Tilpasningsdyktig og modulær datavarehusløsning

Rapport fra datavarehusprosjekt

Anders Andersen
NORUT IT
aa@computer.org

Geir Egil Myhre
Jupiter SystemPartner
geirmy@stud.cs.uit.no

Odd-Halvard Bjørnstad
Jupiter SystemPartner
odd-halvard.bjornstad@jupiter.no

Kolbjørn Engeseth
Jupiter SystemPartner
kolbjorn.engeseth@jupiter.no

Versjon 1.8

21. november 2001

Et generelt datavarehus-system består av en rekke komponenter. Integrasjonskomponenten (kalt GENDVH i figur 1) har som oppgave å fylle datavarehuset med data som den henter fra kildene (KILDER i figur 1). En spesifikasjon (DVHSPES i figur 1) spesifiserer hvilke data fra kildene som datavarehuset skal inneholde. En analysekomponent (GENRAP i figur 1) generer rapporter basert på spørringer mot datavarehuset.

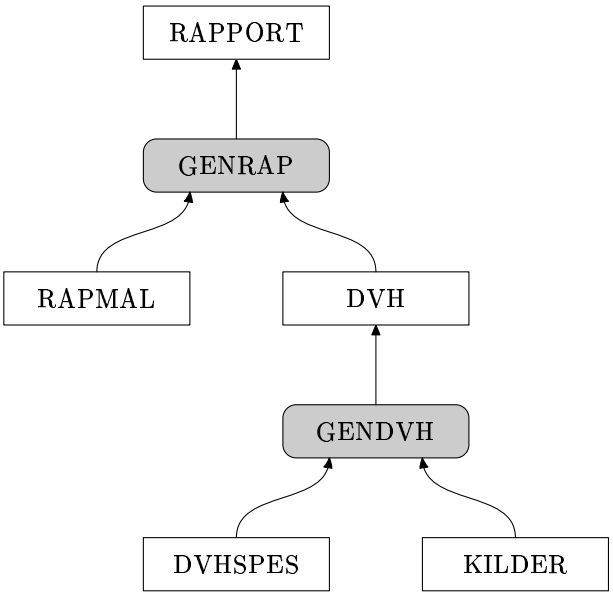
Problemer med et tradisjonelt slikt datavarehus-system er at integrasjonskomponenten må lages spesielt for hvert enkelt oppdrag og at analysekomponenten kun generer statiske rapporter (rapporter i henhold til krav fra kunden). Dette datavarehusprosjektet har undersøkt muligheten for en tilpasningsdyktig og modulær datavarehusløsning som adresserer disse problemene.

1 Generere datavarehus

Et datavarehus genereres fra et sett av kilder basert på hvilke informasjonen som er nødvendig for å få generert de rapportene som kunden har behov for. Det kan være flere og ulike typer kilder som er basis for et datavarehus. Ofte er en eller flere av kildene databaser hvor utdraget er realisert som et sett av database-spørringer (typisk SQL uttrykk). Andre kilder kan være analyseverktøy, regnskapssystemer og periodiske rapporter.

En måte å realisere integrasjonskomponenten er å gjøre det mulig for brukeren å velge (spesifisere) de dataene hun ønsker å fylle datavarehuset med. Dette kan for eksempel realisere i et program med grafisk brukergrensesnitt hvor brukeren markerer de dataene som er interessant. Muligheten til å utføre enkel prosessering på disse dataene før de lastes inn i datavarehuset kan også inkluderes i et slikt program. De sett av data som brukeren kan velge fra er gitt av de kildene som er tilgjengelig. En slik løsning vil måtte lages spesielt for hvert enkelt datavarehus da realiseringen er avhengig av hvilke kilder som er tilgjengelig og hvordan disse dataene kan nås.

Det genererte datavarehuset kan være representert som en database. Fordelene med det er tilgjengeligheten på databasefunksjonalitet som kan benyttes av analysekomponenten (inkludert søk og



Figur 1: Generering av datavarehus og rapport

enkel prosessering). Et komplett databasesystem har en bruk-, drift- og vedlikeholdskostnad, og et sett av ressurskrav, som kan være unødvendig store for mange mindre og mellomstore datavarehusløsninger. Disse kostnadene og ressurskravene begrenser bruksområder og omgivelser for innføring av datavarehus.

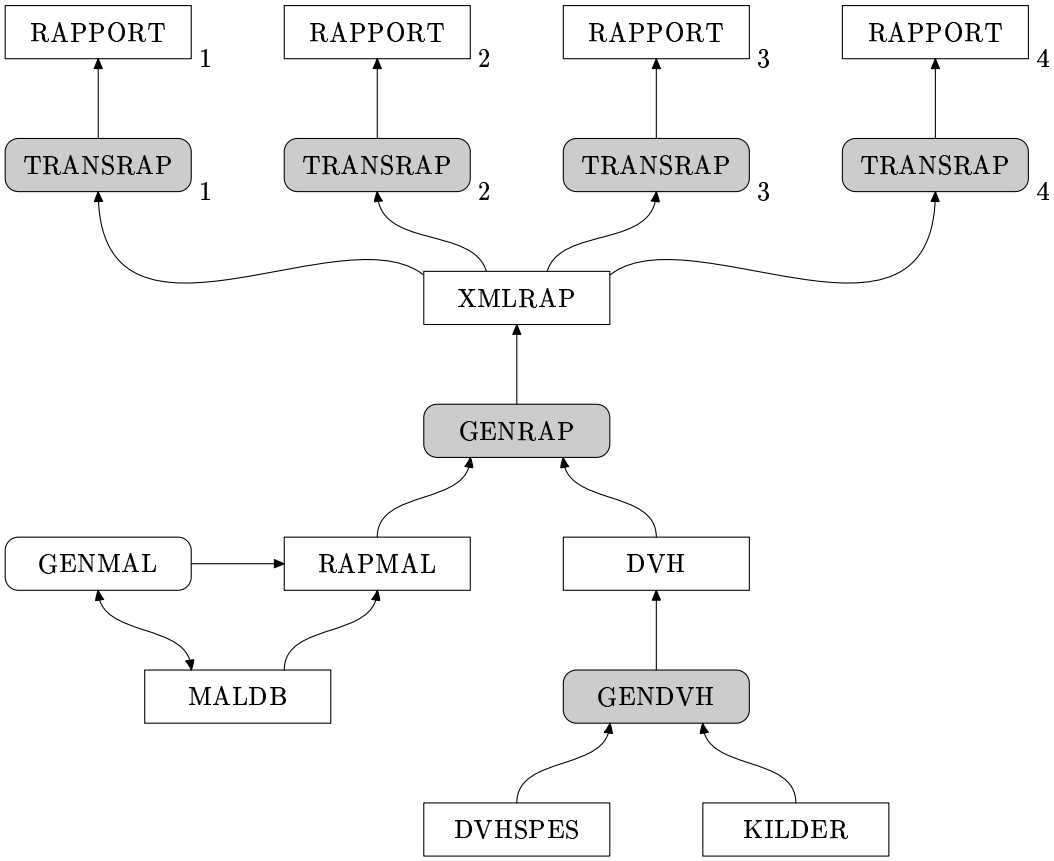
2 Generere rapporter

Innholdet i et datavarehus er en samling informasjon som er valgt ut i fra brukers behov. Fra denne informasjonen skal det kunne genereres ulike typer rapporter basert på brukerens ulike behov. Dette gjøres ved å lage et sett av analysekomponenter som genererer det nødvendige settet av rapporter. Brukeren velger analysekomponent utifra hvilken rapport hun ønsker generert.

En slik løsning krever at for alle ulike typer rapporter eksisterer det en spesifikk analysekomponent. En slik løsning er ikke veldig fleksibel og gir ikke brukeren muligheten til umiddelbart å generere nye typer rapporter hvis behovet for det skulle oppstå. Slike ad-hoc rapporter vil for mange datavarehusbrukere øke nytten av et datavarehus-system betraktelig. Økt fleksibilitet i muligheten for å analysere dataene i et datavarehus vil kunne gi brukere bedre grunnlag for å fatte beslutninger. I mange situasjoner er det umulig på forhånd å vite alle mulige analyser og rapporter som kan være nyttig for brukerene. Konsekvensen er at det eksisterende sett av analysekomponenter som følger med et datavarehus ikke er tilstrekkelig.

Løsningen er at brukeren kan generere sine egne analysekomponenter ved behov. Dette må kunne gjøres på et rimelig høyt nivå slik at brukeren ikke må kunne beherske detaljer om hvordan datavarehuset er bygd opp og virker. Dette kan gjøres med et verktøy som tilbyr et grafisk brukergrensesnitt (GENMAL i figur 2) og som kan brukes for å generere nye analysekomponenter (eller rapport-maler) basert på den informasjonen vi finner i datavarehuset. Verktøyet må altså ved oppstart analysere den informasjonen vi finner i datavarehuset og enkelt gi brukeren mulighet for spesifisere hvilke data som skal hentes ut og hvordan den skal analyseres.

Et annet poeng er at en generert rapport kan presenteres på ulike former. Rapporten kan være et dokument i ulike format (HTML, PDF, PowerPoint presentasjon, XML) med forskjellige presentasjonformer (tall, grafikk, tekst).



Figur 2: Generering av ulike rapporter fra et datavarehus (DVH)

3 XML som utvekslingsformat

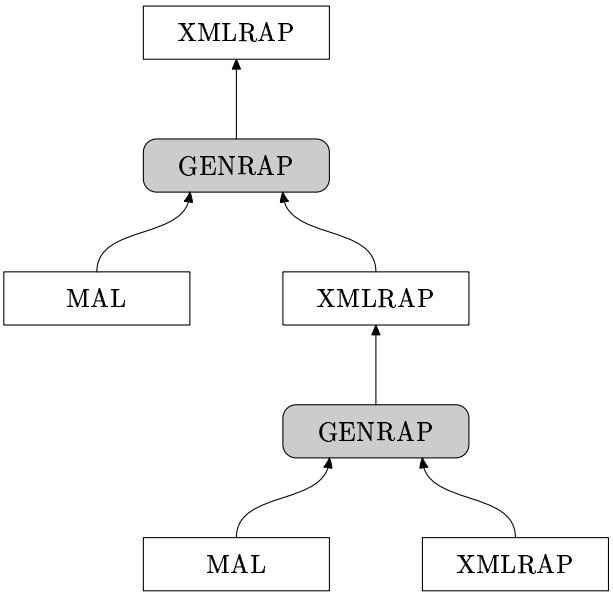
XML er et format for strukturert informasjon. Et XML dokument har en struktur som vanligvis er spesifisert i en DTD fil. Denne strukturen kan sees på som typen (eller klassen) til et sett av XML filer og den forteller oss hvilke informasjon et XML dokument kan inneholde og hvordan denne informasjonen er organisert i dokumentet. XML sin popularitet har også resultert i at tilgjengeligheten av verktøy for å manipulere og jobbe med XML dokumenter er meget bra.

XML er derfor et godt valg som et rapport-format i et datavarehus-system. Det betyr at analysekomponentene generer rapporter i XML som kan etterarbeides (prosesseres) til endelige rapporter (som for eksempel et HTML dokument).

Figur 2 skisseres et slikt datavarehus-system. Vi ser at analysekomponenten (GENRAP) genererer en XML-rapport (XMLRAP) som transformeres (med TRANSRAP) til ulike former for rapporter (RAPPORT).

4 XML rapport som resultat og kilde

Hvis vi ser nærmere på datavarehus-systemet skissert i figur 2 så legger vi merke til at vi har tre ulike typer resultater (eller genererte dokumenter/databaser). Disse er generert i ulike typer transformasjoner (uttrekk eller prosessering) med utgangspunkt i en (eller flere) kilder og en mal eller spesifikasjon (figuren viser DVHSPES og RAPMAL, men vi må ha en tilsvarende mal



Figur 3: Rapporter på ulike nivå

for hver enkelt rapporttype). De ulike resultatene eller rapportene (DVH, XMLRAP, RAPPORT) og deres maler, viser en rekursiv struktur fra en generell rapport til en spesialisert rapport via en analysekomponent eller rapportgenerator (GENRAP). Figur 3 skisserer en slik generalisering av et datavarehus-systemet.

Vi kan altså tenke oss å representere både datavarehus og rapporter fra et datavarehus som et XML-dokument (XMLRAP). En rapport fra et datavarehus kan da på tilsvarende måte sees på som et nytt datavarehus (et utdrag av det opprinnelige datavarehuset). Dette er spesielt interessant hvis man tenker seg at man i gitte situasjoner kun ønsker et utdrag av et opprinnelig datavarehus (for eksempel i tilfelle hvor en selger er ute å reiser og ønsker et utdrag av datavarehuset, med fokus på de kundene han skal besøke, tilgjengelig på sin mobile enhet).

Man kan også tenke seg at større bedrifter genererer et stort datavarehus som inneholder store informasjonsmengder. Ulike avdelinger i bedriften kan da genererer utdrag av dette stor datavarehuset med fokus på sine informasjonsbehov. Til slutt kan man se for seg ansatte som generer personlige datavarehus som gir støtte for deres personlige informasjonsbehov og kanskje også spesialiserte datavarehus for en gitt oppgave (for eksempel et kundebesøk).

Ved et slikt hierarki av datavarehus kan den opprinnelige kilden (eller noen av de opprinnelige kildene) og den endelige rapporten (eller noen av de endelige rapportene) være de eneste dataene som ikke er representert som et XML-dokument. Vi kan tenke oss at den opprinnelige kilden er en databaser og den endelige rapporten er et HTML dokument.

Hvis man velger å representere de ulike nivå av datavarehus i et XML-dokument, så er muligheten for å generere en rapport (eller et nytt datavarehus) mange. Det mest opplagte er å bruke XSLT hvor man beskriver selve transformasjonen (spesifikasjonen) i et eget XML-dokument. Mal for å generere rapporten er altså en XML-dokument i XSLT formatet som beskriver den transformasjonen som skal foretas. XSLT i lag med XPath gir oss veldig stor fleksibilitet i hvordan rapporter vi kan generere fra en XML-kilde. XPath er en måte å uttrykke utvalg, elementer eller deler av et XML dokument. XPath er for XML hva SQL select-uttrykk er for SQL databaser.

Det er også mulig å gjøre transformasjonen fra en XML-kilde til en rapport med andre verktøy. Disse vil ofte være basert på enten SAX eller DOM (eller begge). SAX er en programeringsmodell hvor XML dokumentet håndteres sekvensielt og hvor strukturelementene (merker som gir et XML

dokument struktur og som finnes i start og slutt par) generer en hendelse som programmereren kan fange og behandle. DOM er en programeringsmodell hvor et XML dokument og dets struktur blir representert som et tre som programmereren kan aksessere og manipulere.

5 Gjennomføring

Eksisterende datavarehusløsninger fra Jupiter SystemPartner ble brukt for å evaluere en tilpasningsdyktig og modulær datavarehusløsning basert på XML og de observasjonene som er beskrevet over. Dette viste seg veldig lovende. Det gjenstår ennå å få realisert en komplett datavarehusløsning, men de forsøkene som har vært gjort i dette prosjektet viser at den beskrevne modellen er et godt utgangspunkt for å nå målene om fleksibilitet.

A Ressurser

XML `http://www.w3.org/`
`http://www.xml.org/`

XSLT `http://www.w3.org/Style/XSL/`
`http://saxon.sourceforge.net/`

SAX `http://www.saxproject.org/`
`http://java.sun.com/xml/`

DOM `http://www.w3.org/DOM/`
`http://java.sun.com/xml/`

B Reiserapport

I forbindelse med prosjektet deltok prosjektdeltakerene på Dataforeningens Geilo-seminar 2001. Seminaret haddde tittelen «Teknologier for komponentbaserte, distribuerte systemer med J2EE, EJB, .NET, XML, C# og Java». Innholdet på årets seminar ble funnet såpass relevant og interessant at det er inkludert i prosjektaktivitetene til dette datavarehusprosjektet. Deltakerne på seminaret var en fin blanding av akademikere og representatnter for IT-industrien og IT-brukere. Mange interessante diskusjoner og mye utveksling av erfaringer foregår på et slikt seminar utenom det fastlagte programmet. Nedenfor følger en kort beskrivelse av noen av høydepunktene (fra vårt ståsted):

Jon Oldevik fra SINTEF Tele og Data presenterte «Modelldreven Arkitektur». Denne presentasjonen viste hvordan UML brukes i modelldreven arkitektur til å spesifisere ulike faser av prosjekter basert på CORBA, Enterprise JavaBeans (J2EE), Microsoft .NET og XML. Et nøkkelbegrep her er interoperabilitet. Det vil si at systemer på tvers av ulike komponent infrastrukturer kan modeleres og utvikles.

Fra Tore Ragnhildstveit i Microsoft Consulting Services fikk vi en underholdene introduksjon til Microsoft .NET, og da spesielt Web-services og XML-basert teknologi i denne platformen.

Sune Jakobsson fra Telenor FoU ga en overordnet introduksjon til XML før han gikk mer i detalj på XML-teknologien SOAP. SOAP er informasjonsbæreren i mange nye nettbaserte applikasjoner og tilbyr en XML basert metode-kall mekanisme som (vanligvis) bruker HTTP som transportprotokoll.

Jon Ølnes fra PKI Consulting Services og Universitetet i Tromsø presenterte problemet med sikkerhet i distribuerte systemer. Han belyste aspekter knyttet til tillit, tiltrodde tjenester, sikkerhetsarkitekturer og krypto-grafisk beskyttelse, i tillegg til at han diskuterte forholdet mellom kvalitetssikring og sikkerhet.

Seminaret ble avsluttet med en paneldebatt med tittelen «Ny muligheter – Eller for mange like teknologier?» Spørsmålene som ble stilt og diskutert var følgende:

- Har vi gode nok teknologier?
- Hvilke vil overleve?
- Hvilke vil det være mest nyttige å bruke i dag og i fremtiden?
- Har vi nå verktøyene for distribuert systemutvikling, eller vil neste år bringe oss stadig nye teknologier?

Mer informasjon om seminaret er tilgjengelig her:

`http://dataforeningen.no/ostlandet/geilo2001/index.shtml`