TRIAD **Arbetsrapport**

- Krav på IA
- Nästa Generation Modellering
- Avancerad utbildning för handledare
- Katalogprinciper
- Verktyg
- Informationspridning

- Rapport K nr 1: IRDS
- Rapport K nr 2: IRDS Modeller och modellnivåer
- Rapport K nr 3: Kopping begreppsmodell - relationsmodell
- Rapport K nr 4: IBM:s Repository Manager- en Introduktion
- Rapport K nr 5: IBM:s Repository Manager: Datamodelleringsbegreppen
- Rapport K nr 6: IBM:s Repository Manager: Begreppsmodellering i Information Model

IBM:s Repository Manager – en introduktion

Stig Berild

Rapporten är skriven i och för TRIAD delprojekt Katalogprinciper.

TRIAD-projektet drivs gemensamt av Televerket, Posten, Statskontoret , Ericsson Data Services och SISU, Svenska Institutet för Systemutveckling.
För vidare information kontakta SISU Informationscentrum, tel 08-752 16 00, fax 08-752 68 00,Box 1250, 164 28 Kista.

Innehåll

IBMs Repository Manager: En introduktion

Innhold

IBMs Repository Manager:

En introduktion

0. Bakgrund

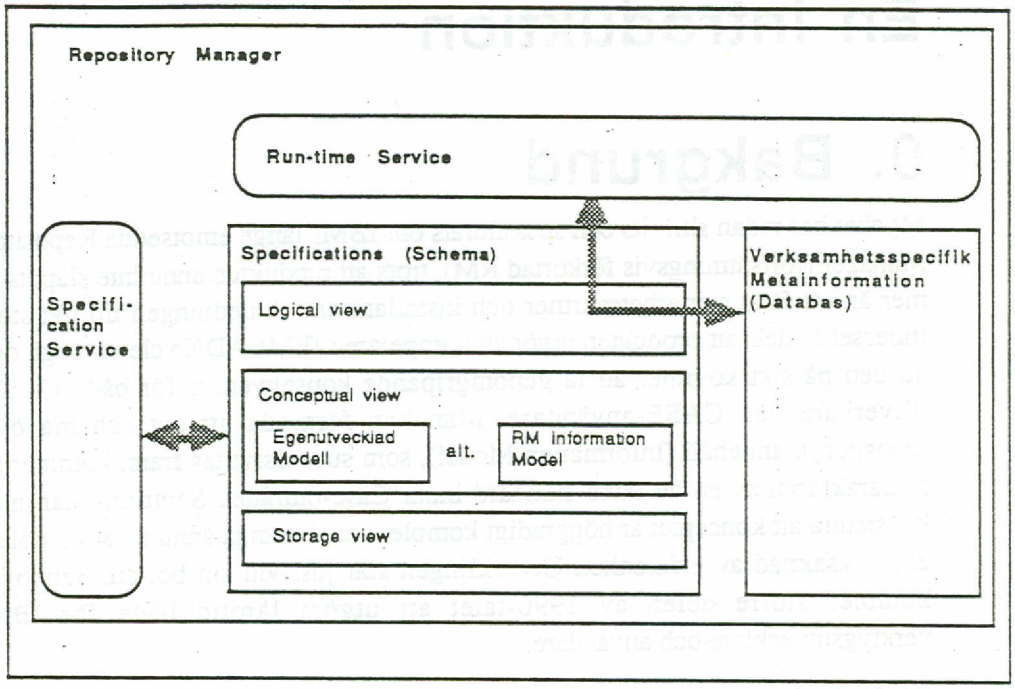
Mycket har redan skrivits och spekulerats om IBMs länge emotsedda Repository Manager (fortsättningsvis förkortad RM), trots att produkten ännu inte släppts till mer än ett fåtal samarbetspartner och installationer. Anledningen till det stora intresset är dels att produkten utgör en hörnpelare i IBMs AD/Cycle-strategi, dels att den på sikt kommer att få genomgripande konsekvenser för både CASE-tillverkare och CASE-användare. Man kan förmoda att det schema över repositoryts innehåll (Information Model), som successivt tas fram, kommer att få karaktären av en de facto-standard inom Case-området. Samtidigt kan man konstatera att konceptet är höggradigt komplext med många ännu olösta problem eller avsaknad av erfarenhet. Utvecklingen står just vid sin början. Sannolikt kommer större delen av 1990-talet att utgöra lärotid både för IBM, verktygsutvecklare och användare.

Därför är det mycket viktigt för alla berörda parter, inte minst för företagens informationsstrateger och andra beslutsfattare, att följa med, förstå och vara beredd.

I ett antal kortfattade dokument kommer vi att översiktligt presentera olika aspekter på RM och dess Information Model. På grund av viss brist på tillgängligt material och den under överskådlig tid förväntade snabba utvecklingen kommer därutöver en löpande bevakning och rapportering att behövas.

Tänkta läsare är både beslutsfattare, utvecklare av metoder och modeller, systemutvecklare, CASE-användare och andra, som känner att repository-filosofin på sikt kommer att påverka den egna arbetsrollen samt företagets syn på informationen som en resurs. Visserligen kan det hävdas att den normale hanteraren av repository-information aldrig kommer i direktkontakt med RM utan alltid indirekt via något Case-verktyg. Dock ger RM-kunskap en automatiskt djupare förståelse av förutsättningarna för Case-verktygens funktionalitet, de data verktygen kan hantera samt villkoren för att de ska kunna dela "kunskap". Mycket kommer dessutom att skrivas om repositories, inte minst IBMs. Det kommer att bli viktigt att snabbt kunna tolka och sovra i informationsflödet.

Föreliggande rapport ger en översiktlig introduktion till RM och dess olika delar. Kommande rapporter kommer istället att fokusera på en viss aspekt åt gången. Tanken är att varje "ämne" ska kunna avhandlas med ett omfång av ca 5-15 sidor för att presentationen inte ska upplevas för tung. De som inte är intresserade av allt kan därigenom enkelt hoppa över valda delar. Figur 1 nedan kommer att återkomma i varje rapport för att indikera vilken del av "repositoryt" det aktuella temat berör.



FIGUR 1

1. Syfte

Datakataloger tenderar att bli alltmer heltäckande och avancerade. Det är inte längre bara fråga om en termkatalog kanske kompletterad med uppgifter om vilka program som hanterar vilka termer. Blicken har höjts från data i implementerade system till betraktande av information som en vital resurs i en verksamhet, något som påverkar och är delaktigt i det mesta av skeendet. En förståelse av informationens betydelse för en verksamhet kräver en helhetssyn, där data- och funktionsaspekterna i informationsbehandlingen får en uttömmande och integrerad beskrivning, men där denna även kopplas till en integrerad beskrivning av den bedrivna verksamheten.

För att markera ambitionsökningen har man i engelsk vokabulär bytt ut data dictionary mot repository (enligt Webster's: "one that contains or stores something non-material, eg 'the book is a repository of knowledge'"). Det svenska datakatalog har ännu inte fått en etablerad motsvarighet. Översättning enligt förvaringsplats, rikt förråd, skattkammare, guldgruva e dyl känns inte lika välfunnet, varför vi än så länge tenderar låna det engelska begreppet. Denna rapportserie utgör inget undantag.

På sikt kan ett repository alltså förväntas innehålla information om bla

* verksamheten

* verksamhetens informationsbehov

* informationsservice i form av applikationer och data

* hård- och mjukvara

samt alla erforderliga samband mellan dessa aspekter.

IBMs Repository Manager har byggts med syftet att kunna hantera denna mångskiftande information i ett repository. Systemet består av ett stort antal funktioner (services) som är uppbyggda enligt avancerade och mycket välgenomtänkta principer. I denna sin egenskap utgör RM en vital del i förverkligandet av IBMs AD/Cycle-koncept genom att vara "den gemensamma nämnaren" för all däri hanterad information. AD/Cycle kommer att beskrivas i ett separat dokument.

2. Jämfört med konventionella DBMS

RM är byggt för att hantera information om en verksamhet, dess informationsbehov och informationsbehandling. RMs grundläggande uppbyggnad är dock generell såtillvida att den i princip tillåter hantering av vilken typ av information som helst. Vi arbetar inte med order och artiklar i en orderdatabas men väl med funktioner, informationsbehov, begrepp mm i en repositorydatabas. Vad motiverar i så fall RM, som ett tillskott i den redan stora floran databashanterare, exv IBMs DB2?

Några möjliga orsaker:

* Informationsstrukturen i ett repository är höggradigt komplex, en förutsättning som inte gynnar relationsmodellen. En semantiskt rikare och mer problemnära specifikation och anpassat gränssnitt underlättar förståelse, minskar felrisker och tillåter flexiblare ändringsarbete.

  RM baseras på en variant av Entity-Relationshipmodellen.

* I vanliga databasapplikationer hanterar man fleranvändarmiljö genom transaktionsbegreppet. Under en transaktion låses berörda delar av en databas för andra än den aktuella transaktionen. Transaktionen är en sammanhållen operation som i allmänhet har kort varaktighet (registrering eller uppdatering av en order, uppdatering av ett lagersaldo i samband med inleverans etc). Tekniken är komplex, men numer välutvecklad.

  En repository-användare, exv en databaskonstruktör, kan behöva dagar innan en design har nått ett naturligt slut. Transaktionerna har i repository-sammanhang ofta en tidsutdräkt som gör låsning ohållbar. Därtill kommer att den information som hanteras inom en sådan transaktion sannolikt är mer komplex, dvs täcker en större del av databasen. Låsning blir en tung mekanism.

  RM avser istället att lösa detta problem genom versionshantering. Två databasanvändare kan arbeta mot samma typ av information, men var och en har sin version av informationen. Bekymret skjuts upp till senare dvs till tidpunkten för sammanjämkning av alla utestående versioner. Hur detta i praktiken kommer att utföras har inte gått att få klarlagt. Sannolikt är detta fortfarande ett bekymmer även för RM-utvecklarna. Tanken var från början att utnyttja object-begreppet (se avsnitt 9 nedan) som en ändamålsenlig enhet för versionskontroll. Av olika skäl har denna ansats släppts. Enligt artikeln Paul Winsberg "AD/Cycle: Where are we?", InfoDB, vol 5, nr 3, arbetar IBM nu på en versionsindelning ända ner på varje enskild komponent i repositorydatabasen. För närvarande gäller att när behov av ny version

uppstår tas en kopia av hela verksamhetsmodellen. Repositorydatabasen innehåller då två versioner av samma verksamhetsmodell osv till behövligt antal versioner, som var och en utvecklas vidare helt separerat från varandra. Som synes återstår mycket arbete inom denna sektor.

* En vanlig databashanterare hanterar endast data och enligt ett generellt gränssnitt som ska passa så många användningsområden som möjligt. Om man avgränsar användningsområdet kan såväl data- som funktionsaspekten specialanpassas. Systemet blir kraftfullare för sitt ändamål men kanske inte säkert lika generellt.

  RM har visserligen specialanpassats för redovisat ändamål men kan inte sägas ha fått en begränsning i applicerbarhet. Snarare är det så att modelleringsbegreppen har blivit fler och mer avancerade. Exempel på sådana är fyra olika typer av *policies*. *Derivation policies* är algoritmer för att beräkna attributvärden, *integrity policies* reglerar exv tillåtna attributvärden och vilka relationships som ska eller inte ska finnas, *trigger policies* initierar viss aktivitet när vissa operationer utförs mot databasen och *security policies* skyddar mot otillåten åtkomst till data. I viss utsträckning har även ett objektorienterat synsätt, med sin integrerade syn på data och funktion, givits inträde.

3. Referens till IRDS

IRDS är en förkortning för Information Resource Dictionary System. Det förknippas normalt med pågående standardiseringsarbeten inom ANSI och ISO. Ett Information Resource Dictionary (IRD) avses i dessa sammanhang innehålla samma typ av information som vi ovan sagt ingår i ett repository. IRD är ett mer adekvat namn i dessa sammanhang eftersom ett repository i princip kan innehålla vilken typ av information som helst. Normalt avser man dock samma sak. RM kan med andra ord fungera som ett IRDS. Dokumentet "Information Technology - Information Resource Dictionary System (IRDS) Framework", ISO 10027, 1990 har antagits som en standard. Däri ingår formuleringen av de så kallade fyra datanivåerna som en central del. Dessa har tidigare beskrivits och exemplifierats i rapporterna [TRIAD K nr 1 och 2].

Den första eller nedersta nivån, **Application Level**, finns med som en nivå för fullständighets skull. Där lagras förekomster av verksamhetsinformation, exv information om alla artiklar, kunder och order.

Nivå 2 kallas **IRD Level** och innehåller en beskrivning av den information som finns på nivå 1 (dvs schemat i en normal databasapplikation) men även alla andra uppgifter som vi tidigare sagt att ett repository eller IRD kan innehålla. Denna nivå svarar alltså mot innehållet i RMs databas. Se vidare avsnitt 4.

De två nedersta nivåerna innehåller verksamhetsspecifik information.

RMs schema, dvs beskrivningen av vilken typ av information som får hanteras i repositoryt, svarar mot **IRD Schema**, som ingår i nivå 3, **IRD Definition Level**. Nivå 3 kan även innehålla exv gamla eller nya versioner av schemat, ännu inte färdiga beskrivningar, mm. Se vidare avsnitt 5.

Nivå 4 kallas **IRD Definition Schema Level**. Den beskriver vilken typ av information som får hanteras på nivå 3, mao kan ses som ett rent schema för nivå 3. I RM är det helt enkelt den uppsättning modelleringsbegrepp (inklusive deras samband) som står till buds för att beskriva nivå 3. Se vidare avsnitt 6.

IRDS identifierar och strukturerar även ett antal olika gränssnitt varav det viktigaste är det gränssnitt (**IRDS Services Interface**), som reglerar all tänkbar hantering av data på både nivå 2 och nivå 3. RM är att ses som en **Processor** som kan ombesörja alla de typer av operationer, som gränssnittet reglerar. IRDS Framework pekar på möjligheten för en dylik gränssnittsprocessor att ta hjälp av en ordinär databasprocessor för delar av det erforderliga arbetet. RM innefattar denna konstruktion i och med att man nyttjar DB2 för lagring av repository-informationen.

För att konkretisera resonemanget, gör vi följande jämförelse:

Antag att vi av någon anledning valt att använda SQL, istället för RM, för att hantera repository-information. SQL-databasen ligger då på nivå 2. Vilka tabeller (process, infoflow, table ...), tabellkolumner (processname, cardinality, version number, ...), vyer och index som ska finnas, beskrivs i SQL-schemat med hjälp av datadefinitionssatser som CREATE, ALTER, DROP. Därmed etableras den del av nivå 3, som svarar mot IRD Schema. För att kunna skapa nivå 3 på rätt sätt måste man ha förstått betydelsen av SQLs modelleringsbegrepp, nämligen table, column, view, indexes mm samt hur de hänger ihop. Dessa begrepp används nämligen i SQLs datadefinition och/eller datahanteringssatser. Modelleringsbegreppen blir nivå 4. SQL fullgör här rollen som IRDS Services Interface Processor.

4. Repository-data

Med CASE-glasögonen på kan information jämställas med den verksamhetsspecifika information som hanteras i CASE-verktygets databas. Det kan vara datamodeller, flödesmodeller, programstrukturer, funktionsstrukturer, målmodeller etc allt efter Case-verktygets specialiteter. Tanken bakom RM är att inom AD/Cycle-konceptets ram kunna erbjuda ett gemensamt och gemensamt tillgängligt datalager för många olika typer av verktyg och behov. En förutsättning för detta är att man kan tolka informationen i databasen.

Därmed över till nästa avsnitt om RMs schema.

5. Repository-schema

5.1. Inledning

RMs schema sammansätts av tre views:

* Conceptual view, som beskriver vilka typer av information som kan hanteras. (Vad-perspektivet)
* Storage view, som beskriver hur informationen i conceptual view lagras i en DB2-databas. (Hur-perspektivet)
* Logical view, som beskriver den skiftande användningen av informationen i conceptual view för olika behov. (När-perspektivet)

För varje view finns en data- och en funktionsaspekt.

5.2. Conceptual view

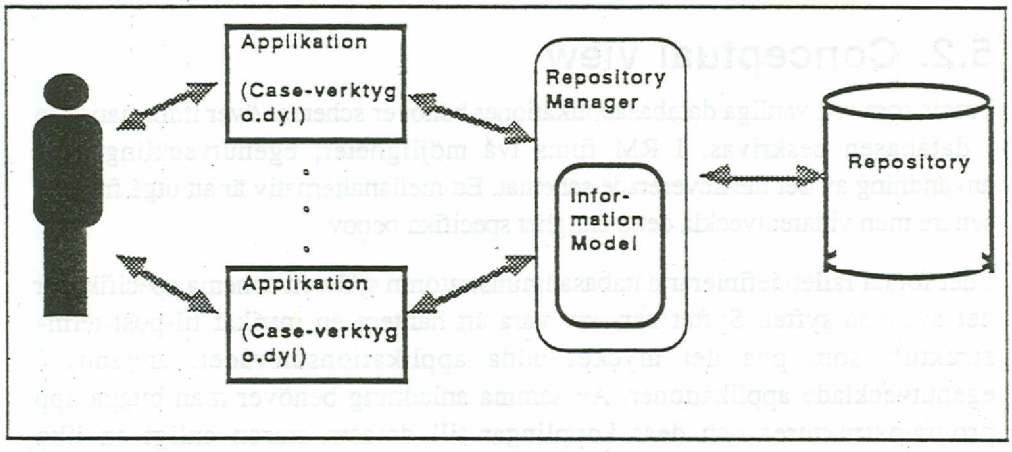
Precis som vid vanliga databasapplikationer behöver schemat över informationen i databasen beskrivas. I RM finns två möjligheter; egenutveckling eller användning av det medlevererade schemat. Ett mellanalternativ är att utgå från det senare men vidareutveckla detta enlighet specifika behov.

I det första fallet definierar databasadministratören själv ett schema specifikt för det avsedda syftet. Syftet kan exv vara att hantera en intrikat fil-post-term-struktur, som pga det mycket udda applikationsområdet, används i egenutvecklade applikationer. Av samma anledning behöver man bygga upp programstrukturer och dess kopplingar till datastrukturen enligt en lika egenutvecklad teknik. Inga Case-verktyg bedöms kunna ge önskad service. Med RM öppnas möjligheten att specificera ett schema efter eget huvud och behov. Med hjälp av egenutvecklat verktyg kan man sedan via schemat operera på data i RMs databas. Man har valt RM framför en vanlig databashanterare bla av de skäl som diskuterats under avsnitt 2 ovan. Ett annat skäl var perspektivet mot framtiden. En marknadsanalys visar att företagets produkter i allt högre grad kommer att behöva integreras med andra produkter och att data kommer att behöva utbytas mellan flera olika produkttyper. Strukturen på data och program kommer att behöva ensas med omvärlden för att vara förståelig, integrerbar och utbytbar. Därmed närmar vi oss det andra alternativet.

AD/Cycle-konceptet bygger på idén om kommunicerbar information mellan många behov och Case-produkter. En förutsättning för detta är att man förstår varandras data, talar samma språk. Ett gemensamt schema är där en avgörande förutsättning. Case-verktygen a och b måste känna sig övertygade om att den företeelse de båda kommunicerar under exv beteckningen *programmodul*

verkligen förstås och uppfattas på samma sätt av båda parter. Tillkommer sedan verktygen c, d och e gäller det i än högre grad att via förhandlingar se till att inga missförstånd uppstår. IBM har, för att underlätta, eller sannolikare, för att göra sådana förhandlingar överflödiga (här bortsett från andra strategiska bedömningar), utvecklat och utvecklar ett fördefinierat schema som följer med leveransen av RM. Schemat kallas **Information Model**. Tanken är att den på sikt ska omfatta i stort sett all typ av information, som kan tänkas behöva kommuniceras mellan verktyg. Information Model kan revideras och utökas men då fjärmar man sig återigen snabbt från grundidén. Variationsrikedom ställs mot enhetlighet. Uttryckskraft ställs mot kommunikationskraft.

De personer som arbetar med verksamhetsspecifika beskrivningar (nivå 2) upplever i normalfallet repositoryt som en databas man kommunicerar med över ett egenskrivet eller inköpt Case-verktyg e dyl. Verktyget befinner sig i rollen som databasapplikation. Hur dessa personer upplever informationen och dess struktur är helt en fråga om vilket användargränssnitt verktyget erbjuder. Observera att detta gränssnitt kan vara något helt annat än verktygets gränssnitt mot repositoryt. Se figur 2.



FIGUR 2

Vem är då intresserad av Information Model? I första hand är modellen ett intresse för verktygsutvecklare, som behöver veta villkoren för kommunikation med repositoryt. Även informationsstrateger, utvärderare och inköpare av verktyg behöver ha klarhet i repositoryts struktur som en referensram i olika typer av bedömningar. Frågor typ

- Vilken information hanterar verktyg X "bakom ytan"?
- Kan verktygen X och Y förmodas kunna kommunicera via repositoryt?
- Inom vilka vitala delar av Information Model saknar vi verktygstäckning?
- Verktyget X börjar bli omodernt. Vilka andra verktyg klarar samma jobb, dvs täcker in åtminstone samma del av modellen?

kommer att bli vanliga.

En förutsättning för svar är givetvis att verktygstillverkare (i den mån verktyget inte är egenutvecklat) släpper kunskap om det egna verktygets koppling till Information Model. Förmodligen kommer marknaden så småningom att kräva detta.

Vi ska i sammanhanget inte heller glömma de användare som känner behov av att vara på "fast mark" i sitt arbete och inte bara i händerna på verktygsgränssnittet. Det är inte osannolikt att arbetet både kan bli mer produktivt och att fel kan undvikas, om denne vet, eller åtminstone kan ana, vilken information som egentligen hanteras mot repositoryt i de olika verktygsfunktionerna.

Information Model är omfattande och kommer därför att beskrivas mer utförligt i separata rapporter, bla [Triad K nr 6].

5.3. Storage view

För hanteringen av data använder sig RM av IBMs relationsdatabashanterare DB2. Schemat är ju formulerat genom conceptual view, se ovan. Som redan nämnts formuleras conceptual view i en Entity-relationship-orienterad begreppsapparat (se vidare avsnitt 6, nedan). För att schemat ska kunna hanteras av DB2 måste det formuleras om i en struktur och enligt en teminologi som DB2 förstår, dvs enligt relationsmodellen. RM gör detta automatiskt, om man så önskar, inklusive prestandabefrämjande indexes och kod. Önskar databasadministratören själv påverka lagringsutformningen, finns vissa möjligheter till detta. Storage view är den prestandaanpassade DB2-versionen av conceptual view. En conceptual view har alltid bara relatering till en enda aktiv storage view.

5.4. Logical view

Operationer mot repositoryt specificeras och grupperas i **repository functions**, se avsnitt 8, nedan. En repository function arbetar normalt bara mot en begränsad del av conceptual view, benämnt **logical view**. Olika repository functions kan referera till samma logical view. En logical view är uppbyggd av ett antal så kallade **templates**, där varje template i princip specificerar vilka av en entity type´s attribut (ur conceptual view), som är av intresse. Templates kan bilda hierarkier i form av **template trees**. Den till synes rätt begränsade och enkla strukturen hos ett template är sannolikt en följd av att de grundläggande operationerna på repositorydata uttrycks med referens till templates. Avbildningen till det använda programmeringsspråkets interna datastruktur sker också med referens till templates. Med hjälp av template trees kan mer komplexa strukturer byggas upp.

Logical view och dess koppling till repository function kommer att beskrivas mer utförligt i en separat rapport.

6. Datamodelleringsspråk

När vi skapar en vanlig databastillämpning med hjälp av SQL, uttrycker vi vårt schema med hjälp av SQLs modelleringsbegrepp. Till de viktigaste begreppen hör *table* och *column* samt vetskapen om att en column ingår som en del i table. Dessa begrepp finns exv i syntaxen för DDL-satsen (här i förenklat skick och med begreppen understrukna):

CREATE TABLE <table name> (<column name, ...)

En förenklad syntax för DML-satsen SELECT ser på motsvarande sätt ut enligt:

SELECT <column name>, ... FROM <table name>

RM bygger inte på relationsmodellen utan hämtar sina begrepp från Entity-Relationship-sfären. De grundläggande begreppen är

Entity type

Attribute type

Relationship type.

Begreppen kan sägas utgöra RM's datamodelleringsspråk eller begreppsapparat. Både de som definierar ett RM schema och de som formulerar operationer på en RM databas, har en syntax baserad på dessa begrepp. I första hand är det verktygsutvecklare som kommer i kontakt dessa språk, men även användare av databas-information kommer indirekt att på olika sätt bli berörda av terminologin.

Vi har därför funnit det motiverat med en utförligare presentation i en separat rapport [Triad K nr 5].

7. Gränssnitt

Ett gränssnitt består normalt av en datadefinitionsdel (Data Definition Language eller DDL) och en datahanteringsdel (Data Manipulation Language eller DML). Det första används för att etablera och allmänt hantera schemat medan det andra uttrycker operationer på databasen i termer av schemat. Även begreppen Specification Service respektive Run-time Service används.

RMs DDL har vi ännu inte någon dokumentation om.

DML beskrivs under den del av CPI (Common Programming Interface) som går under beteckningen Repository Reference. DML-satserna formuleras inom en *repository function*. De exekveras när funktionen anropas. Satserna är av typen

"addera en entitetsförekomst inklusive attributvärden"

"sök första/nästa/sista entitetsförekomst av viss typ".

Det finns även ett flertal mer sammansatta operationer som går under benämningen *built-in functions*.

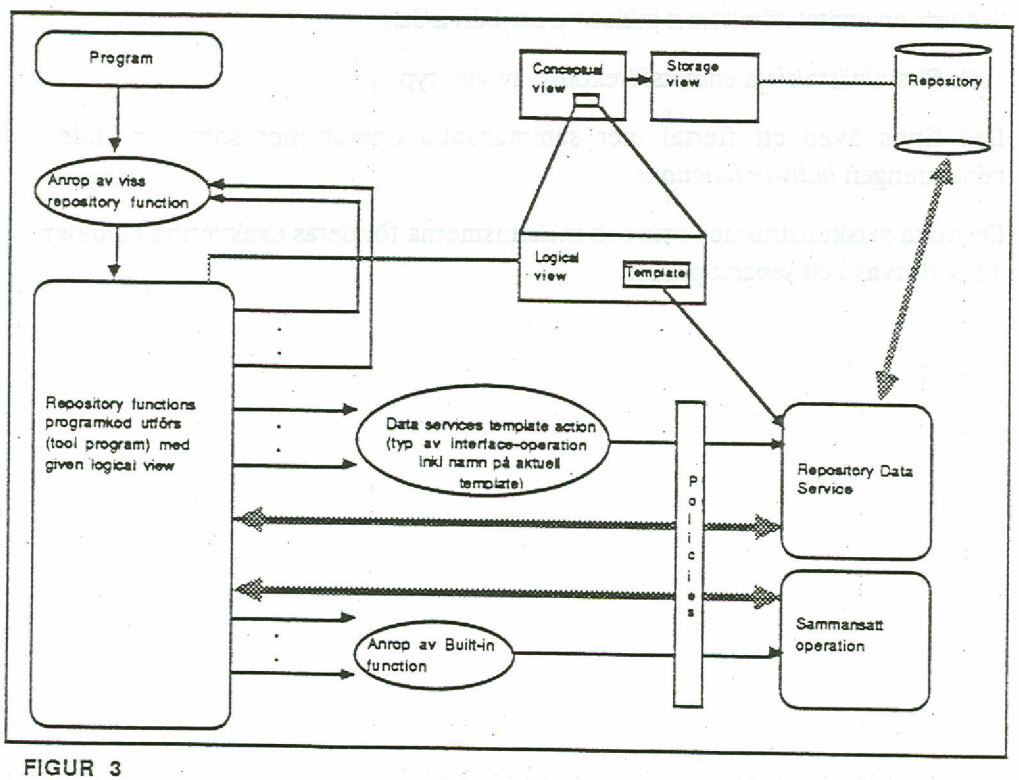
De olika satskonstruktionerna och mekanismerna för deras exekvering kommer att beskrivas i ett separat papper.

8. Tool - Function

Uppbyggnaden av funktionalitet mot ett repository är grovt som följer.

Informationen i repositoryt hanteras genom ett *tool*. Ett tool är sammansatt av en eller flera *repository functions*. En repository function associeras till en *logical view* (se ovan), en uppsättning *policies* (se avsnitt 2) och bearbetningslogik i form av programkod (*tool program*). I programkoden finns bla anrop till gränssnittsoperationer (se ovan) via *template actions* och även anrop av andra repository functions. Figur 3 ger en sammanfattande bild av hur de olika delarna samverkar.

En utförligare beskrivning kommer i separat rapport. Den rapporten är även den naturliga platsen för genomgång av uppbyggnaden av logical view .



FIGUR 3

9. Övrigt

I RM finns även begreppet *object*. Vi har ännu inte full klarhet i dess syfte och egenskaper. Helt kort kan sägas att begreppet i viss mån svarar mot objektbegreppet inom objektorienteringsområdet. Där finns exv arvshierarkier. Inkapslat inom ett object är ett antal methods där varje method svarar mot en repository function. Inkapslat är även en uppsättning data i generell bemärkelse, alltså inte bara repository-data. I en IBM-rapport anger man bla syftet vara att åstadkomma en bas för versionshantering av sammansatta data. Vi återkommer med fylligare information.

10. Sammanfattning

Avsikten med föreliggande rapport har varit att ge en översikt över ett antal centrala komponenter och egenskaper hos IBMs Repository Manager. Beskrivningen har gjorts medvetet kortfattad för att hålla nere volymen. Där vi tror det finns behov av utförligare presentation i separata rapporter, har detta noterats. Arbete pågår.

Den lämnade informationen har inhämtats från manualer och tidskriftsartiklar. Risken för missuppfattningar är uppenbar. Även källartiklarna kan innehålla fel. Informationen är dessutom under nuvarande turbulenta läge att betrakta som "färskvara". Vår strävan är att ge korrekt information men vi ber om överseende när felaktigheter smugit sig in i texten. När fel konstateras kommer vi att reda ut detta i inledningen till en följande rapport.

TRIAD utvecklar IA

Televerket har just tagit första steget in i sin nya IA-organisation och Posten håller på att bygga upp sin nya DA-organisation. Båda organisationerna har sett nyttan att inför 90-talet gå vidare tillsammans i TRIAD-projektet som drivs tillsammans med SISU. Statskontoret deltar också i projektet för att på sikt kunna föra ut nya synsätt och hjälpmedel inom den civila statliga sektorn.

Ericsson Data Services deltar med tyngdpunkten i den del som handlar om att utveckla kompetenta modelleringsledare, delprojektet "Avancerad utbildning för modelleringsledare".

Modelleringsmetoder är centrala i bedrivandet av verksamheten inom informationsadministrationen. Därför arbetar ett delprojekt med utvecklandet av "nästa generation modelleringsmetod" som skall sättas i händerna på informationsadministratören. Siktet är att fördjupa och bredda dagens modelleringsmetoder och där hämta in kunskap från pågående forskning och utveckling internationellt. (faktaruta om IAS91).

Som stöd för informationsadministrationen behövs verktyg. Inom TRIAD arbetar man där inom två områden, kataloger och verktyg.

Delprojektet kataloger arbetar dels med att utforma den informationsmodell som måste kunna täckas av en katalog, dels med att granska och följa utvecklingen av produkter inom området t ex IBM:s "Repository" och Digitals "CDD". Dessutom följer man standardiseringen internationellt kring IRDS. För parterna i projektet liksom för andra organisationer är detta ett tungt område både vad gäller kommande investeringar ekonomiskt och vad gäller kompetenta resurser för en kommande övergång till "repository-världen". - Det inledande skedet syftar till att bygga upp en kunskapsplattform, som sedan kommer att kunna utnyttjas för kravställande och planering och genomförande av övergång från dagens kataloghantering till morgondagens.

Den andra verktygshanterande delen inom TRIAD-projektet, delprojektet "verktyg för informationsadministration", syftar till att ta fram verktyg för uttag och dokumentering av modeller. Betoningen ligger på människa datorgränssnitt och i första skedet görs utveckling av HYBRIS-gränssnittet med prototyper för Posten och för Televerket.

För att hålla ett helhetsperspektiv på projektets delar och för att ha inpassningen av funktionen Informationsadministration i organisationens övriga verksamhet arbetar delprojektet "Krav på IA". I delprojektet arbetar man dels med att kartlägga dagens krav på dataadministration och projicera till morgondagens krav på IA. Dessutom skall man skapa en bild av IA-verksamhetens innehåll och organisation. Från detta i sin tur ställer man krav på övriga delprojekt. Vilka krav skall ställas på kompetens, metoder, hjälpmedel typ kataloger och gränssnitt?

TRIAD projektet är stort

Budgeten för TRIAD-projektet löper på 10 MSEK per år under en treårsperiod som startar vid kalenderåret 1991 års början och som alltså beräknas avslutad vid utgången av 1993.

TRIAD-projektet är ett tillämpningsprojekt

Det innebär att parterna, Televerket, Posten, Statskontoret, EDS och SISU går in med såväl persontidssatsningar som ekonomiska och att STU, Styrelsen för Teknisk Utveckling, bidrar med ett ekonomiskt tillskott som svarar mot ungefär 40 % av den insatta persontiden.

Öppet för fler deltagare

Parterna i TRIAD-projektet vill gärna öka tempot och bredda perspektivet och vill därför gärna ha fler parter in i projektet. Dessa parter får då enligt SISU:s tårtprincip "betala för en tårtbit, men ät hela tårtan", tillgång till projektets resultat med en insats som ger stor "price performance".

Nya deltagare kan gå in i hela projektet eller i det eller de delprojekt som verkar intressantast. En förutsättning är att man framförallt är beredd att satsa kompetent personal. För de flesta intressenter bord detta vara ett utmärkt sätt att driva personalutveckling för personer t ex inom DA-området, samtidigt som man bygger upp beredskapen inför 90-talets IA-verksamhet.

Kompetensutveckling viktigt resultat

En viktig effekt för parterna av deras medverkan i TRIAD är kompetensutveckling. Man satsar på att ta in personer som så småningom eller redan idag arbetar med DA och IA för att ge dem en djup och "frontlinje"-mässig kompetens. Detta skall utnyttjas när man successivt för in resultaten i den egna organisationen. Projektdeltagarna har alltså en viktig roll som kunskapsförmedlare i den egna organisationen. Dessutom ger projektarbetet deltagarna tillfälle till en egen utveckling inom det professionella området som är unik.

Informationsspridning

Det sjätte delprojektet "Informationsspridning" har till uppgift att sörja för att i första hand parterna men också SISU:s övriga intressenter successivt kan följa och tillgodogöra sig resultat från TRIADprojektet. Seminarier, rapporter och referensgruppsverksamhet är led i den verksamheten.