TRIAD Newsletter K 26

Introduktion till

OpenODB

Stig Berild

TRIAD-PROJEKTET DRIVS GEMENSAMT AV TELIA, POSTEN, STATSKONTORET, ERICSSON OCH SISU, SVENSKA INSTITUTET FÖR SYSTEMUTVECKLING. FÖR VIDARE INFORMATION KONTAKTA SISU, ELECTRUM 212, 164 40 KISTA, TEL 08-752 16 00, FAX 08-752 68 00.

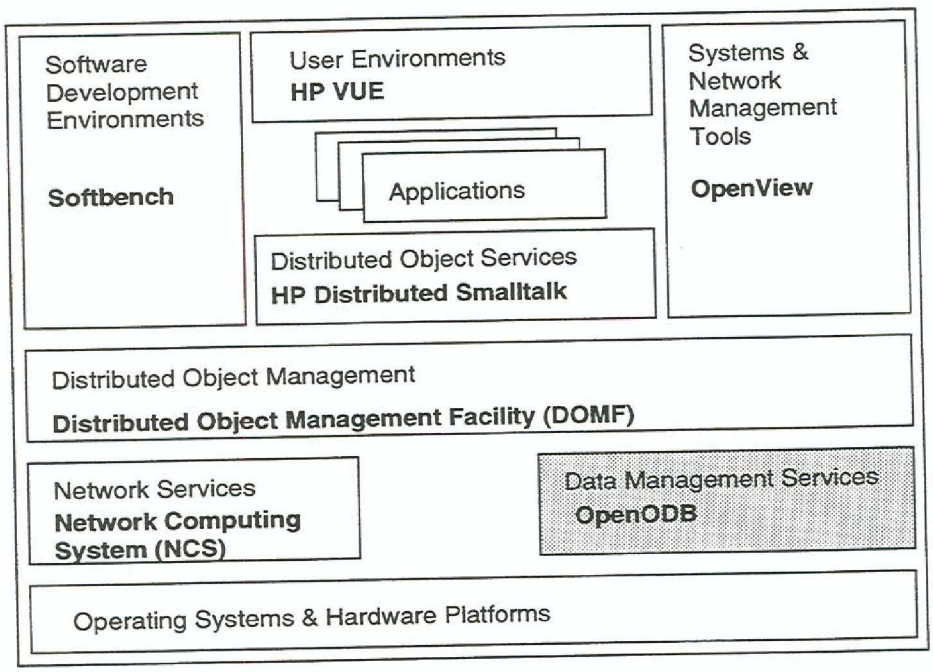
Rapporterna beställs från:
SISU, Electrum 212, 164 40 Kista, Fax 08-752 68 00.
Rapporterna är endast tillgängliga för Triad-parterna och är avgiftsfria.

Introduktion till OpenODB,
en ODBMS från Hewlett-Packard

Innehåll

1. Inledning

OpenODB är en komponent i Hewlett-Packards öppna objektorienterade arkitektur under beteckningen DOCP (Distributed Object Computing Program). Se figur 1 nedan. HP-produkter inom de olika rutorna anges i fet stil. Det kan vara värt att notera att vissa av dessa produkter sedermera fått stå modell för standardiseringsaktiviteter. Exempelvis är OMGs CORBA-specifikation baserad på DOMF, OSF's DCE på NCS och OSF's DME på Openview. HP var för övrigt en av grundarna av OMG.



Figur 1

OpenODB lanserades 1991 som en produkt framtagen ur forskningsprototypen IRIS.

Det är en så kallad objektorienterad databashanterare, men med egenskaper som skiljer den från "vanliga" objektorienterade databashanterare. Den tillämpar inte en metamodell från ett objektorienterat programmeringsspråk typ C++, än mindre har den en direkt integrering med ett sådant. OpenODB ligger snarare betydligt närmare en RDBMS, men med ett mer objektorienterat gränssnitt. Man har definierat ett eget gränssnittsspråk. Språket baseras på SQL-syntax men med anpassningar till en objektorienterad metamodell. Därav beteckningen OSQL.

Man ser sina konkurrenter snarare från RDBMS- än från OODBMS-hållet. Inriktningen är mot större kommersiella tillämpningar med en höggradig fleranvändarprofil (upp till ca 500 samtidiga användare). Man ser i huvudsak korta transaktioner och behov av en konventionell databastillämpnings alla stödfunktioner (exempelvis backup, recovery, behörighet, ...), något som inte alltid dagens OODBMS svarar upp mot. På sikt (allteftersom OODBMS blir mer funktionellt heltäckande) kommer denna skiljelinje antagligen att suddas ut. Transformation av existerande relationsdatabas-tillämpningar till OpenODB borde rimligtvis vara betydligt enklare än till en renodlad OO-modell. HP väljer själva att kalla OpenODB för en hybrid-lösning snarare än en objektorienterad sådan.

2. OpenODBs objektmodell

Metamodellnivån visas i figur 2 uttryckt i BM-notation. Den är en egenutvecklad objektmodell utan speciell likhet med vare sig renodlade objektmodeller eller mer konventionella ER-liknande modeller, i alla händelser inte vid första påseende. Bortser man från skillnaden i benämningar på modellens olika typer av företeelser framträder dock ett antal likheter. Det som i andra modellansatser kallas Object type eller Entity type kallas här enkelt nog för **Type**. En förekomst av en Type kallas dock Object. I den fortsatta texten väljer vi för enkelhetens skull att kalla ett Object för objekt.

Företeelser som står för någon typ av värde och som representeras i form av någon slags symbol (lexikala objekt) modelleras som **Domain**. I de flesta modellansatser brukar man skilja på attributtyp och sambandstyp (attribute type och relationship type) beroende på om det relaterade är en Domain eller en Type. I OpenODB går båda varianterna under den generella beteckningen **Function.** Namnet på en funktion behöver bara vara unik för den Type funktionen tillhör.

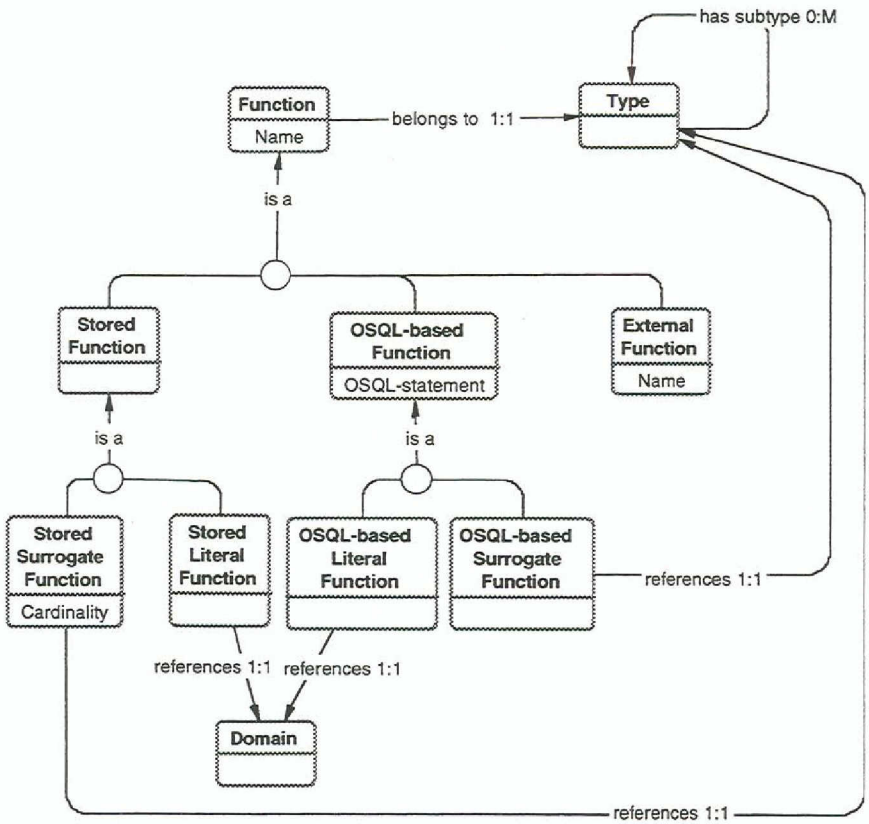
Function specialiseras på första subnivå i

- Stored Function, vars innebörd representeras och lagras i form av ett namn
- OSQL-based Function, som härleds ur andra functions och villkor genom ett uttryck formulerat i OSQL, d v s vanliga härledningar
- External Function, som är en mer eller mindre komplex härledning formulerad
  - i form av SQL-sats mot den aktuella underliggande databasen
  - i form av SQL-satser mot någon annan databas under egal databashanterare
  - operativsystemkommando
  - rutinanrop till någon till applikationen länkad rutin

För både Stored Functions och OSQL-based Functions gäller att de kan referera till (eller mer formellt "vara definierade över") antingen värden ur en Domain eller objekt ur en Type. Det förstnämnda alternativet, d v s en värdefunktion, kallas **literal function,** medan det andra alternativet, d v s en objektfunktion kallas **surrogate function**. Att beteckningen "surrogate" används istället för exempelvis "object" beror antagligen på att objekt i databasen representeras av interna identifierare. Dessa går ibland i litteraturen (speciellt under 1970-talet och första halvan av 1980-talet) under beteckningen "surrogate".

Som synes av metamodellen är Functions binära. Specialiseringsstrukturer kan definieras (*has subtype*). OpenODB innehåller arvsmekanismer som opererar i enlighet med dessa strukturer. Multipla arv kan hanteras.

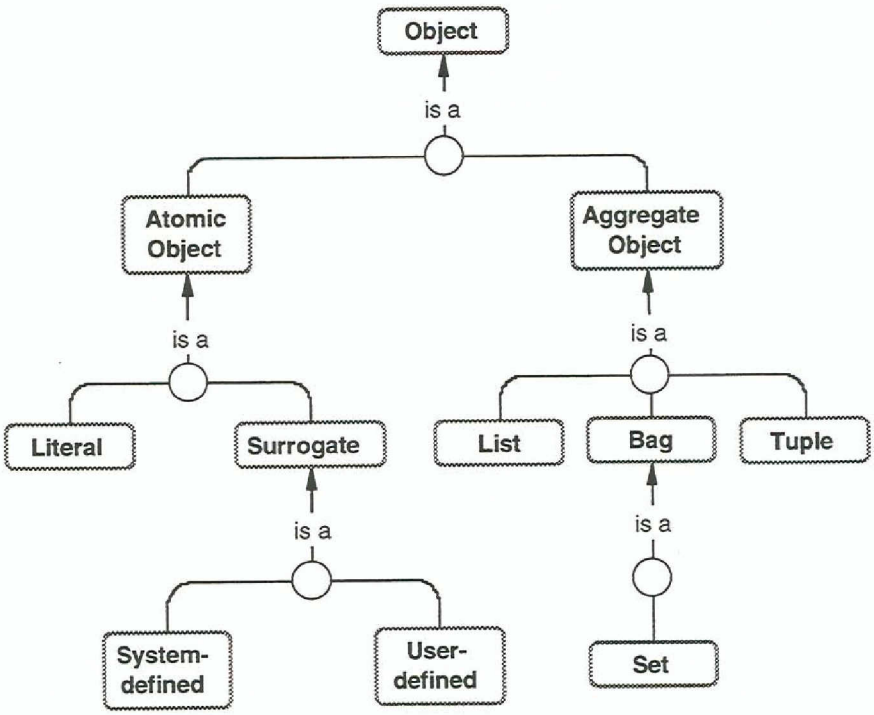
OpenODB klarar också hantering av polymorfism, d v s avgör vilken Function som ska exekveras beroende på typ för aktuellt objekt. Principen för sen bindning används.



Figur 2

Intressant att notera här är att ett objekt kan tillhöra fler än en Type samtidigt. Type kan dessutom ändras efter önskemål. Vilka för- och nackdelar denna frihetsgrad för med sig behöver antagligen en egen utredning. Vad händer exempelvis vid byte? Vilka villkor måste vara uppfyllda för att byte ska kunna ske? På vilket sätt blir objektet "introducerat" bland den nya typens funktioner? I andra modelleringsansatser arbetar man helt efter roll-principen och uttrycker gemenskap mellan rollerna (exempelvis i form av samma underliggande fysiska objekt) i form av villkor på sambandstyper och i specialiseringshierarkier.

En fördefinierad superhierarki har upprättats för objekt-nivån. Den visas i figur 3.



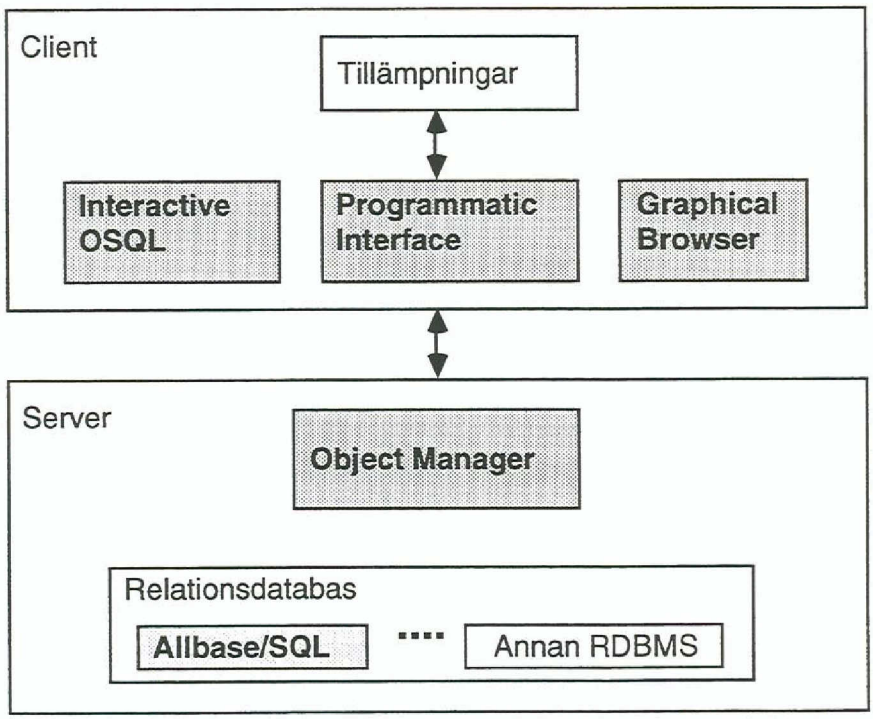
Figur 3

Bag och Set är oordnade mängder av objekt där Bag får innehålla duplikat men inte Set. List är en ordnad mängd objekt som kan innehålla duplikat. Tuple är också ordnad och jämfört med List begränsad till en fast uppsättning objekt.

I OpenODB hanteras schemadata och databasdata på samma sätt, d v s enligt samma grundmodell. Detta gör att OSQL kan användas för operationer både på data och schemadata, något som erbjuder intressanta möjligheter till komplexa frågor. Även hantering av problematiken med *referential integrity* underlättas. Dynamisk utökning av schemat under exekvering kan i princip hanteras, vilket också tillåts.

3. OpenODBs uppbyggnad och egenskaper

OpenODB antas operera i en client/server-miljö. På server-sidan finns den objektorienterade databasen hanterad av *Object Manager*, på client-sidan de tillämpningar som behöver kunna hantera databasens data. Client-sidan formulerar sina operationer mot Object Manager i enlighet med ett till metamodellen anpassat gränssnittsspråk, *OSQL*.



Figur 4

OSQLs likhet med SQL består i snarlik syntaktisk uppbyggnad och så långt möjligt överensstämmande hjälpord. OSQL innehåller även vissa satskonstruktioner som återfinns i normala programmeringsspråk (exempelvis while-, for each-iteraktioner, villkorssatser och sekvenseringar). OSQL-satser kan formuleras och exekveras interaktivt (*interactive OSQL*). OSQL-satser kan

även integreras i C-kod som ett *programmatic interface*, vilket gör att det kan användas i anslutning till varje språk som kan länkas med C. OSQL innehåller till sist en funktion för grafisk presentation av databasens struktur benämnt *graphical browser*. Browsern kan också, via schemat, användas för sökningar mot databasen. Som synes är OSQL client-sidans språk för att formulera önskade operationer mot databasen. Tidigare nämnda external operations är anropsbara från OSQL.

HP arbetar för att få OSQL accepterat för standardisering inom ANSI.

4. OSQL-exempel

Det finns inte utrymme att gå in på syntaxen för OSQL. Några exempel på konstruktioner, nedan, ger förhoppningsvis en viss uppfattning om språkets uppbyggnad. Schema-definitioner ("DDL") visas i figur 5.

```
CREATE TYPE Employee
      FUNCTIONS    (Name CHAR,
                    Salary FLOAT,
                    Picture LARGE);
```

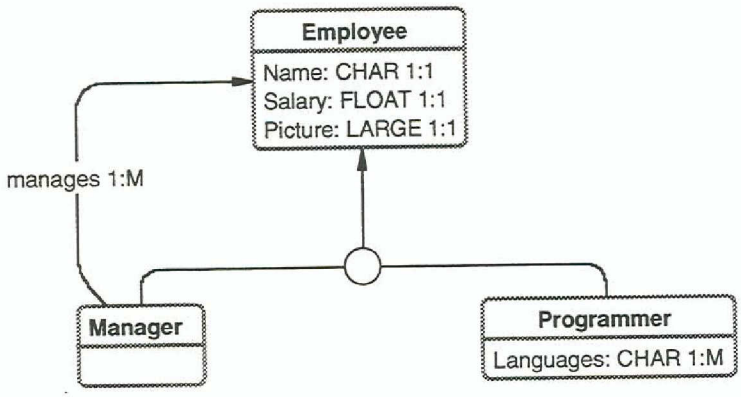
```
CREATE TYPE Programmer SUBTYPE OF
 Employee
      FUNCTIONS (Language SETTYPE (CHAR));
```

```
CREATE TYPE Manager SUBTYPE OF
 Employee
      FUNCTIONS (Manages SETTYPE (Employee));
```

```
CREATE FUNCTION WorksFor
(Employee e) ->Manager AS OSQL
      SELECT mgr
      FOR EACH Manager mgr
      WHERE e IN Manages (mgr)
```

Figur 5

Motsvarigheten (utan härledningen WorksFor), uttryckt i BM-notation visas i figur 6.



Figur 6

Exempel på data-operationer ("DML"), givet schemat i figur 5, visas i figur 7.

```
CREATE OBJECT AS Programmer
    FUNCTIONS (Name, Salary, Languages)
    :bob ('Bob Cox', 55000, SET ('PL/1', 'C')),
    :sue ('Sue Smith', 65000, SET ('Cobol'));

CREATE OBJECT AS Manager
    FUNCTIONS (Name, Salary, Manages)
    :al ('Al Ott', 70000, SET (:bob,:sue)),
    :jim ('Jim Hill', 100000, SET ());

SELECT Salary (:bob);                          55000

SELECT Name (Manages (:al));                   'Bob Cox'
                                               'Sue Smith'

SELECT Name (m)                                'Al Ott'
    FOR EACH Manager m;                        'Jim Hill'
```

Figur 7

5. Övrigt

Object Manager använder i dagsläget en relationsdatabashanterare i botten. Från början var det HPs egen Allbase/SQL men numer finns även andra alternativ, exempelvis OnLine från Informix. Databashanterarens kapacitet vad gäller distribution av data gäller indirekt även OpenODB. Samma sak gäller behörighet. HP och Informix samarbetar på flera plan runt OpenODB, vilket ju är naturligt eftersom HP är delägare i Informix.

Med risk för snabbt inaktuella uppgifter: OpenODB körs under UNIX på HP9000-plattform. En grundlicens kostar i dagsläget ca $100.000.- vilket inkluderar faciliteter för 1-8 användare, utbildning och viss konsulterande rådgivning.

TIDIGARE UTGIVNA PUBLIKATIONER AV TRIADGRUPPEN

Verksamhetskrav på informationsadministration

V 1: IA och verksamhetens krav – erfarenheter från offentlig förvaltning
V 2: Fallstudie av IA-projektet vidTeleverket
V 3: IA-erfarenheter från företag och myndigheter
V 4: Den gemensamma informationsmarknaden – en referensram för handlingsfrihet och konkurrenskraft
V 5: ...fråga är guld. Lokal affärsstyrning utifrån den egna verksamhetens data.

Modellering

N 1: Modelleringsansatser för begrepps- och datamodellering – Beskrivning och försök till jämförelse
N 2: Generering av konceptuella modeller från policydokument
N 3: Espritprojektet Tempora
N 4: Prövning av regelbaserad metodik inom Posten
N 5: En kokbok i remodellering – utkast
N 6: Datorstöd för modellintegration
N 7: Modellbaserad kunskapsinsamling
N 8: Modellkvalitet
N 9: Samband mellan dokument och modeller
N 10: Modelleringshandboken
1 – Översikt
2 – Modelleringsledarens bashandledning
3 – Modellering i grupp
4 – Kommunikation
5 – Arbetsgångar
6 – Modelleringsväskan
7 – Objektorienterad verksamhetsanalys
8 – Basmodeller
9 – Regelmodellering i praktiken
10 – Business Process Reengineering
11 – Namnsättning
12 – Tolkning av grafiska modeller
N 11: Ett+Ett=Ett – Två praktikers erfarenheter av modellintegrering

Kunskapsförmedling

H 1: Handledarutbildning för modelleringsledare, avancerad
H 2: Slutrapport HUMLA prototyp
H 3: Utbildning i Informationsadministration
H 4: Spridning av Hybris – en fallstudie vid Telia

Uttagssystem

U 1: Hybris i Unix-miljö
U 2: DEBRIS
U 3: Hybris DOS/PimWin på Posten
U 4: Program för sökning i databaser – en marknadsöversikt
U 5: Att nå och förstå data – möjligheter och begränsningar

Katalogprinciper

K 1: IRDS
K 2: IRDS Modeller och modellnivåer
K 3: Koppling begreppsmodell – relationsmodell
K 4: IBM:s Repository Manager – en Introduktion
K 5: IBM:s Repository Manager: Datamodelleringsbegreppen
K 6: IBM:s Repository Manager: Begreppsmodellering i Information Model
K 7: IBM Repository Manager: Attribut- och värdemodellering i Enterprise Submodel
K 8: Navigering i Repository
K 9: TRIAD Newsletter – IRDS inom ISO. Dagsläget
K 10: TRIAD Newsletter – ISO/IRDS. Händelseutvecklingen 91/92
K 11: Samverkan mellan resurskataloger – visioner eller behov
K 12: AD/Cycle I Information Model – Processer och och informationsflöden mellan processer
K 13: AD/Cycle I Information Model – Info Flows inom Processmodellen
K 14: AD/Cycle I Information Model – Relationsdatabasmodellering
K 15: AD/Cycle I Information Model – Härledningsspecifikationer i begreppsmodellen
K 16: IA-prototyp
K 17: Repository AD/Cycle – International Users Group
K 18: RAD-konferensen i Chicago, 1992
K 19: Vad händer inom ANSI-IRDS?
K 20: Information Warehouse – vad är det?
K 21: CDIF – en översikt
K 22: PCTE – en översikt
K 23: XLII – en öppen och flexibel utvecklingsmiljö
K 24: Hybris IA/DA – En IA-prototyp vid Telia
K 25: Introduktion till GDMO-standarderna

KORT OM TRIAD

Triad är namnet på ett treårigt samarbetsprojekt kring informations-administration och dataadministration, IA/DA, som Telia, Posten, Ericsson, Statskontoret och SISU bedriver. Syftet är att utveckla parternas synsätt, metoder och hjälpmedel inom detta område. Arbetet inom Triad är uppdelat i delprojekt som är sammanförda i tre block.

Beställarblocket vänder sig dels till dem som är verksamhetsansvariga och måste ta ställning till IA-/DA-satsningar, dels till dem som har ansvaret för IA/DA inom en organisation. Delprojekten inom detta block arbetar med att formulera verksamhetens krav på IA/DA samt studerar och beskriver roller, organisation och arbetsformer för IA-/DA-arbete.

Utförarblocket vänder sig till dem som arbetar med IA/DA. Delprojekten arbetar med modellering, data- och resurskataloger samt uttagssystem.

Kunskapsförmedling är det block som ser till att resultaten kommer Triad-parterna till godo. Detta sker bland annat genom kurser, seminarier samt genom att rapporter som denna ges ut.