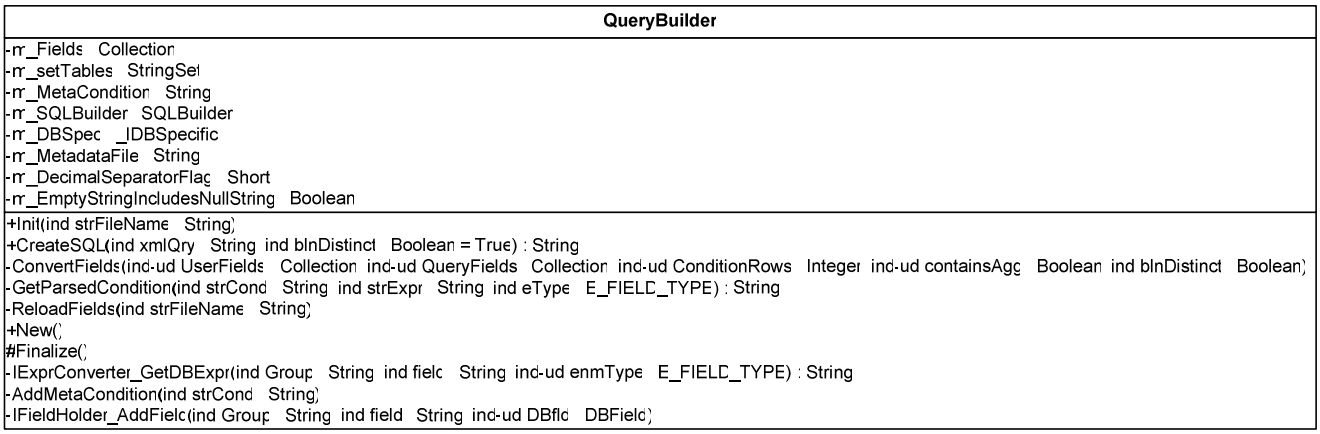
13 Appendiks: Klassediagrammer

13.1 eFuture Motor

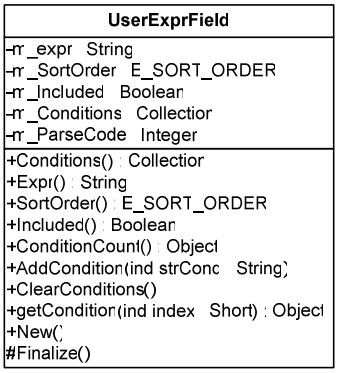
Klassediagrammerne for eFuture motoren er inddelt efter pakkeoversigten fra Figur 4-2.

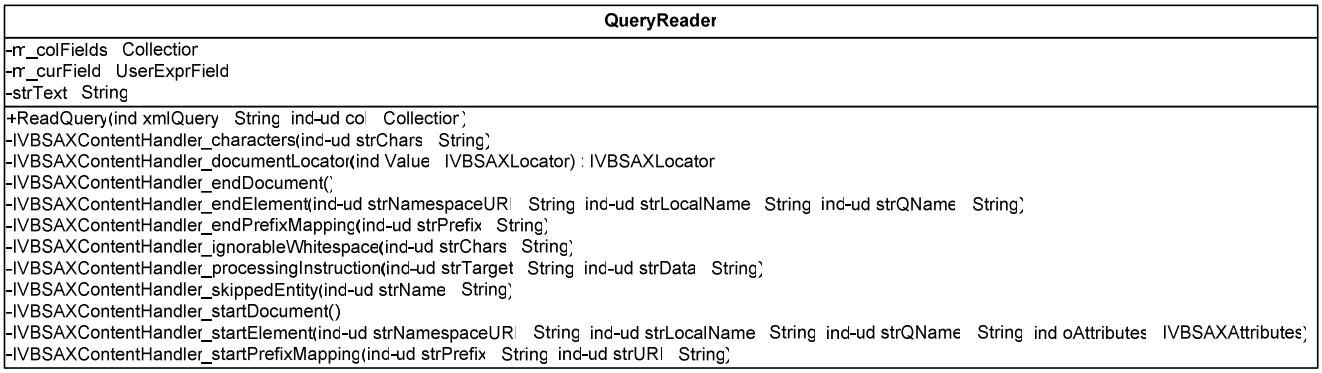
Public Classes



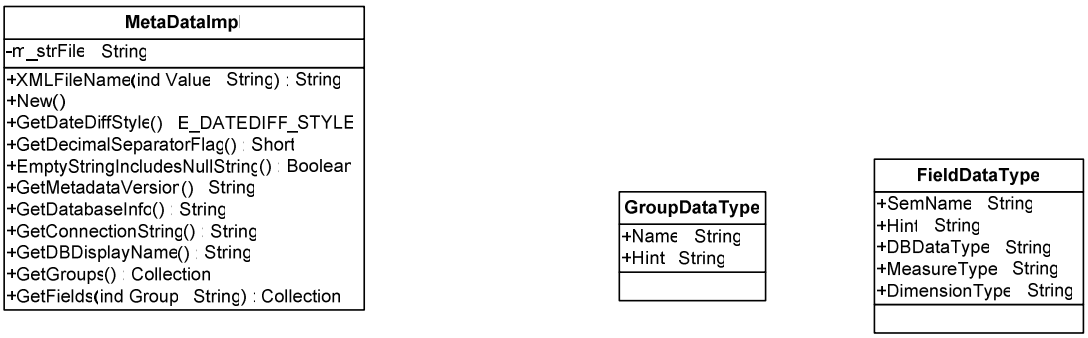


XML Query Handling





Matadata Handling



GetMetaFields
-Group String -CurrGroup String -CurrTAG String -Fields Collection -CurrField FieldDataType -strText String
+transferFields(ind-ud GroupFields Collection ind-ud FindGroup String) -IVBSAXContentHandler_startElement(ind-ud strNamespaceURI String ind-ud strLocalName String ind-ud strQName String ind attributes IVBSAXAttributes) -IVBSAXContentHandler_endElement(ind-ud strNamespaceURI String ind-ud strLocalName String ind-ud strQName String) -IVBSAXContentHandler_characters(ind-ud text String) -IVBSAXContentHandler_documentLocator(ind Value IVBSAXLocator) : IVBSAXLocator -IVBSAXContentHandler_endDocument() -IVBSAXContentHandler_endPrefixMapping(ind-ud strPrefix String) -IVBSAXContentHandler_ignorableWhitespace(ind-ud strChars String) -IVBSAXContentHandler_processingInstruction(ind-ud target String ind-ud data String) -IVBSAXContentHandler_skippedEntity(ind-ud strName String) -IVBSAXContentHandler_startDocument() -IVBSAXContentHandler_startPrefixMapping(ind-ud strPrefix String ind-ud strURI String)

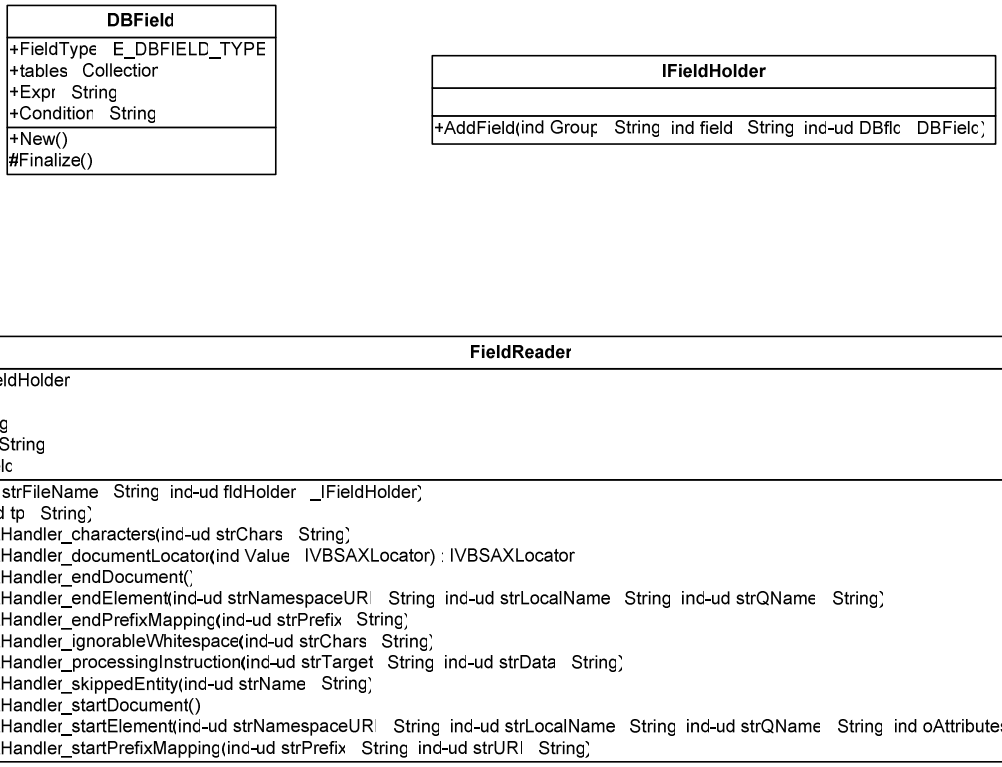
GetMetaGroups
-Groups Collectior
+transferFields(ind-ud Group Collection) -IVBSAXContentHandler_startElement(ind-ud strNamespaceURI String ind-ud strLocalName String ind-ud strQName String ind attributes IVBSAXAttributes) -IVBSAXContentHandler_endElement(ind-ud strNamespaceURI String ind-ud strLocalName String ind-ud strQName String) -IVBSAXContentHandler_characters(ind-ud text String) -IVBSAXContentHandler_documentLocator(ind Value IVBSAXLocator) : IVBSAXLocator -IVBSAXContentHandler_endDocument() -IVBSAXContentHandler_endPrefixMapping(ind-ud strPrefix String) -IVBSAXContentHandler_ignorableWhitespace(ind-ud strChars String) -IVBSAXContentHandler_processingInstruction(ind-ud target String ind-ud data String) -IVBSAXContentHandler_skippedEntity(ind-ud strName String) -IVBSAXContentHandler_startDocument() -IVBSAXContentHandler_startPrefixMapping(ind-ud strPrefix String ind-ud strURI String)

Matadata Handling - fortsat

ConnectionStringReader
-CON_TAG : String = 'ConnectionString' -DISPNAME_TAG : String = 'DisplayName' -SERVER_TAG : String = 'Server' -META_VERS_TAG : String = 'MetaVersion' -DB_TAG : String = 'Database' -m_strCon : String -m_strDispName : String -m_strServerName : String -m_strMetaVersion : String -strText : String
+ConnectionString() : String +DisplayName() : String +ServerName() : String +MetadataVersion() : String -IVBSAXContentHandler_documentLocator(ind Value : IVBSAXLocator) : IVBSAXLocator -IVBSAXContentHandler_characters(ind-ud strChars : String) -IVBSAXContentHandler_endDocument() -IVBSAXContentHandler_endElement(ind-ud strNamespaceURI : String, ind-ud strLocalName : String, ind-ud strQName : String) -IVBSAXContentHandler_endPrefixMapping(ind-ud strPrefix : String) -IVBSAXContentHandler_ignorableWhitespace(ind-ud strChars : String) -IVBSAXContentHandler_processingInstruction(ind-ud strTarget : String, ind-ud strData : String) -IVBSAXContentHandler_skippedEntity(ind-ud strName : String) -IVBSAXContentHandler_startDocument() -IVBSAXContentHandler_startElement(ind-ud strNamespaceURI : String, ind-ud strLocalName : String, ind-ud strQName : String, ind oAttributes : IVBSAXAttributes) -IVBSAXContentHandler_startPrefixMapping(ind-ud strPrefix : String, ind-ud strURI : String)

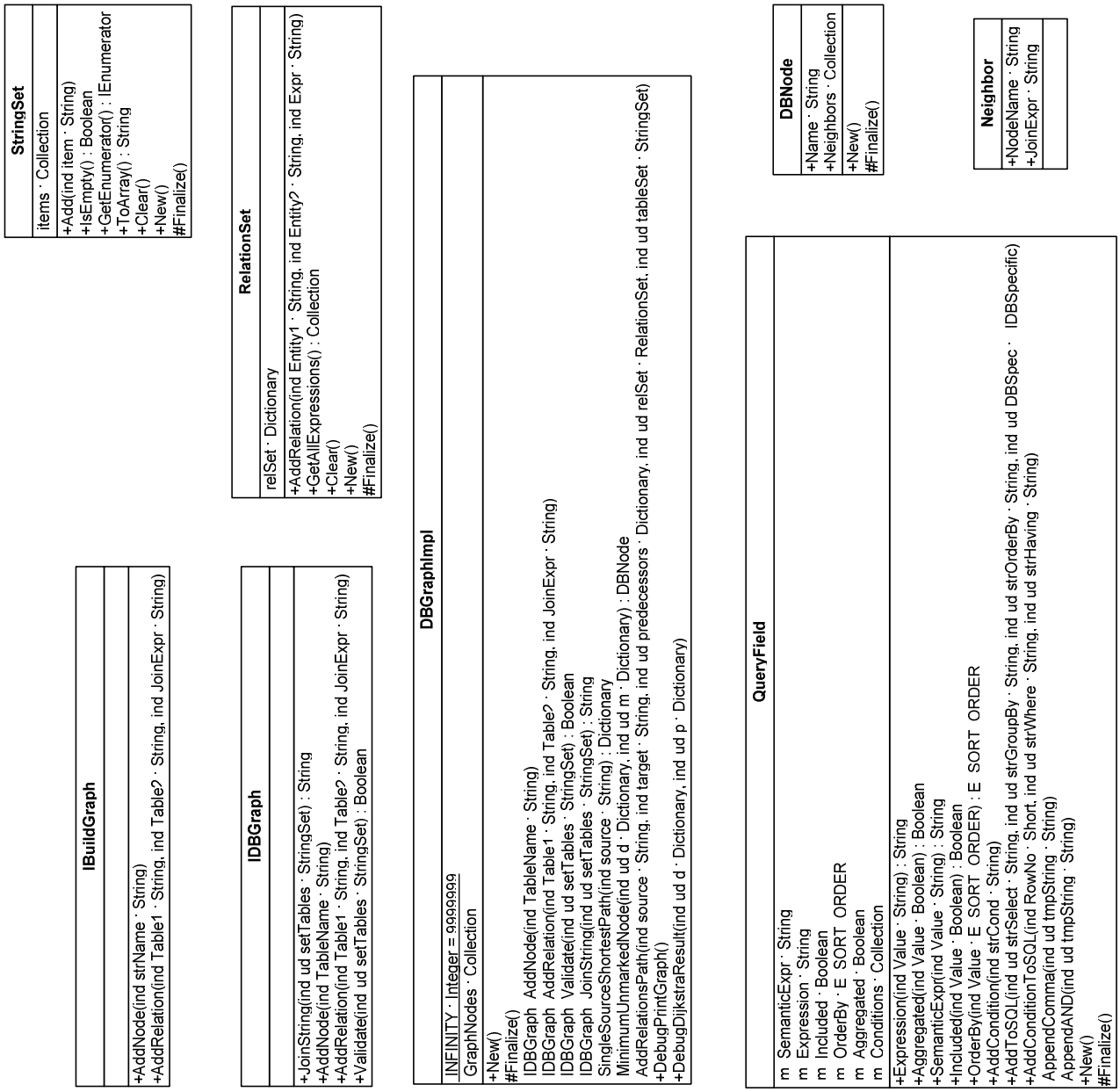
OptionsReader
-DATEDIFF_STYLE_TAG : String = 'DateDiffStyle' -SEPARATOR_FLAG_TAG : String = 'DecimalSeparatorFlag' -NULL_STRING_TAG : String = 'EmptyStringIncludesNullString' -OPTIONS_TAG : String = 'Options' -m_strDateDiffStyle : String -m_strSeparatorFlag : String -m_strEmptyNullStringFlag : String -strText : String
+DateDiffStyle() : String +SeparatorFlag() : String +EmptyNullStringFlag() : String -IVBSAXContentHandler_documentLocator(ind Value : IVBSAXLocator) : IVBSAXLocator -IVBSAXContentHandler_characters(ind-ud strChars : String) -IVBSAXContentHandler_endDocument() -IVBSAXContentHandler_endElement(ind-ud strNamespaceURI : String, ind-ud strLocalName : String, ind-ud strQName : String) -IVBSAXContentHandler_endPrefixMapping(ind-ud strPrefix : String) -IVBSAXContentHandler_ignorableWhitespace(ind-ud strChars : String) -IVBSAXContentHandler_processingInstruction(ind-ud strTarget : String, ind-ud strData : String) -IVBSAXContentHandler_skippedEntity(ind-ud strName : String) -IVBSAXContentHandler_startDocument() -IVBSAXContentHandler_startElement(ind-ud strNamespaceURI : String, ind-ud strLocalName : String, ind-ud strQName : String, ind oAttributes : IVBSAXAttributes) -IVBSAXContentHandler_startPrefixMapping(ind-ud strPrefix : String, ind-ud strURI : String)

DB Field Handling

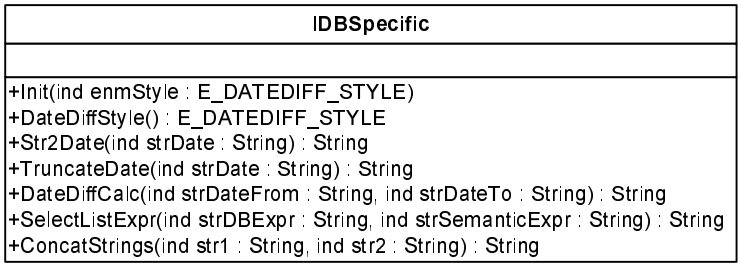


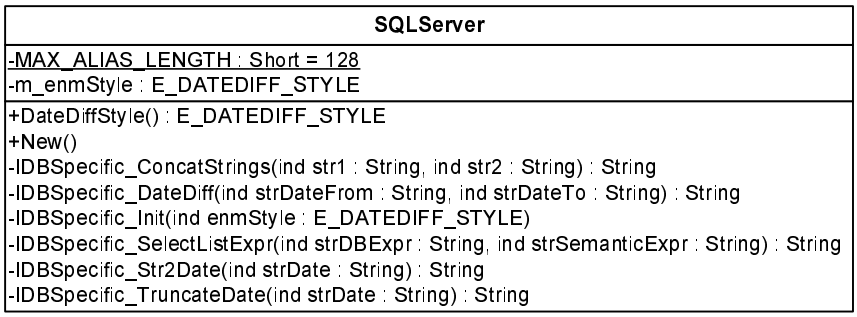
SQL Construction

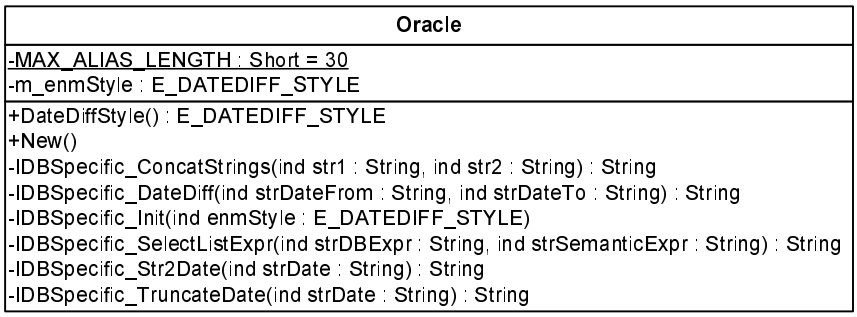
SQL Construction – fortsat



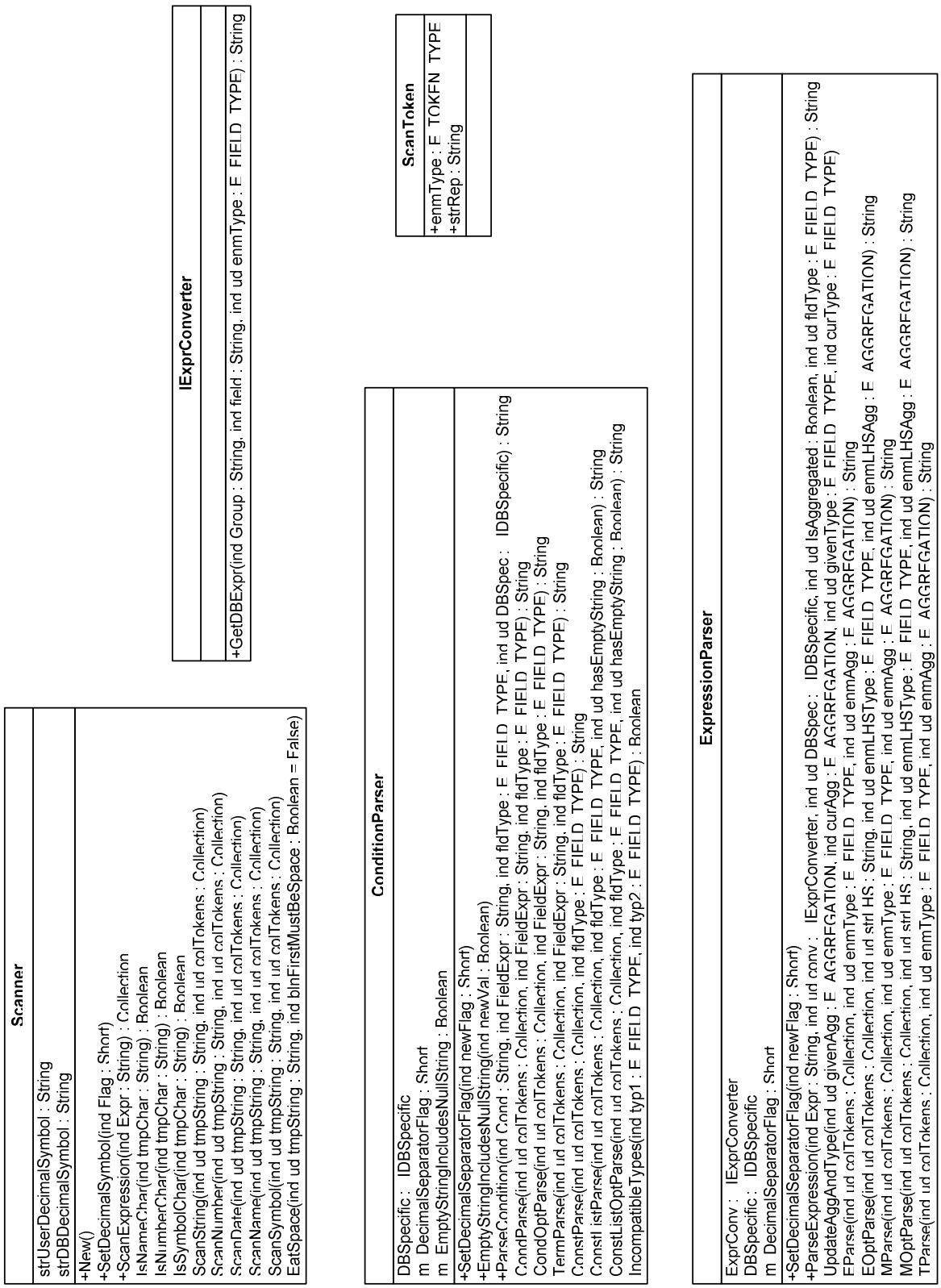
DB Specification



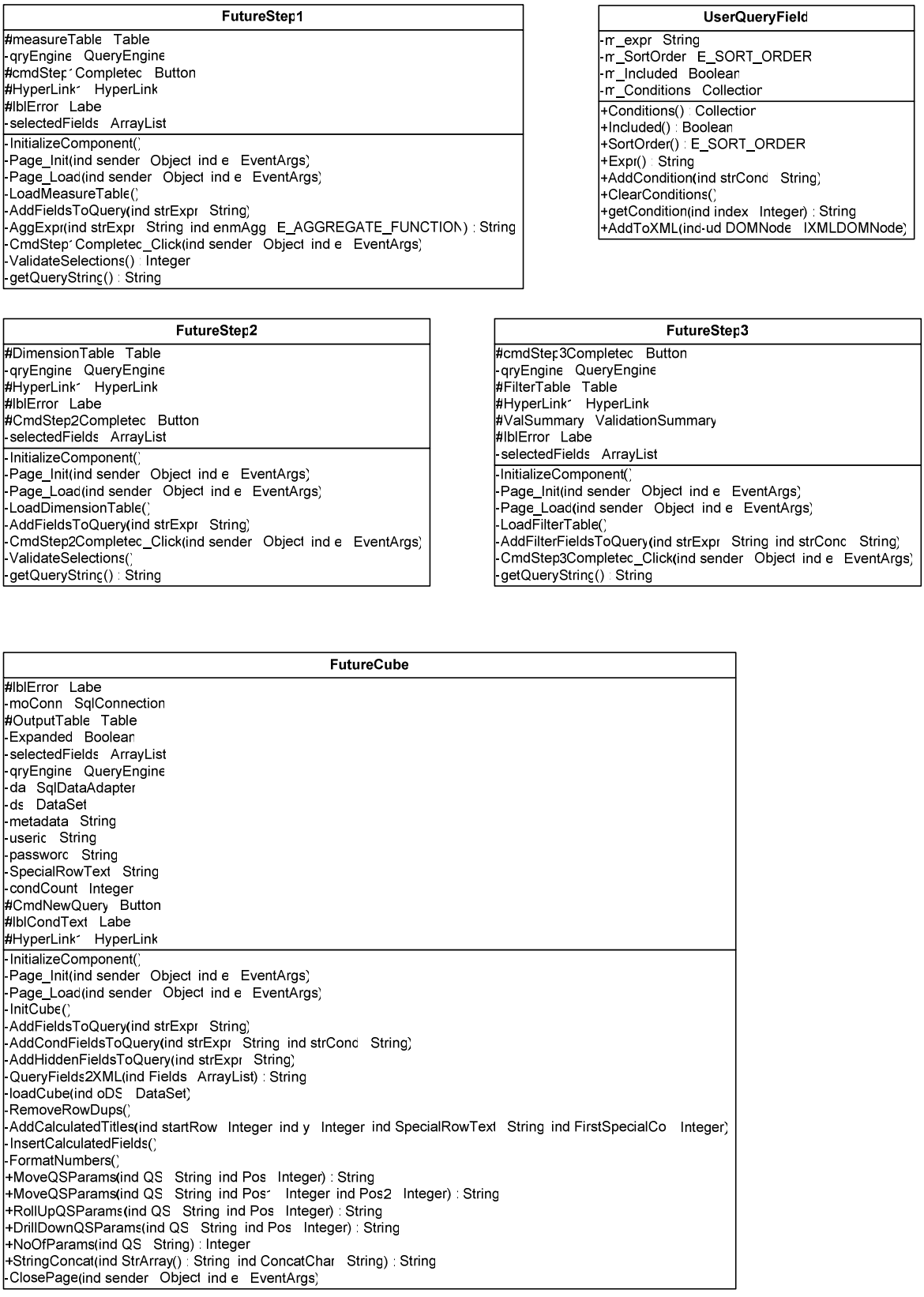




Parser/Scanner



13.2 UI Input og UI Output



14 Appendiks: Testdokumentation

Modultest

Her følger metafilerne, som benyttes i modultesten.

”Meta0401.xml”

```
<?xml version="1.0" ?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
	<MetaVersion>1</MetaVersion>
	<Options>
		<DateDiffStyle>1</DateDiffStyle>
		<DecimalSeparatorFlag>2</DecimalSeparatorFlag>
		<EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
	</Options>
	<Database>
		<Server>SQLServer</Server>
		<ServerVersion>7.0</ServerVersion>
		<DisplayName>Northwind</DisplayName>
		<ConnectionString>InitialCatalog=Northwind;Datasource=ol37144
		</ConnectionString>
	</Database>
	<Relations>
		<Relation>
			<Table1>Products</Table1>
			<Table2>Suppliers</Table2>
			<JoinExpression>Products.SupplierID = Suppliers.SupplierID
			</JoinExpression>
		</Relation>
	</Relations>
	<Structure>
		<Group name="Product" hint="Product Hint">
			<Field SemName="Product ID">
				<Presentation>
					<MeasureType>1</MeasureType>
					<DimensionType>1</DimensionType>
				</Presentation>
				<DBInfo>
					<DBTable>Products</DBTable>
					<DBExpression type="int">Products.ProductID
					</DBExpression>
				</DBInfo>
			</Field>
			<Field SemName="Product Name">
```

```
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>Products</DBTable>
                                <DBExpression type="string">Products.ProductName
                                </DBExpression>
                        </DBInfo>
                </Field>
        </Group>
        <Group name="Supplier" hint="Supplier Hint">
                <Field SemName="Company">
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>Suppliers</DBTable>
                                <DBExpression type="string">Suppliers.CompanyName
                                </DBExpression>
                        </DBInfo>
                </Field>
        </Group>
    </Structure>
</FutureMeta>
```

”Meta0402.xml”

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
    <MetaVersion>1</MetaVersion>
    <Options>
        <DateDiffStyle>1</DateDiffStyle>
        <DecimalSeparatorFlag>error</DecimalSeparatorFlag>
        <EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
    </Options>
    <Database>
        <Server>SQLServer</Server>
        <ServerVersion>7.0</ServerVersion>
        <DisplayName>Northwind</DisplayName>
        <ConnectionString>Initial Catalog=Northwind;Datasource=ol37144
        </ConnectionString>
```

```
</Database>
<Relations>
	<Relation>
		<Table1>Products</Table1>
		<Table2>Suppliers</Table2>
		<JoinExpression>Products.SupplierID = Suppliers.SupplierID
		</JoinExpression>
	</Relation>
</Relations>
<Structure>
	<Group name="Product" hint="Product Hint">
		<Field SemName="Product ID">
			<Presentation>
				<MeasureType>1</MeasureType>
				<DimensionType>1</DimensionType>
			</Presentation>
			<DBInfo>
				<DBTable>Products</DBTable>
				<DBExpression type="int">Products.ProductID
				</DBExpression>
			</DBInfo>
		</Field>
		<Field SemName="Product Name">
			<Presentation>
				<MeasureType>1</MeasureType>
				<DimensionType>1</DimensionType>
			</Presentation>
			<DBInfo>
				<DBTable>Products</DBTable>
				<DBExpression type="string">Products.ProductName
				</DBExpression>
			</DBInfo>
		</Field>
	</Group>
	<Group name="Supplier" hint="Supplier Hint">
		<Field SemName="Company">
			<Presentation>
				<MeasureType>1</MeasureType>
				<DimensionType>1</DimensionType>
			</Presentation>
			<DBInfo>
				<DBTable>Suppliers</DBTable>
				<DBExpression type="string">Suppliers.CompanyName
				</DBExpression>
```

```
                        </DBInfo>
                    </Field>
                </Group>
        </Structure>
</FutureMeta>
```

”Meta0403.xml”

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
        <MetaVersion>1</MetaVersion>
        <Options>
                <DateDiffStyle>error</DateDiffStyle>
                <DecimalSeparatorFlag>2</DecimalSeparatorFlag>
                <EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
        </Options>
        <Database>
                <Server>SQLServer</Server>
                <ServerVersion>7.0</ServerVersion>
                <DisplayName>Northwind</DisplayName>
                <ConnectionString>Initial Catalog=Northwind;Datasource=ol37144
                </ConnectionString>
        </Database>
        <Relations>
                <Relation>
                        <Table1>Products</Table1>
                        <Table2>Suppliers</Table2>
                        <JoinExpression>Products.SupplierID = Suppliers.SupplierID
                        </JoinExpression>
                </Relation>
        </Relations>
        <Structure>
                <Group name="Product" hint="Product Hint">
                        <Field SemName="Product ID">
                                <Presentation>
                                        <MeasureType>1</MeasureType>
                                        <DimensionType>1</DimensionType>
                                </Presentation>
                                <DBInfo>
                                        <DBTable>Products</DBTable>
                                        <DBExpression type="int">Products.ProductID
                                        </DBExpression>
                                </DBInfo>
```

```
                </Field>
                <Field SemName="Product Name">
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>Products</DBTable>
                                <DBExpression type="string">Products.ProductName
                                </DBExpression>
                        </DBInfo>
                </Field>
        </Group>
        <Group name="Supplier" hint="Supplier Hint">
                <Field SemName="Company">
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>Suppliers</DBTable>
                                <DBExpression type="string">Suppliers.CompanyName
                                </DBExpression>
                        </DBInfo>
                </Field>
        </Group>
    </Structure>
</FutureMeta>
```

”Meta0404.xml”

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
    <MetaVersion>1</MetaVersion>
    <Options>
        <DateDiffStyle>0</DateDiffStyle>
        <DecimalSeparatorFlag>2</DecimalSeparatorFlag>
        <EmptyStringIncludesNullString>falseerror</EmptyStringIncludesNullString>
    </Options>
    <Database>
        <Server>SQLServer</Server>
        <ServerVersion>7.0</ServerVersion>
        <DisplayName>Northwind</DisplayName>
```

```
        <ConnectionString>Initial Catalog=Northwind;Datasource=ol37144
        </ConnectionString>
    </Database>
    <Relations>
        <Relation>
            <Table1>Products</Table1>
            <Table2>Suppliers</Table2>
            <JoinExpression>Products.SupplierID = Suppliers.SupplierID
            </JoinExpression>
        </Relation>
    </Relations>
    <Structure>
        <Group name="Product" hint="Product Hint">
            <Field SemName="Product ID">
                <Presentation>
                    <MeasureType>1</MeasureType>
                    <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Products</DBTable>
                    <DBExpression type="int">Products.ProductID
                    </DBExpression>
                </DBInfo>
            </Field>
            <Field SemName="Product Name">
                <Presentation>
                    <MeasureType>1</MeasureType>
                    <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Products</DBTable>
                    <DBExpression type="string">Products.ProductName
                    </DBExpression>
                </DBInfo>
            </Field>
        </Group>
        <Group name="Supplier" hint="Supplier Hint">
            <Field SemName="Company">
                <Presentation>
                    <MeasureType>1</MeasureType>
                    <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Suppliers</DBTable>
```

```
                        <DBExpression type="string">Suppliers.CompanyName
                        </DBExpression>
                    </DBInfo>
                </Field>
            </Group>
        </Structure>
</FutureMeta>
```

”Meta0405.xml”

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
        <MetaVersion>1</MetaVersion>
        <Options>
            <DateDiffStyle>0</DateDiffStyle>
            <DecimalSeparatorFlag>2</DecimalSeparatorFlag>
            <EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
        </Options>
        <Database>
            <Server>SQLServer</Server>
            <ServerVersion>7.0</ServerVersion>
            <DisplayName>Northwind</DisplayName>
            <ConnectionString>Initial Catalog=Northwind;Datasource=ol37144
            </ConnectionString>
        </Database>
        <Relations>
            <Relation>
                <Table1>Products</Table1>
                <Table2>Suppliers</Table2>
                <JoinExpression>Products.SupplierID = Suppliers.SupplierID
                </JoinExpression>
            </Relation>
        </Relations>
        <Structure>
            <Group name="Product" hint="Product Hint">
                <Field SemName="Product ID">
                    <Presentation/>
                    <DBInfo>
                        <DBTable>ProductsError</DBTable>
                        <DBExpression type="int">Products.ProductID
                        </DBExpression>
                    </DBInfo>
                </Field>
```

```
                <Field SemName="Product Name">
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>Products</DBTable>
                                <DBExpression type="string">Products.ProductName
                                </DBExpression>
                        </DBInfo>
                </Field>
        </Group>
        <Group name="Supplier" hint="Supplier Hint">
                <Field SemName="Company">
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>Suppliers</DBTable>
                                <DBExpression type="string">Suppliers.CompanyName
                                </DBExpression>
                        </DBInfo>
                </Field>
        </Group>
    </Structure>
</FutureMeta>
```

”Meta0406.xml”

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
    <MetaVersion>1</MetaVersion>
    <Options>
        <DateDiffStyle>0</DateDiffStyle>
        <DecimalSeparatorFlag>2</DecimalSeparatorFlag>
        <EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
    </Options>
    <Database>
        <Server>SQLServer</Server>
        <ServerVersion>7.0</ServerVersion>
        <DisplayName>Northwind</DisplayName>
        <ConnectionString>Initial Catalog=Northwind;Datasource=ol37144
```

```
        </ConnectionString>
    </Database>
    <Relations>
        <Relation>
            <Table1>Products</Table1>
            <Table2>Suppliers</Table2>
            <JoinExpression>Products.SupplierID = Suppliers.SupplierID
            </JoinExpression>
        </Relation>
        <Relation>
            <Table1>Products1</Table1>
            <Table2>Suppliers1</Table2>
            <JoinExpression>Products.SupplierID = Suppliers.SupplierID
            </JoinExpression>
        </Relation>
    </Relations>
    <Structure>
        <Group name="Product" hint="Product Hint">
            <Field SemName="Product ID">
                <Presentation>
                    <MeasureType>1</MeasureType>
                    <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Products</DBTable>
                    <DBExpression type="int">Products.ProductID
                    </DBExpression>
                </DBInfo>
            </Field>
            <Field SemName="Product Name">
                <Presentation>
                    <MeasureType>1</MeasureType>
                    <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                    <DBTable>Products</DBTable>
                    <DBExpression type="string">Products.ProductName
                    </DBExpression>
                </DBInfo>
            </Field>
        </Group>
        <Group name="Supplier" hint="Supplier Hint">
            <Field SemName="Company">
                <Presentation>
```

```
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>Suppliers</DBTable>
                        <DBExpression type="string">Suppliers.CompanyName
                        </DBExpression>
                    </DBInfo>
                </Field>
            </Group>
        </Structure>
</FutureMeta>
```

”Meta0407.xml”

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
        <MetaVersion>1</MetaVersion>
        <Options>
            <DateDiffStyle>0</DateDiffStyle>
            <DecimalSeparatorFlag>2</DecimalSeparatorFlag>
            <EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
        </Options>
        <Database>
            <Server>SQLServer</Server>
            <ServerVersion>7.0</ServerVersion>
            <DisplayName>Northwind</DisplayName>
            <ConnectionString>Initial Catalog=Northwind;Datasource=ol37144
            </ConnectionString>
        </Database>
        <Relations>
            <Relation>
                <Table1>Products</Table1>
                <Table2>Suppliers</Table2>
                <JoinExpression>Products.SupplierID = Suppliers.SupplierID
                </JoinExpression>
            </Relation>
            <Relation>
                <Table1>Products</Table1>
                <Table2>Suppliers</Table2>
                <JoinExpression>Products.SupplierID = Suppliers.SupplierID
                </JoinExpression>
            </Relation>
```

```
	</Relations>
	<Structure>
		<Group name="Product" hint="Product Hint">
			<Field SemName="Product ID">
				<Presentation>
					<MeasureType>1</MeasureType>
					<DimensionType>1</DimensionType>
				</Presentation>
				<DBInfo>
					<DBTable>Products</DBTable>
					<DBExpression type="int">Products.ProductID
					</DBExpression>
				</DBInfo>
			</Field>
			<Field SemName="Product Name">
				<Presentation>
					<MeasureType>1</MeasureType>
					<DimensionType>1</DimensionType>
				</Presentation>
				<DBInfo>
					<DBTable>Products</DBTable>
					<DBExpression type="string">Products.ProductName
					</DBExpression>
				</DBInfo>
			</Field>
		</Group>
		<Group name="Supplier" hint="Supplier Hint">
			<Field SemName="Company">
				<Presentation>
					<MeasureType>1</MeasureType>
					<DimensionType>1</DimensionType>
				</Presentation>
				<DBInfo>
					<DBTable>Suppliers</DBTable>
					<DBExpression type="string">Suppliers.CompanyName
					</DBExpression>
				</DBInfo>
			</Field>
		</Group>
	</Structure>
</FutureMeta>
```

”Meta0408.xml”

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
	<MetaVersion>1</MetaVersion>
	<Options>
		<DateDiffStyle>0</DateDiffStyle>
		<DecimalSeparatorFlag>2</DecimalSeparatorFlag>
		<EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
	</Options>
	<Database>
		<Server>SQLServer</Server>
		<ServerVersion>7.0</ServerVersion>
		<DisplayName>Northwind</DisplayName>
		<ConnectionString>Initial Catalog=Northwind;Datasource=ol37144
		</ConnectionString>
	</Database>
	<Relations>
		<Relation>
			<Table1>Products</Table1>
			<Table2>Suppliers</Table2>
			<JoinExpression>Products.SupplierID = Suppliers.SupplierID
			</JoinExpression>
		</Relation>
	</Relations>
	<Structure>
		<Group name="Product" hint="Product Hint">
			<Field SemName="Product ID">
				<Presentation>
					<MeasureType>1</MeasureType>
					<DimensionType>1</DimensionType>
				</Presentation>
				<DBInfo>
					<DBTable>Products</DBTable>
					<DBExpression type="int">Products.ProductID
					</DBExpression>
				</DBInfo>
			</Field>
			<Field SemName="Product ID">
				<Presentation>
					<MeasureType>1</MeasureType>
					<DimensionType>1</DimensionType>
				</Presentation>
				<DBInfo>
					<DBTable>Products</DBTable>
```

```
                        <DBExpression type="int">Products.ProductID
                        </DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Product Name">
                    <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>Products</DBTable>
                        <DBExpression type="string">Products.ProductName
                        </DBExpression>
                    </DBInfo>
                </Field>
            </Group>
            <Group name="Supplier" hint="Supplier Hint">
                <Field SemName="Company">
                    <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>Suppliers</DBTable>
                        <DBExpression type="string">Suppliers.CompanyName
                        </DBExpression>
                    </DBInfo>
                </Field>
            </Group>
        </Structure>
</FutureMeta>
```

Kildekoden til unittest-klienten:

```
'*************************************************************************
' Class:    UnitTestClient
'
' Summary:  This form is used for unittesting the FutureCore component.
'
'*************************************************************************
Option Strict On
Public Class UnitTestClient
	Inherits System.Windows.Forms.Form

	Private objBuilder As New FutureCore.QueryBuilder()

#Region " Windows Form Designer generated code "

	Private Sub cmdOpenMeta_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdOpenMeta.Click
		OFDMeta.Filter = "All Files (*.*)|*.*|XML Files (*.xml)|*.xml"
		OFDMeta.FilterIndex = 2
		OFDMeta.ShowDialog()
		tbMetaFile.Text = OFDMeta.FileName
	End Sub

	Private Sub cmdClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdClear.Click
		TBResult.Clear()
	End Sub

	Private Sub cmdExecute_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdExecute.Click
		Try
			objBuilder.Init(tbMetaFile.Text)
			TBResult.Text = "Initialized QueryBuilder"
		Catch ex As FutureCore.FutureCoreException
			Dim strText As String
			strText = "*********************************" & vbNewLine
			strText = strText & "ERROR" & vbNewLine & vbNewLine
			strText = strText & ex.Message & vbNewLine & vbNewLine
			strText = strText & "Description: " & ex.InnerException.Message &
vbNewLine
			strText = strText & vbNewLine & "*****************************" &
vbNewLine
			TBResult.Text = strText
```

```
            End Try
        End Sub
End Class
```

Integrationstest

“Northwind.xml” – metadatafil til test af Microsoft SQL Server kan findes i appendiks 12.

”Oracle.xml” – metadatafil til test af Oracle database

```
<?xml version="1.0"?>
<!DOCTYPE FutureMeta SYSTEM "FutureMeta.dtd">
<FutureMeta>
        <MetaVersion>1</MetaVersion>
        <Options>
                <DateDiffStyle>0</DateDiffStyle>
                <DecimalSeparatorFlag>2</DecimalSeparatorFlag>
                <EmptyStringIncludesNullString>false</EmptyStringIncludesNullString>
        </Options>
        <Database>
                <Server>ORACLE</Server>
                <ServerVersion>8</ServerVersion>
                <DisplayName>Oracle Test</DisplayName>
                <ConnectionString>Provider=MSDAORA.1;DataSource=traine_t</ConnectionString>
        </Database>
        <Relations>
                <Relation desc="rel1">
                        <Table1>member</Table1>
                        <Table2>rental</Table2>
                        <JoinExpression>member.member_id = rental.member_id(+)
                        </JoinExpression>
                </Relation>
                <Relation desc="rel3">
                        <Table1>title</Table1>
                        <Table2>title_copy</Table2>
                        <JoinExpression>title.title_id = title_copy.title_id</JoinExpression>
                </Relation>
                <Relation desc="rel">
                        <Table1>title_copy</Table1>
                        <Table2>rental</Table2>
                        <JoinExpression>title_copy.title_id = rental.title_id and
title_copy.copy_id = rental.copy_id</JoinExpression>
                </Relation>
                <Relation desc="rel">
```

```
                <Table1>member</Table1>
                <Table2>res_view</Table2>
                <JoinExpression>member.member_id = res_view.member_id
                </JoinExpression>
        </Relation>
</Relations>
<Structure>
        <Group name="Members" hint="Employees">
                <Field SemName="Id">
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>member</DBTable>
                                <DBExpression type="int">member.member_id
                                </DBExpression>
                        </DBInfo>
                </Field>
                <Field SemName="Last name">
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>member</DBTable>
                                <DBExpression type="string">member.last_name
                                </DBExpression>
                        </DBInfo>
                </Field>
                <Field SemName="First name">
                        <Presentation>
                                <MeasureType>1</MeasureType>
                                <DimensionType>1</DimensionType>
                        </Presentation>
                        <DBInfo>
                                <DBTable>member</DBTable>
                                <DBExpression type="string">member.first_name
                                </DBExpression>
                        </DBInfo>
                </Field>
                <Field SemName="Address">
                        <Presentation>
                                <MeasureType>0</MeasureType>
```

```
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>member</DBTable>
                        <DBExpression type="string">member.address
                        </DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="City">
                    <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>member</DBTable>
                        <DBExpression type="string">member.city</DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Phone no">
                    <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>member</DBTable>
                        <DBExpression type="string">member.phone</DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Join date">
                    <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>member</DBTable>
                        <DBExpression type="date">member.join_date
                        </DBExpression>
                    </DBInfo>
                </Field>
            </Group>
            <Group name="Video" hint="Cool projects">
                <Field SemName="Video ID">
                    <Presentation>
                        <MeasureType>1</MeasureType>
```

```
                <DimensionType>1</DimensionType>
            </Presentation>
            <DBInfo>
                <DBTable>Title</DBTable>
                <DBExpression type="int">title.title_id</DBExpression>
            </DBInfo>
        </Field>
        <Field SemName="Title">
            <Presentation>
                <MeasureType>1</MeasureType>
                <DimensionType>1</DimensionType>
            </Presentation>
            <DBInfo>
                <DBTable>title</DBTable>
                <DBExpression type="string">title.title</DBExpression>
            </DBInfo>
        </Field>
        <Field SemName="Description">
            <Presentation>
                <MeasureType>0</MeasureType>
                <DimensionType>1</DimensionType>
            </Presentation>
            <DBInfo>
                <DBTable>title</DBTable>
                <DBExpression type="string">title.description
                </DBExpression>
            </DBInfo>
        </Field>
        <Field SemName="Rating">
            <Presentation>
                <MeasureType>2</MeasureType>
                <DimensionType>1</DimensionType>
            </Presentation>
            <DBInfo>
                <DBTable>title</DBTable>
                <DBExpression type="string">title.rating</DBExpression>
            </DBInfo>
        </Field>
        <Field SemName="Category">
            <Presentation>
                <MeasureType>1</MeasureType>
                <DimensionType>1</DimensionType>
            </Presentation>
            <DBInfo>
```

```
                        <DBTable>title</DBTable>
                        <DBExpression type="string">title.category
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Release date">
                <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>title</DBTable>
                        <DBExpression type="date">title.title_id</DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Copy">
                <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>title_copy</DBTable>
                        <DBExpression type="int">title_copy.copy_id
                        </DBExpression>
                </DBInfo>
        </Field>
</Group>
<Group name="Rental" hint="Task hint">
        <Field SemName="Book date">
                <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>rental</DBTable>
                        <DBExpression type="date">rental.book_date
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Actual return date">
                <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
```

```
                <DBInfo>
                        <DBTable>rental</DBTable>
                        <DBExpression type="date">rental.act_ret_date
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Due date">
                <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>rental</DBTable>
                        <DBExpression type="date">rental.exp_ret_date
                        </DBExpression>
                </DBInfo>
        </Field>
</Group>
<Group name="Reservation" hint="Assignments">
        <Field SemName="Reserved date">
                <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>res_view</DBTable>
                        <DBExpression type="date">res_view.res_date
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Title">
                <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                </Presentation>
                <DBInfo>
                        <DBTable>res_view</DBTable>
                        <DBExpression type="string">res_view.title
                        </DBExpression>
                </DBInfo>
        </Field>
        <Field SemName="Description">
                <Presentation>
                        <MeasureType>0</MeasureType>
```

```
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>res_view</DBTable>
                        <DBExpression type="string">res_view.description
                        </DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Rating">
                    <Presentation>
                        <MeasureType>2</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>res_view</DBTable>
                        <DBExpression type="string">res_view.rating
                        </DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Category">
                    <Presentation>
                        <MeasureType>1</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>res_view</DBTable>
                        <DBExpression type="string">res_view.category
                        </DBExpression>
                    </DBInfo>
                </Field>
                <Field SemName="Release date">
                    <Presentation>
                        <MeasureType>0</MeasureType>
                        <DimensionType>1</DimensionType>
                    </Presentation>
                    <DBInfo>
                        <DBTable>res_view</DBTable>
                        <DBExpression type="date">res_view.release_date
                        </DBExpression>
                    </DBInfo>
                </Field>
            </Group>
        </Structure>
</FutureMeta>
```

Kildekoden til test-klienten:

```
'*************************************************************************
' Class:    frmTestClient
'
' Summary:  This form is used for testing the functionality of the FutureCore component.
'
'*************************************************************************
Option Strict On

Imports System.Xml
Imports System.IO

Public Class frmTestClient
    Inherits System.Windows.Forms.Form

       Dim frmDBlogin As New DBlogin()
       Dim qryEngine As New FutureCore.QueryEngine()

       Private strMetaDBName As String
       Private strGroup As String
       Private selectedFields As New ArrayList()
       Private colFields As Collection

       Private Enum E_AGGREGATE_FUNCTION
              F_NONE = 0
              F_COUNT = 1
              F_SUM = 2
              F_MAX = 3
              F_MIN = 4
       End Enum

#Region " Windows Form Designer generated code "

       Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

       End Sub

       Private Sub cmdSelectDB_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdSelectDB.Click
              OpenFileDialog1.Filter = "All Files (*.*)|*.*|XML Files (*.xml)|*.xml"
              OpenFileDialog1.FilterIndex = 2
              OpenFileDialog1.ShowDialog()
```

```
            strMetaDBName = OpenFileDialog1.FileName
            frmDBlogin.GetUser()
            If (Not frmDBlogin.LoginCancelled) Then
                  Me.lvGroups.Items.Clear()
                  LoadGroups()
            End If
            ClearQuery()
      End Sub

      Private Sub lvGroups_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles lvGroups.Click
            If Me.lvGroups.SelectedItems.Count > 0 Then
                  strGroup = Me.lvGroups.SelectedItems.Item(0).Text
                  LoadFields()
            End If
      End Sub

      Private Sub lvFields_DoubleClick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles lvFields.DoubleClick
            If Me.lvFields.SelectedItems.Count > 0 Then
                  AddFieldsToQuery(strGroup & "." &
Me.lvFields.SelectedItems.Item(0).Text)
            End If
      End Sub

      Private Sub cmdNew_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdNew.Click
            ClearQuery()
      End Sub

      Private Sub cmdSubmitQuery_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdSubmitQuery.Click
            Dim frmResult As QueryResult
            Dim ds As DataSet

            Try
                  If Me.selectedFields.Count > 0 Then
                        ds = qryEngine.GetDataset(strMetaDBName, frmDBlogin.UserName,
frmDBlogin.PSWD, QueryFields2XML(selectedFields))
                        If ds.Tables(0).Rows.Count < 1 Then
                              MsgBox("There are no data for this query")
                        Else
                              frmResult = New QueryResult
                              frmResult.ExecuteQuery(ds)
```

```
                End If
            Else
                MsgBox("There are no selected fields for this query")
            End If
        Catch ex As FutureCore.FutureCoreException
            tbOutput.AppendText(ex.ToString & ex.InnerException.Message)
        Catch ex As Exception
            tbOutput.AppendText(ex.ToString)
        End Try
    End Sub

    Private Sub LoadGroups()
        Dim colGroups As Collection
        Dim grp As FutureCore.GroupDataType

        Try
            colGroups = qryEngine.GetGroups(strMetaDBName)

            For Each grp In colGroups
                Me.lvGroups.Items.Add(grp.Name)
                'itm.ToolTipText = grp.Hint
            Next grp
            If Me.lvGroups.Items.Count > 0 Then
                Me.lvGroups.Items(0).Selected = True
            End If
            If Me.lvGroups.SelectedItems.Count > 0 Then
                strGroup = Me.lvGroups.SelectedItems.Item(0).Text
                LoadFields()

            End If
        Catch ex As FutureCore.FutureCoreException
            Me.lvGroups.Items.Clear()
            Me.lvFields.Items.Clear()
            tbOutput.AppendText(ex.ToString & ex.InnerException.Message)
        Catch ex As Exception
            Me.lvGroups.Items.Clear()
            Me.lvFields.Items.Clear()
            tbOutput.AppendText(ex.ToString)
        End Try
    End Sub

    Private Sub LoadFields()
        Dim fld As FutureCore.FieldDataType
```

```
            Try
                  colFields = qryEngine.GetFields(strMetaDBName, strGroup)

                  Me.lvFields.Items.Clear()
                  For Each fld In colFields
                        Me.lvFields.Items.Add(fld.SemName)
                        'itm.ToolTipText = fld.Hint
                  Next fld
            Catch ex As FutureCore.FutureCoreException
                  tbOutput.AppendText(ex.ToString & ex.InnerException.Message)
            Catch ex As Exception
                  tbOutput.AppendText(ex.ToString)
            End Try
      End Sub

      Private Sub ClearQuery()
            Dim count As Integer
            selectedFields.Clear()
            lvQuery.Items.Clear()
            tbOutput.AppendText("Query has been cleared" & vbNewLine)
      End Sub

      Private Function AggExpr(ByVal strExpr As String, ByVal enmAgg As
E_AGGREGATE_FUNCTION) As String
            Select Case enmAgg
                  Case E_AGGREGATE_FUNCTION.F_NONE
                        Return strExpr
                  Case E_AGGREGATE_FUNCTION.F_SUM
                        Return "SUM(" & strExpr & ")"
                  Case E_AGGREGATE_FUNCTION.F_MAX
                        Return "MAX(" & strExpr & ")"
                  Case E_AGGREGATE_FUNCTION.F_MIN
                        Return "MIN(" & strExpr & ")"
                  Case E_AGGREGATE_FUNCTION.F_COUNT
                        Return "COUNT(" & strExpr & ")"
            End Select
      End Function

      Private Sub AddFieldsToQuery(ByVal strExpr As String)
            Dim fld As New UserQueryField

            fld.Included = True
            fld.SortOrder = UserQueryField.E_SORT_ORDER.E_ASCENDING
            fld.Expr = strExpr
```

```
            selectedFields.Add(fld)

            lvQuery.Items.Add(strExpr)
            tbOutput.AppendText(strExpr & " has been added to Query" & vbNewLine)
      End Sub

      Private Function QueryFields2XML(ByVal Fields As ArrayList) As String
            Dim xmldoc As MSXML2.DOMDocument
            Dim header As MSXML2.IXMLDOMNode
            Dim attr As MSXML2.IXMLDOMAttribute
            Dim elem As MSXML2.IXMLDOMElement
            Dim fldElem As MSXML2.IXMLDOMElement
            Dim fld As UserQueryField

            xmldoc = New MSXML2.DOMDocument

            ' Create header info tag
            header = xmldoc.createNode(XmlNodeType.ProcessingInstruction, "xml", "")
            xmldoc.appendChild(header)
            attr = xmldoc.createAttribute("version")
            attr.value = "1.0"
            header.attributes.setNamedItem(attr)
            attr = xmldoc.createAttribute("encoding")
            attr.value = "ISO-8859-1"
            header.attributes.setNamedItem(attr)

            ' create document element
            elem = xmldoc.createElement("FutureQuery")
            xmldoc.appendChild(elem)

            ' add fields
            For Each fld In Fields
                  fldElem = xmldoc.createElement("UserField")
                  elem.appendChild(fldElem)
                  fld.AddToXML(CType(fldElem, MSXML2.IXMLDOMNode))
            Next fld

            Return xmldoc.xml

      End Function

      Private Sub cmdAdd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdAdd.Click
            AddFieldsToQuery(Me.tbExpr.Text)
```

```
            Me.tbExpr.Text = ""
      End Sub

      Private Sub SetAggVisibility(ByVal strDBType As String)
            Select Case UCase(strDBType)
                  Case "STRING", "DATE"                    'Hide SUM
                        MenuItem2.Visible = False
                  Case "INT", "REAL"                 'Show SUM
                        MenuItem2.Visible = True
            End Select
      End Sub

      Private Sub lvFields_MouseUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles lvFields.MouseUp
            If e.Button = MouseButtons.Right Then
                  Dim fld As FutureCore.FieldDataType
                  For Each fld In colFields
                        If fld.SemName = Me.lvFields.SelectedItems.Item(0).Text Then
                              SetAggVisibility(fld.DBDataType)
                        End If
                  Next
                  Me.ContextMenu1.Show(lvFields, New Point(e.X, e.Y))
            End If
      End Sub

      Private Sub MenuItem1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MenuItem1.Click
            If Me.lvFields.SelectedItems.Count > 0 Then
                  AddFieldsToQuery(AggExpr(strGroup & "." &
Me.lvFields.SelectedItems.Item(0).Text, E_AGGREGATE_FUNCTION.F_COUNT))
            End If
      End Sub

      Private Sub MenuItem2_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MenuItem2.Click
            If Me.lvFields.SelectedItems.Count > 0 Then
                  AddFieldsToQuery(AggExpr(strGroup & "." &
Me.lvFields.SelectedItems.Item(0).Text, E_AGGREGATE_FUNCTION.F_SUM))
            End If
      End Sub

      Private Sub MenuItem3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem3.Click
            If Me.lvFields.SelectedItems.Count > 0 Then
```

```
                    AddFieldsToQuery(AggExpr(strGroup & "." &
Me.lvFields.SelectedItems.Item(0).Text, E_AGGREGATE_FUNCTION.F_MAX))
             End If
       End Sub

       Private Sub MenuItem4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem4.Click
             If Me.lvFields.SelectedItems.Count > 0 Then
                    AddFieldsToQuery(AggExpr(strGroup & "." &
Me.lvFields.SelectedItems.Item(0).Text, E_AGGREGATE_FUNCTION.F_MIN))
             End If
       End Sub

       Private Sub MenuItem5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem5.Click
             Dim strCondition, message As String
             Dim fld As UserQueryField
             If Me.lvQuery.SelectedItems.Count > 0 Then
                    message = "Enter condition"
                    strCondition = InputBox(message)
                    For Each fld In selectedFields
                           If lvQuery.SelectedItems.Item(0).Text = fld.Expr Then
                                  fld.AddCondition(strCondition)

       lvQuery.SelectedItems.Item(0).SubItems.Add(strCondition)
                           End If
                    Next
             End If
       End Sub

       Private Sub MenuItem6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem6.Click
             If Me.lvQuery.SelectedItems.Count > 0 Then
                    Dim fld As UserQueryField
                    Dim fld2remove As UserQueryField
                    For Each fld In selectedFields
                           If lvQuery.SelectedItems.Item(0).Text = fld.Expr Then
                                  fld2remove = fld
                           End If
                    Next
                    selectedFields.Remove(fld)
                    lvQuery.Items.Remove(lvQuery.SelectedItems.Item(0))
                    tbOutput.AppendText(fld.Expr & " has been removed from the Query" &
vbNewLine)
```

```
            End If
      End Sub

      Private Sub cmdSaveQuery_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles cmdSaveQuery.Click
            Try
                  Dim strXML As String
                  Dim xmldoc As MSXML2.DOMDocument

                  If selectedFields.Count > 0 Then
                        strXML = QueryFields2XML(selectedFields)
                  End If

                  Dim strFile As String

                  SaveFileDialog1.Filter = "All Files (*.*)|*.*|XML Files
(*.xml)|*.xml"
                  SaveFileDialog1.FilterIndex = 2
                  SaveFileDialog1.ShowDialog()

                  xmldoc = New MSXML2.DOMDocument
                  xmldoc.loadXML(strXML)
                  xmldoc.save(SaveFileDialog1.FileName)
            Catch ex As Exception
                  tbOutput.AppendText(ex.ToString)
            End Try
      End Sub

      Private Sub cmdOpenQuery_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdOpenQuery.Click
            Try
                  Dim frmResult As QueryResult
                  Dim ds As DataSet
                  Dim filename, input, xmlQuery As String

                  OpenFileDialog1.Filter = "All Files (*.*)|*.*|XML Files
(*.xml)|*.xml"
                  OpenFileDialog1.FilterIndex = 2
                  OpenFileDialog1.ShowDialog()

                  If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
                        filename = OpenFileDialog1.FileName

                        Dim sr As StreamReader = File.OpenText(filename)
```

```
                        input = sr.ReadLine()
                        While Not input Is Nothing
                                xmlQuery += input
                                input = sr.ReadLine()
                        End While
                        sr.Close()
                End If

                ds = qryEngine.GetDataset(strMetaDBName, frmDBlogin.UserName,
frmDBlogin.PSWD, xmlQuery)

                If ds.Tables(0).Rows.Count < 1 Then
                        MsgBox("There are no data for this query")
                Else
                        frmResult = New QueryResult
                        frmResult.ExecuteQuery(ds)
                End If

        Catch ex As Exception
                tbOutput.AppendText(ex.ToString)
        End Try
    End Sub

    Private Sub cmdGetSQL_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmdGetSQL.Click
        Try
                Dim frmResult As QueryResult
                Dim filename, input, xmlQuery, sqlString As String

                OpenFileDialog1.Filter = "All Files (*.*)|*.*|XML Files
(*.xml)|*.xml"
                OpenFileDialog1.FilterIndex = 2
                OpenFileDialog1.ShowDialog()

                If OpenFileDialog1.ShowDialog() = DialogResult.OK Then
                        filename = OpenFileDialog1.FileName

                        Dim sr As StreamReader = File.OpenText(filename)

                        input = sr.ReadLine()
                        While Not input Is Nothing
                                xmlQuery += input
                                input = sr.ReadLine()
```

```
                    End While
                    sr.Close()
                End If

                sqlString = qryEngine.GetSQL(strMetaDBName, xmlQuery)

                tbOutput.AppendText(sqlString)

            Catch ex As Exception
                tbOutput.AppendText(ex.ToString)
            End Try
        End Sub
End Class
```

Systemtest

Testen udføres på testmaskinen OL37144 ved brug af 'Northwind.xml' metadatafilen.

Test Område: Definering af forespørgsel		**Test Case: 2.1**		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
1	Start Internet Explorer og indtast følgende adresse: http://localhost/WebFuture/FutureStep1.aspx	Skærmbilledet "Step 1 – Select measure" vises med en oversigt over measures.		
2	Klik på "Proceed to Step 2"	Der vises en fejlbeskrivelse: "You haven't selected a measure for the query"		
3	Vælg 'Count on ' for Product ID Vælg 'Count on ' for Order ID Klik på "Proceed to Step 2"	Der vises en fejlbeskrivelse: "You can only select one measure for the query. Please modify your selections"		
4	Vælg 'Count on ' for Order ID Klik på "Proceed to Step 2"	Skærmbilledet "Step 2 – Select dimensions" vises med en oversigt over dimensioner.		

Test Område: Definering af forespørgsel		Test Case: 2.1		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
5	Klik på “Proceed to Step 3”	Der vises en fejlbeskrivelse: ”You haven’t selected any dimensions for the query”		
6	Vælg 6 dimensioner ved at sætte flueben ud for dem. Klik på “Proceed to Step 3”	Der vises en fejlbeskrivelse: ”You can only select five dimensions for the query. Please modify your selections”		
7	Vælg de fem dimensioner: Product.Category, Order.Year, Order.Month, Ship to.Country og Shipper.Company Klik på “Proceed to Step 3”	Skærmbilledet ”Step 3 – Select conditions” vises med en oversigt over grupper og felter.		
8	Klik på “Submit Query”	Kuben, som indeholder 5 dimensioner og 1 measure, vises på skærmen.		

Test Område: Definering af forespørgsel		Test Case: 2.2		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
1	Start Internet Explorer og indtast følgende adresse: http://localhost/WebFuture/FutureStep1.aspx	Skærmbilledet ”Step 1 – Select measure” vises med en oversigt over measures.		
2	Vælg ‘Count on ‘ for Order ID Klik på “Proceed to Step 2”	Skærmbilledet ”Step 2 – Select dimensions” vises med en oversigt over dimensioner.		
3	Vælg de to dimensioner: Order.Year og Shipper.Company Klik på “Proceed to Step 3”	Skærmbilledet ”Step 3 – Select conditions” vises med en oversigt over grupper og felter.		

Test Område: Definering af forespørgsel		**Test Case: 2.2**		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
4	Indtast følgende condition: Product ID = 10 Klik på “Submit Query”	Kuben vises på skærmen, og den indeholder følgende data: (1996, Federal Shipping, 3) (1996, United Package, 1) (1996, COUNT, 4) (1997, Federal Shipping, 6) (1997, Speedy Express, 3) (1997, United Package, 10) (1997, COUNT, 19) (1998, Federal Shipping, 1) (1998, Speedy Express, 1) (1998, United Package, 8) (1998, COUNT, 10)		
5	Klik på ”New Query…”	Skærmbilledet ”Step 1 – Select measure” vises med en oversigt over measures.		
6	Vælg ‘Count on ‘ for Order ID Klik på “Proceed to Step 2”	Skærmbilledet ”Step 2 – Select dimensions” vises med en oversigt over dimensioner.		
7	Vælg de to dimensioner: Order.Year og Shipper.Company Klik på “Proceed to Step 3”	Skærmbilledet ”Step 3 – Select conditions” vises med en oversigt over grupper og felter.		

Test Område: Definering af forespørgsel		Test Case: 2.2		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
8	Indtast følgende condition: Ship To.Country like 'Den%' Klik på "Submit Query"	Kuben vises på skærmen, og den indeholder følgende data: (1996, Federal Shipping, 3) (1996, COUNT, 3) (1997, Federal Shipping, 3) (1997, Speedy Express, 5) (1997, United Package, 3) (1997, COUNT, 11) (1998, Federal Shipping, 1) (1998, Speedy Express, 1) (1998, United Package, 2) (1998, COUNT, 4)		

Test Område: Definering af forespørgsel		Test Case: 2.3		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
1	Start Internet Explorer og indtast følgende adresse: http://localhost/WebFuture/FutureStep1.aspx	Skærmbilledet "Step 1 – Select measure" vises med en oversigt over measures.		
2	Vælg 'Sum of ' for Order Details.Total Klik på "Proceed to Step 2"	Skærmbilledet "Step 2 – Select dimensions" vises med en oversigt over dimensioner.		

Test Område: Definering af forespørgsel		Test Case: 2.3		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
3	Vælg de to dimensioner: Order.Year og Customer.Country Klik på “Proceed to Step 3”	Skærmbilledet ”Step 3 – Select conditions” vises med en oversigt over grupper og felter.		
4	Indtast følgende condition: Customer.Country like ’U%’ Klik på “Submit Query”	Kuben vises på skærmen, og den indeholder følgende data: (1996, UK, 9.654,00) (1996, USA, 41.907,80) (1996, SUM, 51.561,80) (1997, UK, 27.832,60) (1997, USA, 121.037,70) (1997, SUM, 148.870,30) (1998, UK, 23.129,91) (1998, USA, 100.621,48) (1998, SUM, 123.751,39)		

Test Område: Definering af forespørgsel		Test Case: 2.4		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
1	Start Internet Explorer og indtast følgende adresse: http://localhost/WebFuture/FutureStep1.aspx	Skærmbilledet ”Step 1 – Select measure” vises med en oversigt over measures.		

Test Område: Definering af forespørgsel		**Test Case: 2.4**		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
2	Vælg 'Maximum of ' for Order Details.Total Klik på "Proceed to Step 2"	Skærmbilledet "Step 2 – Select dimensions" vises med en oversigt over dimensioner.		
3	Vælg de to dimensioner: Order.Year og Manager.First Name Klik på "Proceed to Step 3"	Skærmbilledet "Step 3 – Select conditions" vises med en oversigt over grupper og felter.		
4	Klik på "Submit Query"	Kuben vises på skærmen, og den indeholder følgende data: (1996, N/A, 2.304,00) (1996, Andrew, 8.432,00) (1996, Steven, 10.540,00) (1996, MAX, 10.540,00) (1997, N/A, 5.268,00) (1997, Andrew, 10.540,00) (1997, Steven, 10.329,20) (1997, MAX, 10.540,00) (1998, N/A, 15.810,00) (1998, Andrew, 15.810,00) (1998, Steven, 10.540,00) (1998, MAX, 15.810,00)		

Test Område: Definering af forespørgsel		Test Case: 2.5		
Nr.	**Input**	**Forventet resultat**	**Aktuelt resultat**	**OK**
1	Start Internet Explorer og indtast følgende adresse: http://localhost/WebFuture/FutureStep1.aspx	Skærmbilledet ”Step 1 – Select measure” vises med en oversigt over measures.		
2	Vælg ‘Minimum of ‘ for Product.On Stock Klik på “Proceed to Step 2”	Skærmbilledet ”Step 2 – Select dimensions” vises med en oversigt over dimensioner.		
3	Vælg de to dimensioner: Product.Category og Supplier.Country Klik på “Proceed to Step 3”	Skærmbilledet ”Step 3 – Select conditions” vises med en oversigt over grupper og felter.		
4	Indtast følgende condition: Product.Category like ’Seafood’ Klik på “Submit Query”	Kuben vises på skærmen, og den indeholder følgende data: (Seafood, Australia, 42) (Seafood, Denmark, 5) (Seafood, France, 62) (Seafood, Germany, 10) (Seafood, Japan, 24) (Seafood, Sweden, 11) (Seafood, USA, 85) (Seafood, MINIMUM, 5)		